O‘ZBEKISTON RESPUBLIKASI
OLIY VA O‘RTA MAXSUS TA’LIM VAZIRLIGI
DENOV TADBIRKORLIK VA PEDAGOGIKA INSTITUTI

MAMAJANOV R.Y., RAJABOV T.J., MERGANOV SH.M.,
MENGTURAYEV F.Z, XUSHBOQOV I.U., XOLBEKOV A.M.

ZAMONAVIY DASTURLASH TILLARI VA TEXNOLOGIYALARI (python)

Denov-2022

MUNDARIJA

KIRISH

Python dasturlash tili hozirgi kunda top reytinglarida yuqorida turgan dasturlash tillaridan biridir. Biz ushbu qo‘llanmada Python dasturlash tilini bosqichma-bosqich o‘rganib chiqishingiz uchun ushbu qo‘llanmani yozishga qaror qildik. Kelajakda siz bu dasturlash tilini turli IT yo‘nalishiga qaratilgan sohalarda ishlatish imkoniyatiga ega bo‘lasiz: Web-dasturlashdan boshlab turli xildagi o‘yinlargacha dasturlash imkoniyati mavjud.Python bu universal professional dasturlash tili bo‘lib, uni har qanday turdagi dasturiy mahsulotlar ishlab chiqish maqsadida foydalanish mumkin.

Googleda dasturlash tillarini o‘rganish maqsadida qilingan so‘rovlarni tahlil qilish natijasiga ko‘ra (http://pypl.github.io/PYPL.html) Python bugungi kunda dasturchi bo‘lish uchun o‘rganilayotgan dunyodagi 1-raqamli dasturlash tili deb qaralmoqda. Ushbu kurs Python dasturlash tilini boshlang‘ich darajasidan boshlab, ushbu tilning yuqori darajasigacha bo‘lgan mavzularni qamrab olgan. Siz umuman dasturlashdan habardor bo‘lmasangiz ham, yoki biroz habardor bo‘lsangiz ham yoki uni yuqori darajada o‘rganishni hohlasangiz ham, bizning darslarga qo‘shilishingiz mumkin!

O‘quv qo‘llanma 3-qismdan tashkil topgan:

1-qismda asosiy to‘g‘ri dasturlashning asoslarni o‘rganishga va dasturchidek fikrlashni o‘rgatishga asoslangan.

2-qismda Python dasturlash tilidan foydalanib dasturiy loyihlarni qanday qilib mutahasis kabi ishlab chiqish va dasturchi ishini osonlashtiruvchi turli imkoniyatlar yoritilgan.

3-qismda asosiy e’tibor, ma’lumotlar bilan ishlashga qaratilgan, xususan, ma’lumotlarni saqlash, ma’lumotlarni tahlil qilish va ma’lumotlarni visual tasvirlash keltirilgan.

Shuningdek, darslar nafaqat dasturlashga oid mavzularni, balki, dars davomidagi amaliy ishlar, qiziqarli muammolarning avtomatlashtirilgan yechimlari, jamoaviy/mustaqil loyihalar, ajoyib testlar, savol-javob sessiyalari va yana turli interaktiv qismlarni o‘z ichiga olgan.

I-BOB. PYTHON DASTURLASH TILI VA SINTAKSISI

1.1. Python dasturlash tili va uning imkoniyatlari

Python (talaffuzi: Piton) - umumiy-maqsadli dasturlash uchun keng tarzda foydalaniladigan yuqori darajali dasturlash tili hisoblanadi. Ushbu dasturlash tili Guido van Rossum tomonidan yaratilgan va birinchi marta 1991-yilda foydalanib koʻrilgan.

Python har xil platformalar uchun yozilgan, masalan Windows, Linux, Mac OS X, Palm OS, Mac OS va boshqalar. Python Microsoft.NET platformasi uchun yozilgan realizatsiyasi ham mavjud boʻlib, uning nomi – Iron Python.

Python tili Perl, C va Java kabi koʻp oʻxshashliklarga ega. Biroq, tillar orasida aniq farqlar mavjud. Har bir tilning alifbosi boʻlgani singari Python dasrurlash tili ham oʻz alifbosiga ega. Python dasturlash tilining alifbosi katta va kichik lotin harflari, arab raqamlari, maxsus belgular va xizmatchi soʻzlaridan tashkil topgan. Python mashhur dasturlash tili. Python dasturlash tiliga 1991 yil Golland dasturchisi Grido Van Rossu asos solgan.

Python – bu oʻrganishga oson va shu bilan birga imkoniyatlari yuqori boʻlgan zamonaviy dasturlash tillari qatoriga kiradi. Python yuqori darajadagi ma’lumotlar strukturasi va oddiy lekin samarador obyektga yoʻnaltirilgan dasturlash uslublarini taqdim etadi.

Stack Overflow saytining 2019-yildagi dasturchilar oʻrtasida dasturlash tillari boʻyicha olib borilgan soʻrovnomasida eng qulay va koʻp foydalaniladigan dasturlash tillari roʻyxatida JavaScriptdan soʻng ikkinchi oʻrinni egallagan. Shu bilan bir qatorda dunyoning Twitter, Pinterest, HP, Symantec, Instagram va Groupon kabi yirik korxonalar aynan Python dasturlash tilidan foydalanmoqda.

Rasmiy sayt – *www.python.org*

Python mashhur dasturlash tili. U Guido van Rossum tomonidan yaratilgan va 1991 yilda chiqarilgan.

Python – bu oʻrganishga oson va shu bilan birga imkoniyatlari yuqori boʻlgan oz sonlik zamonaviy dasturlash tillari qatoriga kiradi. Python yuqori darajadagi

ma’lumotlar strukturasi va oddiy lekin samarador obyektga yo‘naltirilgan dasturlash uslublarini taqdim etadi.

Stack Overflow saytining 2019-yildagi dasturchilar o‘rtasida dasturlash tillari bo‘yicha olib borilgan so‘rovnomasida, eng qulay va ko‘p foydalaniladigan dasturlash tillari ro‘yxatida Python JavaScriptdan so‘ng ikkinchi o‘rinni egallagan. Shu bilan bir qatorda dunyoning Twitter, Pinterest, HP, Symantec, Instagram va Groupon kabi yirik korxonalar aynan Python dasturlash tilidan foydalanmoqda. YouTube, DropBox, Google va Quora kabi dunyoning mashhur online platformalarining dasturiy ta’minoti ham aynan python dasturlash tilida yozilganligi ushbu dasturlash tiliga bo‘lgan talabning yuqori ekanligini anglatadi. Python nafaqat web sohasida balki sun’iy intellekt va robotexnika sohasida ham yuqori talabga egatil hisoblanadi.

YouTube, DropBox, Google va Quora kabi dunyoning mashhur online platformalarining dasturiy ta’minoti ham aynan Python dasturlash tilida yozilganligi ushbu dasturlash tiliga bo‘lgan talabning yuqori ekanligini anglatadi. Python nafaqat web sohasida balki sun’iy intellekt va robotexnika sohasida ham yuqori talabga ega til hisoblanadi.

Python turli platformalarda ishlaydi. (Windows, Linux, Mac va h.k). Python ingliz tiliga o‘xshash oddiy sintaksisga ega.

Python dasturlash tili boshqa dasturlash tillariga qaraganda dasturchiga kamroq kod yozish imkonini beradi.

Python da WEB, Desktop va Mobile dasturlar yaratish imkoniyati mavjud.

Python dasturlash tilida kutubxonalar anchagina ko‘p! deyarli barcha ishingizni kutubxonalar orqali bajarasiz!

Python, Cpython, Pyjamas, Shed Skin, Pyrex orqali C#, C++, C, Java, Javascript kutubxonalari va kodlari bilan ishlashda yordam beradi. PyGame va PyKyra orqali har qanday o‘yin yaratish mumkin. ENG MUHIMI. Sintaksis juda toza va tushunarli, ortiqcha arifmetikasiz. Kalkulyator kodini ko‘rib chiqishingiz mumkin. Bazi faylasuflar gapi: Beautiful is better than ugly. Explicit is better than implicit. Simple is better than complex. Complex is better than complicated.

Readability counts. Eslatib oʻtamiz bu til va Google ishlayapti. Bu yerda asosiylari keltirib oʻtildi.

Bu tilni sintaksisi juda tushunarli va tez oʻrganishga yordam beradi. Paradigmlariga kelsak Multi-paradigm: object-oriented, imperative, functional, procedural, reflective.Python dasturlash tili sodda va oʻqilishi oddiy boʻlgan dasturlash tili boʻlib u inglizcha soʻzlarni qoʻllab quvvatlaydi va kalit soʻzlar oʻrnida shuning uchun bu boshqacha koʻrinishga ega. Python Interpretori: Bu tarjimon tomonidan ish vaqtida qayta ishlanganligini va uni bajarishdan oldin dasturni kompilyatsiya qilishning hojati yoʻqligini bildiradi. Bu PERL va PHP ga oʻxshaydi.

Python xotiradan kam joy egallaydi va ishlash tezlgi ancha yuqori! Python dasturlash tilining eng katta imkoniyati uning ochiq kodligida va kutubxonalarning koʻpligida va shu bilan birga undagi kodni ixchamligida hisoblanadi. Hozir sizga 3 ta dasturlash tilida C, Java va Python dasturlash tilida "Salom, dasturga xush kelibsiz" soʻzini yozib koʻramiz!

Python – bu oʻrganishga oson va shu bilan birga imkoniyatlari yuqori boʻlgan osonlik zamonaviy dasturlash tillari qatoriga kiradi. Python yuqori darajadagi ma’lumotlar strukturasi va oddiy lekin samarador obyektga yoʻnaltirilgan dasturlash uslublarini taqdim etadi. Python quyidagi sohalarda ishlatiladi:

- Web dasturlash (serverlar bilan)
- Dasturiy ta’minot
- Matematika
- Tizim skriptlari
- Tizimli dasturiy ta’minot

Savol. Nima uchun aynan Python dasturlash tilini oʻrganish kerak ?

Buning sabalari juda oddiy. Masalan,

- ✓ Oddiy, oʻrganishga qulay, 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.

✓ Koʻplab platformalarda hech qanday oʻzgartirishlarsiz ishlay oladi (Windows, Mac, Linux, Raspberry Pi va boshqalar).

✓ Interpretatsiya(Интерпретируемый) qilinadigan til.

✓ Kengayishga (Расширяемый) moyil til. Agar dasturni biror joyini tezroq ishlashini xoxlasak shu qismni C yoki C++ dasturlash tillarida yozib keyin shu qismni Python kodi orqali ishga tushir imkoniyati(chaqirsa) mavjudligi.

Juda ham koʻp xilma-xil kutubxonalarga ega.

- xml/html fayllar bilan ishlash;
- http soʻrovlari bilan ishlash;
- GUI(grafik interfeys);
- Web dastur tuzish;
- FTP bilan ishlash;
- Rasmli audio, video fayllar bilan ishlash;
- Robototexnikada.

Matematik va ilmiy hisoblashlarni dasturlashda juda ham qoʻl keladi. Pythonni katta proyektlar (loyihalar)da ishlatish mumkin. Bularga misol qilib Google, Instagram, YouTube ni misol qilib koʻrsatsak boʻladi. Koʻpgina loyiahalar aynan Python dasturlash tilida yozilgan. Chunki, uni chegarasi yoʻq, imkoniyati yuqori. Shuningdek, u sodda va universalligi bilan dasturlash tillari orasida eng yaxshisidir.

Python dasturlash tilini yaratilishi 1980-yil oxiri 1990-yil boshlaridan boshlangan. Oʻsha paytlarda uncha taniqli boʻlmagan Gollandiyaning CWI instituti xodimi Gvido van Rossum ABC tilini yaratilish loyihasida ishtirok etgan edi. ABC tili Basic tili oʻrniga talabalarga asosiy dasturlash konsepsiyalarini oʻrgatish uchun moʻljallangan til hisoblanar edi. Bir kun Gvido bu ishlardan charchadi va 2 hafta davomida oʻzining Macintoshida boshqa oddiy tilning interpretatorini yozdi, bunda u albatta ABC tilining baʼzi bir gʻoyalarini oʻzlashtirdi. Shuningdek, Python 1980-1990-yillarda keng foydalanilgan Algol-68, C, C++, Modul3 ABC, SmallTalk tillarining koʻplab xususiyatlarini oʻziga olgandi. Gvido van Rossum bu

tilni internet tarmog‘i orqali tarqata boshladi. Bu paytda o‘zining “Dasturlash tillarining qiyosiy taqrizi” veb sahifasi bilan internetda 1996-yilgacha Ctiv Mayevskiy taniqli edi. U ham Macintoshni yoqtirardi va bu narsa uni Gvido bilan yaqinlashtirdi. O‘sha paytlarda Гвидо Би-би-си “Mongi Pythonning havo sirki” komediyasining muxlisi edi va o‘zi yaratgan tilni Monti Peyton nomiga Python deb atadi. 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 obyektga yo‘naltirilgan dasturlash (OYD) vositalari paydo bo‘ldi. Bir qancha vaqt o‘tib Gvido Gollandiyadan Amerikaga ko‘chib o‘tadi va uni CNRI korparatsiyasiga ishlashga taklif etishadi. U o‘sha yerda ishladi va korparatsiya shug‘ullanayotgan loyihalarni Python tilida yozadi va bo‘sh ish vaqtlarida tilni interpretatorini rivojlantirib boradi. Bu 1990-yil Python dasturlash tilini 1.5.2 versiyasi paydo bo‘lguncha davom etadi. Gvidoning asosiy ishi korparatsiyani loyihalarini yaratishga vaqti ketar edi.

Python dasturlash tilida yaratilgan yirik loyihalar.:

1.2.Python dasturlash tili sintaksisi.

- Satr oxiri instruksiyaning oxiri hisoblanadi (nuqta vergul shart emas)
- Har bir qator boshidagi bo‘sh joy(otstup) muhim ahamiyatga ega.

Kiritilgan amallar bo‘sh joylarning kattaligiga qarab bloklarga 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.

Har bir dasturlash tilining o‘ziga xos kalitli so‘zlari mavjud. Jumladan Python dasturlash tilida**.**

Kalit so‘zlar	
False – yolg`on.	del – obyektni yo`qotish.
if - agar.	class – metod va atributlarda iborat.
True - rost.	not –mantiqiy INKOR amali.
else-for/else yoki if/elsega qarang.	continue – sikldan keyingi iteratsiyaga
None - “bo`sh” obyekt.	or –mantiqiy YOKI amali o’tish.
elif – aks holda, agar.	and – mantiqiy VA amali.
while – while sikli	from – moduldan bir nechta funksiyani import qilish.
for – for sikli.	lambda-yashirin funksiyani aniqlash.
with / as – konteks menejeri.	import – moduldan import.
def – funksiyani aniqlash.	del – obyektni yo`qotish.
break – sikldan chiqish.	

Dastur tuzishda va kiritshda ham imkoniyatlari yuqoriligini ko‘rish mumkin. Masalan. **C dasturlash tili:**

```
#include<stdio.h>
int main (int arge, char ‘ ‘ argv)
{
printf(“Salom talaba:\n”;);
}
```

Java dasturlash tili:

```
public class Salom
{
Public static void main(String argv()}
{
System.out.println(“Salom, dasturga xush kelibsiz!”);
}
}
```

Python dasturlash tili:

```
print (" Salom, dasturga xush kelibsiz ")
```

Bundan ko‘rinib turibdiki Python dasturlash tilida dasturchi tomonidan kiritiladigan kod ixchamlashtirilgan va sizning vaqtingizni kam oladi. Python dasturlash tilini ishlashi uchun sizdan kuchli kompyuter talab qilmaydi. Yozgan kodingizni netbook, oddiyroq harakteriskaga ega kompyutyerda ishlatish hattoki qo‘lingizdagi Android smart phoningizda ham ishlatish imkoniyati mavjud.

Server uchun juda yaxshi dasturlash tili hisoblanadi. Hozirgi vaqtda hakkerlar uchun yaratilgan dasturlash tili degan nom olgan dasturchilar orasida.

Dasturlash tilini web dasturlashda ham qo‘llasangiz bo‘ladi?.

Misol uchun kutubxonachi.uz sayti Pythonda ishlab chiqilgan. Google inc da ham Python dan keng foydalanishadi jumladan, Amazon, ZTE va bir qancha mashhur kompaniyalarda Python dasturlash tilidan foydalanishadi.

Python dasturlash tili sintaksisi o‘zi kabi juda sodda:

- satr oxiri instruksiyaning oxiri hisoblanadi (nuqta vergul shart emas) va Pythonda sintaksis juda sodda tuzilishga ega. Quyida *“Salom, dasturga xush kelibsiz”* gapini ekranga chiqaruvchi kod ko‘rsatilgan:

```
print (" Salom, dasturga xush kelibsiz ")
```

Har bir qator boshidagi bo‘sh joy(отступ) muhim ahamiyatga ega. Kiritilgan amallar bo‘sh joylarning kattaligiga qarab bloklarga 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.

Python dasturlash tilining alifbosi quyidagi jadvalda aks etgan:

№	Alifbo nomlari	Tashkil etuvchilari
1	Katta va kichik lotin harflari	A,B,C,…,X,Y,Z,a,b,c,…,x,y,z
2	Arab raqamlari	0,1,2,3,4,5,6,7,8,9
3	Maxsus belgilar	Arifmetik amallar: “+” (qo‘shish), “-” (ayirish), “*” (ko‘paytirish), “/” (bo‘lish), qavslar, tinish belgilari va b.
4	Xizmatchi so‘zlar	If, for, print,class va b.

Python tilida dastur yozishda quyidagi tayanch tushunchalar qo‘llaniladi:

№	Tayanch tushuncha nomlari	Tavsifi
1	O‘zgaruvchilar	*Dastur ishlaganda qiymati o‘zgaradigan miqdorlar*
2	Doimiy(o‘zgarmas)lar	*Dastur ishlaganda qiymati o‘zgarmaydigan miqdorlar*
3	Ifodalar	*Mos amallar bilan bog‘langan o‘zgarmaslar, o‘zgaruvchilar va funksiyalar*
4	Operatorlar	*Dasturlash tilining tugallangan biror amalini berish uchun mo‘ljallangan ko‘rsatmasi*
5	Funksiya va protseduralar	*O‘z nomiga ega bo‘lgan alohida dastur qismlari*
6	Nishonlar	*Dsaturda boshqarish uzatilayotgan operatorni ko‘rsatadi*

7	Identifikatorlar	*Doimiy (o‘zgarmas)lar, o‘zgaruvchilar, protseduralar, funksiyalar, modullar, dasturlar nomi tushuniladi*

Odatda dasturlash tillarida xat boshi kodni oson o‘qilishi uchun ishlatiladi. Ammo Pythonda *xat boshi* kodning blokini ajratib ko‘rsatadi. Misol keltiramiz:

```
if 5 > 2:
    print("Besh ikkidan katta")
```

Agar kodimizni mana bunday tarzda yozsak dasturda xatolik yuz beradi:

```
if 5 > 2:
print("Besh ikkidan katta")
    print("Besh ikkidan katta")
          ^
    IndentationError: expected an indented block
```

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

Dastur kodi	Dastur natijasi
`a = 1; b = 2; print(a, b)`	`1 2`

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

Izohlar hosil qilish.

Izohlar # belgisi bilan hosil qilinadi va python o‘sha qismni kod deb qabul qilmaydi:

Dastur kodi
`# Bu yerda izoh bor` `print("Salom,dasturga xush kelibsiz")`
Dastur natijasi
`Salom,dasturga xush kelibsiz`

Izohlarni kod yozilgan qator oxiriga yozish ham mumkin:

Dastur kodi
`print("Salom,dasturga xush kelibsiz")` `# Bu yerda izoh bor`
Dastur natijasi
Salom,dasturga xush kelibsiz

Kodning biror qismini izohga kiritsak o‘sha qism natija bermaydi. Quyidagi holatda Salom, dasturga xush kelibsiz jumlasi ekranga chiqmaydi:

Dastur kodi
`# print ("Salom, dasturga xush kelibsiz ")` `print ("Dasturlashni o'rganamiz")`
Dastur natijasi
`Dasturlashni o'rganamiz`

Izohlar dastur kodini o‘qiyotganlar uchun foydali bo‘ladi va dastur nima qilishini oson tushunishga yordam beradi. Unga yechimdagi muhim joylarni, muhim bo‘lgan qismlarni yozish mumkin.

Ko‘p qatorli izohlar

Python ko‘p qatorli izohlar hosil qilish uchun alohida belgiga ega emas. Shuning uchun har bir qator uchun alohida # belgisi ishlatiladi. Ammo 3 talik qo‘shtirnoq ichiga yozilgan matnni o‘zgaruvchiga biriktirilmasa ko‘p qatorli izoh sifatida ishlatish mumkin:

Dastur kodi	

`"""`
`Bu izoh ko‘p qatorli izohdir`
`Men Python dasturlash tilini`
`yozilish qoidalarini sodda bo‘lgani`
`uchun tez o‘rganyapman`
	`'''`
	`Bu izoh kop qatorli izohdir`
	`Men Python dasturlash tilini`
	`yozilish qoidalarini sodda bo‘lgani`
	`uchun tez o‘rganyapman`

""" print("Bilim ol!")	''' print(" Bilim ol!")
Dastur natijasi	
Bilim ol!	

Pythonda o‘zgaruvchilar (variable)

O‘zgaruvchi - kompyuter xotirasida ma’lum bir qiymatni saqlash uchun ajratilgan joy. Soddaroq qilib tushuntirsak, o‘zgaruvchini quti ichidagi narsani qiymat deb tasavvur qilish mumkin. Pythonda qiymatlar son, matn, ro‘yxat va hokazo ko‘rinishida bo‘lishi mumkin.

Biror ma’lumotni saqlash va uning ustida turli amallarni bajarish uchun bizga o‘zgaruvchilar yordam beradi. O‘zgaruvchining qiymati, o‘z nomi bilan aytib turibdiki, o‘zgarishi mumkin. Unda xohlagan qiymatni saqlash mumkin. O‘zgaruvchilar kompyuter xotirasidagi joy bo‘lib, u yerda siz biror ma’lumotni saqlaysiz. O‘zgaruvchining konstantadan farqi, o‘zgaruvchiga dastur ishlashi davomida (run time) murojaat qilib, uning qiymatini o‘zgartira olamiz. Konstantaga esa oldindan ma’lum bir qiymat beriladi va bu qiymatni o‘zgartirib bo‘lmaydi.

O‘zgaruvchilarni nomlash. O‘zgaruvchilarga nom berishda quyidagi qoidalarga amal qiling:

- o‘zgaruvchi nomi harf yoki pastki chiziq (_) bilan boshlanishi kerak;
- o‘zgaruvchi nomi raqam bilan boshlanishi mumkin emas;
- o‘zgaruvchi nomida faqatgina lotin alifbosi harflari (A-z), raqamlar (0-9) va pastki chiziq (_) qatnashishi mumkin.
- o‘zgaruvchi nomida bo‘shliq (пробел) bo‘lishi mumkin emas. O‘zgaruvchi nomida katta-kichik harflar turlicha talqin qilinadi (ism, ISM, va Ism uchta turli o‘zgaruvchi). Qo‘shimcha qoida sifatida:
- o‘zgaruvchi nomini kichik harflar bilan yozing.
- o‘zgaruvchi nomida 2 va undan ortiq so‘z qatnashsa ularning orasini pastki chiziq (_) bilan ajrating (*ism_sharif="Xolbekov_Abdusattor"*).

O‘zgaruvchiga tushunarli nom bering (y=20 emas yosh=20, d="Korea" emas davlat = "Uzbekistan" va hokazo). Shuningdek o‘zgaruvchilarga Pythonda ishlatiladigan funksiyalar va maxsus kalit so‘zlarning (keywords) nomini bermang. Kalit so‘zlar ro‘yhatini ko‘rish uchun Spyder konsolida avval help() deb yozing va Enter tugmasini bosing. Keyin esa keywords deb kiritib, yana Enter bosing.

To‘g‘ri nomlangan o‘zgaruvchilar:

programuz = "Python"

program_uz = "JAVA"

programUz = "C++"

PROGRAMUZ = "THINKPAD"

program2 = "MySql"

Noto‘g‘ri nomlangan o‘zgaruvchilar:

2program = "MySql"

program-uz = "Python"

program uz = "Python"

O‘zgaruvchi qisqa nomga ega bo‘lishi mumkin (masalan, x va y) yoki ko‘proq tavsiflovchi nom (yosh, ism, nom va h.k). Python o‘zgaruvchilar uchun qoidalar:

O‘zgaruvchilar [a-Z] harf bilan yoki "_"pastga belgi bilan boshlanishi zarur! O‘zgaruvchilar nomi raqam bilan boshlanmaydi! O‘zgaruvchi [a-z] harf bilan yoki "_"pastga belgi bilan boshlanib raqamlar ham aralashtirsa bo‘ladi. (_ism99, nom92 va h.k)

Python dasturlash tilida o‘zgaruvchilarning turini alohida e’lon qilish shart emas. O‘zgaruvchi turi, unga berilgan qiymat orqali avtomatik aniqlanadi.

Masalan:

Dastur kodi	Dastur natijasi		
x=5 **y='salom'**	O‘zgaruvchi nomi	O‘zgaruvchiga berilgan qiymat	O‘zgaruvchi turi

z=1.5	x	5	int
	y	salom	str
	z	1.5	float

Pythonda o‘zgaruvchilar turini type()funksiyasi orqali aniqlash mumkin.

Masalan:

Dastur kodi

```
x=5
y='salom'
z=1.5
print(" x o'garuvchisining turi:",type(x))
print(" y o'garuvchisining turi:",type(y))
print(" z o'garuvchisining turi:",type(z))
```

Dastur natijasi

```
x o'garuvchisining turi: <class 'int'>
y o'garuvchisining turi: <class 'str'>
z o'garuvchisining turi: <class 'float'>
```

Boshqa dasturlash tillaridagi kabi Python dasturlash tilida ham o‘zgaruvchi qiymati o‘zgarib turadi. Masalan:

	Dastur kodi	
1	`rangi='oq'`	***Izoh.*** Dastur kodi qatorma-qator bajariladi. Demak 1-qatorda turgan “rangi” nomli o‘zgaruvchi “oq” string turdagi qiymatni o‘zlashtirdi. 3-qatorda esa “rangi” nomli oldingi qiymat o‘rniga yangi “qizil” string turdagi qiymatni o‘zlashtirdi. Demak bundan
2	`rusmi='J E N T R A'`	
3	`rangi='qizil'`	
4	`print(rusmi,' ',rangi)`	

		ko‘rinadiki “rangli” nomli o‘zgaruvchining oxirgi qiymati “qizil”. Shu sababdan dastur natijasi **J E N T R A qizil**
	Dastur natijasi	
	J E N T R A qizil	

Bir nechta o‘zgaruvchiga qiymat o‘zlashtirish.

Pythonda bir nechta o‘zgaruvchiga mos holda qiymatlarni bir qatorning o‘zida o‘zlashtirish mumkin:

Dastur kodi

```
x, y, z = "Olma", "Banan", "Nok"
print('x =',x)
print('y =',y)
print('z =',z)
```

Dastur natijasi

```
x = Olma
y = Banan
z = Nok
```

Va aksincha, bir qiymatni bir nechta o‘zgaruvchiga o‘zlashtirish ham mumkin:

Dastur kodi

```
x = y = z = "Meva"
print('x = ',x)
print('y = ',y)
print('z = ',z)
```

Dastur natijasi

```
x =  Meva
y =  Meva
```

```
z =  Meva
```

O‘zgaruvchilarni xotiraga muloqat metodida kiritish uchun input()funksiyasidan foydalaniladi. input()funksiyasi o‘zgaruvchiga to‘g‘ridan-to‘g‘ri string turdagi qiymatni o‘zlashtiradi.

Masalan: a = 2 b = vali c = 2.5 o‘zgaruvchilar qiymatlari

Dastur kodi

```
a=input("a=")
b=input("b=")
c=input("c=")
print(a)
print(b)
print(c)
```

Dastur natijasi

```
2
vali
2.5
```

O'zgaruvchilarning xotiradagi holati

O'zgaruvchi nomi	O'zgaruvchi qiymati	O'zgaruvchi turi
a	2	str
b	vali	str
c	2.5	str

int yoki float turdagi o‘zgaruvchilarni e’lon qilish quyidagi tartibda amalga oshiriladi.

int turi: int(input())

float turi: float(input())

a = 1 b = 1.5 c = salom o'zgaruvchilar qiymatlari

Dastur kodi

```
a=int(input('a ='))
b=float(input('b ='))
c=input('c =')
```

```
print(a)
print(b)
print(c)
```

Dastur natijasi

```
1
1.5
salom
```

O'garuvchilarning xotiradagi holati

O'zgaruvchi nomi	O'zgaruvchi qiymati	O'zgaruvchi turi
a	1	int
b	1.5	float
c	salom	str

O‘zgaruvchilarni ekranga chiqarish

Pythonda o‘zgaruvchilarni yoki natijalarni ekranga chiqarish uchun print() funksiyasidan foydalaniladi. Dastur natijalari " " - *(qo‘shtirnoq)* yoki ''- *(apostrof)* ichiga olib yozilgan har qanday turdagi ma’lumot matn ko‘rinishida ekranga chiqariladi. Quyidagi jadvalda ma’lumotlarni matn ko‘rinishda chiqarishning turli-xil metodlari ko‘rsatilgan.

