O‘ZBEKISTON RESPUBLIKASI
OLIY TA’LIM, FAN VA INNOVATSIYALAR VAZIRLIGI

SHAHRISABZ DAVLAT PEDAGOGIKA INSTITUTI

“INFORMATIKA VA UNI O‘QITISH METODIKASI” KAFEDRASI

M.O‘.O‘KTAMOV

DASTURLASH TILLARI II

fanidan

(O‘QUV QO‘LLANMA)

«GRAND KONDOR PRINT»
TOSHKENT – 2024

UDK: 484.357:29.17:73
KBK: 58.547.29
O` 458

M.O‘.O‘KTAMOV.
DASTURLASH TILLARI II fanidan (O‘QUV QO‘LLANMA). — Toshkent.: “GRAND KONDOR PRINT”, 2024 y., 188 b.

Muallif:
M.O‘.O‘ktamov - Shahrisabz davlat pedagogika instituti “Informatika va uni o‘qitish metodikasi” kafedrasi o‘qituvchisi.

XXI asrda jadal rivojlanish va turli jarayonlarni avtomatlashtirish hamda robotlashtirish davrida dasturlash tillarini bilish va uni o‘z ish jarayonida ishlata olish texnik va pedagogik ta’lim yo‘nalishda tahsil olayotgan o‘quvchi-talabalar uchun juda muhim deb hisoblanadi. Bu zamonaviy mutaxassislar uchun eng zaruriy talablardan biridir.

Python dasturlash tili samarador yuqori darajadagi ma’lumotlar tuzilmasini hamda oddiy, ammo samarador bo‘lgan obyektga yo‘naltirilgan dasturlash uslublarini taqdim etadi. Undan tashqari, python dasturlash tili o‘rganish uchun oson va shu bilan birga imkoniyatlari juda yuqori bo‘lgan kam sonli dasturlash tillari sirasiga kiradi va shu bilan birgalikda pythonda dasturlash jarayoni juda ham oddiy va qulay tarzda amalga oshiriladi.

Ana shundan kelib chiqib, M.O‘.O‘ktamov tomonidan “Dasturlash tillari II” nomli o‘quv qo‘llanma tayyorlandi. Ushbu o‘quv qo‘llanma 60110600- Matematika va informatika ta’lim yo‘nalishlari uchun “Dasturlash tillari II” fanining o‘quv dasturi va ishchi dastur asosida tayyorlangan. Ushbu o‘quv qo‘llanmada Python dasturlash tiliga kirish, dasturlashning asosiy tushunchalari, dasturlash tillarining tuzilmasi, tarmoqlanish va uzilishlarni tashkil etish operatorlari, takrorlanish operatorlari, funksiyalar, ro‘yxatlar va kortejlar, lug‘at va to‘plamlar, satrlar va ulardan foydalanish, fayllar va fayllar bilan ishlash, grafiklar chizish va ularni qayta ishlash, diagrammalar va uch o‘lchovli grafiklar mavzulari batafsil yoritilgan.

O‘quv qo‘llanmada Python dasturlash tilining barcha imkoniyatlari bo‘yicha nazariy tushunchalar hamda bu tushunchalarni o‘zlashtirish uchun masalalar yechimlari keltirib o‘tilgan. Har bir mavzu bo‘yicha mavzuni mustahkamlash uchun nazariy savollar hamda mustaqil ishlash uchun topshiriqlar ham keltirib o‘tilgan. O‘quv qo‘llanma asosan talabalar va mustaqil o‘rganuvchilar uchun qulay vosita hisoblanadi, chunki har bir mavzu uchun nazariy tushunchalar hamda misollar yechimi keng ravishda yoritib o‘tilgan.

Taqrizchilar:
J.H.Hamrayev- Toshkent kimyo-texnologiya instituti shahrisabz filiali “Tabiiy va aniq fanlar” kafedrasi p.f.f.d, PhD
F.E.Qodirov- Shahrisabz davlat pedagogika instituti “Informatika va uni o‘qitish metodikasi” kafedrasi mudiri, i.f.f.d PhD.

ISBN 978-9910-9763-4-6

MUNDARIJA

Kirish……..……..... 4

1-mavzu. Algoritm, algoritmning berilish usullari, xossalari va turlari ….....7

2-mavzu. Python dasturlash tili va uning imkoniyatlari…….....19

3-mavzu. Python dasturlash tilida interaktiv rejim, birinchi dastur, kiritish va chiqarish operatorlari…….....…….....30

4-mavzu. Python dasturlash tili tarkibidagi arifmetik amallar va mantiqiy amallar…….….....36

5-mavzu. Python dasturlash tili tarkibidagi matematik funksiyalar va ifodalar…….....….49

6-mavzu. Python dasturlash tilida chiziqli, tarmoqlanuvchi, takrorlanuvchi jarayonlarni dasturlash……..…….....56

7-mavzu. Sikl qadamlarini tashlab o‘tish va sikllarni muddatidan oldin tugatish……...….....….77

8-mavzu. Python dasturlash tilida shartli takrorlanuvchi jarayonlarni dasturlash……..…….....81

9-mavzu. Python dasturlash tilida funsiyalarni yaratish va ulardan foydalanish……..…….....…….86

10-mavzu. Python dasturlash tilida ko‘p qiymat qaytaruvchi funksiyalar va ulardan foydalanish…….....…… 99

11-mavzu. Python dasturlash tilida ro‘yxat va kortejlar…….....102

12-mavzu. Python dasturlash tilida lug‘at va to‘plamlardan foydalanish.....117

13-mavzu. Python dasturlash tilida massivlar va ulardan foydalanish….....133

14-mavzu. Python dasturlash tilida satrlar va ulardan foydalanish…….....….146

15-mavzu. Python dasturlash tilida fayllar va ulardan foydalanish….....……158

16- mavzu. Python dasturlash tili tarkibida grafiklar chizish va ularni qayta ishlash…….....……167

17-mavzu. Python dasturlash tili tarkibida diagrammlar va uch o‘lchovli grafiklar chizish…….....176

Foydalanilgan adabiyotlar ro‘yxati…….....….185

Kirish

Hozirgi vaqtda dasturlashga bo‘lgan qiziqish keskin ortib bormoqda. Bu holat kundalik ehtiyojdagi axborot kommunikatsion texnologiyalarning jadallik bilan rivojlanishi sababli yuzaga kelmoqda. Dasturlashni o‘rganish uchun aniq masalalarni yechuvchi dasturlar yozish kerak. Buning uchun algoritmlar tuzishni, dasturlash tillarini va muhitlarini o‘rganish talab etiladi. O‘zbekiston Respublikasi Prezidentining 2017 yil 7-fevraldagi PF-4947-son Farmoni bilan tasdiqlangan 2017-2021 yillarda O‘zbekiston Respublikasini rivojlantirishning beshta ustuvor yo‘nalishi bo‘yicha harakatlar strategiyasi mamlakatning davlat va jamiyat rivojlanishi istiqbolini strategik rejalashtirish tizimiga sifat jihatdan yangi yondashuvlarni boshlab berdi.

O‘zbekistonda ijtimoiy-iqtisodiy sohada, shu jumladan axborot kommunikatsiya texnologiyalari, zamonaviy axborotlashgan jamiyatni rivojlantirish xususida keng ko‘lamli o‘zgarishlar izchillik bilan va aniq maqsad sari amalga oshirilmoqda.

Hozirgi kunda rivojlanish va turli jarayonlarni avtomatlashtirish hamda robotlashtirish davrida dasturlash tillarini bilish va uni o‘z ish jarayonida qo‘llay olish texnik va pedagogik yo‘nalishda taxsil olayotgan o‘quvchi-talabalar uchun muhim deb hisoblanadi. Bu malakali mutaxassislar uchun zaruriy talablardan biridir. Sababi hozirgi kunda axborot texnologiyalari turli-tuman sohalarda keng qo‘llanilishi mumkinligini hech kim ham rad eta olmaydi. Xuddi shuning uchun ham, mazkur o‘quv qo‘llanmaning asosiy maqsadi – o‘quvchi talabalarga Python dasturlash tili misolida hisoblash texnikasi vositalarini ishlatish bo‘yicha bilim va ko‘nikmalarni yetkazib berishdir. Turli sohalarda dasturlash tilining imkoniyatlarini ko‘rsatish, ilmiy va matematika sohalarga oid bir qancha masala va misollarni Python dasturlash tilidan foydalanib yechish misolida aniq va ravshan qilib ko‘rsatib o‘tiladi. Shunday qilib, o‘quv-qo‘llanma o‘quvchi talabalar uchun dasturlash tili vositasida turli xildagi amaliy masalalarni hal qilish bilimlarini rivojlantirishga imkon yaratadi. Mazkur o‘quv-qo‘llanmani o‘qish va undagi materiallarni o‘rganish uchun dasturlash bo‘yicha tajribasi bo‘lishi talab etilmaydi va undan endigina dasturchi bo‘lishni orzu qilganlar ham bemalol foydalanishlari mumkin. Shuni ham ta’kidlab o‘tish lozimki, quyidagi o‘quv-qo‘llanmada zamonaviy Python dasturlash tili imkoniyatlari boshlang‘ich

o‘rganuvchilar, ya’ni maktab o‘quvchilari, talabalar va mustaqil o‘rganuvchilar tushunishi uchun nihoyatda yengil, tushunarli, kerakli izohlar bilan va sodda xalq tilida izhor qilingan. Python dasturlash tili samarador yuqori darajadagi ma’lumotlar tuzilmasini hamda oddiy, ammo samarador bo‘lgan obyektga yo‘naltirilgan dasturlash uslublarini taqdim etadi. Undan tashqari, bu til o‘rganish uchun oson va shu bilan birga imkoniyatlari yuqori bo‘lgan oz sonli dasturlash tillari jumlasiga kiradi va shu bilan birgalikda unda dasturlash jarayoni juda ham oddiy amalga oshiriladi. Python dasturlash tilining rasmiy sayti – www.python.org bo‘lib, uning muallifi Niderlandiyadagi Matematika va informatika ilmiy tadqiqot institutida ishlagan *Gvido van Rossum* deb hisoblanadi. Pythonning o‘ziga xosligi esa uning oddiyligi, o‘rganishga osonligi, sodda sintaksisga egaligi va dasturlash jarayonini boshlash uchun qulay, erkin va ochiq kodlik dasturiy ta’minotga egaligidir. Undan tashqari, o‘z dasturingizni yozish davomida quyi darajadagi detallarni, misol uchun xotirani boshqarishni hisobga olishingizga hech qanday hojat qolmaydi. Bu dasturlash tili ko‘plab platformalarda hech qanday o‘zgartirishlarsiz ishlay oladi va u interpretatsiya qilinadigan tillar jumlasiga mansub. [1]

Bulardan tashqari, Python dasturlash tili imkoniyatlari kengayishga moyil bo‘lgan dasturiy til hisoblanadi. Agar siz dasturingizning biror-bir joyini tezroq ishlashini xohlasangiz, o‘sha qismni C yoki C++ dasturlash tillarida yozib, keyin shu qismni Python kodingiz orqali ishga tushirsangiz (chaqirsangiz) bo‘ladi. Bundan tashqari, Python juda ham ko‘p, foydali hamda xilma-xil dasturlar kutubxonalarga egaligi ham juda muhimdir. Python dasturlash tili sodda va o‘qilishi oddiy bo‘lgan dasturlash tili bo‘lib u inglizcha so‘zlarni qo‘llaydi va u PERL va PHP ga tillariga o‘xshab ketadi. Python interaktiv dasturlash tili bo‘lib, ob’ektga yo‘naltirilgan tillar jumlasiga kiradi, ya’ni, Python ob’ektga yo‘naltirish uslubini yoki dasturiy texnikasini qo‘llab quvvatlaydi. Python boshlovchi dasturchilar tilidir, ya’ni u boshlang‘ich dasturchilar

[1] Sh.A.Mengliyev, O.A.Abdug‘aniyev, S.Q.Shonazarov, D.Sh.Turayev. Python dasturlash tili.-Termiz 2021, 3-bet

uchun ajoyib til bo‘lib, oddiy matnni ishlashdan tortib, veb-brauzerlaridagi o‘yinlarga qadar keng ko‘lamdagi ilovalarni ishlab chiqishni qo‘llab quvvatlaydi. [2]

Python dasturlash tili boshqa tillarga nisbatan o‘rganish ancha oson va shu bilan birga imkoniyatlari boy bo‘lgan til hisoblanadi. Ya’ni, til o‘rganishni boshlovchilar uni osonlik bilan o‘rganishlari mumkin, shu bilan bu til yordamida ancha-muncha jiddiy amaliy loyihalarni ham amalga oshirish mumkin.

Python interpretatsiya qilinadigan dasturiy til. Dasturlash tillarini interpretatsiya qilinadigan va kompilyatsiya qilinadigan dasturlash tillariga bo‘lishadi. Aniqroq aytganda, agar dasturlash tilidagi dasturni bajarish interpretatsiya orqali amalga oshirilsa, bunday tillar interpretatsiya qilinadigan til deyiladi. Agar dasturlash tilidagi dasturni bajarish uchun uni avval mashina tiliga o‘tkazish talab qilinsa, bunday tillar kompilyatsiya qilinadigan tillar deyiladi. Aslini olganda, kompyuter uchun yozilgan har qanday dastur interpretatsiya qilinadi. Chunki mashina kodlaridagi dastur kompyuterning miyasi bo‘lgan protsessor tomonidan interpretatsiya qilinadi. Interpretatsiya qilinadigan tillarda yozilgan dasturlar uchun maxsus – interpretator dastur mavjud. Bu interpretator dastur kodlarini bajarilishini ta’minlab beradi. Bu o‘quv - qo‘llanma dasturlashni o‘rganuvchilar hamda ilmiy yoki amaliy maqsadlarni amalga oshirish uchun bu dasturlash tilini o‘rganishi kerak bo‘lgan insonlar uchun mo‘ljallangan. Ushbu qo‘llanmaning asosiy maqsadi - sizga Python tilida dasturlashning nazariy va amaliy asoslarini o‘rgatishdan iboratdir. Dasturlash tilini o‘rganish uchun eng asosiy amal – kitobda berilgan barcha topshiriqlarni o‘z vaqtida, tushungan holda va aniq bajarishdir. Chunki, har qanday soha bo‘yicha chuqur bilim faqatgina amaliyot orqali puxta egallanadi.

[2] Sh.A.Mengliyev, O.A.Abdug‘aniyev, S.Q.Shonazarov, D.Sh.Turayev. Python dasturlash tili.-Termiz 2021, 4-bet

1-MAVZU. ALGORITM, ALGORITMNING BERILISH USULLARI, XOSSALARI VA TURLARI

Reja:

1. Algoritmning tasvirlash usullari
2. Algoritmning xossalari
3. Chiziqli algoritmlar
4. Tarmoqlanuvchi algoritmlar
5. Takrorlanuvchi algoritmlar

Tayanch so‘zlar: Algoritm, algoritm xossalari, algoritm turlari, chiziqli algoritmlar, tarmoqlanuvchi algoritmlar, takrorlanuvchi algoritmlar.

Algoritm tushunchasi.

Algoritm - bu biror masalani yechish uchun bajarilishi zarur bo‘lgan buyruqlarning tartiblangan ketma-ketligidir. Har bir algoritm aniq va tugallangan qadamlarga bo‘lingan bo‘lishi kerak.

Algoritm deb, masalani yechish uchun bajarilishi lozim bo‘lgan amallar ketma-ketligini aniq tavsiflaydigan qoidalar tizimiga aytiladi. Boshqacha aytganda, **algoritm** – boshlang‘ich va oraliq ma’lumotlarni masalani yechish natijasiga aylantiradigan jarayonni bir qiymatli qilib, aniqlab beradigan qoidalarning biror bir chekli ketma-ketligidir.

Buning mohiyati shundan iboratki, agar algoritm ishlab chiqilgan bo‘lsa, uni yechilayotgan masala bilan tanish bo‘lmagan biron bir ijrochiga, shu jumladan, kompyuterga ham bajarish uchun topshirsa bo‘ladi va u algoritmning qoidalariga aniq rioya qilib masalani yechadi.

Algoritm atamasi o‘rta asrlarda yashab ijod etgan buyuk o‘zbek matematigi Al-Xorazmiy nomidan kelib chiqqan. **Algoritm** so‘zi al-Xorazmiyning arifmetikaga bag‘ishlangan asarining dastlabki betidagi **“Dixit Algoritmi” (“dediki al-Xorazmiy”**ning lotincha ifodasi) degan jumlalardan kelib chiqqan. U o‘zi kashf etgan o‘nli sanoq tizimida IX asrning 825 yilidayoq to‘rt arifmetika amallarini bajarish qoidalarini bergan. Arifmetika amallarini bajarish jarayoni esa alxorazm deb atalgan. Bu atama 1747-yildan boshlab algorismus, 1950-yilga kelib algoritm deb ham ataldi.

Bu yerda al-Xorazmiyning sanoq sistemasini rivojlantirishga qo‘shgan hissasi, uning asarlari algoritm tushunchasining yaralishiga sabab bo‘lganligi o‘quvchilarga ta’kidlab o‘tiladi.

Qadimgi Gretsiyalik matematik Evklid 2 ta natural A va B sonlarning eng katta umumiy bo‘luvchisini topish algoritmini taklif etdi. Uning ma’nosi quyidagicha:

Katta sondan kichigini ayirish, natijani katta son o‘rniga qo‘yish va ikkala son tenglashguncha bu amalni takrorlash. Ushbu teng sonlar izlangan natijadir.

Evklid algoritmida A va B sonlarning eng katta umumiy bo‘luvchisi ushbu sonlar ayirmasining eng katta bo‘luvchisi hamda ikkala A, B sonlarning ham umumiy eng katta bo‘luvchisi bo‘lishligidan foydalanilgan.

Evklid algoritmining bu ifodasiga aniqlik yetishmaydi, shuning uchun uni konkretlashtirish zarur bo‘ladi.

Haqiqiy Evklid algoritmi quyidagicha:

1. A sonni birinchi son deb, B sonni ikkinchi son deb qaralsin. 2-qadamga o‘tilsin.

2. Birinchi va ikkinchi sonlarni taqqoslang. Agar ular teng bo‘lsa, 5-qadamga o‘tilsin, aks holda 3-qadamga o‘tilsin.

3. Agar birinchi son ikkinchi sondan kichik bo‘lsa, ularning o‘rni almashtirilsin. 4-qadamga o‘tilsin.

4. Birinchi sondan ikkinchi son ayirilsin va ayirma birinchi son deb hisoblansin. 2-qadamga o‘tilsin.

5. Birinchi sonni natija sifatida qabul qilinsin. Tamom.

Bu qoidalar ketma-ketligi algoritmni tashkil etadi, chunki ularni bajargan ixtiyoriy ayirishni biladigan kishi ixtiyoriy sonlar jufti uchun eng katta umumiy bo‘luvchini topa oladi.

Matematiklar uzoq vaqtlar davomida algoritmlarni bunday ifodalaridan keng foydalanib turli hisoblash algoritmlarini ishlab chiqdilar.

Masalan, kvadrat va kubik tenglamalar ildizlarini topish algoritmlari topildi. Asta-sekin olimlar qiyinroq masalalar ustida bosh qotirib, masalan, ixtiyoriy darajali algebraik tenglamalar ildizlarini topish algoritmlarini qidiradilar. Hatto, XVII – asrda Leybnis ixtiyoriy matematik masalani yechishning umumiy algoritmini topishga urinib

ko‘rgan. Ammo bunga o‘xshash algoritmlarni ko‘rishning iloji bo‘lmagan va asta-sekin buning butunlay imkoni yo‘q degan xulosaga kelingan.

Shunday bo‘lishiga qaramay, algoritm tushunchasining aniq tavsifi berilmagunga qadar, masalaning algoritmik yechimsizligini isbotlash mumkin emas edi.

Algoritm tushunchasi juda qadim zamonlardan shakllanib kelgan. Shunga qaramay, asrimizning yarmiga qadar matematiklar bu ob’yekt haqida ma’lum bir qarashlarga qanoatlanib kelganlar. Algoritm atamasi matematiklar tomonidan faqat konkret masalalarni yechish bilan bog‘liq holda olinar edi.

XX asr boshida matematika asoslarida vujudga kelgan qarama-qarshiliklar va muammolar ularni hal etishga qaratilgan turli konsepsiyalar va oqimlarning vujudga kelishiga olib keldi. 20-yillarga kelib effektiv hisoblash masalalari ko‘ndalang bo‘ldi.

Algoritm tushunchasining o‘zi matematik tadqiqotlar ob’yekti bo‘lib qolganligi uchun aniq va qat’iy ta’rifga muhtoj edi. Bundan tashqari, kompyuter asrini yaqinlashtiruvchi fizika va texnikaning rivojlanishi ham shuni taqozo etar edi.

Algoritmni mukammalroq tushunarli bo‘lishi uchun o‘quvchilarga turli hayotdan, fandan algoritmlarga misollar keltirish va bunga o‘zlari tuzishga harakat qilishlarini taklif etish mumkin. Masalan, sport musobaqasi, ovqat pishirish, turli gadjetlarni ishlatish yoki yo‘l harakati qoidalari algoritmlarini keltirib o‘tish mumkin, yoki matematika oid formula bo‘yicha qiymat hisoblash algoritmi yoki kompyuterni ishlatish bo‘yicha algoritm kabi misollar keltirilishi mumkin.

Algoritmning asosiy xossalari

Algoritm quyidagi asosiy xossalarini o‘z ichiga oladi**:**

1-xossa. **Diskretlilik-**algoritmni chekli sondagi ko‘rsatmalar ketma-ketligi ko‘rinishida ifodalash mumkin. Tugallangan amallar ketma-ketligi qadam deyiladi. Demak, algoritm chekli qadamlardan iborat bo‘lishi kerak.

2-xossa. Tushunarlilik-ijrochiga tavsiya etilayotgan ko‘rsatmalar uning uchun tushunarli bo‘lishi shart, aks holda ijrochi oddiy amalni ham bajara olmay qolishi mumkin. Har bir ijrochining bajara olishi mumkin bo‘lgan ko‘rsatmalar tizimi mavjud.

3-xossa. Aniqlik-ijrochiga berilayotgan ko‘rsatmalar aniq mazmunda bo‘lishi lozim hamda faqat algoritmda ko‘rsatilgan tartibda bajarilishi shart. Algoritmning har bir qoidasi aniq va bir qiymatli bo‘lishi zarurki, bunda vaqtning biror daqiqasida olingan miqdorlar qiymati vaqtning shundan oldingi daqiqasida olingan miqdorlar qiymati bilan bir qiymatli aniqlangan bo‘ladi.

4-xossa. **Ommaviylik-**har bir algoritm mazmuniga ko‘ra bir turdagi masalalarning barchasi uchun yaroqli bo‘lishi lozim. Algoritm bitta masalani yechish uchun emas balki shunga o‘xshash turdosh masalalar sinfini hal etish uchun xizmat qiladi. Masalan, ikki oddiy kasr umumiy maxrajini topish algoritmi har qanday kasrlar umumiy maxrajini topish uchun ishlatiladi.

5-xossa. Natijaviylik-har bir algoritm chekli sondagi qadamlardan so‘ng, albatta, natija berishi kerak. Algoritm masalaning yechimiga chekli sondagi qadamlar ichida olib kelishi yoki masalani "yechib bo‘lmaydi" degan xabar bilan tugashi kerak.

Bu xossalar mohiyatini o‘rganish va konkret algoritmlar uchun qarab chiqish talabalarning xossalar mazmunini bilib olishlariga yordam beradi.

Algoritmning tasvirlash usullari quyidagilardan iborat:

Algoritmning tasvirlash usullari haqida so‘zlaganda algoritmning berilish usullari xilma-xilligi va ular orasida eng ko‘p uchraydiganlari quyidagilar ekanligini ko‘rsatib o‘tish joiz:

1. Algoritmning so‘zlar orqali ifodalanishi. Algoritmlarni ifodalashning eng keng tarqalgan shakli – oddiy tilda so‘zlar bilan bayon qilishdir. Bu nafaqat hisoblash algoritmlarida, balki hayotiy, turmushdagi "algoritm"larga ham tegishlidir.

Masalan, biror bir taom yoki qandolat mahsulotini tayyorlashning retsepti ham oddiy tilda tavsiflangan algoritmdir. Shaharlararo telefon - avtomat orqali aloqa o‘rnatishning o‘ziga xos algoritmidan foydalanasiz. Do‘kondan yangi kir yuvish mashinasi yoki magnitofon sotib olinsa, ishni foydalanishning algoritmi bilan tanishishdan boshlaymiz.

Masalani kompyuterda yechishda ham, ko‘pincha matematika tilini ham o‘z ichiga olgan tabiiy tildan foydalanish mumkin. Algoritmning bunday tildagi yozuvi izlanayotgan natijaga olib keladigan amallar ketma-ketligi ko‘rinishida bo‘lib, odam

tomonidan bir ma’noli idrok etilishi kerak. So‘zlar bilan ifodalangan har bir amal “algoritmning qadami” deb ataladi. Qadamlar tartib nomeriga ega bo‘ladi.

Algoritm ketma-ket, qadam-ba qadam bajarilishi kerak. Agar algoritm matnida "N sonli qadamga o‘tilsin" deb yozilgan bo‘lsa, bu algoritmning bajarilishi ko‘rsatilgan N-qadamdan davom etishini bildiradi.

2. Algoritmning formulalar yordamida berilishi. Algoritm matematik formulalar yordamida ifodalanganda har bir qadam aniq formulalar yordamida yoziladi. Misol tariqasida

$$ax^2 + bx + c = 0 \ (a \neq 0)$$

Kvadrat tenglama yechimlari bo‘lmish x_1 x_2 ni aniqlash algoritmini ko‘rib chiqamiz.

1. a, b, c koeffitsientlar qiymatlari berilsin.

2. $D = b^2-4ac$ diskriminant hisoblansin.

3. D <0 bo‘lsa, tenglamaning haqiqiy yechimlari yo‘q. Faqat haqiqiy ildizlar izlanayotgan bo‘lsa, masala hal bo‘ldi.

4. D = 0 bo‘lsa, tenglama ikkita bir-biriga teng, ya’ni karrali yechimga ega bo‘ladi va ular formulalar bilan hisoblanadi. Masala hal bo‘ldi.

5. D >0 bo‘lsa, tenglama ikkita haqiqiy yechimga ega, ular

$$x_1 = (b + \sqrt{D}) / 2a \ \text{ va } \ x_2 = (-b - \sqrt{D}) / 2a$$

formulalar bilan hisoblanadi. Ya’ni masala hal bo‘ldi.

3. Algoritmning jadval ko‘rinishida ifodalanishi. Algoritmning bu tarzda ifodalanishidan ham ko‘p hollarda foydalanamiz. Misol uchun, maktabda qo‘llanib kelinayotgan to‘rt xonali matematik jadvallar yoki turli xil lotoreyalar jadvallarini keltirish mumkin. Funksiyaning grafiklarini chizishda ham algoritmlarning qiymatlari jadval ko‘rinishidan foydalanamiz. Shu singari jadvallardan foydalanish algoritmlari sodda bo‘lganligi tufayli ularni o‘rganish oson.

4. Algoritmning dastur shaklida ifodalanishi, ya’ni algoritm kompyuter ijrochisiga tushunarli bo‘lgan dastur shaklida beriladi**.**

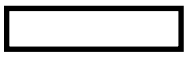
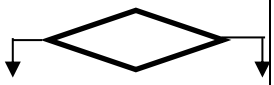
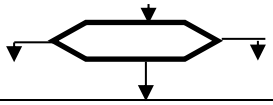
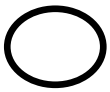
5. Algoritmning algoritmik tilda tasvirlanishi, ya’ni algoritm bir xil va aniq ifodalash, bajarish uchun qo‘llanadigan belgilash va qoidalar majmui algoritmik til orqali ifodalashdir. Algoritmik tillar - algoritmni bir ma’noli tavsiflash imkonini beradigan belgilar va qoidalar majmuidir. Har qanday tillardagidek ular ham o‘z alifbosi, sintaksisi va semantikasi bilan aniqlanadi

Bizga o‘rta maktabdan ma’lum bo‘lgan kompyuterlarsiz algoritmlashga mo‘ljallangan algoritmik tizim algoritmik tilning namunasidir. Algoritmik tilga misol sifatida yana algoritmlarni belgili operatorlar tizimi shaklida tavsiflashni ham ko‘rsatish mumkin. Bu tillar odatdagi tilga o‘xshash bo‘lib, kompyuterda bevosita bajarishga mo‘ljallanmagan. Ulardan maqsad algoritmni bir xil shaklda va tushunarli qilib, tahlil qilishga oson qilib yozishdir.

6. Algoritmlarning grafik shaklda tasvirlanishi. **Masalan**, grafiklar, sxemalar ya’ni blok - sxema bunga misol bo‘la oladi. Blok sxemaning asosiy elementlari quyidagilar: **oval (ellips shakli) -** algoritm boshlanishi va tugallanishi, **to‘g‘ri burchakli to‘rtburchak** - qiymat berish yoki tegishli ko‘rsatmalarni bajarish. **Romb - shart** tekshirishni belgilaydi. Uning yo‘naltiruvchilari tarmoqlar bo‘yicha biri ha ikkinchisi yo‘q yo‘nalishlarni beradi, **parallelogramm** - ma’lumotlarni kiritish yoki chiqarish, **yordamchi algoritmga murojaat -** parallelogramm ikki tomoni chiziq**, yo‘naltiruvchi chiziq -** blok-sxemadagi harakat boshqaruvi, **nuqta-to‘g‘ri chiziq (ikkita parallel) -** qiymat berish.

Algoritmda bajarilishi tugallangan amallar ketma-ketligi **algoritm qadami** deb yuritiladi. Har bir alohida qadamni ijro etish uchun bajarilishi kerak bo‘lgan amallar haqidagi ko‘rsatma buyruq deb aytiladi. Algoritmlarni ko‘rgazmaliroq qilib tasvirlash uchun blok-sxema, ya’ni geometrik usul ko‘proq qo‘llaniladi.

Algoritmning blok-sxemasi algoritmning asosiy tuzilishining yaqqol geometrik tasviri: algoritm bloklari, ya’ni geometrik shakllar ko‘rinishida, bloklar orasidagi aloqa esa yo‘naltirilgan chiziqlar bilan ko‘rsatiladi. Chiziqlarning yo‘nalishi bir blokdan so‘ng qaysi blok bajarilishini bildiradi. Algoritmlarni ushbu usulda ifodalashda vazifasi, tutgan o‘rniga qarab quyidagi **geometrik shakl (blok)** lardan foydalaniladi.

Blokning atalishi	Belgilanishi	Tushunilishi
Hisoblashlar bloki (to‘g‘ri - to‘rtburchak)		Hisoblash amali yoki hisoblash amallari ketma-ketligi
Shartli blok (romb)		Shartlarni tekshirish
Siklik jarayon (oltiburchak)		Siklning boshlanishi
Qism dastur		Qism dastur bo‘yicha hisoblash, standart qism dasturi
Birlashtirish (aylana)		Yo‘nalish chizig‘ini o‘zgartirish
Ma’lumotlarni kiritish va chiqarish parallelogramm)		Ma’lumotlarni kiritish va natijalarni chiqarish
Algoritmning boshi va oxiri (oval)		Boshlash, tamom, to‘xtash
Chiqarish bloki		Ma’lumotlarni qog‘ozga chiqarish

Blok-chizmalarning asosiy afzalligini, algoritmni tasvirlashning yaqqolligi bilan izohlash hisoblanadi.

Masalani yechishning algoritmlarini blok-chizmalar asosida ifoda etib ularning yaqqolligini ta’minlash borasida quyidagi texnik qoidalarni inobatga olish kerak bo‘ladi:

a) bloklarni gorizontal va vertikal holatda joylashtirish kerak;

b) bloklar orasidagi masofani shunday olish kerakki, ularni ulab turuvchi chiziqlar yetarli darajada kalta bo‘lsin;

c) bloklarning geometrik o‘lchami hisoblash jarayonini tavsiflashning hajmiga to‘g‘ri kelishi kerak. Shu bilan birgalikda blokning uzunligi uning balandligiga nisbatan v=1,5*a ko‘rinishda bo‘lishi kerak;

d) blok-chizmalarda bloklarning joylashishi va guruhlarga ajratilishi shunday bo‘lishi kerakki, ularni ko‘rib axborotlarni yaxshi qabul qilish ta’minlanishi lozim

Yuqorida keltirilgan algoritmlarning tasvirlash usullarining asosiy maqsadi, berilgan masalani yechish uchun kerakli bo‘lgan amallar ketma-ketligining eng qulay

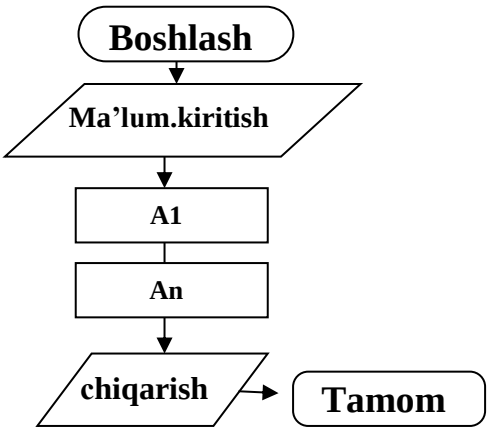
holatini aniqlash va shu bilan bajaruvchi tomonidan dastur yozishni yanada yengillashtirishdan iborat. Aslini olganda dastur ham algoritmning boshqacha ko‘rinishi bo‘lib, u inson va kompyuter o‘rtasidagi muloqotni qulayroq amalga oshirishga mo‘ljallangan.

Chiziqli jarayonlarni algoritmlash

Chiziqli algoritmlar algoritmlarning eng sodda va oddiy ko‘rinishi hisoblanadi. Unda bajariladigan amallar ham, buyruqlar ham qanday tartibda berilgan bo‘lsa shunday tartibda ketma-ket bajariladi.

Chiziqli tuzilishga ega bo‘lgan algoritmlarda ko‘rsatmalar yozilish tartibida bajariladi. Ularning blok-sxemalari ishga tushirish, to‘xtatish, kiritish-chiqarish jarayoni bloki hamda avvaldan ma’lum jarayon bloklari yordamida tuzilib, bir chiziq bo‘ylab ketma-ket joylashgan bo‘ladi.

Chiziqli tuzilishdagi algoritmni tuzish masalani yechish uchun kerak bo‘ladigan boshlang‘ich ma’lumotlarni tashkil qiluvchi o‘zgaruvchilar nomi, ularning turi va o‘zgarish ko‘lamini aniqlashdan boshlanadi. Keyin oraliq va yakuniy natijalar o‘zgaruvchilarining nomlari, turlari va mumkin bo‘lsa o‘zgarish ko‘lamini aniqlash kerak. Endi algoritm mana shu boshlang‘ich ma’lumotlarni qanday qayta ishlab oraliq va yakuniy natijalarni olish kerakligini aniqlashdan iborat bo‘ladi. Chiziqli algoritmlarni quyidagi ko‘rinishda ifodalash mumkin.

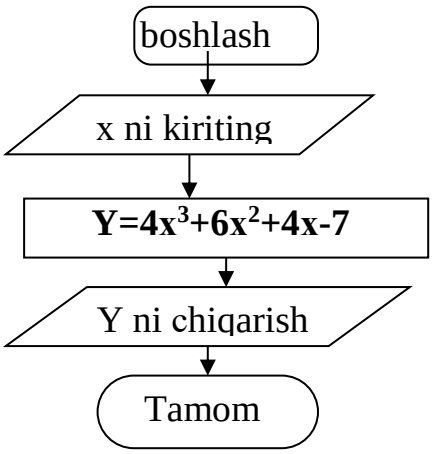


Bu yerda A1,…, AN lar chiziqli algoritmlarda bajarilishi kerak bo‘lgan buyruqlar ketma-ketligidir.

1-misol. $\mathbf{Y=4x^3+6x^2+4x-7}$ funksiyani x ning ixtiyoriy qiymatlarida hisoblash algoritmini tuzing.

Yechish.

Algoritmning blok sxema ko‘rinishi:



Algoritmning matn ko‘rinishi:

1.Boshlash.
2.x ni kiritish.
3.$Y=4x^3+6x^2+4x-7$ ni qiymatini hisoblash.
4.y ni qiymatini chiqarish.
5.Tамом

Tarmoqlanuvchi jarayonlarni algoritmlash.

Shunday hisoblash jarayonlari mavjud bo‘ladiki, bunda qo‘yilgan ayrim mantiqiy shartlarning bajarilishiga qarab, bu jarayonlar bir nechta tarmoqqa bo‘linadi. Hisoblash jarayonlarining shundayiga tarmoqlangan deb ataladiki, unda u birlamchi yoki oraliq ma’lumotlar xususiyatidan kelib chiqqan holda bir yoki bir necha yo‘nalish bo‘yicha bajarilishi mumkin bo‘ladi. Bunda har bir yo‘nalish hisoblash jarayonining tarmog‘i hisoblanadi. U yoki bu tarmoqning tanlanishi mantiqiy shartlarning bajarilishini tekshirish asosida ta’minlanadi. Aniq bir holda jarayon faqat tarmoqlarning bittasi bo‘yicha bajariladi. Boshqa tarmoqlanishlarning bajarilishi mumkin emas.

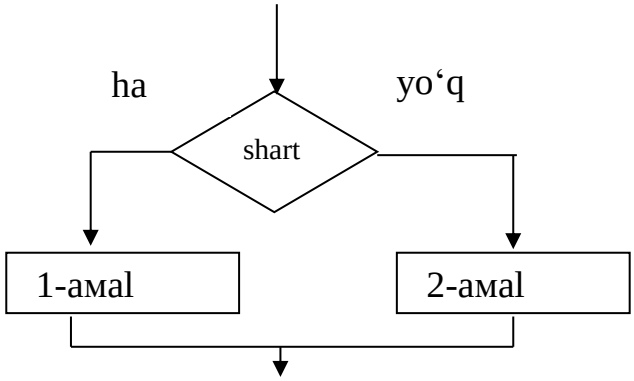
Tarmoqlanuvchi struktura odatda qandaydir mantiqiy shartni tekshirish blokini o‘z ichiga oladi. Tekshirish natijasiga ko‘ra, tarmoq deb ataluvchi u yoki bu amallar ketma-ketligi bajariladi va shu tarmoqlardan hech bo‘lmaganda bittasi bajariladi.

Shartni tekshirish natijasi faqat ikki xil bo‘lganda: bajarilgan hol uchun «Ha» (yoki «+»), bajarilmagan hol uchun «Yo‘q» (yoki «-») belgilari qo‘yiladi.

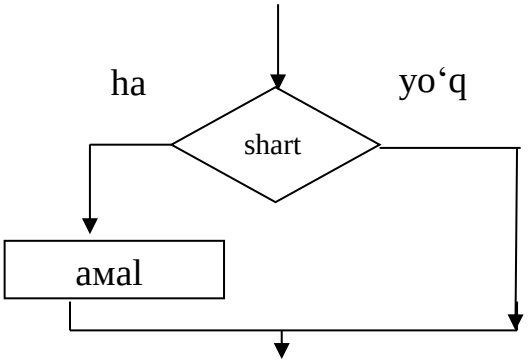
Tarmoqlanish matematik ifoda qiymatining ishorasi bo‘yicha bo‘lganda (arifmetik shart): «>» — musbat, «<» — manfiy va «=» — nolga teng belgilar qo‘yiladi.

Ana shunday jarayonlar uchun algoritmlar tuzishda tarmoqlanuvchi algoritmlardan foydalaniladi. Tarmoqlanuvchi algoritm to‘la va qisqartirilgan ko‘rinishda berilishi mumkin.

Ular quyidagicha sxema orqali ifodalanadi:



a) to‘la ko‘rinish



b) qisqartirilgan ko‘rinish

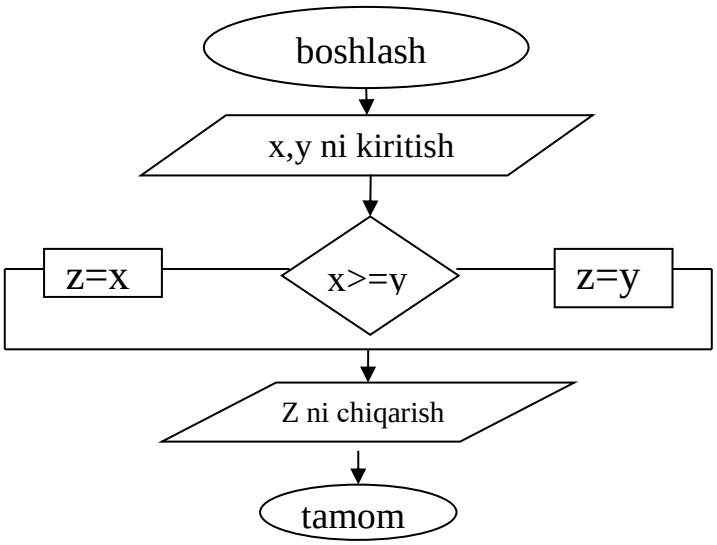
Tarmoqlanuvchi jarayon uchun algoritm tuzishga doir misol keltiramiz.

1-misol. Ixtiyoriy berilgan x va y sonlar ichidan eng kattasini topish algoritmini tuzing.

Yechish. Buning uchun berilgan ikkita sonni taqqoslaymiz: agar **x>=y** bo‘lsa, u holda **x** sonini katta son **z=x** deb, aks holda esa y sonini katta son **z=y** deb qabul qilamiz.

Algoritmning matn va blok sxema ko‘rinishlari:

1. Boshlash.
2. x va y qiymatlarni kiritish.
3. Agar x>= y bo‘lsa, natija x deb olinib 5 ga o‘tilsin.
4. Natija y deb olinsin.
5. Tamom.



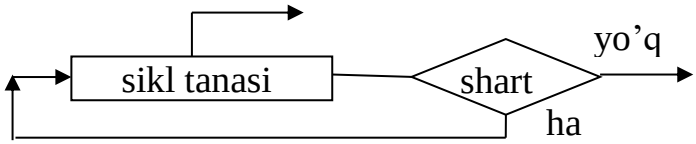
Takrorlanuvchi jarayonlarni algoritmlash.

Shunday hisoblash jarayonlari mavjudki, bunda uning ayrim bo‘laklarini bir necha marta takroran hisoblashga to‘g‘ri keladi. Bunday jarayonlar takrorlanuvchi yoki siklik jarayonlar deyiladi va ular uchun algoritmlar tuzishda takrorlanuvchi algoritmlardan foydalaniladi. Siklik jarayonlar sikl parametri va sikl tanasidan iborat bo‘ladi. Hisoblash jarayonining ko‘p marta takrorlanadigan qismi ichki **sikl tanasi** deb yuritiladi. Sikl tanasi bir necha marta takroran bajariluvchi amallar ketma-ketligidan iborat bo‘ladi. Siklli jarayonlarni algoritmlashda bitta yoki bir nechta parametrlar qatnashadi. Ularning qiymatlarini bir vaqtda o‘zgarishida sikl tanasi ichidagi amallar ketma-ketligi ko‘p marotaba takrorlanadi.

Masalaning qo‘yilishiga qarab siklik jarayonlar ***takrorlanish soni ma’lum bo‘lgan*** va ***takrorlanish soni noma’lum*** bo‘lgan sikllarga bo‘linadi.

Takrorlanuvchi algoritmlar 2 xil ko‘rinishda ifodalanishi mumkin.

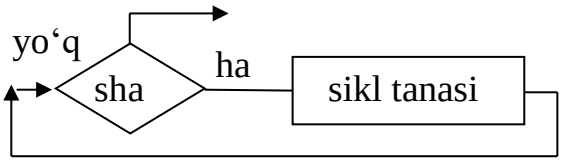
Sikl – gacha takrorlanuvchi algoritm quyidagi ko‘rinishga ega:



a) sikl - gacha

Bu ko‘rinishdagi algoritmda avval sikl tanasi bajarilib, so‘ngra parametrning qiymati, ya’ni sikldan chiqish sharti tekshiriladi. Bu yerda sikl tanasi qo‘yilgan shart bajarilib turguncha takrorlanaveradi.

Sikl - hozircha takrorlanuvchi algoritm quyidagi ko‘rinishga ega:



b) sikl - hozircha

Bu ko‘rinishdagi algoritmlarda avval shart tekshiriladi, so‘ngra agar shart qanoatlantirsa, sikl tanasi bajariladi, aks holda hisoblash to‘xtatiladi.

Aniq berilgan son asosida sikllarni tashkil qilishda sikl parametrining boshlang‘ich va oxirgi qiymatlari, uning har bir takrorlanishidagi sikl parametrining o‘zgarish qonunlari, sikllarning takrorlanish sonlari ko‘rsatilishi kerak bo‘ladi.

Sikl tanasidagi birlamchi ma’lumotlar doimiy kattalik, oddiy o‘zgaruvchan, indeksli o‘zgaruvchan ko‘rinishlarida bo‘lishi mumkin.

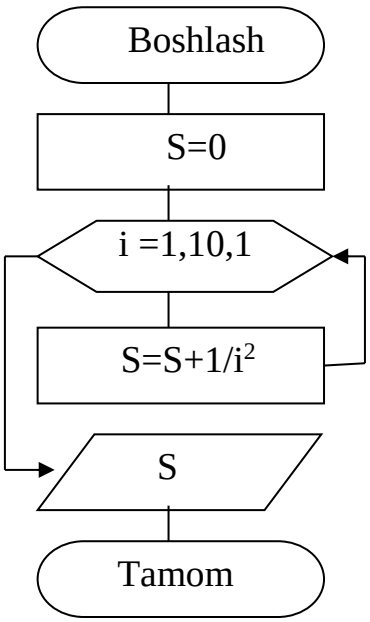
1-misol. $S=\sum_{i=1}^{10}\frac{1}{i^2}=1+\frac{1}{4}+\frac{1}{9}+\ldots+\frac{1}{100}$ yig‘indini hisoblash algoritmini tuzing.

Yechish: Bu yerda sikl parametrlarning barcha qiymatlari ma’lum. Yig‘indini takroriy qiymatini hosil qilish uchun uning boshlang‘ich qiymati sifatida S=0 olinadi. Sikl parametri *i* bo‘lib u *1* dan *10* gacha qiymatlarni qabul qiladi. Uning har bir qiymatida $1/i^2$ifoda takroran $S=S+1/i^2$ hisoblanadi.

Matn ko‘rinish:

1. Boshlash.
2. S=0
3. i=1
4. $S=1/i^2$
5. Agar i<100 bo‘lsa, u holda i=i+1 ва 4ga o‘tish.
   Aks holda 6 ga.
6. S ni qiymati chiqarilsin.
7. Tamom.

Blok sxema



2-MAVZU. PYTHON DASTURLASH TILI VA UNING IMKONIYATLARI

Reja:

1. Python dasturlash tili yaratilishi tarixi
2. Python dasturlash tili imkoniyatlari
3. Python dasturlash tilining asosiy operatorlari va sintaksisi.

Tayanch tushunchalar: python dasturlash tili, sintaksis, OYD, linux.

Python dasturlash tili vujudga kelish tarixi, uning imkoniyatlari va python programmasini o‘rnatish

Python dasturlash tilini vujudga kelishi 1980-yil oxiri 1990-yil boshlaridan boshlangan. O‘sha paytlarda uncha taniqli bo‘lmagan Gollandiyaning CWI instituti xodimi Gvido van Rossum ABC tilini yaratilish proyektida ishtirok etgan edi. ABC tili Basic tili o‘rniga talabalarga asosiy dasturlash konsepsiyalarini o‘rgatish uchun mo‘ljallangan til edi. Bir kun Gvido bu ishlardan charchadi va 2 hafta davomida o‘zining Macintoshida boshqa oddiy tilning interpretatorini yozdi, bunda u albatta ABC tilining ba’zi bir g‘oyalarini o‘zlashtirdi. Shuningdek, Python 1980-1990-yillarda keng foydalanilgan Algol-68, C, C++, Modul3 ABC, Small Talk tillarining ko‘plab xususiyatlarini o‘ziga olgandi. Gvido van Rossum bu tilni internet orqali tarqata boshladi. Bu paytda o‘zining “Dasturlash tillarining qiyosiy taqrizi” web sahifasi bilan internetda to 1996-yilgacha Stiv Mayevskiy ismli kishi taniqli edi. U ham Macintoshni yoqtirardi va bu narsa uni Gvido bilan yaqinlashtirdi. O‘sha paytlarda Gvido BBC ning “Monti Paytonning havo sirki” komediyasining muxlisi edi va o‘zi yaratgan tilni Monti Payton nomiga Python deb atadi (ilon nomiga emas). [3]

Til tezda ommalashdi. Bu dasturlash tiliga qiziqqan va tushunadigan foydalanuvchilar soni ko‘paydi. Boshida bu juda oddiy til edi. Shunchaki kichik interpretator bir nechta funksiyalarga ega edi. 1991-yil birinchi OYD (Obyektga Yo‘naltirilgan Dasturlash) vositalari paydo bo‘ldi.

[3] Sh.A.Mengliyev, O.A.Abdug‘aniyev, S.Q.Shonazarov, D.Sh.Turayev. Python dasturlash tili.-Termiz 2021, 7-bet

Bir qancha vaqt o‘tib Gvido Gollandiyadan Amerikaga ko‘chib o‘tdi. Uni CNRI korporatsiyasiga ishlashga taklif etishdi. U o‘sha yerda ishladi va korporatsiya shug‘ullanayotgan proyektlarni Python tilida yozdi va bo‘sh ish vaqtlarida tilni interpretatorini rivojlantirib bordi. Bu 1990-yil Python 1.5.2 versiyasi paydo bo‘lguncha davom etdi. Gvidoning asosiy vaqti korporatsiyani proyektlarini yaratishga ketardi bu esa unga yoqmasdi. Chunki uning Python dasturlash tilini rivojlantirishga vaqti qolmayotgandi. Shunda u o‘ziga tilni rivojlantirishga imkoniyat yaratib bera oladigan homiy izladi va uni o‘sha paytlarda endi tashkil etilgan BeOpen firmasi qo‘llab quvvatladi. U CNRI dan ketdi, lekin shartnomaga binoan u Python 1.6 versiyasini chiqarib berishga majbur edi. BeOpen da esa u Python 2.0 versiyani chiqardi. 2.0 versiyasi bu oldinga qo‘yilgan katta qadamlardan edi. Bu versiyada eng asosiysi til va interpretatorni rivojlanish jarayoni ochiq ravishda bo‘ldi.[4]

Shunday qilib 1.0 versiyasi 1994-yil chiqarilgan bo‘lsa, 2.0 versiyasi 2000-yil, 3.0 versiyasi esa 2008-yil ishlab chiqarildi. Hozirgi vaqtda uchinchi versiyasi keng qo‘llaniladi.

Python dasturlash tili imkoniyatlari

Python – bu o‘rganishga oson va shu bilan birga imkoniyatlari yuqori bo‘lgan oz sonlik zamonaviy dasturlash tillari qatoriga kiradi. Python yuqori darajadagi ma'lumotlar strukturasi va oddiy lekin samarador obyektga yo‘naltirilgan dasturlash uslublarini taqdim etadi.

Pythonning o‘ziga xosligi quyidagilar:

- Oddiy, o‘rganishga oson, sodda sintaksisga ega, dasturlashni boshlash uchun qulay, erkin va ochiq kodli dasturiy ta'minot.
- Dasturni yozish davomida quyi darajadagi detallarni, misol uchun xotirani boshqarishni hisobga olish shart emas.
- Ko‘plab platformalarda hech qanday o‘zgartirishlarsiz ishlay oladi.
- Interpretatsiya (Интерпретируемый) qilinadigan til.

[4] Sh.A.Mengliyev, O.A.Abdug‘aniyev, S.Q.Shonazarov, D.Sh.Turayev. Python dasturlash tili.-Termiz 2021, 7-bet

• Kengayishga (Расширяетый) moyil til. Agar dasturni biror joyini tezroq ishlashini xohlasak shu qismni C yoki C++ dasturlash tillarida yozib keyin shu qismni python kodi orqali ishga tushirsa (chaqirsa) bo‘ladi.

• Juda ham ko‘p xilma-xil kutubxonalarga ega.

• xml/html fayllar bilan ishlash

• http so‘rovlari bilan ishlash

• GUI(grafik interfeys)

• Web ssenariy tuzish

• FTP bilan ishlash

• Rasmli audio video fayllar bilan ishlash

• Robot texnikada

• Matematik va ilmiy hisoblashlarni programmalash

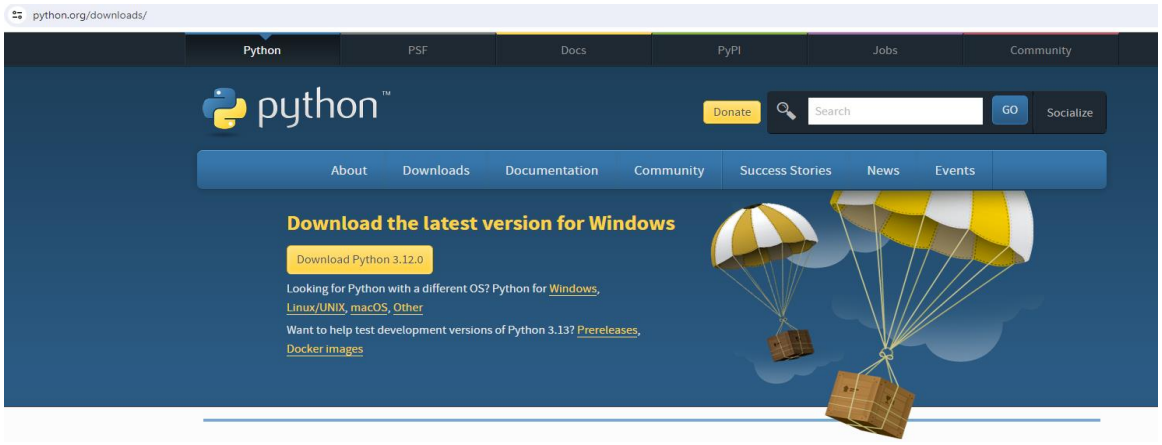
Pythonni katta proyektlarda ishlatish mumkin. Chunki, uni chegarasi yo‘q, imkoniyati yuqori. Shuningdek, u sodda va universalligi bilan programmalash tillari orasida eng yaxshisidir.

