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

NAMANGAN MUXANDISLIK - QURILISH ISNTITUTI

“AXBOROT TIZIMLARI VA TEXNOLOGIYALARI” KAFEDRASI

M. TO‘XTASINOV

“Python dasturlash tili” fani bo’yicha

O’ Q U V –U S L U B I Y M A J M U A

NAMANGAN - 2024

O’ZBEKISTON RESPUBLIKASI
OLIY TA’LIM, FAN VA INNOVATSIYALAR VAZIRLIGI
NAMANGAN MUHANDISLIK - QURILISH INSTITUTI

NamMQI
O'quv-uslubiy boshqarma

Ro’yxatga olindi:
№ _____
202_ yil «___» _____
5
«30» 09 2024 y.

“TASDIQLAYMAN”
O’quv ishlari bo’yicha prorektor
_____ Q.Inoyatov
“09” 09 2024 yil

O'ZBEKISTON RESPUBLIKASI OLIY TA'LIM, FAN VA INNOVATSIYALAR VAZIRLIGI • NAMANGAN MUHANDISLIK - QURILISH INSTITUTI •
DEVONXONA

“AXBOROT TIZIMLARI VA TEXNOLOGIYALARI” KAFEDRASI

M.TO’XTASINOV

“Python dasturlash tili”
fani bo’yicha

O’ Q U V –U S L U B I Y M A J M U A

Namangan-2024

Ushbu o’quv-uslubiy majmuada «Python dasturlash tili» fanini o’qitish bo’yicha ta`lim texnologiyalari, ularni o’quv jarayoniga qo’llash bo’yicha uslubiy tavsiyalar bayon etilgan. Ushbu tavsiyalar didaktik tamoyillar, ma`ruza, amaliy va laboratoriya mashg’ulotlarini o’tish texnologiyalarini ishlab chiqishning usul va vositalari, ularning muhim belgilaridan iborat ta`limni texnologiya qoidalarini hisobga olgan holda loyihalashtirilgan.

O’quv-uslubiy majmua texnika oliy ta`lim muassasalari o’qituvchilari va talabalari hamda «Python dasturlash tili» fanini o’qitishda zamonaviy pedagogik texnologiyalarini qo’llash jarayonlariga qiziquvchilar uchun mo’ljallangan.

Tuzuvchi:	NamMQI Axborot tizimlari va texnologiyalri kafedrasi dotsenti M.To’xtasinov
Taqrizchi:	NamMTI Texnologik jarayonlarni boshqarish va avtomatlashtirish kafedrasi mudiri PhD. R.Rahimov

O’quv-uslubiy majmua Namangan muhandislik - qurilish instituti Uslubiy kengashining 2024 yil “_30__” ___avgustdagi 1- sonli qaroriga muvofiq o’quv jarayonida foydalanish uchun tavsiya etilgan.

MUNDARIJA

SO’Z BOSHI

SO’Z BOSHI

Mazkur O’quv uslubiy majmua “Python dasturlash tili” fanidan “Informatika va AT” ta`lim yo’nalishi uchun mo’ljallangan bo’lib, Energetika va sanoatni axborotlashtirish fakultetining “Informatika va AT” kafedrasi professor-o’qituvchilari tomonidan ishlab chiqilgan. “Python dasturlash tili” fani o’quv uslubiy majmuasini yaratishda yetakchi xorijiy OTMlari o’quv dasturlariga asosiy adabiyotlar ro’yxatiga kiritilgan H. A. Прохоренок, В. А. Дронов. “Python 3 и PyQt 5. Разработка приложений”. JOHNE. GRAYSON. “Python and TkinterProgramming”, Д.Ю.Федоров. “Основы программирования на примере языка Python” adabiyotlardan foylalanildi.

“Python dasturlash tili” fani “Informatika va AT” ta`lim yo’nalishi o’quv rejasiga asosan 5-6- semetrda 48 auditoriya soatlarida o’qitiladi. Python dasturlsh tili imkoniyatlari. Pythonni o’rnatish, Izoxlar, Dastur natijasini chop etish, Ma’lumotlarni kiritish, Pythonda o’zgaruvchilar. Ma’lumot turlari. O’zgaruvchiqa qiymat o’zlashtirish, Pythonda ma’lumot tiplari. Ma’lumot tipini o’zgartirish. O’zgaruvchini o’chirish, Operatorlar. Matematik operatorlar. Munosabat operatorlari. Ketma-ketliklar bilan ishlash operatorlari. O’zlashtirish operatorlari, Shartli operatorlar. Taqqoslash operatorlari. if...else operatori, Sikl operatorlari. For sikli. range() va enumerate() funksiyalari, While sikli. continue operatori. break operatori, Sonlar bilan ishlashning tashqi funksiya va metodlari, math moduli. Matematik funksiyalar. random moduli. Tasodifiy son generasiyasi, Satrlar va ular ustida amallar. Satr yaratish. Maxusu belgilar. Satrlar bilan ishlash amallari. Satrlarni formatlash. format() metodi, Istisno holatlar bilan ishlash, Ro’yxatlar, Kortejlar, To’plamlar va Diapazonlar. Ro’yxat yaratish. 18-amaliy mashg’ulot. Ko’p o’lchamli ro’yxatlar. Ro’yxat elemantlarini saralash, Lug’atlar, Sana va vaqt bilan ishlash to’g’risida umumiy ma`lumotlar keltirilgan, namunaviy misollar asosida tushuntirilgan.

Ushbu o’quv uslubiy majmua amaliy mashg’ulotlar materiallari (amaliy topshiriqlar, namuna, adabiyotlar ro’yxati, tarqatma materiallar, keyslar banki, test savollari)dan tashkil topgan.

I. MA’RUZA MASHG’ULOT MATERIALLARI

1-ma’ruza. Python dasturlash tili bilan tanishish. Python dasturlash tili yaratilishi tarixi. Python dasturlsh tili imkoniyatlari. Pythonni o’rnatish. Dastur tuzilishi. Izoxlar. Dastur natijasini chop etish. Ma’lumotlarni kiritish

Reja

1. Python dasturlash yaratilish tili tarixi
2. Python dasturlash tili imkoniyatlari
3. Pythonni o’rnatish
4. Dastur tuzilishi
5. Izoxlar
6. Dastur natijasini chop etish
7. Ma’lumotlarni kiritish

Kalit so’zlar: *Python, pythonni o’rnatish, dasturlash tillari, izoh, chop etish, ma’lumotlarni kiritish*

Python dasturlash tili yaratilishi tarixi

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

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

Bir qancha vaqt o`tib Gvido Gollandiyadan Amerikaga ko`chib o`tdi. Uni CNRI korparatsiyasiga ishlashga taklif etishdi. U o`sha yerda ishladi va korparatsiya shug`ullanayotgan proektlarni Python tilida yozdi va bo`sh ish vaqtlarida tilni interpretatorini rivojlantirib bordi. Bu 1990 yil Python 1.5.2 versiyasi paydo bo`lguncha davom etdi. Gvidoning asosiy vaqti korparatsiyani proektlarini yaratishga ketardi bu esa

unga yoqmasdi. Chunki uning Python dasturlash tilini rivojlantirishga vaqti qolmayotgandi. Shunda u o`ziga tilni rivojlantirishga imkoniyat yaratib bera oladigan homiy izladi va uni o`sha paytlarda endi tashkil etilgan BeOpen firmasi qo`llab quvvatladi. U CNRI dan ketdi, lekin shartnomaga binoan u Python 1.6 versiyasini chiqarib berishga majbur edi. BeOpen da esa u Python 2.0 versiyani chiqardi. 2.0 versiyasi bu oldinga qo`yilgan katta qadamlardan edi. Bu versiyada eng asosiysi til va interpretatorni rivojlanish jarayoni ochiq ravishda bo`ldi.

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

Python dasturlash tili imkoniyatlari

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

Pythonning o`ziga xosligi

- ✓ Oddiy, o`rganishga oson, sodda sintaksisga ega, dasturlashni boshlash uchun qulay, erkin va ochiq kodlik dasturiy ta'minot.
- ✓ Dasturni yozish davomida quyi darajadagi detallarni, misol uchun xotirani boshqarishni hisobga olish shart emas.
- ✓ Ko'plab platformalarda hech qanday o'zgartirishlarsiz ishlay oladi.
- ✓ Interpretatsiya (Интерпретируемый) qilinadigan til.
- ✓ 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 tushirsa(chaqirsa) bo'ladi.
- ✓ Juda ham ko'p xilma-xil kutubxonalarga ega.
- ✓ xml/html fayllar bilan ishlash
- ✓ http so`rovlari bilan ishlash
- ✓ GUI (grafik interfeys)
- ✓ Web ssenariy tuzish
- ✓ FTP bilan ishlash
- ✓ Rasmli audio video fayllar bilan ishlash
- ✓ Robot texnikada
- ✓ Matematik va ilmiy hisoblashlarni programmalash

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

Qisqacha qilib aytganda Python quyidagi sohalarda faol ishlatiladi:

1. Tizimli dasturlash.
2. Grafik interfeysli dasturlarni ishlab chiqish.
3. Dinamik web-saytlarni yaratish.
4. Komponentlarning integratsiyasi.
5. Ma'lumotlar bazalari bilan ishlash uchun dasturlarni ishlab chiqish.
6. Prototiplarni tezkor yaratish.
7. Ilmiy hisoblash uchun dasturlarni ishlab chiqish.
8. O'yin dasturlar yaratish

Python dasturlarini ishga tushirish uchun nima qilishimiz kerak? Ushbu savolga javob berishdan oldin, dasturlarning kompyuterda qanday ishlashini ko'rib chiqamiz. Dasturlarning bajarilishi operatsion tizim tomonidan amalga oshiriladi (Windows, Linux va boshqalar). Operatsion tizimning vazifalari dastur uchun resurslarni taqsimlash, kiritish/chiqarish qurilmalariga kirishni taqiqlash yoki ruxsat berishdan iborat.