''- *(apostrof)* ichiga olib yozilgan holdagi ko'rinishi

Dastur kodi	Dastur natijasi
`print('Salom dunyo‘)`	Salom dunyo

" " - *(qo'shtirnoq)* ichiga olib yozilgan holdagi ko'rinishi

Dastur kodi	Dastur natijasi
`print("Salom dunyo")`	Salom dunyo

" " - *(qo'shtirnoq)* ichiga ''- *(apostrof)* qatnashgan so'z yozilgan holdagi ko'rinishi

Dastur kodi	Dastur natijasi

print("O'zingni angla")	O'zingni angla
''- *(apostrof)* ichiga " " - *(qo'shtirnoq)* qatnashgan so'z yozilgan holdagi ko'rinishi	
Dastur kodi	**Dastur natijasi**
print('"Python"') print('\"Python\" ')	"Python" "Python"
3 ta """ """ **-** *(qo'shtirnoq)* ichiga olib ***matnni*** qatorma-qator yozilgan holdagi ko'rinishi	
Dastur kodi	**Dastur natijasi**
print("""**Oz-oz o'rganib** **dono bo'lur** **Qatra-qatra yig'ilib** **daryo bo'lur**""")	Oz-oz o'rganib dono bo'lur Qatra-qatra yig'ilib daryo bo'lur
\n *(yangi satrga o'tish belgisi)* belgisi qatnashgan holdagi ko'rinishi ***(matnni qatorma-qator chiqaradi)***	
Dastur kodi	
print("Oz-oz o'rganib dono bo'lur \n Qatra-qatra yig'lib daryo bo'lur ")	
Dastur natijasi	
Oz-oz o'rganib dono bo'lur Qatra-qatra yig'lib daryo bo'lur	
''- *(apostrof)* ichiga **\'** belgisi qatnashgan holdagi ko'rinishi	
Dastur kodi	
print('**Oz-oz o\'rganib dono bo\'lur** ')	
Dastur natijasi	

Oz-oz o'rganib dono bo'lur
\t *(tabulyatsiya belgisi)* belgisi qatnashgan holdagi ko'rinishi
Dastur kodi

print('**\t Oz-oz o\'rganib \n dono bo\'lur** ')
# ikkinchi ko‘rinishi 	
print('**Oz-oz \t o\'rganib \t dono \t bo\'lur** ')

Dastur natijasi

Oz-oz o'rganib dono bo'lur	# ikkinchi ko'rinishi Oz-oz o'rganib dono bo'lur

1.3. Python operatorlari

Operatorlar o‘zgaruvchi va qiymatlar ustida amallar bajarish uchun ishlatiladi. Python operatorlari quyidagilar:

- ✓ Arifmetik operatorlar
- ✓ O‘zlashtirish operatorlar
- ✓ Taqqoslash operatorlari
- ✓ Mantiq operatorlari
- ✓ Aniqlash operatorlari
- ✓ A’zolik operatorlari
- ✓ Bitli operatorlar

Arifmetik operatorlar. Arifmetik operatorlar odatiy matematik amallarni bajarish uchun ishlatiladi:

+ Qo‘shish	x+y
- Ayirish	x-y
* Ko‘paytirish	x*y
/ Bo‘lish	x/y
% Qoldiqli bo‘lish	x%y

// Butunli bo‘lish x//y

O‘zlashtirish operatorlari. O‘zlashtirish operatorlari,

O‘zlashtirish operatori	Dastur kodi	Dastur natijasi

+=	`x = 10`
	`x+=4 # x=x+4 bilan teng kuchli`
	`print(x)`
		14

- =	`x = 10`
	`x-=4 # x=x-4 bilan teng kuchli`
	`print(x)`
		6

***=**	`x = 10`
	`x*=4 # x=x*4 bilan teng kuchli`
	`print(x)`
		40

/=	`x = 10`
	`x/=4 # x=x/4 bilan teng kuchli`
	`print(x)`
		2.5

%=	`x = 10`
	`x%=2 # x=x%2 bilan teng kuchli (qoldiq)`
	`print(x)`
		0

//=	`x = 10`
	`x/=4 # x=x//4 bilan teng kuchli (butun)`
	`print(x)`
		2

****=**	`x = 2`
	`x**=3 #x=x**3 bilan teng kuchli(daraja)`
	`print(x)`
		8

Taqqoslash operatorlari

Taqqoslash operatorlari qiymatlarni o‘zaro taqqoslash uchun ishlatiladi:

= = Teng x == y

!= Teng emas x != y

> Katta x > y

< Kichik x < y

>= Katta yoki teng x >= y

<= Kichik yoki teng x <= y

Mantiq operatorlari.

Mantiq operatorlar shartlarni birlashtirib ishlatish uchun kerak:

- and - Agar ikkala shart ham rost bo‘lsa, rost qiymat qaytaradi.

- or - Kamida bitta shart rost bo‘lsa ham rost qiymat qaytaradi.

- not - Shart qiymatini teskarisiga o‘zgartiradi, ya’ni rost bo‘lsa yolg‘on, yolg‘on bo‘lsa rost bo‘ladi.

==, !=, >= va <= operatorlarni yozganda oraga bo‘sh joy qo‘yib ketish sintaksis xatodir. Yani kompilyator dasturdagi xatoni ko‘rsatib beradi va uni tuzatilishini talab qiladi. Ushbu ikki belgili operatorlarning belgilarining joyini almashtirish, masalan <= ni =< qilib yozish ko‘p hollarda sintaksis hatolarga olib keladi. Gohida esa != ni =! deb yozganda sintaksis hato vujudga ham, bu mantiqiy hato bo‘ladi. Mantiqiy hatolarni kompilyator topa olmaydi. Lekin ular programma ishlash mantig‘ini o‘zgartirib yuboradi. Bu kabi hatolarni topish esa ancha mashaqqatli ishdir (! operatori mantiqiy inkordir). Yana boshqa hatolardan biri tenglik operatori (==) va tenglashtirish, qiymat berish operatorlarini (=) bir-biri bilan almashtirib qo‘yishdir. Bu ham juda ayanchli oqibatlarga olib keladi, chunki ushbu hato aksariyat hollarda mantiq hatolariga olib keladi.

Aniqlash operatorlari. Aniqlash operatorlari o‘zaro 2 ta obyektlarni solishtiradi. Bunda ularning o‘zaro qiymatlarini tengligi bo‘yicha emas, haqiqatdan ham ular bir xil obyekt ekanligi va bir xil xotira yo‘nalishiga egaekanligi bo‘yicha taqqoslanadi. Bu operatorlar 2 ta:

- is - Ikkala o‘zgaruvchi ham bir xil obyekt bo‘lsa rost, aks holda yolg‘on qiymat qaytaradi.

- is not - Obyektlar bir xil bo‘lmasa rost, aks holda yolg‘on qiymat qaytaradi.

Aniqlash operatorlari	Dastur kodi	Dastur natijasi

is	`x = ["olma", "banan"]`
	`y = ["olma", "banan"]`
	`z = x`
	`print(x is z)`
	`print(x is y)`
	`print(x == z)`
		True
		False
		True

is not	`x = ["olma", "banan"]`
	`y = ["olma", "banan"]`
	`z = x`
	`print(x is not z)`
	`print(x is not y)`
	`print(x != z)`
		False
		True
		False

A’zolik operatorlari. A’zolik operatorlari biror ketma-ketlik obyektga tegishli ekanligini tekshiradi:

- in - Belgilangan qiymat obyektda mavjud bo‘lsa, rost qiymat qaytaradi.

- not in - Belgilangan qiymat obyektda mavjud bo‘lmasa, rost qiymat qaytaradi.

x = ["audi", "mustang"]

print("audi" in x)

print("audi" not in x)

True

False

Bitli operatorlar. Bitli operatorlar ikkilik sanoq sistemasi bilan ishlashda kerak bo‘ladi:

- & (AND) - Ikkala bit ham 1 ga teng bo‘lsa, 1 ga o‘rnatiladi.

- | (OR) - Kamida bitta bit 1 ga teng bo‘lsa, ikkala bitni ham 1 ga o‘rnatadi.

- ^ (XOR) - Faqat bitta bit 1 ga teng bo‘lsa, ikkala bitni ham birga o‘rnatadi.

- ~ (NOT) - Barcha bitlarni invertlaydi (teskarisiga o‘zgartiradi)

- << - O‘ngdan chapga nollarni siljitib, chapdagi chetki bo‘laklarni tushirib yuboradi.

- >> - Chapdan o‘ngga bitlarning nusxalari kiritilib siljitib boriladi. O‘ngdagi chetki bitlar tushib qoladi.

1.1. Pythonda ma’lumot turlari

Python dasturlash tilida quyidagi ma’lumot turlari mavjud.

Matn turi: str

Raqam turlari: int, float, complex

Tarkib turlari: list, tuple, range

Xarita turi: dict

Turlarini o‘rnatish: set, frozenset

Boolean turi: bool

Ikkilik turlari: bytes, bytearray, memoryview

Python dasturlash tilida type()Funksiya yordamida istalgan ob’ekt ma’lumotlarini olishingiz mumkin. misol uchun: print(type(attribute))

Python dasturlash tilini o‘rnatish.

Agar siz biror GNU/Linux distributivini ishlatayotgan bo‘lsangiz ko‘p xollarda sizning tizimingizda python o‘rnatilgan bo‘ladi. Buni tekshirib ko‘rish uchun terminalingizdan quyidagi buyruqni ishga tushirib ko‘ring. python -V

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

Windows operatsion tizimiga o‘rnatish uchun www.python.org/downloads web sahifasiga o‘tamiz va u yerdan oxirgi python versiyasini yuklab olamiz. Pythonni o‘rnatish odatiy dasturlarni o‘rnatish kabi kechadi. Hech qanday qiyin joyi yo‘q.

Bugungi kunda dunyoga mashhur ko‘plab kompaniyalar NASA, Google, Yandex, CERN, Apple computer, Dream Works, kosmik teleskop institutlari Pythonni ishlatishadi. Dunyoning rivojlangan mamlakatlari AQSH (Koliforniya Universiteti, Florida Universiteti, Lova Universiteti, Massachustva Texnologiya Universiteti), Kanada (Toronto Universiteti, Alberto Universiteti), Buyuk Britaniya

(Oksford Universiteti), Fransiya, Rossiya, Avstraliya, Ispaniyaning universitet va kollejlarida o‘qitishda Python dasturlash tili qo‘llaniladi.

Pythonning o‘ziga xos jihatlaridan biri bu dasturni yozish davomida quyi darajadagi detallarni, misol uchun xotirani boshqarishni hisobga olishga hojat qolmaydi. Shuningdek dasturni yozish davomida ortiqcha kod yozishdan xalos bo‘linadi. Masalan: massiv elementlarini tartiblash misolini Paskal va Python dasturidagi talqinini taqqoslab solishtirib ko‘raylik.

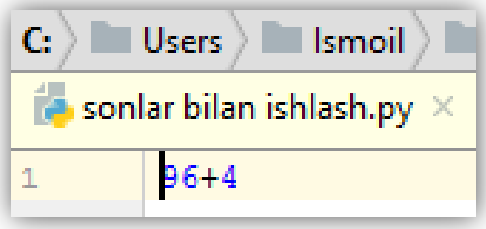
For i:=0 **to** N-1 **do**
For j:=N-2 **downto** i **do**
If A[j] > A[j+1] **then begin**
c:=A[j]; A[j]:=A[j+1];
A[j+1]:=c
end;

```
A.sort()
```

Shu bilan birga Pythonda dastur yozayotganda *begin-end, {}* yoki satr tugagani bildirish uchun *nuqtali vergul (;)* qo‘yilmaydi.

1.5. Pythonda sonlar bilan ishlash

Arifmetik amallar. Pythonda asosiy arifmetik amallar matematikadagi qoidasi bo‘yicha qo‘llaniladi.



Ikki sonning yig‘indisi chapdagi holda yozilsa python dasturlash muhitida bajariladi. Ularning natijasi xotiraga yoziladi. Ya’ni 100 ga teng bo‘ladi.

+ - qo‘shish amali:

Ikki sonni yig‘indisi

Dastur kodi	Dastur natijasi
`print(96 + 4)`	`100`

-- ayirish amali:

Ikki sonni ayirmasi

Dastur kodi	Dastur natijasi
`print(96 - 104)`	`-4`

*** - ko‘paytirish amali:**

Ikki sonni ko‘paytmasi

Dastur kodi	Dastur natijasi
`print(4 * 15)`	`60`

/ - bo‘lish amali:

Ikki sonni bo‘lish

Dastur kodi	Dastur natijasi
`print(12 / 5)`	`2.4`

// - butun qismli bo‘lish amali:

Ikki sonni bo‘linmasi (ushbu amal bo‘lish natijasining faqat butun qisminiqaytaradi, qoldiq qismi tashlab yuboriladi)

Dastur kodi	Dastur natijasi
`print(12 // 5)`	`2`

% - qoldiqli bo‘lish amali:

Ikki sonni bo‘linmasi (ushbu amal bo‘lish natijasining faqat qoldiq qismini qaytarib, butun qismi tashlab yuboriladi)

Dastur kodi	Dastur natijasi
`print(16 % 5)`	`1`

**** - darajaga ko‘tarish (oshirish) amali:**

x^y shaklidagi hisoblashlarda qo‘llaniladi

Dastur kodi	Dastur natijasi
`print(2 ** 2)`	`4`

Ifodada bir nechta arifmetik amallar ketma-ket kelgan bo‘lsa, ular 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:

Dastur kodi	Dastur natijasi
`son = 12//6 + 2 ** 4 * 2 - 5` `print(son)`	`29`

Bu yerda dastlab eng yuqori ustunlikga ega bo‘lgan amal – darajaga ko‘tarish amali bajariladi (2 ** 4 =16). Keyin ko‘paytma (16 * 2 = 32), butun qismli bo‘lish (12 // 6 = 2), qo‘shish (2 + 32 = 34) va ayirish (34 – 5 = 29) amallari bajariladi. Ifoda bajarilishi natijasida 29 soni konsol ekraniga chiqariladi.

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

Dastur kodi	Dastur natijasi

`son = 12//6 + 2 ** 4 * (2 - 5)` `print(son)`	`-46`

Natijada konsol ekraniga -46 soni chiqariladi.

Shuni alohida 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.

1.6. Pythonda satrlar ustida amallar

Satrlar – qo‘shtirnoq ichiga olishan *Unicode* kodidagi belgilar ketma-ketligi orqali ifodalanadi. Pythonda satrlar apostrof ( ) va qo‘shtirnoqlar ( ) orqali berilishi munkin. Uchta ketma-ket kelgan apostrof ham satrlarni ifodalashda ishlatiladi.

Python satr(string) operatorlari:

Operatorlar	Belgilar	Operator vazifasi
Satrlarni birlashtirish operatori	+	Ikki qatorni birlashtirish va yangi String yaratish.
Python string replikatsiya operatori	*	Bir qatorni bir necha marta takrorlash uchun.
in operatori	in	Belgilangan satrda belgi yoki butun pastki qator mavjud bo‘lsa, haqiqiy qiymatni qaytarish uchun, aks holda noto‘g‘ri qaytaradi.
not in operatori	not in	Belgilangan satrda belgi yoki butun pastki qator mavjud bo‘lmasa, haqiqiy qiymatni

		qaytarish uchun, aks holda noto‘g‘ri.
Python taqqoslash operatorlari	<,> <=,>=, ==,!=, <>	Ikki qatorni ASCII qiymatiga qarab solishtirish.

Pythonda satrlarni birlashtirish “ + ” amali orqali amalga oshiriladi va uning ishlatilishi quydagicha:

Dastur kodi	Dastur natijasi
i='Islom' print(i+'bek')	Islombek

Berilgan matnlar bir-biriga qo‘shilib ketmasligi uchun ular orasiga apostrof yoki qo‘shtirnoq belgilari ichiga probel orqali bo‘sh joy qoldirish kerak.

Dastur kodi	Dastur natijasi
i='Assalomu' j='alaykum' print(i+j) print(i+' '+j) print(i,j)	Assalomualaykum Assalomu alaykum Assalomu alaykum

Berilgan matnlarni boshqa turdagi o‘zgaruvchilar bilan birlashtirish talab etilsa, u holda ushbu o‘zgaruvchilarni *str()* funksiyasi orqali satr turiga o‘tkazish kerak.

Dastur kodi	Dastur natijasi
i='Axror' j=19 print(i,str(j),'yoshda')	Axror 19 yoshda

```
i='Axror'
j=78.8
print(i+' og\'rligi'+''+str(j))
```

```
Axror og'rligi 78.8
```

Python string *(*)* replikatsiya operatori:

Dastur kodi	Dastur natijasi
`i='Ismoil'` `print(i*3+' ')`	`Ismoil Ismoil Ismoil`

Satr uzunligi (satrdagi belgilar soni)ni aniqlash uchun *len()* funksiyasidan foydalaniladi:

Dastur kodi	Dastur natijasi
`ism = "Sa\'diya"` `print(len(ism))`	`7`

Dastur kodi	Dastur natijasi
`son = "11111"` `print(len(son))`	`5`

Dastur kodi	Dastur natijasi

`son = "11111"`
`ism = "Sa\'diya"`
`print(len(son+ism))`
	`12`

f-string :

Satrlarni birlashtirishda *f-string* metodidan foydalanish. *f-string* orqali bir nechta satrlarni birlashtirish mumkin.

Dastur kodi	Dastur natijasi

i="Ismoil" f="Xushboqov" i_f = f"{i} {f}" print(i_f)	Ismoil Xushboqov
i="Ismoil" f="Xushboqov" i_f = f"Mening do'stim {i}" print(i_f)	Mening do'stim Ismoil
i="Ismoil" f="Xushboqov" i_f=f"Mening do'stim {i}" f"\nUning familyasi{f}" print(i_f)	Mening do'stim Ismoil Uning familyasi Xushboqov

upper() metodi :

String turdagi o‘zgaruvchilarga o‘zlashtirilgan matnli ma’lumotlarni upper() metodi yordamida katta harflar ko‘rinishida tasvirlash mumkin. Bu metodni qo‘llaganimizda o‘zgaruvchi qiymati o‘zgarmaydi, upper() metodidan olishan natija o‘zgaradi.

Dastur kodi	Dastur natijasi
izoh="Men dasturchiman" print(izoh.upper())	MEN DASTURCHIMAN
Dastur kodi	
kitob="Algoritmlash" daftar="asos" kitob_daftar = f"{kitob} va dasturlash {daftar}lari" print(kitob_daftar.upper())	
Dastur natijasi	

```
ALGORITMLASH VA DASTURLASH ASOSLARI
```

Agar o‘zgaruvchilarga o‘zlashtirilgan matnli ma’lumotlarni xotirada katta harflar ko‘rinishda saqlamoqchi bo‘lsak upper()metodidan foydalanish quyidagicha bo‘ladi.

Dastur kodi

```
kitob="Algoritmlash"
daftar="asos"
kitob_daftar = f"{kitob} va dasturlash {daftar}lari"
kitob_daftar = kitob_daftar.upper()# bu yerda matnli ma’lumotlarni  xotirada katta harflar ko'rinishga o'tkazildi
print(kitob_daftar)
```

Dastur natijasi

```
ALGORITMLASH VA DASTURLASH ASOSLARI
```

lower() metodi :

Bu metod upper() metodiga teska ri ravishda matnlarni kichik harflarga o‘zgartiradi.

Dastur kodi

```
kitob="ALGORITMLASH"
daftar="ASOS"
kitob_daftar = f"{kitob} VA DASTURLASH {daftar}lari"
print(kitob_daftar.lower())
```

Dastur natijasi

```
algoritmlash va dasturlash asoslari
```

title() metodi :

Bu metod matndagi so‘zlarni birinchi harfini katta harflarga o‘zgartiradi.

Dastur kodi
ismi = "stiv" fami = "jobs" ismi_fami = f"{ismi} {fami}" print(ismi_fami.title())
Dastur natijasi
Stiv Jobs

Dastur kodi
ismi = "stiv jobs, franklin, ali" print(ismi.title())
Dastur natijasi
Stiv Jobs, Franklin, Ali

capitalize() metodi :

Bu metod matndagi birinchi so‘zni birinchi harfini katta harflarga o‘zgartiradi.

Dastur kodi
matn = "men tinchlikni sevaman" print(matn.capitalize())
Dastur natijasi
Men tinchlikni sevaman

Dastur kodi
matn = 'python o\'rganishga qulay bo\'lgan dasturlash tili' print(matn.capitalize())
Dastur natijasi
Python o‘rganishga qulay bo‘lgan dasturlash tili

strip() metodi :

Bu metod bo‘shliqlar bilan ishlashda foydalaniladi. Matnning chap tominida mavjud bo‘lgan bo‘shliqni o‘chirishda lstrip()metodidan foydalaniladi.

Dastur kodi
a=" chapdan bo’shliq" print(a.lstrip())
Dastur natijasi
chapdan bo’shliq

Matnning o‘ng tominida mavjud bo‘lgan bo‘shliqni o‘chirishda rstrip()metodidan foydalaniladi.

Dastur kodi
a="chapdan bo’shliq " print(a.rstrip())
Dastur natijasi
chapdan bo’shliq

Matnning chap va o‘ng tominida mavjud bo‘lgan bo‘shliqlarni o‘chirishda strip()metodidan foydalaniladi.

Dastur kodi
a=" chap va o'ngdan bo‘shliq " print(a.strip())
Dastur natijasi
Chap va o'ngdan bo‘shliq
Dastur kodi
a=" men " print(a.strip())
Dastur natijasi
men

Satrlarni almashtirish

Satrlardagi biror satr ostisini boshqa satr ostisiga almashtirish uchun **replace()metodi:**

- **replace(old, new)** – *old* satr ostisini *new* satr ostisiga almashtiradi;
- **replace(old, new, num)** – *num* parametri dastlabki nechta satr ostisini yangisi bilan almashtirish lozimligini ifodalaydi. Qolganlari o‘zgarishsiz qoladi.

replace(old, new):

Dastur kodi

`tug_yil = "31 05 1996"`
`#  bo‘sh joylarni nuqtalarga almashtirish`
`tug_yil_new = tug_yil.replace(" ", ".")`
`print(tug_yil_new)`

Dastur natijasi
`31.05.1996`

Dastur kodi

`tug_yil = "31 05 1996"`
`#  bo‘sh joylarni o'chirish`
`tug_yil_new = tug_yil.replace(" ", "")`
`print(tug_yil_new)`

Dastur natijasi
`31051996`

replace(old, new, num):

Dastur kodi

`tel_raqam = '+998-90-032-36-36'`
`#  chiziqchalarni bittasini o'chirish`
`tel_raqami = tel_raqam.replace("-", "",1)`

```
print(tel_raqami)
```

Dastur natijasi

```
+99890-032-36-36
```

1.7. Mantiqiy operatorlar

Murakkab shartli masalalarni yechishda mantiqiy operatorlardan keng foydalaniladi. Pythonda quyidagi mantiqiy operatorlar mavjud:

and (mantiqiy ko‘paytirish). Murakkab ifodadagi biror bir qism ifodani qiymati *False* bo‘lsa, ifodaning yakuniy qiymati *False* , aks holda *True* qiymat qaytaradi.

Masalan: Uch xonali son berilgan. Uning raqamlari o‘suvchi ketma-ketlik tashkil etishi tekshirilsin. a = 456

Dastur kodi	**Dastur natijasi**

`a=int(input('a=')) # a = 456`
`x = a // 100`
`b = a // 10 % 10`
`c = a % 10`
`z = x < b and b < c`
`print(z)`
	`True`

a = 452

Dastur kodi	**Dastur natijasi**

`a=int(input('a=')) # a = 452`
`x = a // 100`
`b = a // 10 % 10`
`c = a % 10`
	`False`

`z = x < b and b < c` `print(z)`	

or (mantiqiy qo‘shish). Agarda ifodadagi biror bir qism ifoda *True* qiymat qaytarsa, yakuniy natija ham *True*, aks holda *False* bo‘ladi.

Masalan: *a, b* butun sonlar berilgan. Ularning hech bo‘lmaganda bittasi musbat ekanligi tekshirilsin. a=-2 , b=4

Dastur kodi	Dastur natijasi

`a=int(input('a='))`
`#a=-2 b=4`
`b=int(input('b='))`
`z=a>0 or b>0`
`print(z)`
	`True`

Dastur kodi	Dastur natijasi

`a=int(input('a='))`
`#a=-20 b=-6`
`b=int(input('b='))`
`z=a>0 or b>0`
`print(z)`
	`False`

Uch xonali son berilgan. Uning raqamlari o‘suvchi yoki kamayuvchi ketma-ketlik tashkil etishi tekshirilsin. a = 136; a = 751;

Dastur kodi	Dastur natijasi

`a=int(input('a=')) # a = 751`
`x = a // 100`
`b = a // 10 % 10`
`c = a % 10`
	`True`

`z = (x < b and b < c) or (x >b and b>c)` `print(z)`	
Dastur kodi	Dastur natijasi

`a=int(input('a=')) # a = 136`
`x = a // 100`
`b = a // 10 % 10`
`c = a % 10`
`z =(x < b and b < c) or (x > b and b>c)`
`print(z)`
	`True`

not (mantiqiy inkor). Ifodaning qiymatini *True* bo‘lsa, natija *False* va aksincha.

Dastur kodi	Dastur natijasi
`x = False` `print(not x)`	`True`
Dastur kodi	Dastur natijasi
`x = False` `print(x)`	`False`

Agar bitta ifodada bir nechta mantiqiy operatorlar qatnashgan bo‘lsa, u holda ularning ustunligiga alohida e’tibor qatarish kerak. Dastlab ***not*** *operatori keyin* ***and*** *va eng so‘ngra* ***or*** *operatori bajariladi.*

Shuni alohida ta`kidlash kerarki, mantiqiy ifodalarda mantiqiy amallarningbajarilish ketma-ketligini qavslar ( ) yordamida o‘zgartirish mumkin.

Mustaqil bajarish uchun mashqlar:

1. a va b butun sonlari berilgan bo‘lsa, ularni (a>2 va b≤3) bo‘lgan hol uchun tekshirilsin.

2. a, b, c butun sonlari berilgan. Ular ($a \le b \le c$) holat uchun tekshirilsin.

3. a, b, c butun sonlar berilgan. b sonining, a va c sonlar orasida yotishi tekshirilsin.

4. a va b butun sonlar berilgan. Bu sonlardan biri toq ekanligi tekshirilsin.

5. a va b butun sonlar berilgan. Ularning bir xil juftlikka ega ekanligi tekshirilsin.

6. a, b, c butun sonlar berilgan. Faqat ulardan bittasi musbatligi tekshirilsin.

7. a, b, c butun sonlar berilgan. Ulardan faqat ikkitasi bir vaqtda musbat ekanligi tekshirilsin.

8. Butun musbat son berilgan. Uning toqligi va uch xonali ekanligi tekshirilsin.

9. Uch xonali son berilgan. Uning raqamlari o‘suvchi yoki kamayuvchi ketma-ketlik tashkil etishi tekshirilsin.

10. To‘rt xonali son berilgan. Uni chapdan o‘ngga va o‘ngdan chapga o‘qiganda bir xil o‘qilishi tekshirilsin.

11. x, y sonlari berilgan. Ularni koordinatalar deb hisoblab 2- yoki 3-chorakda yotishi tekshirilsin.

12. x, y sonlari berilgan. Ularni koordinatalar deb hisoblab 1- yoki 3-chorakda yotishi tekshirilsin.

13. a,b,c butun sonlar berilgan bo‘lib, ular uchburchakning tomonlarini tashkil etadi. Shu uchburchakning teng yonli ekanligi tekshirilsin.

14. a,b,c butun sonlar berilgan bo‘lib, ular uchburchakning tomonlarini tashkil etadi. Shu uchburchakning to‘g‘ri burchakli ekanligi tekshirilsin.

15. Uchta butun son berilgan. Shu sonlarning uchburchakning tomonlarini tashkil etishi tekshirilsin.

1.8. Pythonda shart operatori

if shart amali shart ifodalarda qo‘llanilib, uning natijasiga ko‘ra dastur bajarilishi u yoki bu yo‘lga yo‘naltiriladi. U quyidagi umumiy ko‘rinishga ega:

if *mantiqiy*

*ifoda**:***

ifodalar

elif *mantiqiy*

*ifoda**:***

ifodalar

else:

ifodalar

Eng sodda shaklda *if* kalit so‘zidan keyin mantiqiy ifoda keladi, agar bu mantiqiy ifoda rostni qaytarsa, u holda ko‘rsatmalarning keyingi bloki bajariladi, ularning har biri yangi satrdan boshlanishi kerak.