Python dasturlash tilini o‘rnatish.

Agarda biz biror GNU/Linux distributivini ishlatayotgan bo‘lsak, aksariyat hollarda bizning tizimingizda **python** dasturi o‘rnatilgan bo‘ladi. Buni tekshirib ko‘rish uchun buyruqlar panelidan quyidagi buyruqni ishga tushirib ko‘ramiz. **python -V**

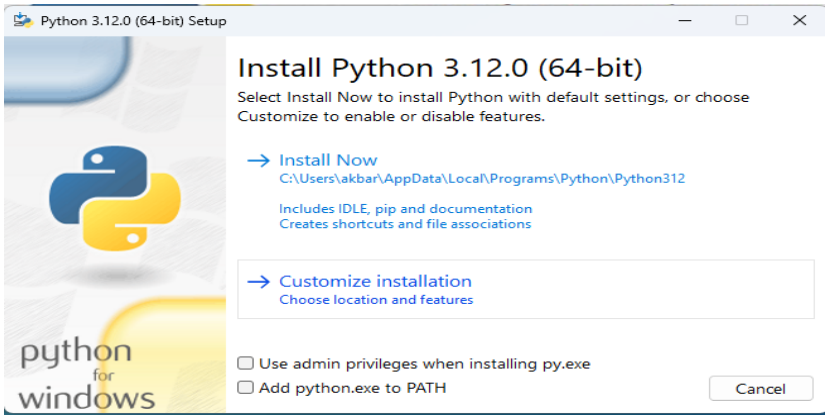
Agar bizda **Python 3.4.3** yozuvi yoki shunga o‘xshash yozuv hosil bo‘lsa unda hammasi joyida.

Windows operatsion tizimiga dasturni o‘rnatish uchun www.python.org/downloads veb sahifasiga o‘tamiz va u yerdan python dasturining oxirgi versiyasini yuklaymiz. Pythonni o‘rnatish boshqa dasturlarni o‘rnatish kabi kechadi.

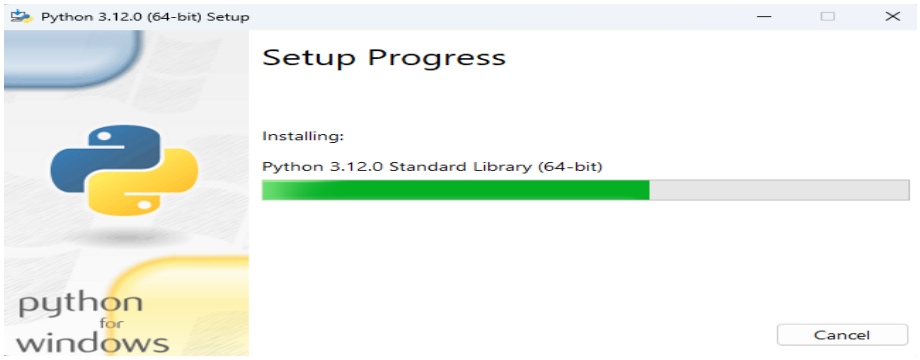


Python dasturining o‘rnatiluvchi fayli.

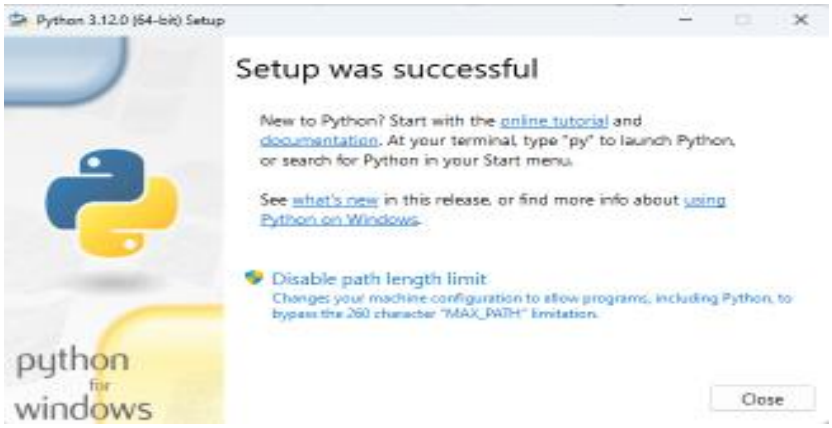
Python dasturlash tilining o‘rnatuvchi paketini ustiga sichqoncha ko‘rsatkichini 2 marta bosamiz va bizga quyidacha muloqot oynasi ochiladi.



Dasturni o‘rnatishni boshlashni ko‘rsatuvchi oyna. Bu yerdan **Install Now**-hozir o‘rnatishni tanlaymiz.



Python dasturi o‘rnatilyapti va bir necha soniyadan so‘ng quyidagicha oyna hosil bo‘ladi:



Python dasturini o‘rnatish tugatilganligi to‘g‘risidagi muloqot oynasi va dasturni o‘rnatish muvoffaqqiyatli kechdi.

Python dasturlash tili asosiy operatorlari va sintaksisi.

Python dasturlash tili sintaksisi o‘zi kabi sodda:

• Satr oxiri instruksiyaning oxiri hisoblanadi (nuqta vergul shart emas).

• Har bir qator boshidagi bo‘sh joy(отступ) muhim ahamiyatga ega. Kiritilgan amallar bo‘sh joylarning kattaligiga qarab **blok**larga birlashadi. Bo‘sh joy istalgancha bo‘lishi mumkin asosiysi bitta kiritilgan blok chegarasida bo‘sh joy bir xil bo‘lishi kerak. Noto‘g‘ri qo‘yilgan bo‘sh joylar xatolik yuz berishiga olib kelishi mumkin. Bitta probel bilan bo‘sh joy hosil qilish yaxshi qaror emas, uni o‘rniga to‘rtta probel yoki *Tab* belgisini ishlatish kerak.

• Pythonga kiritilgan amallar bir xil shablonda yoziladi. Bunda asosiy amal ikki nuqta bilan tugatiladi va uning orqasidan kiritilgan blok kodi ham joylashadi.

Odatda, asosiy amalning ostidagi satr bo‘sh joy bilan ajratiladi.

Bir nechta maxsus holatlar

Bazan bir nechta amalni bitta satrga nuqtali vergul bilan ajratgan holda yozish mumkin.

```
a = 1; b = 2; print(a, b)
```

Buni ko‘p ham qo‘llamang! Yaxshisi bunday qilmang, o‘qishga noqulay.

Bitta amalni bir nechta satrga yozish mumkin faqat aylana, to‘rtburchak va figurali qavslardan foydanish kerak.

Kalit so‘zlar: False – yolg‘on, **true** – rost, **none -** “bo‘sh” obyekt, **and** – mantiqiy VA amali, **with / as** – konteks menejeri, **break** –sikldan chiqish, **class** – metod va

atributlarda iborat, **continue** – sikldan keyingi iteratsiyaga o‘tish, **def** – funksiyani aniqlash, **del** – obyektni yo‘qotish, **elif** – aks holda, agar, **else** – for/else yoki if/elsega qarang, **for** – for sikli, **from –** moduldan bir nechta funksiyani import qilish, **if** – agar, **import** – moduldan import, **is** –xotirani bitta joyida 2 ta obyektni jo‘natsa bo‘ladimi, **lambda** –yashirin funksiyani aniqlash, **not** –mantiqiy inkor amali, **or** –mantiqiy Yoki amali, **while** – while sikli.

Kommentariya. # belgisidan keyin yoziladi va ushbu dastur kodini o‘qiyotgan dasturchi uchun eslatma sifatida xizmat qiladi.

Kommentariy dasturchi uchun foydali bo‘ladi va dastur nima vazifa bajarishini oson tushunishga xizmat qiladi. Unga yechimdagi muhim joylarni va zarur qismlarni yozish mumkin.

Operatorlar va ifodalar

Dasturdagi ko‘p amallar (mantiqiy qatorlar) ifodalardan tashkil topgan. Bunga oddiy misol: 2 + 3. Ifodani operatorlar va operandlarga ajratish mumkin. **Operator**-bu biror bir amalni bajaruvchi va belgilar yordamida yoki zaxiraga olingan so‘zlar yordamida ifodalanadigan birlik. Operatorlar qiymatlar ustida biror bir amalni bajaradi va bu qiymatlar **operandlar** deyiladi. Ushbu holatda 2 va 3 bu operandlar.

Operator	Nomlanishi	Ta'rifi	Misol
+	Qo‘shish	Ikkita obyektning yig‘indisini isoblaydi	3+5 ifoda 8 ga teng; 'a'+'b' ifoda 'ab' ni beradi.
–	Ayirish	Ikkata sonning farqini beradi. Agar birinchi operand mavjud bo‘lmasa, uning qiymati 0 ga teng deb olib ketiladi.	2-5 manfiy qiymat beradi, 50–24 ning qiymati esa 26 ga teng.
*	Ko‘paytirish	Ikkita son ko‘paytmasini beradi yoki satrni ko‘rsatilgan miqdorda takrorlangan yangi satrni qaytaradi.	2*4 ifoda 8 bo‘ladi. 'xa'* 4 ifoda 'xaxaxaxa' ni qaytaradi.
**	Darajaga ko‘tarish	x sonini y darajaga ko‘tarilganda hosil bo‘lgan qiymatni qaytaradi.	3**5 ifoda 243 bo‘ladi (ya'ni 3*3*3*3*3)
/	Bo‘lish	'x' ni 'y' ga bo‘lganda hosil bo‘lgan bo‘linmani qaytaradi.	4 / 3 ifoda 1.33333 ni beradi.

//	Qoldiqsiz bo‘lish	Bo‘lishdan hosil bo‘lgan bo‘linmaning qoldiqsiz butun qismini qaytaradi.	4//3 ifoda 1 ni qaytaradi.
%	Qoldiqlik bo‘lish	Bo‘lishdan hosil bo‘lgan qoldiqni qaytaradi.	10%2 ifoda 2 ni beradi. -25.5%2.25 ifoda 1.5 natija beradi.
<<	Chapga surish	Bit sonni chapga ko‘rsatilgan miqdorda suradi.	2<<2 ifoda 8 ni beradi. Ikkilik sanoq tizimida 2 soni 10 ko‘rinishiga ega bo‘ladi. Chapga 2 bit miqdorida surish 1000 beradi, bu o‘nlik sanoq tizimida 8 ni beradi.
>>	O‘ngga surish	Bit sonni o‘ngga ko‘rsatilgan miqdorda suradi.	11 >> 1 ifoda 5 ni beradi. Ikkilik sanoq sistemasida 11 soni 1011 ko‘rinishida bo‘ladi uni 1 bit o‘ngga sursak 101 hosil bo‘ladi va bu onlik sanoq tizimida 5 ni beradi.
&	'Va' bit operatori	Sonlar ustida 'va' bit operatsiyasini bajaradi.	5 & 3 ifoda 1 ni beradi
|	'Yoki' bit operatori	Sonlar ustida 'yoki' bit operatsiyasini bajaradi.	5 | 3 ifoda 7 ni beradi
^	'shartlik yoki' bit operatori	Sonlar ustida 'shartlik yoki' bit operatsiyasini bajaradi.	5 ^ 3 ifoda 6 ni beradi
~	'Emas' bit operatori	'Emas' bit operatsiyasi x soni uchun -(x+1) ga to‘g‘ri keladi.	~5 ifoda 6 ni beradi.
<	Kichik	x qiymat y qiymatdan kichikligini aniqlaydi. Hamma qiyoslash operatorlari True yoki False qaytaradi. Bu so‘zlardagi katta xarflarga e'tibor bering.	5<3 False qaytaradi 3 < 5 ifoda esa True qaytaradi. Ixtiyoriy bir – biri bilan bog‘langan ifodalar tuzish ham mumkin: 3 < 5 < 7 ifoda True ni qaytaradi
>	Katta	x qiymat y qiymatdan katta ekanligini aniqlaydi.	5>3 ifoda True ni beradi.
<=	Kichik yoki teng	x qiymat y qiymatdan kichik yoki teng ekanligini aniqlaydi.	x = 3; y = 6; x <= y ifoda True ni beradi.
>=	Katta yoki teng	x qiymat y qiymatdan katta yoki teng ekanligini anqlaydi.	x = 4; y = 3; x >= 3 ifoda Trueni beradi.

==	Teng	Obyektlarning tengligini tekshiradi	x=2; y=2; x==y ifoda True qaytaradi. x= 'str'; y= 'stR'; x==y ifoda False qaytaradi. x='str'; y='str'; x==y ifoda True qaytaradi.
!=	Teng emas	Obyektlar teng emasligi to‘g‘riligini tekshiradi.	x=2; y=3; x!=y ifoda True qaytaradi.
Not	Mantiqiy 'emas'	Agar x True bo‘lsa, operator False qaytaradi. Agar x False bo‘lsa operator True qaytaradi.	x = True; not x ifoda False qaytaradi.
And	Mantiqiy 'va'	x and y ifoda False qaytaradi agar x False bo‘lsa. Aks holda y ning qiymatini qaytaradi.	x = False; y = True; x and y ifoda False qaytaradi. Bu holda Python y ning qiymatini tekshirib o‘tirmaydi sababi 'and' operatoridan chapdagi ifoda qismi False ga teng va butun ifoda qiymati boshqa operatorlar qiymatlariga bog‘liqsiz ham False bo‘ladi.
Or	Mantiqiy 'yoki'	x or y agar x True ga teng bo‘lsa True qaytaradi aks xolda y ning qiymatini qaytaradi.	x = True; y = False; x or y ifoda True qiymat qaytaradi.

2.1 jadval. Operatorlar va ularning qo‘llanilishi

Matematik amallar va o‘zlashtirishlarni qisqacha yozish

Ko‘pincha bir o‘zgaruvchi ustida biror bir matematik amal bajarib, natijani shu o‘zgaruvchining o‘ziga o‘zlashtirish zaruriyati tug‘iladi. Ushbu holatda amallarni qisqacha yozish mumkin. Biz a=2; a=a*3 ni quyidagicha yozishimiz mumkin:

```
a = 2; a *= 3
```

Amallar bajarilish ketma-ketligi

4+5*6 ifodada qaysi amal birinchi bajariladi: qo‘shishmi yoki ko‘paytirish?

Matematika fanida ko‘paytirish amali birinchi bajarilishi ko‘rsatilgan. Bundan kelib chiqadiki, ko‘paytirish operatori qo‘shish operatoriga qaraganda katta ustunlikka(prioritet) ega. Quyidagi jadvalda Python dasturlash tili operatorlari prioriteti ko‘rsatilgan. Bunda yuqoridan pastga qarab Python dasturlash tili operatorlari ustunligi oshib boradi. Bu shuni bildiradiki, istalgan ifodada Python dasturi oldin eng quyidagi operatorlarni hisoblaydi va keyin esa yuqoridagilarini hisoblaydi. Amaliyotda

amallarni qavslar bilan aniq ajratish tavsiya etiladi. Bu dastur kodini oson va qulay o‘qishga yordamlashadi.

Operator	**Izoh**
Or	Mantiqiy 'yoki'
And	Mantiqiy 'va'
Not x	Mantiqiy 'emas'
in, not in	Tegishlilikni tekshirish
is, is not	Bir xillikni tekshirish
<, <=, >, >=, !=, ==	Taqqoslash
|	'yoki' bit operatori
^	'shartlik yoki' bit operatori
&	'va' bit operatori
<<, >>	Surilishlar
+, –	Qo‘shish va ayirish
*, /, //, %	Ko‘paytirish, bo‘lish, qoldiqsiz bo‘lish va qoldiqlik bo‘lish
+x, -x	Musbat va manfiy
~x	'emas' bit operatori
**	Darajaga ko‘tarish
x.attribute	Atributga link
x[index]	Indeks bo‘yicha murojat
x[index1:index2]	Kesib olish
f(argumentlar …)	Funksiyani chaqirish
(ifoda, …)	Kortej (Связка или кортеж)
[ifoda, …]	Ro‘yxat (Список)
{kalit:qiymat, …}	Lug‘at (Словарь)

2.2 jadval. Python dasturlash tili operatorlarining prioritetlari.

Yuqoridagi jadvalda bir xil ustunlikka ega bo‘lgan operatorlar bir qatorda joylashgan. Masalan '+' va '-'.

Hisoblash tartibini o‘zgartirish

Ifodalarni osongina o‘qish uchun qavslarni ishlatish mumkin. Masalan, 2+(3*4) ni tushunish oson 2+3*4 ifodadan ko‘ra. Qavslarni o‘ylab ishlatish lozim. Ortiqcha qavslarni ishlatishdan saqlaning. Masalan: (2+(3*4)).

Qavslarni ishlatishni ya'na bir ustunligi hisoblash tartibini o‘zgartirish imkonini yaratadi. Masalan, qo‘shish amalini ko‘paytirish amalidan oldin bajarish kerak bo‘lsa, quyidagicha yozish mumkin:

(2 + 3) * 4.

Masala. Son o‘qida 2 nuqta orasidagi masofani aniqlang.

```
x1 = float(input("x1="))
x2 = float(input("x2="))
m = abs(x2 - x1)
print("masofa", m)
```

```
>>>
= RESTART: C
y
x1=9
x2=3
masofa 6.0
```

Masala. Berilgan A va V sonlarining qiymatlarini o‘zaro almashtiring. A va V ning yangi qiymati ekranga chiqarilsin.

```
A = int(input("A="))
V = int(input("V="))

print("A=", V)
print("V=", A)
```

```
>>>
= RESTAF
y
A=9
V=8
A= 8
V= 9
>>>
```

Masala. Kubning a tomoni berilgan. Uning hajmi va to‘la sirtini aniqlang.

```
a = int(input("a= "))
V=a*a*a
s = a * 6
print("V =", V)
print("S =", s)
```

```
>>>
= RESTART: C:/U
y
a= 9
V = 729
S = 54
>>>
```

Masala. Paralelepepidning tomonlari a, b, s berilgan. Uning hajmini va to‘la sirtini hisoblang.

```
a = int(input("a= "))
b = int(input("b= "))
c = int(input("c= "))
V=a*b*c
s = 2*(a*b+ b*c+a*c )
print("V =", V)
print("S =", s)
```

```
>>>
= RESTART: C:/Us
y
a= 9
b= 8
c= 5
V = 360
S = 314
>>>
```

Masala. Doiraning radiusi berilgan. Uning uzunligi va yuzasi aniqlang

```
pi = 3.15
R = float(input("R= "))
L = 2*pi*R
S= pi*R*R
print("L=", L)
print("S=", S)
```

```
>>>
=== RESTART: C:/Users/
R= 9
L= 56.699999999999996
S= 255.14999999999998
>>>
```

Masala. Ikkita son berilgan. Ularning o‘rta arifmetigini aniqlang.

```
a = int(input("a= "))
b = int(input("b= "))
c=(a+b)/2
print("O'rta arifmetik=", c)
```

```
>>>
==== RESTART: C:/U
a= 9
b= 7
O'rta arifmetik= 8
>>>
```

Masala. Ikkita manfiy bo‘lmagan son berilgan. Ularning o‘rta geometrigini aniqlang.

```
import math
a = int(input("a= "))
b = int(input("b= "))
c = math.sqrt(a * b)
print("O'rta geometrik:", c)
```

```
>>>
==== RESTART: C:/Users
a= 9
b= 4
O'rta geometrik: 6.0
>>>
```

Masala. Aylananing yuzasi berilgan. Uning diametri va radiusini aniqlang.

```
import math
S = int(input("S= "))
d = math.sqrt(4 * S / math.pi)
R = d / 2
print("d=", d)
print("R=", R)
```

```
>>>
=== RESTART: C:/Users/
S= 63
d= 8.95623198216277
R= 4.478115991081385
>>>
```

3-MAVZU. PYTHON DASTURLASH TILIDA INTERAKTIV REJIM, BIRINCHI DASTUR, KIRITISH VA CHIQARISH OPERATORLARI

Reja:

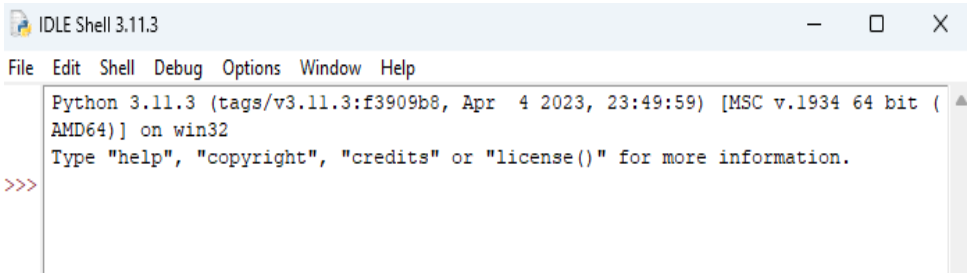
1. Python dasturlash tilining IDLE rejimida interaktiv dastur yaratish;
2. Python dasturlash tilida ma’lumotlar turlari;
3. Python dasturlash tilida faylli dastur yaratish;

Tayanch so‘zlar. IDLE rejimi, interaktiv, faylli dastur, ENTER, restart, int, float, str.

Python dasturlash tilining IDLE rejimida interaktiv dastur yaratish va o‘zgaruvchi turlaridan foydalanish

Python dasturlash tili shaxsiy kompyuterga o‘rnatilgandan so‘ng IDLE rejimi yuklanadi. IDLE rejimi bu interaktiv rejim hisoblanadi, interaktiv rejimning qulayligi qisqa jarayonlarni dasturlashga juda qo‘l keladi. Bunda boshqa dasturlash tillari kabi dasturning umumiy tuzilishini to‘liq keltirish shart emas. O‘zgaruvchilar tabaqalashtirilgan holda e’lon qilish shartlari keltirilmaydi, xotira bo‘yicha muammolar e’tiborga olinmaydi.

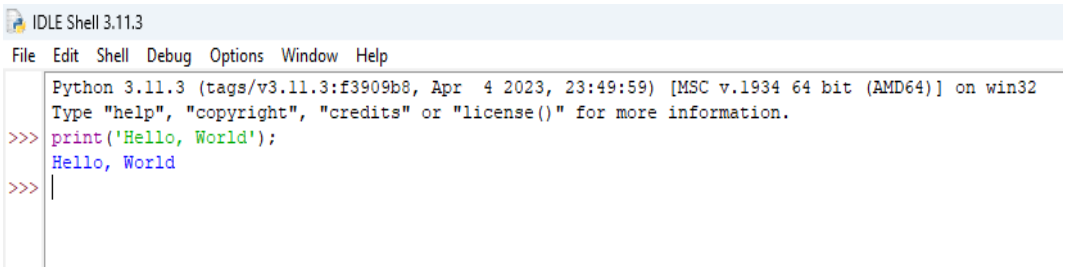
Python dasturlash tili IDLE rejimi yuklangandan so‘ng quyidagi oyna hosil bo‘ladi.



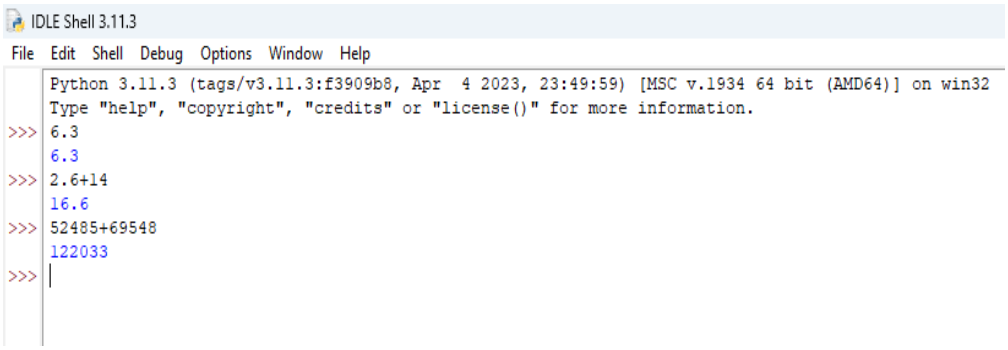
3.1-rasm. Python dasturlash tili IDLE rejimining asosiy oynasi.

Python dasturlash tili IDLE rejimida birinchi dastur doimiy an’anaga muvofiq HELLO WORLD so‘zini ekranga chiqarishni qarab o‘tamiz.

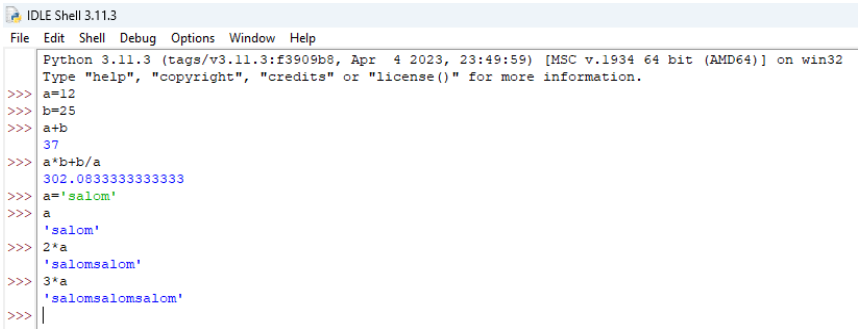
Satrli ma’lumotlar bittalik qo‘shtirnoq ichiga olib yoziladi.



Bunda >>> belgidan so‘ng sonlarni kiritish va hisoblash jarayonlarini to‘g‘ridan to‘g‘ri amalga oshirish mumkin. Har bir son yoki amal kiritilgandan so‘ng ENTER tugmasi bosiladi va natijaga ega bo‘linadi. Python tilida interaktiv rejim quyidagicha ishlatiladi.



Bu jarayon bizga kalkulyator vazifasini ham bajaradi, kalkulyatordagi xotira muammosi bu yerda mavjud emas. Python dasturlash tili IDLE rejimida o‘zgaruvchilar bilan ishlash quyidagicha amalga oshiriladi:

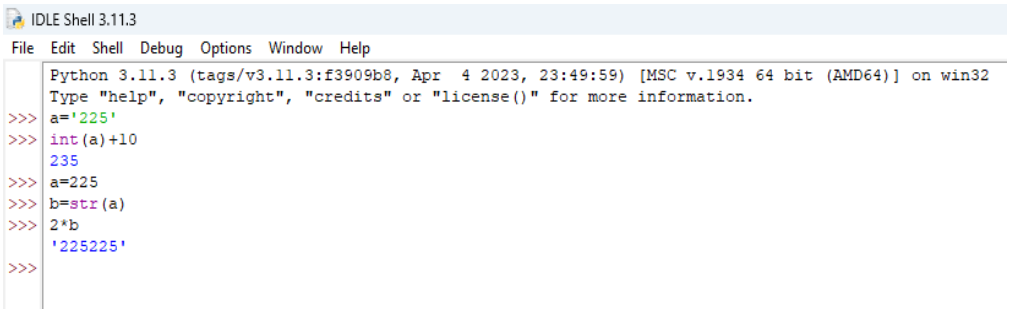


Yuqoridagi dasturga e’tibor bersak, bunda o‘zgaruvchilar e’lon qilinmaydi, interpretator qiymatga qarab turlarni aniqlaydi. Python dasturlash tilida buyruqlar boshqa dasturlash tillari kabi ; bilan tugatilish shart emas. Satrli ma’lumotlarga ko‘paytirish amali juda keng imkoniyatda ishlatilishi mumkin. Ixtiyoriy sonni satr ko‘rinishga satrni esa son ko‘rinishida foydalanish mumkin.

Int(satr) – bu funksiya satrni butun songa aylantiradi.

Float(satr) – bu funksiya satrni haqiqiy songa aylantiradi.

Str(son) – bu funksiya sonni satrga aylantiradi.



Haqiqiy sonlar faqat nuqta bilan yoziladi, vergul esa sonlarni bir biridan ajratish ucun xizmat qiladi. Python dasturlash tili IDLE rejimining qulayligi dasturlashni o‘rganayotganda yoki masala kodining ma’lum bir qismini sinab ko‘rish vaqtida juda qulaydir. Agar boshqa kompilyatsiya qilinadigan dasturlash tilda ishlasangiz, avval kodni asl dasturlash tilida yozishingiz, keyin kompilyatsiya qilishingiz kerak bo‘lgan faylni ishga tushirishingiz kerak bo‘ladi.

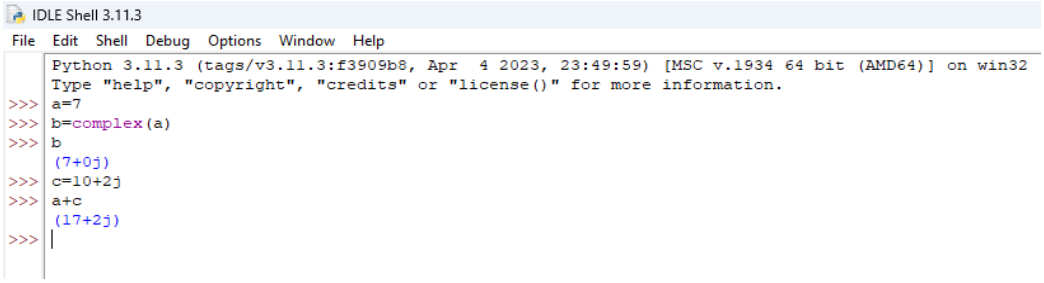
Python dasturlash tilida sonlarni uch xil turlari aniqlangan:

- Butun son
- Haqiqiy
- Kompleks

Complex(son) – bu funksiya sonni komleks songa aylantiradi

Python dasturlash tilining boshqa dasturlash tillaridan ustunligi kompleks sonlar ustida to‘g‘ridan to‘gri amal bajarish imkoniyati mavjudligi.

Python dasturlash tilida butun va haqiqiy sonlardan foydalanish tajribalarini yuqoridagi misollarda keltirib o‘tdik, complex sonlardan python dasturlash tilida quyidagicha foydalanamiz.



Python dasturlash tilida ma'lumotlar turlari

Dasturlashda ma'lumotlar turi muhim tushunchadir. O‘zgaruvchilar har xil turdagi ma'lumotlarni saqlashi mumkin va har xil turlar har xil ishlarni bajarishi mumkin.

Python sukut bo‘yicha quyidagi toifalarga o‘rnatilgan ma'lumotlar turlariga ega:

Ma'lumotlar turini olish. Funksiyadan foydalanib, har qanday ob'ektning

Matn turi:	**str**
Raqamli turlar:	**int, float, complex**
Tartib turlari:	**list, tuple, range**
Xaritalash turi:	**dict**
To‘plam turlari:	**set,frozenset**
Mantiqiy turi:	**bool**
Ikkilik turlari:	**bytes, bytearray, memoryview**
Yo‘q turi:	**NoneType**

ma'lumotlar turini olishingiz mumkin **type():**

Misol. x o‘zgaruvchisining ma'lumotlar turini chop eting:

```
x = 5
print(type(x))
```

Ma'lumotlar turini sozlash. Pythonda ma'lumotlar turi o‘zgaruvchiga qiymat berganingizda o‘rnatiladi:

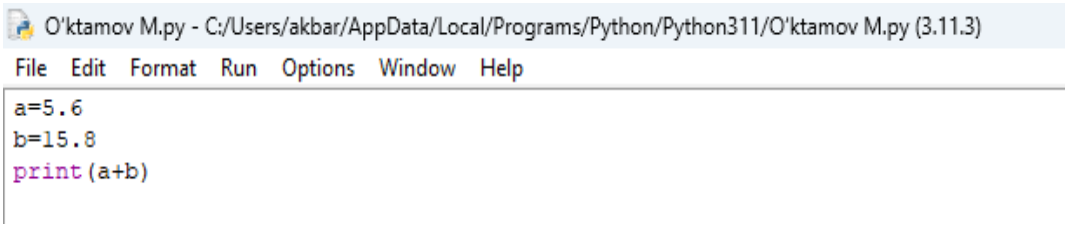
Example	**Data Type**
x = "Hello World"	str
x = 20	int
x = 20.5	float
x = 1j	complex
x = ["apple", "banana", "cherry"]	list
x = ("apple", "banana", "cherry")	tuple
x = range(6)	range
x = {"name" : "John", "age" : 36}	dict
x = {"apple", "banana", "cherry"}	set
x = frozenset({"apple", "banana", "cherry"})	frozenset
x = True	bool
x = b"Hello"	bytes
x = bytearray(5)	bytearray
x = memoryview(bytes(5))	memoryview
x = None	NoneType

Python dasturlash tilida faylli dastur yaratish, kiritish va chiqarish operatorlari

Python dasturlash tilida ko‘pgina hollarda, dasturchi masala kodini faylga saqlaydi va natijani fayl kod orqali amalga oshiradi. Bu jarayon boshqa dasturlash

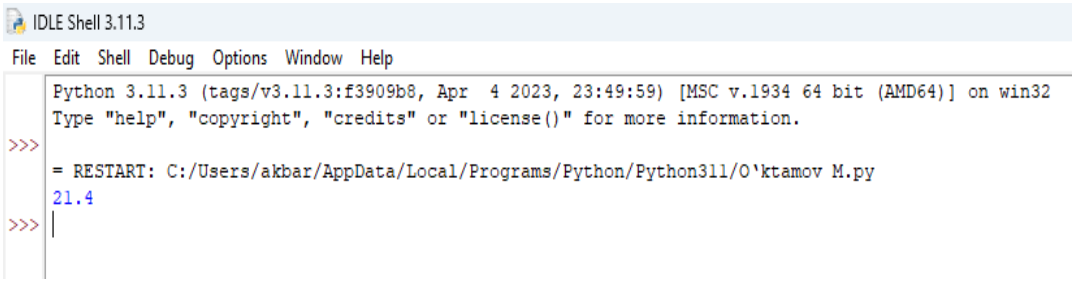
tillari kabi alohida fayl yaratish orqali dastur tuziladi va yaratilgan dastur RUN tugmasi orqali ishga tushiriladi. Bu jarayon skript yozish deb nomlanadi. Python dasturlash tilida yaratilgan fayl **.py** kengaytmaga ega bo‘ladi.

Python dasturlash tilida skript yozish uchun IDLE interaktiv rejimida **File → New File** (yoki **<Ctrl> + N** tugmachalarini bosing) ni tanlash orqali amalga oshiriladi. Yaratilgan faylga dastur tuzib **RUN** menyusi tarkibidan **Run Module F5** komandasi tanlanadi, natijada dastur natijasi interaktiv rejim oynasida aks etadi. Quyidagi dasturga e’tibor bering.



1.2.2-rasm. Faylli dastur oynasi

Dastur natijasi quyidagicha bo‘ladi.



Skript dastur yaratish jarayonida ma’lumotlarni kiritish (o‘qish) va chiqarish (yozish) operatorlari ishlatiladi. Bunda kiritish operatori ma’lumotlarni faqat satr ko‘rinishida qabul qiladi. Sonli ma’lumotlarni qayta ishlash uchun yuqoridagi int(); yoki float(); funksiyasi yordamida amalga oshiriladi. Kiritish operatori o‘zgaruvchiga birlashtiriladi, kiritish operatorining umumiy ko‘rinishi quyidagicha.

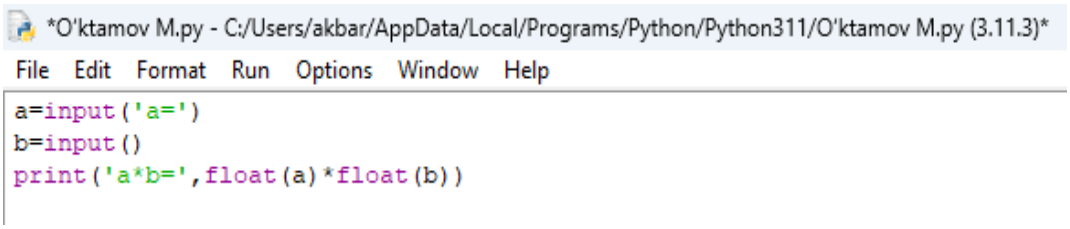
o‘zgaruvchi = input() yoki **o‘zgaruvchi = input(‘izoh’)**

Kiritish operatorining umumiy ko‘rinishi quyidagicha.

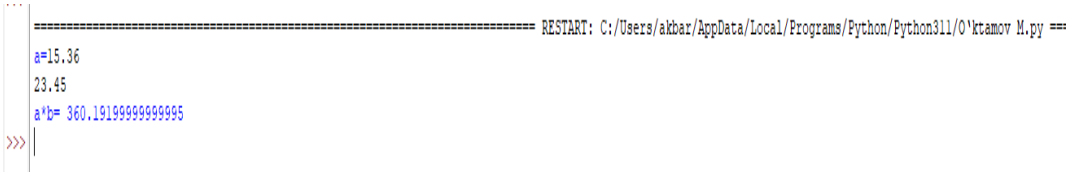
Print(‘izoh’, o‘zgaruvchi)

Kiritish va chiqarish operatorlarini yozilishi quyidagi dastur tarkibida keltirilgan.

Misol. Ikkita *a* haqiqiy va *b* butun son berilgan bu sonlarni ko‘paytmasini ekranga chiqaring.



Dastur natijasi



Yuqoridagi dasturda **a=input('a=');** izohni chiqargan holda a ga qiymat qabul qiladi, **b=input()** esa izohsiz b ga qiymat qabul qiladi. **Input()** funksiyasi tarkibida bir vaqtni o‘zida izohli va izohsiz kiritishni ishlatish mumkin. Kiritish operatori klaviaturadan kiritilgan ma’lumotlarni o‘qiydi va o‘zgaruvchan nomga yozadi. Chiqarish operatori esa o‘zgaruvchidagi ma’lumotni ekranga chop etadi.

4-MAVZU. PYTHON DASTURLASH TILI TARKIBIDAGI ARIFMETIK AMALLAR VA MANTIQIY AMALLAR

Reja:

1. Python dasturlash tili tarkibidagi arifmetik amallar
2. Ta’minlash operatori
3. Python dasturlash tili tarkibidagi mantiqiy amallar

Tayanch so‘zlar: *unar, binar, amal, round, div, mod, true, false.*

Python dasturlash tili tarkibida dastur tuziladigan vaqtda albatta matematik ifodalar, amal ishoralar va mantiqiy amallar ishtirok etishi mumkin. Dastur dasturlash tillarida amallar matematikadan yozilishi bilan farq qiladi. Dasturlash asoslarida amallarni ikki turga ajratamiz:

-arifmetik amallar;

-mantiqiy amallar;

Python dasturlash tilida arifmetik amallar

Berilganlarni qayta ishlash uchun dasturlash tillarida amallarning juda keng majmuasi aniqlangan. Amal - bu qandaydir harakat bo‘lib, u bitta (unar) yoki ikkita (binar) operandlar ustida bajariladi, hisob natijasi uning qaytaruvchi qiymati hisoblanadi.

Tayanch arifmetik amallar dasturlash tilida quyidagicha yoziladi.

Amal nomi	**Amal belgisi**		**Misol**
Qo‘shish	+	x+y	print(7+5) # 12
Ayirish	-	x-y	print(7-5) # 2
Ko‘paytirish	*	x*y	print(7*5) # 35
Bo‘lish	/	x/y	print(7/5) # 1.4
Bo‘linmaning butun qiymatini hisoblash	//	x//y	print(7//4) # 1
Bo‘linmaning qoldig‘ini hisoblash	%	x%y	print(7%5) # 2
Darajaga ko‘tarish x^y	**	x**y	print(5**2) # 25

Agar ifodada bir nechta arifmetik amallar ketma-ket kelgan bo‘lsa, ular prioriteti (ustunligi) bo‘yicha bajariladi. Dastlab, yuqori prioritetga ega bo‘lgan amallar bajariladi. Amallarning prioriteti kamayish tartibida quyidagi jadvalda ifodalangan:

Amallar	Yo‘nalish
**	Chapdan-o‘nga
*, /, //, %	Chapdan-o‘nga
+, -	Chapdan-o‘nga

Misol sifatida quyidagi ifodani qaraymiz:

```
son = 12//7 + 2 ** 5  *  3 - 4
print(son)  # 93
```

Bu yerda dastlab eng yuqori prioritetga ega bo‘lgan amal – darajaga ko‘tarish amali bajariladi (2 ** 5 =32). Keyin ko‘paytma (32 * 3 = 96), butun qismli bo‘lish (12 // 7 = 1), qo‘shish (1 + 96 = 97) va ayirish (97 – 4 = 93) amallari bajariladi. Ifoda bajarilishi natijasida 93 soni konsol ekraniga chiqariladi.

Amallarni qavsga olish orqali ularning bajarilish ketma-ketligini o‘zimiz xohlagan tartibga keltirib olishimiz ham mumkin. Masalan, yuqoridagi ifodani quyidagicha qayta yozamiz:

```
son = 12//7+2**5*(3–4)
print(son)  # -31
```

Natijada konsol ekraniga -31 soni chiqariladi.

Shuni ta’kidlash kerakki, arifmetik amallar butun sonlar uchun qanday tartibda bajarilsa, suzuvchan nuqtali haqiqiy sonlar uchun ham xuddi shunday bo‘ladi. Agarda ifodada loaqal bitta haqiqiy son ishtirok qilsa natija haqiqiy turda ifodalanadi.

Yuqoridagi barcha arifmetik amallarni o‘zlashtirish amali (=) bilan birgalikda (arifmetik amal va undan keyin “=” belgisi ketma-ket yoziladi) ishlatish mumkin. Masalan: +=, -=, *=, /=, //=, %=, **=. Bunday hollarda ifodaning o‘ng tomonidagi barcha amallar hisoblanib, chiqqan natija chap tomondagi o‘zgaruvchi natijasi bilan mos arifmetik amal bajariladi va natija yana chap tomondagi o‘zgaruvchiga yuklanadi. Masalan:

```
son = 4
son += 5  # son = son + 5 amaliga teng kuchli, son=9 bo‘ladi
print(son)  # 9
son -= 1 print(son)  # 8
son *= 4 print(son)  # 32
son //= 2 print(son)  # 16
son **= 2 print(son)  # 256
```

Sonlarning turli sanoq sistemalarda tasvirlanishi

Aksariyat hollarda, sonli o‘zgaruvchilarni aniqlashda ularga qiymat o‘nlik sanoq sistemasida beriladi. O‘nlik sanoq sistemalardan tashqari Python dasturlash tilida sonlarni ikkilik, sakkizlik va o‘n oltilik sanoq sistemalarida ham ifodalash mumkin. Sonni ikkilik sanoq sistemasida ifodalash uchun qiymat oldiga ***0b***, sakkizlikda ***0o*** va o‘n oltilikda ***0x*** belgilar jufligi qo‘yiladi. Masalan:

```
son1 = 5
son2 = 0b110  # 6
son3 = 0o11  # 9
son4 = 0x1a  # 26
print(son1,son2,son3,son4)  # 5 6 9 26
print(son1 +son3)  # 14
```

Turli xil sanoq sistemalarda ifodalangan sonlar ustida bajarilgan amallar natijasi o‘nlik sanoq sonlarda ifodalaniladi. Shuning uchun yuqoridagi dasturda print(son1+son3) ifodaning natijasi 14 soni o‘nlik sanoq sistemasida ekranga chiqariladi.

Bundan tashqari ixtiyoriy butun sonni ikkilik, sakkizlik va o‘n oltilik sanoq sistemalarida ifodalash mumkin. Masalan, quyidagi dasturda 15 sonining turli sanoq sistemalardagi ifodalanishi tasvirlangan:

```
son = 15
print("{0}".format(son))  # 15
print("{0:0b}".format(son))  # 1111
print("{0:07b}".format(son))  # 0001111
print("{0:0o}".format(son))  # 17
print("{0:0x}".format(son))  # f
```

Yuqoridagi dasturning 4-satrida keltirilgan {0:07b} ifodadagi 7 soni yozuvida nechta raqam bo‘lishi kerakligini ifodalaydi. Shuning uchun 0001111 natija hosil qilingan. Sonni ifodalovchi qiymat, ko‘rsatilgan uzunlikda bo‘lmasa, u holda qiymat

old qismi 0 raqami bilan to‘ldiriladi (yuqorida 1111 qiymat old qismiga 000 raqamlar ketma-ketligi qo‘yilgan).

Dasturda amallarni qisqa holda qo‘llash

Amal belgisi	Ifodaning qisqa yozilishi	Ifodaning to‘liq yozilishi	Misol x=4
+=	x+=y	x=x+y	x+=1 # 5
-=	x-=y	x=x-y	x–=2 # 2
*=	x*=y	x=x*y	x*=2 # 8
/=	x/=y	x=x/y	x/=2 # 2
//=	x//=y	x=x//y	x//=2 # 2
%=	x%=y	x=x%y	x%=2 # 0
**=	x**=y	x=x**y	x**=2 # 16

Ifodalar amallarning bajarilish tartibini bildiradi. Ifodalar o‘zgaruvchi, doimiy, qavs va amallardan tashkil topadi.

Matematik ifoda	Ifodaning dasturlash tilida yozilishi
$y = \frac{x^2 + x - 3}{x^2 + 5x} + \frac{1}{x}$	y=(x**2+x–3)/(x**2+5*x)+1/x

Misol. To‘rt xonali son berilgan. Ushbu son birinchi va oxirgi xonasi raqamining ko‘paytmasini topish dasturini tuzing.

```
>>> print(4 xonali son kiriting')
4-xonali son kiritish uchun
>>> x=int(input())
4568
>>> a= x// 1000
>>> b= x % 10
>>> c= a * b
>>> print(c, '-birinchi va oxirgi raqa
ko’paytmasi, x)
32- birinchi va oxirgi raqami
ko’paytmasi 4568
```

x=4568
a=4568//1000=4
b=4568%10=8
c=4*8=32

Python dasturlash tilida shart ifodalari

Shart ifodalarini bir qator amallar taqdim qiladi. Ushbu amallarning barchasi ikkita operand qabul qiladi va natija sifatida *boolean* turidagi mantiqiy qiymat qaytaradi. Faqatgina ikkita mantiqiy qiymat mavjud, ular *True*-ifoda rost *False*-ifoda yolg‘on qiymatlardir.

Taqqoslash amallari. Eng sodda shart ifodalariga taqqoslash amallari misol bo‘lib, ular ikki qiymatni taqqoslash uchun ishlatiladi. Python quyidagi taqqoslash amallarini qo‘llab-quvvatlaydi: [5]

== - agar ikki operand teng bo‘lsa *True*, aks holda *False* qiymat qaytaradi;

!= - agar ikki operand teng bo‘lmasa *True*, aks holda *False* qiymat qaytaradi;

> (dan katta) – agar birinchi operand ikkinchisidan katta bo‘lsa *True*, aks holda *False* qiymat qaytaradi;

< (dan kichik) – agar birinchi operand ikkinchisida kichik bo‘lsa True, aks holda False qiymat qaytaradi;

>= (dan katta yoki teng) – agar birinchi operand ikkinchisidan katta yoki teng bo‘lsa *True*, aks holda *False* qiymat qaytaradi;

<= (dan kichik yoki teng) – agar birinchi operand ikkinchisidan kichik yoki teng bo‘lsa *True*, aks holda *False* qiymat qaytaradi;

Python dasturlash tilida amallarni funksiyalar orqali ham amalga oshirish imkoniyati mavjud.

Python dasturlash tilida ifodasi	**Izoh**
abs(x)	Modul
round(x)	Yaxlitlash
round(x, n)	n – xonagacha yaxlitlash
pow(x, y)	Darajaga ko‘tarish
divmod(x, y)	Butun va qoldiqli bo‘lish

Round(x) funksiya sonning butun qismigacha yaxlitlaydi, *round(x,n)* funksiyasi sonning *n* – xonasigacha yaxlitlaydi, *pow(x,y)=x**y* ga teng kuchli va *a,b=divmod(x,y)* funksiyasi bir vaqtda *x* ni *y* ga bo‘lib butun va qoldiq qismlarini oladi. Funksiyali amallarni ishlash jarayoni tushunarli bo‘lishi uchun, ularni interaktiv rejimda sinab ko‘ramiz.

Misol. Funksiyali amallarni bajarilishi

```
>>> y=-5
>>> abs(y)
5
```

[5] www.fayllar.org/pythonga_kirish_page=6

```
>>> x=12.32568
>>> round(x)
12
>>> y=13.652
>>> round(y)
14
>>> round(13.652,2)
13.65
>>> a=2
>>> b=3
>>> pow(a,b)
8
>>> a=5
>>> b=2
>>> x,y=divmod(a,b)
>>> x
2
>>> y
1
```

Ta’minlash operatori

Ma’lum bir ifodaning natijasi biror o‘zgaruvchiga ta’minlash uchun Python dasturlash tilida “=” belgisi bilan ishlatiladi va uning umumiy ko‘rinishi quyidagicha:

<o‘zgaruvchi>=<ifoda>;

Python dasturlash tilida ta’minlash operatori amallar yordamida ham ishlatiladi.

Qo‘shish qiymat berish bilan	(+=);
Ayirish qiymat berish bilan	(-=);
Ko‘paytirish qiymat berish bilan	(*=);
Bo‘lish qiymat berish bilan	(/=);
Bo‘lish qoldig‘ini olish qiymat berish bilan	(%=)

Yuqoridagi holatlarning umumiy ko‘rinishi quyidagicha:

<o‘zgaruvchi><amal>=<ifoda>;

s+=x; ning ma’nosi *s=s+x;*

Ta’minlash operatorini ishlash jarayoni tushunarli bo‘lishi uchun, ularni interaktiv rejimda sinab ko‘ramiz.

Misol. Ta’minlash operatorida foydalanish

```
>>> x=5
```

```
>>> y=2
>>> x*=y
>>> x
10
>>> x/=2
>>> x
5.0
>>> x%=y
>>> x
1.0
```

Python dasturlash tilida *s=+x* amali *s=x* amaliga teng kuchli hisoblanadi, *s=x+* va *s=x++* amallari python dasturlash tilida aniqlanmagan.

Mantiqiy amallar

Mantiqiy turdagi o‘zgaruvchi xotiradan 1 bayt joy egallaydi va 0 (false, yolg‘on) yoki (true, rost) qiymatni qabul qiladi. Mantiqiy tur o‘zgaruvchilar qiymatlar o‘rtasidagi munosabatlarni ifodalaydigan mulohazalarni rost (true) yoki yolg‘on (false) ekanligi tavsifida qo‘llaniladi va ular qabul qiladigan qiymatlar matematik mantiq qonuniyatlariga asoslanadi. Mantiqiy o‘zgaruvchini quyidagicha faollashtiramiz.

<o‘zgaruvchi> = qiymat

Bu yerda qiymat *True* yoki *False* bo‘lishi mumkin.

Taqqoslash amallari python dasturlash tilida quyidagi jadvalda berilgan ko‘rinishida bajariladi.

Nomi	Pythonda ifodalanishi	Misol	Natija
Tenglik	==	12==50, 5==5	False, True
Teng emas	!=	100!=50, 50!=50	True, False
Katta	>	100>50, 50>50	True, False
Katta yoki teng	>=	100>=50, 50>=50	True, True
Kichik	<	100<50, 50<50	False, False
Kichik yoki teng	<=	100<=50, 50<=50	False, True

Python dasturlash tilida uchta mantiqiy bog‘lash mulohazalari mavjud, mantiqiy mulohazalar ustida amallar quyidagicha:

- **inkor;**
- **konyunksiya;**

- dizyunksiya;

1) **inkor** – A mulohazani inkori deganda A rost bo‘lganda yolg‘on yoki yolg‘on bo‘lganda rost qiymat qabul qiluvchi mulohazaga aytiladi. Python tilida inkor – *not A* bilan beriladi. Masalan, A mulohaza inkori *not A* ko‘rinishida yoziladi;

2) **konyunksiya** - ikkita A va B mulohazalar konyunksiyasi yoki mantiqiy ko‘paytmasi «A and B» ko‘rinishga ega. Bu mulohaza faqat A va B mulohazalar rost bo‘lgandagina rost bo‘ladi, aks holda yolg‘on bo‘ladi (odatda «and» amali «va» deb o‘qiladi).

3) **dizyunksiya** – ikkita A va B mulohazalar dizyunksiyasi yoki mantiqiy yig‘indisi «A or B» ko‘rinishda yoziladi. Bu mulohaza rost bo‘lishi uchun A yoki B mulohazalardan biri rost bo‘lishi yetarli. Odatda «or» amali «yoki» deb o‘qiladi.

Amal	Tavsifi	Izoh
and	Mantiqiy ko‘paytirish	Murakkab ifodada qatnashgan barcha qism ifodalar True bo‘lsa, ifodaning yakuniy qiymati True, aks holda False qiymatni qaytaradi.
or	Mantiqiy qo‘shish	Murakkab ifodada qatnashgan barcha qism ifodalardan kamida bittasi True bo‘lsa, ifodaning yakuniy qiymati True, aks holda False qiymatni qaytaradi.
not	Mantiqiy inkor	Ifodaning qiymati True bo‘lsa, False qiymatni qaytaradi va aksincha.