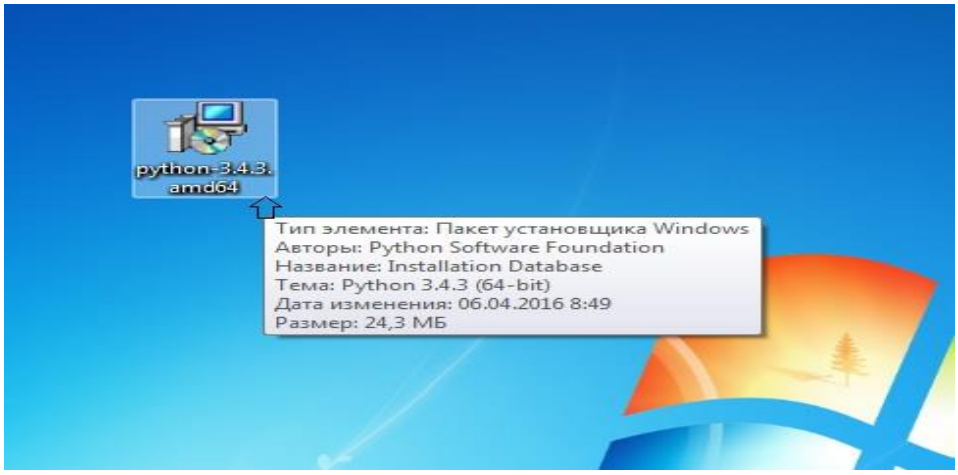
Python dasturlarini ishga tushirish uchun sizga Python tarjimoni (virtual mashina) kerak bo'ladi. Ushbu dastur Python dasturchisidan operatsion tizimning barcha xususiyatlarini yashiradi, shuning uchun Windows tizimida Python-da dastur yozib, siz uni, masalan, GNU / Linux-da ishlatishingiz va bir xil natijaga erishishingiz mumkin.

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**

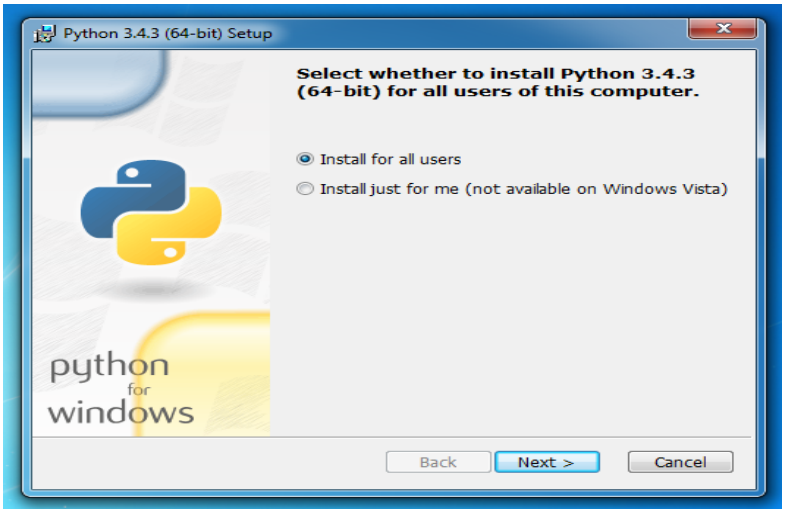
Agar sizda **Python 3.4.3** yozuvi yoki shunga o'xshash yozuv hosil bo'lsa unda hammasi joyida.

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.



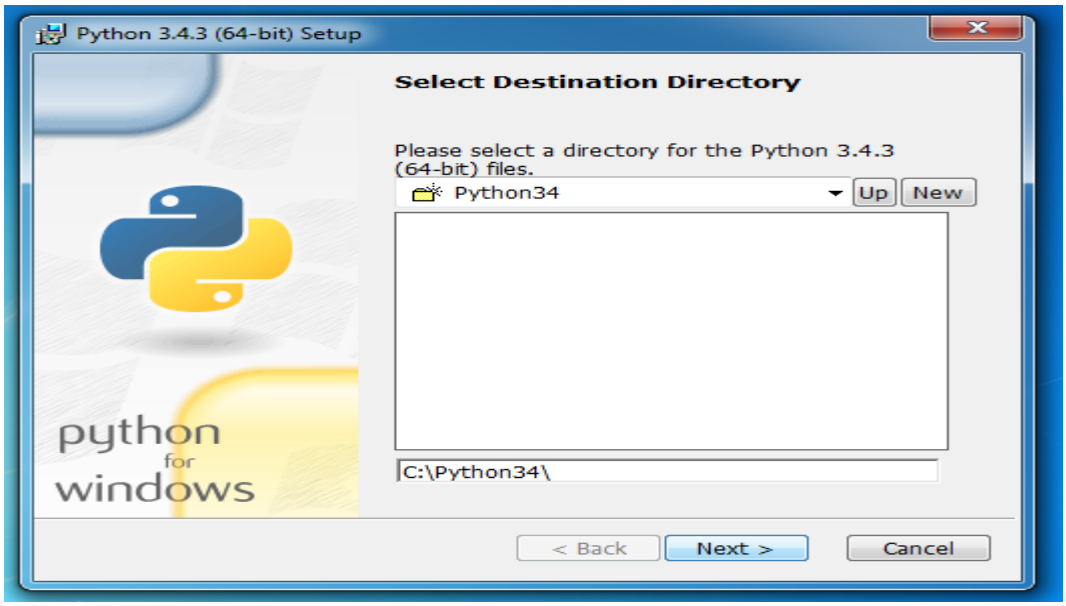
1.1.1-chizma. Python dasturining o`rnatiluvchi fayli.

Python dasturlash tilining o`rnatuvchi paketini ustiga sichqoncha ko`ratkichini 2 marta bosamiz va bizga quyidagi oyna hosil bo`ladi.



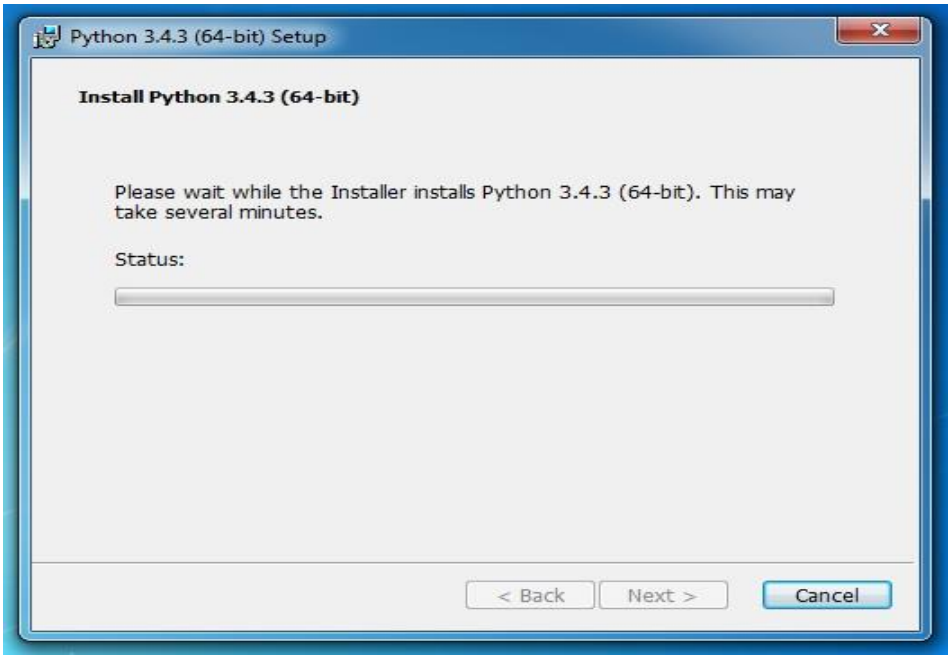
1.1.2-chizma. Python dasturini o`rnatishni boshlashni ko`rsatuvchi oyna.

Bu yerda **Install for all users**-barcha foydalanuvchilar uchun. **Install just for me**- faqat siz uchun, agar buni tanlab istalyatsiya qilsak ya’ni o`rnatsak Windows Vista operatsion sistemasida xatolik yuz beradi va dastur ishlamaydi. Shuning uchun **Install for all users** ni tanlaganimiz maqul. Keyin next tugmasi bosamiz.



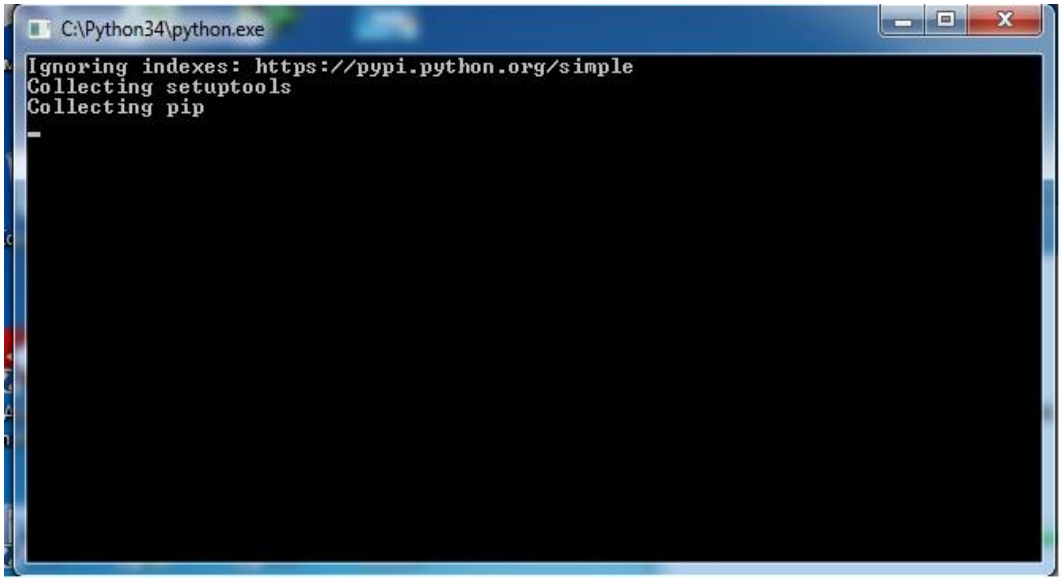
1.1.3-chizma. Python dasturini o`rnatilish joyini ko`rsatish oynasi.

Bu yerda esa Python dasturlash tilini qayerda o`rnatilishi ko`rsatilayapti.



1.1.4-chizma. Python dasturini o`rnatilish jarayoni.

Python o`rnatilyapti va bir necha sekunddan so`ng quyidagi oyna namoyon bo`ladi:



1.1.5-chizma. PIP kutubxonasini qo`shish jarayonida hosil boladigan oyna.

Bunda Console rejimida dastur ishga tushib **pip** kutubxonasini qo`shadi.

1.1.6-chizma. Python dasturini o`rnatish tugallanganligi haqidagi muloqot oynasi.