Dastur kodi

```
chet_tili = "fransuz"
if chet_tili == "fransuz":
    print(f"Salom {chet_tili} tili")
print("yakunlandi")
```

Dastur natijasi

```
Salom fransuz tili
yakunlandi
```

Bu holda chet_tili o‘zgaruvchisining qiymati " fransuz " bo‘lganligi sababli, *if* bloki bajariladi, unda faqat bitta ifoda mavjud - print(f"Salom {chet_tili} tili"). Natijada, konsol quyidagi qatorlarni ko‘rsatadi:

Salom fransuz tili

yakunlandi

Kodning oxirgi qatoriga e’tibor bering, unda "yakunlandi" xabari ko‘rsatiladi. U satr boshidan xat boshi bilan ajratilmagan, shuning uchun u *if* blokiga tegishli emas va *if* operatoridagi ifoda **False** qiymatini qaytarsa ham baribir bajariladi. Lekin unga ham xatboshi ajratsak *if* blokiga tegishli bo‘ladi. Ya’ni

Dastur kodi

```
chet_tili = "fransuz"
if chet_tili == "fransuz":
    print(f"Salom {chet_tili} tili")
    print("yakunlandi")
```

Dastur natijasi

```
Salom fransuz tili
yakunlandi
```

if shart operatorining eng sodda ko‘rinishida *if* kalit so‘zidan keyin mantiqiy ifoda yoziladi va ikki nuqta (:) qo‘yiladi. Keyingi qatordan amallar yoziladi. Shuni alohida ta’kidlash kerakki Pythonda boshqa tillardagi kabi if shart amalini tana qismini ifodalovchi maxsus belgilar mavjud emas (manasal c++, c# da *{,}* blok belgilari ishlatiladi). Shu sababli uning tana qismidagi ifodalar *if* kalit so‘ziga nisbatan bitta xat boshi (to‘rtta probel belgisi) belgisi tashlab yoziladi.

else bloki:

Agar to‘satdan *if* ifodasi **False** qiymatini qaytarsa, alternativ yechimni aniqlashimiz kerak bo‘lsa, biz *else* blokidan foydalanishimiz mumkin:

Dastur kodi

```
chet_tili = "o'zbek tili"
if chet_tili == "fransuz":
    print(f"Salom fransuz tili")
else:
    print(f"Salom {chet_tili}")
print('yakunlandi')
```

Dastur natijasi

```
Salom o'zbek tili
yakunlandi
```

if chet_tili == **"fransuz"** sharti "*True*" ni qaytarsa, u holda *if* bloki bajariladi, aks holda *else* bloki bajariladi va bu holda **if** chet_tili == **"fransuz"****False** qiymatini qaytarganligi sababli, *else* blokidagi ifoda bajariladi.

Bundan tashqari, *else* blokining ko‘rsatmalari ham satr boshidan xatboshi ajratish kerak. Masalan, yuqoridagi misolda print(**'yakunlandi'**)xat boshi bilan ajratilmagan, shuning uchun u *else* blokiga kiritilmagan chet_tili == **"fransuz"** sharti qanday bo‘lishidan qat’iy nazar bajariladi. Ya'ni, konsol quyidagi qatorlarni ko‘rsatadi:

```
Salom o'zbek tili
Yakunlandi
```

else blokida qator boshidan xatboshi ajratish kerak bo‘lgan bir nechta iboralar ham bo‘lishi mumkin:

elif:

Agar siz bir nechta muqobil shartlarni kiritishingiz kerak bo‘lsa, unda siz qo‘shimcha *elif* bloklaridan foydalanishingiz mumkin.

Dastur kodi

```
til = "o'zbek"
if til == "english":
    print("Hello")
    print("World")
elif til == "o'zbek":
    print("Salom")
    print("Dunyo")
else:
    print("Привет")
    print("мир")
```

Dastur natijasi

```
Salom
Dunyo
```

Birinchidan, Python *if* shartni tekshiradi. Agar rost bo‘lsa, *if* blokidagi operatorlar bajariladi. Agar bu shart **False** qiymatini qaytarsa, Python *elif* shartini tekshiradi.

Agar *elif*dan keyingi ifoda **True** bo‘lsa, u holda *elif* blokidagi gaplar bajariladi. Ammo agar u **False** ga teng bo‘lsa, *else* blokidagi ko‘rsatmalar bajariladi.

Ixtiyoriy ravishda turli shartlar uchun bir nechta *elif* bloklarini belgilashingiz mumkin. ***Misol uchun:***

Dastur kodi

```
til = "o'zbekcha"
if til == "english":
    print("Hello")
elif til == "german":
    print("Hallo")
elif til == "o'zbekcha":
    print("Assalomu-alaykum")
else:
    print("Привет")
```

Dastur natijasi

```
Assalomu-alaykum
```

if/elif/else ifodalarini *elif* va *else* bloklariga joylashtirish mumkin:

Dastur kodi

```
til = "o'zbek"
vaqt = "tong"
if til == "o'zbek":
    if vaqt == "tong":
        print("Xayrli tong")
    else:
        print("Xayrli kech")
```

```
else:
    if vaqt == "tong":
        print("Доброе утро")
    else:
        print("Добрый вечер")
```

Dastur natijasi

```
Xayrli tong
```

Misol: Uchta butun son berilgan. Ular orasidan musbatlari soni topilsin.

Dastur kodi

```
a,b,c = 10,-8,2
i=0
if a > 0:
    i+=1
if b > 0:
    i+=1
if c > 0:
    i+=1
print('musbat sonlar soni',i,'ta')
```

Dastur natijasi

```
musbat sonlar soni 2 ta
```

Ushbu misolda to‘liq bo‘lmagan (*bir matralik shartopertori*) shart operatoridan foydalanilgan. Bu yerda *if* xizmatchi so‘zidan keyin a>0 mantiqiy ifoda kelgan. Tana qismi bitta ifodadan tashkil topgan, ya’ni i+=1 va u *if* ga nisbatan bitta xat boshi tashlab yozilgan.

Misol: Uchta son berilgan. Ular orasidan eng kattasi topilsin.

Dastur kodi

```
son1,son2,son3 = -5,0,2
if (son1 > son2) and (son1 > son3):
```

```
    print(son1)
elif (son2 > son1) and (son2 > son3):
    print(son2)
else:
    print(son3)
```

Dastur natijasi

```
2
```

Ushbu misolda shart operatorining murakkab ko‘rinishi ifodalangan. Bu misolni yechimini topishda *and* mantiqiy ko‘paytirish operatoridan foydalanilgan.

Misol: Uchta butun son berilgan. Ulardan bittasi qolgan ikkitasidan ishorasi bilan farq qilsa, shu farq qiluvchi sonning tartib nomeri aniqlansin.

Dastur kodi

```
son1 = 37
son2 = 53
son3 = -25
if (son1 > 0 and son2 > 0):
    print("farqli son",son3,"tartib raqami" ,3)
if (son1 > 0 and son3 > 0):
    print("farqli son",son2,"tartib raqami" ,2)
if (son3 > 0 and son2 > 0):
    print("farqli son",son1,"tartib raqami" ,1)
if (son1 < 0 and son2 < 0):
    print("farqli son",son3,"tartib raqami" ,3)
if (son1 < 0 and son3 < 0):
    print("farqli son",son2,"tartib raqami" ,2)
if (son3 < 0 and son2 < 0):
    print("farqli son",son1,"tartib raqami" ,1)
```

Dastur natijasi

```
farqli son -25 tartib raqami 3
```

Ushbu dasturda to‘liq bo‘lmagan shart operatoridan foydalanildi. Misolning yechimini topish uchun 6 ta kombinatsiya ko‘rib chiqildi.

Misol. Uchta son berilgan. Ularning ikkita kattasining yig‘indisi chiqarilsin.

Dastur kodi

```
son1 = -37
son2 = -3
son3 = 2
if son1 > son2 > son3:    # if (son1 > son2 and
son2 > son3): bilan teng kuchli
    print(son1 + son2)
if son2 > son1 > son3:
    print(son1 + son2)
if son3 > son1 > son2:
    print(son1 + son3)
if son1 > son3 > son2:
    print(son1 + son3)
if son2 > son3 > son1:
    print(son2 + son3)
if son3 > son2 > son1:
    print(son2 + son3)
```

Dastur natijasi

```
-1
```

Mustaqil bajarish uchun mashqlar:

1. Butun son berilgan. Agar u musbat bo‘lsa unga 1 qo‘shilsin, aks holda o‘zgarishsiz qoldirilsin.Olishan son chiqarilsin.
2. Butun son berilgan. Agar u manfiy bo‘lsa unga 1 qo‘shilsin, aks holda 2 ayirib tashlansin. Olishan son chiqarilsin.
3. Butun son berilgan. Agar u manfiy bo‘lsa 2 ayirilsin, 0 ga teng bo‘lsa 10 bilan almashtirilsin.Olishan son chiqarilsin.

4. Uchta butun son berilgan. Ular orasidan musbatlari va manfiylari soni topilsin.
5. Uchta butun son berilgan. Ular orasidan musbatlari va manfiylari soni topilsin.
6. Ikkita haqiqiy turga tegishli *a* va *b* o‘zgaruvchilari berilgan. Ularning qiymatlari quyidagicha qayta taqsimlansin: *a* ga kichigi *b* ga kattasi, *a* va *b* larning yangi qiymatlari chiqarilsin.
7. Ikkita butun tipga tegishli *a* va *b* o‘zgaruvchilar berilgan. Agar ularning qiymatlari teng bo‘lmasa har bir o‘zgaruvchiga qiymatlar yig‘indisi berilsin, aks holda har bir o‘zgaruvchiga 0 qiymat qiymatlansin. O‘zgaruvchilarning natijaviy qiymatlari chiqarilsin.
8. Uchta son berilgan. Ular orasidan eng kichigi topilsin.
9. Uchta son berilgan. Ular orasidan o‘rtachasi topilsin.
10. Uchta son berilgan. Ularning ikkita kattasining yig‘indisi chiqarilsin.
11. Haqiqiy tipga tegishli uchta *a*, *b*, *c* o‘zgaruvchilar berilgan. Agar o‘zgaruvchilarning qiymatlari o‘sish tartibida joylashgan bo‘lsa, ularning qiymatlari ikki marta oshirilsin, aks holda har bir o‘zgaruvchining qiymati teskarisi bilan almashtirilsin. O‘zgaruvchilarning natijaviy qiymatlari chiqarilsin.
12. To‘rtta butun son berilgan. Ulardan bittasi qolgan uchtasidan farq qilsa (juft toqligi bilan) bu sonning tartib nomeri chiqarilsin.
13. Tekislikda butun sonlardan iborat koordinataga ega nuqta joylashgan. Agar u koordinata boshi bilan ustma-ust tushsa 0, *Ox* o‘qida joylashgan bo‘lsa 1, *Oy* o‘qida joylashgan bo‘lsa 2, aks holda 3, qiymat chiqarilsin.
14. *Ox* va *Oy* o‘qlarida yotmaydigan nuqta koordinatalari berilgan. Uning qaysi chorakka tegishli ekanligi aniqlansin.
15. To‘g‘ri to‘rtburchakning 3 ta uchi butun sonlardan iborat koordinatalar bilan berilgan. Shu uchlar orasidagi tomonlar koordinata o‘qlariga parallel bo‘lsa, to‘rtburchakning to‘rtinchi uchining koordinatasi topilsin.

1.9.While sikli

while sikli ba’zi bir shartning haqiqatini tekshiradi va agar shart rost bo‘lsa, u holda siklning ko‘rsatmalarini bajaradi.

While (shart) :

ifoda

ifodalar

while kalit so‘zidan keyin shartli ifoda keladi, bu ifoda *True* ni qaytarsa, keyingi ko‘rsatmalar bloki bajariladi. *while* sikliga tegishli barcha ifodalar keyingi qatorlarda joylashgan va *while* kalit so‘zining xatboshidan boshlanishi kerak.

Dastur kodi

```
raqam = 1
while raqam < 5:
    print(f" sana = {sana}")
    raqam+= 1
print("Dastur tugadi")
```

Dastur natijasi

```
 raqam = 1
 raqam = 2
 raqam = 3
 raqam = 4
 Dastur tugadi
```

Bunday holda, raqam o‘zgaruvchisi 5 dan kichik bo‘lsa, while sikli ishlaydi.

Dastur kodi

```
raqam = 1
while raqam < 5:
    print(f" sana = {sana}")
    raqam+= 1
print("Dastur tugadi")
```

Dastur natijasi
raqam = 1 raqam = 2 raqam = 3 raqam = 4 Dastur tugadi

E’tibor bering, ular while operatorining boshidan - xatboshi ajratadi. Bu Pythonga ular siklga tegishli ekanligini aniqlash imkonini beradi.

Shuni ham yodda tutingki, oxirgi qatordagi print(**"Dastur tugadi"**) ga xatboshi ajratilmagan, shuning uchun u *while* siklining bir qismi emas. Ya’ni *while* shart operatorining tana qismigi kirmaydi.

Siklning butun jarayonini quyidagicha ifodalash mumkin:

1. *raqam* o‘zgaruvchisining qiymati tekshiriladi - u 5 dan kattami. Va o‘zgaruvchi boshida 1 ga teng bo‘lganligi sababli, bu shart **True** ni qaytaradi, shuning uchun sikl ko‘rsatmalari bajariladi.

While shart operatorining tanasida joylashgan ifodalar *raqam = 1* ni konsolga bosib chiqaradi.Va keyin o‘zgaruvchining sonining qiymati bittaga oshiriladi - endi u 2 ga teng bo‘ladi. sikl ko‘rsatmalari blokini bir marta bajarish iteratsiya deb ataladi. Ya'ni, shu tarzda, birinchi takrorlash siklda amalga oshiriladi.

2.raqam < 5 sharti yana tekshiriladi. Bu hali ham **True**, chunki *raqam = 2*, shuning uchun sikl operatorlari bajariladi. While shart operatorining tanasida joylashgan ifodalar raqam = 2 ni konsolga chop etadi.Va keyin o‘zgaruvchan raqamning qiymati yana bittaga oshiriladi - endi u 3 ga teng. Shunday qilib, ikkinchi takrorlash amalga oshiriladi.

3. *raqam < 5* sharti yana tekshiriladi. Bu hali ham **True**, chunki *raqam = 3*, shuning uchun sikl ko‘rsatmalari bajariladi. While shart operatorining tanasida joylashgan ifodalar *raqam = 3* ni konsolga chop etadi.Va keyin o‘zgaruvchan raqamning qiymati yana bittaga oshiriladi - endi u 4 ga teng. Shunday qilib, uchinchi takrorlash amalga oshiriladi.

4. *raqam* < 5 sharti yana tekshiriladi. Bu hali ham To‘g‘ri, chunki *raqam* = *4*, shuning uchun sikl operatorlari bajariladi. While shart operatorining tanasida joylashgan ifodalar *raqam* = *4* ni konsolga chop etadi.Va keyin o‘zgaruvchan raqamning qiymati yana bittaga oshiriladi - endi u 5 ga teng. Shunday qilib, uchinchi takrorlash amalga oshiriladi.

5. *raqam* < 5 sharti yana tekshiriladi. Lekin endi u **False**, chunki *raqam* = 5, shuning uchun sikl tugadi. Barcha sikl tugadi. Bundan tashqari, sikldan keyin aniqlangan harakatlar allaqachon bajarilgan. Shunday qilib, bu sikl to‘rtta o‘tish yoki to‘rtta takrorlashni amalga oshiradi.

Shuningdek, while sikli uchun qo‘shimcha *else* blokidan foydalanish mumkin.

Dastur kodi

```
t_raqam = 1
while t_raqam < 5:
    print(f"raqam = {t_raqam}")
    t_raqam += 1
else:
    print(f"raqam = {t_raqam}. Sikl tugallandi")
print("Dastur tugadi")
```

Dastur natijasi

```
raqam = 1
raqam = 2
raqam = 3
raqam = 4
raqamr = 5. Sikl tugallandi
Dastur tugadi
```

Ya’ni, bu holda birinchi navbatda shart tekshiriladi va while operatorlari bajariladi. Keyin shart **Fals**e ga aylanganda *else* blokidagi operatorlar bajariladi.

E’tibor bering, *else* blokidagi ifodalar ham sikl ifodasining boshidan xatboshi ajratadi. Natijada:

```
raqam = 1
raqam = 2
raqam = 3
raqam = 4
raqamr = 5. Sikl tugallandi
Dastur tugadi
```

Bunday holda raqam <5 sharti dastlab **False** bo‘ladi, shuning uchun sikl takrorlanmaydi va to‘g‘ridan-to‘g‘ri *else* blokiga o‘tadi.

Misol. *n* butun son berilgan. Butunga bo‘lish va qoldiqini aniqlash operatsiyalaridan foydalanib *n* sonida “2” raqami borligi aniqlansin. Agar bor bo‘lsa “*true*” aks holda “*false*” chiqarilsin.

Dastur kodi

```
"""
Created on Tue Mar  1 21:49:50 2022
@author: Ismoil
"""
n=int(input('n='))
z=False
while(n>0):
    x=n%10
    n=n//10
    z=z or x==2
print(z)
```

Dastur natijasi

```
n=1256
True
```

```
n=1556

False
```

Misol. n(n>1) butun son berilgan. fk Fibonachchi sonlar ketma-ketligi quyidagicha aniqlansa, $f_1=1$, $f_2=1$, $f_k=f_{k-2}+f_{k-1}$ k=3, 4…, n sonining Fibonachchi sonlar ketma-ketligida uchrashi tekshirilsin. Agar n soni uchrasa *true*, aks holda *false* chiqarilsin.

Dastur kodi

```
"""
Created on Tue Mar  3 21:30:50 2022
@author: Ismoil
"""
f1=1
f2=1
z=False
fk=0
n=int(input('n = '))
while fk<=n:
    fk=f1+f2
    f1=f2
    f2=fk
    z=z or fk==n
print(z)
```

Dastur natijasi

n = 5 True	n = 9 False

Mustaqil bajarish uchun mashqlar:

1. n(n>1) butun son berilgan. Agar u tub son bo‘lsa true, aks holda false chiqarilsin.
2. a va b butun musbat sonlari berilgan. Evklid algoritmidan foydalanib ularning eng katta umumiy bo‘luvchisi topilsin (EKUB). Agar b≠0 bo‘lsa EKUB(a,b)=EKUB(b,a mod b) aks holda EKUB(a,0)=a.

3. n(n>1) butun son berilgan. fk Fibonachchi sonlar ketma-ketligi quyidagicha aniqlansa, $f_1=1$, $f_2=1$, $f_k=f_{k-2}+f_{k-1}$ k=3, 4…,n sonining Fibonachchi sonlar ketma-ketligida uchrashi tekshirilsin. Agar n soni uchrasa true, aks holda false chiqarilsin.
4. n(n>1) butun son berilgan. fk Fibonachchi sonlar ketma-ketligi $f_1=1$, $f_2=1$, $f_k=f_{k-2}+f_{k-1}$ uchun n dan katta 1-Fibonachchi soni topilsin.
5. n(n>1) butun son berilgan. fk Fibonachchi sonlar ketma-ketligi quyidagicha aniqlanadi. $f_1=1$,$f_2=1$, $f_k=f_{k-2}+f_{k-1}$ k=3, 4….n=f_k bo‘lsa f_{k+1} va f_{k-1}(oldingi va keyingi) Fibonachchi sonlari chiqarilsin, aks holda 0 chiqarilsin.
6. n(n>1) butun son berilgan. fk Fibonachchi sonlar ketma-ketligi quyidagicha aniqlanadi. $f_1=1$,$f_2=1$, $f_k=f_{k-2}+f_{k-1}$ k=3, 4…n=f_k bo‘lsa, k (Fibonachchi sonining tartib nomeri) chiqarilsin, aks holda 0 chiqarilsin.
7. n(n>0) butun son berilgan. Agar u 3 sonining darajasidan iborat bo‘lsa true, aks holda false chiqarilsin.
8. n(n>0) butun son berilgan. U 2 ning biror bir darajasidan iborat bo‘lsa n=2k, shu darajaning ko‘rsatkichi k butun soni topilsin.
9. n(n>0) butun son berilgan. n ikki factorial hisoblansin. Bu yerda n!!=n(n-2)(n-4)… (oxirgi ko‘paytuvchi agar n-juft bo‘lsa 2 ga, toq bo‘lsa 1 ga teng.) Butun tip diapozonidan oshib ketishining oldini olish uchun bu ko‘paytma natija haqiqiy tipli o‘zgaruvchiga qiymatlanadi.
10. n(n>0) butun son berilgan. Kvadratdan ildiz chiqarish formulasidan foydalanmay kvadrati ndan katta eng kichik k soni topilsin. ($k^2>n$)
11. n butun son berilgan. Kvadratdan ildiz chiqarish formulasidan foydalanmay kvadrati n dan katta bo‘lmagan eng katta butun k soni topilsin. ($k^2\leq n$)
12. n(n>1) butun son berilgan. $3^k>n$ tengsizlik o‘rinli bo‘ladigan eng kichik k butun soni topilsin.
13. n(n>1) butun son berilgan. $3^k<n$ tengsizlik o‘rinli bo‘ladigan eng katta k butun soni topilsin.
14. n(n>1) butun son berilgan. 1+2+…+k yig‘indining n dan katta yoki teng bo‘lishini ta`minlaydigan eng kichik k butun soni va yig‘indining qiymati

chiqarilsin. (1+2+…+ k ≥ n) n(n>1) butun son berilgan. 1+2+…+k yig‘indining n dan kichik yoki teng bo‘lishini ta’minlaydigan eng katta k butun son va yig‘indining qiymati chiqarilsin. (1+2+…+k ≤ n)

15. Bankdagi boshlang‘ich qo‘yilma summa 1000 so‘m bo‘lsa va u har oyda p foiz ko‘payib borsa (p-haqiqiy son, 0<p<25) necha oydan so‘ng qo‘yilma 1100 so‘mdan oshishi(o‘tgan oylar soni) k, hamda qo‘yilmaning oxirgi miqdori s(haqiqiy son) chop etilsin.

1.10. For sikli

Yana bir takrorlash operatori – *for* operatori hisoblanadi. For sikli ketma-ketlikni takrorlash uchun ishlatiladi (ya'ni ro‘yxat, kortej, lug‘at, to‘plam yoki satr).Bu boshqa dasturlash tillaridagi *for* kalit so‘ziga o‘xshamaydi va boshqa ob’ektga yo‘naltirilgan dasturlash tillarida bo‘lgani kabi iterator metodi kabi ishlaydi. *for* takrorlash operatori qandaydir sonlar kollektsiyasidagi har bir son uchun chaqiriladi. Sonlar kollektsiyasi *range*() funksiyasi, *list*() funksiyasi yoki [,] qavslarda foydalanuvchi tomonidan shakllantirilgan ro‘yxatlar orqali hosil qilinadi. Quyida *for* takrorlash operatorining formal aniqlanishi keltirilgan:

for int_var *in* funksiya_range:
    ifodalar

for kalit so‘zidan keyin *int_var* o‘zgaruvchisi (o‘zgaruvchi nomi ixtiyoriy bo‘lishi mumkin) keladi va u faqat butun turdagi qiymatlar qabul qiladi, undan keyin *in* kalit so‘zi (*in* operatori) va *range* funksiyasi chaqirilgan va oxirida “:” belgisi bilan takrorlash operatori asosiy qismi tugaydi. *for* takrorlash operatorining tana qismi bir yoki bir nechta instruktsiyalardan tashkil topishi mumkin va ular asosiy qismga nisbatan bitta xat boshi tashlab yoziladi.

Takrorlash operatori bajarilganda *range()* funksiyasi hosil qilgan sonlar kollektsiyasidan sonlar ketma-ket *int_var* o‘zgaruvchisiga uzatiladi. Sikl bo‘yicha barcha sonlar ketma-ket o‘tib bo‘lingandan keyin takrorlash operatori o‘z ishini tugatadi. Hatto satrlar ham takrorlanadigan ob’ektlar bo‘lib, ular belgilar ketma-ketligini o‘z ichiga oladi:

Misol. *nom* o‘zgaruvchisiga o‘zlashtirilgan “Python” so‘zini *for* operatori

yordamida harflar koʻrinishida chop qiling.

Dastur kodi

```
nom = "Python"
for i in nom:
    print(i)
```

1-qadam. *for* operatoridan keyin kelgan i oʻzgaruvchisi nom oʻzgaruvchisining 0-chi oʻrnida turgan “P” harfini oʻziga yuklaydi va chop qiladi.

2-qadam. *for* operatoridan keyin kelgan i oʻzgaruvchisi nom oʻzgaruvchisining 1-chi oʻrnida turgan “y” harfini oʻziga yuklaydi va chop qiladi.

3-qadam. *for* operatoridan keyin kelgan i oʻzgaruvchisi nom oʻzgaruvchisining 2-chi oʻrnida turgan “t” harfini oʻziga yuklaydi va chop qiladi.

4-qadam. *for* operatoridan keyin kelgan i oʻzgaruvchisi nom oʻzgaruvchisining 3-chi oʻrnida turgan “h” harfini oʻziga yuklaydi va chop qiladi.

5-qadam. *for* operatoridan keyin kelgan i oʻzgaruvchisi nom oʻzgaruvchisining 4-chi oʻrnida turgan “o” harfini oʻziga yuklaydi va chop qiladi.

6-qadam. *for* operatoridan keyin kelgan i oʻzgaruvchisi nom oʻzgaruvchisining 5-chi oʻrnida turgan “n” harfini oʻziga yuklaydi va chop qiladi.

Bu jarayon *nom* oʻzgaruvchisidagi birorta elementi qolmgauncha davom etadi. Natijada:

Dastur natijasi

```
P
y
t
h
o
n
```

Sikldan chiqish. *break* va *continue* operatorlari. Sikllarni boshqarish uchun *break* va *continue* kabi maxsus operatorlardan foydalaniladi. *break* operatori

sikldan chiqish uchun ishlatiladi. *continue* operatori siklning navbatdagi iteratsiyasiga o‘tish uchun ishlatiladi.

Odatda *break* operatori siklda shart operatorlari bilan birga qo‘llaniladi, masalan. ***Break*** operatori yordamida biz siklni barcha elementlardan o‘tmasdan oldin to‘xtatishimiz mumkin:

Misol. “Python” so‘zidagi “t” harfigacha bo‘lgan harflarni chop qiling.

Dastur kodi

```
nom = "Python"
for i in nom:
    print(i)
    if i == 't':
        break
```

Dastur natijasi

```
P
y
t
```

Continue:

Misol. “Python” so‘zidagi “t” harfini qoldirib chop qiling.

Dastur kodi

```
nom = "Python"
for i in nom:
    if i == 't':
        continue
    print(i)
```

Dastur natijasi

```
P
y
h
```

```
o
n
```

***range* funksiyasi**. *range* funksiyasining quyidagi shakllari mavjud:

range(stop) – *0* dan *stop* gacha (*stop* kirmaydi) bo‘lgan barcha sonlarni qaytaradi;

range(start, stop) – *start(kiradi)* dan *stop (kirmaydi)* gacha bo‘lgan barcha butun sonlarni qaytaradi;

range(start, stop, step) – *start (kiradi)* dan *stop (kirmaydi)* gacha bo‘lgan barcha butun sonlar *step* qadam bilan hosil qilinadi va qaytaradi.

range(stop) – 0 dan *stop* gacha (*stop* kirmaydi) bo'lgan barcha sonlarni qaytaradi;

Dastur kodi	Dastur natijasi

`son = 10`
`for i in range(son): """`
`bu yerda i ning`
`dastlabki qiymati 0 dan`
`boshlanadi`
`va i ning qiymati 1 ga`
`oshib boradi"""`
`print (i,end(' '))`
`""" bu yerda i ning`
`qiymati qator shaklida`
`chop etiladi"""`
	`0 1 2 3 4 5 6 7 8 9`

range(start, stop) – start(kiradi) dan *stop (kirmaydi)* gacha bo‘lgan barcha butun sonlarni qaytaradi;

Dastur kodi	Dastur natijasi
`son = 10`	`1 2 3 4 5 6 7 8 9`

```
for i in range(1,son):
""" bu yerda i ning dastlabki qiymati 1 dan boshlanadi
  va i ning qiymati 1 ga oshib boradi"""
    print (i,end(' ')) """ bu yerda i ning qiymati qator shaklida chop etiladi"""
```

range(start, stop, step) – *start (kiradi)* dan *stop (kirmaydi)* gacha bo'lgan barcha butun sonlar *step* qadam bilan hosil qilinadi va qaytaradi.