Misol: Lolaning tug‘ilgan kuni – 15 mart. Ushbu dastur mantiqiy amallar yordamida tug‘ilgan kunni yoki aksincha ekanligini tekshiradi.

```
>>> day = 15
>>> month = 3
>>> day == 15 and month == 3
True
>>> day = 15
>>> month = 3
>>> not (day == 15 and month == 3)
False
```

and amali ikkala shart ham rost ekanligini tekshiradi.

not amali natijani uning inkoriga almashtiradi. Tug‘ilgan kundan boshqa barcha kunlar uchun natija – False.

```
>>> day=15
>>> month=3
>>>(day==15 and month==3) or (day==1 and month==1)
True
```

or amali qavs ichidagi shartlardan kamida bittasi **True** bo‘lsa, **True**ni qaytaradi.

Taqqoslash amallari yordamida satrlarni ham taqqoslash mumkin.

```
>>> name='Book shop'
>>> name == 'Book shop'
True
>>> name == 'book shop'
False
>>> name == 'Book shop '
False
```

Satr to‘liq mos kelganligi uchun rost qiymatni qaytaradi.

Birinchi harfi kichik harfda yozilgani uchun yolg‘on qiymatni qaytaradi.

Satr oxirida ortiqcha probellar qo‘yilgani uchun yolg‘on qiymatni qaytaradi.

Mantiqiy amallarni bajarilish jadvali quyidagicha.

A	B	Not A	Not B	A and B	A or B
0	0	1	1	0	0
0	1	1	0	0	1
1	0	0	1	0	1
1	1	0	0	1	1

Taqqoslash va mantiqiy amallarni ishlash jarayoni tushunarli bo‘lishi uchun, ularni interaktiv rejimda sinab ko‘ramiz.

Misol. Taqqoslash amallaridan foydalanish.

```
>>> a=12
>>> b=-7
>>> a>b
True
>>> z=a<b
>>> z
False
>>> not z
True
>>> z=a==b
>>>  z
False
>>>  a!=b
True
>>>  a>=b
True
>>>  a<=b
False
>>>
```

Python dasturlash tilida mantiqiy amallardan foydalanishda albatta quyidagilarga e’tibor bering:

- O‘zgaruvchiga boshlang‘ich qiymatlarni berishda = belgisi oldidan va orqasidan bitta bo‘sh joy(probel) quying;

- O‘zgaruvchiga boshlang‘ich qiymatlarni berishda True va False kabi yozilish kerak ya’ni birinchi harfi katta harflarda.

Masala. Kvadratning tomoni berilgan. Uning perimetrini aniqlang.

```
a = float(input("a= "))
p = 4 * a
print("P =", p)
```

```
>>>
= RESTART: C:/
y
a= 6
P = 24.0
>>>
```

Masala. Kvadratning tomoni a berilgan. Uning yuzini hisoblang.

```
a = float(input("a= "))
s = a * a
print("S =", s)
```

```
>>>
= RESTART:
y
a= 5
S = 25.0
>>>
```

Masala. To‘g‘ri to‘rtburchakning tomonlari berilgan. Uning yuzasi va perimetrini hisoblang.

```
a = int(input("a= "))
b = int(input("a= "))
p=2*(a+b)
s = a * b
print("P =", p)
print("S =", s)
```

```
>>>
= RESTART: C:/
y
a= 9
a= 7
P = 32
S = 63
>>>
```

Masala. Aylananing diametri berilgan. Uning uzunligini hisoblang.

```
pi = 3.15
d = float(input("d= "))
L = pi * d
print("L=", L)
```

```
>>>
=== RESTART:
d= 7
L= 22.05
>>>
```

Masala. A butun soni berilgan. A ni rostlikka tekshiring: “A soni musbat".

```
A = int(input("a= "))
r= A > 0
print("A soni musbat:", r)
```

```
>>>
=== RESTART: C:/Users
a= -9
A soni musbat: False
>>>
```

Masala. A butun soni berilgan. A ni rostlikka tekshiring: “A soni toq son”.

```
a=int(input('a='))
print(bool(a%2==1))
```

```
File Edit Shell Debug Opti
Python 3.12.0 (tag
AMD64)] on win32
Type "help", "copy
>>
= RESTART: C:/User
a=83
True
```

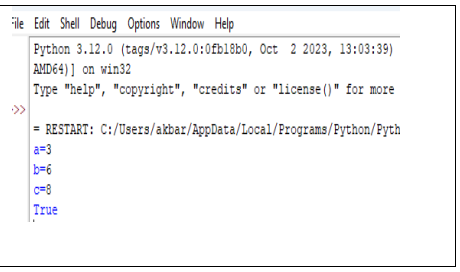
Masala. 2 ta A va B butun sonlari berilgan. Ularni rostlikka tekshiring: “A>2 va B<=3”.

```
a=int(input('a='))
b=int(input('b='))
c=bool(a>2 and b<=3)
print(c)
```

```
File Edit Shell Debug Options Window Help
Python 3.12.0 (tags/v3.12.0:0fb18b0, Oct
AMD64)] on win32
Type "help", "copyright", "credits" or "li
>>
= RESTART: C:/Users/akbar/AppData/Local/Pr
a=9
b=7
False
```

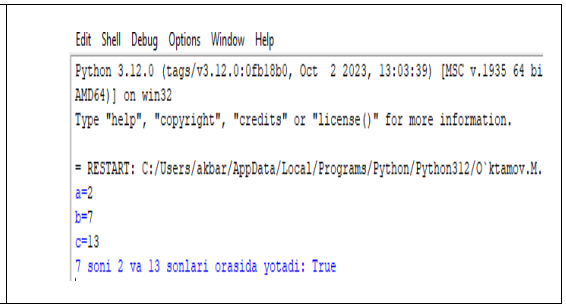
Masala Uchta A, B, C butun sonlar berilgan. Ularni rostlikka tekshiring: “A<=B<=C”

```
a=int(input('a='))
b=int(input('b='))
c=int(input('c='))
hisoblash=bool(a<=b and b<=c)
print(hisoblash)
```



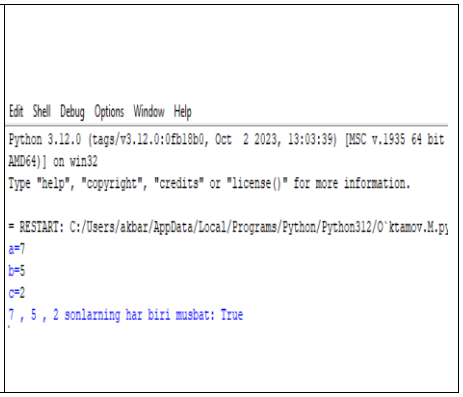
Masala. Uchta A, B, C butun sonlar berilgan. Ularni rostlikka tekshiring: “B soni A va C sonlari orasida yotadi”.

```
a=int(input('a='))
b=int(input('b='))
c=int(input('c='))
natija=bool(a<b and b<c)
print(b,'soni',a,'va',c,'sonlari
orasida yotadi:',natija)
```



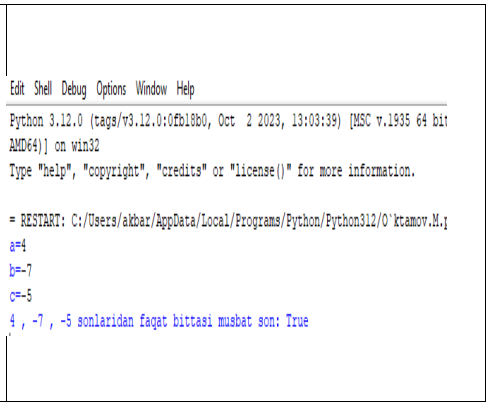
Masala Uchta A, B, C butun sonlar berilgan. Ularni rostlikka tekshiring: “A, B, C sonlarning har biri musbat”.

```
a=int(input('a='))
b=int(input('b='))
c=int(input('c='))
musbat=bool(a>0 and b>0 and c>0)
print(a,',',b,',',c,'sonlarning har biri
musbat:',musbat)
```



Masala Uchta A, B, C butun sonlar berilgan. Ularni rostlikka tekshiring: “A, B, C sonlaridan faqat bittasi musbat son”.

```
a=int(input('a='))
b=int(input('b='))
c=int(input('c='))
bitta_musbat=bool((a>0 and b<0 and
c<0)or(a<0 and b>0 and c<0)or(a<0 and
b<0 and c>0))
print(a,',',b,',',c,'sonlaridan faqat bittasi
musbat son:',bitta_musbat)
```



Masala. Musbat butun son berilgan. Ularni rostlikka tekshiring: “Berilgan son uch xonali toq son”.

```
a=int(input('a='))
b=bool(a>99 and a<1000 and a%2==1)
print('Berilgan son uch xonali toq son:',b)
```

```
Edit Shell Debug Options Window Help
Python 3.12.0 (tags/v3.12.0:0fb18b0, Oct  2 2023, 13:03:39) [MSC v.1935 64 bi
AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.

= RESTART: C:/Users/akbar/AppData/Local/Programs/Python/Python312/O`ktamov.M.
a=523
Berilgan son uch xonali toq son: True
```

Masala. A, B, C sonlar berilgan (A soni noldan farqli). $D=B^2-4AC$ diskriminantdan foydalanib, jumlani rostlikka tekshiring: “$Ax^2+Bx+C=0$ kvadrat tenglama haqiqiy ildizga ega”.

```
import math
a=int(input('a='))
b=int(input('b='))
c=int(input('c='))
d=math.pow(b,2)-4*a*c
natija=bool(a!=0 and d>=0)
print(natija)
```

```
Edit Shell Debug Options Window Help
Python 3.12.0 (tags/v3.12.0:0fb18b0, Oct  2 2023,
AMD64)] on win32
Type "help", "copyright", "credits" or "license()

= RESTART: C:/Users/akbar/AppData/Local/Programs/
a=1
b=5
c=6
True
```

Masala. x, y sonlar berilgan. Ularni rostlikka tekshiring: “Koordinatalari (x,y) bo‘lgan nuqta, koordinata choragining ikkinchisida yotadi”.

```
x=int(input('x='))
y=int(input('y='))
chorak_2=bool(x<0 and y>0)
print('Koordinatalari (',x,',',y,") bo‘lgan nuqta koordinata choragining ikkinchisida yotadi:",chorak_2)
```

```
Edit Shell Debug Options Window Help
Python 3.12.0 (tags/v3.12.0:0fb18b0, Oct  2 2023, 13:03:39) [MS
C v.1935 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more inf
ormation.

= RESTART: D:/Madadjon/O`ktamov.M.py
x=-2
y=7
Koordinatalari ( -2 , 7 ) bo‘lgan nuqta koordinata choragining
ikkinchisida yotadi: True
```

Masala. a, b, c butun sonlar berilgan. Ularni rostlikka tekshiring: “a, b, c tomonli uchburchak yasash mumkin”.

```
a=int(input('a='))
b=int(input('b='))
c=int(input('c='))
print(bool((a+b)>c or (a+c)>b or (b+c)>a))
```

```
Edit Shell Debug Options Window Help
Python 3.12.0 (tags/v3.12.0:0fb18b0, Oct
AMD64)] on win32
Type "help", "copyright", "credits" or "

= RESTART: C:/Users/akbar/AppData/Local/
a=3
b=5
c=7
True
```

Masala. Shaxmat doskasining ikkita turli (x1, y1), (x2, y2) koordinalari berilgan (1-8 oraliqda yotuvchi butun sonlar). Ularni rostlikka tekshiring: “Ot bir yurishda bir maydondan ikkinchisiga o‘ta oladi”.

```
x1=int(input('x1='))
y1=int(input('y1='))
x2=int(input('x2='))
y2=int(input('y2='))
print(bool((x1>=1 and x1<=8 and y1>=1 and y1<=8 and x2>=1 and x2<=8 and y2>=1 and y2<=8) and (abs(y2-y1)==2 or abs(x2-x1)==2 and abs(y2-y1)==1)))
```

```
Edit  Shell  Debug  Options  Window  Help
Python 3.12.0 (tags/v3.12.0:0fb18b0,
AMD64)] on win32
Type "help", "copyright", "credits"

= RESTART: D:/Madadjon/O`ktamov.M.py
x1=6
y1=6
x2=4
y2=4
True
```

5-MAVZU. PYTHON DASTURLASH TILI TARKIBIDAGI MATEMATIK FUNKSIYALAR VA IFODALAR

Reja:

1. Python dasturlash tilida matematik funksiyalar.
2. Python dasturlash tilida ifodalar;

Tayanch so‘zlar: ifoda, import, math, standart funksiya, min, max, sum.

Barcha dasturlash tillari kabi *Python* dasturlash tilida ham matematik ifodalar ma’lum bir standartlar asosida yoziladi. Ifodalar tarkibidagi matematik funksiyalar Python tilida standart fuksiyalar yordamida yoziladi, agar ifoda tarkibidagi funksiya standart funksiya tarkibida bo‘lmasa, alohida funksiya yaratib olish kerak.

Ifoda - sonlar, harflarni arifmetik amallar va qavslar bilan birlashtirilgan yozuvga aytiladi.

Python dasturlash tilidagi ifodalar tarkibidagi amallarni bajarilishi matematikadagi amallarni bajarilish tartibiga mos keladi. Python tilida arifmetik amallarni yozilishi yuqoridagi mavzuga asosan yoziladi. Ifodalar tarkibidagi noma’lumlar faqatgina lotin alifbosida yozilishi kerak. Ifoda tarkibida kasr sonning surati yoki maxrajida ikki va undan ortiq hadlar bo‘lsa, python tilida ular albatta qavsga olinishi kerak.

Python dasturlash tilida matematik funksiyalar

Python dasturlash tili tarkibida mavjud bo‘lgan matematik funksiyalar **standart funksiyalar** deb ataladi.

Ifodalar tarkibidagi funksiyalarni Python dasturlash tilida ifodalash uchun satandart funksiyalardan foydalaniladi. Funksiyalarni python dasturlash tilida ifodalash uchun ularni argumentlarini albatta qavsga olib yozish kerak.

Python dasturlash tilida matematik funksiyalardan foydalanish uchun albatta python tili tarkibidagi matematik funksiyalar kutubxonasiga murojaat qilish kerak. Matematik funksiyalar kutubxonasiga murojaat qilish quyidagicha **from math import*** ko‘rinishda bo‘ladi**.**

№	Python dasturlash tilida ifodalanishi	Matematik ifodalanishi

1.	abs()	sonning absolyut qiymati.
2.	%	birinchi argumentni ikkinchi argumentga bo‘lgandagi qoldiq.
3.	floor(x)	xga teng yoki kichik eng katta butun son
4.	ceil(x)	x ga teng yoki katta eng kichik butun son.
5.	fmod(x,y)	xni yga bo‘lgandagi qoldiq qism.
6.	factorial(x)	x ning faktoriali. N!=1*2*3*…*n.
7.	hypot(x,y)	sqrt(x*x+y*y)
8.	modf(x)	juftlikda xning butun va kasr qismini qaytaradi.
9.	pow(x,y)	x**y.
10.	math.ceil()	eng yaqin yuqori butun songacha yaxlitlaydi.
11.	math.floor()	eng yaqin pastki butun songacha yaxlitlaydi.
12.	trunc(x)	sonning butun qismi trunc(5.8)=5, trunc(-5.8)=-5
13.	sqrt(x)	x ning kvadrat ildiz
14.	log(x),log2(x),log10(x)	ln x, $\log_2 x$, $\log_{10} x$
15.	log(x,a)	$\log_a x$
16.	sin(x), cos(x), tan(x)	sinx, cosx, tgx trigonametrik funksiyalar
17.	asin(x), acos(x), atan(x)	arcsinx, arccosx, arctgx teskari trigonametrik funksiyalar
18.	degrees(x)	radiandan gradusga o‘tkazish funksiyasi
19.	radians(x)	gradusdan radianga o‘tkazish funksiyasi
20.	sinh(x), cosh(x), tanh(x)	giperbolik Sinx, cosx ,tgx trigonametrik funksiyalar
21.	asinh(x), acosh(x), atanh(x)	giperbolik Arcsinx, arccosx, arctgx teskari trigonametrik funksiyalar
22.	hypot(x,y)	katetlari x va y bo‘lgan to‘g‘ri burchakli uchburchakning gepatinuzasini topish
23.	factorial(x)	x faktorialni hisoblash funksiyasi
24.	gamma	x ning gamma funksiyasi
25.	pi	π soni π=3.1415…
26.	e	eksponentsial funksiya e=2.71…

Matematik funksiyalarning bajarilish jarayoni, matematikada qanday bo‘lsa python dasturlash tilida ham xuddi shunday amalga oshiriladi.

Matematik funksiyalarni ishlash jarayoni tushunarli bo‘lishi uchun, ularni interaktiv rejimda sinab ko‘ramiz. Chunki interaktiv rejim bir vaqtning uzida natija qaytaradi.

Misol. Matematik funksiyalarni bajarilishi

```
>>> from math import*
>>> x=15.8
>>> trunc(x)
15
```

```
>>> trunc(16.3)
16
>>> trunc(-12.7)
-12
>>> trunc(-12.2)
-12
>>> sqrt(225)
15.0
>>>log(e),log2(8),log10(x)
(1.0, 3.0, 1.1038037209559568)
>>> log(81,3)
4.0
 >>> cos(pi)
-1.0
>>> atan(180)
1.565240828394204
>>> radians(180), degrees(pi/3)
(3.141592653589793, 59.99999999999999)
```

abs() - sonning absolyut qiymati

```
a=int(input('Sonni kiriting='))
Absolut_qiymat = abs(a)
print('Absolut
qiymat',Absolut_qiymat)
```

Qo‘shimcha funksiyalar

Python dasturlash tilida standart kutubxona tarkibida ketma ketliklar ustida bir nechta maximum, minimum, summa kabi funksiyalar aniqlangan. Boshqa dasturlash tillarida bu funksiyalar alohida algoritmlar yordamida tuzib olinadi, python dasturlash tilida esa bu funksiyalar tayyor holda saqlanadi. Qo‘shimcha funksiyalar quyidagi jadval ko‘rinishida amalga oshiriladi.

№	Python dasturlash tilida ifodalanishi	Matematik ifodalanishi
1	max(a,b,…)	Sonlar yoki kortej ichidan eng kattasini topish. max(2,-8)=2
2	min(a,b,…)	Sonlar yoki kortej ichidan eng kichigini topish. min(2,-8)=-8
3	sum(a,b,…)	Sonlar yoki kortej yig‘indisini topish. sum(2,8)=-6
4	sorted(a,b,…)	Sonlarni tartiblash. sorted(3,12,-9)=(-9,3,12)

Qo‘shimcha funksiyalarni ishlash jarayoni tushunarli bo‘lishi uchun, ularni interaktiv rejimda sinab ko‘ramiz. Chunki interaktiv rejim bir vaqtning uzida natija qaytaradi.

Misol. Qo‘shimcha funksiyalarni bajarilishi

```
>>> max(12,5,-8,7)
12
>>> min(12,5,-8,7)
-8
>>> a=(1,2,3,0,-12,123)
>>> a
(1, 2, 3, 0, -12, 123)
>>> max(a), min(a), sum(a)
(123, -12, 117)
>>> sorted(a)
[-12, 0, 1, 2, 3, 123]
>>>
```

Misol. Quyidagi ifodalarni python tilida ifodalash.

Matematik ifodasi

$$y = (x + \sin x)^3 - \cos^2 x + \frac{1 + \log_a x}{\sqrt{t - x^2}}$$

Python tilida ifodalanishi

y=pow((x+sin(x)),3)+sqr(cos(x))+(1+log(x,a))/(sqrt(t-x**2));

Misol. Quyidagi ifodalarni Python tilida ifodalash.

Matematik ifodasi

$$y = |x - 2| + \sin x - \left|\frac{4}{\sqrt{t - x^2}}\right|$$

Python tilida ifodalanishi

y=abs(x-2)+sin(x)-abs(4/sqrt(t-x**2));

Python dasturlash tilida sanoq sistemasining qo‘llanilishi

Maktab darsligidagi informatika fanidan bizga ma’lumki, sonlar nafaqat o‘nlik sanoq sistemasida emas, balki boshqa sanoq sistemalarida ham bo‘lishi mumkin. Misol uchun, kompyuter ikkilik sanoq sistemasidan foydalanadi ya’ni 15-soni ikkilik sanoq sistemasida (kompyuterda) 1111 ko‘rinishida ifodalanadi. Bundan tashqari sonlarni bir sanoq sistemasidan ikkinchi sanoq sistemasiga o‘tkazish mumkin. Python dasturlash tili buning uchun bir qancha funksiyalarni taqdim etadi:

int([object],[sanoq sistemasi asosi]) - butun sonni berilgan sanoq sistemasidan o‘nlik sanoq sistemasiga o‘tkazadi.

bin(x)- butun sonni ikkilik sanoq sistemasiga o‘tkazadi.

hex(x)- butun sonni o‘n oltilik sanoq sistemasiga o‘tkazadi.

oct(x)- butun sonni sakkizlik sanoq sistemasiga o‘tkazadi.

```
Python 3.11.3 (tags/v3.11.3:f3909b8, Apr  4 2023, 23:49:59) [MSC v.1934 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> bin(15)
'0b1111'
>>> oct(15)
'0o17'
>>> hex(15)
'0xf'
>>> int('1111',2)
15
>>> int('0b1111',2)
15
>>> a=int('15') # satrni songa o`tkazadi
>>> b=int(10.5) # haqiqiy sonni butun songa qaytaradi
>>> print(a,b)
15 10
```

Masala. Uzunlik santimetrda berilgan bo‘lsa, uni metrda ifodalang.

```
L = int(input("L= (sm da)"))
m = L // 100
print(m, "metr")
```

```
>>>
=== RESTART: C:/
L= (sm da)900
9 metr
>>>
```

Masala. Agar og‘irlik kilogramda berilgan bo‘lsa, uni tonnaga aylantiring.

```
m = int(input("M= (kg da)"))
t = m // 1000
print(t, "tonna")
```

```
>>>
=== RESTART: C:/Us
M= (kg da)70000
70 tonna
>>>
```

Masala. Faylning hajmi baytlarda berilgan. Bo‘lib butunni olish operatsiyasidan foydalanib fayl hajmining to‘liq kilobaytlarda ifodalovchi programma tuzing.

```
m = int(input("baytda="))
t = m // 1024
print(t, "Kb")
```

```
>>>
=== RESTART: C:/Use
baytda=6956300
6793 Kb
>>>
```

Masala. A va V (A>V) musbat sonlari berilgan. A kesmada, V kesmani necha marta joylashtirish mumkinligini aniqlovchi programma tuzing.

```
A = int(input("A="))
V = int(input("V="))
t = A // V
print(t, "marta")
```

```
=== RESTART: C
A=9633
V=45
214 marta
>>>
```

Masala. A va V (A>V) musbat sonlar berilgan. A kesmada V kesmani necha marta joylashtirish mumkin. A kesmada V kesmaning joylashmagan qismini aniqlovchi dastur tuzing.

```
A = int(input("A="))
V = int(input("V="))
t = A // V
m = A % V
print(t, "marta")
print(m, "qolgan qismi")
```

```
>>>
=== RESTART: C:/Users/Use
A=9635
V=23
418 marta
21 qolgan qismi
>>>
```

Masala. 2 xonali son berilgan. Oldin uning o‘nliklar xonasidagi raqamni, so‘ng birlar xonasidagi raqamni chiqaruvchi dastur tuzing.

```
A = int(input("A="))
t = A // 10
m = A % 10
print(t, "o'nlik")
print(m, "birlik")
```

```
>>>
=== RESTART: C:/
A=89
8 o'nlik
9 birlik
>>>
```

Masala. 2 xonali son berilgan. Uning raqamlari yig‘indisini aniqlovchi dastur tuzing.

```
A = int(input("A="))
t = A // 10
m = A % 10
d=t+m
print("natija", d)
```

```
=== RESTART:
A=96
natija 15
>>>
```

Masala. 2 xonali son berilgan. Uning raqamlari o‘rnini almashtirishdan hosil bo‘lgan sonni aniqlovchi dasturini tuzing.

```
A = int(input("A="))
t = A // 10
m = A % 10
d=m*10+t
print("natija", d)
```

```
=== RESTAR
rams/Pytho
A=89
natija 98
>>>
```

Masala. 3 xonali son berilgan. Uning yuzlar xonasidagi raqamini aniqlovchi dastur yarating.

```
A = int(input("A="))
t = (A// 100) % 10
print("yuzlik", t)
```

```
>>>
= RESTART: C
ms/Python/Py
A=789
yuzlik 7
>>>
```

Masala. 3 xonali son berilgan. Oldin uni birliklar xonasidagi raqamni so‘ng o‘nliklar xonasidagi raqamni chiqaruvchi dastur tuzing.

```
A = int(input("A="))
t = A % 10
m= (A // 10) % 10
print("birlik", t)
print("o'nlik", m)
```

```
= RESTART: C:/
ms/Python/Pyth
A=789
birlik 9
o'nlik 8
>>>
```

Masala. 2 nuqta orasidagi masofani toping.

```
x1 = float(input("x1="))
x2 = float(input("x2="))
m = abs(x2 - x1)
print("masofa", m)
```

```
>>>
= RESTART: C
Y
x1=9
x2=3
masofa 6.0
```

6-MAVZU. PYTHON DASTURLASH TILIDA CHIZIQLI, TARMOQLANUVCHI VA TAKRORLANUVCHI JARAYONLARNI DASTURLASH

Reja:

1. Python dasturlash tilida chiziqli jarayonlarni dasturlash
2. Python dasturlash tilida tarmoqlanuvchi jarayonlarni dasturlash
3. Python dasturlash tilida takrorlanuvchi jarayonlarni dasturlash

Tayanch so‘zlar: type, help, operator, chiziqli dastur, o‘zgaruvchi, If, else, elif, qisqa shartli, to‘liq shartli, for, sikl, takrorlanish, parametrli takrorlanish, ichma ich.

Python dasturlash tilida chiziqli jarayonlarni dasturlash

Odatda tabiat yoki jamiyatda uchraydigan turli muammo, masala yoki jarayonlarni o‘rganishni kompyuter yordamida olib borish uchun, birinchi navbatda, qaralayotgan masala, jarayon - obyektning matematik ifodasi, ya’ni matematik modelini ko‘rish kerak bo‘ladi. Qaralayotgan obyektning matematik modelini yaratish juda murakkab jarayon bo‘lib, o‘rganilayotgan obyektga bog‘liq ravishda turli soha mutaxassislarining ishtiroki talab etiladi. Umuman, biror masalani kompyuter yordamida yechishni quyidagi bosqichlarga ajratish mumkin. Qo‘yilgan chiziqli masalani kompyuterda yechish uchun, avval uning matematik modelini, keyin algoritmini va dasturini tuzish kerak bo‘ladi. Har qanday murakkab algoritmni ham uchta asosiy struktura yordamida tasvirlash mumkin. Bular ketma-ketlik, ayri va takrorlash strukturalaridir. Bu strukturalar asosida chiziqli, tarmoqlanuvchi va takrorlanuvchi hisoblash jarayonlarining algoritmlarini tuzish mumkin.[6]

Programmalash tilida tuzilgan dasturlar odatda 3 ta jarayonga asoslanib tuziladi. Dasturlash tili operatorlari yechilayotgan masala algoritmini amalga oshirish uchun ishlatiladi. Operatorlar chiziqli va boshqaruv operatorlariga bo‘linadi. Aksariyat boshqa dasturlash tillarida operatorlar nuqtali vergul (‘;’) belgisi bilan tugallanadi, python dasturlash tilida esa ; nuqtali vergulni quyish shart emas. Dastur tuzish vaqtida

[6] Sh.A.Mengliyev, O.A.Abdug‘aniyev, S.Q.Shonazarov, D.Sh.Turayev. Python dasturlash tili.-Termiz 2021, 34-bet

buyruqlar ketma-ketligi uzluksiz bajarilib boshqa shartlar talab etilmasa, dastur chiziqli hisoblanadi.

Tarif: Chiziqli algoritmlarga asoslanib python dasturlash tilida tuzilgan dasturlar **chiziqli dasturlar** deyiladi.

Chiziqli dasturlar tarkibiy qismi bo‘lgan operator va buyruqlarda hech qanday shart yoki takrorlanish bajarilmaydi. Chiziqli dasturlar tarkibidagi buyruqlar, albatta, bir marta bajariladi.

Masala. 2 ta haqiqiy sonlar berilgan. Ularning yig‘indisi, ayirmasi va ko‘paytmasini hisoblang.

```
        a=float(input('a='));
        b=float(input('b='));
        s=a+b; d=a-b; k=a*b;
print('yigindi=',s,'\n''ayirma=',d,'\n'
'kopaytma=',k)
```

```
Edit  Shell  Debug  Options  Window  Help
Python 3.12.0 (tags/v3.12.0:0fb18b0,
 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" c
.

= RESTART: D:/Madadjon/O`ktamov.M.py
a=35
b=7
yigindi= 42.0
ayirma= 28.0
kopaytma= 245.0
```

Masala. Tomonlari A va B ga teng to‘g‘ri to‘rtburchakning yuzi va perimetrini hisoblang.

```
        a=float(input('a='))
        b=float(input('b='))
        s=a*b;
        p=2*(a+b);
print('yuzi=',s,'\n''perimetri=',p)
```

```
Edit  Shell  Debug  Options  Window  Help
Python 3.12.0 (tags/v3.12.0:0fb18b0,
AMD64)] on win32
Type "help", "copyright", "credits" o

= RESTART: D:/Madadjon/O`ktamov.M.py
a=25
b=19
yuzi= 475.0
perimetri= 88.0
```

Masala. 2 ta musbat son berilgan, bu sonlarning o‘rta arifmetik va o‘rta geometrik qiymatlarini aniqlaydigan dastur tuzing.

```
import math
a=float(input('a='));
b=float(input('b='));
s=(a+b)/2;
d=math.sqrt(a*b);
print(‘O‘rta arifmetik
 qiymati=’,s, ’\n’
‘O‘rta geometrik qiymati=,d)
```

```
Edit  Shell  Debug  Options  Window  Help
Python 3.12.0 (tags/v3.12.0:0fb18b0, Oct  2
AMD64)] on win32
Type "help", "copyright", "credits" or "lic

= RESTART: D:/Madadjon/O`ktamov.M.py
a=65
b=29
O‘rta arifmetik qiymati= 47.0
O‘rta geometrik qiymati= 43.41658669218482
```

Masala. Teng tomonli uchburchakning tomoni A ga teng. Uchburchakning yuzini toping.

```
import math
a=float(input('tomoni='));
s=math.pow(a,2)*math.sqrt(3)/4;
print('yuzi=',s)
```

```
Edit Shell Debug Options Window Help
Python 3.12.0 (tags/v3.12.0:0fb18b0, Oct  2 2023, 13:03:39
AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for mor

========== RESTART: D:/Madadjon/O`ktamov.M.py
tomoni=15
yuzi= 97.42785792574934
```

Masala. Berilgan sonning butun qismini aniqlang.

```
import math
A=float(input('A='));
B=math.floor(A);
print(A,'sonining butun qismi=',B)
```

```
Edit Shell Debug Options Window Help
Python 3.12.0 (tags/v3.12.0:0fb18b0, Oc
AMD64)] on win32
Type "help", "copyright", "credits" or

= RESTART: D:/Madadjon/O`ktamov.M.py
A=26.158
26.158 sonining butun qismi= 26
```

Python dasturlash tilida chiziqli jarayonlar buyruqlar ketma-ketligi asosida bajariladi, tarmoqlanuvchi jarayonlarni dasturlashda esa, buyruqlar ma’lum bir shartlar asosida tarmoqlanish bo‘yicha bajariladi. Python dasturlash tilida tarmoqlanuvchi(shartli) jarayonlarni bir necha turlarga bo‘lingan holda amalga oshirish mumkin.

Python dasturlash tilida tarmoqlanuvchi jarayonlarni dasturlash

Python dasturlash tilida shart operatorlari shartni tekshirish uchun ishlatiladi. Bu tilda shart operatorini bir necha xil ko‘rinishi mavjud:

if (mantiqiy ifoda): - mantiqiy ifoda rost bo‘lgan holda qandaydir kod bajarilishi uchun ishlatiladi.

if (mantiqiy ifoda):…**else – agar** mantiqiy ifoda rost bo‘lsa, birinchi ifodalar bloki bajariladi(bu blok “**if-blok**” deb nomlanadi), **aks holda** keyingi ifodalar bloki bajariladi(bu blok “**else-blok**” deb nomlanadi).

if (mantiqiy ifoda):…**elif(mantiqiy ifoda):**…**else -** oldingi shart yolg‘on bo‘lganda keyingi shart tekshiriladi. Bu ifoda o‘zida 2 ta bir-biriga bog‘liq bo‘lgan **if else-if else** ifodani bir ifodada **if elif else** saqlaydi. Bu esa dasturni o‘qishni osonlashtiradi.

Shart operatorining umumiy ko‘rinishi:

if <shart>

<operator 1>

else

<operator 2>

Shartli operator sintaksisi: ***if*** (<shart>) <operator1> ***else*** <operator2>. Shart **<shart>** ixtiyoriy shartli ifoda bo‘lishi mumkin. Agar u rost bo‘lsa **operator1** bajariladi. Aks xolda **operator2** bajariladi. Bu ixtiyoriy murakkablikdagi tekshirishlar ketma ketligini hosil qilishga imkon beradi. Bu ketma - ketlikda shartli operator to‘la yoki qisqa shaklda bo‘lishi mumkin. Shuning uchun ***if*** va ***else*** operatorlarini bir-biriga mos qo‘yishda xatolik kelib chiqishi mumkin. Tilning sintaksisi bo‘yicha ichki joylashtirilgan shartli operatorlarda har bir ***else*** eng yaqin **if** ga mos keladi.

Pythonda bir nechta shartlarni tekshirish uchun if-elif else operatoridan foydalaniladi

If yordamida biz faqatgina bitta shartni tekshira olamiz va uning natijasiga ko‘ra (True/False) dasturimiz ma’lum bir amallarni bajaradi. Agar dastur davomida bir nechta shartlarni tekshirishimiz talab qilinsa, if-elif-else ketma-ketligidan foydalanamiz.

Bu ketma-ketlikning ko‘rinishi quyidagicha:

if <shart 1>

<operator 1>; **elif** <shart 2>

<operator 2>**;**

…

elif <shart N> <operator N>**;**

else

<operator N+1>

if-elif-else ketma-ketligida Python avval **if <shart1>** ni tekshiradi, shart bajarilmasa, keyingi **elif** ga o‘tadi, birinchi elif sharti bajarilmasa, keyingi elif ga o‘tadi va hokazo davom etaveradi.

IF

if kalit so‘zi biror shartning bajarilishi yoki bajarilmasligini tekshiradi. Masalan, bir qiymat ikkinchisidan kattaligi yoki ular o‘zaro teng emasligi va hokazo kabi

shartlarni tekshirish mumkin. Hozir oddiy misol qilib a sonni b sonidan katta ekanlgini tekshirib ko‘ramiz. Agar shart bajarilsa, “**HA**” degan yozuv ekranga chiqadi: [7]

```
a = 50
b = 30
if a>b:
print("HA")
```

Shart tekshirilgach, bajariladigan amalni keyingi qatorda yozishda, xuddi abzatsdan yozgan kabi yozish kerak aks holda dasturda xatolik yuz beradi. Tushunish uchun avval yuqoridagi kodga qarang, keyin quyidagi kodga e’tibor bering. Bu kodimiz ishga tushganda xatolik yuz beradi. Chunki so‘nggi qator abzatsdan boshlanishi kerak edi.

```
a = 50
b = 30
if a>b:
print("HA")
```

else

else kalit so‘zi “**aks holda**” ma`nosini beradi. Bu shartimiz bajarilmaganda qaysi amalni bajarish kerakligini ko‘rsatish uchun qo‘llaniladi. Misol uchun, a soni b sonidan katta bo‘lsa, “**HA**” yozuvini ekranga chiqaramiz, agar bu shart bajarilmasa, “**YO‘Q**” yozuvini ekranga chiqaramiz:

```
a = 50
b = 90
if a>b:
   print("HA")
else:
print("YO‘Q")
```

elif

Agarda bir emas, balki ko‘proq shartlarni tekshirishga to‘g‘ri kelsa, biz **elif** kalit so‘zini ishlatamiz. Bunday holatda **if** kalit so‘zi bilan shart tekshiriladi, qolganlari esa **elif** kalit so‘z bilan tekshiriladi.

[7] www.bilimlar.uz/wp-content/uploads/2021/02/k100001.pdf

```
a = 50 b = 30
if a>b:
print('a soni b sondan katta')
elif a==b:
   print('ular o‘zaro teng')
elif a<b:
print('a soni b sondan kichik')
else:
print('Hech qaysi shart bajarilmadi !!! ')
```

Pass

if kalit so‘zi bilan shart tekshirilgandan keyin bajariladigan amalni albatta yozishimiz kerak. Aks holda dasturda xatolik yuz beradi. Ammo hali nima amal bajarish kerakligini o‘ylab ko‘rmagan bo‘lsak, u yerga **pass** so‘zini qo‘yish kifoya. Bu so‘z tufayli dastur ishga tushganda aynan o‘sha qismni hisobga olmasdan o‘tib ketadi. Natijada dasturning qolgan qismlariga bu ta’sir qilmaydi.[8]

```
a = 33
b = 99
if b>a:
pass
print('pass qo‘yilmaganda bu yozuv ekranga chiqmasdi')
```

Misol uchun 2 ta son kiritilganda ularni bir-biri bilan taqqoslaydigan dastur va uning natijasini ko‘raylik:

```
x=int(input('x=')); y=int(input('y='));
if x<y:
print(x,'kichik',y,'dan')
elif x==y:
print(x,'ga',y,'teng‘)
else:
print(x,'katta',y,'dan')
```

```
Edit  Shell  Debug  Options  Window  Help
Python 3.12.0 (tags/v3.12.0:0fb18b0, Oct  2 2023, 13:03:39) [MS
AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more inf

= RESTART: D:/Madadjon/O`ktamov.M.py
x=15
y=25
15 kichik 25 dan

========== RESTART: D:/Madadjon/O`ktamov.M.py =====
x=9
y=9
9 ga 9 teng

========== RESTART: D:/Madadjon/O`ktamov.M.py =====
x=25
y=6
25 katta 6 dan
```

1-holatda x ga 15, y ga 25 qiymatlarini berganimizda natija: “15 kichik 25 dan” javobi chiqadi,

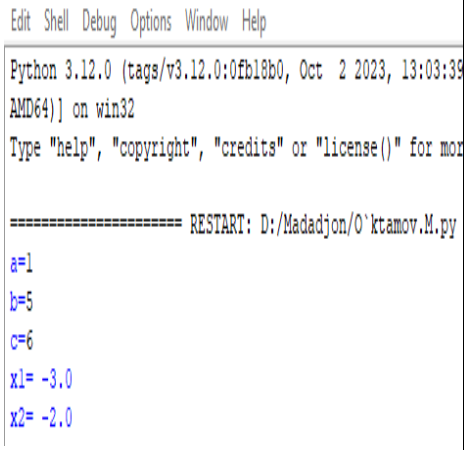
2-holatda x ga 9, y ga 9 qiymatlarini berganimizda natija: “9 ga 9 teng” javobi chiqadi,

[8] www.bilimlar.uz/wp-content/uploads/2021/02/k100001.pdf

3-holatda esa x ga 25, y ga 6 qiymatlarini berganimizda natijamiz: “25 katta 6 dan” javoblari chiqadi.

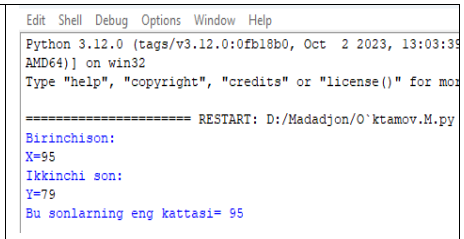
Masala. $Ax^2+Bx+C=0$ kvadrat tenglamaning haqiqiy ildizlarini topish dasturi.

```
import math
a=float(input('a='));
b=float(input('b='));
c=float(input('c='));
d=math.pow(b,2)-4*a*c
if d>0:
x1=(-b-math.sqrt(d))/(2*a)
x2=(-b+math.sqrt(d))/(2*a)
print('x1=',x1,'\nx2=',x2)
elif d==0:
x=-b/(2*a)
print('Tenglama bitta ildizga ega\nx=',x)
else:
print('Diskreminant 0 dan kichik yechim mavjud emas!')
```



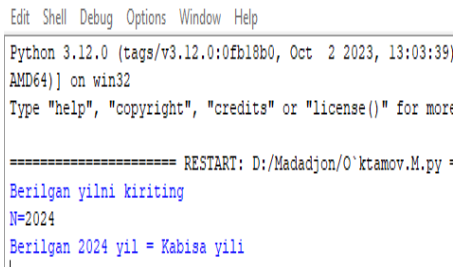
Masala. Ikkita X sonlarning kattasini tanlash (EKT) dasturini yarating.

```
X=int(input('Birinchison:\nX='));
Y=int(input('Ikkinchi son:\nY='));
 if X>Y:
print('Bu sonlarning eng kattasi=',X)
elif X==Y:
print('Bu sonlar bir-biriga teng! ')
else: print('Bu sonlarning eng kattasi=',Y)
```



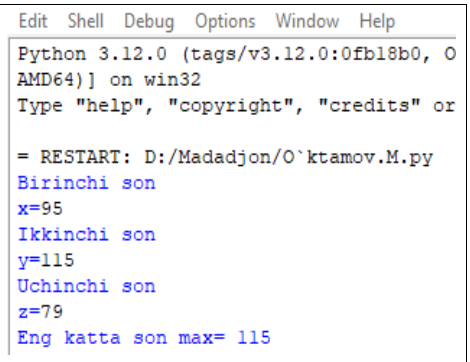
Masala. N sonli yil kabisa yili bo‘lishi yoki bo‘lmasligini aniqlaydigan dastur tuzing. Agar N soni 100 ga karrali son bo‘lmasa va uning oxirgi ikki raqami 4 ga karrali son bo‘lsa, u holda N-yil kabisa yilidir. Agar N soni 100 karrali bo‘lsa, u holda N soni 400 ga karrali bo‘lgandagina mazkur yil kabisa yili bo‘ladi.

```
N=int(input("Berilgan yilni kiriting\nN="))
Y=N%100;
Z=N%10
if Y!=0 and Z%4==0:
print('Berilgan',N, 'yil = Kabisa yili')
else:
print('Berilgan',N, 'yil = Kabisa yili emas!')
```



Masala. 3 ta son berilgan, ularning eng kattasini topuvchi dastur tuzing.

```
x=int(input("Birinchi son\nx="))
y=int(input("Ikkinchi son\ny="))
z=int(input("Uchinchi son\nz="))
if x>y:    max=x
else:
max=y
if max>z:    max=max
else: max=z
print("Eng katta son max=",max)
```



Python dasturlash tilida takrorlanuvchi jarayonlarni dasturlash

Dastur tuzish jarayonida ba’zi bir masalalarni algoritmlari tarkibidagi buyruqlar ikki va undan ortiq marta bajarilishiga to‘g‘ri keladi. Agar algoritm tarkibidagi bir necha marta takrorlanishi kerak bo‘lgan buyruqlarni takrorlanuvchi jarayonlar asosida dasturlash tillarida tasvirlanmasa, bu buyruqlarni barchasini bajarish murakkablashadi. Elektron hisoblash mashinalarini insoniyatdan farqi shundaki, insoniyatda bir nechta buyruqlarni bajarish davomida toliqish holatlari bo‘lishi mumkin, elektron hisoblash mashinalariga takrorlanishni qanchaligini ma’lum bir buyruqlar asosida berilsa, ular barchasini charchamasdan bajaradi. Ba’zi bir takrorlanuvchi jarayonlarni, takrorlanish formulasini chiqarib oddiy hisoblash mumkin, lekin ixtiyoriy ketma-ketliklar yig‘indisini hisoblash oddiy usullar bilan hal etilmaydi, bunday holatlarda takrorlanuvchi jarayonlardan foydalaniladi.

Tarif: Algoritmning qandaydir qismidagi buyruqlar ikki va undan ortiq bajarilishiga **takrorlanuvchi jarayonlar** deyiladi.

Python dasturlash tilida 2 xil sikldan foydalaniladi. Bular **while** va **for** sikllaridir. Ularning qulayligi shuki, bu sikllar belgilangan nuqtaga yetmaguncha ko‘rsatilgan amalni qayta-qayta bajaraveradi. Shu sababli biz bir amalni qayta-qayta yozib o‘tirmaymiz. **while** va **for** sikllari qo‘llanish usuli va joyiga ko‘ra farqlanadi.

Python dasturlash tilida while siklining qo‘llanilishi

Odatda **while** sikliga bir shart berish kerak bo‘ladi va o‘sha shart bajarilmaguncha u biz ko‘rsatgan amalni qayta va qayta bajaraveradi. **while** siklining umumiy ko‘rinishi quyidagicha:

```
while (shart):
sikl_tanasi
```

While sikl operatorining ishlash tartibi quyidagicha:

Agar (shart) rost (**true**) qiymatga ega bo‘lsa, **sikl_tanasi** bajariladi. Qachonki shart yolg‘on (**false**) qiymatga teng bo‘lsa sikl tugatiladi.

Agar (shart) true qiymatga ega bo‘lmasa sikl tanasi biror marta bajarilmaydi.

Misol tariqasida 1 da 10 gacha bo‘lgan sonlarni ekranga chiqarishimiz kerak bo‘lsa, uni quyidagicha amalga oshirishimiz mumkin:

Avval, boshlang‘ich nuqtani belgilab olamiz, ya’ni o‘zgaruvchi 1 ga teng bo‘ladi. Undan keyin shunday shart beramizki toki o‘sha shart o‘zgaruvchi 11 dan kichik ekan, uni har safar ekranga chiqarib shu songa 1 ni qo‘shib ketaversin. Natijada o‘zgaruvchimiz toki 10 ga yetgunga qadar ushbu amalni bajaraveradi. 11 ga yetganda esa shart bajarilmaydi va sikl to‘xtaydi.

```
i = 1
while i < 11:
print(i)
   i+=1
```

break

break kalit so‘zi siklni to‘xtatadi. Asosiy sikl davom etayotgan bo‘lsa ham, biz belgilagan nuqtada siklni to‘xtatadi. Masalan yuqoridagi misolni olamiz. Uni shunday

o‘zgartiramizki, o‘zgaruvchimizning qiymati 5 ga yetganda sikl to‘xtaydi va qolgan sonlarni ekranga chiqarmaydi: [9]

```
i = 1
while i < 11:
print(i)
if i == 5:
break   i+=1
```

continue

continue kalit so‘zi bilan siklning ba’zi nuqtalaridan sakrab o‘tish mumkin. Masalan biz 8 dan tashqari 1 dan 10 gacha bo‘lgan sonlarni ekranga chiqaramiz. Bunda 8 soni hisobga olinmay undan o‘tib ketiladi:

```
i = 1 while i < 11:     i+=1
if i == 8:     continue
print(i)
```

else

else kalit so‘zi sikl to‘xtagandan so‘ng ham yana bir amal bajarish imkoni beradi. Masalan, sikl to‘xtagandan so‘ng to‘xtaganligi haqida ma’lumot ekranga chiqsin:

```
i = 1
while i < 11:     print(i)
i+=1
else:
print("sikl to‘xtadi")
```

For sikli

For operatori Python dasturlash tilida C va Paskal dasturlash tillarida qo‘llanishidan yetarlicha farq qiladi.

Pythonda **for** operatori biroz murakkabroq, ammo **while** sikliga qaraganda ancha tezroq bajariladi.

For…in operatori obyektlar ketma-ketligida iteratsiyani amalga oshiradi, ya’ni, bu sikl har qanday iteratsiya qilinadigan obyekt bo‘ylab o‘tadi(satr yoki ro‘yxat bo‘ylab) va har bir o‘tish vaqtida sikl tanasini bajaradi.

[9] https://www.bilimlar.uz/wp-content/uploads/2021/02/k100001.pdf.page

Python dasturlash tilida **for** sikli ko‘pincha to‘plam va ro‘yxatlar bilan ishlatiladi. **For** sikli bilan to‘plam yoki ro‘yxatning har bir elementiga murojaat qilish mumkin. Masalan, quyidagi berilgan ro‘yxatning har bir elementini ekranga chiqaramiz:

```
mashina = ["nexia", "damas", "matiz"]
for in mashina:
print(a)
```

Python dasturlash tilida satr bo‘ylab sikl

Satr bo‘ylab sikl amalga oshirilsa, satrdagi har bitta harfga murojaat bo‘ladi. Chunki satr harflar to‘plamidan tashkil topgan. Masalan “dasturlash” so‘zning barcha harflarini ekranga chiqaramiz:

```
for a in "dasturlash":
   print(a)
```

break

break kalit so‘zi siklni to‘xtatadi, hattoki sikl to‘xtamagan bo‘lsa ham. Masalan, “**dasturlash**” so‘zining harflarini birma-bir ekranga chiqarish siklini ishga tushiramiz va “**s**” harfiga yetganda siklni to‘xtatamiz:

```
for x in "dasturlash":
   print(x)
if x == "s":       break
```

Endi e’tibor beradigan bo‘lsak, yuqoridagi kodda print buyrug‘ini break buyrug‘idan oldinroq qo‘ygan edik. Shu uchun avval **“s”** harfi ekranga chiqdi, so‘ng sikl to‘xtadi. Endi print buyrug‘ini pastroqqa qo‘yadigan bo‘lsak, bunda **“s”** harfi ekranga chiqmay qoladi, chunki sikl undan avvalroq to‘xtaydi.

```
for x in "dastur":
if x == "s":       break
   print(x)
```

continue

continue kalit so‘zi siklning ayrim joylaridan sakrab o‘tadi. Aniqroq qilib aytganda sikl davomida ayrim nuqtalarga kelganda ko‘rsatilgan amalni bajarmay ketadi.

Masalan, “**dasturchi**” so‘zidagi harflarni ekranga chiqaramiz va shunda “**t**” harfini tashlab ketamiz:

```
for x in "dasturchi":
if x == 't':        continue
   print(x)
```

range() va xrange()

range() funksiyasi biror amalni belgilangan marta bajarish yoki biror oraliqdagi sonlarga murojaat qilish uchun qo‘llaniladi. Bunda **range()** ichiga kerakli son qo‘yiladi va sanoq avtomatik tarzda 0 dan boshlanib ko‘rsatilgan songacha davom etadi. Ammo uning o‘zi hisobga kirmaydi. [10]

Tushunish uchun misol ko‘ramiz. 0 dan 5 gacha (5 soni hisobga kirmaydi) bo‘lgan sonlarni ekranga chiqaramiz:

```
for x in range(5):
   print(x)
0
1
2
3
4
```

Yuqorida biz **range()** funksiyasida sanoq avtomatik 0 dan boshlanishini aytib o‘tdik. Biz uni o‘zimiz istagan sondan boshlashimiz mumkin.

Masalan 1 dan 5 gacha bo‘lgan sonlarni ekranga chiqaramiz. Bunda sanoq 1 dan boshlanishi uchun 1 sonini ham kiritamiz. Demak, biz 1 dan 6 gacha bo‘lgan oraliqni kiritamiz:

```
for x in range(1,6):
   print(x)
1
2
3
4
5
```

range() funksiyasida sonlar avtomatik ravishda bittaga ortib boradi. Biroq bu holatni ham o‘zgartirish mumkin. Bunda esa oraliqni ko‘rsatgandan so‘ng sanoq nechtaga ortishini ham kiritamiz. Shunda funksiya ichidagi dastlabki ikkita son oraliqni, uchinchi son esa sanoq nechtaga ortishini ko‘rsatadi.

[10] https://www.bilimlar.uz/wp-content/uploads/2021/02/k100001.pdf

Masalan, 1 dan 10 gacha bo‘lgan faqat juft sonlarni ekranga chiqarmoqchimi. Bunda oraliqni 2 dan 11 gacha deb belgilaymiz. Shunda sanoq 2 dan boshlanadi va 10 gacha davom etadi. Har safar sanoq ikkitaga ortishi uchun uchinchi bo‘lib 2 soni kiritamiz:

```
for x in range(2, 11, 2):
print(x)
2
4
6
8
10
```

Katta diapazondagi raqamlardan foydalanib ro‘yxatni yaratishda esa **range()** funksiyasi o‘zini oqlamaydi yoki ba’zi hollarda xotira yetishmaydi.

```
>>> l=range(10000000)
Traceback (innermost last):
  File "<stdin>", line 1, in ?
MemoryError
```

Shunday hollarda Python dasturlash tilida **xrange()** funksiyasidan foydalanamiz.

else

else kalit so‘zi sikl tugagach ham yana bir amal bajarish imkonini beradi. Ko‘pincha bundan sikl tugagani haqida ma’lumot berishda foydalaniladi.