Dasturni o`rnatish muofaqqiyatli tugallandi.

Dastur tuzilishi

Pythonda har bir instruksiya alohida satraga yoziladi. Ko’pgina dasturlash tillari(masalan, PHP, Perl va boshqa)da har bir instruksiya nuqtali vergul bilan yakunlanadi. Pythonda ham instruksiya ohiriga nuqtlil vergul qo’yish mumkin, biroq majburiy emas.

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

Pythonga kiritilgan amallar bir xil shablonda yoziladi. Bunda asosiy amal ikki nuqta bilan tugatiladi va uning orqasidan kiritilgan blok kodi ham joylashadi. Odatda, asosiy amalning ostidagi satr bo`sh joy bilan ajratiladi.

Pythonda nuqtali vergul quyish tavsiya etilmaydi. Satr yakuni instruksiya yakuni hisoblanadi. Shunga qaramay, bir necha instruksiyani bir satrga yozish kerak bo’lsa nuqta verguldan foyadalanish mumkin va quyidagi misolda keltiriladi.

1-misol. Bir nechta instruksiyani bir satrda qo’llash

```
>>> x = 5;  y = 10;  z  = x + y  #  uch instruksiya bir satrda

>>> print (z)
```

Pythonda dastur yozilishinig yana bir muhim hususiyati shundaki, kodlar blokini ajratsih uchun alohiada belgi ishlatilmaydi. Faqat blok tekislanishi bilan ajratiladi.

Masalan, PHPda while sikli tanasi figurali qavs yordamida quyidagicha yoziladi:

```
$i = 1;

while ($ i < 11) {

echo $i.  "\ n";

$i ++;

}
```

```
echo " Dastur tamom ";
```

Python tilida esa boshqacha yoziladi:

```
i = 1

while i < 11:

    print(i)

    i += 1

print("Dastur tamom")
```

Diqqat qiling, blok ichidan tashqari barcha instruksiyalar bir hil joylashgan. Blok tanasi esa alohida probellar satrida joylashgan. Misoliizda ikkita instruksiya o'n marta bajariladi. Blok yakunlangach yana instruksiya dastlabki probel tekisligiga joylashtiriladi. Bu print("Dastur tamom") instruksiyasidir.

Agar bir blok ichidagi probeller turlicha bo'lsa, xatolik kelib chiqadi. Odatda to'rt probel yoki bitta tabulyatsiya belgisi bilan ajratiladi.

Agar blok bitta instruksiyadan ibotar bo'sa uni asosiy instruksiya bilan bir satrga joylashtirish mumkin. Masalan:

```
for i in range (l, 11) :

print (i)

print ("Dastur tamom" )
```

yuqoridfagi dasturni quyidagicha ham yozish mumkin:

```
for i in range (l, 11) : print (i)
print ("Dastur tamom")
```

Agar konstruksiya juda uzun bo'lsa quyidagi usullardan birida uni yangi satrga bo'lib yozish mumkin.

- ✓ satr oxiriga "\" belgisini joylashtirish yordamida va undan keingi belgi navbatdagi satrdan yoziladi. Masalan:

Пример:

```
x = 15 + 20 \
  + 30
print (x)
```

- ✓ ifodani aylanma qavslar yordamida yozish. Bu ancha yaxshiroq usul bo'lib, ixtiyoriy ifodani satrlarga joylashtirish mumkin bo'ladi. Masalan:

```
x = (15 + 20  # bu izoh
+ 30)
print (x)
```

- ✓ ro’yxat va lug’atlarni aniqlashda ularni bir nechta satrga yozish mumkin. Bunda mos ravishda kvadrat va aylama qavslardan foydalaniladi. Ro’yxatni aniqlashga doir misol:

```
arr = [15,  20,  # bu izoh
       30]
print (arr)
```

Lug’atni aniqlashga doir misol:

```
arr = {" x" :  15,  "y":  20,   # bu izoh
       "z" :  30 }

      print (arr)
```

Izohlar

Izohlar dastur matnini tushintirish uchun foydalaniladi va dasturni izohlaydi. Izohlar ihtiyoriy matn, ko’rsatalardan iborat bo’lishi mumkin va ular bajarilmaydi. Ular dastur kodini o'qiyotganlar uchun foydali bo'ladi va dastur nima qilishini oson tushunishga yordam beradi. Izohlarga yechimdagi muhim joylarni, muhim bo'lgan qismlarni yozish mumkin. Pythonda bir satrli izoh ishlatiladi va u # belgisidan boshlanadi. Masalan: *# bu izoh*

 # print — bu funksiya

print ('salom dunyo! ')

bir satrli izoh satr boshidan balki instruksiyaning ihtiyoriy joyidan boshlanishi mumkin va unadan keying belgilar izoh sifatida qabul qilinadi. Maslan:

print *('salom dunyo!') # print — bu funksiya*

Agar # belgisi aylanma qavs yoki apastrof ichida joylashgan bo’lsa, u izoh belgisi hisoblanmaydi:

print ("# Bu izoh emas")

Python da ko’p satrli izohlar mavjud bo’lmagani uchun, bazan ko’p satrli izohlar uchtalik qo’shtirnoq (yoki uchtalik apostrof) orqali berilishi mumkin:

"""

Bu instruksiya bajarilmaydi

Faqat izoh sifatida foydalanish mumkin

print ("Bu ham izohning uchinchi satri!")

"""

Yuqoridagi satrlar bajarilmaydi ular faqat dasturda o’qish uchun holos.

Dastur natijasini chop etish

Dastur natijasini ekranga chiqarish uchun *print()* funksiyasidan foydalaniladi. Funksiya quyidagi formatda beriladi:

print ([<Obyektlar>] [, sep= ' '] [, end=' \n'] [, file=sys.stdout] [, flush=False])

print() funksiyasi obyektni satrga o’tkazadi va uni stdout standart chiqarishga uzatadi. *file* parametri yordamida chiqarish boshqa joyga masalan, faylga yo’naltirilishi mumkin. Bunda *flush* paramtri qiymati False bo’lsa chiqariladigan qiymat faylga yozilishi majburiy bo’ladi. Chop etishlarni yo’naltirish bo’yicha fayllar mavzusida batafsil tanishib chiqamiz.

Satr chop etilgach avtomatik satr keying satrga o’tkaziladi:

print ("1-satr")

print ("2-satr ")

Natija:

1-satr

2-satr

Agar bir nechta qiymatni bitta satrda chiqarish zarur bo’lsa, qiymatlar vergul yordamida print() funksiyasidagi birinchi parametrda beriladi:

print ("1-satr", "2-satr”)

Natija:

1-satr 2-satr

Misoldan ko’rinib turganidek, chop etiladigan satrlar orasiga avtomati probel qo’yiladi. **sep** parametri yordamida boshqa belgini ajratgich sifatifda ko’rsatish mumkin.

Masalan satrlar probelsiz chiqishi uchun quyidagicha yoziladi:

print ("1-satr","2-satr", sep="")

Natija:

1-satr2-satr

Obyektni chiqargandan keyin ohirida keying satrga o'tish belgisi qo’shiladi. Agar keying chiqariladigan satr oldingisi bilan bir satrga joylashishi zarur bo’lsa end

parametrida quyidagicha probel belgisi ko’rsatiladi:

```
print ("1-satr", "2-satr", end=" ")
print ("3-satr)
```

Natija:

```
1-satr 2-satr 3-satr
```

Agar aksincha bolishini ya’ni keying chop etish natijasi yangi satrdan chiqishini hohlasangiz *print*() funksiyasida end parametrini ko’rsatmang. Masalan:

```
for n in range (l, 5):
print (n, end=" ")
print()
print ("Bu matn yangi satrda chop etiladi")
```

Natija quyidagicha chop etiladi:

```
1 2 3 4
Bu matn yangi satrda chop etiladi
```

Bu yerda biz for siklidan foydalanib sonlar ketma ketlini chop etdik. Sikldan keyingi print() operatori navbatdagi print ("Bu matn yangi satrda chop etiladi") operatorida chop etiladigan satrni yangi satrdan chop etish uchun parametrsiz shaklda yozildi.

Ma’lumotlarni kiritish

Python 3 da ma’lumotlarni kiritish uchun stdin standartidan ma’lumot oluvchi input () funksiyasi dan foydalaniladi. Funksiya quyidagi umumiy ko’rinishga ega:

[<Qiymat> =] input ([<habar>])

Misol sifatida foydalanuvhci bilan salomlashish dasturini ko’rib chiqamiz.

2-misol:

```
# -*- coding: utf-8 -*-
name = input("Ismingizni kiriting: ")
print("Salom,", name)
input("Dasturdan chiqish uchin <Enter> tugmasini bosing")
```

Dastur natijasi:

```
==== RESTART: C:/Users/user/AppData/Local/Programs/Python/Python37/14.py ====
Ismingizni kiriting: Umidjon
Salom, Umidjon
Oynani yopish uchun <Enter> tugmasini bosing
```

Nazorat savollari

1. Python qanday til?
2. Python yaratuvchilarini bilasizmi?
3. Python qanday imkoniyatlarni taqdim etadi?
4. Python muharrirlarni bilasizmi?
5. Python qanday o’rnatiladi?
8. Pythonda operatorlar qanday ajratiladi?
9. Pythonda blok qanday beriladi?
10. Python shell muhiti vazivasi nima?
11. Pythonda izohlar qanday beriladi?
12. Pythonda dastur chop etish operatori?
13. Pythonda ma’lumot kiritish operatori.

2-ma’ruza. O’zgaruvchilar. O’zgaruvchini nomlash. Ma’lumot turlari. O’zgaruvchiqa qiymat o’zlashtirish. Ma’lumot tipini aniqlash. Ma’lumot tipini o’zgartirish. O’zgaruvchini o’chirish.

Reja

1. O’zgaruvchini nomlash
2. Ma’lumot turlari
3. O’zgaruvchiga qiymat o’zlashtirish
4. Ma’lumot tipini aniqlash
5. Ma’lumot tipini o’zgartirish
6. O’zgaruvchini o’chirish

Kalit so’zlar: *o’zgaruvchini, ma’lumot turlari, o’zgaruvchini o’chirish, ma’lumot tipini aniqlash, ma’lumot tipini o’zgartirish, o’zgaruvchini o’chirish*

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.