Dastur kodi	**Dastur natijasi**

`son = 10`
`for i in range(1,son,2):  """ bu yerda i ning dastlabki 1 dan boshlanadi`
`  va i ning qiymati 2 ga oshib boradi"""`
`  print (i,end=(' '))  """ bu yerda i ning qiymati qator shaklida chop etiladi"""`
	1 3 5 7 9

Ichma-ich joylashgan sikllar. Biror bir takrorlash operatori tanasida boshqa takrorlash operatorining ishlatilishiga ichma-ich joylashgan sikl deyiladi.

Misol. 1 ni bir marta 2 ni ikki marta va.hakazo n ni n marta chiqaradigan dastur tuzing.

Dastur kodi	**Dastur natijasi**

`for i in range(9):`
`  for j in range(i):`
`  print(i, end=(' '))`
	1
	2 2
	3 3 3

`    print(" ")`	`4 4 4 4`
	`5 5 5 5 5`
	`6 6 6 6 6 6`
	`7 7 7 7 7 7 7`
	`8 8 8 8 8 8 8 8`

For siklida ham *else* blokidan foydalanish mumkin. Masalan :

Dastur kodi	Dastur natijasi

`for i in range(1,4):`
`    print(i)`
`else:`
`    print('sikl tugadi')`
	`1`
	`2`
	`3`
	`sikl tugadi`

Misol. *n (n>1)* butun son berilgan. Butun tipli *fk* fibonachchi sonlar ketma-ketligi quyidagicha aniqlanadi. *f1=0; f2=1; fk=fk-2+fk-1,k=3,4,..f1, f2,…,fn* elementlari chiqarilsin.

Dastur kodi	Dastur natijasi

`n=int(input('n='))`
`f1=0`
`f2=1`
`for i in range(1,n+1):`
`    fk=f1+f2`
`    f1=f2`
`    f2=fk    print(fk," ")`
	`n=4`
	`1`
	`2`
	`3`
	`5`

Misol *n (n>3)* butun son berilgan. *ak* butun sonli ketma-ketlik quyidagicha aniqlanadi. $a_1=1;\ a_2=2;\ a_3=3\ a_k=a_{k-1}+a_{k-2}-2a_{k-3},\ k=4,5,...a_1,\ a_2,\ ...,\ a_n$ elementlari chiqarilsin.

Dastur kodi	Dastur natijasi

```
"""
Created on Tue Mar 1
16:14:53 2022
@author: Ismoil
"""
n=int(input('n='))
a1=1
a2=2
a3=3
print(a1,a2,a3,end=' ')
for i in range(3,n):
    ak=a3+a2-2*a1
    a1=a2
    a2=a3
    a3=ak
    print(ak,end=' ')
```

```
n=5
1 2 3 3 2
```

Mustaqil bajarish uchun mashqlar:

1. 1 *kg* konfetning narxi berilgan. 0,1, 0,2, …, 1 *kg* konfetning bahosi chiqarilsin.
2. 1 *kg* konfetning narxi berilgan. 1,2, 1,4, …, 2 *kg* konfetning bahosi chiqarilsin.
3. 2 ta *a* va *b* butun sonlar berilgan. (*a*<*b*) *a* dan *b* gacha bo‘lgan butun sonlar yig‘indisi topilsin.
4. 2 ta *a* va *b* butun sonlar berilgan. (*a*<*b*) *a* dan *b* gacha bo‘lgan sonlarning ko‘paytmasi topilsin.
5. *a* va *b* butun sonlar berilgan. (*a*<*b*) *a* dan *b* gacha bo‘lgan sonlarning kvadratlar yig‘indisi topilsin.
6. *n*(*n*>*0*) butun soni berilgan *n* (Yig‘indi haqiqiy son). Yig‘indi hisoblansin.

7. *n* butun soni berilgan $n^3+(n+1)^3+(n+2)^3...+(2n)^3$. (Yig‘indi butun son). Yig‘indi hisoblansin.
8. *n* butun soni berilgan 1,1·1,2·1.3·...{1,*n*}(*n* ta ko‘paytuvchi). Ko‘paytma hisoblansin.
9. *n*(*n>0*) butun soni berilgan. *1,1-1,2+1,3-...* Ifodaning qiymati topilsin. Shart operatori qo‘llanilmasin.
10. *n(n>0)* butun soni berilgan. Quyidagi formuladan foydalanib berilgan sonning kvadrati hisoblansin: $n^2=1+3+5+...+(2n-1)$. Har bir qadamdagi yig‘indi chiqarilsin(natijada 1 dan *n* gacha bo‘lgan butun sonlarning kvadrati chiqadi).
11. *a* va *n* sonlari berilgan. Bitta siklдan foydalanib *a* sonining 1 dan *n* gacha bo‘lgan darajalari chiqarilsin.
12. *a* va *n* sonlari berilgan. $1+ a + a^2 + a^3 + ... + a^n$. Bitta siklдan foydalanib yig‘indi hisoblansin.
13. *a* va *n* sonlari berilgan.$1-a+a^2-a^3+...+(-1)^n a^n$. Bitta siklдan foydalanib ifodaning qiymati hisoblansin. Hisoblashda shart operatoridan foydalanilmasin.
14. *n(n>0)* butun son berilgan. *n*!=1·2·...·*n* (*n*-faktorial) ko‘paytma hisoblansin. Ifodaning natijasi butun sonlar diapazonidan chiqib ketishi mumkinligi hisobga olinib, natijani saqlash uchun haqiqiy tipli o‘zgaruvchidan foydalanilsin va natija ham haqiqiy son ko‘rinishida chiqarilsin.
15. *n* butun soni berilgan (*n>0*). *1!+2!+...+n!.* Bitta siklдan foydalanib yig‘indi hisoblansin.

1.11. Funksiya

Funksiya kod bloki bo‘lib, u faqat chaqirilganda ishlaydi. Funksiyaning yozilish qoidasi quyidagicha:

```
def funksiya_nomi([parametrlar ro‘yxati]):
    amallar
```

Funksiya yaratish va chaqirish.

Pythonda funksiya *def* kalit so‘zi yordamida aniqlanadi . Pythonda funksiya *def* kalit so‘zi yordamida aniqlanadi .

Dastur kodi

```
# funksiyani yaratish quyidagiicha:
def funksiya():
    print('Salom Python muxlislari')
# funksiyani chaqirish
funksiya()
```

Dastur natijasi

```
Salom Python muxlislari
```

*Argumentlar***.**

Axborot funksiyalarga argument sifatida uzatilishi mumkin. Argumentlar funksiya nomidan keyin qavslar ichida ko‘rsatiladi. Xohlagancha argumentlar qo‘shish mumkin, ularni vergul bilan ajratib qo‘yish shart. *Argumentlar* soni odatiy bo‘lib, funksiya to‘g‘ri argumentlar soni bilan chaqirilishi kerak. Ya’ni, agar sizning funksiyangiz 2 ta argumentni qabul qilsa, siz funksiyani 2 ta argument bilan chaqirishingiz kerak, ko‘p emas va kam emas. Agar argumentlar soni noma’lum bo‘lsa, parametr nomidan oldin “ * ” belgisi qo‘yiladi.

Quyidagi misolda bitta argument (ismi) bilan funksiya mavjud. Funksiya chaqirilganda, to‘liq ismini chop etish uchun funksiya ichida ishlatiladigan ismni o‘tkazamiz:

Dastur kodi

```
def funksiya(ismi):
    print(ismi + "boy")
funksiya("Ali")
funksiya("Afvzal")
funksiya("Ahmad")
```

Dastur natijasi

```
Aliboy
Afvzalboy
Ahmadboy
```

Qiymat qaytarish ***(return)***

Funksiyaga qiymat qaytarishiga ruxsat berish uchun quyidagi return funksiyasidan foydalanamiz:

Misol. Sonning kvadratini hisoblaydigan funksiya yasang.

Dastur kodi

```
# funksiyani yaratishni 1-metodi
def kvadrati(x):
    return '1-metod',x*x
print(kvadrati(10))
# funksiyani yaratishni 2-metodi
def kvadrati(x):
    print('2-metod',x*x)
print(kvadrati(10))
```

Dastur natijasi

```
('1-metod', 100)
2-metod 100
None
```

Kvadrati() nomli funksiyani yasashning ikki xil metodini ko‘rdik.

1-metodda return funksiyasini qatnashtirib yasadik. Albatta bunda funksiyadan foydalanganda uning natijalarini ko‘rishda print funksiyasidanfoydalanamiz.

2-metodda print funksiyasini qatnashtirib yasadik. Bu metodda chop etishning kamchiliklarini kuzatish mumkin, ya’ni natajani chop etish funksiyasi orqali ko‘rilganda ikki xil qiymat ko‘rsatadi birinchida aniq javob chiqadi. Ikkinchisida

None (bo‘sh qiymat) ko‘rsatadi. Bunga asosiy sabab funksiyaga ikki marta (takroran) murojat amalga oshiriladi.

Misol. *k* butun musbat sonining *n*-raqamini qaytaradigan (nomerlash o‘ngdan chapga qarab bajarilgan) butun tipli ***DigitN(k,n)*** funksiyasi tasvirlansin. Agar *n* raqamlar sonidankatta bo‘lsa funksiya -1 qaytarsin. Berilgan 5 ta butun musbat k_1, $k_2, \ldots k_5$ sonlari uchun (1, 5) oraliqdao‘zgaruvchi *n* soniga mos raqamlar topilsin.

Dastur kodi

```
def DigitN(k,m):
    i = 0
    while k > 0:
        n = k % 10
        k = k // 10
        i = i + 1
        if i == m:
            return n
    if i < m:
        return -1
print(DigitN (12456789, 6))
```

Dastur natijasi

```
4
```

Misol. Uchta sondan kichigini topadigan ***mini (x,y,z)*** nomli funksiyasini yarating.

Dastur kodi

```
def mini(x,y,z):
    if x<y and x<z:
        min=x
    if y<x and y<z:
        min=y
    if z<x and z<y:
```

```
            min=z
    return min
x=int(input('x='))
y=int(input('y='))
z=int(input('y='))
print(mini(x,y,z))
```

Dastur natijasi

```
x=-9
y=6
y=0
-9
```

Misol. Sonning darajasini hisoblaydigan ***daraja*** *(x,y)* nomli funksiya yarating.

Dastur kodi

```
def daraj(a, b):
    i = 0;
    s = 1
    while i < b:
        i += 1
        s *= a
    return s
x = int(input('x='))
y = int(input('y='))
print(daraja(x, y))
```

Dastur natijasi

```
x=-9
y=3
-729
```

Misol. Ikki sonni taqqoslovchi ***solishtir(x,y)*** nomli funksiya yarating.

Dastur kodi

```
def solishtir(x,y):
    """Ikki sonni solishtiruvchi funksiya"""
    if x>y:
        print(f"{x} > {y}")
    elif x<y:
        print(f"{y} > {x}")
    else:
        print(f"{x} = {y}")
solishtir(10,20)
solishtir(-9,12)
solishtir(1223*5,5**4)
solishtir(-5,-5)
```

Dastur natijasi

```
20 > 10
12 > -9
6115 > 625
-5 = -5
```

Mustaqil bajarish uchun mashqlar:

1. 2 ta *r1, r2 (r1>r2)* radiusli markazlari umumiy aylanalar bilan chegaralangan xalqa yuzasini hisoblovchi haqiqiy tipli ***RingS****(r1, r2)* funksiyasi tasvirlansin(*r1* va *r2* haqiqiy). Bu funksiyadan foydalanib ichki va tashqi radiuslari berilgan 3 ta xalqaning har biri uchun yuzalar hisoblansin.
2. *[A; B]* dagi barcha butun sonlar yig‘indisini hisoblovchi butun tipli ***Range****(A, B)* funksiya tasvirlansin(*a* va *b* – butun). Agar *a>b* bo‘lsa funksiya 0 qaytarsin. Bu funksiyadan foydalanib, *a, b, c* sonlari berilganda *[a, b]* va *[b,c]* segmentlardagi butun sonlarning yig‘indilari hisoblansin.

3. Nol bo‘lmagan haqiqiy *a* va *b* sonlari ustida 1 ta arifmetik amal bajaruvchi haqiqiy tipli ***Calc****(a,b,op)* funksiyasi tasvirlansin. Bu yerda *op* parametri 1 bo‘lsa “ayirish”, 2 bo‘lsa “ko‘paytirish”, 3 bo‘lsa “bo‘lish’, boshqa hollarda “qo‘shish” amaliga ekvivalent hisoblanadi. Bu funksiyadan foydalanib, berilgan *a* va *b* sonlari uchun *n1, n2, n3* operatsiyalardagi qiymatlar chop etilsin.
4. Koordinata boshida yotmaydigan (tekislikdagi) nuqtaning qaysi chorakda joylashganligini aniqlovchi butun tipli ***Quarter****(x,y)* funksiyasi tasvirlansin. Bu funksiyadan foydalanib, berilgan 3 ta koordinata boshida yotmaydigan nuqtalarning qaysi choraklarda joylashganligi aniqlansin.
5. Agar berilgan butun son juft bo‘lsa *“true”* aks holda *“false”* qiymat qaytaruvchi mantiqiy tipli ***Even(****k*) funksiyasi tasvirlansin. Bu funksiyadan foydalanib, berilgan 10 ta butun sondan iborat nabordagi juft sonlarning miqdori topilsin.
6. Berilgan butun *k(k>0)* parametr, biror butun sonning kvadratiga teng bo‘lsa *“true”* aks holda *“false”* qiymat qaytaruvchi mantiqiy tipli ***IsSquare****(k)* funksiyasi tasvirlansin. Bu funksiyadan foydalanib berilgan 10 ta butun sondan iborat nabordagi to‘la kvadrat bo‘lgan sonlar miqdori aniqlansin.
7. Berilgan butun *k (k>0)* parametr, 5 ning biror darajasiga teng bo‘lsa *true* aks holda *false* qiymatini qaytaruvchi mantiqiy tipli ***IsPowerS****(k)* funksiyasi tasvirlansin. Bu funksiyadan foydalanib berilgan 10 ta butun sondan iborat nabordagi 5 ning darajalariga teng bo‘lgan sonlarning miqdori topilsin.
8. Berilgan butun *k (k>0)* parametr *n (n>1)* ning biror darajasiga teng bo‘lsa *true* aks holda *false* qiymat qaytaruvchi mantiqiy tipli ***IsPowerN****(k,n)* funksiyasi tasvirlansin. Bu funksiyadan foydalanib berilgan 10 ta butun sondan iborat nabordagi *n* ning darajalariga teng bo‘lgan sonlarning miqdori topilsin.
9. *n (n>1)* parametr tub son bo‘lsa *true*, aks holda *false* qiymat qaytaruvchi mantiqiy tipli ***IsPrime****(n)* funksiya tasvirlansin. Har bir 1 dan katta bo‘lgan 10 ta sondan iborat sonlar nabori berigan. Bu funksiyadan foydalanib berilgan nabordagi tub sonlar miqdori aniqlansin.

10. Berilgan *k* butun musbat sondagi raqamlar miqdorini aniqlovchi butun tipli ***Digit Count****(k)* funksiyasi tasvirlansin. Bu funksiyadan foydalanib berilgan 5 ta musbat butun sonning har biri uchun raqamlari soni aniqlansin.

11. Butun *k* parametr ***palindrom*** bo‘lsa *true* aks holda *false* qiymat qaytaradigan mantiqiy tipli ***Ispalindron(K)*** funksiyasi tasvirlansin. (palendrom son – o‘ng va chapdan bir xil o‘qiladigan sondir). Funksiyani tasvirlashda ***Digit count*** va ***Digit N*** funksiyalaridan foydalanish mumkin. Bu funksiyadan foydalanib berilgan 5 ta butun musbat sondan iborat nabordagi palendrom sonlar miqdori aniqlansin.

12. Agar burchak o‘lchovi gradusda berilgan bo‘lsa uni radianda ifodalovchi haqiqiy tipli ***DegToRad****(d)* funksiyasi tasvirlansin(*d* haqiqiy son *0<d<360*). Graduslarda berilgan 4 ta burchak o‘lchovlarining har biri uchun radian qiymatlari aniqlansin.

13. *n* faktorialni hisoblovchi haqiqiy tipli ***Fact****(n)* funksiyasi tasvirlansin. Bu funksiyadan foydalanib berilgan 5 ta butun musbat sonning har biri uchun faktoriallar hisoblansin.

14. *n* !! ni hisoblovchi haqiqiy tipli ***Fact2****(n)* funksiyasi tasvirlansin. *n* !! bu agar *n* toq bo‘lsa *n* !!=1·3·5·...· *n* agar *n* juft bo‘lsa *n* !!= 2 · 4 · 6 ·...· *n* Bu funksiyadan foydalanib berilgan 5 ta butun musbat sonlarning har biri uchun *n* !! lar hisoblansin.

15. *fk* Fibonachchi sonlarining *n*-hadini hisoblaydigan butun tipli ***Fib****(n)* funksiyasi tasvirlansin.Bu funksiyadan foydalanib *n1, n2, ..., n5* nomerlarga to‘g‘ri keluvchi Fibonachchi sonlari topilsin.

1.12. Lokal vaglobal

Lokal o‘zgaruvchilar:

Lokal qamrovli o‘zgaruvchilarni yaratish uchun ularni dasturning qolgan qismidan ajratilgan alohida kod blokiga qo‘yish kifoya. Lokal o‘zgaruvchini amalda ko‘rish uchun quyidagi misoldagi kabi *f* funksiyada 100 qiymatiga ega x nomli butun son obyektini ishga tushirish kifoya:

Dastur kodi

```
def f():
    x = 100
    print(x)
f()
```

Dastur natijasi

```
100
```

Bu yerda x chegaralangan doiraga ega, chunki u faqat *f* funksiyasi doirasida mavjud.Ushbu funksiyani dasturning tashqi qismidan chaqirish orqali siz ekranda butun son qiymatining chiqishini ko‘rishingiz mumkin.Biroq, agar siz *f* funksiya doirasidan tashqarida print funksiyasi yordamida x o‘zgaruvchisini chop etishga harakat qilingandan, kompilyator darhol xatoga yo‘l qo‘yadi:

Dastur kodi

```
def f():
    x = 100
    print(x)
f()
print(x)
```

Dastur natijasi

```
Traceback (most recent call last):
  File
"C:/Users/Ismoil/AppData/Local/Programs/Python/Python3
7-32/100.py", line 5, in <module>
    print(x)
NameError: name 'x' is not defined
```

Global o‘zgaruvchilar:

Dasturning istalgan qismida ba’zi qiymatlardan foydalanish uchun siz global o‘zgaruvchini e’lon qilishingiz kerak. Buni amalga oshirish uchun siz kod maydonidan alohida o‘zgaruvchini yaratishingiz kerak, ma’lum bir kod bloki bilan

cheklangan, masalan, funksiya. Quyidagi misol x deb nomlangan butun sonli ma’lumotlar turini identifikatsiyalashni ko‘rsatadi, keyinchalik *f* funksiyasidagi chop etish metodi yordamida ekranga chop etiladi:

Dastur kodi

```
x = 100
def f():
    print(x)
f()
print(x)
```

Dastur natijasi

```
100
100
```

Dastur natijalaridan ko‘rinib turibdiki, 100 qiymati nafaqat *f* orqali, balki odatdagi print funksiyasi yordamida ham ko‘paytiriladi. Shunday qilib, x ga kirish bunday ob’ektning global ko‘lami tufayli kodning istalgan qismidan amalga oshiriladi. Ammo ba’zi bir funksiyada global o‘zgaruvchining qiymatini o‘zgartirish mumkin? Bunday tajriba natijalari quyidagi kod parchasida keltirilgan:

Dastur kodi

```
x = 100
def f():
    x = 200
f()
print(x)
```

Dastur natijasi

```
100
```

Global kalit so‘zi:

f funksiyasi x nomli o‘zgaruvchiga 200 qiymatini belgilaydi, ammo, kutilganidan farqli o‘laroq, tashqi chop etish metodi dastlab x ga tegishli bo‘lgan

100 raqamini chiqaradi. Buning sababi shundaki, dastur mahalliy va global qamrovli ikki xil x ob’ektni yaratadi. Global kalit so‘z vaziyatni tuzatishga yordam beradi:

Dastur kodi

```
x = 100
def f():
    global x
    x = 200
f()
print(x)
```

Dastur natijasi

```
200
```

x o‘zgaruvchisini global deb belgilash orqali siz uning f funksiyasi doirasidan tashqarida aniqlangan asl qiymatiga murojaat qilishingiz mumkin. Endi, 200 raqamini x ga qo‘yganimizdan so‘ng, chop etish uchun chaqiruv kutilgan natijani, ya'ni o‘zgartirilgan qiymatni chop etadi.

Shunday qilib, Python dasturlash tilidagi o‘zgaruvchilar doirasi platformaning muhim qismidir. Uning barcha xususiyatlari bilan to‘g‘ri o‘zaro ta'sir qilish juda ko‘p murakkab xatolardan qochish imkonini beradi. Alohida ob’ektlarning ko‘rinishini xavfsizroq nazorat qilish uchun global va lokal bo‘lmagan kalit so‘zlar qo‘llaniladi.

1.13.Modullar

Modullar funksiyalardan tashkil topadi. Ilovangizga kiritmoqchi bo‘lgan funksiyalar to‘plamini o‘z ichiga olgan fayl. Shu o‘rinda bir o‘rinli savol tug‘iladi: Pythonda modullar qanday yaratiladi? Javob hayratlanarli darajada oddiy: modul - bu funksiyalarni o‘z ichiga olgan har qanday fayl. Pythondagi modul boshqa dasturlarda qayta ishlatilishi mumkin bo‘lgan yagona kod faylidir.

Fayl nomi modul nomini ifodalaydi. Keyin ushbu faylda bir yoki bir nechta funksiya aniqlanishi kerak.

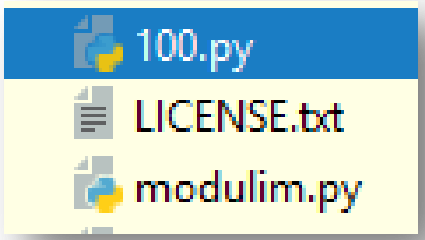
Modul yaratish uchun kerakli kodni .py fayl kengaytmali fayl ko‘rinishida saqlash kifoya:

```
def kitob(nom):
    print("Salom, " + nom)
```

Ushbu kodni birinchimodul.py nomli faylga saqlanadi. Birinchimodul.py nomli modulni import qilinadi va kitob funksiyasi chaqiriladi:

Eslatma: Moduldan funksiyadan foydalanganda quyidagi sintaksisdan foydalaniladi: **modul_name.function_name.**

Aytaylik, asosiy dastur fayli *100.py* deb ataladi. Va biz unga tashqi modullarni bog‘lashimiz kerak. Buning uchun avvalo yangi modulni aniqlaymiz: *100.py* bilan bir xil papkada yangi fayl yaratamiz, biz uni *modulim.py* deb nomlaymiz.



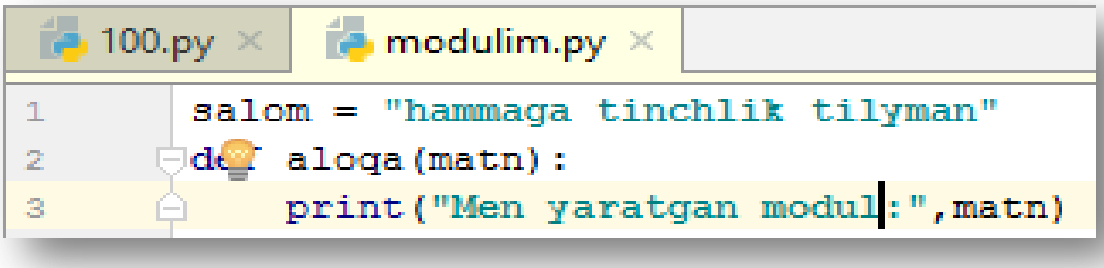
Agar PyCharm yoki boshqa IDE ishlatilsa, ikkala fayl ham bitta loyihaga joylashtiriladi. Shunga ko‘ra, modul *modulim* deb ataladi.

Unda quyidagi kodni aniqlaymiz:

salom o‘zgaruvchisi va *aloqa* funksiyasi bu yerda aniqlanadi, u parametr sifatida ba’zi matnlarni oladi va uni konsolga chop etadi.

Asosiy dastur faylida - *modulim.py* moduldan foydalaniladi:

Dastur kodi



modulim nomli modul ichidan *aloqa* nomli funksiya yoki *salom* nomli o‘zgaruvchini chaqirib ishlatish mumkin. Ya’ni bu quyidagicha ko‘rinishda

amalga oshiriladi.

100.py | modulim.py

```
1  import modulim  # modulim modulini ulanadi
2  # bu yerda salom o'zgaruvchining qiymatini ko'rsatish
3  print(modulim.salom)  # hammaga tinchlik tilyman
4  # aloqa funksiyasini chaqiriladi
5  modulim.aloqa("Salom men modulman")  # Men yaratgan modul: Salom men modulman
```

Dastur natijasi

```
hammaga tinchlik tilyman
Men yaratgan modul: Salom men modulman
```

Moduldan foydalanish uchun uni *import* kalit so‘zi yordamida import qilish kerak, undan keyin modul nomi: import *modulim*.

Modul funksionalligini global nomlar maydoniga bog‘lash: Moslashtirishning yana bir metodi **from** kalit so‘zi yordamida modul funksiyalarini joriy modulning global nom maydoniga import qilishni o‘z ichiga oladi:

Dastur kodi

100.py | modulim.py

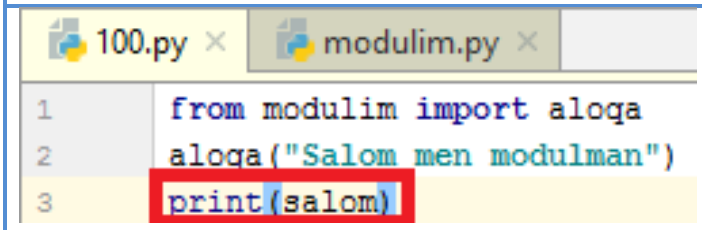
```
1  from modulim import aloqa
2  aloqa("Salom men modulman")
```

Dastur natijasi

```
Men yaratgan modul: Salom men modulman
```

from kalit so‘'zi yordamida chaqirilgan har bir funksiya yoki o‘zgaruvchi nomma-nom chaqirilishi kerak. Aks holda quyidagi ko‘rinishdagi xatolik yuz beradi.

Dastur kodi



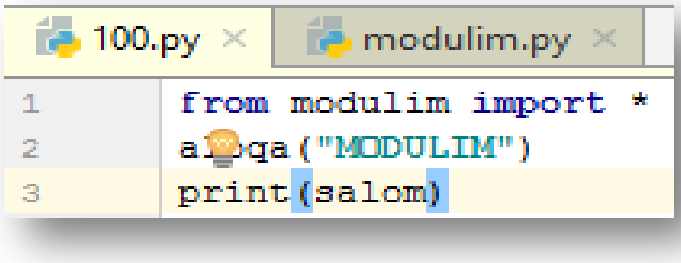
Dastur natijasi

```
Traceback (most recent call last):
  File
"C:/Users/Ismoil/AppData/Local/Programs/Python/Python3
7-32/100.py", line 3, in <module>
    print(salom)
NameError: name 'salom' is not defined
```

(Xatolikning kelib chiqish sababi 100.py fayli ichida ishlatilayotgan salom nomli o‘zgaruvchining qiymati qanday ekanligi va qaysi modulga tegishli ekanligi kompilyatorga no‘malum.)