Masalan, “**dasturlash**” so‘zini 5 marta ekranga chiqarmoqchimiz va sikl tugagach shu haqida xabar beramiz. Bunda e’tibor jihati shuki, **range()** funksiyasi bilan biz sanoq asosida sonlarni ekranga chiqarmayapmiz, balki shuncha marta bir xil amalni bajaryapmiz:

```
for x in range(5):
print("dasturlash ")
else:
   print("\nSikl tugadi!")
```

Sikl ichida sikl

Sikl ichida sikl qo‘llanilganda ichki sikl tashqi siklning har bitta bosqichida bir martadan bajariladi. Hozir biz har bitta rangni har bir mashina bilan birgalikda qo‘llab ko‘ramiz:

```
color = ["qora", "oq", "qizil"]
mashina = ["Spark", "Nexia", "Lacetti"]
 for x in color:
for y in mashina:
print(x,y)
```

pass

for sikli ham xuddi **while** sikli singari bo‘sh bo‘lishi mumkin emas. Ya’ni sikl davomida albatta nima amal bajarilishini kiritishimiz lozim. Ammo bu amal hali aniq bo‘lmasa kodimizda xatolik yuz bermasligi uchun **pass** kalit so‘zidan foydalanamiz va dastur ishga tushganda o‘sha qism hisobga olinmay ketiladi. Masalan, hozir sikl davomida bajarilishi kerak bo‘lgan amalni kiritmay pass kalit so‘zini kiritamiz. Bunda xatolik yuz bermaydi, chunki pass kalit so‘zi qo‘yilgan. Ammo hech qanday amal ham bajarilmaydi, chunki biror amal bajarish haqida buyruq berilmagan. [11]

```
for x in range(5):
   pass
```

Masala. k va n butun sonlari berilgan (n > 0). k sonini n marta chiqaruvchi dastur yarating.

```
k = int(input("Butun sonni kiriting: "))
n = int(input("Necha marta chiqarishni istaysiz? (n > 0): "))

if n <= 0:
    print("n soni musbat bo'lishi shart!")
else:
    for i in range(n):
        print(k)
```

```
>>>
=== RESTART: C:/Users/User/AppData/Local/Program
Butun sonni kiriting: 78
Necha marta chiqarishni istaysiz? (n > 0): 8
78
78
78
78
78
78
78
78
>>>
```

Masala. a va b butun sonlari berilgan (a < b), a va b sonlari orasidagi barcha butun sonlarni (a va b ni ham) chiqaruvchi va chiqarilgan sonlar sonini chiqaruvchi dastur yarating. (a va b xam chiqarilsin).

[11] https://www.bilimlar.uz/wp-content/uploads/2021/02/k100001.pdf

```
a = int(input("Butun sonni kiriting (a): "))
b = int(input("Butun sonni kiriting (b): "))
d = 0
for i in range(a, b + 1):
    print(i)
    d += 1
print("Chiqarilgan sonlar soni:", d)
```

```
>>>
=== RESTART: C:/Users/User/AppDa
Butun sonni kiriting (a): 8
Butun sonni kiriting (b): 12
8
9
10
11
12
Chiqarilgan sonlar soni: 5
```

Masala. a va b butun sonlari berilgan (a < b), a va b sonlari orasidagi barcha butun sonlarni (a va b dan tashqari) kamayish tartibida chiqaruvchi va chiqarilgan sonlar sonini chiqaruvchi dastur yarating.

```
a = int(input("Butun sonni kiriting (a): "))
b = int(input("Butun sonni kiriting (b): "))
d = 0
if a < b:
    for i in range(b - 1, a, -1):
        print(i)
        d += 1
else:
    for i in range(a - 1, b, -1):
        print(i)
        d += 1
print("Chiqarilgan sonlar soni:", d)
```

```
>>>
=== RESTART: C:/Users/User/AppD
Butun sonni kiriting (a): 9
Butun sonni kiriting (b): 12
11
10
Chiqarilgan sonlar soni: 2
```

Masala. 1 kg konfetning narxi berilgan (haqiqiy son). 1, 2, ..., 10 kg konfetni narxini chiqaruvchi dastur yarating.

```
n= float(input("Bir kilogram konfet narxini kiriting: "))
for k in range(1, 11):
    u = k * n
    print(f"{k} kg konfet narxi: {u}")
```

```
>>>
=== RESTART: C:/Users/User/AppData/Local/Prog
Bir kilogram konfet narxini kiriting: 780
1 kg konfet narxi: 780.0
2 kg konfet narxi: 1560.0
3 kg konfet narxi: 2340.0
4 kg konfet narxi: 3120.0
5 kg konfet narxi: 3900.0
6 kg konfet narxi: 4680.0
7 kg konfet narxi: 5460.0
8 kg konfet narxi: 6240.0
9 kg konfet narxi: 7020.0
10 kg konfet narxi: 7800.0
```

Masala. 1 kg konfetning narxi berilgan (haqiqiy son). 0.1, 0.2, ..., 0.9, 1 kg konfetni narxini chiqaruvchi dastur yarating.

```
n = float(input("Bir kilogram konfet narxini kiriting: "))
for k in range(1, 11):
    u = k * n
    print(f"{k / 10} kg konfet narxi: {u/10}")
```

```
=== RESTART: C:/Users/User/AppData/Local/Progr
Bir kilogram konfet narxini kiriting: 78
0.1 kg konfet narxi: 7.8
0.2 kg konfet narxi: 15.6
0.3 kg konfet narxi: 23.4
0.4 kg konfet narxi: 31.2
0.5 kg konfet narxi: 39.0
0.6 kg konfet narxi: 46.8
0.7 kg konfet narxi: 54.6
0.8 kg konfet narxi: 62.4
0.9 kg konfet narxi: 70.2
1.0 kg konfet narxi: 78.0
```

Masala. 1 kg konfetning narxi berilgan (haqiqiy son). 1.2, 1.4, ..., 2 kg konfetni narxini chiqaruvchi dastur yarating.

```
n = float(input("Bir kilogram konfet narxini kiriting: "))
for k in range(12, 21, 2):
    u = k / 10 * n
    print(f"{k / 10} kg konfet narxi: {u}")
```

```
Bir kilogram konfet narxini kiriting: 1200
1.2 kg konfet narxi: 1440.0
1.4 kg konfet narxi: 1680.0
1.6 kg konfet narxi: 1920.0
1.8 kg konfet narxi: 2160.0
2.0 kg konfet narxi: 2400.0
```

Masala. a va b butun sonlari berilgan (a < b). a dan b gacha bo‘lgan barcha butun sonlar yig‘indisini chiqaruvchi dastur yarating.

```
a = int(input("Butun sonni kiriting (a): "))
b = int(input("Butun sonni kiriting (b): "))

if a >= b:
    print("Xatolik: a < b sharti bajarilmadi!")
else:
    y = sum(range(a, b + 1))
    print(f"{a} dan {b} gacha sonlar yig'indisi: {y}")
```

```
=== RESTART: C:/Users/User/AppData/Local
Butun sonni kiriting (a): 12
Butun sonni kiriting (b): 19
12 dan 19 gacha sonlar yig'indisi: 124
```

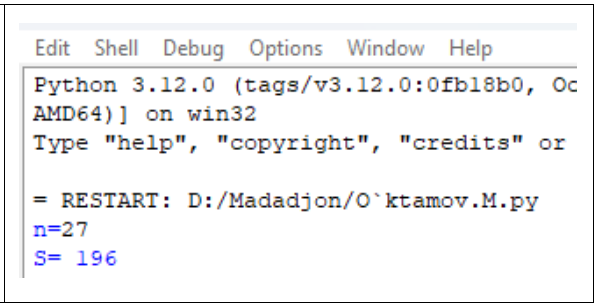
Masala. a va b butun sonlari berilgan (a < b). a dan b gacha bo‘lgan barcha butun sonlar ko‘paytmasini chiqaruvchi dastur yarating.

```
a = int(input("Butun sonni kiriting (a): "))
b = int(input("Butun sonni kiriting (b): "))
if a >= b:
    print("Xatolik: a < b sharti bajarilmadi!")
else:
    k = 1
    for i in range(a, b + 1):
        k*= i
    print(f"ko'paytma: {k}")
```

```
>>>
=== RESTART: C:/Users/User/AppData/L
Butun sonni kiriting (a): 12
Butun sonni kiriting (b): 16
ko'paytma: 524160
```

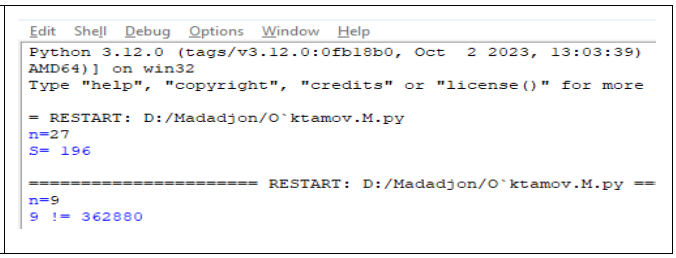
Masala. 1 dan n gacha bo‘lgan sonlarning faqat toq raqamlarining yig‘indisini hisoblovchi dastur yarating.

```
n=int(input('n='))
S=0
for i in range(1,n+1):
if i%2==1:
S+=i
print('S=',S)
```



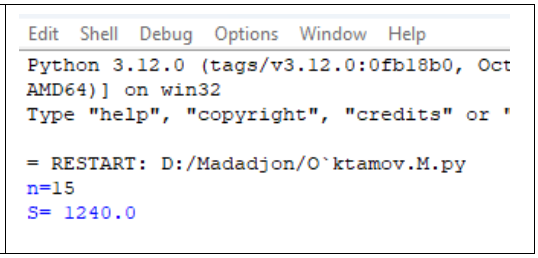
Masala. N! hisoblash talab qilingan bo‘lsin, bunda N natural son. Yechish. N<34 bo‘lganda natural sonlar faktorialini hisoblash mumkin.

```
n=int(input('n='));
p=1;
for i in range(1,n+1):
    p=p*i;
print(n,'!=',p)
```



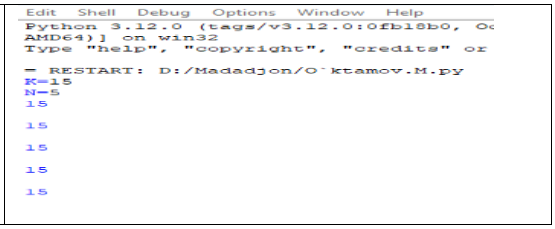
Masala. 1 dan n gacha bo‘lgan natural sonlar kvadratlari yig‘indisini toping. Yechish. Izlanayotgan yig‘indini S bilan belgilaymiz.

```
import math
n=int(input('n=')); s=0;
for i in range(1,n+1):
 s=s+math.pow(i,2);
print('S=',s)
```



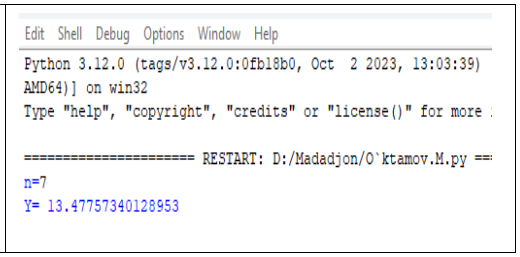
Masala. k va n butun sonlari berilgan (n>0). k sonini n marta chiqaruvchi dastur tuzing.

```
k=int(input('K='));
n=int(input('N='));
for i in range(1,n+1):
  print(k,'\n');
```



Masala. 1 dan n gacha bo‘lgan sonlardan sikl qadami 1 ga teng holda kvadrat ildiz chiqaring.

```
import math
 n=int(input('n=')); y=0;
for i in range(1,n+1):
 y=y+math.sqrt(i);
print('Y=',y);
```



Masala. Bir kilogramm konfetning narxi berilgan (haqiqiy son), 1, 2, ...,10 kg konfetning narxini chiqaruvchi dastur tuzing.

```
n=float(input("Bir kg konfet narxini kiriting:"));
for i in range(1,11):
s=i*n;
   print(s,"so‘m\n");
```

```
========== RESTART: D:/Madadjon/O`ktamov.M.py =
Bir kg konfet narxini kiriting:15000
15000.0 so‘m
30000.0 so‘m
45000.0 so‘m
60000.0 so‘m
75000.0 so‘m
90000.0 so‘m
105000.0 so‘m
120000.0 so‘m
135000.0 so‘m
150000.0 so‘m
```

Masala. a va b butun sonlari berilgan (a<b), a dan b gacha bo‘lgan barcha butun sonlar yig‘indisini chiqaruvchi dastur tuzing.

```
 a=int(input('a='));
b=int(input('b='));
s=0;
if a<b:
for i in range(a,b+1):
      s=s+i;
print('S=',s);
```

```
Edit  Shell  Debug  Options  Window  Help
Python 3.12.0 (tags/v3.12.0:0fb18b0, (
AMD64)] on win32
Type "help", "copyright", "credits" or

= RESTART: D:/Madadjon/O`ktamov.M.py
a=15
b=20
S= 105
```

Masala. n butun soni berilgan (n>0). Quyidagi yig‘indini hisoblovchi dastur tuzing: S=1+1/2+1/3+...+1/n.

```
n=int(input('n=')); s=0;
for i in range(1,n+1):
   s=s+1/i;
print('S=',s);
```

```
Edit  Shell  Debug  Options  Window  Help
Python 3.12.0 (tags/v3.12.0:0fb18b0, C
AMD64)] on win32
Type "help", "copyright", "credits" or

= RESTART: D:/Madadjon/O`ktamov.M.py
n=17
S= 3.439552522640758
```

Masala. n butun soni berilgan (n>0). Shu sonning kvadratini quyidagi formula asosida hisoblovchi dasturini tuzing. S=1+3+5+...+(2*n-1).

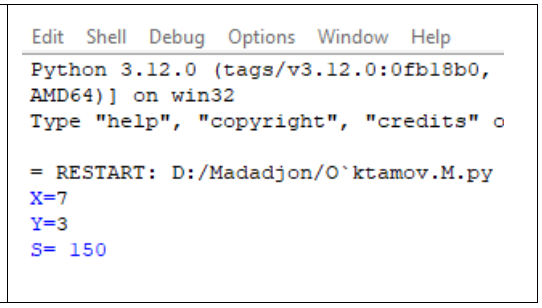
```
n=int(input('n=')); s=0;
for i in range(1,n):
   s=s+(2*i-1);
print('S=',s);
```

```
Edit  Shell  Debug  Options  Window  Help
Python 3.12.0 (tags/v3.12.0:0fb18b0, C
AMD64)] on win32
Type "help", "copyright", "credits" or

= RESTART: D:/Madadjon/O`ktamov.M.py
n=23
S= 484
```

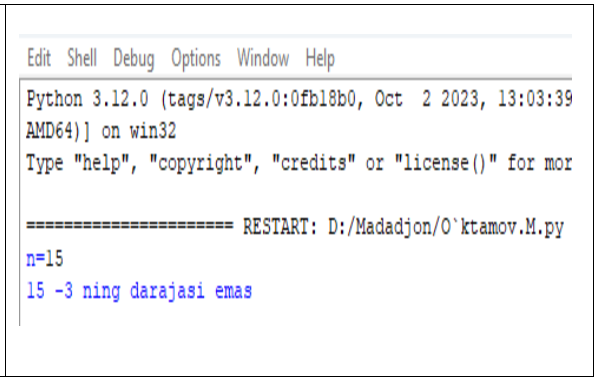
Masala. Tomonlari X va Y ga teng to‘rtburchak yuzini X=3 dan 5 gacha Y=1 dan 4 gacha 1 ga teng qadam bilan o‘zgarganda hisoblang.

```
        x=int(input('X='));
y=int(input('Y=')); s=0;
        for i in range(3,x+1):
            for j in range(1,y+1):
                s=s+i*j;
        print('S=',s);
```



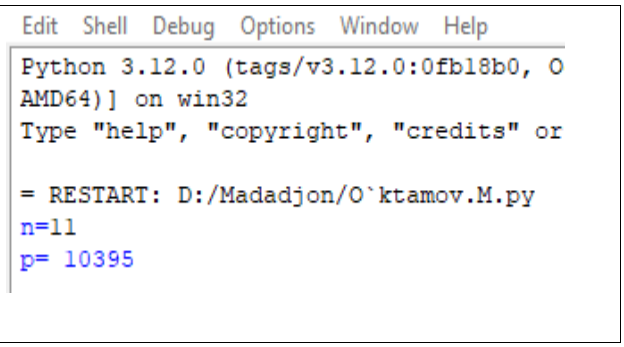
Masala. n butun soni berilgan (n > 0). Agar n soni 3 ning darajasi bo‘lsa “3 - ning darajasi”. aks xolda “3 - ning darajasi emas” degan natija chiqaruvchi dastur tuzing. Qoldiqli bo‘lish va bo‘lish amallarini ishlatmang.

```
        n=int(input('n='));
        i=1;
        while i<n:     i*=3;
        if n==i:
print(n,'-3 ning darajasi');
        else:
print(n,'-3 ning darajasi emas');
```



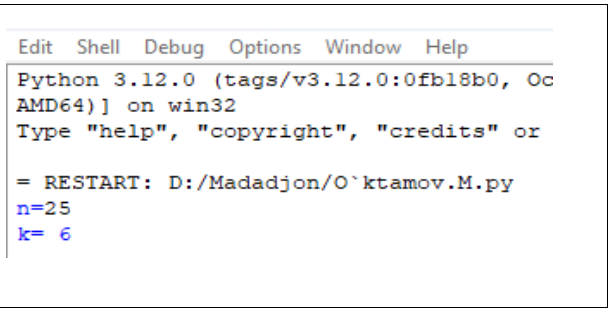
Masala. n natural soni berilgan (n > 0). Quyidagi ifodani hisoblovchi dastur tuzing: n!! = n * (n - 2) * (n - 4). Agar n juft bo‘lsa oxirgi ko‘paytuvchi 2, toq bo‘lsa 1 bo‘ladi.

```
n=int(input('n='));
        p=1;
        while 2<=n:
        p=p*n;
        n=n-2;
        print('p=',p);
```



Masala. n natural soni berilgan (n>0**).** Kvadrati n dan katta bo‘ladigan eng kichik butun k sonini (k^2>n) aniqlovchi dastur tuzing. Ildizdan chiqaruvchi funksiyadan foydalanmang.

```
        import math;
        n=int(input('n='));
        k=0;
        while
(math.pow(k,2)>n)==False:
            k+=1;
```



```
print('k=',k);
```

Masala. n natural soni berilgan (n>1**).** 3^k<=n shartni qanoatlantiruvchi eng katta butun к sonini aniqlovchi dastur tuzing.

```
n=int(input('n='));
k=0;
while n>=3:
    n/=3;
k+=1;
print('k=',k);
```

```
Edit  Shell  Debug  Options  Window  Help
Python 3.12.0 (tags/v3.12.0:0fb18b0, O
AMD64)] on win32
Type "help", "copyright", "credits" or

= RESTART: D:/Madadjon/O`ktamov.M.py
n=29
k= 1
```

Masala. n natural soni berilgan (n>1). (1+2+3+...+k)>=n shart bajariladigan eng kichik k sonini aniqlovchi dastur tuzing. 1 dan k gacha bo‘lgan yig‘indi ham ekranga chiqarilsin.

```
n=int(input('n='));
k=0; s=0;
while n>s:
k=k+1;
s=s+k;
print('k=',k,'=>s=',s,'\n');
```

```
Edit  Shell  Debug  Options  Window  Help
Python 3.12.0 (tags/v3.12.0:0fb18b0, O
AMD64)] on win32
Type "help", "copyright", "credits" or

= RESTART: D:/Madadjon/O`ktamov.M.py
n=14
k= 1 =>s= 1

k= 2 =>s= 3

k= 3 =>s= 6

k= 4 =>s= 10

k= 5 =>s= 15
```

Masala. a soni berilgan (a>1). (1+1/2+1/3+...+1/k)<=a shart bajariladigan eng katta к sonini aniqlovchi dastur tuzing. Yig‘indi ham ekranga chiqarilsin.

```
a=int(input('a='));
k=0; s=0;
while a>s:
k=k+1; s=s+1/k;
print('s=',s,'\n');
if s>a:
s-=1/k;   k-=1;
print('k=',k);
```

```
Edit  Shell  Debug  Options  Window  Help
Python 3.12.0 (tags/v3.12.0:0fb18b0, O
AMD64)] on win32
Type "help", "copyright", "credits" or

= RESTART: D:/Madadjon/O`ktamov.M.py
a=2
s= 1.0

s= 1.5

s= 1.8333333333333333

s= 2.083333333333333

k= 3
```

Masala. n va m butun musbat sonlari berilgan (n>m). n sonini m soniga bo‘lib butun hamda qoldiq qismlarini bo‘lish va qoldiqni olish amallarini ishlatmasdan topuvchi dastur tuzing.

```
        n=int(input('n='));
        m=int(input('m='));
        butun=0;
        while n>m:
        n=n-m;   butun+=1;
print("Butun=",butun,"\nQoldiq=",n);
```

```
Edit  Shell  Debug  Options  Window  Help
Python 3.12.0 (tags/v3.12.0:0fb18b0, O
AMD64)] on win32
Type "help", "copyright", "credits" or

= RESTART: D:/Madadjon/O`ktamov.M.py
n=26
m=7
Butun= 3
Qoldiq= 5
```

Masala. n butun soni berilgan (n>0). Uni bo‘lib butun va qoldiq qismlarini aniqlash orqali, berilgan son raqamlarini teskari tartibda chiqaruvchi dastur tuzing.

```
        import math;
n=int(input('n='));
        while n>0:
        i=n%10;
n=math.floor(n/10);
        print(i, end='');
```

```
Edit  Shell  Debug  Options  Window  Help
Python 3.12.0 (tags/v3.12.0:0fb18b0, (
AMD64)] on win32
Type "help", "copyright", "credits" o

= RESTART: D:/Madadjon/O`ktamov.M.py
n=178
871
```

Masala. a va b butun musbat sonlari berilgan. Berilgan sonlarning eng katta umumiy bo‘luvchisini aniqlovchi dastur tuzing.

```
        a=int(input('a='));
        b=int(input('b='));
        while a!=b:
        if a>b:   a=a-b;
        else:     b=b-a;
        print('EKUB=',a);
```

```
Edit  Shell  Debug  Options  Window  Help
Python 3.12.0 (tags/v3.12.0:0fb18b0, O
AMD64)] on win32
Type "help", "copyright", "credits" or

= RESTART: D:\Madadjon\O`ktamov.M.py
a=5
b=15
EKUB= 5
```

7-MAVZU. SIKL QADAMLARINI TASHLAB O‘TISH VA SIKLLARNI MUDDATIDAN OLDIN TUGATISH

Reja:

1. Break operatori va uning umumiy ko‘rinishi.
2. Continue operatori va uning umumiy ko‘rinishi;

Tayanch so‘zlar. Break operatori, continue operatori.

Python dasturlash tilida takrorlanuvchi jarayonlar qadamlarini tashlab o‘tish va takrorlanuvchi jarayonlarni muddatidan oldin tugatish imkoniyati ham mavjud. Python dasturlash tilida bunday imkoniyatlarni break va continue operatorlari amalga oshiradi. Break va continue operatorlarini ishlash prinsiplari Python va C++ tillarida bir xildir.

Break operatori va uning umumiy ko‘rinishi

Dasturlash tillarida algoritm bajarilayotgan vaqtda ma’lum bir sabablarga ko‘ra to‘satdan algoritm tarkibidagi takrorlanish o‘z ish faoliyatini to‘xtatish kerak bo‘lib qoladi. Bunday holatlarda **break** operatoridan foydalaniladi.

Break operatori ko‘p holatlarda takrorlanish jarayonlarida ishlatiladi. Break operatori vazifasi o‘zi turgan takrorlanish ish foaliyatini to‘xtatishdan iborat, agar break operatori dastur bosh tanasida joylashgan bo‘lsa dastur xatolik beradi break faqat takrorlanish tanasida bo‘ladi.

Python dasturlash tilida asosan takrorlanish jarayonida takrorlanishlar soni uning tarkibidagi ifodaga bog‘liq bo‘lib qoladi shunday vaziyatlarda takrorlanishni to‘xtatish uchun break operatoridan foydalanish maqsadga muvofiq.

Misol: Agar i 3 dan katta bo‘lsa, siklni tugating.

```
    for i in range(9):
if i > 3:
        break
print(i)
```

Misol: 1 dan n gacha sonlar tarkibidan, birinchi x ga karrali songacha bo‘lgan sonlarni 2 ga ko‘paytirib ekranga chiqaring. Dastur tuzishda birinchi x ga karrali son chiqqanda dastur to‘satdan to‘xtash kerak bo‘ladi, bu jarayonni break amalga oshiradi.

```
    n=input('n=')
    n=int(n)
```

```
x=input('x=')
x=int(x)
for i in range(1,n+1):
 if i%x!=0:
print(i*2)
else:       break;
```

Yuqoridagi dastur bajarilishi davrida break operatorigacha bo‘lgan operatorlar bajariladi qolganlari esa bajarilmasdan dastur takrorlanishdan chiqib ketadi. Chunki break operatori takrorlanish ichida joylashgan, shuning uchun dastur natijasi 1*2=2 dan boshlab 4*2=8 gacha bajariladi.

Misol: n dan m gacha sonlar tarkibidan, birinchi x ga karrali songacha bo‘lgan sonlar yig‘indisi va ko‘paytmasini ekranga chiqaring.

```
n=input('n=') n=int(n)
m=input('m=') m=int(m)
x=input('x=') x=int(x)
s=0 p=1
for i in range(n,m+1):     if i%x!=0:
s+=i        p*=i
else:  break;
print('s=',s)
print('p=',p)
```

Yuqoridagi dastur bajarilish davrida takrorlanish operatori faqat i=8 gacha bajariladi, lekin 8 hisobga olinmaydi chunki 8 to‘rtga karrali bu vaqtda break ishlaydi, chunki break operatori bajarilgandan so‘ng takrorlanishdan chiqib ketadi.

Continue operatori

Dasturlash tillarida ba’zi bir holatlarda dastur tarkibidagi buyruqlar faqatgina ma’lum bir qadamlarda tashlab keyingisi bajarilish kerak bo‘lgan holatlar ham mavjud. Bunday holatlarda break operatoridan foydalana olmaymiz, chunki break o‘zi turgan takrorlanishdan chiqib ketadi. Dastur tanasida bitta buyruq bajarilmasdan keyingisiga o‘tish uchun **continue** operatoridan foydalaniladi.

Continue operatori ham break operatoriga o‘xshaydi, lekin bunda faqat bitta bo‘yruqni cheklab o‘tadi. Asosan takrorlanish jarayonlarida ma’lum bir holatlarda

takrorlanishning ba’zi bir holatlari hisoblanmaslik kerak bo‘lganda continue operatoridan foydalanish maqsadga muvofiq bo‘ladi.

Takrorlanish jarayonida takrorlanishning biror bir qadamini tashlab ikkinchisiga o‘tish uchun continue buyrug‘idan foydalaniladi.

Misol: Agar i o‘zgaruvchisi 3 bo‘lsa, iteratsiyani o‘tkazib yuboring, lekin keyingi iteratsiya bilan davom eting.

```
    for i in range(9):
  if i == 3:
        continue
  print(i)
```

```
Edit  Shell  Debug  Options  Window  Help
Python 3.12.0 (tags/v3.12.0:0fb18b0, O
AMD64)] on win32
Type "help", "copyright", "credits" or
= RESTART: D:\Madadjon\O`ktamov.M.py
0
1
2
4
5
6
7
8
```

Misol: while siklida davom kalit so‘zidan foydalaning:

```
    i = 0
    while i < 9:
      i += 1
      if i == 3:
        continue
  print(i)
```

```
Edit  Shell  Debug  Options  Window  Help
Python 3.12.0 (tags/v3.12.0:0fb18b0,
AMD64)] on win32
Type "help", "copyright", "credits" o
= RESTART: D:\Madadjon\O`ktamov.M.py
1
2
4
5
6
7
8
9
```

Misol: 1 dan n gacha sonlar ichida 3 ga karrali bo‘lmaganlarini ikkiga ko‘paytirib ekranga chiqaring.

```
    n=input('n=') n=int(n)
    for i in range(1,n+1):
    if i%3!=0:
    print(2*i)
else:    continue
```

```
Edit  Shell  Debug  Options  Window  Help
Python 3.12.0 (tags/v3.12.0:0fb18b0
AMD64)] on win32
Type "help", "copyright", "credits"
= RESTART: D:\Madadjon\O`ktamov.M.p
n=6
2
4
8
10
```

Yuqoridagi dasturda for operatori 3 ga karrali bo‘lgan sonlarni hisobga olmasdan keyingisiga o‘tib ketadi. Bu esa continue operatori imkoniyatini amalga oshiradi.

Biz siklimizga shart qo‘shishimiz mumkin, agar nomer 9 bo‘lsa, sikldan chiqing.

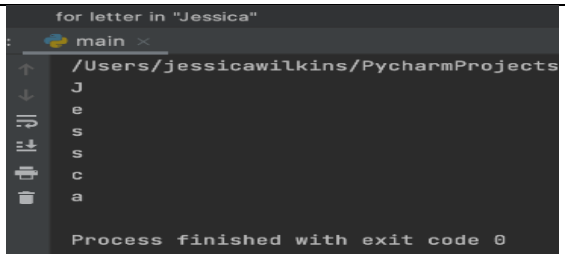
```
num = 5
while num < 20:
   print('Current number :', num)
   num = num + 1
   if num == 9:
      break
```

```
Run:  main
/Users/jessicawilkins/PycharmProjects
Current number : 5
Current number : 6
Current number : 7
Current number : 8

Process finished with exit code 0
```

Agar biz yoki siklning continue joriy iteratsiyasini o‘tkazib yuborishimiz va keyingi iteratsiyaga o‘tishimiz kerak bo‘lsa, biz buni quyidagicha bajaramiz.

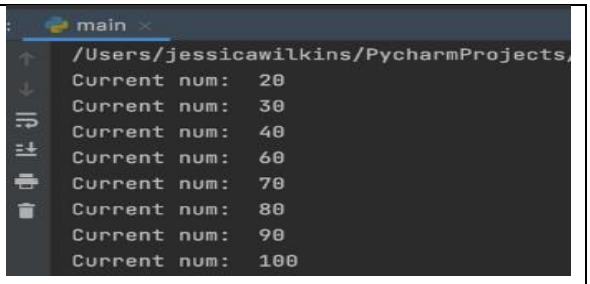
```
for letter in "Jessica":
    if letter == "i":
        continue
    print(letter)
```



E'tibor bering, "i" harfi konsolda chop etilmagan va continue bu iteratsiyani o‘tkazib yuborgan.

Biz ushbu keyingi misolda sikl yordamida raqamlarni 10 ga oshib bosamiz. Biz siklga shart qo‘shamiz, agar raqam 50 bo‘lsa, bu takrorlashni o‘tkazib yuboring va keyingisiga o‘ting.

```
num = 10
while num < 100:
    num = num + 10
    if num == 50:
        continue
    print("Current num: ", num)
```



Ko‘rib turganingizdek, yuqoridagi shart tufayli 50 raqami konsolda chop etilmaydi.

Xulosa qilib aytadigan bo‘lsak, Pythondagi va iboralari joriy siklning qismlarini o‘tkazib yuborish uchun continue yoki sikldan butunlay chiqib ketish uchun esa break amali ishlatiladi.

8-MAVZU. PYTHON DASTURLASH TILIDA SHARTLI TAKRORLANUVCHI JARAYONLAR DASTURLASH

Reja:

1. Shartli sikl operatori.
2. While operatori va uning umumiy ko‘rinishi.

Tayanch so‘zlar. Shartli sikl, shart, while, shartli takrorlanish operatori.

Shartli sikl operatori

Bazi bir masalalarni yechish algoritmlari tarkibida takrorlanishlar qandaydir shartlarga asosan bajariladi. Har bir takrorlanish jarayoni bajarilish qadamida shart tekshirilib o‘tib borilaveradi, qachonki shart yolg‘on bo‘lgandagina takrorlanish jarayoni to‘xtatiladi. Masalan yig‘indisi *S* ga teng bo‘lgan natural sonlar sonini topish yoki umumiy hadi *n* dan kichik bo‘lgan cheksiz kamayuvchi geometrik progressiyani hadlar sonini topish kabi masalalarda shartli takrorlanuvchi operatorlardan foydalaniladi. Agar algoritm tarkibidagi bir necha marta takrorlanishi kerak bo‘lgan buyruqlarni takrorlanuvchi jarayonlar asosida dasturlash tillarida tasvirlanmasa, bu buyruqlarni barchasini bajarish murakkablashadi.

Shartli takrorlanuvchi algoritmlarni shartli takrorlanuvchi jarayonlar ham deb ataymiz.

Tarif: Agar takrorlanishlar soni ma’lum bir shartlar asosida aniqlansa, bunday jarayonlar shartli takrorlanuvchi jarayonlar deyiladi.

Shartsiz o‘tish operatori va tarmoqlanuvchi operatorlar yordamida ham shartli takrorlanuvchi jarayonlarni dasturlash imkoniyati mavjud. Lekin bunday holatlarda bitta amalni bajarish uchun bir nechta operatorlarni ishlatish kerak bo‘ladi. Shartli takrorlanuvchi operatorlar bajarilish holatlariga qarab turlarga ajratiladi. Shart asosida takrorlanuvchi jarayonlarni python dasturlash tilida shartli sikl operator yordamida amalga oshiriladi.

While operatori

Shart oldi takrorlanuvchi jarayonlar bajarilish holati har bir takrorlanish oldidan shart tekshirilib keyin takrorlanish tanasidagi operatorlar bajariladi. Agar takrorlanish

holati boshidan shart yolg‘on qiymat qabul qilsa, takrorlanish bir marta ham bajarilmaydi.

Shart oldi takrorlanuvchi operatorlarning python dasturlash tilida ifodalash uchun while operatori yordamida tasvirlanadi.

Takrorlash strukturasi bir ifoda yoki operatorlarni ma'lum bir shart to‘g‘ri (true) bo‘lishi davomida qaytarish imkonini beradi. Qaytarilayotgan ifoda shartga ta'sir ko‘rsatishi kerak. Ma'lum bir vaqt o‘tgandan keyin shart falsega o‘zgartilishi kerak. Bo‘lmasa while (davomida) ish jarayoni tugatilmaydi va cheksiz bajarilib qoladi, bu esa mumkin emas. While faqat o‘zidan keyin kelgan ifodaga ta'sir qiladi. Agar biz bir guruh amallarni qaytarmoqchi bo‘lsak, : dan keyin enter bilan operatorlarni yozishimiz kerak. Shart takrorlanishning boshida tekshirilganligi sababli, agar shart noto‘g‘ri bo‘lib chiqsa, takrorlanish bajarilmasligi ham mumkin.

Ta’rif: Agar shartli takrorlanuvchi jarayonlar tarkibidagi shart takrorlanishdan oldin tekshirilsa, shart oldi takrorlanuvchi jarayonlar deyiladi.

Takrorlanuvchi operator tarkibiga beriladigan shart tahlil qilinib yozilish kerak, chunki shart hech qachon yolg‘on qiymat qabul qilmasa, dastur cheksiz ishlashga to‘g‘ri keladi. Takrorlanish hech qachon cheksiz bo‘lishi mumkin emas, aks holda algoritmning diskretlik xossasi buziladi.

Shart oldi takrorlanish operatori yani while operatorining umumiy ko‘rinishi quyidagicha.

while <shart>:

operatorlar

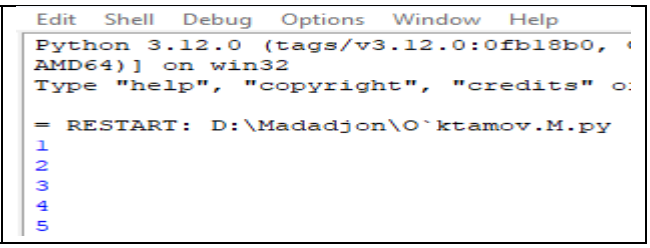
Agar shart chin qiymat qabul qilib tursa operatorlar bajarilaveradi, qachonki shart yolg‘on bo‘lgandagina takrorlanish o‘z ish faoliyatini to‘xtatadi.

While operatori tarkibidagi shart yolg‘on qiymat qabul qilganda operatorlar bajarilmasdan qoladi, agar shart chin qiymat qabul qilganda operatorlar bajariladi. Ba’zi hollarda shart takrorlanish boshidan yolg‘on qiymat qabul qiladi, bunda takrorlanish bir marta ham bajarilmaydi. Shart chin qiymat qabul qilib, lekin takrorlanish tanasida shart tarkibi o‘zgartirilmasa, takrorlanish cheksiz bo‘lib qoladi.

While da sikl

while siklida avval shart tekshiriladi so‘ngra sikl tanasi ishga tushadi. Bunga quyidagicha misol keltirishimiz mumkin.

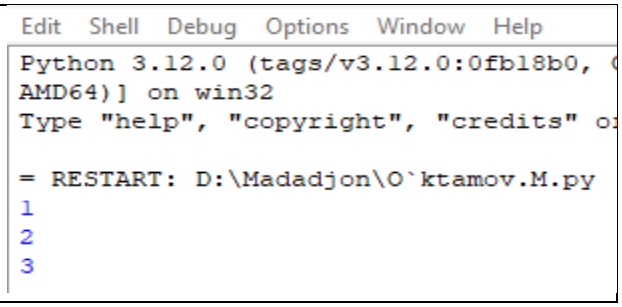
```
i = 1
while i < 6:
  print(i)
  i += 1
```



While da break

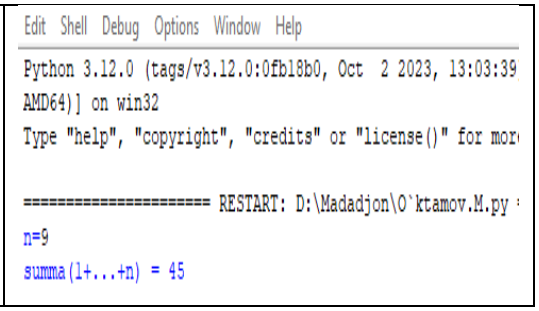
Break operatori bu siklni ushbu buyruqdan keyingi amallarni bajarmaslikni ta'minlaydi.

```
i = 1
while i < 6:
  print(i)
  if i == 3:
    break
  i += 1
```



Masalan:

```
sum = 0
n = int(input("n="))
i = 1 while i <= n:
sum = sum + i
i += 1
print("summa(1+...+n) =", sum)
```



Yuqoridagi misolda 1 dan *n* gacha bo‘lgan sonlar yig‘indisi hisoblash dasturi *while* operatori yordamida amalga oshirilgan. E`tibor berilsa *while* operatorining instruksiyalari undan keyingi qatorda bitta xat boshi tashlab yozilgan. Ushbu holatda *while* operatori 2 ta instuksiyalardan tashkil topgan (*sum = sum + i* va *i +=1*).

Misol: Python so‘zi ekranga n mart chiqarilsin.

Bu masalani for sikl operatori yordamida ham ifodalash mumkin, lekin while operatorining mohiyatini o‘rganish uchun oddiy masala yordamida qaraymiz.

```
n=input('n=')
n=int(n)
i=1
while i<=n:
    print(i,'python')
    i+=1
```

```
Edit  Shell  Debug  Options  Window  Help
Python 3.12.0 (tags/v3.12.0:0fb18b0, Oc
AMD64)] on win32
Type "help", "copyright", "credits" or

= RESTART: D:\Madadjon\O`ktamov.M.py
n=3
1 python
2 python
3 python
```

While operatorini dasturlash tarkibida ishlatish vaqtida doimo takrorlanish tarkibidagi shart bilan tekshiriladigan bitta o‘zgaruvchi olish kerak. Bu masalada shart bilan tekshirish uchun *i* o‘zgaruvchisi tanlandi. Takrorlanish sonini *n* orqali *i* bilan solishtirish natijasida aniqlanadi.

While operatorining ishlash jarayoni yuqoridagi misolda quyidagicha.

Boshlang‘ich holatda i=1 shart i<=n(1<=3) chin

Qadam 1: Python so‘zi ekranda chiqariladi i=2; shart i<=n(2<=3)

Qadam 2: Python so‘zi ekranda chiqariladi i=3; shart i<=n(3<=3)

Qadam 3: Python so‘zi ekranda chiqariladi i=4; shart i<=n(4<=3) yolg‘on takrorlanish to‘xtatiladi.

Misol. n berilganda k!<=n shartni qanoatlantiruvchi eng katta k sonini aniqlang.

```
n=input('n=');
n=int(n);
p=1;
k=1;
while(p<=n):
    k+=1
    p=p*k
print(k-1)
```

```
Edit  Shell  Debug  Options  Window  Help
Python 3.12.0 (tags/v3.12.0:0fb18b0, Oct
AMD64)] on win32
Type "help", "copyright", "credits" or '

= RESTART: D:\Madadjon\O`ktamov.M.py
n=15
3
```

Bu masalani Python dasturlash tilidagi ko‘rinishiga e’tibor bersak oxirida *print(k-1)* operatori yozilgan, buni mohiyati shart chin qiymatida bitta qadam ortiq bajariladi, shuning uchun *k-1* holat bo‘yicha chiqariladi. Bu jarayonni n= 7 qiymat berib, qo‘lda test qilib ko‘rsangiz tushunish oson bo‘ladi.

While operatori tarkibidagi shart ba’zi hollarda o‘zgarmas qiymat ko‘rinishda ham beriladi, bunda shart 0 bilan solishtiriladi, agar qiymat ortib borsa, dastur cheksiz takrorlanish mumkin, bunday holatlarda o‘zgarmas qiymatni kamaytirish kerak.

```
n=input('n=')
n=int(n)
while n:
    n+=1
    print(n)
```

```
Edit  Shell  Debug  Options  Window  Help
Python 3.12.0 (tags/v3.12.0:0fb18b0,
AMD64)] on win32
Type "help", "copyright", "credits" o

= RESTART: D:\Madadjon\O`ktamov.M.py
n=-10
-9
-8
-7
-6
-5
-4
-3
-2
-1
0
```

Yuqoridagi dastur tarkibidagi shart n faqat 0 bilan solishtiriladi 0 dan farqli bo‘lsa, takrorlanish bajarilaveradi, demak takrorlanish -10 dan boshlab toki 0 gacha bajariladi.

Bunday holatlarda takrorlanish cheksiz bo‘lib qolish ham mumkin, shart 0 bilan solishtirishni e’tiborga olgan holda, shart yozilish kerak. Takrorlanish cheksiz bo‘lgan holatini quyidagi dastur orqali tekshiramiz.

Quyidagi masalada 1 dan n gacha sonlarning yig‘indisini while da hisoblaymiz:

```
    n=int(input('n=')); s=0;
i=1; while i<=n:
    s=s+i;    i=i+1;
    print('while=',s);
```

```
Edit  Shell  Debug  Options  Window  Help
Python 3.12.0 (tags/v3.12.0:0fb18b0, Oc
AMD64)] on win32
Type "help", "copyright", "credits" or

= RESTART: D:\Madadjon\O`ktamov.M.py
n=50
while= 1275
```

9-MAVZU. PYTHON DASTURLASH TILIDA FUNKSIYALARNI YARATISH VA ULARDAN FOYDALANISH

Reja:

1. Qism dasturlar;
2. Funksiya tanasini faollashtirish;
3. Global va lokal o‘zgaruvchilar;
4. Funksiyaga argument berishni soddalashtirish.

Tayanch so‘zlar. Qism dasturlar, funksiya, global, bir qiymat, argument.

Python dasturlash tili tarkibida dastur tuzish vaqtida ma’lum bir jarayonlar bir necha marta bajarilishi yoki bir necha marta murojaat qilinishi mumkin. Dasturlash tillari tarkibida bir necha marta bajarilish kerak bo‘lgan jarayonlarni bir marta tasvirlab, keyin shu dasturga murojaat qilish imkoniyati mavjud. Dasturlash tillari tarkibida bir necha marta bajarilishi mumkin bo‘lgan holatlarni qism dastur sifatida e’lon qilish va kerakli joyga shu qism dasturga murojaat qilish mumkin.

Ta’rif: Dasturlash tilida yaratilgan dastur tarkibida ma’lum bir vazifani bajaruvchi kichik dasturlar qism dasturlar deyiladi.

Qism dasturlarning mohiyati berilgan masala tarkibi ma’lum bir vazifani bajarish kerak bo‘ladi, lekin masala tarkibida ham kichik bir vazifani bajarish kerak bo‘ladi shunday vazifalarni qism dastur yordamida hal etish mumkin. Bunday masalalar asosan matematik masalalar tarkibida ko‘p bo‘lishi mumkin, yoki boshqa sohalarda ham uchrab turadi. Masalan biror bir tashkilotning ma’lumotlar bazasi berilganda uning tarkibidagi xodimlarning oylik maoshini hisoblash jarayonini qism dastur yordamida hisoblash maqsadga muvofiq. Agar tashkilot ma’lumotlar bazasi tarkibidagi xodimlarning oylik maoshini qism dastur yordamida hisoblanmasa, har bir xodim uchun oylik maoshini hisoblash jarayonini keltirish kerak, qism dasturdan foydalansa har bir xodim uchun qism dasturga murojaat qilib qo‘yiladi. Dastur tuzish vaqtida qism dasturlardan qachon foydalanamiz, agar siz hal etadigan masala yoki muammo tarkibida ma’lum bir jarayonlar ikki va undan ortiq sodir bo‘lsa, o‘sha jarayonni qism dastur sifatida e’lon qilish kerak va kerakli joyda qism dasturga murojaat qilish kerak. Masalan, quyidagi masalaga e’tibor bering.

$$y = \frac{a^{1+2+\cdots+n}}{(1+2+\cdots+n)^x}$$

Bu masala tarkibiga e’tibor qaratsak, birdan n gacha bo‘lgan sonlar yig‘indisi bir necha marta bajarilyapti, bu masalani hal etish uchun tuziladigan dastur tarkibida birdan n gacha bo‘lgan sonlar yig‘indisini hisoblash jarayonini qism dastur qilib e’lon qilish kerak. Agar qism dastur sifatida e’lon qilinsa, masalani hal etishda qism dasturga ikki marta murojaat qilish asosida berilgan masalani hal etish mumkin.

Yuqorida keltirilgan masalalarga o‘xshash masalalarni hal etish usulini samarasini oshirish uchun qism dasturlardan foydalaniladi. Qism dasturlar python dasturlash tilida qisqacha qilib funksiyalar deb ataladi. Demak, funksiyalar ma’lum bir vazifani bajaruvchi dastur tarkibdagi qism dasturlar ekan.

Ta’rif: Dasturlash tilida tuzilgan dastur tarkibidagi ma’lum bir vazifalarni bajaruvchi qism dasturlar funksiyalar deyiladi.

Dastur ishlash jarayoni samarasi asosan ikki tur bo‘yicha oshiriladi, ya’ni xotira hajmi va ishlash tezligi bo‘yicha. Dasturning ishlash tezligini oshirish uchun, albatta, dastur tarkibi sodda va ixcham bo‘lishi talab etiladi. Dastur tarkibi sodda va ixcham bo‘lishi uchun imkoniyat boricha qism dasturlardan foydalanish kerak bo‘ladi. Funksiyalar python dasturlash tilida ma’lum bir vazifani bajaruvchi dastur tarkibidagi kichik dasturlar, demak, funksiyani hosil qilish uchun dastur tarkibida ma’lum bir vazifalar bir necha marta bajarilish kerak bo‘ladi. Python dasturlash tilida har qanday funksiya hech bo‘lmaganda bitta natija qaytaradi.

Dastur tarkibidagi funksiyaga uning nomi bilan murojaat qilinadi, dastur bajarilish vaqtida funksiya nomi uchrasa, kompilyator shu funksiyaning tanasiga murojaat qilib natijani funksiya nomiga qaytaradi va dastur keyingi qadamlarni bajaradi. Python dasturlash tilida funksiyalar asosan ikki guruhga ajratiladi, ya’ni standart va standart bo‘lmagan funksiyalar. Standart funksiyalar python dasturlash tili tarkibida biror bir kutubxona tarkibiga joylashtirilgan bo‘ladi. Standart funksiyalar haqida mazkur qo‘llanmaning 1-mavzusida batafsil yoritilgan. Standart bo‘lmagan funksiyalar dastur tarkibida yaratiladi va unga nomi bilan murojat qilinadi. Python dasturlash tilida dasturlashning asosiy qismlaridan biri funksiyalardir. Funksiyalarning

foydasi shundaki, katta hajmli masala bir necha kichik bo‘laklarga ajratilib, har qaysiga alohida funksiya yozilganda, masala yechish algoritmi ancha soddalashadi. Bunda dasturchi yozgan funksiyalar pythonning standart kutubxonasi va boshqa firmalar yozgan kutubxonalar ichidagi funksiyalar bilan birlashtiriladi. Bu esa ishni bir muncha osonlashtiradi. Ko‘p holda dasturda takroran bajariladigan amallarni funksiya sifatida yozish va kerakli joyda ushbu funksiyani chaqirish mumkin. Funksiyani programma tanasida ishlatish uchun u chaqiriladi, yani uning ismi yoziladi va unga kerakli argumentlar beriladi. () qavslar ushbu funksiya chaqirig‘ini ifodalaydi. Masalan: foo(); k = square(l); Demak, agar funksiya argumentlar olsa, ular () qavs ichida yoziladi. Argumentsiz funksiyadan keyin esa () qavslarning o‘zi qo‘yiladi.

Funksiya tanasini faollashtirish

Python dasturlash tilida funksiyalardan foydalanish uchun, albatta, funksiyalarni dastur tarkibida alohida yozilish kerak. Funksiyalar python tilining ixtiyoriy joyida yozilishi mumkin. Funksiyalar dasturchi ishini juda yengillashtiradi. Funksiyalar yordamida programma modullashadi, qismlarga bo‘linadi. Bu esa keyinchalik dasturni rivojlantirishni osonlashtiradi. Dastur yozilish davrida xatolarni topishni yengillashtiradi. Funksiya nomi funksiya bajaradigan vazifadan kelib chiqqan holda qo‘yilishi maqsadga muvofiq, chunki dastur tuzuvchi mutaxassislar uchun umumiy holda tushunish oson bo‘lishi kerak. Masalan, ketma-ketliklarni yig‘indisini hisoblash funksiyasiga **sum(),** to‘rtinchi darajali ildizni hisoblash uchun **sqrt4(),** sonning ekubini hisoblash uchun **ekub(),** sonning ekukini hisoblash uchun **ekuk()** va faktorialni hisoblash uchun **fakt()** deb nomlash maqsadga muvofiq bo‘ladi. Funksiya tarkibiga kiritilishi kerak bo‘lgan o‘zgaruvchi argumentlar, albatta, qavs ichida vergul bilan ajratilib yozilishi kerak.

Funksiya tanasini tasvirlash jarayoni ikki qismdan iborat bo‘ladi, ya’ni funksiya sarlavhasi va funksiya tanasidan iborat bo‘ladi. Funksiya tanasini tasvirlash jarayonida funksiya sarlavhasidan keyin nuqtali vergul qo‘yilmaydi va *def* xizmatchi so‘z orqali boshlanib, natija **return** xizmatchi so‘zidan keyin probel bilan yoziladi.

Funksiya tanasini dastur tarkibiga yozishda funksiyaga murojaat qilishdan bir qadam oldin ixtiyoriy joyda yozish mumkin, murojaatdan keyin yozilsa dastur xatolik

qaytaradi. Funksiya tanasi tarkibi yozilishida xuddi boshqa dastur tuzilishi kabi unda ishlatiladigan o‘zgaruvchilar e’lon qilinadi, buyruqlar nuqtali vergul yordamida ajratilish shart emas. Funksiyalarni python dasturlash tilida yozilish jarayonining umumiy ko‘rinishi quyidagicha bo‘ladi:

def <funksiya nomi>(<argumentlar>):

funksiya tanasi

return natija

Funksiyalar tanasini tasvirlashda funksiya qaytaradigan qiymat yoki ifoda return so‘zidan keyin probel bilan yozilishi kerak.

Funksiyalar parametrlarni, ya’ni funksiyaga berilishi mumkin bo‘lgan qiymatlarni qabul qiladi va ular ustida biror bir amal bajarishi mumkin. Bu parametrlar o‘zgaruvchilarga o‘xshaydi. Faqat bulardan farqi esa bu o‘zgaruvchilarning qiymati funksiyani chaqirish vaqtida o‘rnatilishidir. Funksiya ish boshlagan vaqtda ularga qiymat biriktirilgan bo‘ladi.

Parametrlar funksiya aniqlanayotgan vaqtda qavs ishida vergul bilan ajratilgan holda beriladi. Qiymatni ularga funksiyani chaqirganimizda biriktiramiz. Funksiya e’lon qilinayotgan vaqtda ko‘rsatilgan nomlar **parametrlar**, funksiyani chaqirayotganimizda unga berilgan qiymatlar **argumentlar** deyiladi.

Funksiya – bu ko‘p marta ishlatiladigan dastur bo‘lagidir. Funksiyalar ma’lum buyruqlar blokini ko‘rsatilgan nom bilan saqlash va shu blokni dasturning istalgan joyida, istalgan miqdorda bajarish imkonini yaratadi.