Python dasturlash tilida barcha ma’lumotlar obyektlar sifatida beriladi. Har bir obyekt ma’lumot tipiga va qiymatiga ega. Obyektga ruxsat o’zgaruvchi orqali aniqlashtiriladi. O’zgaruvchi insalizasiyasida obyekt (kompyuter xotirasidagi obyekt manzili) ga havola saqlanadi. Keyinchalik shu havola sababli dasturdan obyektni o’zgartirish mumkin bo’ladi.

O’zgaruvchini nomlash

Har bir o’zgaruvchi lotin alifbosi, raqamlar, ostga chizish belgisidan iborat bo’lgan va raqmdan boshlanamaydigan unikal nomga ega bo’lishi kerak. Bundan tashqari o’zgaruvchini nomlashda ostgi chiziqni nom boshida kelishi tavsiya etilmaydi, chunki ko’p bunday nomlangan identifikator tilda maxsus vazifani bajaradi. Ular ustma-ust tushib qolib, dasturda xatolik kelib chiqishi mumkin. Bundan tashqari kalit so’zlardan o’zgaruvchi nomi sifatida foydalanish ta’qiqlanadi.

Qisqa qilib aytganda o'zgaruvchilarni nomlashda quyidagi qoidalarga amal qilish kerak:

- O'zgaruvchining birinchi belgisi alifbo harfi (ASCII simvollari katta va kichik registrda) yoki “_” (ostki chiziq) simvoli bo'lishi mumkin(lekin odatda tavsiya etilmaydi).

- O'zgaruvchilarning qolgan qismi harflardan (ASCII simvollari katta va kichik registrda), “_" (ostki chiziq) simvoli va raqamlardan(0-9) tashkil topishi mumkin.
- O'zgaruvchilar nomlashda katta va kichik registrlar farqlanadi. Masalan, *myname va myName – bular boshqa-boshqa o'zgaruvchi hisoblanadi.*
- O'zgaruvchilarni to'g'ri nomlashga misollar*: i, _my_name, name_23, a1b2_c3*
- O'zgaruvchilarni noto'g'ri nomlashga misollar: 2things, ' ', my-name, >a1b2_c3 va “o'zgaruvchi qo'shtirnoqda"

Kalit so’zlar

False – yolg`on.

True - rost.

None - “bo`sh” obyekt.

and – mantiqiy VA amali.

with / as – konteks menejeri.

break –tsikldan chiqish.

class – metod va atributlarda iborat.

continue – tsikldan keyingi iteratsiyaga o`tish.

def – funksiyani aniqlash.

del – obyektni yo`qotish.

elif – aks holda, agar.

else – for/else yoki if/elsega qarang.

for – for tsikli.

from – moduldan bir nechta funksiyani import qilish.

if - agar.

import – moduldan import.

is –xotirani bitta joyida 2 ta obyektni jo`natsa bo`ladimi.

lambda –yashirin funksiyani aniqlash.

not –mantiqiy inkor amali.

or –mantiqiy Yoki amali.

while – while tsikli.

Ma’lumot turlari

Python 3 dasturlash tilida obyektlar quydagi turlarda bo’lishi mumkin:

- bool – ma’lumotlar mantiqiy turi. True yoki False (1 yoki 0 ga teng kuchli) qiymatiga ega bo’lihi mumkin:

```
>>> type (True) ,  type (False)
(< class  'bool'> ,  <class  'bool '> )
>>> int (True), int (False)
(1, 0)
```

- NoneType – None qiymatli obyekt(qiymati yo’qligini anglatadi):

```
>>> type (None)
<class 'NoneType '>
```

Mantiqiy ifodada None qiymati False kabidir:

```
>>> bool (None )
False
```

- int – butun son. Son qiymati chegarasi faqat operativ xotira sig’imiga bog’liq:

```
>>>  type (214 7 48 364 7) ,  type (9 9999999999 9999999999 999 )
(< class  'int'> ,  <class  'int'> )
```

- float – suzuvchi vergulli son:

```
>>>  type( 5.1) ,  type (8 .5e- 3)
(< class 'float '> ,  <class  'float ' >)
```

- complex – kompleks son:

```
>>>  type (2 +2j )
<class 'complex'>
```

- str - Unicode-satr:

```
>>>  type(“Satr”)
<class 'str '>)
```

- bytes – o’zgarmas baytlar ketma-ketligi:

```
>>>  type (bytes (“Satr”,  "ut f- 8") )
<class 'bytes ' >)
```

- bytearray – o’zgaruvchan baytlar ketma-ketligi:

```
>>>  type (bytearray ("Satr",  "utf-8"))
<class 'bytearray'>)
```

- list – ro’yxat. list boshqa dasturlash tillaridagi massivlarning analogidir:

```
>>>  type ([1 ,  2,  3 ])
<class  'list'>)
```

- tuple - kortej:

```
>>> type ((1, 2, 3)
<class 'tuple'>)
```

- range -diapazon:

```
>>> type ( range (l , 10))
<class 'range '>
```

- dict - lug'at. dict ma'lumot turi bohsqa dasturlash tillariadagi assosiativ massivning analogidir:

```
>>> type ({" x ": 5, " y": 20})
<class 'dict '>
```

- set – to'plam (unikal obyektlar to'plami):

```
>>> type ({"a", "b", "c"})
<class 'set '>
```

- frozenset – o'zgarmas toplam:

```
>>> type (frozenset ([ "a", "b", "c"]))
<class 'frozenset '>
```

- ellipsis – uch nuqta shakli yoki Ellipsis so'zini anglatadi. ellipsis turi qirqim olish kengaytmasi sifatida ham ishlatiladi:

```
>>> type (…) , … , ... is Ellipsis
(<class 'ellipsis'> , Ellipsis, True )
>>>class C ():
def __getitem__(self, obj) : return obj
>>> c = C()
>>> c[…, 1:5 , 0: 9:1 , 0]
(Ellips is, slice (l, 5, None) , slice (0, 9, 1) , 0)
```

- function - funksiya:

```
>>> def func() : pass
>>> type (func)
<class 'function'>
```

- module - modul:

```
>>> import sys
>>> type (sys )
<class 'module '>
```

- type – ma'lumotar turi va sinfi. Hayron bo'lmang! Pythonda hamma ma'lumotlar hatto ma'lumot turlarining o'zi ham obyekt hisoblanadi!

```
>>> class C: pass
>>> type (C)
<class 'type'>
>>> type (type( "") )
<class 'type '>
```

Asosiy ma'lumot turari o'zgaruvchan va o'zgarmaslarga bo'linadi. O'zgaruvchan turlarga ro'yxatlar, lug'atalar, va bytearray tegishli. Ro'yxat elementini o'zgartirish misoli:

```
>>> arr = [1, 2, 3]
```

```
>>> arr[0] = 0  # Ro'yxatning birinchi elementi o'zgaradi
>>> arr
[0, 2, 3]
```

O'zgaras turlarga sonlar, satrlar, kortejlar, diapazonlar va bytes turlari tegishlidir. Masalan, agar ikkita boshqa satrdan satr olmoqchi bo'lsak konkatenasiya operasiyasiadan foydalanish kerak va yangi obyektga havola o'zgaruvchig ao'zlashtiriladi:

```
>>> str1 = "Gultoji"
>>> str2 = "xo'roz"
>>> str3 = str1 + str2 # Konkatenasiya
>>> print(str3)
Gultojixo'roz
```

Bundan tashqari, ma'lumotlar turlari izchillikdagi va aks etuvchilarga bo'linadi. Izchillikga satrlar, ro'yxatlar, kortejlart, diapazonlar, byte va bytearray turlari aks etuvchiga lug'at turlari tegishlidir.

Izchillikdagi va aks etuvchi iterasiya mexanizmini qo'llab-quvvatlaydi va _next_ ( ) yoki next ( ) funksiyalari yordamida elementlar bo'yicha yurib chiqishi mumkin. Masalan, ro'yxat elementini chiqarish quyidagicha:

```
>>> arr = [1, 2]
>>> i = iter(arr)
>>> i.__next__()        # __next__() metodi
1
>>> next(i)  # next() funksiyasi
2
```

Agar lug'atdan foydalanilsa, har bir iterasiya kalit qaytaradi:

```
>>> d = {"x": 1, "y": 2}
>>> i = iter(d)
>>> i.__next__() # kalit qaytariladi
'y'
>>> d[i.__next__()] # kalit bo'yicha qiymat qaytaradi
1
```

Amaliyotda bun usullardan keng foydalanilmaydi. Buning o'rniga ko'pincha for sikli operatoridan foydalaniladi. Masalan, ro';yxat elementlari quyidagicha chop etiladi:

```
>>>  for i in [1, 2]:
print (i)
```

So'zdagi harflarni chop etish misolini ko'ramiz. Misol uchun so'zdagi har bir harfga tire qo'yamiz:

```
>>> for i in "Salom":
print(i + " -", end=" ")
```

Natija:

S-a-l-o-m

Lug’at elementlarini saralash maslasi:

```
>>> d = {"x": 1, " y":  2}
>>> for key in d:
print (d[key])
```

O’zgaruvchiqa qiymat o’zlashtirish

Python dasturlash tilida *dinamik tiplashtirishdan* foydalaniladi. Bu shuni anglatadiki, o’zgaruvchi o’zlashtirgan qiymatiga qarab interpretator avtomatik ravishda ma’lumotlar turidan biriga ajratadi. O’zgaruvchiga qiymat “=” operatori yordamida o’zlashtiriladi (beriladi). Misollar yordamida ko’ramiz:

```
>>> x  =  7              #  int tipli
>>> y=  7. 8             #  float tipli
>>> sl  =  "Satr "       #  sl  o’zgaruvhcisi “Satr ” qiymatini o’zlashtirdi
>>> s2  =  ' Satr '      #  s2  o’zgaruvhcisi ham “Satr ” qiymatini o’zlashtirdi
>>> b = True             #  b o’zgaruvchisi mantiqiy True qiymatini o’zlashtirdi
```

Bir satrda birdaniga bir nechta o’zgaruvhciga qiymat o’zlashtirish mumkin:

```
>>> x  =  y  =  10
>>> x,  y
(10, 10 )
```

Qiymat o’zlashtirilgach o’zgaruvchida obyekt emas balki, obyektga havola saqlanadi.