Agar global nomlar maydoniga barcha funksiyalarni **import** qilish kerak bo‘lsa, unda alohida funksiyalar va o‘zgaruvchilar nomlari o‘rniga yulduzcha belgisi * dan foydalanishingiz mumkin:

Dastur kodi



Dastur natijasi

```
Men yaratgan modul: MODULIM
hammaga tinchlik tilyman
```

Ammo shuni ta’kidlash kerakki, global nomlar maydoniga import qilish funksiyalar nomlari to‘qnashuvi bilan to‘la. Misol uchun, agar bir xil faylda uni chaqirishdan oldin xuddi shu nom bilan belgilangan funksiya mavjud bo‘lsa, u holda joriy faylning funksiyasi chaqiriladi:

Dastur kodi

100.py modulim.py

```
1  from modulim import *
2  def aloqa(matn):
3      print(f"Izoh: {matn}")
4  aloqa("Har holatga yechim topsa bo'ladi")
```

Dastur natijasi

```
Izoh: Har holatga yechim topsa bo'ladi
```

Taxalluslarni o‘rnatish:

Modul va uning funksiyalarini **import** qilishda biz ularga taxalluslarni (qisqartma nom) o‘rnatishimiz mumkin. Bu **as** kalit sozidan keyin taxallus yordamida amalga oshiriladi. Misol uchun, modul uchun taxallus o‘rnatamiz:

Dastur kodi

100.py modulim.py

```
1  import modulim as m_m # modulim moduli m_m taxallus bilan taqqoslanadi
2  m_m.aloqa("m_m  taxallusiga o'zgardi")
3  print(m_m.salom)
```

Dastur natijasi

```
Men yaratgan modul: m_m  taxallusiga o'zgardi
hammaga tinchlik tilyman
```

Xuddi shunday, alohida modul funksiyalari uchun taxalluslarni o‘rnatish mumkin:

Dastur kodi

100.py | modulim.py

```
1  from modulim import aloqa as birlashish # bu yerda aloqa nomli funksiya nomi
2  #  birlashish nomiga o'zgardi
3  from modulim import salom as assalom # bu yerda salom o'zgaruvchisi
4  # assalom nomiga o'zgardi
5  print(assalom)
6  birlashish("Salom")
```

Dastur natijasi

```
Pythonni chuqur o'rgan
Men yaratgan modul: Salom
```

random moduli:

Random moduli tasodifiy sonlarni yaratishni boshqaradi. Uning asosiy vazifalari:

- **random():** 0,0 dan 1,0 gacha tasodifiy son hosil qiladi
- **randint():** ma'lum bir diapazondan tasodifiy sonni qaytaradi
- **randrange():** ma'lum raqamlar to'plamidan tasodifiy sonni qaytaradi
- **shuffle():** ro'yxatni aralashtirib yuboradi
- **choice():** ro'yxatning tasodifiy elementini qaytaradi

Dastur kodi

```
import random
son = random.random()   # qiymati 0.0 dan 1.0 gacha
print(son)
son = random.random()*10    # qiymati 0.0 dan 10.0 gacha
print(son)
```

Dastur natijasi

```
0.5808309296428447
9.388620642508513
```

```
0.4680583845063119
1.699267106945519
```

randint(min, max) funksiyasi min va maks ikki qiymatlari orasidagi tasodifiy butun sonni qaytaradi.

Dastur kodi

```
import random
oraliq_son = random.randint(15, 55)  # qiymati 20 dan 35 gacha
print(oraliq_son)
```

Dastur natijasi

19	42

Randrange() funksiyasi berilgan raqamlar to‘plamidan tasodifiy butun sonni qaytaradi.

Dastur kodi

```
import random
raqamlar = random.randrange(10)  # qiymat 0 dan 10 gacha
print(raqamlar)
oraliqlar = random.randrange(2, 10)  # 2, 3, 4, 5, 6, 7, 8, 9 oralig‘idagi qiymat
print(oraliqlar)
juft_son = random.randrange(2, 10, 2)  # 2, 4, 6, 8 oralig‘idagi qiymat
print(juft_son)
```

Dastur natijasi

2 7 4	1 4 4

Ro‘yxat bilan ishlash:

Ro‘yxatlar bilan ishlash uchun tasodifiy modulda ikkita funksiya aniqlanadi: ***shuffle()*** funksiyasi ro‘yxatni tasodifiy aralashtirib yuboradi va ***choice()*** funksiyasi ro‘yxatdan bitta tasodifiy elementni qaytaradi:

Dastur kodi

```
import random
avtolar = ['jentra','cobalt','malibu']
print("avtoning asl holdagi ro‘yxat i:",avtolar)
random.shuffle(avtolar)
print("avtoning aralashgan ro‘yxat i:",avtolar)
avto_nomi = random.choice(avtolar)
print("avtoni tasodfiy tanlash:",avto_nomi)
```

Dastur natijasi

```
avtoning asl holdagi ro‘yxat i:['jentra','cobalt',
'malibu']
avtoning aralashgan ro‘yxat i:['jentra', 'malibu',
'cobalt']
avtoni tasodfiy tanlash: malibu
```

```
avtoning asl holdagi ro‘yxat i: ['jentra', 'cobalt',
'malibu']
avtoning aralashgan ro‘yxat i: ['cobalt', 'malibu',
'jentra']
avtoni tasodfiy tanlash: jentra
```

Math moduli:

Pythonda o‘rnatilgan matematik modul matematik, trigonometrik va logarifmik amallarni bajarish uchun funksiyalar to‘plamini taqdim etadi. Quyidagi jadvalda modulning asosiylari keltirilgan:

1.	pow(num, power)	**sonni darajaga ko‘tarish**
2.	**sqrt(num)**	sonning kvadrat ildizi

3.	**ceil(num)**	sonni eng yaqin butun songa yaxlitlash
4.	**floor(num)**	sonni eng yaqin eng kichik butun songa yaxlitlash
5.	**factorial(num)**	sonning faktoriali
6.	**degrees(rad)**	radianlardan darajaga o‘tkazish
7.	**radians(grad)**	darajadan radianga o‘tkazish
8.	**cos(rad)**	burchakning radiandagi kosinusu
9.	**sin(rad)**	radianlarda burchak sinusi
10.	**tan(rad)**	burchakning radiandagi tangensi
11.	**acos(rad)**	radiandagi burchakning yoy kosinusu
12.	**asin(rad)**	burchakning radiandagi yoyi
13.	**atan(rad)**	burchakning radiandagi arktangensi
14.	**log(n, base)**	n sonining asosi logarifmi
15.	**log10(n)**	n sonining o‘nlik logarifmi

Dastur kodi

```
from  math import*


# 2 raqamini 5 ning darajasiga ko‘tarish
n1 = pow(2, 5)
print("n ning m-chi darajasi",n1)   # 32


# xuddi shunday operatsiyani bajarish mumkin
n2 = 2 ** 5
print(n2) # 32


# sonning kvadrat ildizi
print(sqrt(9))   # 3


# eng yaqin eng katta butun son
```

```
print(ceil(3.58))  # 4
# eng yaqin eng kichik butun son
print(floor(3.58))  # 3

# radianlardan gradusga o'tkazish
print('radian_gradus =',degrees(3.141598))  # 180

# gradusdan radianga o'tkazish
print('gradus_radian =',radians(180))  # 3.1415.....
# kosinus
print("kosinus=",cos(radians(120)))  # -
0.4999999999999998
# sinus
print("sinus=",sin(radians(0)))  #  0.0
# tangens
print("tangens=",tan(radians(1)))  #
0.017455064928217585
#logarifim
print("log=",log(9, 3))  # 2.0
#  10li logarifim
print("lg=",log10(1000))  # 3.0
```

Dastur natijasi

```
n ning m-chi darajasi 32.0
32
3.0
4
3
radian_gradus = 180.0003063267404
gradus_radian = 3.141592653589793
kosinus= -0.4999999999999998
```

```
sinus= 0.0
tangens= 0.017455064928217585
log= 2.0
lg= 3.0
```

math moduli shuningdek, **pi** va **e** kabi bir qator o‘rnatilgan konstantalarni taqdim etadi:

Dastur kodi

```
from  math import*
print('pi=',pi)
print("e=",e)
radius = 5

# radiusi 5 bo‘lgan  doira yuzi
yuza = pi * pow(radius,5)
print('Doira yuzi=',yuza)

# 10 ning natural logarifmi
son = log(1000, e)
print("natural logarifm=",son)
```

Dastur natijasi

```
pi= 3.141592653589793
e= 2.718281828459045
Doira yuzi= 9817.477042468103
natural logarifm= 6.907755278982137
```

locale moduli:

Pythonda raqamlarni formatlashda avvalboshdan Anglo-Saxon tizimidan foydalanadi, bunda butun sonning raqamlari bir-biridan vergul bilan, kasr qismi

esa nuqta bilan ajratiladi. Yevropada masalan, raqamlar nuqta bilan, kasr va butun qismlar vergul bilan ajratilgan boshqa bir tizim ishlatiladi:

```
# Anglo-Saxon tizimi
1,234.567
# Yevropa tizimi
1.234,567
```

To‘g‘ridan-to‘g‘ri raqamlar va valyutalarni formatlash uchun locale moduli ikkita funksiyani qamrab oladi:

- currency(num): valyutani formatlaydi
- format_string(str, num): str turdagi sonni o’rniga son qo‘yadi.

Har bir to‘ldiruvchidan oldin foiz belgisi % qo‘yiladi, masalan: `"%d"`

To‘ldiruvchidan oldin, nuqtadan keyin kasr sonlarni ko'rsatishda kasr qismida qancha belgi ko'rsatilishi kerakligini belgilashingiz mumkin: `%.2f`

```
import locale
locale.setlocale(locale.LC_ALL, "de")
locale.setlocale(locale.LC_ALL, "de_DE")
number = 12345.6789
formatted = locale.format_string("%f", number)
print(formatted)     # 12345,678900
formatted = locale.format_string("%.2f", number)
print(formatted)     # 12345,68
formatted = locale.format_string("%d", number)
print(formatted)     # 12345
formatted = locale.format_string("%e", number)
print(formatted)     # 1,234568e+04
money = 234.678
```

```
formatted = locale.currency(money)
print(formatted)     # 234,68 €
```

1.14.Istisno holatlar bilan ishlash

Istisnolar nima?

Istisno - bu oddiy bajarilish jarayonini buzadigan dasturlarning bajarilishi paytida sodir bo‘ladigan hodisa (masalan, kalit Lug‘atda topilmaganda ko‘tarilgan KeyError.) Istisno xatoni ifodalovchi Python obyektidir..

Pythonda istisno bu *BaseException* sinfidan olingan ob’ekt bo‘lib, u usul ichida sodir bo'lgan xato hodisasi haqida ma’lumotni o‘z ichiga oladi. Istisno obyekti quyidagilarni o‘z ichiga oladi :

- Xato turi (istisno nomi)
- Xato sodir bo'lgan dasturning holati
- Xato xabari xato hodisasini tasvirlaydi.

Mumkin bo'lgan buzilish holatlarining har xil turlarini ko‘rsatish uchun istisnolar foydalidir.Misol uchun, quyida bir nechta standart istisnolar mavjud

- FileNotFoundException
- Import xatosi
- Runtime Error
- Nom xatosi
- TypeError

Pythonda biz *try* blokiga istisno qo‘yishimiz va uni istisno blokida tutishimiz mumkin.

Nima uchun Exceptiondan foydalanish kerak?

Standartlashtirilgan xatolarni qayta ishlash : O‘rnatilgan istisnolardan foydalanish yoki aniqroq nom va tavsifga ega maxsus istisno yaratish, siz xato hodisasini tuzatishga yordam beradigan xato hodisasini mos ravishda belgilashingiz mumkin.

Tozalash kodi : Istisnolar xatoga ishlov berish kodini oddiy koddan ajratib turadi, bu bizga katta kodni osongina saqlashga yordam beradi.

Sog‘lom dastur : Istisnolar yordamida biz xato hodisasini samarali hal qila oladigan mustahkam dastur ishlab chiqa olamiz

Istisnolarning tarqalishi : agar ichki o‘rnatilgan funksiyada biron-bir xato hodisasi yuz bergan bo‘lsa, uni aniq ushlab turish va oldinga yo‘naltirish shart emas;

Turli xil xato turlari : o‘rnatilgan istisnolardan foydalanishingiz yoki o‘zingizning shaxsiy istisnolaringizni yaratishingiz va ularni umumiy ota-sinf bo‘yicha guruhlashingiz yoki xatolarni haqiqiy sinfiga ko‘ra farqlashingiz mumkin.

Misol : `FilenotfoundError` xabari diskda fayl mavjud bo‘lmaganda ekranga chiqariladi.

Dastur kodi

```
fayl = open("fayl.txt", "r")
if fayl:
    print("fayl muvaffaqiyatli ochildi")
```

Dastur natijasi

```
FileNotFoundError: [Errno 2] No such file or directory: 'fayl.txt'
```

Istisnolarni boshqarish uchun *try* va *except* blokirovka qilish.

Istisno sodir bo'lganda, Python dastur bajarilishini to‘xtatadi va istisno xabarini yaratadi. Istisno holatlar bilan shug‘ullanish tavsiya etiladi. Istisno keltirishi mumkin bo'lgan shubhali kod xavfli kod deb ataladi.

Istisnolarni hal qilish uchun *try* va *except* block dan foydalanishimiz kerak. Blok ichida istisno keltirishi mumkin bo'lgan xavfli kodni va *try blok* ichidagi tegishli ishlov berish kodini aniqlang *except*.

Sintaksisi

```
try :
    # try blokidagi ifodalar
except :
    # try blokida istisno sodir bo'lganda bajariladi
```

Misol. Ushbu misolda biz foydalanuvchidan maxraj qiymatini so'raymiz. Agar foydalanuvchi raqam kiritsa, dastur baholaydi va natijani

chiqaradi. Agar foydalanuvchi raqamli bo'lmagan qiymatni kiritsa, *try* bloki *ValueError istisno* qiladi va biz buni "*ValueError*dan tashqari" birinchi tutib olish blokidan foydalanib, "Kiritilgan qiymat noto‘g‘ri" xabarini chop etish orqali aniqlay olamiz.

Deylik, foydalanuvchi maxrajni nol sifatida kiritadi. Bunday holda, try bloki a ni tashlaydi *ZeroDivisionError* va biz buni ikkinchi blokdan foydalanib, "Nolga bo'linib bo‘lmaydi" xabarini chop etish orqali aniqlaymiz.

Dastur kodi

```
try:
    a = int(input("a qiymatini kiriting:"))
    b = int(input("b qiymatini kiriting:"))
    c = a/b
    print("b ga bo'lishning javobi:", c)
except ValueError:
    print("Kiritilgan qiymat noto'g'ri")
except ZeroDivisionError:
    print("Nolga bo'lib bo'lmaydi")
```

Dastur natijasi(*1-xolat*)

```
a qiymatini kiriting:3
b qiymatini kiriting:2
b ga bo'lishning javobi: 1.5
```

Dastur natijasi (*2-xolat*)

```
a qiymatini kiriting:3
b qiymatini kiriting:0
Nolga bo'lib bo'lmaydi
```

Dastur natijasi (*3-xolat*)

```
a qiymatini kiriting:olti
Kiritilgan qiymat noto'g'ri
```

1.15. Massivlar (python array)

Pythondagi massiv bir xil turdagi ob’ektlarni saqlash uchun ishlatiladigan tartiblangan ma’lumotlar strukturasidir. Funksional maqsadlariga ko‘ra, ular ro‘yxatlarga o‘xshaydi, lekin ular kiritilgan ma’lumotlar turiga, shuningdek, ularning hajmiga nisbatan ba’zi cheklovlarga ega. Ushbu xususiyatga qaramay, massivlar Python dasturlash tilida ma’lumotlar to‘plamlari bilan ishlash uchun juda funksional vositadir.

Massivni yaratish va to‘ldirish:

Python3da yangi massivni qo‘shishdan (yaratishdan) oldin siz bunday ob’ekt bilan ishlash uchun mas'ul bo‘lgan kutubxonani import qilishingiz kerak. Buning uchun dastur fayliga `from array import *`qatorni qo‘shishingiz kerak. Yuqorida aytib o‘tilganidek, massivlar bitta doimiy ma’lumotlar turi bilan o‘zaro ta'sir qilish uchun mo‘ljallangan, buning natijasida ularning barcha elementlari bir xil o‘lchamga ega. Massiv funksiyasidan foydalanib, siz yangi ma’lumotlar to‘plamini yaratishingiz mumkin. Quyidagi misol Python massivini to‘ldirishni ko‘rsatadi - yuqorida tavsiya etilgan metod yordamida butun sonlarni yozish.

Dastur kodi

```
from array import *
tartib_raqam = array('i', [2, 5, 4, 0, 8])
```

Ko‘rib turibdiki, `array` funksiyasi ikkita argumentni oladi, ulardan birinchisi yaratilayotgan massivning turi, ikkinchisi esa uning qiymatlarining dastlabki ro‘yxati. Bu holda `i` ikki bayt xotirani egallagan ishorali butun sondir. Buning o‘rniga boshqa primitivlardan foydalanish mumkin, masalan, 1 baytlik belgi (`c`) yoki 4 baytlik `float` (`f`).

Shuni yodda tutish kerakki, massiv faqat bitta turdagi ma’lumotlarni saqlashi mumkin, aks holda dastur chaqiruvi muvaffaqiyatsiz bo‘ladi.

Massivga element qo‘shish:

Python massiviga yangi element qo‘shish uchun *insert* metodidan foydalaniladi. Buni amalga oshirish uchun avval yaratilgan ob’ekt orqali chaqirish

va ikkita qiymatni argument sifatida kiritish kerak. Birinchisi (4) massivdagi yangi elementning indeksi, ya'ni uni joylashtirish kerak bo‘lgan joy uchun javob beradi, ikkinchisi (3) esa qiymatning o‘zi.

Dastur kodi

```
from array import *
tartib_raqam = array('i', [2, 5, 4, 0, 8])
tartib_raqam.insert(4, 3)
print(tartib_raqam)
```

Shuni esda tutish kerakki, avval yaratilgan ob’ekt massivga tegishli bo‘lgan turdagi ma’lumotlarni qo‘shish mumkin. Bunday operatsiyani bajarishda mavjud elementlar soni dasturning joriy holatiga qarab ortadi.

Dastur natijasi

```
array('i', [2, 5, 4, 0, 3, 8])
```

Massiv elementini olib tashlash(o‘chirish):

Pythonda argumenti element indeksi (3) bo‘lgan `pop` metodi yordamida massivdan keraksiz elementlarni olib tashlash mumkin. Yangi element qo‘shilganda bo‘lgani kabi, misolda ko‘rsatilganidek, metod oldindan yaratilgan ob’ekt orqali chaqirilishi kerak.

Dastur kodi

```
from array import *
tartib_raqam = array('i', [2, 5, 4, 0, 8])
tartib_raqam.pop(3)
print(tartib_raqam)
```

Ushbu amalni bajargandan so‘ng, massivning mazmuni mavjud xotira katakchalari soni joriy elementlar soniga mos keladigan tarzda o‘zgartiriladi.

Dastur natijasi

```
array('i', [2, 5, 4, 8])
```

Dasturdagi har qanday ma’lumotlar bilan ishlashda vaqti-vaqti bilan ularni tekshirish zarurati tug‘iladi, bu ularni ekranda ko‘rsatish orqali osonlik bilan amalga oshirilishi mumkin. `print()` funksiyasi bu amalni bajarishiga yordam beradi. Argument sifatida avval yaratilgan va to‘ldirilgan massivning elementlaridan birini oladi. Quyidagi misolda u *for* sikli yordamida qayta ishlanadi, bunda ma’lumotlar massivining har bir elementiga avval aytib o‘tilgan `print()` funksiyasiga o‘tish uchun vaqtinchalik identifikator beriladi.

Dastur kodi

```
from array import *
raqam_s = array('i', [42, 55, 48, 10, 8])
for i in raqam_s:
    print(i,end(' ')
```

Dastur natijasi

```
42 55 48 10 8
```

Yuqoridagi kodning chiqishi Python massivining chiqishidir - ilgari tayinlangan barcha butun qiymatlarni takrorlang va navbat bilan bitta qatorda chiqaring.

Massiv uzunligini o‘lchash:

Dasturni bajarish jarayonida massivning o‘lchami o‘zgarishi mumkinligi sababli, ba'zan massivdagi elementlarning joriy sonini bilish foydali bo‘ladi. `len()` funksiyasi Pythonda massiv uzunligini (hajmini) butun son qiymati sifatida olish uchun ishlatiladi.

Dastur kodi

```
from array import *
t_raqam = array('i', [10, 2263, 300, 44, 556])
print(len(t_raqam))
```

Dastur natijasi
10 2263 300 44 556

1.15.Pythonda ro‘yxatlar, lug‘atlar, kortejlar va to‘plamlar

Ro‘yxatlar

Ma’lumotlar to‘plamlari bilan ishlash uchun Python ro‘yxatlar, kortejlar va lug‘atlar kabi o'rnatilgan turlarni taqdim etadi.

Ro‘yxat (ro‘yxat) – elementlar to‘plami yoki ketma-ketligini saqlaydigan ma’lumotlar turi. Ko‘pgina dasturlash tillarida massiv deb ataladigan o‘xshash ma’lumotlar tuzilmasi mavjud. Ro‘yxatlar bir o‘garuvchida bir nechta elementlarni saqlash uchun ishlatiladi. Ro‘yxatlar Pythonda ma’lumotlar to‘plamini saqlash uchun ishlatiladigan 4 ta o‘rnatilgan ma’lumotlar turlaridan biri, qolgan 3 tasi Tuple , Set va Dictionary bo‘lib, ularning barchasi turli sifat va foydalanishga ega.

Ro‘yxat yaratish:

Ro‘yxat ni yaratish uchun kvadrat qavslar [] ishlatiladi, ularning ichida ro‘yxat elementlari vergul bilan ajratilgan. Masalan, raqamlar ro‘yxat ini aniqlaymiz:

```
raqamlar = [1, 2, 3, 4, 5]
```

Xuddi shunday, boshqa turdagi ma’lumotlar bilan ro‘yxat larni belgilash mumkin, masalan, qatorlar ro‘yxatini aniqlaymiz:

```
gullar = ["atirgul","lola", "qizgaldoq"]
```

Ro‘yxat yaratish uchun list() funksiyasidan ham foydalanishingiz mumkin:

```
numbers1 = []
numbers2 = list()
```

Ushbu ikkala ro‘yxat ta’riflari o‘xshash - ular bo‘sh ro‘yxat yaratadi.

Ro‘yxat da faqat bir xil turdagi ob’ektlar bo'lishi shart emas. Bir vaqtning o‘zida satrlarni, raqamlarni, boshqa ma’lumotlar turlarining ob’ektlarini bir xil ro‘yxatga qo‘yish mumkin. Bu ko‘rinish quyidagicha:

Dastur kodi

```
aralash_turlar = [ "python", False,1996, 1.992]
print(aralash_turlar)
```

Dastur natijasi

```
['python', False, 1996, 1.992]
```

Ro‘yxat elementlarini tekshirish uchun standart chop etish `print()` funksiyasidan foydalanish mumkin, bu ro‘yxat mazmunini foydalanuvchiga qulay shaklda chop etadi:

Dastur kodi

```
raqamlar = [1, 2, 3, 4, 5]
gullar = ["atirgul","lola", "qizgaldoq"]
print(raqamlar)
print(gullar)
```

Dastur natijasi

```
[1, 2, 3, 4, 5]
['atirgul', 'lola', 'qizgaldoq']
```

Ro‘yxat konstruktori qiymatlar to‘plamini olishi mumkin, ular asosida ro‘yxat tuziladi:

Dastur kodi

```
raqamlar1 = [1, 2, 3, 4, 5]
raqamlar2 = list(raqamlar1)
print(raqamlar2)  # [1, 2, 3, 4, 5]
matnlar = list("Python")
print(matnlar)  # ['P', 'y', 't', 'h', 'o','n']
```

Dastur natijasi

```
[1, 2, 3, 4, 5]
['P', 'y', 't', 'h', 'o', 'n']
```

Agar bir xil qiymat bir necha marta takrorlanadigan ro‘yxatni yaratish kerak bo‘lsa, unda yulduzcha ***** belgisidan foydalanish mumkin, ya’ni aslida mavjud ro‘yxatga ko‘paytirish amali qo‘llalaniladi:

Dastur kodi

```
sonlar = ['ro\'yxat'] * 4 # ro’yxat so'zi 4 marta takrorlanadi
print(sonlar)

ism = ["Python"] * 3 # Python so'zi 3 marta takrorlanadi
print(ism)
meva = ["uzum", "olma"] * 2 # uzum olma so'zlari 2 marta takrorlanadi
print(meva)
```

Dastur natijasi

```
["ro'yxat", "ro'yxat", "ro'yxat"", "ro'yxat"]
['Python', 'Python, 'Python']
['uzum', 'olma', 'uzum', 'olma']
```

Ro‘yxat elementlariga murojaat qilganda, ro‘yxatdagi elementning raqamini ko‘rsatadigan indekslardan foydalanish kerak. Indekslar noldan boshlanadi. Ya’ni, birinchi element indeks 0, ikkinchi element indeks 1 bo‘ladi va hokazo.

Dastur kodi

```
aralash = ["Non", 5.65,89]
#ro‘yxat  boshidan elementlarni chop etish
print(aralash[0])   # Non
print(aralash[1])   # 5.65
print(aralash[2])   # 89
```

```
#ro‘yxat  oxiridan elementlarni chop etish
print(aralash[-2])   # 5.65
print(aralash[-1])   # 89
print(aralash[-3])   # Non
```

Dastur natijasi

```
Non
5.65
89
5.65
89
Non
```

Ro‘yxat elementini o‘zgartirish mumkin. Unga yangi qiymat belgilash kifoya:

Dastur kodi

```
aralash = ["Non", 5.65,89]
print('dastlabki ro\'yhat =',aralash)
aralash[2] = 'sakson to\'qqiz'
# uchunchi element almashtirildi
print('3-chi element yangi qiymat =',aralash[2])

# sakson
print('yangilangan ro\'yxat =',aralash)
# ['Non', 5.65, "sakson"]
```

Dastur natijasi

```
dastlabki ro’yxat = ['Non', 5.65, 89]
3-chi element yangi qiymat = sakson to'qqiz
yangilangan ro‘yxat  = ['Non', 5.65, "sakson"]
```

Ro‘yxatga elementlarni qo‘shish va o‘chirish:

Ro‘yxatga element qo‘shish *append, insert* metodi yordamida amalga oshiriladi. elementlarni o‘chirishda esa *remove, clear, pop* metodlaridan foydalaniladi, masalan:

Dastur kodi

```
ism = ['olim','anvar','islom']
ism.append('Sarvar')
print(ism)
```

Dastur natijasi

```
['olim', 'anvar', 'islom', 'Sarvar']
```

append metodi berilgan elemntni ro‘yxatni oxiridan qo‘shadi

Ro‘yxatga element qo‘shish uchun yana boshqa bir metod *insert* metodidan foydalaniladi.

Dastur kodi

```
ism = ['olim','anvar','islom']
ism.insert(1,'Sarvar')
print(ism)
```

Dastur natijasi

```
['olim', 'Sarvar', 'anvar', 'islom']
```

insert metodi ro‘yxatning istalgan qismidan element qo‘shish imkonini beradi. Unda kiritlayotgan elementning indeksi bilan birga kiritiladi, shu sababli elementni ro‘yxatning istalgan qismidan kiritsh mumkin.

Misol. *n (n>0)* butun son berilgan. Dastlabki *n* ta musbat toq sonlarni saqlaydigan *bir* o‘lchamli butun sonli massiv tashkil etilsin.

Dastur kodi

```
n = int(input('n = '))# toq sonlar soni kiritiladi
son = [] # bo'sh ro'yxat  e'lon qilinadi
```

```
i = 0 # indeks 0 dan boshlanadi
j = 1 # birinchi toq son qiymati
while i < n: # shart yolg'on bo'lmaguncha bajariladi
    son.append(j) # bo‘sh ro‘yxat ga 1-element qo'shiladi
    j+=2 # toq sonni ifodalash
    i += 1  # indeks qiymati 1 ga oshiriladi
print(son)
```

Dastur natijasi

```
 n = 5
 [1, 3, 5, 7, 9]
```

Dastur ishga tushirilganda foydalanuvchidan ro‘yxatda joylashishi mumkin bo‘lgan toq sonlar soni *n* so‘raladi. Kiritilgan *n* soni siklda tormoz (*to’xtatish*) vazifasini bajaradi.

Elementlarni o‘chirishda esa *remove*, *clear*, *pop* metodlaridan foydalaniladi, masalan:

Dastur kodi

```
ism = ['olim','anvar','islom']
ism.remove('anvar')
print(ism)
```

Dastur natijasi

```
['olim', 'islom']
```

Ro‘yxat dagi takrorlanib kelgan elementlarni *remove* metodidan foydalanib faqat bitta birinchi uchragan elementni o‘chirish mumkin. Elementlar soni bir nechta berilgan bo‘lsa ular takrorlanish operatorlari orqali o‘chiriladi, masalan:

Dastur kodi

```
ism =['Azim','Valijon','Asilbek','Valijon','Valijon']
for i in ism:
```

```
    ism.remove('Valijon')
print(ism)
```

Dastur natijasi

```
['Azim', 'Asilbek']
```

Ro‘yxatlarni taqqoslash. Ikkita ro‘yxat bir xil elementlardan tashkil topgan bo‘lsa, bunday ro‘yxatlar teng hisoblanadi.

Dastur kodi

```
son1 = [0, 1, 2, 3, 4]
son2 = list(range(5))
if son1 == son1:
    print("Bu ro‘yxatlar teng.")
else:
print("Bu  ro‘yxatlar teng emas.")
```

Dastur natijasi

```
Bu ro‘yxatlar teng
```

Bu holatda ikkila ro‘yxat teng hisoblanadi.