Funksiyalar **def** zaxira so‘zi orqali aniqlanadi. Bu so‘zdan keyin funksiya **nomi**, undan keyin qavs va ana shu qavs ichida bir necha o‘zgaruvchilarni ko‘rsatish mumkin bo‘ladi va oxirida esa ikki nuqta (:) yoziladi. Shulardan so‘ng funksiyani tashkil qiluvchi buyruqlar bloki yoziladi. Quyidagi misolda buning oson ekanligini ko‘rish mumkin. Sodda funksiyaga misol.

```
def ikta_sum():
 a=int(input('a='));
b=int(input('b='));
 sum=a+b;
 return sum;
 print(ikta_sum());
```

```
Edit  Shell  Debug  Options  Window  Help
Python 3.12.0 (tags/v3.12.0:0fb18b0, O
AMD64)] on win32
Type "help", "copyright", "credits" or

= RESTART: D:\Madadjon\O`ktamov.M.py
a=50
b=25
75
```

Bu misolda ikki son yig‘indisini hisoblovchi funksiya ko‘rsatilgan. Bu funksiya birorta argument qabul qilmaydi, yig‘indini hisoblab natijani chiqaradi. Shundan so‘ng **print** operatori tanasida natijani chiqarish uchun chaqiriladi. Bu funksiyani shunday o‘zgartiramizki, qiymatni qaytarmasdan chiqarsin. Buning uchun **print** operatorini funksiya tanasiga kiritish etarli:

```
def ikkita_sum():
 a=int(input('a='));
b=int(input('b='));
 sum=a+b;
 print('sum=',sum);
 ikkita_sum();
```

```
Edit  Shell  Debug  Options  Window  Help
Python 3.12.0 (tags/v3.12.0:0fb18b0, Oct
AMD64)] on win32
Type "help", "copyright", "credits" or '

= RESTART: D:\Madadjon\O`ktamov.M.py
a=20
b=30
sum= 50
```

a va **b** o‘zgaruvchilarni argument sifatida e’lon qilishimiz mumkin, bu holda funksiya tanasida ularni tariflash talab etilmaydi.

```
def ikkita_sum(a,b):
 sum=a+b;
 print('sum=',sum);
a=int(input('a='));
 b=int(input('b='));
ikkita_sum(a,b);
```

```
Edit  Shell  Debug  Options  Window  Help
Python 3.12.0 (tags/v3.12.0:0fb18b0, Oct
AMD64)] on win32
Type "help", "copyright", "credits" or '

= RESTART: D:\Madadjon\O`ktamov.M.py
a=250
b=300
sum= 550
```

Argument orqali uzatilgan qiymatni o‘z ichiga oluvchi o‘zgaruvchi, funksiya **parametri** deyiladi.

Ko‘rilgan misollarda funksiya argumenti qiymati bo‘yicha uzatiladi, ya’ni argumentlar funksiya ichida o‘zgarib, ular funksiya tashqarisidagi qiymatlarga ta’sir qilmaydi:

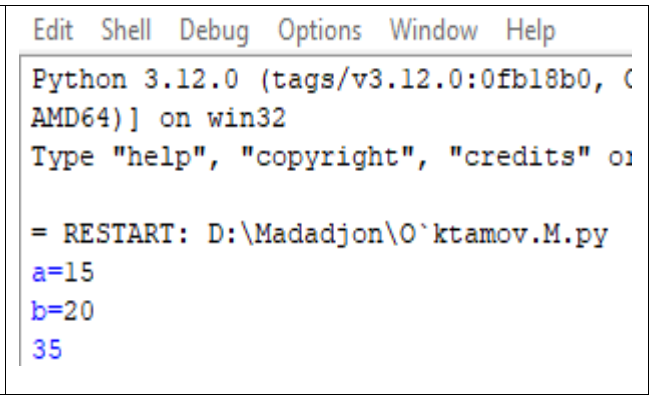
```
def ikkita_sum(a):
 sum=a+10; return
sum; a=int(input('a='));
b=int(input('b='));
print(ikkita_sum(a));
 print(b);
```

```
Edit  Shell  Debug  Options  Window  Help
Python 3.12.0 (tags/v3.12.0:0fb18b0, (
AMD64)] on win32
Type "help", "copyright", "credits" o

= RESTART: D:\Madadjon\O`ktamov.M.py
a=25
b=10
35
10
```

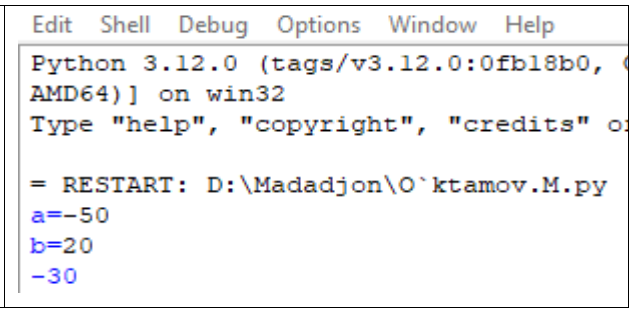
Misol. Python dasturlash tilida ikki sonning yig‘indisini hisoblash uchun yig() fuksiya yarating va unga murojaat qilishni tasvirlang.

```
def yig(a,b):
y=a+b
return y
a=input('a=')
b=input('b=')
a=int(a)
b=int(b)
z=yig(a,b)
print(z)
```



Yuqoridagi masalani hal etish uchun *yig(a,b)* funksiyasi yaratildi, funksiya tanasini dastur boshida tasvirlandi. Dastur bajarilish vaqtida kompilyator dastur tarkibida *yig(a,b)* funksiyasini uchratganda bajarilish qadami *yig(a,b)* funksiya tanasiga o‘tib natijani hisoblab qaytib keladi va bajarilish qadami buyruqlar ketma-ketligi bo‘yicha bajariladi. Yuqoridagi masalani ikkinchi ko‘rinishda ham bajarish mumkin, ya’ni funksiya tanasi dasturni ixtiyoriy joyida keltirilishi mumkin.

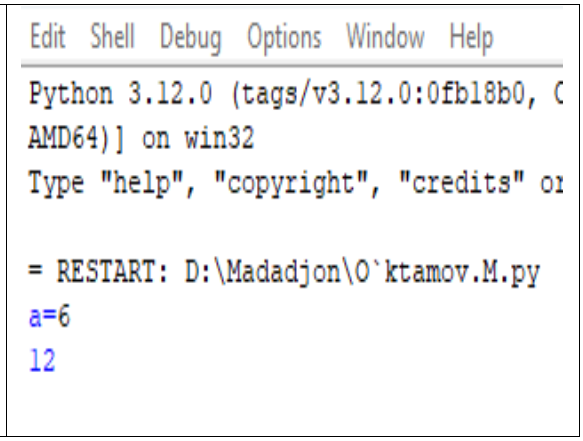
```
a=input('a=')
b=input('b=')
def yig(a,b):
 y=a+b    return y a=int(a)
b=int(b)
z=yig(a,b)
print(z)
```



Ikki sonning yig‘indisini hisoblash uchun keltirilgan dasturning ikkinchi ko‘rinishi faqat *yig(a,b)* funksiyasi dastur ichida yozilgan. Return xizmatchi so‘zidan keyin funksiya qaytaradigan qiymat natijasini ifodalovchi ifodani ham yozish mumkin.

Masala: Sonning natural bo‘luvchilar yig‘indisini aniqlash uchun bul_yig(x) funksiyasini yarating.

```
def bul_yig(x):
  s=0
  for i in range(1,x+1):
    if(x%i==0):
       s=s+i;
  return s
a=input('a=')
a=int(a);
z=bul_yig(a)
print(z)
```



Yuqoridagi masalalarga e’tibor qaratsak masala tarkibidagi funksiyalardan dastur tarkibida ixtiyoriy joyida ixtiyoriy marta foydalanish mumkin.

Global va lokal o‘zgaruvchilar

Python dasturlash tilida tuziladigan dasturlar tarkibida bir nechta o‘zgaruvchilardan foydalaniladi. Python dasturlash tilida funksiyalar mavzusidan keyin o‘zgaruvchilar ikki turga ajratiladi, ya’ni global va lokal o‘zgaruvchilar.

Ta’rif: Dastur tarkibining ixtiyoriy joyida foydalanish mumkin bo‘lgan o‘zgaruvchilar **global o‘zgaruvchilar** deyiladi.

Global o‘zgaruvchilar dasturning ixtiyoriy qismida o‘z qiymatini saqlaydi, hattoki, dasturning ixtiyoriy joyida o o‘z qiymatini ushlab qoladi. Global o‘zgaruvchilar qism dasturning tashqarisida faollashtiriladi.

Python dasturlash tilidagi funksiyalar tarkibidagi o‘zgaruvchilar global hisoblanmaydi. Global bo‘lmagan o‘zgaruvchilar faqatgini o‘z qism funksiya tarkibiga tegishli bo‘ladi.

Ta’rif: Python dasturlash tilidagi funksiyalar tarkibidagi o‘zgaruvchilar **lokal o‘zgaruvchilar** deyiladi.

Dastur tarkibidagi qism funksiyalar tarkibidagi barcha o‘zgaruvchilar lokal o‘zgaruvchilar hisoblanadi, funksiya tarkibidagi o‘zgaruvchilar faqatgina funksiyaning tarkibi ichida o‘rinli bo‘ladi. Global va lokal o‘zgaruvchilarni aniqlash uchun quyidagi orqali aniqlaymiz.

O‘zgaruvchilar funksiyalarda lokal ko‘rinish sohasiga ega. Bu shuni bildiradiki, xatto lokal va tashqi o‘zgaruvchilar bir xil nomga ega bo‘lsa ham, lokal o‘zgaruvchi o‘zgarishi tashqi o‘zgaruvchiga ta’sir qilmaydi.

```
    def get_sum():
a=int(input("lokal o‘zgaruvchi a="));
    print(a);
b=int(input("global o‘zgaruvchi b="));
    get_sum();
    print(b);
```

```
Python 3.12.0 (tags/v3.12.0:0fb18b0,
AMD64)] on win32
Type "help", "copyright", "credits" c

= RESTART: D:\Madadjon\O`ktamov.M.py
global o‘zgaruvchi b=30
lokal o‘zgaruvchi a=10
10
30
```

Lokal o‘zgaruvchini global qilish mumkin, agar uning nomi oldidan **global** kalit so‘zi ko‘rsatilsa. Agar tashqi o‘zgaruvchi **global** sifatida e’lon qilingan bo‘lsa, unga ixtiyoriy funksiyadan murojaat qilish mumkin:

```
    def get_sum():
       global a;
    b=int(input('b='));
    a=int(input('a='));
    print(a);
    print(b);
    get_sum();
```

```
Python 3.12.0 (tags/v3.12.0:0fb18b0, O
AMD64)] on win32
Type "help", "copyright", "credits" or

= RESTART: D:\Madadjon\O`ktamov.M.py
b=50
a=40
40
50
```

O‘zgaruvchi xayot davri deb u mavjud bo‘lgan dastur bajarilish intervali tushuniladi. Lokal o‘zgaruvchilar ko‘rinish sohasi funksiya bo‘lgani uchun, ularning xayot davri ular ta’riflangan funksiya bajarilish vaqti bilan belgilanadi. Bu shuni bildiradiki, har xil funksiyalarda bir - biridan mustaqil ravishda bir xil nomli o‘zgaruvchilar ishlatilishi mumkin. Lokal o‘zgaruvchi har gal funksiya chaqirilganda yangidan initsializatsiya qilinadi, shuning uchun quyidagi misolda keltirilgan sanovchi funksiyaning qaytaruvchi qiymati har gal 1 ga teng bo‘ladi:

```
    def a():
     a=int(input('a='));
    return a+1;
    print(a());
```

```
Python 3.12.0 (tags/v3.12.0:0fb18b0, C
AMD64)] on win32
Type "help", "copyright", "credits" or

= RESTART: D:\Madadjon\O`ktamov.M.py
a=0
1
```

Rekursiya tushunchasi

Rekursiya deb shunday konstruksiyaga aytiladiki, bunda funksiya o‘zini-o‘zi chaqiradi. To‘g‘ri rekursiya va nisbiy rekursiya bir – biridan farq qiladi. Funksiya agar tanasida o‘ziga murojaat mavjud bo‘lsa, **to‘g‘ri rekursiv** deyiladi. Funksiya boshqa funksiyani chaqirsa va bu funksiya o‘z navbatida birinchi funksiyani chaqirsa, bunday funksiya nisbiy rekursiv funsiya deyiladi.

Rekursiyani qo‘llashga klassik misollar sifatida darajaga oshirish va son faktorialini hisoblash keltirish mumkin. Bu misollar rekursiyani tushuntirish qulay

bo‘lgani uchun klassik hisoblanadi, lekin ular iteratsion usullarga nisbatan afzallikka ega emas.

```
x=int(input('x=='))
y=int(input('y=='))
def degre(x,y):
    result=1
    while y>0:        y-=1
        result*=x
    return result
print(degre(x,y))
```

```
Python 3.12.0 (tags/v3.12.0:0fb18b0, O
AMD64)] on win32
Type "help", "copyright", "credits" or

= RESTART: D:\Madadjon\O`ktamov.M.py
x=3
y=4
81
```

Bu misol quyidagi qoidaga asoslangan $\mathbf{x^y}$ ekvivalent $\mathbf{x*x^{(y-1)}}$. Bu kodda $\mathbf{2^4}$ hisoblash masalasi, $\mathbf{2*2^3}$ hisoblashga keltiriladi. So‘ng $\mathbf{2*2^3}$ ni hisoblash $\mathbf{2*2^2}$ ni hisoblashga keltiriladi, toki ko‘rsatkich nolga teng bo‘lmaguncha. Bu misolning iteratsion varianti quyidagi ko‘rinishga ega:

```
x=int(input('x=='))
y=int(input('y=='));
def degre(x,y):
    result=1
    while y>0:        y-=1
        result*=x
    return result
print(degre(x,y))
```

```
Python 3.12.0 (tags/v3.12.0:0fb18b0, (
AMD64)] on win32
Type "help", "copyright", "credits" or

= RESTART: D:\Madadjon\O`ktamov.M.py
x=3
y=4
81
```

Bu kodni tushunish osonligidan tashqari, u samaraliroqdir, chunki siklni bajarish funksiyani chaqirishga nisbatan tez bajariladi.

Funksiyaga argument berishni soddalashtirish

Python dasturlash tili tarkibida qism dastur funksiyaning argumentlarini soddalashtirilgan holatlarda ishlatish imkoniyati mavjud. Bunda asosan jimlik qoidasi bo‘yicha funksiya tarkibidagi o‘zgaruvchi qiymati olinadi, aks holda foydalanuvchi tomonidan berilgan qiymat qabul qilinadi. Bu jarayonni tushunib olish uchun quyidagi dasturlarga e’tibor bering.

Masala. Daraja(a,x) funksiyasini yarating va bu funksiya daraja(a) sifatida ham natija qaytarsin.

```
def daraja(a,x=2):
    y=a**x
    return y
a=2 b=4
z=daraja(a,b)
print('z=',z)
k=daraja(a)
print('k=',k)
t=daraja(1)
print('t=',t
```

```
Python 3.12.0 (tags/v3.12.0:0fb18b0, (
AMD64)] on win32
Type "help", "copyright", "credits" o:

= RESTART: D:\Madadjon\O`ktamov.M.py
z= 16
k= 4
t= 1
```

Yuqoridagi dasturda *daraja(a,x=2)* funksiya ikkita argument bilan shakllantirilgan lekin dasturning asosiy tanasida *z=daraja(a,b), k=daraja(a)* va *t=daraja(1)* ko‘rinishlarida murojaat qilinmoqda. Bunda funksiya argumenti soddalashtirilgan holatda bitta qiymat ham qabul qilishi mumkin, bunday holatlarda funksiya jimlik qoidasi bo‘yicha o‘zining tarkibidagi qiymatni qabul qiladi. *z,k* va *t* o‘zgaruvchilarni qiymatlari har xil bo‘lmoqda, *z* ning argumenti *b* deb berilmoqda, *k* va *t* ning argument qiymatlari berilmayapti.

Masala. Doiraning yuzini hisoblovchi funksiya hosil qiling. Bu funksiya yordamida 3 ta doira yuzini hisoblang.

```
import math
def doira_yuzi(radius):
    yuzi = math.pi * radius**2
    return yuzi
radius1 = 3
radius2 = 5
radius3 = 7
doira1_yuzi = doira_yuzi(radius1)
doira2_yuzi = doira_yuzi(radius2)
doira3_yuzi = doira_yuzi(radius3)
print(f"Radiusi {radius1} bo‘lgan doira yuzi: {doira1_yuzi}")
print(f"Radiusi {radius2} bo‘lgan doira yuzi: {doira2_yuzi}")
print(f"Radiusi {radius3} bo‘lgan doira yuzi: {doira3_yuzi}")
```

```
Python 3.12.1 (tags/v3.12.1:2305ca5, Dec  7 2023, 22:(
AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for

========== RESTART: D:/Madadjon/O`ktamov.
Radiusi 3 bo'lgan doira yuzi: 28.274333882308138
Radiusi 5 bo'lgan doira yuzi: 78.539816339744831
Radiusi 7 bo'lgan doira yuzi: 153.93804002589985
```

Masala. Ixtiyoriy sonning darajasini hisoblovchi Daraja2 nomli funksiya hosil qiling. Daraja2 funksiyasi orqali a, b, c sonlarining darajasini hisoblovchi dastur tuzing.

```
    def Daraja2(a):
        a=a*a;     return a;
    a=int(input('a='));
    b=int(input('b='));
    c=int(input('c='));
print("a sonning  darajasi=",Daraja2(a))
print("b sonning darajasi=",Daraja2(b));
print("c sonning darajasi=",Daraja2(c));
```

```
Python 3.12.0 (tags/v3.12.0:0fb18b0,
AMD64)] on win32
Type "help", "copyright", "credits" o

= RESTART: D:\Madadjon\O`ktamov.M.py
a=7
b=8
c=9
a sonning  darajasi= 49
b sonning darajasi= 64
c sonning darajasi= 81
```

Masala. 2 ta sonning o‘rta arifmetigi va geometrigini hisoblovchi o‘rta_arifmetig_geometrig nomli funksiya yarating, o‘rta_arifmetigi_geometrigi funksiyasi orqali a, b, c, d sonlaridan (a, b), (a, c), (a, d) juftliklarining o‘rta arifmetigi va geometrigini hisoblovchi dastur tuzing.

```
    import math
    def
orta_arifmetig_geometrig(a,b):
        p=math.sqrt(a*b)
        s=(a+b)/2
        print("O‘rta geometrigi=",p)
        print("O‘rta arifmetigi=",s)
    a=int(input('a='))
    b=int(input('b='))
    c=int(input('c='))
    d=int(input('d='))
    orta_arifmetig_geometrig(a,b)
    orta_arifmetig_geometrig(a,c)
    orta_arifmetig_geometrig(a,d)
```

```
Python 3.12.0 (tags/v3.12.0:0fb18b0,
AMD64)] on win32
Type "help", "copyright", "credits" o

= RESTART: D:\Madadjon\O`ktamov.M.py
a=6
b=8
c=5
d=6
O'rta geometrigi= 6.928203230275509
O'rta arifmetigi= 7.0
O'rta geometrigi= 5.477225575051661
O'rta arifmetigi= 5.5
O'rta geometrigi= 6.0
O'rta arifmetigi= 6.0
```

Masala. Uchburchakning_yuzi funksiyasi orqali uchta teng tomonli uchburchakning yuzini hisoblovchi dastur tuzing.

```
    import math
    def Uchburchakning_yuzi(a):
        s=a*a*math.sqrt(3)/4
        return s
    a=int(input('a='))
    print("Uchburchak
yuzi=",Uchburchakning_yuzi(a))
```

```
Python 3.12.0 (tags/v3.12.0:0fb18b0, (
AMD64)] on win32
Type "help", "copyright", "credits" o

= RESTART: D:\Madadjon\O`ktamov.M.py
a=5
Uchburchak yuzi= 10.825317547305483
```

Masala. Natural sonning raqamlar yig‘indisini hisoblovchi raqamlarning_yig‘indisi nomli funksiya hosil qiling. Bu funksiya orqali a, b, c sonlarining yig‘indisini hisoblovchi dastur tuzing.

```
def raqamlarning_yigindisi(n):
    s=0;
    for i in range(1,n+1):
        s=s+i;
    return s;
a=int(input('a='));
b=int(input('b='));
c=int(input('c='));
print("a son raqamlar yig‘indisi=",raqamlarning_yigindisi(a));
print("b son raqamlar yig‘indisi=",raqamlarning_yigindisi(b));
print("c son raqamlar yig‘indisi=",raqamlarning_yigindisi(c));
```

```
Python 3.12.0 (tags/v3.12.0:0fb18b
AMD64)] on win32
Type "help", "copyright", "credits

= RESTART: D:\Madadjon\O`ktamov.M.
a=15
b=25
c=35
a son raqamlar yig'indisi= 120
b son raqamlar yig'indisi= 325
c son raqamlar yig'indisi= 630
```

Masala. Butun musbat sonning raqamlarini teskari tartibda chiqaruvchi teskari_tartibda nomli funksiya hosil qiling. Bu funksiya orqali a, b, c sonlarining raqamlarini teskari tartibda chiqaruvchi dastur tuzing.

```
import math
def teskari_tartibda(n):
    while n>0:
        i=n%10
        n=math.floor(n/10)
        print(i,end='')
a=int(input('a='))
b=int(input('b='))
c=int(input('c='))
teskari_tartibda(a)
print('\n')
teskari_tartibda(b
print('\n')
teskari_tartibda(c)
```

```
Python 3.12.0 (tags/v3.12.0:0fb18b0, O
AMD64)] on win32
Type "help", "copyright", "credits" or

= RESTART: D:\Madadjon\O`ktamov.M.py
a=1234
b=4567
c=9876
4321

7654

6789
```

Masala. 2 ta sonning qiymatini almashtiruvchi almashtirish nomli funksiya hosil qiling. Almashtirish funksiyasi orqali A. B, C, D sonlaridan (A, B), (D, C) juftliklarining qiymatlarini almashtiruvchi dastur tuzing.

```
def almashtirish(x,k):
    y=x;   x=k;   k=y;
    print(x)
    print(k)
a=int(input('a='))
b=int(input('b='))
c=int(input('c='))
d=int(input('d='))
almashtirish(a,b)
almashtirish(c,d)
```

```
Python 3.12.0 (tags/v3.12.0:0fb18b0, C
AMD64)] on win32
Type "help", "copyright", "credits" or

= RESTART: D:\Madadjon\O`ktamov.M.py
a=10
b=11
c=12
d=13
11
10
13
12
```

Masala. X va Y sonlaridan kichigini X ga va kattasini Y ga yozuvchi Minmax(X,Y) funksiyasini hosil qiling. Minimax funksiyasini 4 marta chaqirish orqali a. b, c, d butun sonlaridan kattasini va kichigini aniqlovchi dastur yarating.

```
def Minimax(x,y):
    if x>y:
        max=x;      min=y;
    else:
        max=y;      min=x;
  print("minimum=",min);
  print("maksimum=",max);
a=int(input('a='));
b=int(input('b='));
c=int(input('c='));
d=int(input('d='));
Minimax(a,b); Minimax(c,d);
```

```
Python 3.12.0 (tags/v3.12.0:0fb18b0, (
AMD64)] on win32
Type "help", "copyright", "credits" or

= RESTART: D:\Madadjon\O`ktamov.M.py
a=10
b=11
c=12
d=13
minimum= 10
maksimum= 11
minimum= 12
maksimum= 13
```

10-MAVZU. PYTHON DASTURLASH TILIDA KO‘P QIYMAT QAYTARUVCHI FUNKSIYALAR VA ULARDAN FOYDALANISH

Reja:

1. Ko‘p qiymat qaytaruvchi funksiyalar;
2. Ko‘p qiymat qaytaruvchi funksiyalarni shakllantirish;

Tayanch so‘zlar. *Ko‘p qiymat, return, funksiya tanasi.*

Python dasturlash tilida funksiyalar dastur tarkibidagi kichik dasturlar hisoblanadi, ular dastur bajarilish natijasida bitta qiymat qaytaradi. Masalan, sonning faktarialini, sonlarning ekubini, sonlarning ekukini va hakoza shunga o‘xshash natijalarni qaytaradi. Lekin dastur tarkibida ikki va undan ortiq natija qaytaradigan kichik muammolar ham mavjud.

Masalan, kvadrat funksiyaning ildizlarini aniqlash, unda dastur ko‘pi bilan ikkita qiymat qaytarish kerak, massivlarni o‘sish yoki kamayish tartibida tartiblash va hakoza shunga o‘xshash masalalar ko‘p uchraydi. Bu turdagi masalalarni yechish uchun oddiy funksiyalardan foydalanish maqsadga muvofiq bo‘lmaydi. Python dasturlash tilida ikki va undan ortiq qiymat qaytaradigan funksiyalarni qisqacha qilib ko‘p qiymat qaytaruvchi funksiyalar deb nomlaymiz. Yuklangan funksiyalar chaqirilganda, qaysi funksiyani chaqirish kirish parametrlarining soniga, ularning tiplariga va navbatiga bog‘liq bo`ladi. Yani ism yuklanishida funksiyaning imzosi rol o‘ynadi. Agar kirish parametrlari va ismlari ayni funksiyalarning farqi faqat ularning qaytish qiymatlarida bo‘lsa, bu yuklanish bo‘lmaydi, kompilyator buni xato deb e'lon qiladi. Funksiya yuklanishi asosan ayni ishni yoki amalni farqli usul bilan farqli ma'lumot tiplari ustida bajarish uchun qo‘llaniladi.

Masalan bir fazoviy jismning hajmini hisoblash kerak bo‘lsa, har bir jismning hajmi farqli formula yordamida, ya’ni farqli usul yordamida topiladi, bir jismda radius tushunchasi bor bo‘lsa, boshqasida asos yoki tomon tushunchasi bor bo‘ladi, bu esa farqli ma'lumot tiplariga kiradi. Demak, biz funksiya yuklanishi mexanizmini qo‘llasak bo‘ladi. Bir xil amalni bajaruvchi funksiyalarni ayni nom bilan atashimiz dasturni o‘qib tushunishni osonlashtiradi. Kompilyator biz bergan funksiya imzosidan (imzoga funksiya ismi va kirish parametrlari kiradi, funksiyaning qaytish qiymati esa imzoga

kirmaydi) yagona ism tuzadi, dastur ijrosi davrida esa funksiya chaqirig‘idagi argumentlarga qarab, kerakli funksiyani chaqiradi. Demak, funksiyani chaqirish uning nomiga bog‘liq ekan. Ko‘p qiymat qaytaruvchi funksiyalar esa, albatta, uning imzosida protsedura nomi kirish va chiqish parametrlari, albatta, keltirilishi kerak, chunki ko‘p qiymat qaytaruvchi funksiyalar tarkibida bir nechta qaytariladigan qiymatlar, albatta, biror bir parametrlarga bog‘langan bo‘ladi.

Ko‘p qiymat qaytaruvchi funksiyalarni shakllantirish

Ko‘p qiymat qaytaruvchi funksiyalar, oddiy funksiyalardan farqi shundaki, u faqat bitta qiymat qaytarmaydi, balki bir nechta qiymat qaytarishga mo‘ljallangandir. Yagona nom bilan saqlangan ko‘p qiymat qaytaruvchi funksiyalar yordamida ikki sonning yig‘indisini, ko‘paytmasini, nisbatini va ayirmasini hisoblovchi funksiya yaratish mumkin.

Ta’rif: Python dasturlash tilining dastur tarkibida ikki va undan ortiq qiymat qaytaradigan qism dasturlar ko‘p qiymat qaytaruvchi funksiyalar deyiladi.

Ko‘p qiymat qaytaruvchi funksiyalarni shakllantirishda, albatta, uning kiritish qiymatlar parametrlari keltirilishi kerak. Ko‘p qiymat qaytaruvchi funksiyalarni shakllantirish usullari to‘liq funksiyalarni e’lon qilish usullari bilan bir xil bo‘ladi, bu yerda ham funksiya murojaat qilinishidan oldin shakllantirilgan bo‘lishi kerak.

Ko‘p qiymat qaytaruvchi funksiyalar python dasturlash tilida shakllantirishining umumiy ko‘rinishi quyidagicha bo‘ladi.

```
def <funksiya nomi>(<argumentlar>):
        funksiya tanasi
        return <o‘zgaruvchilar>
```

Ko‘p qiymat qaytaruvchi funksiyalarni shakllantirishda qavs ichida **argumentlar** sifatida kiritish parametrlari tasvirlanadi, keyin return xizmatchi so‘zidan keyin **o‘zgaruvchilar** qiymat qaytaruvchi parametrlar sifatida tasvirlanadi. Bunda return so‘zidan keyingi o‘zgaruvchilar, mos ravishda funksiya tanasidagi o‘zgaruvchilar qiymatlarini qabul qiladi.

Ko‘p qiymat qaytaruvchi funksiyalarni tasvirlash uchun ikki sonning yig‘indisi va ko‘paytmasini hisoblovchi *kop(x,y)* nomi bilan yaratilgan funksiya dasturiga e’tibor bering.

Misol. 2 sonning yig‘indisi va ko‘paytmasini hisoblovchi kup(x,y) funksiya yarating va funksiyadan foydalanish dasturini yarating.

```
def kup(x,y):
  t=x+y
  z=x*y
  return t,z
a=input('a=')
b=input('b=')
a=int(a)
b=int(b)
kup(a,b)
n,m=kup(a,b)
print('a+b=',n)
print('a*b=',m)
```

```
Python 3.12.0 (tags/v3.12.0:0fb18b0,
AMD64)] on win32
Type "help", "copyright", "credits"

= RESTART: D:\Madadjon\O`ktamov.M.py
a=100
b=150
a+b= 250
a*b= 15000
```

kup(x,y) funksiyasi *a* va *b* sonlarining yig‘indisini *n* o‘zgaruvchiga, ko‘paytmasini esa *m* o‘zgaruvchiga saqlaydi. Funksiya qiymat qaytaruvchi parametrlar, albatta, return so‘zidan keyin yoziladi, natijada qism dastur tarkibidagi bir nechta o‘zgaruvchi qiymatlarini dasturning asosiy tanasiga olib chiqish imkonini yaratiladi.

Misol. Chiziqli funksiyaning yechimi cheksiz, yagona va mavjud emaslik holatlarini aniqlaydigan dastur yarating.

```
def chiziqli_fun(a,b):
    if a==b:
        print('cheksiz yechimga ega')
    elif (a==0 and b!=0):
        print('yechimga ega emas')
    elif (a!=0 and b!=0):
        print('yagona yechimga ega')
a=input('a=')
b=input('b=')
a=int(a)
b=int(b)
chiziqli_fun(a,b)
```

```
Python 3.12.0 (tags/v3.12.0:0fb18b0,
AMD64)] on win32
Type "help", "copyright", "credits"

= RESTART: D:\Madadjon\O`ktamov.M.py
a=20
b=25
yagona yechimga ega
```

Yuqoridagi dasturga e’tibor qaratsak, return so‘zi ishlatilmayapti, funksiya natijasi biror o‘zgaruvchiga olib chiqish shart bo‘lmasa return so‘zi ham kerak emas.

11-MAVZU. PYTHON DASTURLASH TILIDA RO‘YXATLAR VA KORTEJLAR

Reja:

1. Ma’lumot to‘plamlari va turlari
2. Ro‘yxatlar va ularning umumiy ko‘rinishi;
3. Kortejlar va ularning umumiy ko‘rinishi;

Tayanch so‘zlar. Ro‘yxat, kortej, dinamik xotira, sorted, append.

Pythonda ma’lumot to‘plamlarining 4 xil turlari mavjud. Biz bulardan odatda bir nechta yoki undan ko‘p qiymatlarni saqlashda ishlatamiz. Bizga kerak bo‘lganda shu to‘plamlarga murojaat qilib tegishli qiymatlarni olamiz.

Har bir ma’lumot to‘plamining o‘z xususiyatlari bor va shunga ko‘ra ularni kerakli joyda tanlab ishlatishimiz mumkin.

List (ro‘yxat) – tartiblangan va o‘zgaruvchan ro‘yxat. Elementlarini dublikatlash mumkin.

Tuple (kortej) – tartiblangan va o‘zgarmas ro‘yxat. Elementlarini dublikatlash mumkin.

Set (to‘plam) – Tartiblanmagan va indekslanmagan to‘plam. Elementlari dublikatlanmaydi.

Dictionary (lug‘at) – tartiblanmagan, o‘zgaruvchan va indekslangan to‘plam. Elementlari dublikatlanmaydi.

Python dasturlash tilida kompyuter xotirasiga bir o‘zgaruvchi yordamida bir nechta qiymatlarda foydalanishga to‘g‘ri keladi. Bir o‘zgaruvchi bilan bir nechta qiymat ustida amallar bajarish boshqa dasturlash tillaridan farqli ravishda, berilgan ma’lumotlar bir turga mansub bo‘lishi shart emas.

Ro‘yxatlar

Biz bu qismda dasturdagi ma’lumot strukturalari bilan tanishishni boshlaymiz. Dasturda 2 asosiy tur ma’lumot strukturalari mavjuddir. Birinchisi statik, ikkinchisi dinamikdir. Statik deganimizda xotirada egallagan joyi o‘zgarmas, dastur boshida beriladigan strukturalarni nazarda tutamiz. Dinamik ma'lumot tiplari dastur davomida o‘z hajmini, egallagan xotirasini o‘zgartirishi mumkin. Python dasturlash tilining

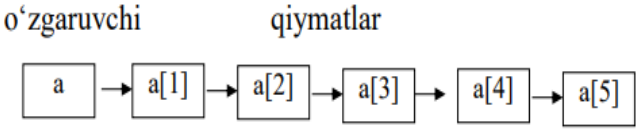
imkoniyatlar kengligining yana bir xususiyati, o‘zgaruvchilarning dinamikligidir. Python dasturlash tilida har xil turdagi bir nechta ma’lumotlarni boshqarish va qayta ishlash imkoniyati mavjud. Bu imkoniyatni python dasturlash tilida ro‘yxatlar(list) amalga oshiradi. Alohida bir o‘zgaruvchini ko‘rsatish uchun ro‘yxat nomi va kerakli o‘zgaruvchi indeksi yoziladi.

Ta’rif: Har xil turga mansub bo‘lgan yagona nom bilan saqlanuvchi tartiblangan ma’lumotlar majmuasi **ro‘yxat(list)** deyiladi.

Ro‘yxatlar yagona o‘zgaruvchi bilan kompyuter xotirasiga saqlanadi, uning elementlari ma’lum bir indekslar bilan tartiblab joylashtiriladi. Python dasturlash tilida ro‘yxatlarni boshqa dasturlash tillaridagi bir o‘lchovli massivlarga o‘xshatish mumkin, lekin pythonda ma’lumotlar bir turga mansub bo‘lmasligi ham mumkin.

Python dasturlash tilida ro‘yxatlardan foydalanishda bozordagi mahsulotlarning narxini olish mumkin. Mahsulot narxlarini ro‘yxat sifatida qaralganda narx1, narx2, narx3, …, narxn ko‘rinishda bir nechta mahsulot narxlarini kompyuter xotirasiga saqlab undan foydalanish mumkin.

Odatda ro‘yxatlar zarurat, katta hajmdagi tartiblangan, lekin chekli elementlarga oid masalalarni hal etishda yuzaga keladi. Dastur ishlatilishi davomida ro‘yxatlar aniq nomga ega bo‘lishi va uning elementlari ma’lum bir turda bo‘lishi kerak. Python dasturlash tilida ro‘yxatlar kompyuter xotirasiga quyidagi shaklda saqlanadi.



Yuqoridagi holat bo‘yicha ro‘yxatlar kompyuter xotirasiga saqlanadi, bunda ro‘yxatning ixtiyoriy elementiga murojaat qilish uchun uning indeks nomeri bo‘yicha murojaat qilinadi.

Ro‘yxatlarni boshlang‘ich qiymatlari bergan holatda faollashtirish quyidagicha amalga oshiriladi.

<ro‘yxat o‘zgaruvchisi>=[qiymat1, qiymat2, . . .]

Ro‘yxatni Python dasturlash tilida faol qilish uchun, albatta, elementlar soni berilish shart emas, ro‘yxatning elementlar soni uning tarkibidagi qiymatlariga qarab aniqlanadi.

List (ro‘yxat)

List(ro‘yxat) - Pythonda erkin turdagi obyektlarning o‘zgaruvchan qatorlashgan kolleksiyasi hisoblanadi (*massivga o‘xshash, lekin tiplar har xil bo‘lishi mumkin*). Ro‘yxatlardan foydalanish uchun ularni tuzish lozim. List – ta’kidlaganimizdek tartiblangan va o‘zgaruvchan ro‘yxatdir. Ro‘yxatni har xil yondashuvlarni qo‘llagan holatda yaratish mumkin. Biz ro‘yxatlarni kvadrat qavslar yordamida hosil qilamiz:

```
mashina = ["Audi", "Mustang", "Ferrari"]
print(mashina)
```

```
Python 3.12.0 (tags/v3.12.0:0fb18b0, 
AMD64)] on win32
Type "help", "copyright", "credits"

= RESTART: D:\Madadjon\O`ktamov.M.py
['Audi', 'Mustang', 'Ferrari']
```

Pythonda List() konstruktori

List ro‘yxatini **list()** konstruktori yordamida hosil qilishimiz mumkin. Bunday holatda esa kvadrat qavslar ishlatilmaydi:

```
mashina=list(("matiz", "damas", "nexia", "spark", "gentra"))
print(mashina)
```

```
[' matiz', ' damas', ' nexia', spark, gentra]
```

Elementlarga murojaat

Ro‘yxat elementlariga murojaat qilish uchun, murojaat qilinayotgan elementning indeksi ko‘rsatiladi. Sanoq har doimgidek 0 dan boshlanadi. Quyidagi dasturimiz ishga tushsa, ekranga ikkinchi element chiqadi:

```
mashina = ["Matiz", "Damas", "Nexia"]
print(mashina[1])
```

```
Damas
```

Manfiy indeks

Manfiy indeks sanoq oxiridan boshlanishini bildiradi. Masalan, -1 eng oxirgi, -2 oxiridan ikkinchi element va hokazo. Quyidagi dasturimiz ishga tushsa, oxirgi element ekranga chiqadi:

mashina = ["Matiz", "Damas", "Nexia"] print(**mashina**[-1])	**Nexia**

Indeks oralig‘i

Ro‘yxatning ma’lum bir qismidagi bir nechta elementni tanlab olish uchun o‘sha indekslar oralig‘ini kiritamiz. Bunda uning boshlanish va oxirgi nuqtalari kiritiladi. Element tanlashda oxirgi nuqta hisobga kirmaydi. Ya’ni boshlang‘ich nuqtadan boshlanib oxirgi nuqtadan bitta oldingi elementgacha olinadi. Hozir biz ro‘yxatdan ikkinchi, uchinchi va to‘rtinchi elementlarni tanlab olamiz:

mashina = ["matiz", "damas", "tiko", "nexia", "gentra"] **print(mashina[1:4])**	**[damas, tiko, nexia]**

Agar indekslar oralig‘ida boshlang‘ich nuqtani olib tashlasak, tanlash ro‘yxat boshidan boshlanadi. Agar oxirgi nuqtani olib tashlasak, tanlash ro‘yxat oxirigacha davom etadi. Quyidagi kodimizda avval ro‘yxat boshidan uchinchi elementgacha, so‘ngra, ikkinchi elementdan ro‘yxat oxirigacha bo‘lgan elementlarni ekranga chiqaramiz:

mashina = ["damas", "matiz", "nexia", "tiko", "gentra"] **print(mashina[:4])** **print(mashina[1:])**	**['damas', 'matiz', 'nexia', 'tiko']** **['matiz', 'nexia', 'tiko', 'gentra']**

Element qiymatini o‘zgartirish

List ro‘yxatidagi xohlagan elementning qiymatini o‘zgartirish mumkin. Buning uchun uning indeksi orqali murojaat qilib, yangi qiymatni kiritamiz. Misol tariqasida ro‘yxatdagi birinchi elementni o‘zgartiramiz:

mashina = ["damas", "matiz", "nexia", "tiko", "gentra"] **mashina[0]="cobalt"** **print(mashina)**	**['cobalt', 'matiz', 'nexia', 'tiko', 'gentra']**

Ro‘yxat bo‘ylab sikl

For siklidan foydalanib ro‘yxatdagi elementlarni tanlab olish ham mumkin. **For** sikli haqida batafsil keyingi mavzularda to‘xtalamiz. Quyidagi misolda esa bu sikl bilan elementlarni qanday ekranga chiqarishni ko‘rib oling:

mashinalar = ["damas", "matiz", "nexia"] for x in mashinalar: print(x)	damas matiz nexia

Elementning mavjudligini tekshirish

Biror elementning ro‘yxatda mavjudligini tekshirish uchun in operatoridan foydalaniladi. Hozir ro‘yxatda nok borligini tekshiramiz:

mashina = ["damas", "matiz", "nexia", "tiko", "gentra"]
if "tiko" in mashina:
print("Ha, tiko bor")
else:
print("tiko yo‘q")
	Ha, tiko bor

Ro‘yxatning funksiya va metodlari

Ro‘yxatni yaratgandan so‘ng ro‘yxatning ustida turli amallarni bajarish kerak bo‘ladi, buning uchun esa Pythonni o‘ziga kiritilgan bir qancha funksiya va metodlar bor.

Metod	Vazifasi
List.append(x)	Ro`yxat oxiridan element qo`shish
List.extend(L)	Oxiriga hamma elementlarni qo`shib list ro`yxatini kengaytiradi.
List.insert(i,x)	i-elementga x qiymatini kiritadi
List.remove(x)	Ro`yxatdan x qiymatga ega elementni o`chiradi
List.pop([i])	Ro`yxatning i-elementini o`chiradi va qaytaradi. Agarda indeks ko`rsatilmagan bo`lsa oxirgi element o`chiriladi
List.index(x,[start],[end])	X qiymatga teng start dan end gacha birinchi elementni qaytaradi
List.count(x)	X qiymatga teng elementlar sonini qaytaradi
List.sort([key=funksiya])	Funksiya asosida ro`yxatni saralaydi
List.reverse()	Ro`yxatni ochadi
List.copy()	Ro`txatning nusxalaydi
List.clear()	Ro`yxatni tozalaydi

Keling **list** ya’ni ro‘yxatda metodlarni qo‘llanilishini misollar yordamida ko‘rib chiqamiz.

Ro‘yxat uzunligi

Ro‘yxatda nechta element borligini aniqlash uchun **len()** funksiyasi ishlatiladi.

```
mashinalar = ["damas", "matiz", "nexia", "tiko", "gentra"]
print(len(mashinalar))
```

```
5
```

Element qo‘shish

append() funksiyasi bilan ro‘yxat oxiridan yangi element qo‘shish mumkin:

```
mashinalar=["damas", "matiz", "nexia", "tiko", "gentra"]
mashinalar.append("cobalt")
print(mashinalar)
```

```
['damas', 'matiz', 'nexia', 'tiko', 'gentra', 'cobalt']
```

Agar biz elementni ro‘yxat oxiriga emas, uning ma’lum bir o‘rniga qo‘shmoqchi bo‘lsak **insert()** funksiyasini ishlatamiz. Buning uchun qo‘shmoqchi bo‘lgan o‘rnimizning indeksini ham kiritamiz. Masalan ro‘yxatning boshiga yangi elementni qo‘shamiz:

```
mashinalar=["damas", "matiz", "nexia", "tiko", "gentra"]
mashinalar.insert(0, "cobalt")
print(mashinalar)
```

```
['cobalt', 'damas', 'matiz', 'nexia', 'tiko', 'gentra']
```

Elementni o‘chirish

Ro‘yxatdan elementni o‘chirishning bir nechta usullari mavjud.

remove() funksiyasi belgilangan elementni ro‘yxatdan o‘chiradi. Bunda uning indeksi emas balki o‘zi ko‘rsatiladi:

```
mashinalar=["damas", "matiz", "nexia", "tiko", "gentra"]
mashinalar.remove("matiz")
print(mashinalar)
```

```
['damas', 'nexia', 'tiko', 'gentra']
```

pop() funksiyasi ko‘rsatilgan indeks bo‘yicha elementni ro‘yxatdan o‘chiradi. Agar indeks ko‘rsatilmasa avtomatik tarzda ro‘yxat oxiridagi elementni o‘chiradi:

```
mashinalar=["damas", "matiz", "nexia", "tiko", "gentra"]
mashinalar.pop()
print(mashinalar)
```

```
['damas', 'matiz', 'nexia', 'tiko']
```

del kalit so‘zi ko‘rsatilgan indeks bo‘yicha element ro‘yxatdan o‘chiriladi. Agar shunchaki ro‘yxat nomi ko‘rsatilsa, butun ro‘yxat o‘chiriladi. Hozir misolimizda, avvalo, bir elementni o‘chiramiz, so‘ngra ro‘yxatning o‘zini o‘chiramiz:

```
mashinalar=["damas", "matiz", "nexia", "tiko", "gentra"]
del mashinalar[1]
```

```
print(mashinalar)
del mashinalar
print(mashinalar)
```

clear() funksiyasi ro‘yxat elementlarini tozalaydi, ya’ni ro‘yxat bo‘m-bo‘sh bo‘lib qoladi:

```
mashinalar=["damas", "matiz", "nexia", "tiko", "gentra"]
mashinalar.clear()
print(mashinalar)
```

```
[ ]
```

Ro‘yxatdan nusxa olish

Bir ro‘yxatdan ikkinchi ro‘yxatni **list2 = list1** ko‘rinishida hosil qilib bo‘lmaydi. Chunki bunda **list2 list1** ga yo‘llanma(silka) bo‘lib qoladi. Shuning uchun **list1** da bo‘lgan o‘zgarishlar **list2** ga ham ta’sir qiladi. Shu sababli bir ro‘yxat ikkinchisiga nusxalanadi. Shunda 2 ta bir xil alohida ro‘yxatlar hosil bo‘ladi.

Ro‘yxatdan nusxa olish uchun **copy()** funksiyasi ishlatiladi.

```
mashinalar1=["damas", "matiz", "nexia", "tiko", "gentra"]
mashinalar2 = mashinalar1.copy()
print(mashinalar2)
```

```
["damas", "matiz", "nexia", "tiko", "gentra"]
```

Ro‘yxatdan nusxa olishning boshqa usuli **list()** funksiyasi:

```
mashinalar1=["damas", "matiz", "nexia", "tiko", "gentra"]
mashinalar2 = list(mashinalar1)
print(mashinalar2)
```

```
["damas", "matiz", "nexia", "tiko", "gentra"]
```

Ro‘yxatlarni qo‘shish

Pythonda ikki yoki undan ko‘p ro‘yxatlarni o‘zaro qo‘shishning turli usullari mavjud. Eng oson yo‘li esa **“+”** operatoridan foydalanishdir.

Shuni eslatish kerakki, ro‘yxat nafaqat satr va harflar, balki sonli o‘zgaruvchilardan ham iborat bo‘ladi:

```
a = [1, 2, 3, 4, 5] b = [5, 6, 7]
c= a + b print(c)
```

```
[1, 2, 3, 4, 5, 5, 6, 7]
```

Bir ro‘yxatga boshqasini qo‘shishning yana bir yo‘li – ikkinchi ro‘yxatning elementlarini bittalab qo‘shib chiqish:

```
meva1 = ["olma", "uzum", "anor"]
meva2 = ["nok", "banan", "apelsin"]
```

```
['olma', 'uzum', 'anor', ['nok', 'banan', 'apelsin'], ['nok',
```

for x in meva2: meva1.append(meva2) print(meva1)	'banan', 'apelsin'], ['nok', 'banan', 'apelsin']]

extend() funksiyasi ham bir ro‘yxatdagi elementlarni ikkinchisiga qo‘shib chiqadi. Qo‘shilayotgan elementlar avtomatik tarzda ro‘yxat oxiridan boshlab qo‘shiladi.

a = [1, 2, 3, 4, 5] b = [5, 6, 7] a.extend(b) print(a)	[1, 2, 3, 4, 5, 5, 6, 7]

count() va index()

count() funksiyasi belgilangan qiymatga teng elementlar sonini aniqlaydi.

index() funksiyasi belgilangan elementning indeksini aniqlaydi. Agar bunday elementlar bir nechta bo‘lsa, faqat birinchisining indeksini aniqlaydi.

Hozir ro‘yxatda nechta 5 soni borligi va uning indeksini aniqlaymiz:

a = [1, 2, 3, 4, 5] x = a.count(5) print(x) x = a.index(5) print(x)	1 4

sort() va reverse()

sort() funksiyasi ro‘yxatni tartiblaydi. Agar ro‘yxat sonlardan tashkil topgan bo‘lsa, o‘sish tartibida, satr yoki harflardan tashkil topgan bo‘lsa, alifbo bo‘yicha tartiblaydi.

reverse() funksiyasi ro‘yxatning joriy holatdagi tartibini teskarisiga o‘zgartiradi.

Hozir ikki xil ro‘yxatni avval tartiblaymiz, so‘ngra ularni teskarisiga o‘zgartiramiz:

mashinalar = ["damas", "matiz", "tiko", "nexia", "gentra"] a = [1, 2, 3, 4, 5] mashinalar.sort() a.sort() print(mashinalar) print(a) mashinalar.reverse() a.reverse() print(mashinalar) print(a)	['tiko', 'matiz', 'nexia', 'damas', 'gentra'] [1, 2, 3, 4, 5] ['gentra', 'damas', 'nexia', 'matiz', 'tiko'] [5, 4, 3, 2, 1]

Tuple (Kortej)

Kortejlar bir nechta obyektlarni birgalikda saqlashga xizmat qiladi. Ularni ro‘yxatlarga o‘xshatish mumkin. Lekin ular ro‘yxatlar kabi boy funksionallikka ega emas. Ularning asosiy jihati qatorlarga o‘xshab o‘zgarmasliklaridir. **Kortej**-elementlar orasini vergul bilan ajratish orqali hosil qilinadi. Kortejga ma’no jihatdan o‘zgarmas ro‘yxat deb ta’rif berdik. Shu o‘rinda savol tug‘iladi. Ro‘yxat bo‘lsa kortej nimaga kerak:

Turli holatlardan himoyalanish. Bu degani kortej o‘zgartirishlardan himoyalangan bo‘ladi, rejali (bu yomon) va tasodifiy (bu yaxshi) o‘zgarishlardan xalos bo‘ladi. [12]

Kortej afzalliklari haqida bilib oldik. Endi kortej bilan qanday ishlashni ko‘ramiz. Bu ro‘yxatlar bilan ishlashga o‘xshaydi.

Tuple ro‘yxati tartiblangan va o‘zgarmas ro‘yxatdir. Uning elementlarini o‘zgartirib bo‘lmaydi. Bu ro‘yxatni oddiy qavslar bilan yoki **tuple()** konstruktori bilan hosil qilinadi:

```
a= ("kitob", "daftar", "ruchka")
b=  tuple(("qog‘oz", "qalam", "qaychi"))
    print(a)
    print(b)
```

```
('kitob', 'daftar', 'ruchka')
("qog‘oz", 'qalam', 'qaychi')
```

Bir elementli to‘plam

Bitta elementli tuple kortejini yaratish uchun element qiymatidan so‘ng vergul (,) qo‘yish lozim. Aks holda bunday kortej hosil bo‘lmaydi:

```
    a = ("kitob",)
    print(type(a))
    b = ("kitob")
    print(type(b))
```

```
    <class 'tuple'>
    <class 'str'>
```

Elementlarga murojaat

Tuple elementiga murojaat qilish uning indeksini ko‘rsatish bilan amalga oshiriladi:

```
    a = ("kitob", "daftar", "ruchka")
    print(a[0])
```

```
    kitob
```

[12] https://www.bilimlar.uz/wp-content/uploads/2021/02/k100001.pdf

Manfiy indeks

Manfiy indeks sanoqning oxiridan boshlanishini anglatadi. Masalan, -1 eng oxirgi, -2 oxiridan ikkinchi element va hokazo.

a = ("kitob", "daftar", "ruchka") print(a[-1])	ruchka

Indeks oralig‘i

Ro‘yxatning ma’lum qismidagi bir nechta elementlarni tanlab olish uchun shu indekslar oralig‘ini kiritishimiz kerak. Bunda uning boshlanish va oxirgi nuqtalari kiritiladi. Element tanlashda oxirgi nuqta hisobga kirmaydi. Ya’ni boshlang‘ich nuqtadan boshlanib oxirgi nuqtadan bitta oldingi elementgacha olinadi.

Hozir ekranga ikkinchi, uchinchi va to‘rtinchi elementlarni tanlab ekranga chiqaramiz:

a=("kitob","daftar",ruchka", "qog‘oz", "qalam") print(a[1:4])	('daftar','ruchka', qog‘oz")

Element qiymatlarini o‘zgartirish

Tuple to‘plamidagi elementni to‘g‘ridan-to‘g‘ri o‘zgartirib bo‘lmaydi. Yuqorida aytganimizdek u o‘zgarmas. Biroq bu muammoning ham yechimi bor. **Tuple** ro‘yxatini avval **list** ro‘yxatiga aylantirib, so‘ngra istalgan elementni o‘zgartiriladi va yana **tuple** ro‘yxatiga aylantiriladi: [13]

a = ("qogoz", "qalam", "ruchka") b = list(a) b[2] = "daftar" a = tuple(b) print(a)	(qogoz, qalam, daftar)

Ro‘yxat bo‘ylab sikl

Tuple to‘plamida for siklidan foydalanib elementlarni tanlab olish mumkin. Quyidagi misolda shu usul bilan elementlarni ekranga chiqaramiz:

a = ("kitob", "daftar", "ruchka") for x in a: print(x)	('daftar','ruchka', qog‘oz")

[13] https://www.bilimlar.uz/wp-content/uploads/2021/02/k100001.pdf

Elementning mavjudligini tekshirish

Biror bir elementning to‘plamda mavjudligini **in** kalit so‘zi orqali tekshiramiz. Masalan, ro‘yxatimizda qalam borligini tekshiramiz:

a = ("qalam", "daftar", "ruchka")
if “qalam” in a:
print(“qalam bor”)
else:
print(“qalam yo‘q”)
	qalam bor

Kortejning funksiya va metodlari

count(x)	kortejdagi x elementi sonini qaytaradi
index(x)	kortejdagi x elementining indeksini qaytaradi
any()	agar kortej elementi mavjud bo‘lsa True qiymat qaytaradi, aks holda (kortej bo‘sh bo‘lsa) False qiymat qaytaradi
max()	kortejning maksimal elementini qaytaradi
min()	kortejning minimal elementini qaytaradi
len()	kortejning uzunligini qaytaradi
sorted()	kortej elementlaridan iborat yangi tartiblangan ro‘yxatni qaytaradi
sum()	kortej elementlari yig‘indisini qaytaradi

Tuplening uzunligi

Tuple to‘plamining uzunligi, yani nechta elementdan tashkil topganligini **len()** funksiyasi bilan aniqlash mumkin:

a = ("damas", "matiz", "tiko", "nexia") print(len(a))	4

Element qo‘shish

Tuple korteji o‘zgarmas bo‘lgani uchun unga element qo‘shib bo‘lmaydi. U boshida nechta element hosil qilgan bo‘lsa, shuncha element bilan qoladi. Ammo istisno tariqasida, yuqorida elementning qiymatini o‘zgartirganimiz kabi shu usulda yangi element qo‘shsa bo‘ladi. [14]

Tuplelarni qo‘shish

Ikki yoki undan ortiq tuplelarni qo‘shish uchun **“+”** operatori foydalanamiz:

a = ("kitob", "daftar", "ruchka") b = ("qalam", "qog‘oz")	('kitob', 'daftar', 'ruchka', 'qalam', "qog‘oz")

[14] https://www.bilimlar.uz/wp-content/uploads/2021/02/k100001.pdf

```
c = a + b
print(c)
```

count() va index()

count() funksiya belgilangan qiymatga teng elementlar sonini aniqlaydi.

index() funksiya belgilangan elementning indeksini aniqlaydi. Agar bunday elementlar bir qancha bo‘lsa, faqatgina birinchisining indeksini aniqlaydi.