Buni albatta gruhli o’zlashtirishda ham hisobga olinadi. Guruxli o’zlashtiurish son, satr va kortej tiplarida qo’llanilishi mumkin, biroq o’zgaruvchan obyektlarda buni amalga oshirish mumkin emas. Masalan:

```
>>> x  =  y  =  [ 1 ,   2 ]       #  Go’yoki ikkita obyekt yaratildi
>>> x,  y
([1, 2], [1, 2])
```

Bu misolda biz ikkita elementdan iborat ro’yxat yaratdik hamda x va y o’zgaruvchilarga qiymatini o’zlashtirdik. Endi y o’zgaruvchisi qiymatini o’zgartirib ko’ramiz:

```
>>> y[l]  =  100  #  Ikkinchi elementni o’zgartirdik
>>> x, y
([1,  100] ,  [1,  100 ] )
```

Misoldan ko’rinib turibdiki, y o’zgaruvchisi qiymatining o’zgarishi x o’zgaruvchisi qiymatini o’zgarishiga ham olib kelmoqda. Demak, har ikki o’zgaruchida bitta obyektga havola ko’rsatilmoqda. Agar ikkita obyektni ko’rsatmoqchi bo’lsak guruxli oz’lashtirishdan emas, alohid ao’zlashtirishdan foydalanish kerak:

```
>>>  x  =  [1,  2]
>>>  y  =  [1,  2]
>>>  y[ l]  =  100              #  Ikkinchi element o’zgarmoqda
>>>  x,  y
([1,  2],  [1,  100 ]  )
```

Ikkita o’zgaruvchi bir obyektga havola ko’rsatayorganini tekshirishga **is** operatori imkon beradi.

Agar o’zgaruvchilar bitta obyektga havola ko’rsatayortgan bo’lsa, is operatori True qiymatini qaytaradi:

```
>>> x = y = [1, 2] # bir obyekt

>>> x is y
True
>>> x = [1, 2]        # turli obyekt
>>> y = [1, 2]        # turli obyekt
>>> x is y
Fa lse
```

Interpretator kodlar samaradorligini oshirish maqsadida kichik sonlar va uzun bo’lmagan satrlarni keshlashni amalga oshiradi. Agar ikkita o’zgaruvchiga 2 sono o’zlashtirilsa, bu o’zgaruvchilar bitta obyektga havola ko’rsatadi degani. Masalan:

```
>>> x = 2; y = 2; z = 2
>>> x is y, y is z
(True, True)
```

Obyektga ko’rsatilgan havolalar miqdorini aniqlashga sys modulidagi getrefcount () metodi imkon beradi:

```
>>> import sys              # sys modulini ulaymiz
>>> sys.getrefcount (2)
304
```

Qachonki obyektga havolalar soni nol ga teng bo’lib qolsa obyekt operativ xotiradan avtomatik o’chiriladi.

Ma’lumot tipini o’zgartirish

Oldingi mavzudan bialsizku Pythonda *turlar dinamikdir*. O’zgaruvchiga qiymat o’zlashtirilgach unda obyekt emas balki, aniqlagan turdagi obyektga havola saqlanadi. Agar keyin o’zgaruvchiga boshqa turdagi qiymat o’zlashtiriladigan bo’lsa, o’zgaruvchi endi unga mos turdagi boshqa obyektga havola saqlaydi. Pythondagi shu tarzdagi ma’lumot turi bu o’zgaruvchining emas, obyektning tavsifidir. O’zgaruvchi faqat o’zida obyektga havloni saqlaydi.

O’zgaruvchiga qiymat o’zlashtirilgach o’zgaruvchi o’zlashtirilgan qiymat turiga mos amallarni bajarishi mumkin bo’ladi. Masalan, satr o’zgaruvchisiga son o’zgaruvchisini qo’shish mumkin emas va bu quyidagicha xatolik keltirib chiqaradi:

```
>>> 2 + "2511"
Traceback (most recent ca ll last ) :
```

File 11 <pyshe ll #0>11, line 1, in <module>
2 + “25”
TypeError: unsupported operand type (s) for +: 'int' and 'str '
Quyidagi funksiyalar ma’lumotlar turii o’zgartirish uchun mo’ljallangan:

- bool([<Obyekt>]) - obyektni mantiqiy turga o’zgartiradi. Masalan :

>>> bool(0) , bool (l) , bool (" ") , bool (“Satr”), bool ([l, 2]), bool ([ ])
(False, True, False, True, True, False)

- int ([<Obyekt> [, <Sanoq sistemasi>]]) – obyektni songa o’zgartiradi. Ikkinchi parameter sifatida sanoq sistemasii ko’rsatish mumkin. (agar ko’rsatilmasa 10 lik sanoq sistemasi deb hisoblanadi). Masalan:

>>> int(7. 5), int ("71")
(7, 71)
>>> int ("71", 10) , int ("71", 8) , int ("0071", 8) , int ("A" , 16)
(71, 57, 57, 10)

Agar o’tkazishga imkon bo’lmasa, istisni haqida habar beriladi:

>>> int ("7ls")
Traceback (most recent call last) :
File "<pyshell#9>" , line 1, in <module>
int ("71s")
ValueError: invalid literal for int () with base 10: '7ls'

- float ([<Son yoki satr>]) - butu son yoki satrni haqiyqiy (suzuvchi vergulli) songa o’tkazadi. Masalan:

>>> float (7) , float ("7. l")
(7.0, 7.1)
>>> float ("Infinity"), float ("-inf")
(inf, -inf)
>>> float ("Infinity") + float ("-inf")
nan

- str ([<Obyekt>]) - Obyektni satrga o’tkazadi. Masalan:

>>> str (125), str([l, 2, 3])
('125', '[1, 2, 3] ')
>>> str((l , 2, 3)), str ({" x" : 5, "y" : 10 })
(' (1, 2, 3) ', "{ 'y ' : 10, 'x' : 5) ")
>>> str (bytes("satr" , "utf-8 "))
"b'\\xd1\\x81\\xdl\\x82\\xdl\\x80\\xd0\\xbe\\xd0\\xba\\xd0\\xb0'"
>>> str(bytearray("satr", "utf-8 "))
"bytearray (b'\\xdl \\x81\\xdl \\x82\ \xdl \\x80\ \xd0\ \xbe\ \xd0\\xba\\xd0\\xb0 ')"

- str (<Oyekt>[, <Kodirovka>[, <Xatolik tahlili>]]) - bytes yoki bytearray turidagi obyektni satrga o’tkazish. Uchnchi parameter sifatida "strict", "replace " yoki "ignore " lardan birini berish mumkin. Masalan:

>>> objl = bytes ("1-satr", "utf- 8 ")

```
>>> obj2 = bytearray ("2-satr", "utf- 8")
>>> str (objl, "utf- 8") , str (obj2, "utf- 8")
('1-satr', '2-satr ')
>>> str(objl, "ascii ", "strict" )
Traceback (most recent call last):
File "<pyshel l#16>" , line 1, in <module>
str (objl, "ascii ", "strict" )
Uni code DecodeError : 'ascii ' codec can 't decode byte
Oxdl in position 0: ordinal not in range (1 28)
>>> str (objl, "ascii", "ignore")
'1'
```

- bytes (<Satr>, <Kodirovka>[, <Xatolik tahlili>]) - satrni bytes tutidagi obyektga o'tkazish. Uchnchi parameter sifatida "strict " (odatiy qiymaat), "replace" yoki "ignore" ni ko'rsatish mumkin.

Masalan:

```
>>> bytes ("satr", "cp1251")
b' \xfl \xf2\xf0 \xee\xea\xe0'
>>> bytes ("satr123", "ascii", "ignore" )
b'123 '
```

Misol sifatida foydalanuvchi tomonidan kiritilgan ikki soni qo'shish masalasini ko'rib chiqaylik

2.4-misol. Ma'lumot kiritish va natija olish

```
# -*- coding: utf-8 -*-
x = input ("x=")          # 5 ni kiritamiz
y = input ("y=")          # 12 ni kiritamiz
print (x + y)
input ()
```

Ushbu dastur natijasini olsak "512" chiqadi. Chunki input() funksiyasi natija sifatida satri qaytaradi. Agar bu sonlar yig'indisini olishni hoxlasak ularni son turiga o'tkazish kerak (2.5-misol)

2.5-misol. Satrni songa o'tkazish

```
# -*- coding: utf-8 -*-
x = int (input ("x="))          # 5 ni kiritamiz
y = int (input ("y="))          # 12 ni kiritamiz
print (x + y)
input ()
```

Bu dastur natijasi 17 bo'ladi. Biroq foydalanuvchi son o'rniga satr kiritsa, xatolik kelib chiqadi. Bu xatolikni qayta ishlashni biz keyinroq ko'rib chiqamiz.

O'zgaruvchini o'chirish

O'zgaruvchini del instruksiyasiu yordamida o'chirish mumkin:

del <o’zgaruvchi>[, …, <o’zgaruvchi_N>]
Bitta o’zgaruvchini o’chirishga misol:

```
>>> x = 10; x
10
>>> del  x;  x
Traceback (most  recent  call  last) :
Fi le  "<pyshell# l>" ,  line 1,  in  <module>
del x; x
NameError:  name  'x'  is  not  defined
```

Bir nechta o’zgaruvchini o’chirishga doir misol:

```
>>> x,  y = 10,  20
>>> del x,  y
```

Nazorat savollari

1. Pythonda oz’garuvchi nima?
2. Pythonda qanday ma’lumot turlari bor?
3. Pythonda o’zlashtirish amali.
4. Pythonda qanday ma’lumot turlari bor?
5. Pythonda ma’lumot turlarini o’zgartirish
6. Pythonda o’zgaruvchini o’chirish.

3-ma’ruza. Operatorlar. Matematik operatorlar. Munosabat operatorlari. Ketma-ketliklar bilan ishlash operatorlari. O’zlashtirish operatorlari.

Reja

1. **Operatorlar.**
2. Matematik operatorlar.
3. Munosabat operatorlari.
4. Ketma-ketliklar bilan ishlash operatorlari.
5. O’zlashtirish operatorlari.
6. Operatorlarni bajarish ketma-ketligi

Operatorlar ma’lumotlar ustida biror amal bajarishga imkon beradi. Masalan, o’zlashtirish operatori o’zgaruvchi qiymatlarini saqlash, matematik operatorlar arifmetik amallar bajarish, konketenasiya operatori satrlarni ulash kabi amalllarni bajarishga hizmat qiladi.

Python 3 dasturlasj tilidagi operatorlarni batafsil ko’rib chiqamiz.