Ro‘yxatlar bilan ishlashda qo‘llaniladigan funksiyalar va metodlar. Ro‘yxatlar bilan ishlashda qo’llaniladigan bir nechta metodlar mavjud bo‘lib, ularning eng asosiylari quyida keltirilgan:

№	Metodlar nomi	Vazifasi
1.	***append(item)***	ro‘yxat oxiriga *item* elementini qo‘shish
2.	***insert(index, item)***	Ro‘yxatga *index* indeksi bo‘yicha *item* elementini qo‘shish;
3.	***remove(item)***	ro‘yxat dan *item* elementini o‘chirish. Ushbu metod ro‘yxat dagi birinchi uchragan *item* elementini o’chiradi. Agar bunday element ro‘yxatda mavjud

		bo’lmasa *ValueError* istisno holati ro‘y beradi
4.	***clear()***	ro‘yxat ni tozalash, ya‘ni ro‘yxatdagi barcha elementlarni o‘chirish;
5.	***index(item)***:	ro‘yxatdagi *item* elementining joylashgan indeksini qiymat sifatida qaytaradi. Agar bunday element ro‘yxat da mavjud bo’lmasa, u holda *ValueError* istisno holati ro‘y beradi
6.	***pop([index])***	ro‘yxatdan *index* indeksi bo‘yicha elementni o‘chiradi va qiymat sifatida qaytaradi. Agar indeks ko‘rsatilmasa ro‘yxatdan oxirgi elementni o‘chiradi va qiymat sifatida qaytaradi. Bundan tashqari agar ro‘yxat bo‘sh bo‘lsa, u holda *ValueError* istisno holati ro‘y beradi
7.	***count(item)***	ro‘yxatdagi *item* elementlar sonini qiymat sifatida qaytaradi
8.	***sort([key])***	ro‘yxat elementlarini tartiblaydi. Kelishuv bo‘yicha, ro‘yxat elementlarini o‘sish bo‘yicha tartiblaydi. *key* parametri orqali tartiblash funksiyasini (mezonini) berish mumkin
9.	***reverse()***	ro‘yxat elementlarini teskari tartibda joylashtirish uchun qo’llaniladi.
10.	***len(list)***	ro‘yxat uzunligini (elementlari sonini) qiymat sifatida qaytaradi
11.	***sorted(list,[key])***	tartiblangan ro‘yxatni qiymat sifatida qaytaradi
12.	***min(list)***	ro‘yxatdagi eng kichik elementni qaytaradi
13.	***max(list)***	ro‘yxatdagi eng katta elementni qaytaradi

Elementni ro‘yxatda mavjudligini tekshirish. Odatda, *index*, *remove* kabi

metodlardan foydalanilganda, ularning parametrlarida ko‘rsatilgan element ro‘yxatda mavjud bo’lmasa, istisno holati yuzaga keladi. Bunday holatlarning oldini olish uchun, oldin ushbu elementning ro‘yxatda mavjudligini tekshirish kerak bo‘ladi. Buning uchun *in* kalit so‘zidan foydalaniladi:

```
companies = ["Microsoft", "Google", "Oracle",
"Apple"]item = "Oracle"  # o'chirish kerak
bo'lgan  element
if item in companies:
    companies.remove(item)

print(companies) # ['Microsoft', 'Google', 'Apple']
```

Yuqoridagi holatda, agar ro‘yxatda *item* elementi mavjud bo‘lsa, *item in companies* ifodasi *True* qiymat qaytaradi. Shuning uchun *if* shart ifodasi agar shu element mavjud bo‘lsagina navbatdagi amallarni bajaradi, natijada istisno holatining oldi olinadi.

***count()* metodi**. Ushbu metod orqali ro‘yxatda biror element nechi marta qatnashganligi topiladi. Masalan:

```
nomlar =
["Anvar","Sobir","Sobir","Qosim"]
nom_soni = nomlar.count("Sobir")
print(nom_soni) #2
```

Tartiblash. Ro‘yxatdagi elementlarni o‘sib borish bo‘yicha tartiblash uchun ***sort()*** metodi qo‘llaniladi:

```
nomlar =["Anvar","Sobir","Sobir","Qosim"]
nomlar.sort()
print(nomlar)  # ['Anvar', 'Qosim', 'Sobir', 'Sobir']
```

Agar teskari tartibda tartiblash kerak bo‘lsa, ***sort()*** metodidan keyin ***reverse()***metodidan foydalanish kifoya qiladi:

```
nomlar =
["Anvar","Sobir","Sobir","Qosim"]
nomlar.sort()
nomlar.reverse()
#nomlar.sort(reverse=True) deb ham ishlatish mumkin
print(nomlar)  # ['Sobir', 'Sobir', 'Qosim', 'Anvar']
```

Shuni alohida ta’kidlash lozimki, tartiblashda ob’ektlar taqqoslanadi. Ob’ekt sifatida son kelsa muammo yo‘q, o‘sish yoki kamayib borish bo‘yicha tartiblanadi. Lekin ob’ekt sifatida satr kelsa, u holda ular mos belgilari bo‘yicha taqqoslanadi. Taqqoslashda belgilarning ASCII kodlari taqqoslanadi. Shuning uchun har qanday katta registrga ega lotin yozuvidagi belgi kichik registrdagi lotin yozuvidagi belgidan kichik bo’ladi, masalan: “Abc” < “abc”:

```
nomlar = ["anvar","Sobir","sobir","Qosim","tolib"]
nomlar.sort()
print(nomlar) # ['Qosim', 'Sobir', 'anvar', 'sobir',
'tolib']
```

Tartiblashda ***sort()*** metodi o‘rniga standart ***sorted()*** funksiyasidan ham foydalanish mumkin bo‘lib, uning quyidagi ikkita shakli mavjud:

- *sorted(list)* – ro‘yxat elementlarini tartiblash uchun;
- *sorted(list, key)* – ro‘yxat elementlarini *key* mezon (funksiya) asosida tartiblash uchun ishlatiladi.

```
nomlar = ["anvar","Sobir","sobir","Qosim","tolib"]
sorted_nomlar = sorted(nomlar, key = str.lower)
print(sorted_nomlar)  # ['anvar', 'Qosim', 'Sobir',
'sobir', 'tolib']
```

sorted() funksiyasidan foydalanilganda qiymat sifatida ushbu funksiya tartiblangan elementlardan tashkil topgan yangi ro‘yxatni qaytaradi, tartiblanayotgan ro‘yxat esa o‘zgarishsiz qoladi.

Minimal va maksimal qiymatlar. Pythonda ***max***, ***min*** deb nomlanuvchi mos ravishda ro‘yxatdan eng maksimal va eng minimal qiymatlarni topish uchun mo‘ljallangan standart funksiyalari mavjud.

```
sonlar = [12, 45, 23, -35, 2]

print(min(sonlar))  # -35

print(max(sonlar))  # 45
```

Ro‘yxatlarni ko‘chirish. Ro‘yxat – o‘zgaruvchan (mutable) turga mansub bo‘lib, agar ikkita o‘zgaruvchi ayni bir ro‘yxatga murojaat qilayotgan bo‘lsa, u holda birining o‘zgarishi ikkinchisiga ham ta’sir qiladi:

```
vil1 = ["Toshkent",
"Xorazm", ]vil2 = vil1
vil2.append('Buxoro')
print(vil1)  # ['Toshkent', 'Xorazm', 'Buxoro']

print(vil2)  # ['Toshkent', 'Xorazm', 'Buxoro']
```

Bu misolda har ikkala *vil1* va *vil2* o‘garuvchilar yordamida ayni bir ro‘yxatga murojaat bo‘lgan . Shuning uchun *vil2* o‘garuvchisi orqali ro‘yxatga yangi element qo‘shilganda mos ravishda *vil1* ham o‘zgargan. Bu holat yuzaki ko‘chirish (*shallow copy*) deyiladi. Albatta ro‘yxatlarni ko‘chirganda ikkita alohida ro‘yxat hosil qiladigan tarzda ham ko‘chirish (*deep copy*) mumkin. Buning uchun ichki ***copy*** moduldagi *deepcopy()* metodidan foydalaniladi:

```
import copy

vil1 = ["Toshkent",
"Xorazm", ]vil2 =
copy.deepcopy(vil1)
vil2.append('Buxoro')
# ikkita alohida ro‘yxat hosil bo‘ldi
```

```
print(vil1)  # ['Toshkent', 'Xorazm']

print(vil2)  # ['Toshkent', 'Xorazm', 'Buxoro']
```

Ro‘yxat qismini ko‘chirish. Agar butun bir ro‘yxatni emas, balki uning bir qismini ko‘chirish zarur bo‘lsa, u holda maxsus sintaksisdan foydalaniladi. Ushbu sintaksis quyidagi shakllarda qo‘llaniladi:

list(:end) - *end* parametri orqali ro‘yxat ning qaysi elementigacha ko‘chirish kerakligini bildiruvchi indeks nomeri beriladi;

list(start:end) – *start*, *end* parametrlar orqali ro‘yxat ning *start* dan *end* gacha bo‘lgan elementlarini ko‘chirish kerakligini bildiruvchi indeks nomerlari beriladi;

list(start:end:step) – *start*, *end*, *step* parametrlar orqali ro‘yxatning *start* dan *end* gacha bo‘lgan elementlarini *step* qadam bilan ko‘chirish kerakligini bildiruvchi qiymatlar beriladi. *step* parametrning kelishuv bo‘yicha qiymati 1 ga tengdir.

```
vil = ["Toshkent", "Xorazm", 'Buxoro', 'Navoi',
'Jizzax']vil1 = vil[:2]
print(vil1)  # ['Toshkent', 'Xorazm']

vil2 = vil[2:4]

print(vil2)  # ['Buxoro', 'Navoi']

vil3 = vil[1:5:2]

print(vil3)  # ['Xorazm', 'Navoi']
```

Ro‘yxatlarni birlashtirish. Ro‘yxatlarni birlashtirish uchun (+) amali qo’llaniladi:

```
vil1 = ["Toshkent", "Xorazm",
'Buxoro']vil2 = ['Navoi',
'Jizzax']
vil = vil1 + vil2

print(vil)
```

Ichma – ich joylashgan ro‘yxat lar. Ro‘yxat elementlari son, satr kabi oddiy turdagi qiymatlargina bo’lib qolmay, balki ro‘yxat ni ham ifodalashi mumkin. Odatda bunday ro‘yxat lardan jadvallar bilan ishlashda ko‘p foydalaniladi. Ushbu holatda tashqi ro‘yxat ning har bir elementi jadvaldagi bitta qatorni ifodalovchi ro‘yxat dan iborat bo’ladi:

```
ishchilar = [
    ["Tolib", 33],
    ["Akmal", 30],
    ["Botir", 27] ]

print(ishchilar[0]) # ["Tolib", 33]
print(ishchilar[0][0]) # Tolib
print(ishchilar[0][1])  # 33
```

Bu yerda ichki ro‘yxat ni elementiga murojaat qilish uchun [][] indekslar juftligidan foydalanilgan. Xususan, *ishchilar[0][1]* – birinchi ichki ro‘yxat ning ikkinchi elementiga murojaat bo‘lgan .

Ro‘yxat ga elementlarni qo‘shish, o‘chirish, o‘zgartirish kabi jarayonlar oddiy ro‘yxatlardagi kabi amalga oshiriladi:

```
ishchilar = [
    ["Tolib", 33],
    ["Akmal", 30],
    ["Botir", 27] ]
```

```
print(ishchilar[0]) # ["Tolib", 33]
print(ishchilar[0][0]) # Tolib
print(ishchilar[0][1])  # 33
# ro‘yxat yaratish ishchi = list()
ishchi.append("Rustam")ishchi.append(21)
# Tashqi ro‘yxatga yaratilgan ro‘yxatni qo‘shish
ishchilar.append(ishchi)
print(ishchilar[-1])  # ["Rustam",
21]
```

```
# Tashqi ro‘yxatning oxirgi elementiga element qo‘shish
ishchilar[-1].append("+998909737066")
print(ishchilar[-1])  # ['Rustam', 21, '+998909737066']
# tashqi ro'yxatning oxirgi elementining oxirgi
elementini# o‘chirish
ishchilar[-1].pop()
print(ishchilar[-1])  # ["Rustam", 21]
# tashqi ro'yxatning oxirgi elementini o‘chirish
ishchilar.pop(-1)
# birinchi elementni o'zgartirish
ishchilar[0] = ["Sobir", 18]
print(ishchilar) # [['Sam', 18],['Akmal', 30],['Botir',
27]]
```

Murakkab ro‘yxatlar elementlariga murojaat ichma – ich joylashgan takrorlashoperatorlari orqali amalga oshirilishi mumkin:

```
xodimlar = [
    ["Sherali", 45],
```

```
    ["Aziz", 56],
    ["Bahrom", 22] ]
for ishchi in ishchilar:
    for i in ishchi: print(i, end="|")
# Konsolga quyidagi ma`lumotlar chiqariladi
# Sherali|45|Aziz|56|Bahrom|22|
```

Mustaqil bajarish uchun mashqlar:

1. n (n>0) butun son berilgan. 2 ning darajalarini saqlaydigan n o‘lchamli butun sonli massiv tashkil etilsin.

2. Butun n (n>1) soni, arifmetik progressiyaning birinchi hadi a va uning ayirmasi d berilgan. Shulardan foydalanib o‘zida arifmetik progressiyaning dastlabki n ta hadini saqlovchi massiv tashkil etilsin.

3. n (n>1) butun soni hamda birinchi hadi b va maxraji q bo‘lgan geometrik progressiya berilgan. Shulardan foydalanib o‘zida geometrik progressiyaning dastlabki n ta hadini saqlovchi massiv tashkil etilsin.

4. n (n>2) butun soni berilgan. $f_1=1$, $f_2=1$, $f_k=f_k-2+f_{k-1}$, k=3,4,…
fk Fibonachchi sonlar ketma-ketligida birinchi n ta elementni o‘z ichiga oladigan n o‘lchamli butun sonli massiv ifodalansin va chop etilsin.

5. n (n>2), a va b butun sonlar berilgan. 1-elementi a ga, 2-elementi b, har bir keyingi elemeti barcha avvalgi elementlar (o‘zidan oldingi barcha element) yig‘indisiga teng bo‘lgan n o‘lchamli butun sonli massiv ifodalansin va chop etilsin.

6. n o‘lchamli a massiv berilgan. Uning elementlari teskari tartibda chiqarilsin.

7. n o‘lchamli butun sonli massiv berilgan. Berilgan massivni indekslari bo‘yicha tartibida tartiblab, massivdagi juft sonlar va ularning miqdori k chiqarilsin.

8. n o‘lchamli, butun sonli massiv berilgan. Berilgan massivdagi barcha toq sonlarni o‘z ichiga oladigan elementlarni o‘sish tartibida tartiblab, chop etilsin hamda ularning miqdori k aniqlansin.

9. n o‘lchamli butun sonli massiv berilgan. Massivdagi juft sonli elementlarining indekslarini o‘sish tartibida, toq sonli elementlarining indekslarini kamayish tartibida tartiblab, massiv chop etilsin.

10. n o‘lchamli a massiv va k($1 \le k \le n$) butun soni berilgan. Massiv elementlari shart operatoridan foydalanmasdan quyidagi tartibda chop etilsin:

$a_k, a_{k-1}, a_{k-2}, \dots a_1$.

11. n o‘lchamli a massiv berilgan(n-juft son). (indekslari o‘sish tartibida) Juft indeksdagi elementlari chiqarilsin. $a_2, a_4, \dots, a_n$. Shart operatoridan foydalanilmasin.

12. n o‘lchamli a massiv berilgan(n-toq son). Massivning toq indeksida turgan elementlari indekslarini kamayish tartibida tartiblab chiqarilsin. a_n, a_{n-2}, a_{n-4}, … a_1 shart operatoridan foydalanilmasin.

13. n o‘lchamli a massiv berilgan. Avval massivning juft indeksli elementlari(indekslarini o‘sish tartibida) keyin toq indeksli elementlari (indekslarini o‘sish tartibida) chiqarilsin: a_2, a_4, a_6, … a_1, a_3, a_5… Shart operatoridan foydalanilmasin.

14. n o‘lchamli a massiv berilgan. Avval toq indeksdagi elementlar, keyin juft indeksdagi elementlar kamayish tartibida chop etilsin.

15. n o‘lchamli a massiv berilgan. Uning elementlari quyidagi tartibda chiqarilsin:

$a_1, a_n, a_2, a_{n-1}, a_3, a_{n-2}$

Kortejlar

Kortej (tuple) elementlar ketma-ketligini ifodalovchi, ko‘p jihatlari bo‘yicha ro‘yxat ga o’xshaydigan, lekin undan farqli ravishda o‘zgarmaydigan (immutable) tur hisoblanadi. Shuning uchun kortejga yangi element qo’shish, undan elementni o‘chirish yoki o‘zgartirish kiritishga ruxsat etilmaydi.

Kortejni hosil qilish uchun oddiy qavs “(,)” dan foydalanib, unda elementlar vergul bilan ajratilgan tarzda joylashtiriladi:

```
user = ("Akmal", 21)

print(user) # ('Akmal', 21)
```

Bundan tashqari kortejni aniqlash uchun elementlar ketma – ketligi vergul bilan ajratilgan holda oddiy qavslarsiz ham amalga oshirsa bo‘ladi:

```
user = "Akmal", 21

print(user) # ('Akmal', 21)
```

Shuni alohida ta’kidlash kerakki, kortej faqat bitta elementdan tashkil topsa ham vergul ishlatiladi:

```
user = "Akmal",

print(user) # ('Akmal',)
```

Ro‘yxat asosida kortej hosil qilish uchun maxsus ***tuple*** funksiyasidan foydalaniladi va uning argumentiga qiymat sifatida ro‘yxat beriladi:

```
user_list = ["Yusuf", 'Tolib',
'Rustam']user_tuple =
tuple(user_list)
print(user_tuple)  # ('Yusuf', 'Tolib', 'Rustam')
```

Kortej elementlariga murojaat xuddi ro‘yxat lardagi kabi indekslari orqali amalga oshiriladi. Indekslar kortej boshiga nisbatan 0 dan, kortej oxiriga nisbatan -1 dan boshlanadi:

```
users = ("Yusuf", 'Qodir', 'Erkin',
'Oybek')print(users[0]) # Yusuf
print(users[2]) # Erkin

print(users[-1])  # Oybek

print(users[1:3])  # ('Qodir', 'Erkin')
```

Kortej o‘zgarmaydigan tur bo‘lganligi sababli uning elementini o‘zgartirib bo‘lmaydi. Masalan: *users[0] = “Rahim”* kabi kod yozilsa, Python interpretatori xatolik to‘g‘risida xabar chiqaradi.

```
def get_user():
    name = "Yusuf"
    age = 33
    is_married = True
    return name, age, is_married
user = get_user()
print(user[0]) # Yusuf
print(user[1]) # 33
print(user[2])  # True
# yoki alohida o‘zgaruvchilarga
yuklanadiname, age, ismarried =
get_user() print(name)  # Yusuf
print(age)  # 33
print(ismarried)  # True
```

len() funksiyasi orqali kortejning uzunligi (elementlari soni) topiladi:

```
user = ("Erkin", 30, True)
print(len(user))  # 3
```

Kortej elementlariga *for* va *while* takrorlash operatorlari orqali murojaat quyidagicha amalga oshiriladi:

for orqali

```
user = ("Erkin", 30, True)
for u in user:
    print(u)
```

while orqali

```
user = ("Erkin", 30, True)
```

```
i=0
while i < len(user):
    print(user[i])
    i +=1
```

Kortejda biror elementning mavjudligini xuddi ro‘yxat lardagi kabi ***in*** operatoridan foydalanib amalga oshiriladi:

```
user = ("Erkin", 30, True)
if 'Erkin' in user: print("Foydalanuvchining
    ismi Erkin")
```

Murakkab kortejlar. Kortej o’z ichiga boshqa kortejlarni element sifatida qamrab olsa, bunday kortejlar murakkab kortejlar hisoblanadi:

```
davlatlar = (
                ("O‘zbekiston", 33.8,
    (("Toshkent", 2.65),
     ("Samarqand", 1.1),
     ("Urganch", 0.223))
     ),
       ("Qozog‘iston",
            17.5,
     (("Nur Sultan", 1.1),
      ("Olmaota", 2.3))
     ))
for davlat in davlatlar:
    nomi,aholi_soni,shaharlar =
    davlatprint(nomi,'-
    ',aholi_soni) print("Katta
    shaharlari:")
```

```
    for shahar in shaharlar:

        print(shahar[0],'-',shahar[1])
```

Bu yerda *davlatlar* korteji ikki elementdan tashkil topgan bo'lib, ular ham kortejlardir va davlat nomi, aholisi, katta shaharlari to'g'risidagi ma`lumotlarni ifodalaydi. O'z navbatida shaharlar ham yana kortejlardan tashkil topgan bo'lib, ularning har bir elementi shahar nomi va aholisini ifodalaydigan kortejdan tashkiltopgan. Xullas, jami ichma-ich joylashgan 4 ta kortejdan iborat berilganlarning murakkab strukturasiga misol keltirilgan.

Lug'atlar

Python dasturlash tilida ro'yxatlar va kortejlar bilan bir qatorda lug'atlar (dictinary) deb nomlanuvchi berilganlarning ichki tuzilmasi mavjud. Lug'atlar ham xuddi ro'yxatlar kabi elementlar to'plamini saqlaydi. Lug'atdagi har bir element unikal kalitga ega bo'ladi va unga biror bir qiymat bog'lanadi.

Lug'at quyidagicha sistaksis bo'yicha aniqlanadi:

dictionary = { kalit1:qiymat1, kalit2:qiymat2, }

Quyida lug'atlarga misol keltirilgan:

```
users = {1: "Tom", 2: "Bob", 3: "Bill"}


elements = {"Au": "Oltin", "Fe": "Temir", "H":
"Vodorod","O": "Kislorod"}
```

Bu yerda *users* ro'yxat ida kalit sifatida son, qiymat sifatida satr qo'llanilgan *element* ro'yxat ida esa qiymat sifatida ham kalit sifatida ham satr ishlatilgan.

Lekin kalitlar va qiymatlar bir turga mansub bo'lishi shart emas. Ular har xilturdagi qiymatlar bo'lishi mumkin:

```
objects = {1: "Tom", "2": True, 3: 100.6}
```

Bundan tashqari bo‘sh lug‘atlarni ham yaratish mumkin:

Ro‘yxatlar yordamida lug‘at xosil qilish. Lug‘atlar tuzilmaviy jihatidan ro‘yxat larga o‘xshamasada, lekin ba’zi bir maxsus ro‘yxatlar asosida *dict()* funkuyasi orqali ro‘yxatlar hosil qilish mumkin. Buning uchun ro‘yxat o‘z navbatida ro‘yxatlar to‘plamidan tashkil topgan bo’lishi kerak. Ichki ro‘yxatlar ikkita elementlardan tashkil topishi shart bo‘lib, mos ravishda birinchi element kalitga, ikkinchi element qiymatga akslantiriladi: Xuddi shu tarzda kortejlarni ham lug‘atlarga aylantirish mumkin. Buning uchun ikki o‘lchamli kortejning ichki kortejlari o‘z navbatida ikkitadan elementdantashkil topgan bo’lishi shart:

```
users_tuple = ( ("100", "Azim"),
    ("200", "Salim"),
    ("300", "Ali") )

users_dict = dict(users_tuple)

print(users_dict)  # {'100': 'Azim', '200':
                   # 'Salim', '300': 'Alibek'}
```

Lug‘at elementini o‘zgartirish. Lug‘at elementiga murojaat qilish uning kaliti yordamida amalga oshiriladi:

dictionary[*kalit*]

Masalan lug‘at elementiga murojaat qilish va uni o‘zgartirish quyidagichaamalga oshiriladi:

```
users = {
    "Bir": "Ali",
    "Ikki": "Vali",
    "Uch": "Sobir"  }
# Lug‘atning "Bir" kalitli elementiga murojaat uchun
```

```
print(users["Bir"])  # Ali

# Lug‘atdagi "Uch" kalitli element qiymatini
o'zgartiramiz

users["Uch"] = "Baxtiyor"

print(users["Uch"])  # Baxtiyor
```

Lug‘at elementiga kaliti orqali qiymat berganda shunday kalit lug‘atda mavjud bo’lmasa, u holda lug‘atga yangi element qo‘shiladi. Masalan, yuqoridagi misolda *users*[*"To‘rt"*] = *"Ibrohim"* tarzida yangi element qo’shishimiz mumkin, Chunki lug‘atda *“To’rt”* kalitli element mavjud emas.

Lekin, lug‘atda mavjud bo‘lmagan kalit orqali uning elementiga murojaat qilinganda, Python interpretatori *KeyError* turidagi istisno xatoligi yuzaga kelganligi haqida xabar chiqaradi. Masalan, yuqoridagi misol uchun *user* = *users*[*“Besh”*] kabi ishlatsak xatolik ro‘y beradi.

Bunday istisno xalotlarning oldini olish uchun Pythonda ***Kalit in Lug‘at*** ifodasidan foydalaniladi. Ushbu ifoda agarda shunday kalitli element lug‘atda mavjud bo‘lsa *True* qiymat, aks holda *False* qiymat qaytaradi, masalan:

```
bahoDict = {"A": 5, "B": 4, "C": 3}

key = "D"

if key in  bahoDict:
    baho =
    bahoDict[key]
    print(baho)
else:

    print("Element topilmadi") # Javob: Element
    topilmadi
```

Shu bilan birga, lug‘atning biror elementini olish uchun *get* metodidan ham

foydalanish mumkin bo’lib u ikki xil shaklda qo’llaniladi:

get(*key*) – lug‘atning *key* kalitli elementni qaytaradi. Agar lug‘atda *key* kalitli element mavjud bo’lmasa *None* qiymati qaytariladi.

get(key, default) - lug‘atning *key* kalitli elementni qaytaradi. Agar lug‘atda

key kalitli element mavjud bo’lmasa *default* qiymati qaytariladi.

Masalan:

```
bahoDict = {"A": 5, "B": 4, "C": 3}

key = "A"

baho =
bahoDict.get(key)
print(baho)  # 5
# yoki

key = "D"
baho = bahoDict.get(key, "Noma'lum
qiymat")print(baho)  # Noma'lum qiymat
```

Lug‘atdan elementni o‘chirish. Lug‘atdan kalit orqali elementni o‘chirish uchun *del* operatoridan foydalaniladi:

```
bahoDict = {"A": 5, "B": 4, "C": 3, "D": 2}
print(bahoDict) # {'A': 5, 'B': 4, 'C': 3, 'D': 2}
del bahoDict["C"]
print(bahoDict) # {'A': 5, 'B': 4, 'D': 2}
```

Shuni alohida ta’kidlash lozimki, agar lug‘atda bunday kalit mavjud bo‘lmasa *KeyError* istisno xatoligi yuzaga keladi. Ushbu xatolikni oldini olish uchun dastlab bunday kalit lug‘atda bor yoki yo‘qligini tekshirish tavsiya qilinadi:

```
bahoDict = {"A": 5, "B": 4, "C": 3, "D": 2}

baho = "A"

if baho  in  bahoDict:
    son =
    bahoDict[baho]del
    bahoDict[baho]
    print(son,
"o'chirildi")else:
    print("Element topilmadi")

# Javob: 5 o'chirildi
```

O‘chirishning boshqa bir metodi – *pop*() metodi orqali amalga oshiriladi. U ikkixil shaklda qo‘llaniladi:

pop(*key*) – *key* kaliti bo‘yicha elementni o‘chiradi va qiymat sifatida o’chirilgan elementni qaytaradi. Agar berilgan kalit bo‘yicha element topilmasa, *KeyError* istisno holati yuzaga keladi;

pop(*key*, *default*) – *key* kaliti bo‘yicha elementni o‘chiradi va qiymat sifatida o‘chirilgan elementni qaytaradi. Agar berilgan kalit bo‘yicha element topilmasa, *default* qiymati qaytariladi.

```
bahoDict = {"A": 5, "B": 4, "C": 3, "D": 2}

key = "A"

baho = bahoDict.pop(key)
print(baho)  # 5
# ikkinchi marta yana shu kalit bo`yicha o`chirishga urinamiz

baho2 = bahoDict.pop(key, "Bunday baho mavjud emas!")

print(baho2)  # Bunday baho mavjud emas!
```

Agar lug‘atdagi barcha elementlarni o‘chirish talab qilinsa, *clear*() metodidan foydalanish mumkin:

```
bahoDict = {"A": 5, "B": 4, "C": 3, "D": 2}

# Lug‘atning barcha elementlarini ekranga chiqaramiz

print(bahoDict)  # {'A': 5, 'B': 4, 'C': 3, 'D': 2}

bahoDict.clear()

# clear metodini qo'llagandan so'ng yana

# lug‘atning barcha elementlarini ekranga chiqaramiz

print(bahoDict)  # {}
```

Lug‘atlarni ko‘chirish va birlashtirish. Lug‘atlarni ko‘chirish uchun *copy()* metodidan foydalanilib, qiymat sifatida ushbu lug‘atning elementlaridan tashkil topgan boshqa lug‘at hosil qilinadi, masalan:

```
l = {"ismi": "Ali", "yoshi": 28}
l2 = l.copy()
print(l)  # {'ismi': 'Ali', 'yoshi': 28}
print(l2) # {'ismi': 'Ali', 'yoshi': 28}
```

Lug‘atlarni birlashtirish uchun *update()* metodidan foydalaniladi:

```
lugat = {"ismi": "Ali", "yoshi": 28}
lugat1 = {"kursi": 1,
"yo‘nalishi": "IO’M"}
lugat.update(lugat1)
print(lugat)# {'ismi': ‘Ali’, 'yoshi': 28,
          'kursi': 1,# 'yo‘nalishi': 'IAT'}
print(lugat1)    # {'ismi': 'Sardor', 'yoshi': 28}
```

Yuqoridagi holatda lugat1 tarkibi o‘zgarishsiz qoladi va `lugat` lug‘at tarkibiga boshqa lug‘at elementlari qo‘shiladi.