Hozir kortejda nechta 3 soni borligi va uning indeksini aniqlaymiz:

```
toq_son = (1, 3, 5, 3, 3, 7)
x = toq_son.count(3)
print(x)
y = toq_son.index(3)
print(y)
```

```
3
1
```

Misol. n ta butun sonli elementdan tashkil topgan ro‘yxat hosil qilib, ro‘yxatning juft elementlarini ikkiga ko‘paytirib toq elementlarini 3ga ko‘paytirib ekranga chiqaring.

```
m=int(input('m='))
a=[int(input())
for i in range(m)]
for i in range(m):
 if a[i]%2==0:
a[i]=a[i]*2
else:
a[i]=a[i]*3
print(a)
```

```
m=5
2
9
7
1
8
[4, 27, 21, 3, 16]
```

Misol. Berilgan ro‘yxatning eng oxirgi elementini chiqaring:

```
thislist=["apple", "banana", "cherry"]
print(thislist[-1])
```

```
cherry
```

Misol. 3-4-5-elementlarni chiqaring:

```
thislist=["apple", "banana", "cherry", "orange",
"kiwi", "melon", "mango"]
print(thislist[2:5])
```

```
['cherry', 'orange', 'kiwi']
```

Misol. 2-elementni o‘zgartiring:

```
thislist=["apple", "banana", "cherry"]
thislist[1] = "blackcurrant"
print(thislist)
```

```
['apple', 'blackcurrant', 'cherry']
```

Misol. "Banan" va "gilos" qiymatlarini "qora smorodina" va "tarvuz" qiymatlari bilan o‘zgartiring:

thislist=["apple", "banana", "cherry", "orange", "kiwi", "mango"] thislist[1:3]=["blackcurrant", "watermelon"] print(thislist)	['apple', 'blackcurrant', 'watermelon', 'orange', 'kiwi', 'mango‘]

Misol. Ikkinchi qiymatni 2 ta yangi qiymat bilan almashtiring:

thislist=["apple", "banana", "cherry"] thislist[1:2]=["blackcurrant", "watermelon"] print(thislist)	['apple', 'blackcurrant', 'watermelon', 'cherry']

Misol. Ikkinchi va uchinchi qiymatlarni 1 ta qiymat bilan almashtiring:

thislist = ["olma", "banan", "gilos"] thislist[1:3] = ["tarvuz"] print(thislist)	['olma', 'tarvuz']

Misol. Uchinchi element sifatida "tarvuz" ni qo‘ying:

thislist = ["olma", "banan", "gilos"] thislist.insert(2, "tarvuz") print(thislist)	['olma', 'banan', 'tarvuz', 'gilos']

Misol. append() Elementni qo‘shish usulidan foydalanish:

thislist = ["olma", "banan", "gilos"] thislist.append("apelsin") print(thislist)	['olma', 'banan', 'gilos', 'apelsin']

Misol. tropicalga elementlarni qo‘shing thislist:

thislist = ["apple", "banana", "cherry"] tropical = ["mango", "pineapple", "papaya"] thislist.extend(tropical) print(thislist)	['apple', 'banana', 'cherry', 'mango‘, 'pineapple', 'papaya']

Misol. Ro‘yxat elementlarini ro‘yxatga qo‘shing:

thislist = ["apple", "banana", "cherry"] thistuple = ("kiwi", "orange") thislist.extend(thistuple) print(thislist)	['apple', 'banana', 'cherry', 'kiwi', 'orange']

Misol. "Banan" ni olib tashlang:

thislist = ["apple", "banana", "cherry"] thislist.remove("banana") print(thislist)	['apple', 'cherry']

Misol. Ikkinchi elementni olib tashlang:

```
thislist = ["apple", "banana", "cherry"]
thislist.pop(1)
print(thislist)
```

```
['apple', 'cherry']
```

Misol. Oxirgi elementni olib tashlang:

```
thislist = ["apple", "banana", "cherry"]
thislist.pop()
print(thislist)
```

```
['apple', 'banana']
```

Misol. Ro‘yxatdagi barcha elementlarni birma-bir chop eting:

```
thislist = ["olma", "banan", "gilos"]
for x in thislist:
  print(x)
```

```
olma
banan
gilos
```

Misol. Barcha elementlarni indeks raqamiga qarab chop eting:

```
thislist = ["olma", "banan", "gilos"]
for i in range(len(thislist)):
  print(thislist[i])
```

```
olma
 banan
 gilos
```

Misol. Barcha indeks raqamlarini o‘tish uchun while sikldan foydalanib, barcha elementlarni chop eting,

```
thislist = ["apple", "banana", "cherry"]
i = 0
while i < len(thislist):
  print(thislist[i])
  i = i + 1
```

```
apple
 banana
 cherry
```

Misol. Ro‘yxatdagi barcha elementlarni chop etadigan qisqa for sikli:

```
thislist = ["apple", "banana", "cherry"]
[print(x) for x in thislist]
```

```
apple
 banana
 cherry
```

Misol. Ro‘yxatni alifbo tartibida tartiblang:

```
thislist=["orange", "mango", "kiwi",
"pineapple", "banana"]
thislist.sort()
print(thislist)
```

```
['banana', 'kiwi', 'mango‘,
'orange', 'pineapple']
```

Misol. Ro‘yxatni raqamlar bo‘yicha tartiblang:

```
thislist = [100, 50, 65, 82, 23]
thislist.sort()
print(thislist)
```

```
[23, 50, 65, 82, 100]
```

Misol. Ro‘yxatni kamayishiga qarab tartiblang:

```
thislist=["orange", "mango", "kiwi",
"pineapple", "banana"]
thislist.sort(reverse = True)
print(thislist)
```

```
['pineapple', 'orange',
'mango‘, 'kiwi',
'banana']
```

Misol. Ikki ro‘yxatni qo‘shing:

```
list1 = ["a", "b", "c"]
list2 = [1, 2, 3]
list3 = list1 + list2
print(list3)
```

```
['a', 'b', 'c', 1, 2, 3]
```

Misol. 2 ro‘yxatni 1 ro‘yxatiga qo‘shing:

```
list1 = ["a", "b" , "c"]
list2 = [1, 2, 3]
for x in list2:
  list1.append(x)
print(list1)
```

```
['a', 'b', 'c', 1, 2, 3]
```

12-MAVZU. PYTHON DASTURLASH TILIDA LUG‘AT VA TO‘PLAMLARDAN FOYDALANISH

Reja:

1. Lug‘atlar va ularning umumiy ko‘rinishi;
2. Lug‘atlarga oid dasturlar;
3. To‘plamlar va ularning umumiy ko‘rinishi;
4. To‘plamga oid dasturlar;

Tayanch so‘zlar: lug‘at, to‘plam, del, pop, set, add.

Dictionary (lug‘at)

Pythondagi lug‘atlar kalit bo‘yicha kirishga ruxsat etuvchi erkin obyektlarning tartiblangan yig‘indisidir. Ularni yana assotsiativli massivlar yoki hesh jadvallar deb nomlaydilar. Soddaroq qilib aytganda, lug‘at xuddi manzillar kitobiga o‘xshaydi, ya’ni biror bir shaxsning nomini bilgan holda uning manzili yoki u bilan bo‘g‘lanish ma’lumotlarini olish mumkin.

Dictionary – tartiblanmagan, o‘zgaruvchan va indeksli to‘plam. Dictionaryda kalit-qiymat (***keyvalue***) tushunchasi mavjud, ya’ni maxsus kalit va unga mos keluvchi qiymatlar juftligidan tashkil topgan. Chap tarafda kalitlar, o‘ng tomonda esa ularga mos keluvchi qiymatlar joylashgan bo‘ladi. Bu quyidagicha amalga oshiriladi:

```
avto = {
   "brend": "Chevrolet",
   "model": "Malibu",
   "yil": 2016
}
print(avto)
```

```
{'brend': 'Chevrolet', 'model': 'Malibu', 'yil': 2016}
```

Dict() konstruktori

dict() konstruktori bilan yangi to‘plam hosil qilish mumkin. U quyidagicha amalga oshiriladi:

```
avto=dict(brend="chevrolet", model="Malibu", yil=2016)
print(avto)
```

```
{'brend': 'chevrolet', 'model': 'Malibu', 'yil': 2016}
```

Elementlarga murojaat

Dictionary elementlariga murojaat qilish uchun ularning kalitlarini kvadrat qavs ichida ko‘rsatish yoki **get()** funksiyasidan foydalanishimiz mumkin bo‘ladi. Quyidagi misolda ikkala usuldan ham foydalanamiz:

```
avto = {
  "brend": "Chevrolet",
  "modeli": "Gentra",
  "yili": 2023
}
x = avto["modeli"]
y = avto.get("yili")
print(x)
print(y)
```

```
Gentra
2023
```

Qiymatlarni o‘zgartirish

Xohlagan qiymatni o‘zgartirishimiz uchun unga kalit orqali murojaat qilamiz, so‘ngra qiymatini o‘zgartiramiz. Masalan quyidagi avtomobil haqidagi ma’lumotda yilni o‘zgartiramiz:

```
mashina = {
  "brend": "Chevrolet",
  "modeli": "Gentra",
  "yili": 2018
}
mashina ["yili"] = 2023
print(mashina)
```

```
{'brend': 'Chevrolet', 'model': 'Gentra', 'yil': 2023}
```

Sikldan foydalanish

Dictionary to‘plamida ham **for** siklidan foydalanib uning elementlariga murojaat qilish mumkin. Bunda qiymatlarga emas, balki kalitlarga murojaat bo‘ladi. Quyidagi misolda to‘plamdagi kalitlarni ekranga chiqaramiz:

```
mashina = {
  "brend": "Chevrolet",
  "modeli": "Gentra",
  "yil": 2023
}
for x in mashina:
    print(x)
```

```
brend
modeli
yil
```

Agar qiymatlarning o‘ziga murojaat qilmoqchi bo‘lsak, **values** funksiyasidan foydalanamiz yoki yuqoridagidan biroz boshqacharoq tarzda amalga oshiramiz.

Quyidagi misolda har 2 ta usuldan ham foydalangan holda qiymatlarni ekranga chiqaramiz:

```
mashina = {
   "brend": "Chevrolet",
   "modeli": "Gentra",
   "yil": 2023
}
 #1-usul for x in mashina:
   print(mashina[x])
 #2-sul for x in mashina.values():
   print(x)
```

Agar kalit va qiymatlarning ikkalasiga ham bir vaqtda murojaat qilmoqchi bo‘lsak, biz **items()** funksiyasidan foydalanamiz:

```
mashina = {
   "brend": "Chevrolet",
    "modeli": "Gentra",
"yili": 2023}
for x,y in mashina.items():
   print(x,y)
```

```
brend Chevrolet
modeli Gentra
yili 2023
```

Kalit so‘z mavjudligini aniqlash

Biror bir kalit to‘plamda bor yoki yo‘q ekanligini aniqlash uchun **in** kalit so‘zi ishlatiladi:

```
mashina = {
   "brend": "Chevrolet",
 "model": "Gentra",
"yili": 2023
}
if "yili" in mashina:
   print("Ha, mavjud") else:
   print("Yo‘q mavjud emas")
```

```
Ha, mavjud
```

Lug‘atning funksiya va metodlari

Dict.clear()	lug‘atni tozalaydi
Dict.copy()	lug‘at nusxasini qaytaradi.
dict.fromkeys(seq[, value])	Seq dan kalitni va Value qiymatlariga ega bo‘lgan lug‘atni yaratadi
Dict.get(key[, default])	kalit qiymatini qaytaradi, lekin u bo‘lmasa xatolik beradi, default (jimlikda None) qaytaradi.
Dict.items()	juftliklarni qaytaradi (kalit, qiymat)
Dict.keys()	lug‘atdagi kalitlarni qaytaradi
Dict.pop(key[default])	kalitni yo‘qotib qiymatni qaytaradi. Agarda kalit bo‘lmasa defaultni qaytaradi.
Dict.popitem()	juftlikni o‘chirib qaytaradi (kalit, qiymat). Agarda lug‘at bo‘sh bo‘lsa KeyError istisnoni chaqiradi. Esingizda tursin lug‘atlar tartibli emas.
Dict.setdefault(key [, default])	kalit qiymatni qaytaradi, lekin u bo‘lmasa xatolik bermaydi, default qiymatga ega kalitni yaratadi (jimlikda None).
Dict.update([other])	other dan juftliklarni (kalit, qiymat) kiritib lug‘atni to‘ldiradi. Mavjud bo‘lgan kalitlar qaytadan yoziladilar. None (eski lug‘at) qaytaradi.
Dict.values()	lug‘atdagi qiymatni qaytaradi.

Dictionary uzunligi

Dictionary to‘plamida nechta element, ya’ni **kalit-qiymat** juftligi borligini aniqlash uchun **len()** funksiyasidan foydalanamiz:

```
avto = {
  "brend": "Chevrolet",
  "model": "Malibu",
  "yil": 2016
}
print(len(avto))
```

```
3
```

Element qo‘shish

Yangi elementni, ya’ni **kalit-qiymat** juftligini qo‘shish quyidagicha amalga oshiriladi. Masalan, biz mashinamizning rangi haqida ma’lumot beruvchi element qo‘shamiz:

```
mashina = {
  "brend": "Chevrolet",
  "modeli": "Gentra",
  "yili": 2023 }
```

```
{'brend': 'Chevrolet', 'modeli': 'Gentra', 'yili': 2023, 'rangi': 'oq'}
```

```
mashina["rangi"] = "oq"
print(mashina)
```

Elementlarni o‘chirish

Dictionary to‘plamidan elementni o‘chirishning turli xil yo‘llari mavjud. Barchasini birma-bir ko‘rib chiqamiz:

Birinchi usul – **pop()** funksiyasi yoki **del** kalit so‘zi yordamida. Ikkalasi ham ko‘rsatilgan kalit bo‘yicha elementni o‘chiradi. Hozir ularni qanday ishlatishni ko‘ramiz:

```
mashina = {
  "brend": "Chevrolet",
  "modeli": "Gentra",
  "yili": 2023 }
mashina.pop("model")
print(mashina)
del mashina ["yil"]
print(mashina)
```

```
{'brend': 'Chevrolet', 'yil': 2023}
{'brend': 'Chevrolet'}
```

Keyingi usul – **popitem()** funksiyasi to‘plamga oxirgi bo‘lib kiritilgan elementni o‘chiradi (*Python 3.7 dan oldingi versiyalarda bu funksiya ixtiyoriy biror elementni o‘chiradi*).

```
avto = {
  "brend": "Chevrolet",
  "model": "Malibu",
  "yil": 2016 }
avto.popitem()
print(avto)
```

```
{'brend': 'Chevrolet', 'model': 'Malibu'}
```

Oxirgi usul – esa **clear()** funksiyasidir. Bu funksiya esa to‘plamni bo‘shatadi, barcha elementlarini o‘chiradi. Natijada to‘plam bo‘m-bo‘sh holatga keladi.

Del kalit so‘zi yordamida to‘plamning o‘zini butkul o‘chirish mumkin. Ma’lumki, biz to‘plam nomi bilan undagi biror kalitni ko‘rsatsak, del o‘sha kalit bo‘yicha elementni o‘chiradi. Ammo endi faqat to‘plam nomini kiritsak, bu kalit so‘zi butun to‘plamni o‘chiradi.

Quyidagi misolda avval to‘plamni bo‘shatamiz, keyin esa butkul o‘chiramiz:

```
mashina = {
   "brend": "Chevrolet",
   "model": "Gentra",
   "yil": 2023
}
mashina.clear()
print(mashina)
 del mashina
print(mashina)
```

Nusxa olish

Agar biror dictionary to‘plamidan nusxa olib aynan uning o‘zidek to‘plam yaratmoqchi bo‘lsak, buni maxsus yo‘l bilan qilish kerak. Bu holatda bizga **copy()** yoki **dict()** maxsus funksiyalari yordamga keladi. Har 2 ta funksiyadan ham foydalanishimiz mumkin. Quyidagi masalada buni ko‘rib chiqamiz:

```
mashina = {
   "brend": "Chevrolet",
   "model": "Gentra",
   "yil": 2023
}
mashina2= mashina.copy()
print(mashina2)
mashina3= dict(mashina)
print(mashina2)
```

```
{'brend': 'Chevrolet', 'model': 'Gentra', 'yil': 2023}
{'brend': 'Chevrolet', 'model': 'Gentra', 'yil': 2023}
```

Joylashtirilgan to‘plamlar

Bitta dictionary to‘plamini o‘z ichiga bir nechta ana shunday to‘plam saqlashi mumkin. Buning uchun ularni quyidagicha hosil qilish kerak:

```
mashina = {
  " mashina1": {
     "model": "Gentra",
     "yil": 2020 }
   " mashina2": {
     "model": "Nexia",
     "yil": 2015 }
   " mashina3": {
     "model": "Cobalt",
     "yil": 2023 } }
print(mashina)
```

Agar allaqachon mavjud to‘plamlarni bitta to‘plamga yig‘moqchi bo‘lsangiz, quyidagicha amalga oshiriladi:

```
mashina1= {
"model": " Cobalt ",
      "yil": 2016    }
 mashina2= {
"model": " Nexia ",
      "yil": 2018    }
 mashina3= {
"model": " Gentra ",
     "yil": 2019    }
 mashina = {" mashina1": mashina1,  " mashina2": mashina2,
" mashina3": mashina3}
  print(mashina)
```

setdefault()

setdefault() fuksiyasi ko‘rsatilgan kalit bo‘yicha element qiymatini qaytaradi. Agar bunday kalit to‘plamda mavjud bo‘lmasa, shu kalit va biz ko‘rsatgan qiymatni yangi element sifatida to‘plamga qo‘shadi.

Quyidagi misolda ko‘ramiz, agar model kaliti to‘plamda mavjud bo‘lsa, bizga uning qiymati ko‘rsatilsin. Aks holda shunday kalitga **Cobalt** qiymatini biriktirib, to‘plamga qo‘shilsin. [15]

```
avto = {
  "brend": "Chevrolet",
  "model": "Gentra",
  "yil": 2023
}
x =avto.setdefault("model", "Cobalt")
 print(avto)
```

```
{'brend': 'Chevrolet',
'model': 'Gentra',
'yil': 2023}
```

update()

update() funksiyasi to‘plamga yangi elementni (*kalit-qiymat juftligi*) qo‘shadi. Bunda har bir vaqtning o‘zida istalgancha elementlar qo‘shsa bo‘ladi.

Quyidagi misolda biz to‘plamga yangi element kiritamiz:

[15] https://www.bilimlar.uz/wp-content/uploads/2021/02/k100001.pdf

mashina = {
"brend": "Chevrolet",
"modeli": "Gentra",
"yili": 2023
}
avto.update({"rang": "oq"})
print(avto)
	{'brend': 'Chevrolet', 'modeli': 'Gentra', 'yili': 2023, 'rang‘: 'oq'}

Set (to‘plam)

Pythonda to‘plamlarda ishlash uchun maxsus **set** deb nomlanuvchi ro‘yxat turi mavjud. Pythondagi to‘plam tasodifiy tartibda va takrorlanmaydigan elementlardan tashkil topgan “konteyner” deyiladi. To‘plam elementlari tartiblanmagan va indekslanmagan ko‘rinishda bo‘ladi.

To‘plamni hosil qilish uchun maxsus qavslardan foydalanamiz. Yoki **set()** konstruktorini qo‘llaymiz:

sonlar1 = {1, 3, 5, 7, 9} print(sonlar1) sonlar2 = set((2, 4, 6)) print(sonlar2)	{1, 3, 5, 7, 9} {2, 4, 6}

Set to‘plamining funksiya va metodlari

- **len(s) –** bu to‘plamdagi elementlar soni(to‘plam hajmi)ni hisoblaydi.
- **n in m -** ‘n’ ‘m’ to‘plamga tegishli bo‘ladimi yoki yo‘qmi shuni tekshiradi.
- **set.isdisjoint(other) -**agarda set va other umumiy elementlarga ega bo‘lmasa rost qiymat qaytaradi.
- **set==other -** setning hamma elementlari otherga tegishli bo‘ladi otherni hamma elementlari setga tegishli bo‘ladi.
- **set.issubset(other)** yoki **set<=other**-set ning hamma elementlari otherga tegishli bo‘ladi.
- **set.issuperset(other)** yoki **set>=other -**analogik holat.
- **set.union(other, …)** yoki **|other|…-**bir qancha to‘plamlar birlashmasi.
- **set.intersection(other, …)** yoki **&other&… -** kesib olish.
- **set.difference(other, …)** yoki **-other-… -** other ga tegishli bo‘lmagan setning hamma elementlar to‘plami.

❖ **set.symmetric_difference(other); set^other -** birinchi to‘plamda uchraydigan, lekin ularning ikkala to‘plamning kesishmasida uchramaydigan elementlar.

❖ **set.copy**-to‘plam nusxasi

To‘plamni to‘g‘ridan-to‘g‘ri o‘zgartiradigan operatsiyalar

❖ **Set.update(other, …); set|=other| … -** to‘plam birlashmasi

❖ **Set.intersection_update(other, …); set&=other&…-** to‘plam kesishmasi

❖ **Set.difference_update(other, ...); set -= other | ...** -to‘plam ayirmasi

❖ **Set.symmetric_difference_update(other);set^=other**-birinchi to‘plamda uchraydigan, lekin ularning ikkala to‘plamning kesishmasida uchramaydigan elementlar tashkil topgan to‘plam.

❖ **Set.add(elem) -** to‘plamga element qo‘shadi.

❖ **Set.remove(elem) -** to‘plamdagi elementni o‘chiradi. Agarda ko‘rsatilgan element to‘plamda mavjud bo‘lmasa KeyError ni qaytaradi.

❖ **Set.discard(elem) -** agar to‘plamda ko‘rsatilgan element bo‘lsa uni o‘chiradi.

❖ **Set.pop() -** to‘plamdagi birinchi elementni o‘chiradi, lekin to‘plam elementlari tartib bilan joylashmagani uchun birinchi element qaysiligini aniq ko‘rsatib bo‘lmaydi.

❖ **Set.clear() -** to‘plamni tozaydi. [16]

To‘plamni yaratish uchun **set()** funksiyasidan ham foydalanish mumkin.

Ushbu funksiyadan foydalanib to‘plam yaratilganda parametriga qiymat sifatida ro‘yxat yoki kortej ham berilishi mumkin:

```
tubSonlar = [2,3,5,7,11]
tubSonlarTuplami = set(tubSonlar)
print(tubSonlarTuplami)
```

{2, 3, 5, 7, 11}

Ayniqsa set() funksiyasi bo‘sh to‘plam hosil qilish uchun juda qulay hisobladi:

```
son = set()
print(son)
```

set()

To‘plam uzunligi (to‘plam elementlari soni) ni topish uchun *len*() funksiyasidan foydalaniladi:

```
son = {3,4,5,6}
print(len(son))
```

4

[16] https://www.bilimlar.uz/wp-content/uploads/2021/02/k100001.pdf.page-43

Elementlarga murojaat

To‘plamlar tartiblanmagan ro‘yxat bo‘lganligi uchun ularning elementlariga indeks orqali murojaat qilib bo‘lmaydi. To‘plam elementlariga murojaat qilish uchun **for** siklidan yoki aniq bir element borligini tekshirish uchun **in** kalit so‘zidan foydalanamiz:

```
sonlar = {1, 3, 5, 7, 9}
 for x in sonlar:
    print(x)
 prin("---------\n")
print(3 in sonlar)
```

Element qo‘shish

To‘plam hosil qilingandan so‘ng uning elementlarini o‘zgartirib bo‘lmaydi, biroq yangi element qo‘shishsa bo‘ladi. Agar to‘plamga 1 ta element qo‘shish kerak bo‘lsa, **add()** funksiyasi, yoki bir nechta element qo‘shish kerak bo‘lsa, **update()** funksiyasidan foydalanamiz.

```
sonlar = {1, 3, 5, 7, 9}
sonlar.add(9)
print(sonlar)
sonlar.update([11, 13, 15])
print(sonlar)
```

Elementni o‘chirish

Elementni to‘plamdan o‘chirish uchun **remove()** va **discard()** funksiyalari ishlatiladi. Bu funksiyalarning farqi shundaki, **remove()** funksiyasi bilan o‘chirmoqchi bo‘lgan elementimiz to‘plamda mavjud bo‘lmasa, kod ishga tushganda xatolik ro‘y beradi. **discard()** funksiyasi bilan esa bu holat ro‘y bermaydi.

Quyidagi misolda 2 ta usul bilan elementlarni o‘chiramiz:

```
sonlar = {1, 3, 5, 7, 9}
 sonlar.remove(1)
print(sonlar)
sonlar.discard(5)
print(sonlar)
```

```
{3, 5, 7, 9}
{3, 7, 9}
```

Elementni to‘plamdan **pop()** funksiyasi bilan ham o‘chirish mumkin. Ammo **pop()** funksiyasi xususiyatiga ko‘ra ro‘yxat oxiridagi elementni o‘chiradi. To‘plam esa tartiblanmagan ro‘yxat. Shuning uchun bu funksiya aynan qaysi elementni o‘chirishini oldindan bilolmaymiz. Biroq o‘chirilgan elementni aniqlash mumkin:

mashina = {"damas", "tiko", "matiz"} x = meva.pop() print(mashina)	{matiz, tiko}

Clear()

clear() funksiyasi to‘plamni bo‘shatadi, ya’ni barcha elementlarini o‘chiradi:

mashina={"damas", "matiz", "nexia"} mashina.clear() print(mashina)	set()

del kalit so‘zi to‘plamni butunlay o‘chiradi:

```
mashina = {"damas", "matiz", "nexia"}
del mashina
print(mashina)
```

To‘plamni qo‘shish

To‘plamlarni o‘zaro bir-biriga qo‘shish uchun maxsus funksiyalar mavjud:

union() funksiyasi 2 ta to‘plam elementlarini boshqa bir yangi to‘plamga o‘zlashtiradi. Agarda to‘plamlarda bir xil elementlar uchrab qolsa, ularning faqat bittasi olinadi. [17]

a = {"a", "b", "c", "d"} b = {"c", "e", "e", "f"} c = a.union(b) print(c)	{'f', 'd', 'c', 'b', 'a', 'e'}

update() funksiyasi bir to‘plam elementlarini boshqa biriga qo‘shadi. Bunda ham bir xil elementlar uchrab qolsa, ularning faqat bittasi olinadi.

Nusxa olish

Biror bir to‘plamning xuddi o‘zidek yana 1 ta to‘plam yaratish uchun nusxa olish kerak. Bu uchun **copy()** funksiyasidan foydalanamiz:

a = {"a", "b", "c", "d"} b = a.copy() print(b)	{'b', 'a', 'd', 'c'}

Muhim funksiyalar

[17] https://www.bilimlar.uz/wp-content/uploads/2021/02/k100001.pdf

Hozir biz ko‘rib chiqmoqchi bo‘lgan funksiyalar to‘plamlar bilan ishlash uchun zarur funksiyalardir. Ular to‘plamlarning o‘ziga xos xususiyatlariga tayangan holda ishlab chiqilgan.

difference(), difference_update()

❖ **difference()** funksiyasi bir to‘plamda bor, lekin ikkinchi to‘plamda yo‘q bo‘lgan elementlardan tashkil topgan to‘plam hosil qiladi.

❖ **difference_update()** funksiyasi agar ikkala to‘plamda bir xil elementlar mavjud bo‘lsa, o‘sha elementni o‘chiradi.

```
x = {"a", "b", "c", "d"}
y = {"g", "c", "e", "d"}
z = x.difference(y)
print(z)
x.difference_update(y)
print(x)
```

```
{'a', 'b'}
{'a', 'b'}
```

interselection(), interselection_update()

❖ **interselection()** funksiyasi qaysi elementlar ikkala to‘plamda ham mavjud bo‘lsa, o‘sha elementlardan tashkil topgan yangi to‘plam hosil qiladi.

❖ **interselection_update()** funksiyasi **x** to‘plamdagi element **y** to‘plamda ham mavjud bo‘lsa, o‘sha elementni qoldiradi. Qolganlarini o‘chirib yuboradi. [18]

```
x = {"a", "b", "c", "d"}
y = {"g", "c", "e", "d"}
z = x.intersection(y)
print(z)
x.intersection_update(y)
print(x)
```

```
{'c', 'd'}
{'c', 'd'}
```

isdisjoint()

isdisjoint() funksiyasi agarda birinchi to‘plamdagi birorta ham element ikkinchi to‘plamda mavjud bo‘lmasa, rost qiymat qaytaradi.

Quyidagi dasturda rost qiymat qaytariladi. Chunki birinchi to‘plamdagi elementlarning hech biri ikkinchi to‘plamda majud emas:

```
x = {"a", "b", "c"}
y = {"l", "m", "n", "o"}
z = x.isdisjoint(y)
```

```
True
```

[18] www.bilimlar.uz/wp-content/uploads/2021/02/k100001.pdf

```
print(z)
```

issubset(), issuperset()

❖ **issubset()** funksiyasi agar birinchi to‘plamdagi barcha elementlar ikkinchi to‘plamda ham mavjud bo‘lsa, rost qiymat qaytaradi.

❖ **issuperset()** funksiyasi esa teskarisi, ya’ni agar ikkinchi to‘plamdagi barcha elementlar birinchi to‘plamda ham mavjud bo‘lsa, rost qiymat qaytaradi.

Quyidagi misolimizda birinchi to‘plamdagi barcha elementlar ikkinchi to‘plamda mavjud, ammo ikkinchi to‘plamdagi barcha elementlar ham birinchi to‘plamda mavjud emas. Shuning uchun avval rost, keyin esa yolg‘on qiymat ekranga chiqadi:

```
x= {"a", "b", "c"}
y= {"l", "m", "n", "o", "k", "q", "t", "b"}
z= x.issubset(y)
print(z)
z = x.issuperset(y)
print(x)
```

```
False
{'b', 'a', 'c'}
```

Symmetric_difference(), symmetric_difference_update()

❖ **symmetric_difference()** funksiyasi 2 ta to‘plamda ham mavjud bo‘lgan bir xil elementlardan tashqari barcha elementlarni olib yangi to‘plam hosil qiladi.

❖ **symmetric_difference_update()** funksiyasi birinchi to‘plamga ikkinchi to‘plamdan o‘zida mavjud bo‘lmagan barcha elementlarni olib qo‘shadi.

```
x= {"a", "b", "c"}
y= {"l", "c", "a", "o", "k", "t", "b"}
z= x.symmetric_difference(y)
print(z)
z = x.symmetric_difference_update(y)
print(x)
```

```
{'l', 'o‘, 't', 'k'}
{'l', 'o‘, 't', 'k'
```

Frozenset. *frozenset* - o‘zgartirib bo‘lmaydigan to‘plamlarni yaratish uchun ishlatiladi. Ushbu turdagi to‘plamga yangi element qo‘shish, o‘chirish yoki element qiymatini o‘zgartirishga ruxsat berilmaydi. *Frozenset* turidagi to‘plam odatda ro‘yxat, kortej yoki oddiy to‘plam (*set*) orqali hosil qilinadi:

```
famil = {"Axmad", "Sardor", "Ikrom"}
fam = frozenset(famil)
print(fam)
```

```
frozenset({'Axmad', 'Sardor', 'Ikrom'})
```

Frozenset turidagi to‘plamlar ustida quyidagi amallarni bajarish mumkin:

len(s) – *s* to‘plam uzunligi (elementlari soni)ni qaytaradi;

x in s – *True* qiymat qaytaradi, agar *x* element s to‘plamning tarkibida mavjud bo‘lsa; *x not in s* – *True* qiymat qaytaradi, agar *x* element s to‘plamning tarkibida mavjud bo‘lmasa;

s.issubset(t) – *True* qiymat qaytaradi, agar *t* to‘plam *s* to‘plamni o‘z ichiga olsa;

s.issuperset(t) – *True* qiymat qaytaradi, agar s to‘plam t to‘plamni o‘z ichiga olsa;

s.union(t) – *s* va *t* to‘plamlarning birlashmasidan tashkil topgan yangi to‘plamni qaytaradi;

s.intersection(t) – *s* va *t* to‘plamlarning kesishmasidan tashkil topgan yangi to‘plamni qaytaradi;

s.difference(t) – *s* to‘plamdan *t* to‘plamni ayirishdan hosil bo‘lgan yangi to‘plamni qaytaradi;

s.copy() – *s* to‘plamning nusxasini qaytaradi.

Misol. Lug‘atning ma'lumotlar turini chop eting:

```
thisdict = {
  "brand": "Chevrolet",
  "modeli": "Malibu",
  "yili": 2015
}
print(type(thisdict)
```

```
class 'dict'>
```

Misol. Lug‘at yaratish uchun dict() usulidan foydalanish:

```
thisdict = dict(name = "Sardor",
age = 18, country = "Uzbekistan")
print(thisdict)
```

```
{'name': 'Sardor', 'age': 18, 'country': 'Uzbekistan'}
```

Misol. Asl lug‘atga yangi element qo‘shing va kalitlar ro‘yxati ham yangilanishini ko‘ring:

```
car = {
"brand": "Chevrolet",
"modeli": "Malibu",
"yili": 2020
}
```

```
dict_keys(['brand', 'model', 'year'])
dict_keys(['brand', 'model', 'year', 'color'])
```

x = car.keys() print(x) #before the change car["color"] = "white" print(x) #after the change	

Misol. Lug‘atda "model" mavjudligini tekshiring:

thisdict = {
"brand": "Chevrolet",
"modeli": "Malibu",
"yili": 2020
}
if "modeli" in thisdict:
print("Yes, 'modeli' is one of the keys in the thisdict dictionary")
	Yes, 'model' is one of the keys in the thisdict dictionary

Misol. "Yil" ni 2020 yilga o‘zgartiring:

thisdict = {
"brand": "Chevrolet",
"model": "Malibu",
"year": 2013
}
thisdict["year"] = 2020
print(thisdict)
	{'brand': Chevrolet, 'model': Malibu‘, 'year': 2020}

Misol. Update() yordamida avtomobilning "yilini" yangilang:

thisdict = {
"brand": "Chevrolet",
"model": "Malibu",
"year": 2015
}
thisdict.update({"year": 2023})
print(thisdict)
	{'brand': 'Chevrolet', 'model': 'Malibu‘, 'year': 2023}

Misol. Usul yordamida lug‘atga rangli element qo‘shing update():

thisdict = {
"brand": "Chevrolet",
"model": "Malibu",
"year": 2015
}
thisdict.update({"color": "red"})
	{'brand': 'Chevrolet', 'model': 'Malibu‘, 'year': 2015, 'color': 'red'}

Misol. Usul pop()belgilangan kalit nomi bilan elementni olib tashlaydi:

thisdict = {
"brand": "Chevrolet",
"model": "Malibu",
"year": 2015
}
thisdict.pop("model")
print(thisdict)
	{'brand': 'Chevrolet', 'year': 2015}

Misol. Usul popitem()oxirgi kiritilgan elementni olib tashlaydi.

thisdict = {
"brand": "Chevrolet",
"model": "Malibu",
"year": 2015
}
thisdict.popitem()
print(thisdict)
	{'brand': 'Chevrolet', 'model': 'Malibu‘}

Misol. Kalit delso‘z belgilangan kalit nomi bilan elementni olib tashlaydi:

thisdict = {
"brand": "Chevrolet",
"model": "Malibu",
"year": 2015
}
del thisdict["model"]
print(thisdict)
	{'brand': 'Chevrolet', 'year': 2015}

13-MAVZU. PYTHON DASTURLASH TILIDA MASSIVLAR VA ULARDAN FOYDALANISH

Reja:

1. Massivlar haqida ma`lumot;
2. Massiv elementlari ustida aniqlangan amallar;
3. Ikki o‘lchovli massivlar;
4. Ikki o‘lchovli massivlarga oid dasturlar;

Tayanch so‘zlar: massiv, bir o‘lchovli massiv, ikki o‘lchovli massiv, random, numpy.

Massiv nima?

Massiv - bu bir vaqtning o‘zida bir nechta qiymatlarni ushlab turishi mumkin bo‘lgan maxsus o‘zgaruvchi.

Agar sizda elementlar ro‘yxati (masalan, avtomobil nomlari ro‘yxati) bo‘lsa, mashinalarni bitta o‘zgaruvchida saqlash quyidagicha ko‘rinishi mumkin:

```
car1 = "Ford"
car2 = "Volvo"
car3 = "BMW"
```

Biroq, agar siz mashinalar orasidan aylanib, ma'lum birini topmoqchi bo‘lsangiz-chi? Va agar sizda 3 ta emas, balki 300 ta mashina bo‘lsa-chi?

Yechim - massiv!

Massiv bitta nom ostida ko‘p qiymatlarni saqlashi mumkin va siz indeks raqamiga murojaat qilish orqali qiymatlarga kirishingiz mumkin.

Massiv elementlariga kirish

Kvadrat qavs yordamida biz massiv elementlariga murojaat qilishimiz mumkin. Indeks raqamiga murojaat qilib, massiv elementiga murojaat qilasiz.

Misol: Birinchi qator elementining qiymatini oling:

x = cars[0]

```
cars = ["Ford", "Volvo", "BMW"]
x = cars[0]
print(x)
```

```
Ford
```

Misol: Birinchi qator elementining qiymatini o‘zgartiring:

cars[0] = "Toyota"

```
cars = ["Ford", "Volvo", "BMW"]
cars[0] = "Toyota"
print(cars)
```

```
['Toyota', 'Volvo', 'BMW']
```

Massiv uzunligi

len() Massiv uzunligini (massivdagi elementlar soni) qaytarish uchun usuldan foydalaning.

Misol: Massivdagi elementlar sonini qaytaring cars:

x = len(cars)

```
cars = ["Ford", "Volvo", "BMW"]
x = len(cars)
print(x)
```

```
3
```

Massiv elementlarini aylanish

Siz for in massivning barcha elementlari bo‘ylab aylanish uchun siklдan foydalanishingiz mumkin.

Misol: Massivdagi har bir elementni chop eting cars:

```
cars = ["Ford", "Volvo", "BMW"]
for x in cars:
  print(x)
```

```
Ford
Volvo
BMW
```

Massiv elementlarini qo‘shish

append() Massivga element qo‘shish uchun usuldan foydalanishingiz mumkin.

Misol: Massivga yana bitta element qo‘shing cars:

cars.append("Honda")

```
cars = ["Ford", "Volvo", "BMW"]
cars.append("Honda")
print(cars)
```

```
['Ford', 'Volvo', 'BMW', 'Honda']
```

Massiv elementlarini olib tashlash

pop() Massivdan elementni olib tashlash uchun usuldan foydalanishingiz mumkin.

Misol: Massivning ikkinchi elementini o‘chiring cars:

cars.pop(1)

```
cars = ["Ford", "Volvo", "BMW"]
cars.pop(1)
print(cars)
```

```
['Ford', 'BMW']
```

remove() Massivdan elementni olib tashlash uchun ham usuldan foydalanishingiz mumkin.

Misol: "Volvo" qiymatiga ega bo‘lgan elementni o‘chiring:

cars.remove("Volvo")

```
cars = ["Ford", "Volvo", "BMW"]
cars.remove("Volvo")
print(cars)
```

```
['Ford', 'BMW']
```

Eslatma: Ro‘yxat **remove()** usuli faqat ko‘rsatilgan qiymatning birinchi takrorlanishini olib tashlaydi.

Massiv usullari

Pythonda siz ro‘yxatlar/massivlarda foydalanishingiz mumkin bo‘lgan o‘rnatilgan usullar to‘plami mavjud.

Usul	Tavsif
append()	Ro‘yxat oxiriga element qo‘shadi
clear()	Ro‘yxatdagi barcha elementlarni olib tashlaydi
copy()	Ro‘yxatning nusxasini qaytaradi
count()	Belgilangan qiymatga ega elementlar sonini qaytaradi
extend()	Joriy ro‘yxatning oxiriga ro‘yxat elementlarini (yoki har qanday takrorlanadigan) qo‘shing
index()	Belgilangan qiymat bilan birinchi elementning indeksini qaytaradi
insert()	Belgilangan joyga element qo‘shadi
pop()	Belgilangan joydagi elementni olib tashlaydi
remove()	Belgilangan qiymatga ega birinchi elementni olib tashlaydi
reverse()	Ro‘yxat tartibini o‘zgartiradi
sort()	Ro‘yxatni tartiblaydi

Eslatma: Pythonda massivlar uchun o‘rnatilgan yordam yo‘q, lekin uning o‘rniga Python ro‘yxatlaridan foydalanish mumkin.

array.typecode **-** Massivning elementlari turini aniqlash uchun ishlatiladi. Agar massivlar bir nechta bo‘lsa array.array(typecode) dan foydalaniladi.

```
from array import *
massiv = array('i', [2, 5, 4, 0, 8])
print(massiv.typecode)
```

```
Python 3.12.0 (tags/v3.12.0:0fb18b0,
AMD64)] on win32
Type "help", "copyright", "credits" (

= RESTART: D:\Madadjon\O`ktamov.M.py
i
```

array.itemsize **-** massivdagi bitta elementning baytdagi hajmini hisoblash uchun ishlatiladi.

```
from array import *
massiv = array('i', [2, 5, 4, 0,8])
print(massiv.itemsize)
```

```
Python 3.12.0 (tags/v3.12.0:0fb18b0,
AMD64)] on win32
Type "help", "copyright", "credits"

= RESTART: D:\Madadjon\O`ktamov.M.py
4
```

array.count*(x)* - massivdagi *x* elementlar sonini qiymat sifatida qaytaradi;

```
from array import *
massiv = array('i', [2,2, 5, 4,4,4, 0,8])
print(massiv.count(4));
```

```
Python 3.12.0 (tags/v3.12.0:0fb18b0,
AMD64)] on win32
Type "help", "copyright", "credits" (

= RESTART: D:\Madadjon\O`ktamov.M.py
3
```

array.fromlist(ro‘yxat)–massivga ro‘yxatdagi elementlarni qo‘shish uchun ishlatiladi.

```
from array import*
massiv= array('i',[1,2,3,4,5])
list=[6,7,8] massiv.fromlist(list)
print(massiv);
```

```
Python 3.12.0 (tags/v3.12.0:0fb18b0, Oct  2
AMD64)] on win32
Type "help", "copyright", "credits" or "lic

========== RESTART: D:\Madadjon\
array('i', [1, 2, 3, 4, 5, 6, 7, 8])
```

array.index(x) **–** massivdagi *x* elementining joylashgan indeksini qiymat sifatida qaytaradi. Agar bunday element massivda mavjud bo‘lmasa, u holda *ValueError* istisno holati ro‘y beradi;

```
from array import *
massiv = array('i', [2, 5, 4, 0, 8])
print(massiv.index(8))
```

```
Python 3.12.0 (tags/v3.12.0:0fb18b0,
AMD64)] on win32
Type "help", "copyright", "credits" (

= RESTART: D:\Madadjon\O`ktamov.M.py
4
```

append() – metodi massivning oxiriga yangi element qo‘shish uchun foydalaniladi.

```
from array import *
massiv = array('i', [1,2,3,4,5])
massiv.append(9)
print(massiv)
```

```
Python 3.12.0 (tags/v3.12.0:0fb18b0,
AMD64)] on win32
Type "help", "copyright", "credits"

= RESTART: D:\Madadjon\O`ktamov.M.py
array('i', [1, 2, 3, 4, 5, 9])
```

array.remove*(x)* - massivdan *x* elementini o‘chirish. Ushbu metod ro‘yxatdagi birinchi uchragan *x* elementini o‘chiradi. Agar bunday element ro‘yxatda mavjud bo‘lmasa *ValueError* istisno holati ro‘y beradi.

```
from array import *
massiv = array('i', [1,2,3,4,5])
massiv.remove(3)
print(massiv)
```

```
Python 3.12.0 (tags/v3.12.0:0fb18b0,
AMD64)] on win32
Type "help", "copyright", "credits"

= RESTART: D:\Madadjon\O`ktamov.M.py
array('i', [1, 2, 4, 5])
```

array.reverse() - massiv elementlarini teskari tartibda joylashtirish uchun qo‘llaniladi . Bundan tashqari, Python massiv bilan ishlashda qo‘llaniladigan bir nechta standart funksiyalarni ham o‘z ichiga qamrab olgan:

```
from array import *
massiv = array('i', [1,2,3,4,5])
massiv.reverse() print(massiv)
```

```
Python 3.12.0 (tags/v3.12.0:0fb18b0,
AMD64)] on win32
Type "help", "copyright", "credits"

= RESTART: D:\Madadjon\O`ktamov.M.py
array('i', [5, 4, 3, 2, 1])
```

array.tolist() -massivni ro‘yxatga aylantirish uchun qo‘llaniladi.

```
from array import *
massiv = array('i', [1,2,3,4,5])
list=massiv.tolist()
print(list)
```

```
Python 3.12.0 (tags/v3.12.0:0fb18b0,
AMD64)] on win32
Type "help", "copyright", "credits"

= RESTART: D:\Madadjon\O`ktamov.M.py
[1, 2, 3, 4, 5]
```

array.tofile*(f)* - massivni ochiq faylga yozish uchun ishlatiladi.

array.fromfile*(F,N)* - fayldan N elementni o‘qiydi va ularni massiv oxiriga qo‘shib qo‘yadi. Ikkilik o‘qish uchun fayl ochilishi kerak. Agar N dan kam element mavjud bo‘lsa, *ValueError* istisnoli tashlanadi, ammo mavjud bo‘lgan elementlar qatorga qo‘shiladi.

```
import array
f=open("array.bin","wb")
massiv=array.array("i",[1,2,3,4,5])
massiv.tofile(f) f.close()
massiv1=array.array("i")
f=open("array.bin","rb")
massiv1.fromfile(f,len(massiv))
print(massiv1)
```

```
Python 3.12.0 (tags/v3.12.0:0fb
AMD64)] on win32
Type "help", "copyright", "cred

========== RESTART:
array('i', [1, 2, 3, 4, 5])
```

array.buffer_info() - tuple(kortej) xotiraning joylashuvi, uzunligini aniqlaydi. Past darajadagi operatsiyalar uchun foydalidir.

```
import array
massiv=array.array('i',[1,2,3,4,5])
print(massiv.buffer_info())
```

```
Python 3.12.0 (tags/v3.12.0:0fb18b0,
AMD64)] on win32
Type "help", "copyright", "credits" c

= RESTART: D:\Madadjon\O`ktamov.M.py
(1881031319568, 5)
```

array.byteswap() - massivning har bir elementida baytlarning tartibini o‘zgartirish. Chunki boshqa bayt tartibida mashinada yozilgan fayldan ma'lumotlarni o‘qishda foydalidir;

```
from array import array
my_array=array('i',[1,2,3,4,5])
my_array.byteswap()
print(my_array)
```

```
Python 3.12.0 (tags/v3.12.0:0fb18b0, Oct  2 2023, 13:03:39) [MSC
AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more info

= RESTART: D:\Madadjon\O`ktamov.M.py
array('i', [16777216, 33554432, 50331648, 67108864, 83886080])
```

array.extend(iter) - massivga ob’ektdan elementlarni qo‘shish uchun foydalanadi.

```
from array import array
my_array=array('i')
my_array.extend([1,2,3,4,5])
print(my_array)
my_array.extend(range(6,10))
print(my_array)
my_array.extend(array('i',[44,55,66]))
print(my_array)
```

```
Python 3.12.0 (tags/v3.12.0:0fb18b0, Oct  2 2023, 13
AMD64)] on win32
Type "help", "copyright", "credits" or "license()" f

= RESTART: D:\Madadjon\O`ktamov.M.py
array('i', [1, 2, 3, 4, 5])
array('i', [1, 2, 3, 4, 5, 6, 7, 8, 9])
array('i', [1, 2, 3, 4, 5, 6, 7, 8, 9, 44, 55, 66])
```

array.frombytes *(b)* - bir qator baytlardan massiv hosil qiladi. Baytlar soni massivdagi bitta element kattaligining ko‘paytmasi bo‘lishi kerak.

```
from array import array
my_array = array('b')
my_array.frombytes(b'123456789')
print(my_array)
```

```
Python 3.12.0 (tags/v3.12.0:0fb18b0, Oct  2 2023, 13
AMD64)] on win32
Type "help", "copyright", "credits" or "license()" f

= RESTART: D:\Madadjon\O`ktamov.M.py
array('b', [49, 50, 51, 52, 53, 54, 55, 56, 57])
```

Pythonda ikki va ko‘p o‘lchovli massivlar

Ikki o‘lchovli massiv. Ba’zi hollarda oddiy bir o‘lchovli massiv ma’lum bir ma’lumot to‘plamini to‘g‘ri ko‘rsatish uchun etarli emas. Python dasturlash tilida ikki o‘lchovli va ko‘p o‘lchovli massivlar mavjud emas, ammo ushbu platformaning asosiy imkoniyatlari ikki o‘lchovli ro‘yxatni tuzishni osonlashtiradi. Ushbu dizayn elementlari quyidagi misolda ko‘rsatilgandek to‘ldirilib, ustunlar va qatorlarga joylashtirilgan.

```
from array import *
d1 = []
for j in range(7):
    d2 = []
for i in range(7):
    d2.append(i)
    d1.append(d2)
for i in d1:
    print(i)
```

```
Python 3.12.0 (tags/v3.12.0:0fb18b0,
AMD64)] on win32
Type "help", "copyright", "credits"

= RESTART: D:\Madadjon\O`ktamov.M.py
[0, 1, 2, 3, 4, 5, 6]
[0, 1, 2, 3, 4, 5, 6]
[0, 1, 2, 3, 4, 5, 6]
[0, 1, 2, 3, 4, 5, 6]
[0, 1, 2, 3, 4, 5, 6]
[0, 1, 2, 3, 4, 5, 6]
[0, 1, 2, 3, 4, 5, 6]
```

Bu yerda biz ikki o‘lchovli ma'lumotlar to‘plamini amalga oshirishning asosiy g‘oyasi bitta katta d1 ro‘yxati ichida bir nechta d2 ro‘yxatlarini yaratish ekanligini ko‘rishimiz mumkin. Ikki o‘lchamli 5×5 matritsani nol bilan avtomatik to‘ldirish uchun ishlatiladi. Qo‘shish va diapazon usullari ushbu vazifani yengishga yordam beradi, ularning birinchisi ro‘yxatga yangi element qo‘shadi (0), ikkinchisi esa uning qiymatini (5) o‘rnatishga imkon beradi. Shuni takidlash kerakki, for uchun har bir yangi tashqi (j) yoki ichki (i) ro‘yxatlarning joriy elementini ifodalovchi o‘z

vaqtinchalik o‘zgaruvchisidan foydalanadi. Ko‘p o‘lchovli ro‘yxatning kerakli katakchasiga uning koordinatalarini to‘rtburchaklar ichida ko‘rsatib, satrlar va ustunlarga e'tibor qaratishingiz mumkin: d1[1][2].

Ko‘p o‘lchovli massiv. Murakkab ro‘yxat sifatida ko‘rsatilgan ikki o‘lchovli qatorda bo‘lgani kabi, ko‘p o‘lchovli qator ham ro‘yxat ichida ro‘yxat tarzida amalga oshiriladi. Quyidagi misolda uch o‘lchamli(5x5x5) massiv yaratishni ko‘rib chiqamiz:

```
from array import *
d1 = []
for k in range(5):
   d2 = []
for j in range(5):
   d3 = []
for i in range(5):
   d3.append(i)
   d2.append(d3)
   d1.append(d2)
for i in d1:
   print(i)
```

```
Python 3.12.0 (tags/v3.12.0:0fb18b0, Oct  2 2023, 13:03:39) [MSC v.1935 64 bit (AMD64)]
2
Type "help", "copyright", "credits" or "license()" for more information.

= RESTART: D:\Madadjon\O`ktamov.M.py
[[0, 1, 2, 3, 4], [0, 1, 2, 3, 4], [0, 1, 2, 3, 4], [0, 1, 2, 3, 4], [0, 1, 2, 3, 4]]
[[0, 1, 2, 3, 4], [0, 1, 2, 3, 4], [0, 1, 2, 3, 4], [0, 1, 2, 3, 4], [0, 1, 2, 3, 4]]
[[0, 1, 2, 3, 4], [0, 1, 2, 3, 4], [0, 1, 2, 3, 4], [0, 1, 2, 3, 4], [0, 1, 2, 3, 4]]
[[0, 1, 2, 3, 4], [0, 1, 2, 3, 4], [0, 1, 2, 3, 4], [0, 1, 2, 3, 4], [0, 1, 2, 3, 4]]
[[0, 1, 2, 3, 4], [0, 1, 2, 3, 4], [0, 1, 2, 3, 4], [0, 1, 2, 3, 4], [0, 1, 2, 3, 4]]
```

Ikki o‘lchovli massivga o‘xshab, to‘rtburchaklar ichidagi ko‘rsatkichlar yordamida yuqorida qurilgan obyekt katakchasiga murojaat qilishimiz mumkin.