Matematik operatorlar

Sonlar ustida amallar bajarish uchun quyidagi operatorlardan foydalaniladi:

- **+** – qo’shish:

```
>>> 10 + 5              # butun son
15
>>> 12 .4 + 5.2          # haqiyqiy son
17.6
>>> 10 + 12 .4           # butun va haqiyqiy sonlar
22 .4
```

- **-** – ayirish:

```
>>> 10 - 5              # butun son
5
>>> 12 .4 - 5.2         # haqiyqiy son
7.2
>>> 12 - 5.2            # butun va haqiyqiy son
6.8
```

- * – ko’paytirish:

```
>>> 10 * 5              # butun son
50
>>> 12 .4 * 5.2         # haqiyqiy son
64 .4 8
```

```
>>>  10  *  5.2              #  butun va haqiyqiy son
52 .0
```

- / – bo'lish. Bo'lish natijasi doimo haqiyqiy son bo'ladi xatto, bo'linma va bo'luvchi butun son bo'lsa ham. Masalan:

```
>>> 10 / 5                  #  bo'linma qoldiqsiz
2.0
>>> 10 / 3                  #  bo'linma qoldiqli
3. 33333333 33333335
>>> 10 .0 / 5.0             #  bo'linuvchi va bo'luvchi haqiyqiy sonlar
2.0
>>> 10.0 / 3.0              #  bo'linuvchi va bo'luvchi haqiyqiy sonlar
3. 3333333333333335
>>>  10 / 5.0               #  butun son haqiyqiy songa bo'linsa
2.0
>>> 10.0 / 5                #  haqiyqiy son butun songa bo'linsa
2. 0
```

- // – kami bilan yaxlitlanadigan bo'lish. Barcha qoldiq tashlab yuboriladi. Maslan:

```
>>> 10 // 5              # bo'linma qoldiqsiz
2
>>>10 // 3              # bo'linma qoldiqli
3
>>>10.0 // 5.0          # haqiyqiy sonli bo'linma
2.0
>>> 10 .0 // 3.0         # haqiyqiy sonli bo'linma (qoldiqli)
3.0
>>> 10 // 5.0            # butun son haqiyqiy songa bo'linganda (qoldiqsiz)
2.0
>>>  10 // 3.0           # butun son haqiyqiy songa bo'linganda(qoldiqli)
3.0
>>>  10 .0 // 5          # haqiyqiy son butun songa bo'linganda(qoldiqsiz)
2.0
>>>  10 .0  11  3        # haqiyqiy son butun songa bo'linganda(qoldiqli)
3.0
```

- % - qoldiqli bo'lish (bo'linma qoldiq qismi):

```
>>> 10 % 5              # butun sonli qoldiqsiz bo'linma
0
>>> 10 % 3               #  butun sonli qoldiqli bo'linma
1
>>> 10.0 % 5.0          #  haqiyqiy sonlar ustida amal
0.0
```

```
>>> 10.0 % 3.0          #  haqiyqiy sonlar ustida amal
1.0
>>>  10  %  5.0         #  butun va haqiyqiy sonlar ustida amal
0.0
>>> 10 % 3.0            #  butun va haqiyqiy sonlar ustida amal
1.0
>>>  10.0 % 5           #  haqiyqiy va butun sonlar ustida amal
0.0
>>>  10.0  %  3         #  haqiyqiy va butun sonlar ustida amal
1.0
```

- ** – darajaga ko'tarish:

```
>>>  10**2,  10.0**2
(100,  100.0)
```

- Unar minus (-) va unar plyus (+) :

```
>>>+10,  +10.0,  -10,  -10 .0 ,  -( - 10) ,  -( -10 .0)
(10,  10.0,  -10,  - 10.0,  10, 10 .0)
```

Ikkilik operatorlar

Pythonda quyidagi ikkilik (bitli) operatorlar ishlatiladi:

- ~ ikkilik inversiya. Har bir bit qarama qarshisiga o'zgaradi:

```
>>> x = 100             #  011 001 00
>>> x= ~x               #  100 110 11
```

- & – ikkilik VA:

```
>>> x  = 100            #  01100100
>>> y  = 75             #  01001011
>>> z  = x & y # 01000000
>>> "{0 :b } & {l: b}={2:b}".format (x, y , z)
'1100100 & 1001011 = 1000000 '
```

- | – ikkilik YOKI:

```
>>>  x  =  100                #  01100100
>>>  y  =  75                 #  01001011
>>>  z  =  x | y              #  01101111
>>> "{0:b } | {l: b } = {2: b }".format (x,  y,  z )
'1100100 | 1001011  =  11011 11 '
```

- ^ - ikkilik istisnoli YOKI:

```
>>>  x  =  100                # 01100100
>>>  y  =  250                # 11111010
>>>  z  =  x ^ y              # 10011110
>>>  "{ 0: b} ^{l :b}  =  {2 :b} ".format (x,  y,  z )
```

'1100100 ^ 11111010 = 1001111 0 '

- << – chapga siljitish –ikkilik ko'rinishdagi soni chapgabir yoki bir nechta o'ringa siljitish va o'ng tomonni nol bilan to'ldirish:

```
>>> x = 100          # 01100100
>>> y = x << 1 # 11001000
>>> z = y << 1 # 10010000
>>> k = z << 2  # 01000000
```

- >> – o'ngga siljitish – ikkilik ko'rinishdagi sonni o'ngga bir yoki bir nechta o'ringa siljitish va agar son musbat bo'lsa, chap tomonni nol bilan to'ldirish:

```
>>> x = 100          # 01100100
>>> y = x >> 1       # 00110010
>>> z = y >> 1       # 00011001
>>> k = z >> 2  # 00000110
```

Agar son manfiy bo'lsa, chap tomondagi razryadar bir bilan to'ldiriladi:

```
>>> x = -127         # 10000001
>>> y = x >> 1       # 11000000
>>> z = y >> 2       # 11110000
>>> k = z << 1       # 11100000
>>> m = k >> 1       # 11110000
```

Sonli ketma-ketliklar bilan ishlash operatorlari

Sonli ketma-ketliklar bilan ishlash operatorlari quyidagicha:

- + - konkatenasiya:

```
>>> print ("1-satr" + "2-satr")      # satrlar konkatenasiyasi
1-satr2-satr
>>> [1, 2, 3] + [4, 5, 6 )          # ro'yxatlar
[1, 2, 3, 4, 5, 6]
>>> (1, 2, 3) + ( 4, 5, 6)          # Kortejlar
(1, 2, 3, 4, 5, 6)
```

- * – takrorlash:

```
>>> "s" * 20                        # Satr
'ssssssssssssssssssss'
>>> [1, 2]* 3                       # Ro'yxat
[1, 2, 1, 2, 1, 2]
>>> (1, 2) * 3                      # Kortej
(1, 2, 1, 2, 1, 2)
```

- in – tegishlilikka tekshirish. Agar element sonli satr tarkibida bo'lsa natija sifatida True ni qaytaradi:

```
>>> "satr" in "Qidirish uchun satr"      # Satr
```

```
True
>>> "2-satr" in "Qidirish uchun satr"        # Satr
False
>>> 2 in [1, 2, 3],  4 in [1, 2, 3]         # Ro'yxat
(True , False)
>>> 2 in (1 , 2, 3), 6 in (1, 2, 3)          # Kortej
(True , False)
```

- not in – tegishli emaslikka tekshirish. Agar element sonli ketma-ketlik tarkibida bo'lmasa True natija qaytaradi:

```
>>> "satr" not in "Qidirish uchun satr"              # Satr
False
>>> "2-satr" not in "Qidirish uchun satr"          # Satr
True
>>> 2 not in [1, 2, 3], 4 not in [1, 2,]          # Ro'yxat
(False, True )
>>> 2 not in (1, 2, 3), 6 not in (1 , 2, 3)       # Kortej
(False, True)
```

O'zlashtirish operatorlari

O'zlashtirish operatorlari qiymatlarni o'zgaruvchilarga saqlash uchun foydalaniladi. Pythonda foydalaniladigan o'zlashtirish operatorlarini ko'rib chiqamiz:

= – o'zgaruvhciga qiymat o'zlashtirish:

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

- += – o'zgaruvchi qiymatini ko'rsatilgan kattalikka oshirish:

```
>>> x = 5;
>>> x += 10                # x=x+10 ga teng kuchli
>>> x
15
```

Konkatenasiya amalini o'zlashtirish uchun ham foydalanish mumkin:

```
>>> s = "Ol" ;
>>> s += "mos"
>>> print (s)
Olmos
```

- - = – o'zgaruvchi qiymatini ko'rsatilgan kattalikka kamaytirish:

```
>>> x = 10; x - = 5  # x = x - 5 ga teng kuchli
>>> x
5
```

- * = – o'zgaruvchi qiymatini ko'rsatilgan kattalikka ko'paytirish:

```
>>> x = 10; x * = 5  # x = x * 5 ga teng kuchli
>>> x
```

50
Sonli ketma ketliklar uchun ham * = amali qo'llaniladi:

```
>>> s = "* "; s * = 20
>>> s
' ********************* '
```

- / = – o'zgaruvchi qiymatini ko'rsatilgan kattalikka bo'lish:

```
>>> x = 10 ; x /= 3          # x = x / 3 ga teng kuchli
>>> x
3. 3333333333333335
>>> y = 10 .0 ; y /= 3.0          # y = y / 3.0 ga teng kuchli
>>> y
3. 3333333333333335
```

- / /= – o'zgaruvchi qiymatini ko'rsatilgan songa bo'linadi, kami bo'yicha yaxlitlanadi va o'zlashtiriladi:

```
>>> x = 10; x //= 3   # x = x // 3 ga teng kuchli
>>> x
3
>>> y= 10 .0 ; y //= 3.0          # y= y // 3.0 ga teng kuchli
>>> y
3.0
```

- % = – o'zgaruvchi qiymatini ko'rsatilag songa bo'ladi va qoldiq qismi o'zlashtiriladi:

```
>>> x = 10; x %= 2          # x = x % 2 ga teng kuchli
>>> x
0
>>> y = 10; y %= 3          # y = y % 3 ga teng kuchli
>>> y
1
```

- ** = – o'zgaruvchi qiymatini ko'rsatilgan darajaga ko'taradi va o'zlashtiriladi:

```
>>> x = 10 ; x * * = 2          # x = x * * 2 ga teng kuchli
>>> x
100
```

Operatorlarni bajarishda ustuvorlik

Operatorlar bajarilish ustuvorligini kamayish tartibida yozamiz:

1. -x, +x, -x, ** – unar minus, unar plyus, ikkilik inversiya, darajaga ko'tarish. Agar unar operator * * operatoridan chapda joylashgan bo'lsa, darajaga ko'tarish katta ustuvorlikka ega, agar o'ngda joylashgan bo'lsa, kichik ustuvorlikka ega. Maslan:

- 10 * * -2 ifodasi quyidagi qavsli ifodaga teng kuchli:

- (10 ** (-2))

2. *, %, /, // – ko'paytirish(takrorlash), qoldiqli bo'lish, bo'lish, kami bo'yicha yaxlitlab bo'lish.