Lug‘at elementlariga murojaat. Lug‘at elementlariga murojaat uning kaliti

orqali amalga oshiriladi. Ayniqsa *for* operatori orqali lug‘at elementlarini uning kaliti orqali olish juda qulay hisoblanadi:

```
talabalar = {
    "010": "Ali",
    "011": "Vali",
    "012": "Sobir" }
for tal in talabalar:
    print(tal, " - ", talabalar[tal])
```

Javobga quyidagi natija chiqariladi:

```
    010 -  Ali
    011- Vali
    012-  Sobir
```

bu yerda *for* operatoridagi *t* o‘garuvchiga ketma – ket lug‘at kaliti qiymatlari yuklanadi (chapdan o‘nga qarab) va shu kalit orqali lug‘at elementiga murojaat amalga oshiriladi.

Lug‘at elementlariga murojaat qilishning yana bir metodi *items*() metodini qo’llash orqali amalga oshiriladi. Yuqoridagi dastur kodi *items()* metodi orqali quyidagicha yoziladi va ayni bir xil natijaga erishiladi:

```
talabalar = {
    "010": "Ali",
    "011": "Vali",
    "012": "Sobir" }
for nomer, ism in talabalar.items():
    print(nomer, " - ", ism)
```

items() metodi qiymat sifatida kortejlar to‘plamini qaytaradi. Har bir kortej elementi kalit (*nomer*) va qiymatlar (*ism*) juftligidan tashkil topadi.

Lug‘atdan faqat kalitlarini olish uchun *keys*() va faqat qiymatlarini olish uchun *values*() metodlaridan foydalaniladi, masalan:

```
talabalar = {
    "010": "Ali",
    "011": "Vali",
    "012": "Sobir" }
# lug‘atning kalitlariga murojaat
print("Kalitlar:")
for kalit in
    talabalar.keys():
    print(kalit, end='; ')
# lug‘atning qiymatlariga murojaat

print("\n Qiymatlar:")
for qiymat in
    talabalar.values():
    print(qiymat, end='; ')
```

Ushbu dastur ishga tushirilganda quyidagi javob ekranga chiqariladi:

```
 Kalitlar:
010; 011; 012;
Qiymatlar:
 Ali; Vali; Sobir;
```

Mustaqil bajarish uchun mashqlar:

1. Lug‘atni qiymatlari bo‘yicha saralang.
2. Lugatni kalitlar bo‘yicha saralang.
3. Lug‘atga yangi kalit va qiymat qo‘shing.
4. Berilgan uchta lug‘atni birlashtiruvchi dastur tuzung.
5. Berilgan qiymati lug‘atda bor yoki yo‘qligini tekshiruvchi funksiya yarating.
6. n butun soni berilgan. {1: 1, 2: 4, 3: 9, …, n: n2} ko‘rinishidagi lug‘at hosil qiling.
7. Berilgan lug‘atdagi qiymatlar yig‘indisini hisoblang.
8. Lug‘atdan berilgan kalit bo‘yicha qiymati o‘chiradigan dastur tuzing.
9. Berilgan lug‘atning bo‘sh yoki bo‘sh emasligini tekshiring.
10. Berilgan satrga ko‘ra quyidagicha lug‘at yarating:

s = `assalom`

d = {`a`: 2, `s`: 2, `l`: 1, `o`: 1, `m`: 1}

II. Fayllar bilan ishlash

Fayl faqat kompyuterda bitlar ketma-ketligi sifatida saqlanadigan ma’lumotlar to‘plamidir. Axborot ma’lumotlar to‘plamida (ma’lumotlar strukturasi) saqlanadi va "fayl nomi" deb ataladi.

Pythonda ikkita turdagi fayllar mavjud:

- Matn
- Binar

Matnli fayllar

Bu odamlar o'qiy oladigan tarkibga ega fayllar. Ular inson tushunadigan belgilar ketma-ketligini saqlaydi. Bloknot va boshqa standart muharrirlar ushbu fayl turini o‘qishi va tahrirlashi mumkin.

Matn ikki formatda saqlanishi mumkin: (.txt) - oddiy matn va (.rtf) - "boyitilgan matn formati".

Binar fayllar

Binar fayllar ma’lumotlarni kodlangan ko'rinishda ko'rsatadi (oddiy belgilar o'rniga faqat nol (0) va bir (1) yordamida). Ko'pgina hollarda, bu faqat bitlarning

ketma-ketligi. Ular *.bin* formatida saqlanadi

Fayldagi har qanday operatsiyani uchta asosiy bosqichga bo'lish mumkin:

1. Faylni ochish
2. Amaliyotni bajarish (yozish, o‘qish)
3. Faylni yopish

Fayllarni yaratish.

Python3da fayl bilan ishlash uchun avval uni yaratish kerak. Buni standart operatsion tizim vositalari yordamida kerakli katalogga o'tish va txt formatida yangi hujjat yaratish orqali amalga oshirish mumkin. Shu bilan birga, shunga o‘xshash holat Python dasturlash tilida ochiq metod yordamida ham amalga oshiriladi, bu fayl nomi va uni qayta ishlash rejimi parametrlari sifatida berilishi kerak.

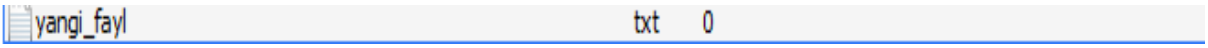
Quyidagi kod fayl o'zgaruvchisini yangi hujjatga ishora qilish uchun olishni ko'rsatadi. Agar ushbu dastur ishga tushirilsa, u manba kodi saqlanadigan papkada yangi_fayl.txt matn faylini yaratadi.

Agar yangi_fayl.txt nomli fayl kodli katalogda allaqachon mavjud bo‘lsa, dastur yangi hujjat yaratmasdan u bilan ishlashni davom ettiradi. Demak, fayl nomi ochiq metodning birinchi parametridir. Undan so'ng darhol ma’lumotlarni qayta

ishlash metodini ko'rsatadigan maxsus xabar keladi. Bunda “w” yozish, ya’ni faylga yozishni bildiradi.

Faylda har qanday manipulyatsiyani amalga oshirgandan so'ng, ma’lumotlar yo‘q olmasligini ta'minlash uchun uni yopish **`close()`** *funksiyasidan foydalangan holda yopish kerak.*

Oldingi misolda faylga kirish uchun nisbiy yo'l ishlatilgan, unda qattiq diskdagi ob’ektning joylashuvi haqida to'liq ma’lumot mavjud emas. Ularni o'rnatish uchun ochiq funksiyaga birinchi argument sifatida mutlaq yo'lni ko'rsatish kerak. Bunday holda, yangi_fayl.txt hujjati dastur papkasida emas, balki D diskda joylashgan bo'ladi.



2.1.Faylni ochish va yopish

Python o'rnatilgan open() funksiyasiga ega. Uning yordamida siz kompyuteringizda istalgan faylni ochishingiz mumkin. Texnik jihatdan Python uning asosida ob’ekt yaratadi. Sintaksis quyidagicha:

```
f = open(file_name, access_mode)
```

`file_name` = ochish uchun fayl nomi

`access_mode` = faylni ochish rejimi. Bu shunday bo‘lishi mumkin: o‘qish, yozish va hokazo. Agar boshqacha ko‘rsatilmagan bo‘lsa, standart rejim (r) o‘qiladi.

Python dasturlash tilida faylni ochish uchun ishlatiladigan maxsus belgilar haqidagi ma’lumot quyidagi jadvalda keltirib o’tilgan.

Belgi	Ma’nosi
***w* (*Write*)**	Fayl yozish uchun ochadi. Agar fayl yo‘q bo‘lsa, u hosil bo'ladi. Bunday fayl allaqachon mavjud bo‘lsa, u yangidan yaratiladi va shunga mos ravishda eski ma’lumotlar o'chiriladi
***r* (*Read*)**	- Fayl o‘qish uchun ochadi. Fayl topilmasa,

	FileNotFoundError xatolik qaytaradi;
***a* (*Append*)**	Faylni qayta yozish uchun fayl ochiladi. Agar fayl yo‘q bo‘lsa, u hosil bo'ladi. Bunday fayl allaqachon mavjud bo‘lsa, ma’lumotlar oxiridan yozish davom ettiriladi.
***b* (*Binary*)**	Binar fayllar bilan ishlash uchun foydalaniladi. *w* va r kabi rejimlar kombinatsiyasi bilan birgalikda ishlatiladi.
t (text)	matn rejimida ochish (standart)
+	o‘qish va yozish uchun bir vaqtning o‘zida ochiq

Ochish metodning ikkinchi argumentidan foydalanib, turli xil fayl rejimlarini birlashtira olish mumkin, masalan, ikkilik rejimda yozilgan ma’lumotlarni o‘qish uchun "rb" dan foydalanish mumkin. Yana bir misol: "r+" va "w+" o‘rtasidagi farq shundaki, ikkinchi holda, agar yo‘q bo‘lsa, yangi fayl yaratiladi. Birinchi holda, istisno tashlanadi. "r+" va "w+" dan foydalanish faylni o‘qish va yozish uchun ochadi.

Faylni yopish

Pythonda fayl ochilgandan so‘ng, uni yopish kerak. Shunday qilib, resurslar bo‘shatiladi va axlat chiqariladi. Faylni ochgandan keyin yopishning eng oson yo‘li `close()` metodidan foydalanishdir.

```
file = open("yangi_fayl.txt",'r')
file.close()
```

Yopilgandan so'ng, bu fayl qayta ochilmaguncha ishlatilmaydi.

Metodlar

Ochiq funksiya tomonidan qaytarilgan ob’ekt mavjud faylga havolani o‘z ichiga oladi. Shuningdek, u to‘rtta asosiy maydon shaklida taqdim etilgan, yaratilgan hujjat haqida ma’lumotni o‘z ichiga oladi. Ularning barchasi nomlari va ma’nolarini o‘z ichiga olgan quyidagi jadvalda tasvirlangan.

Nomi	Ma’nosi
name	fayl nomini qaytaradi

mode	ochilgan rejimni qaytaradi
closed	fayl yopiq bo‘lsa true ni, ochiq bo‘lsa true qiymatini qaytaradi
softspace	agar faylning chiqishi alohida bo‘sh joy belgisini o'z ichiga olmasa, true qiymatini qaytaradi

Fayl xususiyatlarini ko‘rsatish uchun kirish operatoridan, ya’ni nuqtadan foydalanish va keyin buni allaqachon tanish bo‘lgan chop etish `print()` funksiyasiga parametr sifatida o‘tkazish kifoya. Masalan:

Dastur kodi

```
file = open(r"D:\yangi_fayl.txt", "w")
print(file.name)
file.close()
```

Dastur natijasi

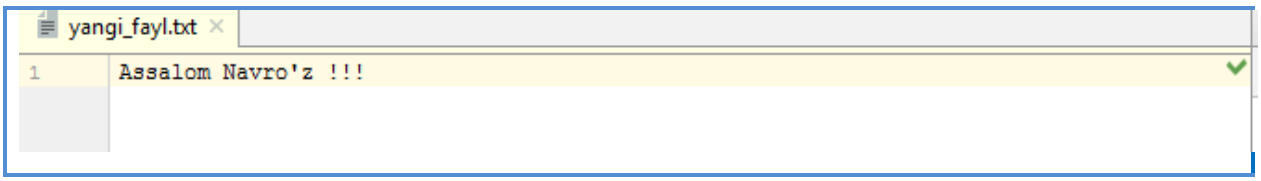
```
D:\yangi_fayl.txt
```

Python3da faylga yozish *w* (*Write*) metodi yordamida amalga oshiriladi. Metod mavjud faylga ishora qiluvchi ob’ektda chaqiriladi. Shuni esda tutish kerakki, buni amalga oshirish uchun avval hujjatni ochish funksiyasidan foydalanib ochish va “w” belgisi bilan yozish rejimini belgilash kerak. *w* (*Write*) metodi argument sifatida matn fayliga yoziladigan ma’lumotlarni oladi. Quyidagi kod misolida “Assalom Navro‘z” qatorining kiritilishi ko'rsatilgan.

Dastur kodi

```
file = open("yangi_fayl.txt", "w")
file.write("Assalom Navro'z !!!")
file.close()
```

Dastur natijasi

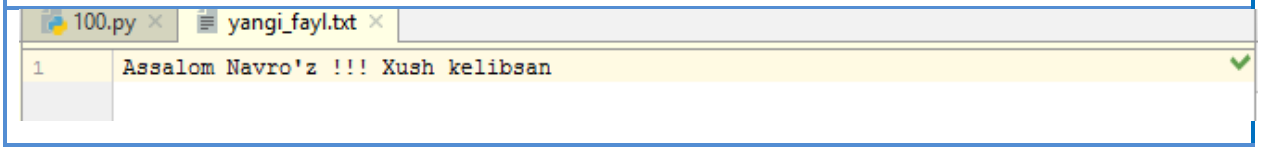


Agar oldin yozilgan ma’lumotlarga yangi ma’lumot qo‘shish kerak bo‘lsa, `open` funksiyani qayta chaqirish kerak, `open` funksiyasiga “a” belgisi qo‘shilishi kerak. Aks holda, **`yangi_fayl.txt`** faylidagi barcha ma’lumotlar butunlay o'chiriladi. Quyidagi kod misoli qo‘shimcha yozish uchun matnli hujjatni ochadi, shundan so‘ng unda "Xush kelibsan" satri boshi bo‘sh joy bilan joylashtiriladi. Shunday qilib, **`yangi_fayl.txt`** faylida “Assalom Navro‘z !!! Xush kelibsan” bo‘ladi. Bularning barchasidan so'ng, faylni majburiy yopish `close` qo‘ylishi shart.

Dastur kodi

```
file = open("yangi_fayl.txt", "a")
file.write(" Xush kelibsan")
file.close()
```

Dastur natijasi

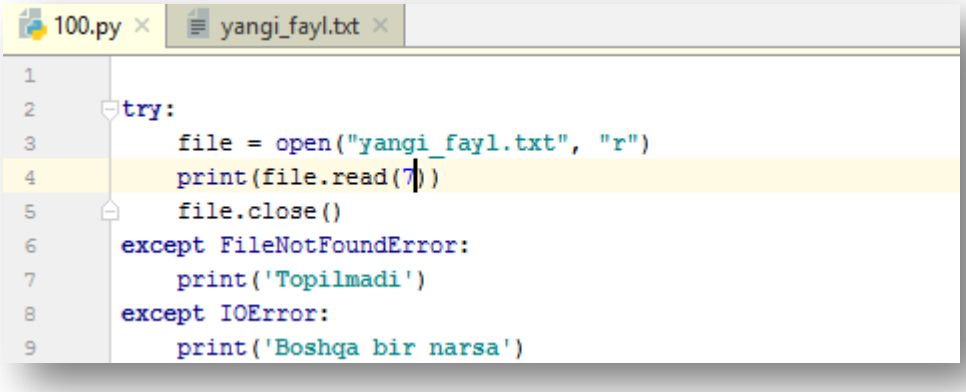


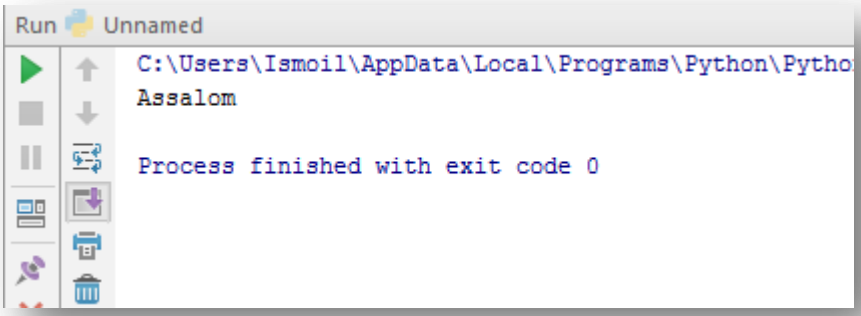
Matn fayliga ma’lumotlarni yozishning eng oddiy protsedurasi shunday amalga oshiriladi. Shuni ta'kidlash kerakki, Python dasturlash tilida hujjatlar bilan yanada ilg'or ishlash uchun juda ko‘p qo‘shimcha vositalar mavjud.

2.2.Faylni o‘qish

Python3da fayldan ma’lumotni o‘qish uchun mavjud hujjatga ishora qiluvchi ob’ektda *read* metodidan foydalanish kerak. Matnli faylni ochishda `open` funksiyaning ikkinchi parametri sifatida “ r ” ni belgilashni ham unutmaslik kerak. Quyidagi misolda *read* metodi **`yangi_fayl.txt`**dan `print` funksiyasiga ma’lumotlarni qaytaradi. Avvalgidek, dastur hujjatni yopish metodi bilan yopish bilan yakunlanadi. *read* metodi butun son parametrini ham qabul qilishi mumkin,

bu o‘qish uchun belgilar sonini uzatish uchun ishlatiladi. Misol uchun, agar siz 7 ni kiritsangiz, dastur faqat Assalom so‘zini o‘qiydi.





with as:

Matnli fayllarni qayta ishlashni biroz avtomatlashtirish uchun bir qator *with as* dan foydalanish tavsiya etiladi. Yopilishi kerak bo'lgan hujjatda `close` ni chaqirish zarurati yo‘q , chunki bu avtomatik ravishda sodir bo'ladi. Bularning barchasi **`yangi_fayl.txt`**dan ma’lumotlar o‘qiladigan quyidagi kod parchasida ko‘rsatilgan. Odatdagidek, ekranga satr ma’lumotlarini chop etish uchun `print` funksiyasi qo‘llaniladi.

Dastur kodi
`with open("yangi_fayl.txt", "r") as file:` `    print(file.read())`
Dastur natijasi
`Assalom Navro'z !!!`

Bundan tashqari, bu holda istisno bilan shug'ullanish kerak emas. Agar ko'rsatilgan nomga ega fayl bo'lmasa, with operatoridagi ichki kodli qatorlar bajarilmaydi. Python dasturlash tilining ko‘rib chiqilgan xususiyatlari yordamida foydalanuvchi ma’lumotlarni fayllarga o‘qish va yozishning asosiy operatsiyalarini osongina bajarishi mumkin.

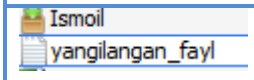
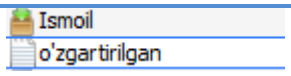
2.3. Pythonda fayl nomini o‘zgartirish

Ushbu misolda “yangilangan_fayl.txt” nomini “o‘zgartirilgan.txt” ga o‘zgartiriladi.

Dastur kodi

```
import os
# Faylning mutlaq yo'li
eski_nomi = r"d:\yangilangan_fayl.txt"
yangi_nomi = r"d:\o'zgartirilgan.txt"
# fayl nomini o'zgartirish
os.rename(eski_nomi, yangi_nomi)
```

Dastur natijasi

Fayl nomini o'zgartirishdan oldin	***Fayl nomini o'zgartirishdan keyin***
	

os.rename()

Misolda ko‘rsatilganidek, biz Pythonda os modulida *rename()* usuli yordamida fayl nomini o‘zgartirishimiz mumkin . Modul operatsion tizimlar bilan o‘zaro ishlash uchun funksiyalarni ta’minlaydi. Ushbu modul **os** Pythonning standart yordamchi modullari ostida keladi.

os.rename(src, dst, *, src_dir_fd=None, dst_dir_fd=None) os.rename()

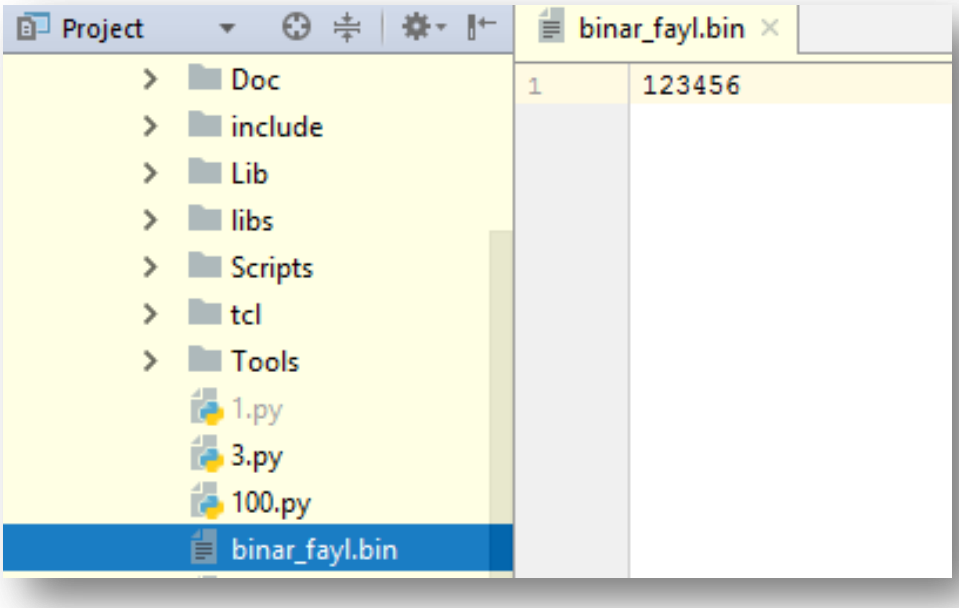
Quyida biz usul uchun o‘tishimiz kerak bo‘lgan parametrlar mavjud

src: Qayta nomlanishi kerak bo‘lgan faylning yo‘li

dst: Yangi nomi o‘zgartirilgan fayl uchun maqsad yo‘li

2.4.Binar fayllar

Binar fayllar *byte* tipidagi satrlardan foydalanadi. Bu shuni anglatadiki, fayldan ikkilik ma’lumotlarni o‘qishda *byte* tipidagi ob’ekt qaytariladi. Binar faylni ochish open() funksiyasi yordamida amalga oshiriladi, uning rejim parametrida “b” belgisi mavjud. Binar fayllarni ochish/yopish haqida batafsil ma’lumot bu yerda tasvirlangan. Matnli fayllardan farqli o‘laroq, binar fayllar “\n” qator oxiri konvertatsiyasini amalga oshirmaydi.



Dastur kodi

```
import pickle
fayl = open("binar_fayl.bin", 'rb')
f = fayl.read()
print("f = ", f)
print("f[5] = ", f[5])
print("f[0] = ", f[0])
print(bin(f[2]))
fayl.close()
```

Dastur natijasi

```
f =  b'123456'
f[5] =  54
```

```
f[0] =  49
0b110011
```

III.Ob’ektga yo‘naltirilgan dasturlash

Dasturiy ta'minotni ishlab chiqishda ob’ektga yo‘naltirilgan yondashuv tizimli dasturlash metodologiyasini ishonchli almashtirish uchun mo‘ljallangan edi. Bu allaqachon eskirgan kontseptsiyaga ko‘ra, har bir alohida dastur kodning funktsional bloklarining ierarxik tuzilishidir.

Ushbu xususiyat tufayli:

- Loyiha ustida ishlashda topshiriqni idrok etishni yaxshilaydi;
- Kod satrlari soni kamayadi;
- Kod yozishning murakkabligini pasaytiradi.

OOPning asosiy tamoyillari quyidagi mexanizmlardir: abstraktsiya, inkapsulyatsiya, Voris va polimorfizm. Ob’ektlar ko‘rinishidagi ma’lumotlarni qayta ishlaydigan dasturlarni yaratish uchun sizga tushunish kerak, shuningdek, barcha to‘rtta paradigmaga har tomonlama rioya qilish kerak.

OOPning asosiy tamoyillarini ko‘rib chiqing:

- ✓ Abstraktsiya OOP ning asosiy maqsadini amalga oshiradi, bu dasturchiga sinflardan foydalanish orqali turli xarakteristikalar va xatti-harakatlarga ega bo‘lgan muayyan turdagi ob’ektlarni shakllantirish imkonini beradi.
- ✓ Inkapsulyatsiya muayyan ob’ektlarni amalga oshirish tafsilotlarini yashirishga va ularning xususiyatlarini tashqi aralashuvdan himoya qilishga yordam beradi.
- ✓ Voris mavjud sinflarni ularning parametrlari va metodlarini avtomatik ravishda yangi ma’lumotlar tuzilmalariga o‘tkazish orqali kengaytirish imkonini beradi.
- ✓ Polimorfizm qo‘llaniladigan ob’ekt sinfiga qarab juda ko‘p turli xil ilovalarga ega bo‘lgan yagona interfeys yaratish uchun ishlatiladi.

  Sinf va ob’ekt yaratish

3.1. Pythonda sinflar va ob’ektlar:

Python tilidagi sinflar dasturlashda ob’ektga yo‘naltirilgan yondashuvning muhim qismidir. Sinf foydalanuvchi tomonidan belgilangan ma’lumotlar turini tavsiflaydi, uning asosida dasturda bir hil ob’ektlar yaratiladi. Qoida tariqasida, ular o‘zlarining hozirgi holatini, shuningdek xatti-harakatlarini amalga oshirishga imkon beradigan ba’zi xususiyatlar va metodlarni o‘z ichiga olishi mumkin.

Dasturingizda yangi sinfni aniqlash uchun siz class kalit so‘zini kiritasiz, so‘ngra yaratilayotgan ma’lumotlar strukturasi nomini ikki nuqta bilan tugatasiz. Quyidagi misol *element* nomli bo‘sh sinfni yaratishni ko‘rsatadi. Ko‘rib turganingizdek, unda hech qanday ma’lumot yo‘q.

Dastur kodi

```
class element:
    pass
element = element()
```

Python sintaksisi ba’zi ifodalar tanasiga ega bo‘lishini talab qiladi: sinf, funksiya, shart va boshqalar. Lekin ba’zida u yerda hech narsa bajarilmasligi kerak. Bunday holda, **pass** ishlatiladi.

`element` sinfining bo‘sh tanasiga qaramasdan, uning asosida noyob identifikatorga ega bo‘lgan aniq ob’ektni yaratish mumkin.

Yangi sinfni belgilash orqali uning asosida xohlagancha ko‘p ob’ektlar yaratish mumkin. Yuqorida aytib o‘tilganidek, bunday ma’lumotlar tuzilmasi ma’lum xususiyatlarni, ya'ni sinfning har bir misoli bilan ta’minlanadigan o‘zgaruvchilarni o‘z ichiga olishi mumkin. Quyida sinf va obyektining oddiy misoli keltirilgan. Misolda satr so‘zi va raqam raqami bilan `versiya` deb nomlangan sinf tasvirlangan.

Dastur kodi

```
class versiya:
    matn = "Python"
```

```
    t_raqam = 3.9
print(versiya.matn + " " + str(versiya.t_raqam))
```

Dastur natijasi

```
Python 3.9
```

Agar siz Data klassi asosida ob’ekt yaratsangiz, u ikkala o‘zgaruvchini ham, dastlab aniqlangan qiymatlarini ham oladi. Shunday qilib, ma’lumotlar ob’ekti yaratildi. Uning `matn` va `t_raqam` deb nomlangan maydonlariga nuqta operatori yordamida sinf misoli orqali qo‘ng‘iroq qilib kirishingiz mumkin. `print` funksiyasi ma’lumotlar ob’ekti maydonlarining qiymatlarini ekranga chop etishga yordam beradi. Shuni unutmangki, raqamni matn qiymati bilan birga chop etish uchun uni satr shakliga aylantirish kerak.

Maydonlarga qo‘shimcha ravishda, maxsus sinf uning barcha nusxalari bilan ta'minlanadigan metodlarni ham o‘z ichiga olishi mumkin. Yaratilgan ob’ekt orqali ma’lum bir metodning bajarilishini, ya'ni nuqta yordamida chaqirishmumkin. Ushbu misol matnni ekranga chop etuvchi `salom_alik` funksiyasiga ega `tur` sinfini ko‘rsatadi.