Masalan, d1 [4] [2] [3].

Massivlar odatda Python dasturlash tilidagi bir xil turdagi ma’lumotlar to‘plamlari bilan o‘zaro aloqada bo‘lish uchun ishlatiladi. Platformaning standart kutubxonasi sizga tegishli funksiyalar yordamida uning tarkibini boshqarish qobiliyatini ta’minlaydigan bunday tuzilma bilan samarali ishlashga imkon beradi. Bundan tashqari, Python sathlar soniga cheklovlarsiz ro‘yxatlarning ko‘p o‘lchovli namoyishini qo‘llab-quvvatlaydi.

1-masala. n natural soni berilgan. Dastlabki n ta toq sondan tashkil topgan massivni hosil qiling va elementlarini chiqaring.

```
import numpy as np
n=int(input('n='))
toq = np.array(range(1,n+1,2), float)
print(toq)
```

```
Python 3.12.1 (tags/v3.12.1:2305ca5, Dec
AMD64)] on win32
Type "help", "copyright", "credits" or "li

= RESTART: D:\Madadjon\O`ktamov.M.py
n=20
[ 1.  3.  5.  7.  9. 11. 13. 15. 17. 19.]
```

2-masala. n natural soni berilgan. 2 sonining dastlabki n ta darajasidan tashkil topgan massivni hosil qiling va elementlarini chiqaring. (1, 2, 4, 8)

```
import numpy as np
n=int(input('n='))
a = np.array(range(n+1), int)
for i in range(n+1):
    a[i]=2**i print(a)
```

```
Python 3.12.1 (tags/v3.12.1:2305ca5,
AMD64)] on win32
Type "help", "copyright", "credits" c

= RESTART: D:\Madadjon\O`ktamov.M.py
n=5
[ 1  2  4  8 16 32]
```

3-masala. n natural soni va arifmetik progressiyaning dastlabki hadi A va ayirmasi D berilgan. Arifmetik progressiyaning dastlabki n ta hadidan tashkil topgan massivni hosil qiling va elementlarini chiqaring.

```
import numpy as np
n=int(input('n='))
A=int(input('A='))
D=int(input('D='))
a = np.array(range(n), int)
for i in range(0,n):
    a[i]=A+D*i
print(a)
```

```
Python 3.12.1 (tags/v3.12.1:2305ca5,
AMD64)] on win32
Type "help", "copyright", "credits"

= RESTART: D:\Madadjon\O`ktamov.M.py
n=7
A=1
D=4
[ 1  5  9 13 17 21 25]
```

4-masala. n natural soni va geometrik progressiyaning dastlabki hadi b va maxraji q berilgan. Geometrik progressiyaning dastlabki n ta hadidan tashkil topgan massivni hosil qiling va elementlarini chiqaring.

```
import numpy as np
n=int(input('n='))
b=int(input('b='))
q=int(input('q='))
massiv = np.array(range(n), int)
massiv[0]=b for i in range(1,n):
    massiv[i]=massiv[i-1]*q
print(massiv)
```

```
Python 3.12.1 (tags/v3.12.1:2305ca5,
AMD64)] on win32
Type "help", "copyright", "credits" o

= RESTART: D:\Madadjon\O`ktamov.M.py
n=7
b=1
q=2
[ 1  2  4  8 16 32 64]
```

5-masala. n natural soni berilgan. Dastlabki n ta Fibonachi sonlaridan tashkil topgan massivni hosil qiling va elementlarini chiqaring.

F[0] = 1; F[1] = 1; F[k] = F[k-1] + F[k-2]; k=2, 3, 4, ...

```
import numpy as np
n=int(input('n='))
massiv = np.array(range(n), int)
massiv[0]=1
massiv[1]=1
for i in range(2,n):
massiv[i]=massiv[i-1]+massiv[i-2]
print(massiv)
```

```
Python 3.12.1 (tags/v3.12.1:2305ca5,
AMD64)] on win32
Type "help", "copyright", "credits"

= RESTART: D:\Madadjon\O`ktamov.M.py
n=7
[ 1  1  2  3  5  8 13]
```

6-masala. n natural soni va A, B butun sonlari berilgan (n > 2). a[0] = A; a[1] = B; boshqa elementlari o‘zidan oldingi barcha elementlari yig‘indisiga teng bo‘lgan massivni hosil qiling va elementlarini chiqaring.

```
import numpy as np
n=int(input('n='))
A=int(input('A='))
B=int(input('B='))
massiv = np.array(range(n), float)
massiv[0]=A massiv[1]=B
s=A+B
for i in range(2,n):
   massiv[i]=s
s+=massiv[i]
print(massiv)
```

```
Python 3.12.1 (tags/v3.12.1:2305ca5,
AMD64)] on win32
Type "help", "copyright", "credits" c

= RESTART: D:\Madadjon\O`ktamov.M.py
n=7
A=1
B=3
[ 1.  3.  4.  8. 16. 32. 64.]
```

7-masala. n ta elementdan tashkil topgan massiv berilgan. uning elementlarini teskari tartibda chiqaruvchi programma tuzilsin.

```
import numpy as np
n=int(input('n='))
massiv = np.array(range(n), int)
for i in range(0,n):
massiv[i]=int(input('massiv[i]='))
for i in range(n-1,-1,-1):
print(massiv[i],end=';')
```

```
Python 3.12.1 (tags/v3.12.1:2305ca5,
AMD64)] on win32
Type "help", "copyright", "credits"

= RESTART: D:\Madadjon\O`ktamov.M.py
n=7
massiv[i]=1
massiv[i]=2
massiv[i]=3
massiv[i]=4
massiv[i]=5
massiv[i]=6
massiv[i]=7
7;6;5;4;3;2;1;
```

8-masala. n ta elementdan tashkil topgan massiv berilgan. Massiv elementlari orasidan toqlarini chiqaruvchi va ularning sonini chiqaruvchi programma tuzilsin. Massiv elementlar: 4 5 7 8 6 9 Natija: 5 7 9 toqlar soni = 3

```
    import numpy as np
    n=int(input('n='))
massiv = np.array(range(n), int)
    for i in range(0,n):
 massiv[i]=int(input('massiv[i]='))
    k=0 for i in range(0,n):
       if massiv[i]%2==1:
    print(massiv[i],end=';')
k+=1 print("\n",k," ta toq son")
```

```
Python 3.12.1 (tags/v3.12.1:2305ca5,
AMD64)] on win32
Type "help", "copyright", "credits" (

= RESTART: D:\Madadjon\O`ktamov.M.py
n=4
massiv[i]=1
massiv[i]=2
massiv[i]=3
massiv[i]=4
1;3;
 2  ta toq son
```

9-masala. n ta elementdan tashkil topgan massiv berilgan. Massiv elementlari orasidan juftlarini chiqaruvchi va ularning sonini chiqaruvchi programma tuzilsin.

Massiv elementlar: 4 5 7 8 6 9 Natija: 4 8 6 juftlar soni = 3

```
    import numpy as np
    n=int(input('n='))
    massiv = np.array(range(n), int)
    for i in range(0,n):
    massiv[i]=int(input('massiv[i]='))
    k=0 for i in range(0,n):
     if massiv[i]%2==0:
    print(massiv[i],end=';')        k+=1
    print("\n",k," ta juft son")
```

```
Python 3.12.1 (tags/v3.12.1:2305ca5,
AMD64)] on win32
Type "help", "copyright", "credits" (

= RESTART: D:\Madadjon\O`ktamov.M.py
n=4
massiv[i]=1
massiv[i]=2
massiv[i]=3
massiv[i]=4
2;4;
 2  ta juft son
```

10-masala. n ta elementdan tashkil topgan massiv berilgan. Dastlab massiv elementlari orasidan juftlarini chiqaruvchi, keyin massiv elementlari orasidan toqlarini chiqaruvchi programma tuzilsin. Massiv elementlar: 4 5 7 8 6 9 Natija: 4 8 6 5 7 9

```
    import numpy as np
    n=int(input('n='))
    massiv = np.array(range(n), int)
    for i in range(0,n):
       massiv[i]=int(input('massiv[i]='))
    for i in range(0,n):
       if massiv[i]%2==0:
          print(massiv[i],end=';')
    for i in range(0,n):
       if massiv[i]%2==1:
          print(massiv[i],end=';')
```

```
Python 3.12.1 (tags/v3.12.1:2305ca5,
AMD64)] on win32
Type "help", "copyright", "credits" (

= RESTART: D:\Madadjon\O`ktamov.M.py
n=8
massiv[i]=1
massiv[i]=2
massiv[i]=3
massiv[i]=4
massiv[i]=5
massiv[i]=6
massiv[i]=7
massiv[i]=8
2;4;6;8;1;3;5;7;
```

11-masala. m va n butun musbat sonlari berilgan. m x n o‘lchamli matritsani shunday hosil qilingki, uning har bir i - satri elementlari 10 * i ga teng bo‘lsin.

(i =0, 1, m -1)

```
import numpy as np
m=int(input('m='))
n=int(input('n='))
massiv=np.array([[0 for i in range(m)]
                 for j in range(n)])
for i in range(m):
   for j in range(n):
massiv[i][j]=int(input('massiv[i][j]='))
for i in range(m):
   for j in range(n):
      massiv[i][j]*=10
      print(massiv[i][j],end=' ')
   print('')
```

```
Python 3.12.1 (tags/v3.12.1:2305ca5,
AMD64)] on win32
Type "help", "copyright", "credits"

= RESTART: D:\Madadjon\O`ktamov.M.py
m=2
n=2
massiv[i][j]=10
massiv[i][j]=20
massiv[i][j]=30
massiv[i][j]=40
100 200
300 400
```

12-masala. m va n butun musbat sonlari berilgan. m x n o‘lchamli matritsani shunday hosil qilingki, uning har bir j - ustuni elementlari 5 * j ga teng bo‘lsin.

(j =0, 1, ..., n -1)

```
import numpy as np
m=int(input('m='))
n=int(input('n='))
massiv=np.array([[0 for i in range(m)]
for j in range(n)]) for i in range(m):
for j in range(n):
massiv[i][j]=5*j
 print(massiv[i][j],end=' ')
print('')
```

```
Python 3.12.1 (tags/v3.12.1:2305ca5,
AMD64)] on win32
Type "help", "copyright", "credits"

= RESTART: D:\Madadjon\O`ktamov.M.py
m=4
n=4
0 5 10 15
0 5 10 15
0 5 10 15
0 5 10 15
```

13-masala. m va n butun musbat sonlari berilgan. m x n o‘lchamli matritsani shunday hosil qilingki, undagi eng katta elementini toping.

```
import numpy as np
m=int(input('m='))
n=int(input('n='))
massiv=np.array([[0 for i in range(m)]
for j in range(n)]) for i in range(m):
for j in range(n):
massiv[i][j]=int(input('massiv[i][j]='))
print(massiv.max())
```

```
Python 3.12.1 (tags/v3.12.1:2305ca5,
AMD64)] on win32
Type "help", "copyright", "credits"

= RESTART: D:\Madadjon\O`ktamov.M.py
m=2
n=2
massiv[i][j]=1
massiv[i][j]=2
massiv[i][j]=3
massiv[i][j]=4
4
```

14-masala. m va n butun musbat sonlari berilgan. m x n o‘lchamli matritsani shunday hosil qilingki, undagi eng kichik elementini toping.

```
    import numpy as np
    m=int(input('m='))
    n=int(input('n='))
    massiv=np.array([[0 for i in range(m)]
    for j in range(n)])
    for i in range(m):
    for j in range(n):
massiv[i][j]=int(input('massiv[i][j]='))
print(massiv.min())
```

```
Python 3.12.1 (tags/v3.12.1:2305ca5,
AMD64)] on win32
Type "help", "copyright", "credits" c

= RESTART: D:\Madadjon\O`ktamov.M.py
m=2
n=2
massiv[i][j]=1
massiv[i][j]=2
massiv[i][j]=3
massiv[i][j]=4
1
```

15-masala. m x n o‘lchamli massiv berilgan. Shu massivning k-satr ustunini toping.

```
    import numpy as np
    m=int(input('m='))
    n=int(input('n='))
    massiv=np.array([[0 for i in range(m)]
    for j in range(n)])
    k=int(input('k='))
    massiv=[[0 for i in range(m)]
    for j in range(n)] for i in range(m):
    for j in range(n):
massiv[i][j]=int(input('massiv[i][j]='))
    for i in range(m):
print("Massiv[",i,"][",k,"]=",massiv[i][k])
```

```
Python 3.12.1 (tags/v3.12.1:2305ca5,
AMD64)] on win32
Type "help", "copyright", "credits" (

= RESTART: D:\Madadjon\O`ktamov.M.py
m=3
n=3
k=2
massiv[i][j]=1
massiv[i][j]=2
massiv[i][j]=3
massiv[i][j]=4
massiv[i][j]=5
massiv[i][j]=6
massiv[i][j]=7
massiv[i][j]=8
massiv[i][j]=9
Massiv[ 0 ][ 2 ]= 3
Massiv[ 1 ][ 2 ]= 6
Massiv[ 2 ][ 2 ]= 9
```

16-masala. m x n o‘lchamli massiv berilgan. Shu massivning d-satr qatorini toping.

```
    import numpy as np
    m=int(input('m='))
    n=int(input('n='))
massiv=np.array([[0 for i in range(m)]
    for j in range(n)])
    d=int(input('d=')) for i in range(m):
    for j in range(n):
 massiv[i][j]=int(input('massiv[i][j]='))
    for j in range(n):
print("Massiv[",d,"][",j,"]=",massiv[d][j])
```

```
Python 3.12.1 (tags/v3.12.1:2305ca5,
AMD64)] on win32
Type "help", "copyright", "credits"

= RESTART: D:\Madadjon\O`ktamov.M.py
m=2
n=2
d=1
massiv[i][j]=1
massiv[i][j]=2
massiv[i][j]=3
massiv[i][j]=4
Massiv[ 1 ][ 0 ]= 3
Massiv[ 1 ][ 1 ]= 4
```

14-MAVZU. PYTHON DASTURLASH TILIDA SATRLAR VA ULARDAN FOYDALANISH

Reja:

1. Satrlar
2. Satrlar ustida aniqlangan amallar.
3. Satrlarga oid dasturlar.

Tayanch so‘zlar: Satr, ord, chr, str, asciiz.

Pythonda satrlar

Satrlar – bu belgilar ketma-ketligi. Ko‘p hollarda satrlar so‘zlar jamlanmasidan tashkil topadi. Pythonda satrlar bilan ishlash juda qulay. Bir qancha satr literallari mavjud. Pythonda satrlar qo‘shtirnoq yoki bittalik tirnoqlar bilan ifodalanadi. Ularni **print()** funksiyasi bilan ekranga chiqaramiz.

```
print ("Salom") print ('Python')
```

Apostrof va qo‘shtirnoqdagi satrlar bir narsa. Uni ikki xil variantda keltirilishiga sabab literallarga apostrof va qo‘shtirnoq belgilarini maxsus xizmatchi belgilardan foydalanmasdan kiritish mumkinligi deb hisoblanadi.

```
Ism = "San'atbek"
Gap = 'Men "Python dasturlash tili" nomli kitob yozdim'
```

Satrni o‘zgaruvchiga biriktirish

Satrni o‘zgaruvchiga biriktirish uchun tenglik belgisi ishlatiladi:

```
a = "Salom" print(a)
```

Ko‘p qatorli satr

Ko‘p qatorli satrni hosil qilish uchun uchtalik qo‘shtirnoq yoki tirnoqlardan foydalaniladi. Bunda satr qanday holatda yozilgan bo‘lsa shundayligicha natija beradi:

```
a= """ Bu ko‘p qatorli satr """
b= ''' Bu ham ko‘p qatorli  satr '''
print(a)
print(b)
```

Satr konstantalarini birlashtirish uchun ularni yonma-yon joylashtirishning o‘zi kifoya. Python avtomat ularni birlashtiradi. Misol uchun: "Ismingiz" "kim?" avtomat "Ismingiz kim?" ga aylanadi. **Eslatma:** Bir tirnoq va qo‘sh tirnoqdagi satrlar bir-biridan hech ham farq qilmaydi.

Satr – bu massiv

Satrni alohida belgilar ketma-ketligidan tashkil topgan massiv deb hisoblash mumkin. Pythonda belgini ifodalovchi ma’lumot turi yo‘qligi uchun bitta belgi deb, bitta elementdan tashkil topgan satrga aytiladi. Satrning istalgan elementiga murojaat qilish mumkin. Buning uchun kvadrat qavslardan foydalanamiz va elementning satrdagi o‘rnini kiritamiz.

Eslatma: Elementning satrdagi o‘rnini ifodalash uchun sanoq o dan boshlanadi. Ya’ni satrdagi birinchi elementning o‘rni 0 ga, ikkinchi elementning o‘rni 1 ga teng va hokazo.

Quyidagi misolimizda biz ikkinchi elementni ekranga chiqaramiz:

```
a = "Dasturlash" print(a[1])
```

Satrlar ustida amallar

Shunday qilib satrlar bilan ishlash haqida gapirdik, endi satrlarning funksiyalari va metodlari haqida gapiramiz. Quyida satrlarning barcha funksiya va metodlari keltirilgan.

Konkatenatsiyalash (qo‘shish)

```
>>> s1='python'
>>> s2='dasturlash'
>>> s1+s2
>>> 'pythondasturlash'
```

Satrni takrorlash (dublikat qilish)

```
>>> print ('python'*3)
pythonpythonpython
```

Satr uzunligi (len() funksiyasi)

```
>>> gap='bu satrning uzunligi qancha'
>>> len (gap)
27
```

Indeks bo‘yicha chiqarish

```
>>> s='dasturlash'
>>> s[0]
'd'
>>> s[2]
'a'
>>> s[-2]
's'
```

Misoldan ko‘rinib turibdiki Python manfiy indeks bo‘yicha chiqarishga ruxsat etadi, lekin hisoblash qator oxiridan boshlanadi. Satrdan qism ajratib olishni uning oxiridan boshlash uchun manfiy indeks ishlatiladi.

Satrdan qism ya’ni kesma ajratib olish. Satrdan bir nechta elementdan tashkil topgan qismini ajratib olish mumkin. Buning uchun ajratilayotgan qismining boshlang‘ich elementining satrdagi o‘rni va oxirgi elementining satrdagi o‘rni olinadi. Ularni bildirgan sonlar orasiga: belgisi qo‘yilgan holda kvadrat qavs ichiga yoziladi. Kesmani ajratib olish operatori:[**X:Y**].

X - kesmaning boshi, **Y** esa - oxiri. Y raqamli belgi kesmaga kirmaydi. Jimlik holatida birinchi indeks 0 ga teng, ikkinchi indeks esa qator uzunligiga teng bo‘ladi.

```
>>> s='dasturlash'
>>> s [3:5]
'tu'
>>> s [2:-2]
'tula'
>>> s [:6]
'dastur'
>>> s [1:]
'asturlash'
>>> s [:]
'dasturlash'
```

Bundan tashqari kesmani ajratib olishda qadamni belgilash mumkin:

```
>>> s [ : : -1 ]
'hsalrutsad'
>>> s [2 : : 2]
'suls'
```

ord va *len* funksiyalar

Agar satr Unicode belgilaridan tashkil topgan bo‘lsa, u holda ***ord()*** funksiyasi yordamida belgining Unicode kodlashdagi sonli qiymatini olish mumkin:

```
print(ord("A"))    # 65
```

Satr uzunligi (satrdagi belgilar soni) ni olish uchun ***len()*** funksiyasidan foydalanish mumkin:

```
string = "hello world"
length = len(string)
print(length)   # 11
```

Satrlarni ajratish

for sikl operatori yordamida satrning barcha elementlarini ajratib olish mumkin:

```
string = "hello world"
for char in string:
    print(char)
```

Satrdan qidirish

Pythonda satrdan satr ostini qidirish uchun ***find()*** metodni foydalaniladi, bu metod satr ostining satrdagi birinchi belgisining indeksni qaytaradi va uchta shaklga ega:

find(str) – satr ositini satrdan izlash boshidan oxirigacha o‘tkaziladi;

find(str, start) – izlash *start* indeksdan boshlanadi;

find(str, start, end) – izlash *start* indeksdan *end* indeksgacha bo‘ladi.

Satr osti topilmasa -1 qiymatini qaytaradi.

```
welcome = "Hello world! Goodbye world!"
index = welcome.find("wor")
print(index)  # 6
# поиск с 10-го индекса index = welcome.find("wor", 10)
print(index)  # 21
 # поиск с 10 по 15 индекс index = welcome.find("wor", 10, 15)
print(index)  # -1
```

Satrlarni almashtirish

Satrlardagi biror satr ostini boshqa satr ostiga almashtirish uchun ***replace()*** metodidan foydalaniladi:

replace(old, new) – ***old*** satr ostini ***new*** satr ostiga almashtiradi;

replace(old, new, num) – ***num*** parametri dastlabki nechta satr ostini yagisi bilan almashtirish lozimligini ifodalaydi. Qolganlari o‘zgarishsiz qoladi.

```
phone = "+1-234-567-89-10"
 # defislarni bo‘sh joylarga almashtirish
edited_phone = phone.replace("-", " ")
print(edited_phone)  # +1 234 567 89 10
# defislarni o‘chirish
edited_phone = phone.replace("-", "")
print(edited_phone)  # +12345678910
# bitta defisni o‘chirish
edited_phone = phone.replace("-", "", 1)
print(edited_phone)  # +1234-567-89-10
```

Satr ostilariga ajratish

spilt() metodi satrlarni ajratuvchiga bo‘liq ravishda satr ostilariga ajratish uchun qo‘llaniladi. Ajratuvchilar sifatida ixtiyoriy satr yoki belgilarni berish mumkin. Ushbu metod quyidagi ko‘rinishga ega:

split() – ajratuvchi sifatida bo‘sh joy (probel) qo‘llaniladi;

split(delimeter) – ajratuvchi sifatida *delimeter* ishlatiladi;

split(delimeter, num) – ajratuvchi sifatida *delimeter* ishlatiladi, *num* esa nechta bo‘lakka bo‘lish kerakligini ko‘rsatadi. Agar satrni *delimeter* orqali bo‘laklarga ajratganda, bo‘laklar soni *num* da keltirilgan sondan katta bo‘lsa, u xolda dastlabki *num* ta bo‘lak ajratiladi, qolgan qismi ajratilmaydi.

```
text = "Ali, Vali, G‘ani - Sobir, Said qani?"
 txt = text.split()
print("Parametrsiz ishlatish, bu erda ajratuvchi probel bo‘ladi:")
print(txt)
print("3-chi bo‘lak:") print(txt[2])
print("Ajratuvchi sifatida vergul ishlatilgan:")
txt = text.split(",") print(txt)
print("2-chi bo‘lak:") print(txt[1])
print("Ajaratuvchi probel va bo‘laklar soni 3 ta:")
txt = text.split(" ", 3) print(txt)
print("4-chi bo‘lak:") print(txt[3])
```

Satrlarni birlashtirish

Satrlarni oddiy “Qo‘shish” amali bilan ham birlashtirish mumkin ekanligini ko‘rib chiqdik. ***join()*** metodi yordamida ham satrlarni birlashtirish mumkin, bu metod

satrlardan iborat ro‘yxatni bitta satrga aylantiradi. Buning uchun berilgan satrni ajratuvchi sifatida qarab satrdan *join()* metodiga murojaat qilinadi:

```
words = ["Let", "me", "speak", "from", "my", "heart", "in",
"English"]
# ajratuvchi – bo‘sh joy
sentence = " ".join(words)
print(sentence)
# Let me speak from my heart in English
# ajratuvchi – vertical chiziq sentence = " | ".join(words)
print(sentence)  # Let | me | speak | from | my | heart |
#in | English
```

join() metodiga ro‘yxat o‘rniga oddiy satrni ham berish mumkin, bu holda ajratuvchi satrdagi belgilar o‘rniga tushadi:

```
word = "hello"
joined_word = "|".join(word)
print(joined_word)      # h|e|l|l|o
```

Satrga oid funksiyalar

Pythonda satr bilan ayrim amallarni bajarish uchun maxsus funksiyalar bor. Ularning soni ancha ko‘p, hammasini eslab qolish esa qiyin. Shuning uchun kerakli funksiyani manbadan tanlab foydalangan ma’qul. Hozir esa shulardan ba’zilarini ko‘rib chiqamiz.

Metodlarni chaqirganga Pythondagi satrlar o‘zgarmaydigan ketma-ketliklar darajasiga kirishini inobatga olishimiz kerak. Bu degani hamma funksiyalar va metodlar faqat yangi satrni tuzishi mumkin.

```
>>> s='spam'
>>> s[1]='b'
Traceback (most recent call last):
  File "<pyshell#27>", line 1, in <module>
    s[1]='b'
TypeError: 'str' object does not support item assignment
>>> s=s[0]+'b'+s[2:]
>>> s
'sbam'
```

Shuning uchun hamma metodlar yangi satrni qaytaradilar, va u keyin boshqa nomga ega bo‘ladi.

S = ‘str’; S = “str”; S = ‘’‘str’‘’; S = “”“str”“”- satrlarni literallari

S = “s\np\ta\nbbb” - ekran bilan ishlash ketma-ketliklari

S = r”C:\temp\new” - formatlashtirilmagan satrlar

S = b”byte” - baytlar qatori

S1+S2 - konkatenatsiya (qo‘shish)

S1*3 - satrni takrorlash

S[i] - indeks bo‘yicha murojaat

S[i:j:step] - step qadamli i elementdan boshlab j elementgacha bo‘lgan kesmani ajratib olish.

S.rfind(str,[start],[end]) - satrdan satr ostini qidirish. Oxirgi kirish raqamini yoki 1 ni qaytaradi

S.index(str,[start],[end]) - satrdan satr ostini qidirish. Birinchi kirish raqamini qaytaradi yoki ValueError istisnosini chaqiradi

S.rindex(str,[start],[end]) - satrdan satr ostini qidirish. Oxirgi kirish raqamini qaytaradi yoki ValueError istisnosini chaqiradi.

S.isdigit() - satrda raqamlar ishtirok etganligini tekshirish.

S.isalpha() - satr faqat harflardan iboratligini tekshirish.

S.isalnum() - satr harf yoki raqamlardan iboratligini tekshiradi.

S.islower() - satr quyi registrdagi belgilardan iboratligini tekshiradi**.**

S.isspace() - satrda ko‘rinmaydigan belgilar borligini tekshirish (probel, sahifani o‘tkazish belgisi(‘\p’), yangi satrga o‘tish(‘\n’), koretkani qaytarish(‘\r’), gorizontal tabulyatsiya(‘\t’) va vertikal tabulyatsiya)

S.istitle() - satrda so‘zlar bosh harf bilan boshlanishini tekshirish.

S.starswith(str) - s satr str shablonidan boshlanishini tekshirish.

S.endswith(str) - s satr str shabloni bilan tugashini tekshirish.

S.join(ro‘yxat) - s ajratuvchiga ega ro‘yxatdan qatorni yig‘ish.

Ord(belgi) - belgiga mos ASCII kodni qaytaradi.

Chr(son) - ASCII kodga mos belgini qaytaradi.

S.capitalize() - satrning birinchi belgisi yuqori registrda qolganlarini quyi registrga o‘tkazadi.

S.center(width,[fill]) - chegaralari bo‘yicha fill (jimlik holatida probel) belgisi turuvchi markazlashtirilgan satrni qaytaradi.

S.expandtabs(tabsize) - joriy ustungacha bir yoki bir qancha probellar bilan tabulyatsiyaning hamma belgilari almashtirilgan satr nusxasini qaytaradi. Agarda TabSize ko‘rsatilmagan bo‘lsa tabulyatsiya hajmi 8 probelga teng bo‘ladi.

S.lstrip([chars]) - satr boshidagi probel belgilarini olib tashlash.

S.rstrip([chars]) - satr oxiridan probel belgilarini olib tashlash.

S.strip([chars]) - satr boshidan va oxiridan probel belgilarini olib tashlash.

S.partition(shablon) - birinchi shablon oldida turuvchi qismni keyin shablonni o‘zini va shablondan keyin turuvchi qismga ega kortejni qaytaradi. Agarda shablon topilmasa satrga ega bo‘lgan kortejni qaytaradi, avval ikki bo‘sh satr keyin satrni o‘zini.

S.rpartition(sep) - oxirgi shablon oldida turuvchi qismni keyin shablonni o‘zini va shablondan keyin turuvchi qismni qaytaradi. Kortej qator o‘zidan va undan keyin ikkita bo‘sh qatordan iborat bo‘ladi.

S.swapcase() - quyi registrdagi belgilarni yuqori registrga, yuqorilarni esa quyiga o‘tkazadi.

S.title() - har bitta so‘zning birinchi harfini yuqori registrga qolganlarini esa quyi registrga o‘tkazadi.

S.zfill(width) - qator uzunligini Widthdan kam qilmaydi agar kerak bo‘lsa birinchi belgilarni nollar bilan to‘ldiradi.

S.strip() - funksiyasi satrning boshi va oxirida bo‘shliqlar bo‘lsa, olib tasdiqlaydi:

S.lower() - funksiyasi satrdagi barcha so‘zlarni kichik harf bilan yozilishini ta’minlaydi:

S.upper() - funksiyasi satrdagi barcha so‘zlarni katta harflar bilan yozadi:

S.replace() - funksiyasi satrdagi bir so‘z yoki harfni boshqasi bilan almashtiradi.

Satrlarni tekshirish

Aniq bir jumla yoki harf(belgi) satrda bor yoki yo‘qligini **in** yoki **not** kalit so‘zlari bilan tekshirish mumkin. Bunday holatda qidirilaytgan jumla bor bo‘lsa **True** (rost),

aks holda **False** (yolg‘on) qiymati qaytariladi. Quyidagi kodimizda “ol” jumlasi borligini tekshirib ko‘ramiz:

```
txt = "Olmaxon yerdagi olmani olib ketdi" x = "ol" in txt
print(x)
```

Endi satrda “ol” jumlasi yo‘qligini tekshiramiz. Bu yerda “ol” jumlasi satrda borligi uchun **False** (yolg‘on) qiymati qaytariladi:

```
txt = "Olmaxon yerdagi olmani olib ketdi" x = "ol" not in txt
print(x)
```

Satrlar formati

Biz satr va sonli o‘zgaruvchilarni birgalikda to‘g‘ridan to‘g‘ri ishlata olmaymiz. Satr ichida sonli o‘zgaruvchini qo‘llash uchun **format()** funksiyasidan foydalaniladi. Bu funksiya sonli qiymatni olib satrli o‘zgaruvchiga aylantiradi va **{}** belgisi qo‘yilgan joy o‘rnida o‘sha qiymatni joylashtiradi.

```
yosh = 21
matn = "Mening yoshim {} da"
print(matn.format(yosh))
```

1-masala. N ta so‘zdan tashkil topgan matnda berilgan so‘z necha marta uchrashini aniqlang.

```
matn=str(input('Matnni kiriting:'));
belgi=str(input('Izlanayotganbelginikiriting:'));
m=len(matn);
n=len(belgi);
b=0;
for i in range((m-n)+1):
    if matn[i:i+1]==belgi:
        b=b+1;
if b==0:
    print("Matnda izlanayotgan belgi yo‘q");
else:
    print("Matnda izlanayotgan belgi",b,"ta");
```

```
Python 3.12.1 (tags/v3.12.1:2305ca5,
AMD64)] on win32
Type "help", "copyright", "credits"

= RESTART: D:\Madadjon\O`ktamov.M.py
Matnni kiriting:dasturlash
Izlanayotgan belgini kiriting:a
Matnda izlanayotgan belgi 2 ta
```

2-masala. ASC II jadvalidan kichik lotin harflarni chiqaring.

```
n=int(input('Raqam kiriting:'));
m=int(input('Raqam kiriting:'));
for i in range(n,m+1):
print(chr(i),end='');
```

```
Python 3.12.1 (tags/v3.12.1:2305ca5,
AMD64)] on win32
Type "help", "copyright", "credits"

= RESTART: D:\Madadjon\O`ktamov.M.py
Raqam kiriting:97
Raqam kiriting:122
abcdefghijklmnopqrstuvwxyz
```

3-masala. Berilgan so‘z ikkiyoqlama bo‘lishini aniqlang.

```
import math;
matn=str(input("Matn kiriting:"));
s=len(matn);t=0;
 for i in range(1,math.ceil(s/2)):
   if matn[:i]==matn[-i:]:
        t=t+1;
print("So‘z ikkiyoqlama");
elif t==s%2:
 print("So‘z ikkiyoqlama emas");
```

```
Python 3.12.1 (tags/v3.12.1:2305ca5,
AMD64)] on win32
Type "help", "copyright", "credits" (

= RESTART: D:\Madadjon\O`ktamov.M.py
Matn kiriting:ikki
So‘z ikkiyoqlama
```

4-masala. Berilgan matndagi hamma I harflarni olib tashlang.

```
matn=str(input("Matn kiriting:"));
olish='i';
s=matn.replace(olish,");
print("Matnda i harfi olib tashlangan:",s);
```

```
Python 3.12.1 (tags/v3.12.1:2305ca5, Dec  7 2023, 2
AMD64)] on win32
Type "help", "copyright", "credits" or "license()"

= RESTART: D:\Madadjon\O`ktamov.M.py
Matn kiriting:dasturlash tillari 2
Matnda i harfi olib tashlangan: dasturlash tllar 2
```

5-masala. Berilgan matnni teskarisi tartibda yozing.

```
matn=str(input('Matn kiriting:'));
s=matn[::-1];
print("Teskari matn:");
print(s);
```

```
Python 3.12.1 (tags/v3.12.1:2305ca5,
AMD64)] on win32
Type "help", "copyright", "credits" (

= RESTART: D:\Madadjon\O`ktamov.M.py
Matn kiriting:Dasturlash tillari 2
Teskari matn:
2 irallit hsalrutsaD
```

6-masala. Berilgan matndagi P harflarni J harflarga almashtiring.

```
matn=str(input("Matn kiriting:"));
s=matn.replace("P","J");
print("Matnda P harfi olib tashlab J
harflariga almashtirish:");
   print(s);
```

```
Python 3.12.1 (tags/v3.12.1:2305ca5, Dec  7 2023, 22:0
AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for

= RESTART: D:\Madadjon\O`ktamov.M.py
Matn kiriting:Python dasturlash tillari 2
Matnda P harfi olib tashlab J harflariga almashtirish:
Jython dasturlash tillari 2
```

7-masala. Matnda uzunligi K ta belgidan katta bo‘lgan so‘zlarni ajratilib yangi matnga yozish dasturini tuzing**.**

```
matn=str(input('Matn kiriting:'));
k=int(input('K ta belgi:'));
s=matn[0:k];
print('Ajratilgan katta harfdagi
matn',s.upper());
```

```
Python 3.12.1 (tags/v3.12.1:2305ca5, Dec
AMD64)] on win32
Type "help", "copyright", "credits" or "l

= RESTART: D:\Madadjon\O`ktamov.M.py
Matn kiriting:Dasturlash tillari 2
K ta belgi:9
Ajratilgan katta harfdagi matn DASTURLAS
```

8-masala. Inglizcha - o‘zbekcha lug‘atni tuzing. Bunda inglizcha so‘z kiritilganda uning tarjimasi natija sifatida olinishini ta’minlang.

```
    matn=str(input("So‘z kiriting:"));
    if matn=='REM':
    print("Izoh");
    if matn=='IF':     print('Agar');
    if matn=='FOR':  print('Uchun');
    if matn=='INPUT': print('Kiritish');
    if matn=='STOP':  print("To‘xta");
if matn=='PRINT':print("Chop etish");
    if matn=='RUN':   print("Bajar");
```

```
Python 3.12.1 (tags/v3.12.1:2305ca5,
AMD64)] on win32
Type "help", "copyright", "credits" c

= RESTART: D:\Madadjon\O`ktamov.M.py
So'z kiriting:RUN
Bajar
```

9-masala. Matndagi so‘zlarda nechta unli harflar borligini aniqlovchi dastur tuzing.

```
   matn=str(input('Matn kiriting:'));
   unli='iouha'; s=0;
   for i in range(len(matn)):
      for j in range(len(unli)):
         if matn[i:i+1]==unli[j:j+1]:
            s=s+1;
   print("Matnda izlanayotgan unli
so‘z",s,'ta bor');
```

```
Python 3.12.1 (tags/v3.12.1:2305ca5, De
AMD64)] on win32
Type "help", "copyright", "credits" or

= RESTART: D:\Madadjon\O`ktamov.M.py
Matn kiriting:dasturlash
Matnda izlanayotgan unli so'z 4 ta bor
```

10-masala. Satr berilgan. Satrning oxirgi belgisini birinchi belgi qiluvchi dastur tuzing.

```
        satr = input('satr: ')
        satr = list(satr) a = []
        a.append(satr.pop(-1))
        a.extend(satr)
        satr = '' for i in a:
        satr += i
        print(satr)
```

```
Python 3.12.1 (tags/v3.12.1:2305ca5,
AMD64)] on win32
Type "help", "copyright", "credits"

= RESTART: D:\Madadjon\O`ktamov.M.py
satr: Dasturlash
hDasturlas
```

11-masala. Satr berilgan. Ushbu satrdagi kichik harflarni katta harflarga o‘giruvchi dastur tuzing.

```
satr = input('satr: ')
l = list(satr)
for j in range(len(l)):
if 65 <= ord(l[j]) and ord(l[j]) <= 90:
 l[j] = chr(ord(l[j]) + 32)
elif 97 <= ord(l[j]) and ord(l[j]) <= 122:
 l[j] = chr(ord(l[j])-32)
satr ='' for i in l:
satr += i
print(satr)
```

```
Python 3.12.1 (tags/v3.12.1:2305ca5,
AMD64)] on win32
Type "help", "copyright", "credits"

= RESTART: D:\Madadjon\O`ktamov.M.py
satr: DaStuRlaSh
dAsTUrLAsH
```

12-masala. Berilgan matnning orasidagi K-simvoldan N-simvolgacha bo‘lgan belgilarni ajrating**.**

```
matn=str(input('Matn kiriting:'));
BelgiK=int(input('K ta belgi:'));
BelgiN=int(input('N ta belgi:'));
s=matn[BelgiK:BelgiN-BelgiK];
print("Ajratilgan matn",s);
```

```
Python 3.12.1 (tags/v3.12.1:2305ca5,
AMD64)] on win32
Type "help", "copyright", "credits"

= RESTART: D:\Madadjon\O`ktamov.M.py
Matn kiriting:Dasturlash tillari 2
K ta belgi:0
N ta belgi:10
Ajratilgan matn Dasturlash
```

13-masala. Topishmoq topish va uning javobini tahlil qilish dasturini tuzing**.**

```
print("Yer tagida oltin qoziq?");
matn=str(input("Javobni kiriting:"));
if matn=='sabzi':
print("Siz to‘g‘ri javob berdingiz");
else:
print("Siz noto‘g‘ri javob berdingiz");
```

```
Python 3.12.1 (tags/v3.12.1:2305ca5, Dec  7 2023
AMD64)] on win32
Type "help", "copyright", "credits" or "license(

= RESTART: D:\Madadjon\O`ktamov.M.py
Yelkasi bor, boshi yo`q, oyog`i bor qo`li yo`q?
Javobni kiriting:stul
Siz to'g'ri javob berdingiz
```

14-masala. Matndagi INFORMATIKA so‘zini ALGORITM so‘zi bilan almashtirish dasturini tuzing.

```
matn1=str(input("Matn kiriting:"));
matn2=str(input("Almashtiriladigan matn kiriting:"));
s=matn1.replace(matn1,matn2);
print("Almashtirilgan matn:",s);
```

```
Python 3.12.1 (tags/v3.12.1:2305ca5, De
AMD64)] on win32
Type "help", "copyright", "credits" or

= RESTART: D:\Madadjon\O`ktamov.M.py
Matn kiriting:Dasturlash
Almashtiriladigan matn kiriting:Python
Almashtirilgan matn: Python
```

15-masala. Tushirib qoldirilgan harf o‘rniga H harfni yozishni o‘rgatuvchi dastur tuzing.

```
matn=str(input("Matn kiriting:"));
b=str(input('Harf kiriting:'));
 t=matn.replace(b,"");
print(t);
```

```
Python 3.12.1 (tags/v3.12.1:2305ca5, Dec
AMD64)] on win32
Type "help", "copyright", "credits" or "l

= RESTART: D:\Madadjon\O`ktamov.M.py
Matn kiriting:DasturlashHHH tillariHH 2H
Harf kiriting:H
Dasturlash tillari 2
```

15-MAVZU. PYTHON DASTURLASH TILIDA FAYLLAR VA ULARDAN FOYDALANISH

Reja:

1. Fayllarni faollashtirish;
2. Fayllar ustida amallar bajarish.
4. Fayldan ma’lumot o‘qish
5. Fayl tarkibiga ma’lumotlarni yozish
6. Fayl tarkibidagi ma’lumotlarni o‘chirish

Tayanch so‘zlar: fayl, open, read, write, del, mantiqiy nom, fizik nom.

Python dasturlash tili tarkibida fayllar bilan ishlash uchun barcha turdagi imkoniyatlar e’tiborga olingan. Dasturlash tillarida fayllar dastur tarkibidagi o‘zgaruvchilar qabul qiladigan qiymatlarni saqlash uchun ishlatiladi. Dasturlash asoslari fani rivojlanib borishi bilan dasturlash tillari tarkibida fayllar bilan ishlash ham rivojlanib bormoqda. Dasturlash asoslari tarkibida murakkab masalalarni hal etishda o‘zgaruvchilarni qiymatlari soni ko‘p bo‘lsa, bunday holatlarda qo‘lda kiritish qulay hisoblanmaydi. O‘zgaruvchilarning qiymatlari soni ko‘p bo‘lgan holatlarda qiymatlar fayllarda saqlansa dasturning o‘zi qiymatlarni tezkor holda fayllardan qabul qilib oladi. Kompyuterning ma’lum bir joyida qiymatlar fayllar asosida saqlanadi.

Ta’rif: Kompyuterda alohida nom bilan saqlanadigan ma’lum bir turga mansub bo‘lgan ma’lumotlar majmuasi fayl deb nomlanadi.

Python dasturlash tili tarkibida fayllarga murojaat qilish uchun fayl nomiga to‘g‘ridan to‘g‘ri murojaat qilib bo‘lmaydi, fayllarga murojaat qilish uchun fayllarni dastur bilan bog‘lash uchun alohida o‘zgaruvchi qabul qilinadi.

Fayllar ustida amallar bajarish

Python dasturlash tili tarkibida fayllar ustida amallar bajarish uchun yuqoridagi holatlar bo‘yicha oldin faollashtirish va mantiqiy nomi bilan bog‘lashi shart. Fayllar ustida quyidagi amallar mavjud.

1) Fayllardan ma’lumot o‘qish
2) Fayllarga ma’lumot yozish
3) Fayllardan ma’lumot o‘chirish

Open() - pythonda fayllar bilan ishlashning asosiy funksiyasi funksiya hisoblanadi

Open() - funksiya ikkita parametrni oladi; *fayl nomi* va *rejimi*.

Faylni ochishning to‘rt xil usuli (rejimi) mavjud:

"r"- O‘qish - standart qiymat. Faylni o‘qish uchun ochadi, agar fayl mavjud bo‘lmasa, xato

"a"- Qo‘shish - faylni qo‘shish uchun ochadi, agar u mavjud bo‘lmasa, uni yaratadi

"w"- Write - faylni yozish uchun ochadi, agar u mavjud bo‘lmasa, uni yaratadi

"x"- Yaratish - belgilangan faylni yaratadi, agar fayl mavjud bo‘lsa, xatoni qaytaradi

Bundan tashqari, fayl ikkilik yoki matn rejimi sifatida ishlashi kerakligini belgilashingiz mumkin

"t"- Matn - standart qiymat. Matn rejimi

"b"- Ikkilik - ikkilik rejim (masalan, tasvirlar)

Sintaksis

O‘qish uchun faylni ochish uchun fayl nomini ko‘rsatish kifoya:

```
f = open("demofile.txt")
```

Yuqoridagi kod bir xil:

```
f = open("demofile.txt", "rt")
```

"r" - o‘qish va matn uchun standart qiymatlar bo‘lgani uchun "t" ularni ko‘rsatish shart emas.

Eslatma: Fayl mavjudligiga ishonch hosil qiling, aks holda siz xatoga yo‘l qo‘yasiz.

Bizda Python bilan bir papkada joylashgan quyidagi fayl bor deb faraz qilaylik:

demofile.txt
Hello! Welcome to demofile.txt This file is for testing purposes.
Good Luck!

Faylni ochish uchun o‘rnatilgan **open()** funksiyadan foydalaning.

Funktsiya fayl mazmunini o‘qish usuliga open() ega bo‘lgan fayl ob'ektini qaytaradi:

read()

```
f = open("demofile.txt", "r")
print(f.read())
```

```
Hello! Welcome to demofile.txt
This file is for testing purposes.
Good Luck!
```

Agar fayl boshqa joyda joylashgan bo‘lsa, fayl yo‘lini ko‘rsatishingiz kerak bo‘ladi, masalan:

```
f=open("D:\\myfiles\welcome.txt","r")
print(f.read())
```

```
Welcome to this text file!
This file is located in a folder named "myfiles", on the D drive.
Good Luck!
```

Faylning faqat qismlarini o‘qish

Odatiy bo‘lib, read() usul butun matnni qaytaradi, lekin siz qaytarmoqchi bo‘lgan belgilarni ham belgilashingiz mumkin:

Misol: Faylning birinchi 5 ta belgisini qaytaring:

```
f = open("demofile.txt", "r")
print(f.read(5))
```

```
Hello
```

Satrlarni o‘qish

Usul yordamida bitta qatorni qaytarishingiz mumkin readline():

Misol: Faylning bir qatorini o‘qing:

```
f = open("demofile.txt", "r")
print(f.readline())
```

```
Hello! Welcome to demofile.txt
```

Fayldan o‘qish

Fayldan o‘qish uchun *r* ***(Read)*** rejimida ochiladi va uning mazmunini turli usullar bilan o‘qish mumkin:

- ***readline()*** - fayldan bir qator o‘qiydi;
- ***read()*** - faylning butun tarkibini bir qatorga o‘qiydi;
- ***readlines()*** - faylning barcha satrlarini ro‘yxatga oladi.

Masalan, biz yuqorida yozilgan fayllarni satrlar bo‘yicha ko‘rib chiqamiz:

```
with open("salom.txt", "r") as fayl:
for satr in fayl:
    print(satr, end="")
```

Biz, albatta, har bir qatorni o‘qish uchun *readline()* metodini ishlatmasak ham, har bir yangi satrni olish uchun ushbu metod avtomatik ravishda chaqiriladi. Shuning

uchun ham, *readline()* metodini siklda chaqirishdan ma’no yo‘q va satrlar yangi satr "\n" belgisi bilan ajratilganligi uchun, yangi satrga chop qilish zaruryati qolmaydi va *end=""* qiymati *print* metodining ikkinchi parametri sifatida uzatiladi.

Endi satrlarni alohida o‘qish uchun *readline()* metodini to‘g‘iridan-to‘g‘ri chaqiramiz:

```
with open("salom.txt", "r") as fayl:
str1 = fayl.readline()    print(str1, end="")
str2 = fayl.readline()    print(str2)
```

Konsol ekraniga quyidagi natijalar chiqariladi:

salom olam, hayr olam *readline()* metodini alohida qatordagi satrlarni o‘qish uchun *while* siklida ham foydalanish mumkin:

```
with open("salom.txt", "r") as fayl:
   satr = fayl.readline()    while satr:
      print(satr, end="")       satr = fayl.readline()
```

Fayl kichik bo‘lsa, *read*() metodidan foydalanib, uni birdan o‘qishingiz mumkin:

```
with open("salom.txt", "r") as fayl:
   mazmun = fayl.read()    print(mazmun)
```

Hamda, *readlines()* metodi yordamida fayldagi barcha satrlar ro‘yxatga o‘qib olinadi, ya`ni elementlari fayldagi satrlardan tashkil topgan ro‘yxat hosil qilinadi:

```
with open("salom.txt", "r") as faly:
mazmun = fayl.readlines()    str1 = mazmun [0]    str2 = mazmun [1]
print(str1, end="")    print(str2)
```

Ba’zida fayldagi ma’lumotlar ASCIIdagi belgilardan farqlanishi mumkin. Ushbu holatda fayldan berilganlarni o‘qish to‘g‘ri bo‘lishi uchun kodlash parametrini ishlatib kodlashni aniq belgilab olishimiz mumkin:

```
faylnomi = "salom.txt" with open(faylnomi, encoding="utf8") as file:
      matn = file.read()
```

Faylni o‘chirish

Faylni o‘chirish uchun siz OS modulini import qilishingiz va uning **os.remove()** funksiyasini ishga tushirishingiz kerak:

Misol: "demofile.txt" faylini olib tashlang:

```
import os
os.remove("demofile.txt")
```

Fayl mavjudligini tekshiring:

Xatolikka yo‘l qo‘ymaslik uchun faylni o‘chirishdan oldin uning mavjudligini tekshirishingiz mumkin:

Misol: Fayl mavjudligini tekshiring, *keyin* uni o‘chiring:

```
    import os
if os.path.exists("demofile.txt"):
 os.remove("demofile.txt")
else:
  print("The file does not exist")
```

Jildni o‘chirish

Butun jildni o‘chirish uchun quyidagi **os.rmdir()** usuldan foydalaning:

Misol: "Mening papkam" jildini olib tashlang:

```
    import os
os.rmdir("myfolder")
```

Eslatma: Siz faqat bo‘sh papkalarni olib tashlashingiz mumkin.

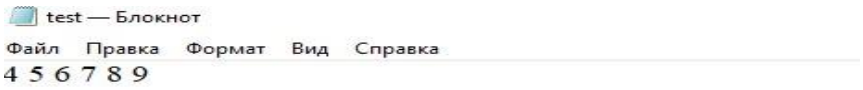
Quyidagi dastur orqali foydalanuvchi tomonidan kiritilgan satrlar massivi dastlab faylga yozish amalga oshirilgan, so‘ngra ularni fayldan konsolga qayta o‘qib, chop qilish amalga oshirilgan:

```
    # fayl nomi
    FILENAME = "habarlar.txt"
    # bo‘sh ro‘yxat aniqlaymiz
     xabarlar = list()
     for i in range(4):
        xabar = input("Satrni kiriting " + str(i + 1) + ": ")
    xabarlar.append(xabar + "\n")
     # ro‘yxatni faylga yozish
    with open(FILENAME, "a") as fayl:
    for xabar in xabarlars:        fayl.write(xabar)
     # xabarlarni fayldan o‘qiymiz
    print("Xabarlarni o‘qish") with open(FILENAME, "r") as fayl:
    for xabar in fayl:
          print(xabar, end="")
```

Dastur ishlashining namunasi:

```
    Satrni kiriting 1: salom
    Satrni kiriting 2: tinchlik so‘zi
    Satrni kiriting 3: buyuk ish
    Satrni kiriting 4: Python
    Xabarlarni o‘qish Salom tinchlik so‘zi buyuk ish
    Python
```

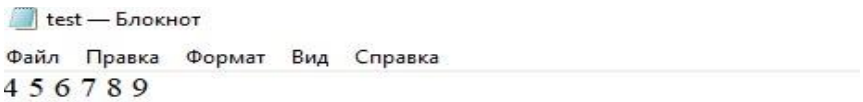
Misol. *f(test.txt)* faylning barcha elementlarini ekranga chiqarish dasturini tuzing.
test.txt fayl ko‘rinishi



f=open('test.txt','r') a=f.read() for i in a: print(i.strip())	4 5 6 7 8 9

Misol. *f(test.txt)* faylning barcha elementlarini teskari tartibda ekranga chiqarish dasturini tuzing.

test.txt fayl ko‘rinishi



f=open('test.txt','r') a=f.read() for i in reversed(a): print(i.strip())	9 8 7 6 5 4

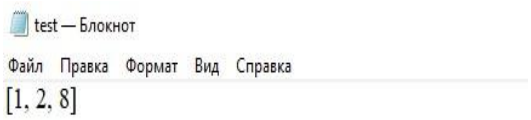
Fayl tarkibiga ma’lumotlarni yozish

Python dasturlash tilida ma’lumotlarni ikki turda yozishni aytgan edik. Python dasturlash tilida birinchi tur bo‘yicha faylga ma’lumot yozish uchun <fayl mantiqiy nomi>=open('fayl fizik nomi','w'), ikkinchi tur bo‘yicha faylga ma’lumot yozish uchun <fayl mantiqiy nomi>=open('fayl fizik nomi','a') buyruqlari oldin yozilishi shart undan so‘ng uning tarkibiga ma’lumot yozish mumkin. Python dasturlash tilida faylga ma’lumot yozishning umumiy ko‘rinishi quyidagicha bo‘ladi:

<faylni mantiqiy nomi> .write(o‘zgaruvchi)

Fayl tarkibiga ma’lumot yozilganda uning oldingi ma’lumotlari o‘chirilib, yangi ma’lumotlar yoziladi yoki fayl oxiridan yoziladi.