3. +, - – qo’shish (konkatenasiya), ayirish.
4. <<, >> – ikkilik siljitish.
5. & – ikkilik VA.
6. ^ – ikkilik istisnoli YOKI.
7. | – ikkilik YOKI.
8. =, +=, -=, *=, /=, //=, %=, ** = – o’zlashtirish

Eslatib o’tamiz! Qavslar yordamida ifoda hisoblanishdagi ustuvorlik tartibini o’zgartirish mumkin.

Nazorat savollari

1.Pythonda operatorlar?
2.Pythonda matyematik operatorlar
3.Pythonda munosabat operatorlari.
4.Pythonda ketma-ketliklar bilan ishlash operatorlari.
5.Pythonda o’zlashtirish operatorlari.
6.Pythonda operatorlarni bajarish ketma-ketligi

4-ma’ruza. Shartli operatorlar. Taqqoslash operatorlari. if...else operatori

Reja

1. Shartli operatorlar.
2. Taqqoslash operatorlari.
3. if...else operatori

Shartli opearatorlar tegishli mantiqiy ifodaning qiymatiga ko’ra dasturning alohida bir qismini bajarilish yoki bajarilmasligini ta’minlaydi. Mantiqiy ifoda faqat ikkita qiymat qabul qiladi: *true* (rost) yoki *false* (yolg’on), hisoblashda 1 yoki 0 sifatida qabul qilinadi.

```
>>> True + 2          # 1 + 2 ga teng kuchli
3
>>> False + 2         # 0 + 2 ga teng kuchli
2
```

Mantiqiy qiymat yoki ifodani o’zgaruvchida saqlash mumkin:

```
>>> x = True ; y = False
>>> x, y
(True , Fal se )
```

Ixtiyoriy obyekt mantiqiy kontekstda rost (true) yoki yolg’on(False) shaklida interpretatsiya qilinadi. Mantiqiy qiymatni aniqlah uchun bool() funksiyasidan foydalanish mumkin. Quyidagi obyektlar True qiymatni qaytaradi:

- Nolga teng bo’lmagan ixtiyoriy son:

```
>>> bool (l), boo1 (20) , bool (- 20)
(True , True , True )
>>> bool (l.0), bool(0.1), bool (-20.0)
(True, True, True )
```

- Bo’sh bo’lmagan obyekt:

```
>>> bool ("0") , bool ([0, None] ), bool( (None, ) ), bool ({"x": 5})
(True, True, True, True)
```

Quyidagi obyektlar False kabi interpretasiyalanadi:

- Nolga teng son :

```
>>> bool (0) , bool (0.0)
(False, False)
```

- Bo’sh obyekt:

```
>>> bool (" "), bool ([ ]), bool (( ))
(False, False , False )
```

- None qiymati:

```
>>> bool (None )
False
```

Taqqoslash operatorlari

Taqqoslash operatorlaridan mantiqiy ifodalarda foydalaniladi. Ularni sanab o'tamiz:

- = = – teng:

```
>>> 1 = = 1, 1 = = 5
(True , False)
```

- ! = - teng emas:

```
>>> 1 ! = 5, 1 ! = 1
(True , False )
```

- < – kichik :

```
>>> 1 < 5, 1 < 0
(True , False)
```

- > – katta :

```
>>> 1 > 0, 1 > 5
(True , False )
```

- <= - kichik yoki teng:

```
>>> 1 <= 5, 1 <= 0, 1 <= 1
(True , False, True )
```

- >= - katta yoki teng:

```
>>> 1 >= 0, 1 >= 5, 1 >= 1
(True , False, True )
```

- in – sonli ketma-ketlikka tegishlilikni tekshirish:

```
>>> "satr" in "qidirish uchun satr"          # Satr
True
>>> 2 in [1 , 2, 3) , 4 in [1, 2, 3]        # Ro'yxat
(True , False)
>>> 2 in (1, 2, 3) ' 4 in (1, 2, 3)         # Kortejlar
(True , False )
```

in operatoridan shuningdek, lug'at kalitiga tegishlilikni tekshirishda ham foydalaniladi:

```
>>> "x" in {" x" : 1, "y" : 2}, "z" in {" x" : 1, "y" : 2}
(True , False )
```

- not in - sonli ketma-ketlikka tegishli emaslikni tekshirish:

```
>>> "satr" not in "qidirish uchun satr" # Satr
False
>>> 2 not in [1, 2, 3] , 4 not in [1, 2, 3]      # Ro'yxat
(False, True)
>>> 2 not in (1 , 2, 3) , 4 not in (1 , 2 , 3) # Kortej
(False , True )
```

- is – ikkita o'zgaruchi bir obyektga havola qilinganini tekshiradi. Agar o'zgaruvchilar bior obyektga havola qilingan bo'lsa, is operatori True qiymat qaytaradi:

```
>>> x = y = [1, 2)
>>> x is y
True
>>> x = [1, 2] ; y= [1, 2)
>>> x is y
False
```

Sonlar va uzun bo'lmagan satrlar bilan ishlashda interpretator samaradorligini oshirish uchun keshlashdan foydalanishi seziladi. Turli o'zgaruvchilarda saqlanadigan ikkita bir xil qiymat bitta obyektda saqlanishini va o'zgaruvchilarda bir obyektga havola saqlanishini anglatadi. Masalan:

```
>>> x = 2; y = 2; z = 2
>>> x is y, y is z
(True , True )
```

- is not – ikki o'zgaruvchi turli obyektlarga havola qilinishini tekshiradi. Agar shunday bo'lsa, True qiymat qaytaradi:

```
>>> x = y = [1, 2 ]
>>> x is not y
False
>>> x = [1 , 2] ; y= [1, 2)
>>> x is not y
True
```

Mantiqiy ifoda not operatori yordamida invertirlanadi (teskari qiymatga o'giriladi):

```
>>> x = 1; y = 1
>>> x = y
True
>>> not (x == y) , not x == y
(False , False )
```

Agar x va y o'zgaruvchilar teng bo'lsa True qiymat qaytariladi, not operatori qo'llanilgach esa ifoda False qaytaradi. Aylanama qavslarni qo'yish shart emas.

Mantiqiy ifodada birdaniga bir nechta shartni ko'rsatish mumkin:

```
>>> x = 10
>>> 1 < x < 20, 11 < x < 20
(True, False )
```

Bir nechta mantiqiy ifodalar bitta katta mantiqiy ifodaga quyidagilar operatorlar yordamida birlashtirilishi mumkin:

- and – mantiqiy VA. Agar x and y ifodadagi x False bo'lsa x ni, aks holda y qaytaradi:

```
>>> 1 < 5 and 2 < 5      # True and True == True
True
```

```
>>> 1 <  5  and 2 > 5                # True and False == False
False
>>> 1 > 5 and 2 < 5                  # False and True == False
False
>>> 10 and 20 ,  0 and 20, 10 and 0
( 20,  0,  0)
```

- or – mantiqiy YOKI. Agar x or y ifodada x False bo'lsa, y ni qolgan hollarda x ni qaytaradi:

```
>>> 1 <  5  or  2 <  5               # True  or True == True
True
>>> 1 < 5 or 2 >  5                  # True or False  ==  True
True
>>>  1  >  5  or  2  <  5            # False  or  True  ==  True
True
>>>  1  >  5  or  2  >  5             # False  or  False  ==  False
Fa lse
>>>  10  or  20,  O  or  20,  10  or  o
(1 0,  20,  10)
>>> 0  or  "" or  None or [] or "s"
's'
```

Taqqoslash operatorlarni ustuvorligini kamayish tartibida sanaymiz:

1. <, >, <=, >=, =, ! =, <>, is, is not, in, not in.
2. not – mantiqiy inkor.
3. and – Mantiqiy VA.
4. or – mantiqiy YOKI.

if...else tarmoqlanish operatori

if...else tarmoqlanish operatori mantiqiy ifoda natijasi bog'liq holda dastur qismining bajarilishi yoki bajarilmasligini ta'minlaydi. Operator umumiy holda quyidagi ko'rinishda bo'ladi:

if <mantiqiy ifoda>:

<agar mantiqiy ifoda rost qiymat qabul qilsa bajariladigan operatorlar to'plami>

[elif <mantiqiy ifoda>:

<agar mantiqiy ifoda rost qiymat qabul qilsa bajariladigan operatorlar to'plami>

]

[else:

<agar mantiqiy ifoda yolg'on qiymat qabul qilsa bajariladigan operatorlar to'plami>]

Kodlar blokini yozishda probellar ahamiyatini unutmaslik kerak. Quyidagi misolni ko'rib chiqamiz:

4.1-misol. Sonni juftlikka tekshirish

```
#  -*- coding :  ut f-8  - *-
x  =  int(input("Sonni kiriting:"))
if x % 2 == 0:
        print (x, " – juft son")
else:
        print (x, "  - toq son")
input ()
```

Agar blok bitta insturuksiyadan iborat bo’lsa, uni bir satrga yozish mumkin:

```
# -*-coding:utf-8-* -
x  = int(input("Sonni kiriting: ") )
if x%2 == 0: print (x , " – Juft son" )
else: print (x, " – Toq son ")
input ( )
```

Bu holda satr ohiri instruksiya ohiri hisoblanadi. Blok ichida bir nechta operator bo’lsa ularni bir satrga nuqtali vergul bilan ajratib yozish mumkin:

```
- * - coding:  utf-8 - * -
x = int(input ("Sonni kiriting:"))
if x%2 == 0: print(x , end=" " ) ; print (" – juft son" )
else: print (x, end=" " ) ; print ( " – toq son" )
input( )
```

Eslatma! Amalda operatorlarni bir satrga nuqtali vergul bilan ajratib yozish tavsiya etilmaydi.

if ..else operatori birdaniga bir nechat shartlarni tekshirishga imkon beradi. Misol orqali ko’rib chiqamiz (4.2-misol)