Dastur kodi

```
class tur:
    def salom_alik(self):
        print("Assalomu-alaykum!")
turlar = tur()
turlar.salom_alik()
```

Dastur natijasi

```
Assalomu-alaykum!
```

Mustaqil bajarish uchun mashqlar:

Sodda sinflarga doir misollar

1. Ixtiyoriy ikkita son berilgan. Ularning yig‘indisini, ayirmasini, ko‘paytmasini va nisbatini toping.

2. To’g‘ri burchakli uchburchakning katetlari a va b. Uchburchakning yuzi va perimetrini toping.

3. Radiusi R bo‘lgan aylananing uzunligini va doiraning yuzini toping.

4. Doiraning yuzi S. Uning radiusini toping.

5. A(x1,y1) va B(x2,y2) nuqtalar koordinatalari bilan berilgan. Ular orasidagi masofani toping.

6. Tekislikda uchburchak uchlarining koordinatasi A(x1,y1) , B(x2,y2) va S(x3,y3). Uchburchakning yuzini toping.

7. Ikki sonning o‘rta arifmetik va o‘rta geometrik qiymatlarini toping.

8. To‘g‘ri to‘rtburchak diagonali uchlarining koordinatalari berilgan. Uning yuzini toping.

9. Uch xonali butun son berilgan. Bu son raqamlarining yig’indisini va ko‘paytmasini toping.

3.2.Self argumenti

`Self` nima uchun kerakligini va Python funksiyalarida nimani anglatishini ko‘rib chiqaylik. Ko‘rib turdiki, sinfdagi metodning yagona atributi `self` kalit so‘zidir. Joriy ob’ektga bog‘lash uchun uni har bir funksiyaga qo‘yish kerak. Shuningdek, ushbu kalit so‘zdan foydalanib, tavsiflangan metodda sinf maydonlariga kirish mumkin. Shunday qilib, `Self` ob’ektning identifikatorini almashtiradi.

Dastur kodi

```
class telefon:
    nom = "T E L E F O N"
    ovoz = " jiring!"
    def ovoz_toni(self):

        print(self.nom +self.ovoz + self.ovoz + " degan
ovoz chiqaradi")
ovozlar = telefon()
```

```
ovozlar.ovoz_toni()
```

Dastur natijasi

```
T E L E F O N jiring! jiring! degan ovoz chiqaradi
```

Yuqori qismida telefonni tasvirlaydigan `telefon` sinfi joylashgan. Unda mos ravishda "T E L E F O N" "jiring!" boshlang'ich qiymati bo'lgan `nom` va `ovoz` o'zgaruvchilar mavjud. `ovoz_toni` funksiyasi telefonni ekranda tegishli xabarni ko'rsatish orqali jiringlashga olib keladi. Buning uchun `print` () funksiyasi `nom` va `ovoz` o'zgaruvchilariga kirishdan foydalanadi. Keyinchalik, `telefon` sinfining namunasini yaratish va unga `ovoz_toni`ni bog'lash kerak.

3.3.Konstruktor

Konstruktor sinf ob'ektini yaratish uchun ishlatiladi. Shunday qilib, yuqorida, biz `tur` sinfining ob'ektlarini yaratganimizda, biz hech qanday parametrlarni qabul qilmaydigan va barcha sinflar bilvosita ega bo'lgan standart konstruktordan foydalandik:

Biroq, `__init__()` deb nomlangan maxsus metod yordamida sinflarda konstruktorni aniq belgilashimiz mumkin (har bir tomonda ikkita chiziqcha). Misol uchun, `tur` sinfini unga konstruktor qo'shish orqali o'zgartiramiz:

Dastur kodi

```
class tur:
    def __init__(self):
        print("Assalomu-alaykum!")
tur()
```

Dastur natijasi

```
Assalomu-alaykum!
```

Endi ob'ekt yaratishda: `tur` sinfidan `__init__()` konstruktori chaqiriladi, u konsolga `Assalomu-alaykum!` qatorini chop etadi.

3.4.Destruktor

Obyekt bilan ishlash tugagandan so‘ng del operatorini ishlatish orqali uni xotiradan o‘chirib tashlash mumkin:Shuni eslatish joizki, bu ishni qilish shart emas, ya‘ni, skrip ishlashi tugashi bilanham obyektlar xotiradan o‘chiriladi. Bundan tashqari, `__del__` ichki metodini aniqlash orqali sinfdagi destruktorni qayta aniqlash mumkin. Bu metod del operatori ishlaganda yoki ob’ektlar xotiradan avtomatik o‘chganda ishlaydi. Masalan:

Dastur kodi

```
class talaba:
# Konstruktor


    def __init__(self, ismi):
        self.ismi = ismi # Ismni o'rnatamiz
    def __del__(self):
        print(self.ismi, "Sizning ismingiz xotiradan
o'chdi")


    def talabgor(self):
        print("Salom ", self.ismi, "xush kelibsiz")
a = input('ismingiz nima?')
talaba = talaba(a)
talaba.talabgor()
del talaba # Xotiradan o'chirish
```

Dastur natijasi

```
ismingiz nima? Ismoil
Salom  Ismoil xush kelibsiz !
Ismoil sizning ismingiz xotiradan o'chdi
```

3.5.Inkapsulyatsiya

Sinflardagi atributlar ommaviydir, ya'ni dasturning istalgan joyidan biz ob’ektning atributini olishimiz va uni o'zgartirishimiz mumkin. Masalan:

Dastur kodi

```
class mavjudot:
    def __init__(self, laqabi):
        self.laqabi = laqabi  # Laqab joylashtiramiz
        self.mavjudot_yoshi = 15  # Yoshini
belgilaymiz

    def namoyish(self):
        print(f"Laqabi: {self.laqabi}\nYoshi:
{self.mavjudot_yoshi}")

nom = mavjudot("Layka")
nom.laqabi = "Vaska"  # nom atributini o'zgartiring

nom.mavjudot_yoshi = 45  # yosh atributini
o'zgartirish
nom.namoyish()
```

Dastur natijasi

```
Laqabi: Vaska
Yoshi: 45
```

Ammo bu holda, biz, masalan, odamning yoshi yoki ismiga noto‘g‘ri qiymat berishimiz mumkin, masalan, salbiy yoshni ko'rsatishimiz mumkin. Ushbu xatti-harakatlar nomaqbuldir, bu ob’ekt atributlariga kirishni nazorat qilish masalasini ko‘taradi.

Inkapsulyatsiya ob’ektga yo'naltirilgan dasturlashda asosiy tushunchadir. To‘g‘ridan-to‘g‘ri Python dasturlash tilida inkapsulyatsiyaga kelsak, siz sinf

atributlarini xususiy qilib yashirishingiz va ularga xususiyatlar deb ataladigan maxsus metodlar orqali kirishni cheklashingiz mumkin.

Yuqorida tavsiflangan sinfni undagi xususiyatlarni belgilash orqali o‘zgartiramiz:

Dastur kodi

```
class askar:
    def __init__(self, ism):
        self.__ism = ism  # ism joylash
        self.__yosh = 19  # yoshini belgilash
    def yosh_oralig(self, yosh):
        if 18 < yosh < 27:
            self.__yosh = yosh
        else:
            print("Yoshingiz to'g'ri kelmaydi")

    def yosh_nusxa(self):
        return __self.yosh

    def ismdan_nusxa(self):
        return __self.ism

    def namoyish(self):
        print(f"Ismi: {self.__ism}\nYoshi:
{self.__yosh}")
nom = askar("Botir")
nom.namoyish()  # Ismi: Botir  Yoshi: 19
nom.yosh_oralig(29)  # Yoshingiz to'g'ri kelmaydi
nom.yosh_oralig(22)
nom.namoyish()  # Ismi: Botir  Yoshi: 22
```

Dastur natijasi

```
Ismi: Botir
Yoshi: 19
Yoshingiz to'g'ri kelmaydi
Ismi: Botir
Yoshi: 22
```

Xususiy atribut yaratish uchun uning nomining boshiga qo‘sh chiziqcha qo‘yiladi: self.__ismi. Biz bunday atributga faqat bir sinfdan kira olamiz. Ammo biz ushbu sinfdan tashqarida murojat qilolmaymiz. Masalan, ushbu atributga qiymat berish hech narsa qilmaydi:

Dastur kodi

```
nom.__yosh=43
print(nom.__yosh)
```

Dastur natijasi

```
43
```

Chunki bu holda yangi __yosh atributi oddiygina dinamik tarzda aniqlanadi, lekin uning self.__yosh atributiga hech qanday aloqasi yo‘q .

Biroq, biz hali ham foydalanuvchining yoshini tashqaridan belgilashimiz kerak bo‘lishi mumkin. Xususiyatlar aynan shu maqsadda. Bitta xususiyatdan foydalanib, biz atributning qiymatini olishimiz mumkin:

```
def yosh_nusxa(self):
    return __self.yosh
```

Bu metod ko‘pincha oluvchi yoki *aksessuar* deb ham ataladi. Yoshni o‘zgartirish uchun boshqa xususiyat belgilangan:

```
def yosh_oralig(self, yosh):
    if 18 < yosh < 27:
        self.__yosh = yosh
    else:
        print("Yoshingiz to'g'ri kelmaydi")
```

Ushbu metod getter yoki aksessor deb ham nomlanadi. Bu yerda biz allaqachon shartlarga qarab, yoshni tiklash yoki yo‘q ligini hal qilishimiz mumkin.

Har bir xususiy atribut uchun bunday juft xususiyatlarni yaratish shart emas. Shunday qilib, yuqoridagi misolda biz shaxsning ismini faqat konstruktordan belgilashimiz mumkin. Va qabul qilish uchun `ismdan_nusxa` metodi aniqlanadi.

3.6.Vorislik

Voris mavjud sinf asosida yangi sinf yaratish imkonini beradi. Inkapsulyatsiya bilan bir qatorda voris ob'ektga yo'naltirilgan dasturlashning asoslaridan biridir.

Vorisning asosiy tushunchalari subklass va supersinfdir. Subklass barcha umumiy atributlar va metodlarni yuqori sinfdan voris qilib oladi. Yuqori sinf, shuningdek, asosiy (asosiy sinf) yoki ota (ota sinf) deb ataladi va pastki sinf - olishan (hosil bo'lgan sinf) yoki pastki (bola sinf) deb ataladi.

Sinf vorisi sintaksisi quyidagicha:

```
class sinfostisi (supersinf):
    sinfostisi_azolari
```

Masalan, bizda shaxsni ifodalovchi talaba sinfi mavjud:

```
class abiturent:
    def __init__(self, ism):
        self.__ism = ism

    @property
    def ism(self):
        return self.__ism

    def namoyish(self):
        print(f"Ismi: {self.__ism} ")
```

Aytaylik, bizga qandaydir oliygohda ta'lim olayotgan talabalar sinfi kerak. Biz noldan yangi sinf yaratishimiz mumkin, masalan, talabalar sinfi:

```
class talabalar:
    def __init__(self, ism):
        self.__ism  = ism  #

    @property
    def ism(self):
        return self.__ism

    def display_info(self):
        print(f"Ismi: {self.__ism} ")

    def uqish(self):
        print(f"{self.ism} o‘qish")
```

Biroq, `talabalar` klassi `abiturent` sinfi bilan bir xil atribut va metodlarga ega bo'lishi mumkin, chunki talaba insondir. Shunday qilib, yuqoridagi `talabalar` sinfida faqat `uqish` metodi qo'shiladi, kodning qolgan qismi `talaba` sinfining funksionalligini takrorlaydi. Ammo bir sinfning funksionalligini boshqasida takrorlamaslik uchun, bu holda *vorislik*dan foydalanish yaxshiroqdir.

Shunday qilib, keling, `abiturent` sinfidan `talabalar` sinfini voris qilib olaylik:

Dastur kodi

```
class abiturent:
    def __init__(self, ism):
        self.__ism = ism
    @property
    def ism(self):
        return self.__ism
```

```
    def namoyish(self):
        print(f"Ismi: {self.__ism} ")

class talabalar(abiturent):
    def uqish(self):
        print(f"{self.ism} talaba")

nom = talabalar("Islom")
print(nom.ism)
nom.namoyish()
nom.uqish()
```

Dastur natijasi

```
Islom
Ismi: Islom
Islom talaba
```

`talabalar` sinfi `abiturent` sinfining funksionalligini to‘liq o‘z zimmasiga oladi, faqat `uqish()` metodini qo‘shadi. Shunga ko'ra, `talabalar` ob’ektini yaratishda biz `abiturentdan` voris qilib olishan konstruktordan foydalanishimiz mumkin:

```
nom = talabalar("Islom")
print(nom.ism)
nom.namoyish()
```

Shuningdek, voris qilib olishan atributlar xususiyatlar va metodlarga kirishingiz mumkin. Biroq, `talabalar` __name turidagi shaxsiy atributlarga ega emasligini unutmang. Masalan, biz `uqish` metodida `self.__name` xususiy atributiga murojaat qila olmaymiz:

```
def uqish(self):
```

```
    print(f"{self.__ism} talaba")# xatolik sodir
bo’ladi
```

3.7. Polimorfizim

Pythonda polimorfizm nima?

Python tilidagi polimorfizm ob’ektning ko‘p shakllarni olish qobiliyatidir. Oddiy so‘zlar bilan aytganda, polimorfizm bizga bir xil harakatni turli yo‘llar bilan bajarishga imkon beradi.

Misol uchun, Bobur ofisda bo'lganida xodim sifatida ishlaydi. Biroq, u uyda bo'lganida, u xotin kabi harakat qiladi. Bundan tashqari, u turli joylarda o‘zini boshqacha namoyon qiladi. Shuning uchun, bir xil odam vaziyatga qarab turli shakllarni oladi. Polimorfizm asosan voris bilan qo'llaniladi. Vorisda bolalar sinfi ota-sinfning atributlari va metodlarini voris qilib oladi . Mavjud sinf asosiy sinf yoki ota-sinf deb ataladi va yangi sinf , bola sinf yoki hosila sinf deb ataladi.

Sinfdagi polimorfizm metodlari

Sinf metodlari bilan polimorfizm bir xil metodga ega bo‘lgan turli ob’ektlarni guruhlashda foydalidir. Biz ularni ro‘yxatga yoki kortejga qo‘shishimiz mumkin va ularning metodlarini chaqirishdan oldin ob’ekt turini tekshirishimiz shart emas. Buning o'rniga, Python ish vaqtida ob’ekt turini tekshiradi va to‘g‘ri metodni chaqiradi. Shunday qilib, biz har bir ob’ekt qaysi sinf turiga tegishli ekanligi haqida tashvishlanmasdan metodlarni chaqirishimiz mumkin. Ushbu metodlar har bir sinfda mavjud deb taxmin qilamiz.

Python turli sinflarga bir xil nomdagi usullarga ega bo‘lish imkonini beradi.

Keling, ikki yoki undan ortiq sinflarga bir xil usullarni qo‘shish orqali boshqa sinfni xuddi shu tarzda loyihalashtiramiz.

- Keyin har bir sinfning ob’ektini yarating
- Keyin, barcha ob’ektlarni kortejga qo'shing .
- Oxir-oqibat, ob’ektning sinfini tekshirmasdan
- for sikli va chaqiruv usullaridan foydalangan holda kortejni takrorlang.

Misol. Quyidagi misolda `yoqilgi()` va `max_tezlik()` ikkala sinfda yaratilgan misol usullari.

Dastur kodi

```
class Ferrari:
    def yoqilgi(self):
        print("Benzin")

    def max_tezlik(self):
        print("Maksimal tezlik: 350")

class BMW:
    def yoqilgi(self):
        print("Diesel")

    def max_tezlik(self):
        print("Maksimal tezlik: 240")

ferrari = Ferrari()
bmw = BMW()

# bir xil turdagi ob'ektlarni takrorlash
for car in (ferrari, bmw):

    car.yoqilgi()
    car.max_tezlik()
```

Dastur natijasi

```
Benzin
Maksimal tezlik: 350
Diesel
```

```
Maksimal tezlik: 240
```

Xulosa qildigan bo‘lsak, biz Ferrari va BMW ikkita sinfini yaratdik. Ular bir xil misol usuli nomlariga ega `yoqilgi()` va `max_tezlik()`. Biroq, biz ikkala sinfni ham bog‘lamadik va vorislikdan foydalanmadik.

Biz ikkita turli ob’ektni kortejga joylashtirdik va uni avtomobil o‘zgaruvchisi yordamida takrorladik. Bu polimorfizm tufayli mumkin, chunki biz ikkala sinfga bir xil usulni qo‘shdik, Python avval ob’ektning sinf turini tekshiradi va o'z sinfida mavjud bo‘lgan usulni bajaradi.

Funksiya va ob’ektlar orqali polimorfizmdan foydalanish

Har qanday ob’ektni parametr sifatida qabul qila oladigan va sinf turini tekshirmasdan uning usulini bajaradigan funksiya bilan polimorfizm yaratishimiz mumkin. Bundan foydalanib, biz takroriy usul chaqiruvlari o‘rniga bir xil funksiyadan foydalangan holda ob’ekt harakatlarini chaqirishimiz mumkin.

Dastur kodi

```
class Ferrari:
    def yoqilgi(self):
        print("Benzin")

    def max_tezlik(self):
        print("Maksimal tezlik: 350")

class BMW:
    def yoqilgi(self):
        print("Diesel")

    def max_tezlik(self):
        print("Maksimaltezlik: 240")

ferrari = Ferrari()
```

```
bmw = BMW()
def car_details(obj):
    obj.yoqilgi()
    obj.max_tezlik()

ferrari = Ferrari()
bmw = BMW()
car_details(ferrari)
car_details(bmw)
```

Dastur natijasi

```
Benzin
Maksimal tezlik: 350
Diesel
Maksimal tezlik: 240
```

Mustaqil bajarish uchun mashqlar:

1. Hech qanday o'zgaruvchi va usullarsiz Avtomobil sinfini yarating

2. Vehicle sinfining barcha o‘zgaruvchilari va usullarini voris qilib oladigan bolalar sinfi avtobusini yarating.

3. Har bir sinf misoli (ob'ekti) uchun bir xil qiymatga ega bo'lishi kerak bo'lgan xususiyatni belgilang.

4.Ob’ekt turini tekshiring.

5.School_bus ham Vehicle sinfining namunasi ekanligini aniqlang.

3.8. Sana va vaqt bilan ishlash

datetime moduli

Sana va vaqt bilan ishlaydigan quyidagi asosiy funksiyalar datetime modulida jamlangan:

- date;
- time;
- datetime.

O‘rnatilgan va uchinchi tomon modullari yordamida Pythonda joriy sana va vaqtni olishning ko‘plab usullari mavjud. Quyidagi qadamlar sana va vaqt moduli yordamida joriy sana va vaqtni qanday olish mumkinligini ko‘rsatadi.

1. Datetime modulini import qilish

Pythonning *datetime* moduli sana va vaqtni o‘z ichiga olgan ko‘plab murakkab funksiyalarni bajaradigan funksiyalarni taqdim etadi. Bayonot *datetime* yordamida sinfni import qilish.

from datetime import datetime

Dastur kodi

```
from datetime import datetime
vaqt = datetime.now()
print('Joriy sana vaqti:', vaqt)
print('Turi:', type(vaqt))
```

Dastur natijasi

```
Joriy sana vaqti: 2022-04-12 10:18:57.196632
Turi: <class 'datetime.datetime'>
```

Natijada joriy sana va vaqtni quyidagi formatda oldik.

```
YYYY-MM-DD SS:MM:SS.MS
```

2. Datetime sinfining now() funksiyasidan foydalanish

Joriy *datetime.now()* mahalliy sana va vaqtni qaytaradi. Odatiy bo‘lib, YYYY-mm-dd hh:mm:ss.microsecondsformatda sana vaqtini ifodalaydi.

Pythonda sana va sana vaqti ob’ektlardir. Shunday qilib, biz sana va vaqtni boshqarayotganimizda, bu biz aslida ob’ektlar bilan ishlayotganimizni anglatadi. Masalan, joriy sana va vaqtni *datetime* ob’ektdan alohida ajratib olish mumkin.

Dastur kodi

```
# faqat datetime sinfini import qilish
from datetime import datetime
```

```
# joriy   sana vaqti
now = datetime.now()
joriy_sana = now.date()
print('sana:', joriy_sana)
print("TURI",type(joriy_sana))


joriy_vaqt = now.time()
print('vaqt', joriy_vaqt)
print("TURI",type(joriy_vaqt))
```

Dastur natijasi

```
sana: 2022-04-12
TURI <class 'datetime.date'>
vaqt 10:33:00.255068
TURI <class 'datetime.time'>
```

3. *Date sinfining today() funksiyasidan foydalanish*

Vaqtni emas, faqat joriy sanani xohlasangiz, ushbu qadamdan foydalaning. Sana sinfining today()usuli joriy mahalliy sanani qaytaradi. Python *Datetime* moduli sanalarni ifodalash va boshqarish uchun ***Date*** sinfini taqdim etadi. ***Date*** sinfi ***Grigorian*** kalendarini hisobga oladi.

- ✓ *Datetime* modulidan *date*sinfni import qilish
- ✓ *date.today()* joriy sanani olish uchun usuldan foydalanish.

Dastur kodi

```
from datetime import date
bugungi_kun = date.today()
print('Bugungi sana:', bugungi_kun)
```

Dastur natijasi

```
Bugungi sana: 2022-04-12
```

4. *Vaqt modulidan foydalanish*

time.time() funksiyadan joriy vaqtni suzuvchi nuqtali raqam sifatida davrdan keyingi soniyalarda olish uchun foydalaniladi.

Datetime moduli ***yil, oy, kun, soat, daqiqa, soniya*** kabi induvial komponentga kirish uchun bir nechta atributlarni taqdim etadi.

Misol :Ushbu misolda biz joriy sana vaqtini ajratamiz va ularni yil, oy, kun, soat, daqiqa, soniya va mikrosekundlar kabi o‘zgaruvchilarga ajratamiz.

Dastur kodi

```
from datetime import datetime
# Joriy sana va vaqtni olish
now = datetime.now()
# atributlarni chiqarib tashlash
print("Yil:", now.year)
print("Oy:", now.month)
print("Kun =", now.day)

print("Soat:", now.hour)
print("Minut:", now.minute)
print("Sekund:", now.second)
print("Millisekund:", now.microsecond)
```

Dastur natijasi

```
Yil: 2022
Oy: 4
Kun = 12
Soat: 10
Minut: 41
Sekund: 31
Millisekund: 887176
```

Joriy sana va vaqtni olish uchun Python quyidagi funksiyalaridan foydalanish mumkin.

№	Funksiya	Tavsif
1	`datetime.now()`	Vaqt mintaqasi haqida ma’lumotsiz joriy mahalliy sana vaqtini olish
2	`date.today()`	Joriy sanani olish
3	`time.time()`	Joriy vaqtni soniyalarda olish. U 1970-yil 1-yanvar 00:00:00 dan keyingi soniyalar sonini qaytaradi.
4	`time.ctime()`	Joriy vaqtni odam o‘qiy oladigan formatda olish
5	`time.localtime()`	struct_time A formatida mahalliy vaqtgacha bo'lgan davrdan soniyalarda ifodalangan joriy vaqtni olish
6	`int(time.time() *1000)`	Joriy vaqtni millisekundlarda olish
7	`datetime.now(timezone.utc)`	Joriy UTC vaqtini olish
8	`time.gmtime(time.time())`	Joriy GMT vaqtini olish
9	`datetime.now().isoformat()`	Joriy sana vaqtini ISO formatida olish
10	`datetime.now(pytz.timezone('tz_name'))`	
	Muayyan vaqt mintaqasida joriy vaqtni olish	

3.9. Timedelta moduli

timedelta ikki sana, vaqt yoki sana va vaqt misollari o‘rtasidagi mikrosekundlik piksellar sonigacha bo‘lgan farq bo'lgan davomiylikni ifodalaydi **.**

Sinf *timedelta* Pythonning datetime modulida mavjud . Belgilangan sana va vaqtdan haftalar, kunlar, soatlar, daqiqalar, soniyalar, mikrosoniyalar va millisekundlarni **qo'shish yoki ayirish***timedelta* uchun foydalanish mumkin .

Misol : Ikki sana orasidagi farqni hisoblansin.

Dastur kodi

```
from datetime import datetime


# sana vaqti berilgan
joriy_sana = datetime.now()
x_sana_vaqti = datetime(year=2022, month=4, day=4,
hour=12, minute=30)


# Ikki sana orasidagi farq
# Timedelta olish
timedelta = joriy_sana - x_sana_vaqti


print(timedelta)
print(type(timedelta))
```

Dastur natijasi

```
8 days, 1:22:00.480759
<class 'datetime.timedelta'>
```

Misol : Kelajakdagi sana vaqtini hisoblang.

Keling , ma’lum bir sanaga **to‘rt hafta qo‘shib,***timedelta* kelajakdagi sanalarni hisoblash uchun sinfdan qanday foydalanishni ko‘rib chiqaylik.

Dastur kodi

```
from datetime import datetime, timedelta
kelajak_sana = datetime.now()
print('Berilgan sana:', kelajak_sana)
# belgilangan sanaga 4 hafta qo'shing

berilgan_sana = kelajak_sana + timedelta(weeks=4)
print('Kelajak sana::', berilgan_sana)
```

Dastur natijasi

```
Berilgan sana: 2022-04-12 13:58:43.346077
Kelajak sana:: 2022-05-10 13:58:43.346077
```

Pythonda kelajak va o‘tgan sanalarni hisoblash uchun berilgan sanadan **haftalarni qo‘shish yoki ayirish** uchun *timedelta* sinfining atributidan foydalanishimiz mumkin.

Misol.

Dastur kodi

```
from datetime import datetime, timedelta


joriy_sana = datetime.now()
print("Joriy sana va vaqt : ", joriy_sana)

# 6 haftani ayirish
utgan_sana = joriy_sana - timedelta(weeks=6)
print("O'tgan sana: ", utgan_sana)

# 2 hafta qo'shiladi
kelajak_sana = joriy_sana - timedelta(weeks=2)
print("Kelajak sanasi: ", kelajak_sana)
```

Dastur natijasi

```
Joriy sana va vaqt :  2022-04-12 14:08:03.432087


O'tgan sana:  2022-03-01 14:08:03.432087
Kelajak sanasi:  2022-03-29 14:08:03.432087
```

Mustaqil bajarish uchun mashqlar:

1. Python-da joriy sana va vaqtni chop eting
2. satrni datetime ob'ektiga aylantiring
3. Python-da berilgan sanadan haftani (7 kun) ayirish
4. Sanani quyidagi formatda chop eting
5. Berilgan sananing hafta kunini toping
6. Belgilangan sanaga bir hafta (7 kun) va 12 soat qo'shing
7. Joriy vaqtni millisekundlarda chop eting
8. Quyidagi sana vaqtini satrga aylantiring
9. Joriy kundan boshlab 4 oylik sanani hisoblang
10. Ikki berilgan sana orasidagi kunlar sonini hisoblang

Foydalanilgan adabiyotlar

1. Пол Берри, Изучаем программирование на Python, Москва 2017.

2. Бизли Д. М. Язык программирования Python : справочник : пер. с англ. / Д. М. Бизли. – Киев : ДиаСофт, 2000.

3. Гифт Н. Python в системном администрировании UNIX и Linux : пер. с англ. / Н. Гифт, Д. Джонс. – СПб. : СимволПлюс, 2009.

4. Лейнингем И. Освой самостоятельно Python за 24 часа : пер. с англ. / И. Лейнингем. – М. : Издательский дом «Вильямс»,2001.

5. Лесса А. Python. Руководство разработчика : пер. с англ. / А. Лесса. – СПб. : ДиасофтЮП, 2001.

6. Лутц М. Изучаем Python : пер. с англ. / М. Лутц. – СПб. : Символ-Плюс, 2009.

7. Лутц М. Программирование на Python : пер.с англ. / М. Лутц. – СПб. : Символ-Плюс, 2002.

8. Саммерфельд М. Программирование на Python 3. Подробное руководство : пер. с англ. / М. Саммерфельд. – СПб. : Символ-Плюс, 2009.

9. Сузи Р. А. Python / Р. А. Сузи. – СПб. : БХВ-Петербург, 2002. Сузи Р. А. Язык Python и его применения : учеб. пособие /

10. Р.А. Сузи. – М. : Интернет-Университет информационных технологий: БИНОМ. Лаборатория знаний, 2006.

11. Язык программирования Python / Г. Россум [и др.]. – СПб. : АНО «Институт логики» – Невский диалект, 2001.

12. Saidov D.Y., Python dasturlash tili o‘quv – uslubiy qo‘llanma, Toshkent 2019.

Foydalanilgan internet saytlari

1. https://pynative.com/python-date-and-time-exercise/
2. https://www.w3schools.com/
3. https://metanit.com/python/
4. https://realpython.com/
5. https://all-python.ru/