Misol: a ro‘yxat berilgan ro‘yxat elementlarini f(test.txt) faylga yozish dasturini tuzing.

f=open(‘test.txt’, ‘w’) a=[1,2,8] f.write(str(a)) f.close()	

Dastur natijasiga e’tobor bersak, faylga ma’lumot faqat satr ko‘rinishida yozilganligi sababli.ro‘yxat elementlari to‘g‘ridan to‘g‘ri faylga yozilmoqda.

Ikkinchi tur bo‘yicha ma’lumot yozish degani fayl oxiriga ma’lumot yozish tushuniladi. Faylning oxiriga ma’lumot yozish uchun <fayl mantiqiy nomi>=open('fayl fizik nomi','a') buyrug‘i keltiriladi.

Fayl tarkibidagi ma’lumotlarni o‘chirish

Fayl tarkibidagi ma’lumotlarni o‘chirish uchun Python dasturlash tili tarkibida imkoniyat mavjud. Ma’lumot yozilgan faylni tozalash yoki uning tarkibidagi elementlarni o‘chirish uchun quyidagi kodni yozish kerak.

```
f=open('test.txt', 'w')
```

Yuqoridagi dastur asosida test.txt fayli tozalandi.

Python dasturlash tili tarkibida fayllardan foydalanish jarayonlari yuqoridagi holatlar bo‘yicha amalga oshiriladi. Fayldan ma’lumot o‘qish, faylga ma’lumot yozish va fayldan ma’lumot o‘chirish jarayonlari asosida fayllarga tegishli masalalarni hal etish imkoniyati mavjud.

CSV fayllari bilan ishlash

Ma'lumotni qulay shaklda saqlashning keng tarqalgan fayl formatlaridan biri csv formatidir. CSV faylidagi har bir satr vergul bilan ajratilgan alohida ustunlardan iborat bo‘lgan yozuv yoki satrni aks ettiradi. Aslida, bu format "Vergul bilan ajratilgan qiymatlar (Comma Separated Values)" deb nomlanadi. CSV formati matnli fayl formati bo‘lsa-da, Python u bilan ishlashni soddalashtirish uchun maxsus ajralmas CSV modulini taqdim etadi. Quyidagi misolda modulning ishini ko‘rib chiqamiz:

```
import csv
FILENAME = "users.csv"
 users = [
   ["Ali", 25],
   ["Sobir", 32],
   ["Dilnoza", 14]
] with open(FILENAME, "w", newline="") as fayl:
   writer = csv.writer(fayl)     writer.writerows(users)
 with open(FILENAME, "a", newline="") as fayl:
   user = ["Shaxnoza", 18]
writer = csv.writer(fayl)
writer.writerow(user)
```

Faylga ikki o‘lchovli ro‘yxat yoziladi – har bir satr bitta foydalanuvchini ifodalaydigan jadval. Har bir foydalanuvchi esa ikkita maydon - ism va yoshni o‘z ichiga oladi. Ya'ni, uchta satr va ikki ustunli jadvalni ifodalaydi.

Yozish uchun fayl ochilganda, uchinchi parametr sifatida *newline=""* qiymati ko‘rsatildi - bo‘sh satr operatsion tizimidan qat'i nazar, fayllardan to‘g‘ri satrlarni o‘qishga imkon beradi.

Yozish uchun *csv.writer(file)* funksiyasi tomonidan qaytariladigan *writer* obyektini olishimiz kerak. Ushbu funktsiyaga ochiq fayl topshiriladi. Hamda, mos ravishda yozish *writer.writerows(users)* metodi yordamida amalga oshiriladi. Bu usul qatorlar to‘plamini parametr sifatida oladi. Bizning holatimizda bu ikki o‘lchovli ro‘yxat hisoblanadi.

Agar bitta yozuv qo‘shish zarur bo‘lsa, ya’ni, bir o‘lchamli ro‘yxat, masalan, ["Shaxnoza", 18], bu holda *writer.writerow(user)* metodidan foydalaniladi. Natijada, skriptni ishga tushirgandan so‘ng, quyidagi tarkibga ega bo‘lgan *users.csv* fayli shu papkada paydo bo‘ladi:

```
Ali,25
Sobir,32
```

```
Dilnoza,14
Shaxnoza,18
```

Fayldan o‘qish uchun, aksincha, *reader* obyektini yaratishimiz kerak:

```
import csv
FILENAME = "users.csv"
 with open(FILENAME, "r", newline="") as fayl:
    reader = csv.reader(fayl)     for row in reader:
        print(row[0], " - ", row[1])
```

Reader obyektini olayotganda, biz uning barcha satrlarini ko‘chirib olishimiz mumkin:

```
Ali - 25
Sobir - 32
Dilnoza - 14
Shaxnoza - 18
```

16-MAVZU. PYTHON DASTURLASH TILI TARKIBIDA GRAFIKLAR CHIZISH VA ULARNI QAYTA ISHLASH

Reja:

1. Python dasturlash tilida grafik muhitini faollashtirish;
2. Tekislikda chizma va shakllar chizish;
3. Grafikga ma’lumot yozish;
4. Python dasturlash tilida matematik funksiyalar grafiklarini chizish.

Tayanch so‘zlar: turtle, forward(), backward(), right(), left(), penup(), pendown(), up(), down(), color(), fillcolor(), heading(), position(), goto(), begin_fill(), end_fill(), dot(), stamp(), shape().

Python dasturlash tilida ma’lum bir shakllar va chizmalarni hosil qilish uchun avval, albatta, grafik rejimni hosil qilish kerak, ya’ni grafik kutubxonani faollashtirish kerak. Python dasturlash tilida grafik rejim hosil qilingandan so‘ng uning tarkibiga kerakli chizma va shakllarni hosil qilish buyruqlarini yozish mumkin.

Grafik muhitini faollashtirish

Python dasturlash tili tarkibida boshqa dasturlash tillari kabi grafik rejimi va uning imkoniyatlari mavjud. Chizmalar va sohalarni hosil qilish uchun python dasturlash tilida **matplotlib** kutubxonasini chaqirish kerak.

Matplotlib kutubxonasini faollashtirishning umumiy ko‘rinishi quyidagicha.

from matplotlib.pyplot import*

Grafik rejimi hosil qilingandan so‘ng kompyuter ekranini koordinatalar sistemasini 1-choraki deb qarash kerak. Bunda kompyuter ekraniga chiziladigan shakl va chizmalar xuddi koordinatalar sistemasining 1-chorakida chiziladigandek buruqlar beriladi. Kompyuterning ekrani bir nechta nuqtalar matritsasidan tashkil topgan. Dasturchi tomonidan chizilgan shakl va chizmalar ekran rangi bilan bir xil rangda bo‘lsa, chizilgan shakl va chizmalar ko‘rinmasdan qoladi, shuning uchun chiziladigan shakl, chizma va nuqtalar uchun alohida ranglar ham berilish mumkin.

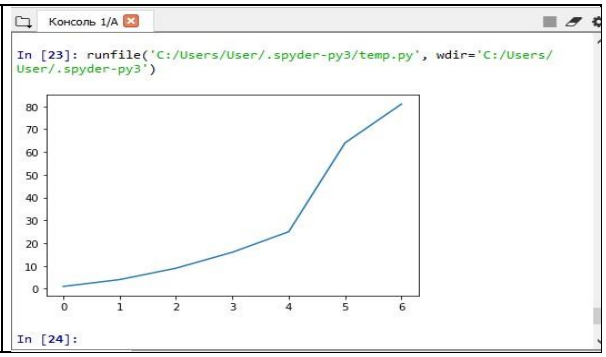
Tekislikda chizma va shakllar chizish

Python dasturlash tili tarkibida shakl va chizmalar nuqtalar ketma-ketligidan hosil bo‘ladi. Python dasturlash tilida nuqta, shakl va chizmalarni rangi va chizma turi

alohida beriladi. Python dasturlash tili tarkibida grafik shakllarni quyidagi funksiyalar orqali chiziladi: ***plot(y), show()***-funksiyasi *y* to‘plam yoki *y* ro‘yxat elementlarini ikki o‘lchovli koordinatalar sistemasida chizish uchun xizmat qiladi.

plot() funksiyasini ishlash jarayonini quyidagi dastur orqali qarab o‘tamiz.

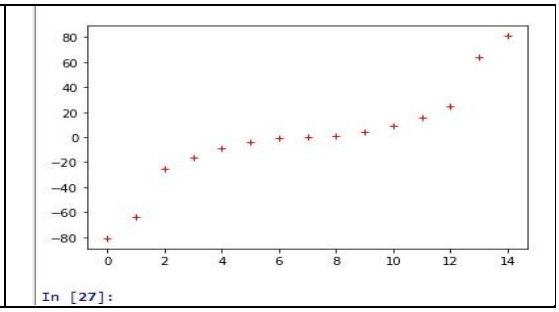
```
from numpy import*
from math import*
from matplotlib.pyplot import*
plot([1,4,9,16,25,64,81])
show()
```



Chiziladigan shakl va chizmalarning chiziq turlari va ranglarini o‘zgartirish ham mumkin.

Misol. A ro‘yxat elementlarini + belgi bilan qizil rangda chizish dasturini tuzing.

```
from numpy import*
from math import*
from matplotlib.pyplot import*
plot([-81,-64,-25,-16,-9,-4,-1,0,1,4,9,16,25,64,81]),'+', color='red');
show()
```



Chiziladigan shakl va chizmalarning chiziq turlari ‘+’, ‘_’, ‘*’, ‘-‘, ‘v’, ‘s’, ‘>’, ‘<’, ‘D’, ‘d’, ‘p’, ‘h’, ‘x’, ‘|’ kabi belgilar ko‘rinishida bo‘lishi mumkin.

Chiziladigan shakl va chizmalarning chiziq ranglari quyidagi jadval ko‘rinishida aniqlanadi.

Rang nomi(O‘zbek tilida)	**Rang nomi(ingiliz tilida)**	**Rang kodi**
Qora	Black	0
Ko‘k	Blue	1
Yashil	Green	2
Och ko‘k	Cyan	3
Qizil	Red	4
Binafsha	Magenta	5
Malla	Brown	6
Ko‘lrang	Lightgray	7
Och qora	Darkgray	8
Och ko‘k	Lightblue	9

Och yashil	Lightgreen	10
To‘q ko‘k	Lightcyan	11
Och qizil	Lightred	12
Och binafsha	Lightmagenta	13
Sariq	Yellow	14
Oq	White	15

Turtle

“Turtle” - bu chizma taxtasi kabi Python -ning o‘ziga xos xususiyati bo‘lib, u bizga turtleni hamma joyini chizishni buyuradi! Turtle.forward (...) va turtle.right (...) kabi funksiyalarni ishlatishimiz mumkin, ular Toshbaqa (Turtle)ni harakatga keltiradi.

Turtlening keng tarqalgan usullari:

Usul	Tavsif
Turtle()	Yangi tutrle ob'ektini yaratadi va qaytaradi
forward()	Toshbaqa (Turtle)ni belgilangan miqdorda oldinga siljitadi
backward()	Toshbaqa (Turtle)ni belgilangan miqdorda orqaga siljitadi
right()	Toshbaqa (Turtle)ni soat yo‘nalishi bo‘yicha aylantiradi
left()	Toshbaqa (Turtle)ni soat sohasi farqli o‘girib
penup()	Toshbaqa (Turtle) qalamini oladi
pendown()	Toshbaqa (Turtle) qalamini qo‘yadi
up()	Toshbaqa (Turtle) qalamini oladi
down()	Toshbaqa (Turtle) qalamini qo‘yadi
color()	Toshbaqa (Turtle) qalamining rangini o‘zgartiradi
fillcolor()	Turtlening rangini o‘zgartirish ko‘pburchakni to‘ldirishda ishlatiladi
heading()	Joriy sarlavhani qaytaradi
position()	Joriy pozitsiyani qaytaradi
goto()	Toshbaqa (Turtle)ni x, y holatiga o‘tkazing
begin_fill()	To‘ldirilgan ko‘pburchak uchun boshlang‘ich nuqtani eslang
end_fill()	Ko‘pburchakni yoping va joriy rang bilan to‘ldiring
dot()	Nuqtani hozirgi holatida qoldiring
stamp()	Hozirgi joyda Toshbaqa (Turtle) shakli haqida taassurot qoldiradi
shape()	"Ok", "klassik", "Toshbaqa (Turtle)" yoki "doira" bo‘lishi kerak

Turtle yordamida chizmachilik

Toshbaqa (Turtle) usullari va funksiyalaridan foydalanish uchun biz Toshbaqa (Turtle)ni import qilishimiz kerak. "Toshbaqa (Turtle)" standart Python to‘plami bilan

to‘ldirilgan va uni tashqaridan o‘rnatish shart emas. Turtle dasturini bajarish uchun yo‘l xaritasi 4 bosqichdan iborat:

1-Turtle modulini import qiling.

2-Boshqarish uchun Toshbaqa (Turtle) yarating.

3-Toshbaqa (Turtle) usullaridan foydalanib, atrofni chizib oling.

4-done () ni ishga tushiring.

Yuqorida aytib o‘tilganidek, Toshbaqa (Turtle)ni ishlatishdan oldin, biz uni import qilishimiz kerak. Biz uni quyidagicha import qilamiz:

```
from turtle import *
# or
import turtle
```

Turtlelar kutubxonasini import qilib, Toshbaqa (Turtle)ning barcha funksiyalarini bizga taqdim etgandan so‘ng, biz yangi chizma taxtasi (oyna) va Toshbaqa (Turtle)ni yaratishimiz kerak. Shunday qilib, biz kodni quyidagicha yozamiz:

```
wn = turtle.Screen()
wn.bgcolor("light green")
wn.title("Turtle")
shakl =Turtle()
```

Endi biz deraza va Toshbaqa (Turtle)ni yaratdik, Toshbaqa (Turtle)ni siljitishimiz kerak. Shakl qaragan tomonga 100 piksel oldinga siljish uchun biz kodlaymiz:

```
shakl.forward(100)
```

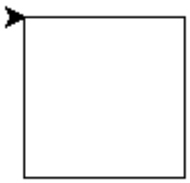
Biz shakl 100 piksel oldinga siljitdik, Ajoyib! Endi biz dasturni done () funksiyasi bilan yakunlaymiz.

```
turtle.done ()
```

Shunday qilib, biz 100 piksel uzunlikdagi chiziq chizadigan dastur yaratdik. Turtle usullari yordamida biz har xil shakllarni chizishimiz va turli ranglarni to‘ldirishimiz mumkin. Pythondagi Turtlelar kutubxonasi yordamida kodlash uchun ko‘plab funksiyalar va dasturlar mavjud. Keling, ba'zi asosiy shakllarni chizishni o‘rganamiz.

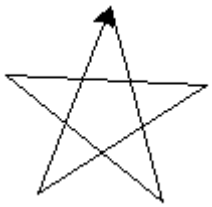
1-shakl: kvadrat

```
from turtle import*
shakl=Turtle()
for i in range(4):
    shakl.forward(80)
    shakl.right(90)
turtle.done()
```



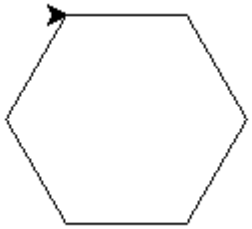
2-shakl: yulduzcha

```
from turtle import*
yulduz=Turtle()
yulduz.right(75)
yulduz.forward(100)
for i in range(4):
    yulduz.right(144)
    yulduz.forward(100)
turtle.done()
```



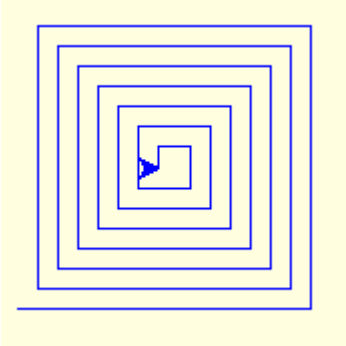
3-shakl: olti burchak

```
from turtle import*
oltiburchak=Turtle()
tomon=6
olti_b=60
burchak=360.0/tomon
for i in range(tomon):
    oltiburchak.forward(burchak)
    oltiburchak.right(olti_b)
turtle.done()
```



4-shakl. Ichma-ich spiral kvadrat

```
from turtle import*
wn=Screen()
wn.bgcolor("light yellow")
wn.title("Turtle")
skk=Turtle()
skk.color("blue")
def sqrfunc(size):
    for i in range(4):
        skk.fd(size)
        skk.left(90)
        size=size-5
sqrfunc(146)
```



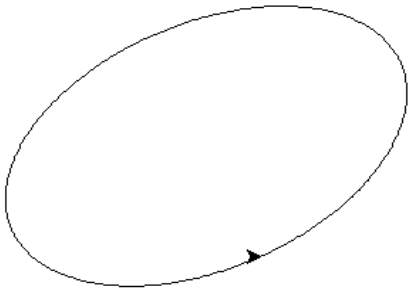
```
sqrfunc(126)
sqrfunc(106)
sqrfunc(86)
sqrfunc(66)
sqrfunc(46)
sqrfunc(26)
turtle.done()
```

5-shakl. Chiziq chizish.

```
from turtle import*
tr=Turtle()
tr.pensize(4)
tr.forward(200)
turtle.done()
```

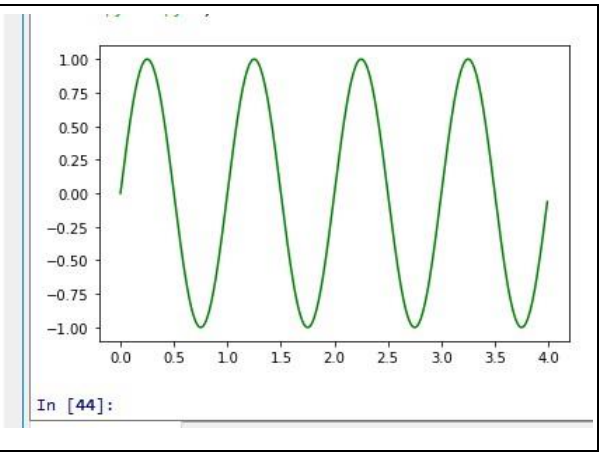
6-shakl. Ellips chizish.

```
from turtle import*
from math import *
a=150
b=90
for i in range(361):
    t=i*(pi/180)
    x=a*sin(t)
    y=b*cos(t)-b
    tilt=25*(pi/180)
    x1=x*cos(tilt)+y*sin(tilt)
    y1=x*sin(tilt)-y*cos(tilt)
    goto(x1,y1)
```



Misol. Ma’lum bir oraliqda sin(x) funksiyasi va uning argumentini koordinatalar sistemasida yashil rang bilan chizish dasturini yarating.

```
        from numpy import*
        from math import*
from matplotlib.pyplot import*
        t=[]
        x=[]
        for i range(400):
        t.append(i*0.01)
        x.append(sin(2*pi*t[i]))
        plot(t,x, color=’green’)
```



show()	

Chizmalarni alohida faylda saqlash

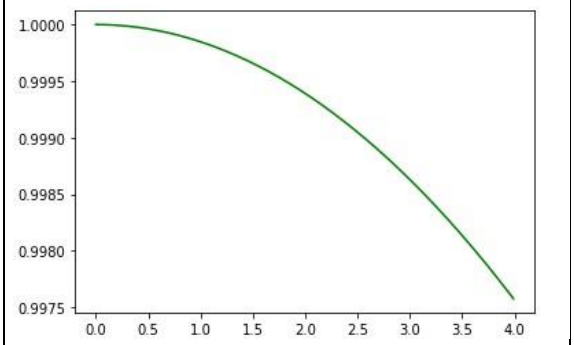
Python dasturlash tilida chiziladigan shakl va chizmalarni alohida .png kengaytmali fayllarga saqlash imkoniyati mavjud. Bu asosan katta turdagi ma’lumotlarni qayta ishlash vaqtida rasmlarni alohida fayl sifatida saqlash imkonini yaratadi. Chiziladigan shakl va chizmalarni alohida faylga quyidagi funksiya orqali amalga oshiramiz.

savefig('sincos.png‘)

Bu funksiya grafikni, dastur saqlangan papkaga saqlaydi, agar boshqa joyga saqlash kerak bo‘lsa albatta adres ‘’ belgi ichiga yozilish kerak. Yuqoridagi funksiyani ishlash jarayonini quyidagi dastur orqali qarab o‘tamiz.

Misol. Ma’lum bir oraliqda cos(x) funksiya grafigini chizing va bu grafikni cosinus.png fayliga saqlash dasturini tuzing.

```
from numpy import*
from math import*
from matplotlib.pyplot import*
t=[] x=[]
for i in range(400):
t.append(i*0.01)
x.append(cos(pi*t[i]/180))
plot(t, x, color='green')
savefig('cosinus.png‘) show()
```



Grafikga ma’lumot yozish

Python dasturlash tilida chiziladigan shakl va chizmalarga ma’lumot yozish mumkin. Bu ma’lumot funksiyaga nom, OX va OY o‘qi bo‘yicha ma’lumot yozish mumkin. Ma’lumotlarni quyidagi funksiyalar amalga oshiradi.

Funksiyaga nom berish funksiyasi:

title(‘text’)

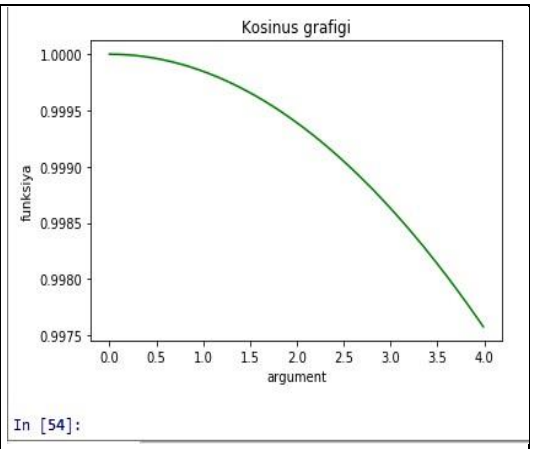
OY o‘qiga ma’lumot yozish: **ylabel(‘text’)**

OX o‘qiga ma’lumot yozish: **xlabel(‘text’)**

Yuqoridagi funksiyani ishlash jarayonini quyidagi dastur orqali qarab o‘tamiz.

Misol. Ma’lum bir oraliqda cos(x) funksiya grafigini chizing va bu grafikni nomini kosinus grafigi, OX o‘qini argument va OY o‘qini funksiya deb nom beruvchi dastur tuzing.

```
from numpy import*
from math import*
from matplotlib.pyplot import*
for i in range(400):
t.append(i*0.01)
x.append(cos(pi*t[i]/180))
plot(t, x, color='green')
title(‘Kosinus grafigi’)
xlabel(‘argument’);
ylabel(‘funksiya’);
show()
```



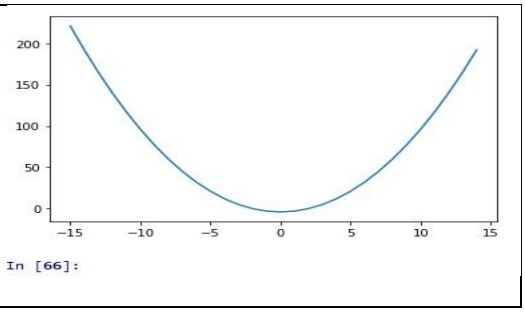
Matematik funksiyalar grafiklarini chizish

Python dasturlash tili tarkibida oddiy chizmalardan tashqari matematik funksiyalar grafiklarini chizish ham mumkin. Matematik funksiyalar grafiklarini chizishda avval, argumentning oraliq qiymati beriladi, bu xuddiki MAPLE tizimidagi kabi aniqlanadi. Argumentning qiymatlari qanchalik katta bo‘lsa grafik ham shunch aniq chiziladi.

Yuqoridagi fikrlarni shakllantirishni quyidagi dastur orqali qarab o‘tamiz.

Misol. Ma’lum bir oraliqda $y=x^2-4$ funksiya grafigini chizish dasturini tuzing.

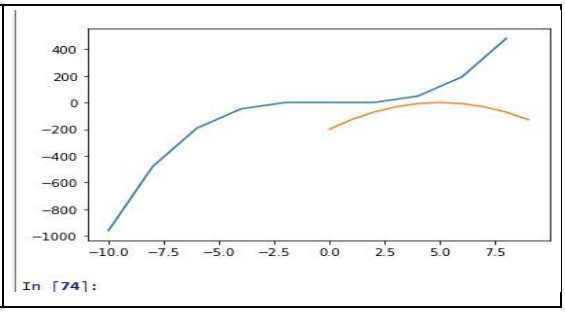
```
from numpy import*
from math import*
from matplotlib.pyplot import*
x=arange(-15,15,1);
y=x***2-4;
plot(x,y)
show()
```



Ikki va undan ortiq funksiyalar grafiklarini bitta sistemaga ham chizish mumkin.

Misol. Ma’lum bir oraliqda $y=x^3-4x$ *va* $y=-2x^2$ funksiya grafiklarini chizish dasturini tuzing.

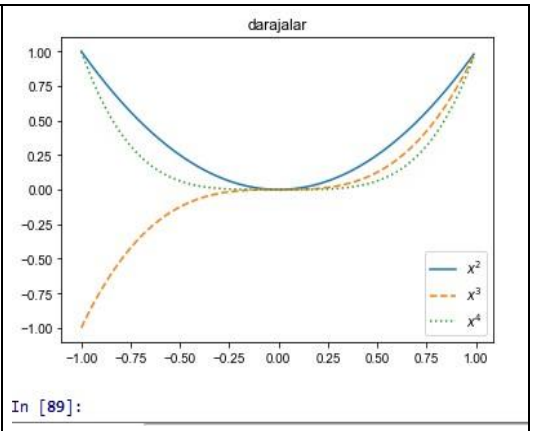
```
        from numpy import*
        from math import*
from matplotlib.pyplot import*
        x=arange(-10,10,2);
        y=x**3-4*x z=-2*x**2 ;
        plot(x,y,z)
        show()
```



Bir nechta funksiyalar grafiklarini chizish va bu funksiyalarni ko‘rinishlarini ham chizish imkoniyatlari mavjud.

Misol. Ma’lum bir oraliqda $y=x^2$, $y=x^3$ *va* $y=x^4$ funksiya grafiklarini chizish dasturini tuzing.

```
        from numpy import*
        from math import*
        from matplotlib.pyplot import*
from matplotlib import rcParams
rcParams ['font.sans-serif']=['Arial']
        t=arange(-1,1,0.01)
        x=t**2 y=t**3 z=t **4
        plot(t,x,label =r'$x ^2 $')
        plot(t,y,'--',label =r'$x^3 $')
        plot(t,z,':',label=r'$x^4 $')
        legend () title ('darajalar')
        show ()
```



17-MAVZU. PYTHON DASTURLASH TILIDA DIAGRAMMALAR VA UCH O‘LCHOVLI GRAFIKLAR CHIZISH

Reja:

1. Diagrammalar;
2. Uch o‘lchovli grafika.
3. Matplotlib yordamida 3 o‘lchovli chiziqli grafik

Tayanch so‘zlar: *diagramma, subplot, mplot3d.*

Diagrammalar

Python dasturlash tili tarkibida ma’lumotlarni vizual tarzda namoyon qilish uchun diagrammalarni shakllantirish imkoniyatlari mavjud.

Diagrammalar python dasturlash tilida ikki turga ajratiladi:

- Ustun ko‘rinishidagi diagrammalar;
- Aylana ko‘rinishidagi diagrammalar.

Ustun ko‘rinishidagi diagrammalar ma’lumotlarni OX va OY o‘qi kesimida vertikal ko‘rinishida shakllantiriladi. Ma’lumotlarni ustun ko‘rinishida shakllantirish uchun quyidagi funksiyalar faollashtiriladi.

subplot()

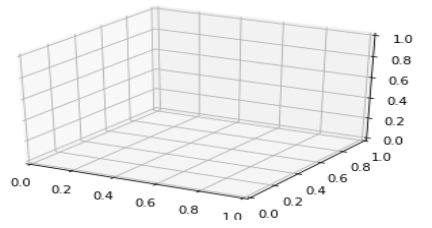
hist()

3D chizmalar ikkita bog‘liq va bitta mustaqil o‘zgaruvchiga ega bo‘lgan ma'lumotlar kabi uch o‘lchovli ma'lumotlarni vizualizatsiya qilish uchun juda muhim vositadir. Ma'lumotni 3D chizmalarida chizish orqali biz uchta o‘zgaruvchiga ega bo‘lgan ma'lumotlarni chuqurroq tushunishimiz mumkin. Biz 3D syujetlarni chizish uchun turli xil matplotlib kutubxona funksiyalaridan foydalanishimiz mumkin.

Matplotlib yordamida uch o‘lchovli chizmalarga misol

Biz birinchi navbatda Matplotlib kutubxonasi yordamida 3D o‘qini chizishdan boshlaymiz. 3D o‘qini chizish uchun plt.axes() proyeksiya parametrini None dan 3D ga o‘zgartirish kifoya.

```
import numpy as np
import matplotlib.pyplot as plt
fig = plt.figure()
ax = plt.axes(projection='3d')
```



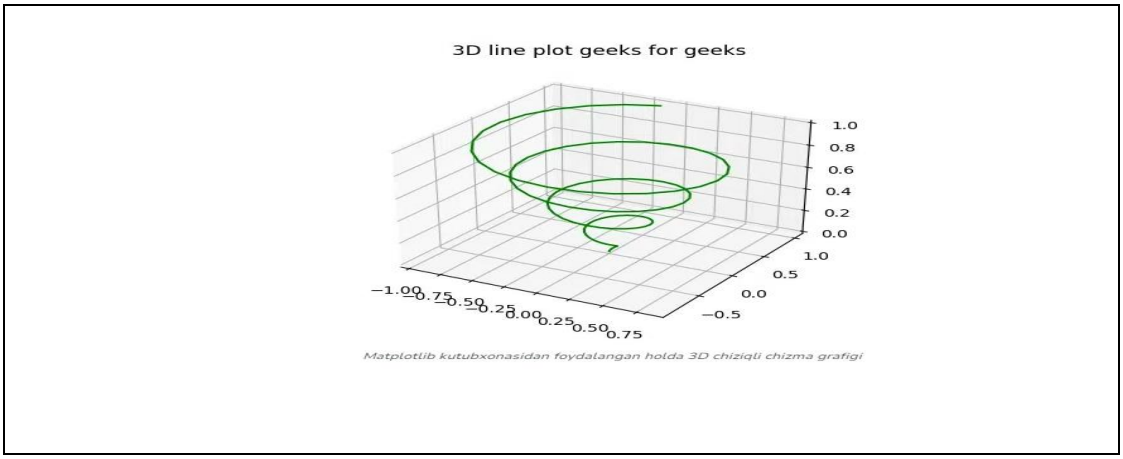
Yuqoridagi sintaksis bilan uch o‘lchovli o‘qlar yoqilgan va ma'lumotlarni 3 o‘lchamda chizish mumkin. 3 o‘lchovli grafik dinamik yondashuvni beradi va ma'lumotlarni yanada interaktiv qiladi. 2 o‘lchamli grafiklar singari, biz 3 o‘lchamli grafiklarni tasvirlash uchun turli usullardan foydalanishimiz mumkin. Biz tarqalish chizmasi, kontur chizmasi, sirt grafigi va boshqalarni yaratishimiz mumkin. Keling, turli xil 3 o‘lchamli chizmalarni ko‘rib chiqaylik.

Chiziqlar va nuqtali grafiklar eng oddiy 3 o‘lchovli grafikdir. Chiziq va nuqtali grafikni mos ravishda chizish uchun ax.plot3d va ax.scatter funksiyalaridan foydalanamiz.

Matplotlib yordamida 3 o‘lchovli chiziqli grafik

3 o‘lchovli chiziqli grafikni tuzish uchun mpl_toolkits kutubxonasidagi mplot3d funksiyasidan foydalanamiz. 3D-da chiziqlarni chizish uchun biz chiziq tenglamasi uchun uchta o‘zgaruvchan nuqtani ishga tushirishimiz kerak. Bizning holatda, biz uchta o‘zgaruvchini *x, y va z* sifatida aniqlaymiz.

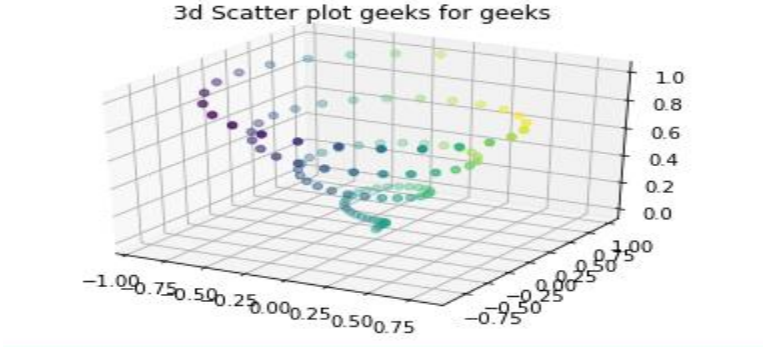
```
# importing mplot3d toolkits, numpy and matplotlib
from mpl_toolkits import mplot3d
import numpy as np
import matplotlib.pyplot as plt
fig = plt.figure()
# syntax for 3-D projection
ax = plt.axes(projection ='3d')
# defining all 3 axis
z = np.linspace(0, 1, 100)
x = z * np.sin(25 * z)
y = z * np.cos(25 * z)
# plotting
ax.plot3D(x, y, z, 'green')
ax.set_title('3D line plot geeks for geeks')
plt.show()
```



Matplotlib yordamida 3 o‘lchovli tarqoq grafik

Tarqalish nuqtalari yordamida bir xil grafikni chizish uchun biz matplotlib-dan scatter() funksiyasidan foydalanamiz . U aniq nuqtalardan foydalangan holda bir xil chiziqli tenglamani tuzadi.

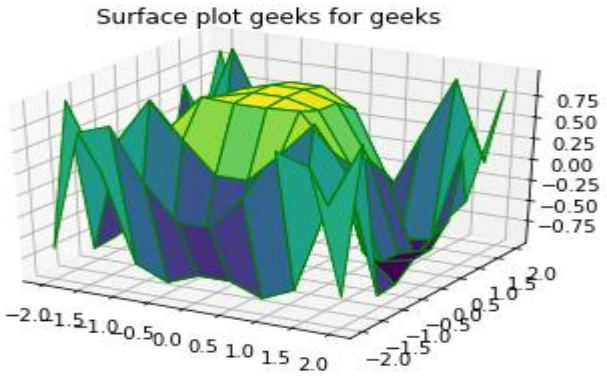
```
# importing mplot3d toolkits
from mpl_toolkits import mplot3d
import numpy as np
import matplotlib.pyplot as plt
fig = plt.figure()
# syntax for 3-D projection
ax = plt.axes(projection ='3d')
# defining axes
z = np.linspace(0, 1, 100)
x = z * np.sin(25 * z)
y = z * np.cos(25 * z)
c = x + y
ax.scatter(x, y, z, c = c)
# syntax for plotting
ax.set_title('3d Scatter plot geeks for geeks')
plt.show()
```



Matplotlib kutubxonasidan foydalangan holda sirt grafiklari

Yuzaki grafiklar va Wireframes grafigi gridlangan ma'lumotlarda ishlaydi. Ular panjara qiymatini oladilar va uni uch o‘lchamli yuzada chizadilar. Sirt grafigini chizish uchun **plot_surface()** funksiyasidan foydalanamiz.

```
# importing libraries
from mpl_toolkits import mplot3d
import numpy as np
import matplotlib.pyplot as plt
# defining surface and axes
x = np.outer(np.linspace(-2, 2, 10), np.ones(10))
y = x.copy().T
z = np.cos(x ** 2 + y ** 3)
fig = plt.figure()
# syntax for 3-D plotting
ax = plt.axes(projection='3d')
# syntax for plotting
ax.plot_surface(x, y, z, cmap='viridis',\edgecolor='green')
ax.set_title('Surface plot geeks for geeks')
plt.show()
```

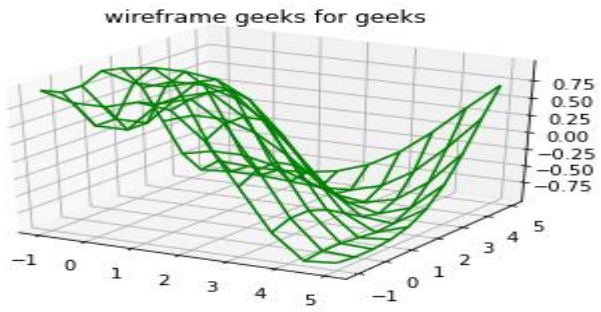


Matplotlib kutubxonasidan foydalangan holda simli ramkalar grafigi

Wireframe grafigini tuzish uchun biz matplotlib kutubxonasidan **plot_wireframe()** funksiyasidan foydalanamiz.

```
from mpl_toolkits import mplot3d
import numpy as np
import matplotlib.pyplot as plt
# function for z axis
def f(x, y):
 return np.sin(np.sqrt(x ** 2 + y ** 2))
# x and y axis
x = np.linspace(-1, 5, 10)
```

```
y = np.linspace(-1, 5, 10)
X, Y = np.meshgrid(x, y)
Z = f(X, Y)
fig = plt.figure()
ax = plt.axes(projection ='3d')
ax.plot_wireframe(X, Y, Z, color ='green')
ax.set_title('wireframe geeks for geeks');
```

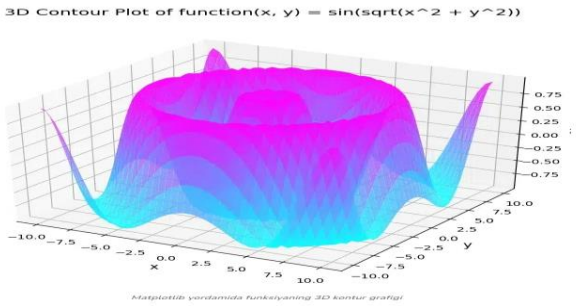


Matplotlib kutubxonasidan foydalangan holda kontur grafiklari

Kontur grafigi barcha kirish ma'lumotlarini ikki o‘lchovli muntazam panjaralarda oladi va Z ma'lumotlari har bir nuqtada baholanadi. Kontur grafigini chizish uchun ax.contour3D funksiyasidan foydalanamiz. Kontur chizmalari optimallashtirish chizmalarini tasavvur qilishning ajoyib usuli hisoblanadi.

```
def function(x, y):
    return np.sin(np.sqrt(x ** 2 + y ** 2))
x = np.linspace(-10, 10, 40)
y = np.linspace(-10, 10, 40)
X, Y = np.meshgrid(x, y)
Z = function(X, Y)
fig = plt.figure(figsize=(10, 8))
ax = plt.axes(projection='3d')
ax.plot_surface(X, Y, Z, cmap='cool', alpha=0.8)
ax.set_title('3D Contour Plot of function(x, y) =\ sin(sqrt(x^2 + y^2))', fontsize=14)
ax.set_xlabel('x', fontsize=12)
ax.set_ylabel('y', fontsize=12)
ax.set_zlabel('z', fontsize=12)
```

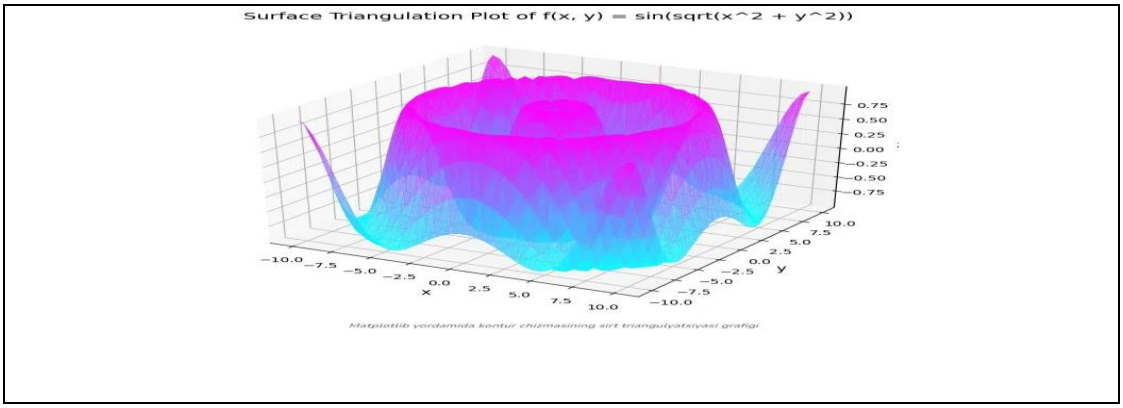
```
plt.show()
```



Pythonda sirt uchburchaklarini chizish

Yuqoridagi grafik ba'zan haddan tashqari cheklangan va noqulay. Shunday qilib, bu usul bilan biz tasodifiy o‘yinlar to‘plamidan foydalanamiz. Bu grafikni chizish uchun **ax.plot_trisurf** funksiyasidan foydalaniladi. Bu unchalik aniq emas, lekin yanada moslashuvchan.

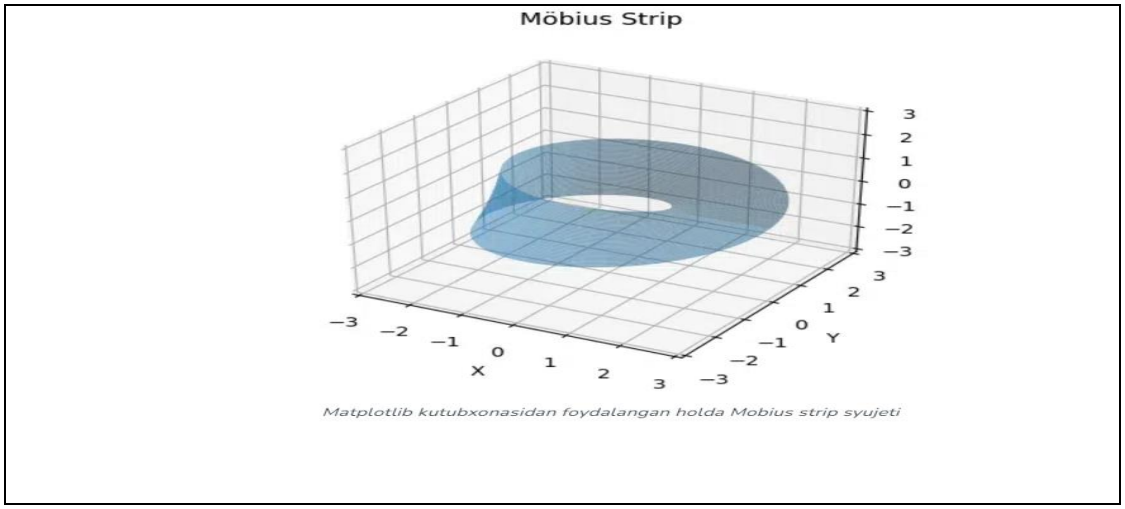
```
import numpy as np
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
from matplotlib.tri import Triangulation
def f(x, y):
    return np.sin(np.sqrt(x ** 2 + y ** 2))
x = np.linspace(-6, 6, 30)
y = np.linspace(-6, 6, 30)
X, Y = np.meshgrid(x, y)
Z = f(X, Y)
tri = Triangulation(X.ravel(), Y.ravel())
fig = plt.figure(figsize=(10, 8))
ax = fig.add_subplot(111, projection='3d')
ax.plot_trisurf(tri, Z.ravel(), cmap='cool', edgecolor='none', alpha=0.8)
ax.set_title('Surface Triangulation Plot of f(x, y) =\
        sin(sqrt(x^2 + y^2))', fontsize=14)
ax.set_xlabel('x', fontsize=12)
ax.set_ylabel('y', fontsize=12)
ax.set_zlabel('z', fontsize=12)
plt.show()
```



Pythonda Mo‘bius chizig‘ini chizish

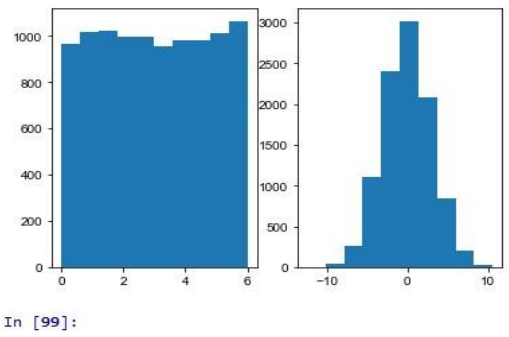
Mo‘bius tasmasi o‘ralgan silindr deb ham ataladi, bu chegarasiz bir tomonlama sirtdir. Mo‘bius tasmasini yaratish uchun uning parametrlari haqida o‘ylang, bu ikki o‘lchovli chiziq va bizga ikkita ichki o‘lcham kerak. Uning burchagi pastadir atrofida 0 dan 2 tagacha, kengligi esa -1 dan 1 gacha.

```
import numpy as np
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
# Define the parameters of the Mo‘bius strip
R = 2
# Define the Mo‘bius strip surface
u = np.linspace(0, 2*np.pi, 100)
v = np.linspace(-1, 1, 100)
u, v = np.meshgrid(u, v)
x = (R + v*np.cos(u/2)) * np.cos(u)
y = (R + v*np.cos(u/2)) * np.sin(u)
z = v * np.sin(u/2)
# Create the plot
fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')
# Plot the Mo‘bius strip surface
ax.plot_surface(x, y, z, alpha=0.5)
# Set plot properties
ax.set_xlabel('X')
ax.set_ylabel('Y')
ax.set_zlabel('Z')
ax.set_title('Mo‘bius Strip')
# Set the limits of the plot
ax.set_xlim([-3, 3])
ax.set_ylim([-3, 3])
ax.set_zlim([-3, 3])
# Show the plot
plt.show()
```



Misol. Random funksiyasi orqali hosil qilingan ikkita ro‘yxat elementlarini ustun ko‘rinishidagi diagramma chizish dasturini tuzing.

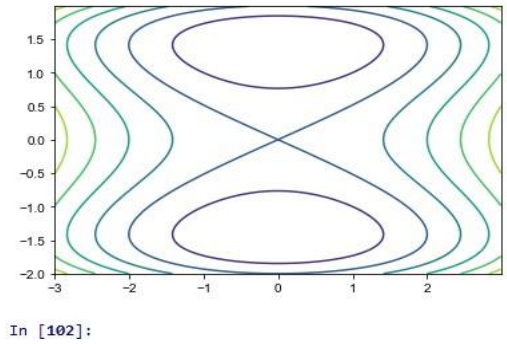
```
from numpy import*
from math import*
from matplotlib import rcParams
from random import uniform, normalvariate
from matplotlib.pyplot import *
v=[]
for i in range(10000):
v.append ( uniform (0,6))
subplot (1,2,1)
hist (v)
w = []
for i in range(10000):
w.append(normalvariate(0, 3))
subplot(1, 2, 2)
hist(w )
show()
```



Aylana ko‘rinishidagi diagrammalar ma’lumotlarni OX va OY o‘qi kesimida aylana ko‘rinishida shakllantiradi. Ma’lumotlarni aylana ko‘rinishida shakllantirish uchun quyidagi aylana tenglamasidan ham foydalaniladi.

Misol. Ikkita to‘plam elementlarini aylana ko‘rinishidagi diagramma chizish dasturini tuzing.

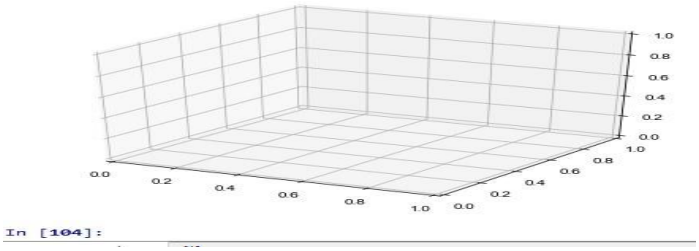
```
    from numpy import *
from matplotlib.pyplot import *
from matplotlib.mlab import*
    x = arange(-3, 3, 0.01)
    y=arange(-2, 2, 0.01)
    X,Y =meshgrid(x,y)
    Z=X**2-4*Y**2+Y**4
    contour (x,y,z)
    show()
```



Uch o‘chovli grafika

Python dasturlash tili tarkibidagi Matplotlib kutubxonasida, ikki o‘lchovli grafikaga qo‘shimcha ravishda, uch o‘lchovli grafikalarni hosil qilish imkoniyati mavjud. Buning uchun **mplot3d** moduli ishlatiladi. Uch o‘lchovli grafiklarni chizish uchun, birinchi navbatda siz uch o‘lchovli ma’lumotlarni shakllantirishingiz kerak bo‘ladi. Bunda **mpl_toolkits.mplot3d** funksiyasini **Axes3D** sinf ob’ekti sifatida o‘rnatiladi. Uch o‘lchovli grafikani faollashtirishni quyidagi dastur orqali qarab o‘tamiz.

```
    from matplotlib.pyplot import *
    from mpl_toolkits.mplot3d import Axes3D fig = figure ()
    Axes3D (fig)
    show()
```



Foydalanilgan adabiyotlar ro‘yxati

1. Mirziyoyev Sh.M. O‘zbekiston Respublikasini yanada rivojlantirish bo‘yicha harakatlar strategiyasi to‘g‘risidagi PF-4947-sonli Farmon. 2017 yil 7 fevral.

2. M.O‘.O‘ktamov DASTURLASH TILLARI I fanidan o‘quv qo‘llanma. — Toshkent.: “GRAND KONDOR PRINT”, 2024 y., 160 b.

3. Aminov I.B., Oqnazarov T.J., Mannabov J.T. Algoritmlash asoslari. Samarqand, 2019.

4. Sh.A. Mengliyev, O.A. Abdug‘aniev, S.Q. Shonazarov, D. Sh. To‘rayev. Python dasturlash tili, 2021.

5. D.Saidov. Python dasturlash tili, Toshkent, 2019.

6. A.Axatov, F.Nazarov. Python tilida dasturlash asoslari (1-qism), Samarqand, 2020.

7. C++ Programming: From Problem Analysis to Program Design. Fifth Edition. Course Technology, 2011.

8. Bjarne Stroustrup. The C++ Programming Language (4th Edition). Addison-Wesley, 2013, 1363 pages.

9. Sh.F.Madraximov, A.M.Ikramov, Q.T.Maxarov, Dasturlash asoslari. O‘quv qo‘llanma // Toshkent, “Mumtoz so‘z”, 2018. 276 bet.

10. Sh.F.Madraximov, S.M.Gaynazarov, “C++ tilida programmalash asoslari” – Toshkent, O‘zMU-2009, 196 bet.

11. A.M.Polatov, Algoritmlar va C++ tilida dasturlash asoslari. O‘quv qo‘llanma // Toshkent, O‘zbekiston Milliy Universiteti, “Universitet” nashriyoti, -2017. 100 bet.

12. Sh.F.Madraximov, A.M.Ikramov, M.R.Babajanov C++ tilida programmalash bo‘yicha masalalar to‘plami. O‘quv qo‘llanma // Toshkent, O‘zbekiston Milliy Universiteti, “Universitet” nashriyoti, -2014. 160 bet.

13. Г.Шилдт –“Полный справочник по C++” – M-2006., 801 стр.

14. E.A.Eshboyev, F.Y.Shodiyev. C++ tilida dasturlash asoslari. Uslubiy qo‘llanma. Qarshi-2017, 190 s.

15. O.M.Shukurov, F.Q.Qorayev, E.A.Eshboyev, B.H.Shovaliyev – “Dasturlashdan masalalar to‘plami”. T-2008 y., 160 s.

16. O.M.Shukurov, E.A.Eshboyev, B.H.Shovaliyev – “Delphi va C++ algoritmik tillarida dasturlash”. Qarshi-2012 y., 228 s.

17. Bjarne Stroustrup. Programming: Principles and Practice using C++ (Second Edition). Addison-Wesley − 2014, 1305 pages.

18. A.A.Xaldjigitov, Sh.F.Madraximov, U.E.Adambayev, E.A Eshboyev, Informatika va programmalash. T.:O‘zMU, 2005 y, -148 s.

19. Т.С.Павловская, Y.С.Шупак C/C++. Структурное программирование. Практикум.-СПб.: Питер-2002, 240 с.

20. А.Е.Мудров «Численные методы для ПЕВМ на языках Бейсик, Фортран и Паскаль». Томск МП «Раско»1991 г. 271 стр.

21. A.U.Abdulhamidov, S.X.Xudoynazarov. Hisoblash usullaridan mashqlar va laboratoriya ishlari. –T.: “O‘zbekiston”, 1995. 230 s.

22. http://www.wikipedia.org

23. http://cppstudio.com

24. http://cplusplus.com

25. http://www.compteacher.ru/programming

26. http://www.intuit.ru

27. www.acm.timus.ru

28. www.geeksforgeeks.org

29. www.uzbekdevs.uz

30. www.w3schools.com/

31. www.python.org

32. www.opensource.com

33. www.eduproger.uz

34. www.codeforces.ru

35. www.fcoders.uz

36. www.ziyonet.uz

37. www.dasturchi.uz

M.O‘.O‘KTAMOV

DASTURLASH TILLARI II

fanidan

(O‘QUV QO‘LLANMA)

Bosishga ruxsat etildi. 27.07.2024 y.
Qog`oz bichimi 60x84 1/16. Times New Roman garniturasida terildi.
Ofset uslubida oq qog`ozda chop etildi.
Nashriyot hisob tabog`i 11.5, Adadi 200. Buyurtma № 72
Bahosi kelishuv asosida

“BROK CLASS SERVIS” MChJ bosmaxonasida chop etildi.
Manzil: Toshkent shahar Zargarlik ko'chasi, Segizbayeva 10a.