4.2-misol. Bir nechta shartlarni tekshirish

```
#  -*- coding : utf-8 - * -
print ("""Qaysi operatson tizimdan foyadalanasiz?
1 - Windows 8
2 - Windows 7
3 -Windows Vista
4 - Windows XP
5 - boshqa""")
os = input ("Javobingizga mos sonni kiriting:")
if os == "1":
        print ("Siz Windows 8 ni tanladingiz")
elif os == "2":
        print (" Siz Windows 7 ni tanladingiz")
elif os == "3":
        print ("Siz Windows Vistani tanladingiz" )
elif  os == "4":
        print ("Siz Windows XP ni tanladingiz")
```

```
elif os = = "5":
        print ("Siz boshqa ni tanladingiz")
elif not os :
        print ("Son kiritmadingiz")
else:
        print ("Siz tanlagan OT ni aniqlay olmadik")
input( )
```

if ... else operatori yana bir boshqa formatda ham qo'llaniladi:

<O'zgaruvchi> = <rost> if <shart> else <yolg'on>

Misol:

```
>>>print ("Ha" if 10%2 == 0 else  "Yo'q" )
Ha
>>> s  =  "Ha"  if 10 % 2 == 0 else  "Yo'q"
>>> s
'Ha'
>>> s = "Ha" if 11% 2 ==0 else "Yo'q"
>>> s
'Yo`q'
```

Nazorat savollari

1. Pythonda shartli operatorlar.
2. Pythonda taqqoslash operatorlari.
3. Pythonda if...else operatori

5-ma’ruza. Sikl operatorlari. For sikli. range() va enumerate() funksiyalari While sikli. continue operatori. break operatori

Reja

1. Sikl operatorlari.
2. For sikli.
3. range() va enumerate() funksiyalari
4. While sikli.
5. continue operatori.
6. break operatori

For sikli

for sikl operatoridan takrorlanishlarni hisoblash va ketma-ketliklar bilan ishlash uchun foydalaniladi. Umumiy ko’rinishi quyidagicha:

for <Joriy element> in <Ketma-ketlik> :
<Sikl ichki instruksiyalari>
[else:
<break operatoridan foydalanilmaganda bajariluvchi blok>
]

Bu yerda:

<Ketma-ketlik> – iterasiya mexanizmini qo’llovchi obyekt. Masalan: satr, ro’yxat, kortej, diapason, lug’at va boshqalar;

<Joriy element> – har bir iterasiyada ushbu parameter orqali ketma-ketlik yoki lug’atning joriy elementiga yo’l ko’rsatiladi;

<Sikl ichki instruksiyalari> – ko’p marta bajariluvchi blok;

Agar sikl ichida break operatori ishlatilmasa, sikl bajarilishi tugashi bilan else instruksiyasi bajarilishiga o’tiladi. Bu blok majburiy emas.

So’zdagi harflarni saralsh misolini ko’rib chiqamiz:

4.4-misol. So’zdagi harflarni saralash

```
for s in "Assalomu alaykum":
  print(s, end=" ")
else:
  print ("\nsikl bajarildi")
```

Natija:

```
A s s a l o m u  a l a y k u m
sikl bajarildi
```

Endi ro’yxat va korej har bir elementini alohida satrda chop etamiz (4.5-misol)

4.5-misol. Ro’yxat va kortejni saralash

```
for x in [1,2,3]:
  print(x)
for y in (1,2,3):
```

```
  print(y)
```

4.6-misol. Lug’at elementlarini saralash

```
>>> arr = {"x" : 1, "y" : 2, " z" : 3}
>>> arr.keys ()
dict_keys ([ 'y ', 'x', 'z'))
>>>for key in arr.keys():     # keys () metodian foydalanilmoqda
print (key, arr [key] )
y 2
x 1
z 3
>>> for key in arr:           # lug’atada iterasiyadan foydalanilmoqda
print (key, arr [key])
y 2
x 1
z 3
```

range() va enumerate() funksiyalari

Shu paytgacha biz sonli ketma-ketliklar (satrlar) elementlarini chop etdik. Endi ro’yxat har bir elementini 2 ga ko’paytirib chop etamiz:

```
arr = (1, 2, 3)
for i in arr:
i = i * 2
print (arr)                    # Natija: (1, 2, 3)
```

Ko’rinib turibdiki, ko’payrtma elementlarga ta’sir etmadi, ya’ni elementni o’ziga ta’sir etib bo’lmadi. Har bir elementga kirish range() funksiyasi yordamida amalga oshiriladi. U quyidagi formatga ega:

range ([<boshlang’ich qiymat>,] <Tugash> [, <Qadam>])

Birichi parametr boshlang’ich qiymatni aniqlaydi. Agar u ko’rsatilmasa, boshlang’ich qiymat 0 dan boshlanadi. Ikkinchi parameter tugash cheagarasini bildiradi. Bu parameter majburiydir. <Qadam> parametri ko’rsatilmasa qadam 1 deb qabul qilinadi. Misol sifatida ro’yxat har bir elementini 2 ga ko’paytitrishni ko’rib chiqamiz:

4.8-misol. range() funksiyasidan foydalanish misoli

```
arr=[1,2,3]
for i in range(len(arr)):
  arr[i]*=2
print(arr )                # Natija: [2, 4, 6]
```

range() funksiyasidan foydalanisahga doir bir nechta misolarni ko’rib chiqamiz. 1 danm 10 gacha bo’lgan butun sonlarni chop etamiz:

```
for i in range (l, 101):  print (i)
```

Qiymatni nafaqat oshirib boorish balki, kamaytirish ham mumkin. 100 dan 1 gacha

barcha sonlarni chop etamiz:

```
for  i  in  range (100,0, -1) :  print (i)
```

Shuningdek qadam faqat bittaga emas hohlagan songa o’zgarib borishi mumkin. 1 dan 100 gahca juft sonlarni chop etamiz:

```
for  i in range (2,  101,  2) : print(i)
```

While sikli

while siklidagi instruksiya mantiqiy ifoda rost bo’lganda bajariladi. While sikli quyidai formatga ega:

<Boshlang’ich qiymat>
while <Shart>:
<Instruksiya>
< Orttirma >
[else:
<agar break operatori ishlatilmasa, bajariluvchi blok>
]

While siklining bajarilish tartibi:

1. O’zgaruvchi- hisoblagich boshlang’ich qiymat o’zlashtiradi.
2. Shart tekshiriladi, agar rost bo’lsa sikl ichidagi instruksiya bajariladi, aks holda sikl yakunlanadi.
3. O’zgaruvchi- hisoblagich <Orttirma> dagi parameter bo’yicha o’zgaradi.
4. 2 qadamga o’tiladi.
5. Agar sikl ichida break operatoridan foydalailmasa, sikl yakunlangach else bloki bajariladi. Bu blok majburiy emas.

1 dan 100 gacha sonlarni while siklodan foydalnib chop etamiz (4.11-misol).

4.11-misol. 1 dan 100 gacha sonlarni chop etish

```
i  =  1  #  <Boshlang’ich qiymat>
while  i < 101 :              #  <Shart>
        print(i)              # <Instuksiya>
        i += 1          # <Orttirma>
```

Eslatma! Agar orttirma ko’rsatilmasa, sikl cheksiz davom etadi. Cheksiz siklni to’xtatish uchun esa <Ctrl>+<C> klavish kombinasiayasdan foydalaniladi.

100 dan 1 gacha barcha butun sonlarni chop etamiz (4.12-misol).

4.12-misol. 100 dan 1 gacha sonlarni chop etish

```
i = 100
while i:
        print (i)
        i - = 1
```

E’tibor bergan bo’lsangiz, sikl shartida mantiqiy ifodadan foydalinmadi. Siklning har bir iterasiyasida sanagich-o’zgaruvchi qiymati bittaga kamayib boradi. Oqibatda u

qachondir 0 ga teng bo'ladi. Bilamizki, 0 mantiqiy amal natijasi sifatida False ga teng kuchli. Undan boshqa ixtiyoriy son esa True ga teng kuchli.

While strukturasi yordamida turli struktura elementlari ustida ishlash mumki. Biroq bu holda while sikli for sikliga nisbatan sekinroq ishlaydi. Misol sifatida ro'yxatning har bir elementini 2 ga ko'paytirishni ko'rib chiqamiz (4.13-misol).

4.13-misol. Ro'yxaty elementlarini saralsh

```
arr = [1, 2, 3]
i, count = 0, len(arr)
while  i<count:
  arr[i]*=2
  i+=1
print(arr)              #  Natija:  [2,  4,  6]
```

continue operatori. Siklning navbatdagi itersiyasiga o'tish

continue operatori sikldagi barcha instruksiyalar bajarilmay, siklning navbatdagi iterasiyasiga o'tish imkonini beradi. Misol sifatida 1 dan 100 gacha bo'lagan butun sonlardan 5 dan 10 gacha oraliqqa kirmaydigan butun sonlar chop etishni ko'rib chiqamiz (4.14-misol).

4.14-misol. continue operatori

```
for i in range (l, 101):
    if  4 < i < 11:
        continue      # siklning keying iterasiyasiga o'tamiz
    print (i )
```

break operatori. Siklni to'xtatish

break operatori sikl bajarilishini zudlik bilan to'xtatish imkonini beradi. 1 dan l00 butun sonlarni chop etishning yana bir usulini ko'rib chiqamiz (4.15-misol).

4.15-misol. Break operatori

```
i  = 1
while True:
        if i > 100:
                break #  siklni to'xtatish
        print (i)
i  + = 1
```

Bu yerda biz shart qiymati sifatida True ni ko'rsatdik. Bu holda sikl cheksiz davom etadi. Biroq break operatoridan foydalanilgani sababli faqat 100 gacha sonlar chop etiladi.

Eslatma! break operatori dastur bajarilishi emas, sikl bajarilishini to'xtatadi va sikldan keyingi ko'rsatma bajariladi.

While siklini break operatori bilan birgalikda ishlatish oldindan ma'lumot miqdori noaniq bo'lgan hollarda ishlatish qulay. Misol sifatida noaniq miqdordagi sonlar yig'indisini hisoblasha dasturini ko'rib chiqamiz (4.16-misol).

4.16-misol. Noaniq miqdordagi sonlar yig'indisini hisoblash

```
print("Natiaja olish uchun 'stop'  so’zini kiriting")
summa=0
while True:
  x=input("Sonni kiriting:")
  if x == "stop":
    break
  x=int(x)
  summa+=x
print("Sonlar yig’indisi: ",  summa)
```

Natija:

Natiaja olish uchun 'stop' so’zini kiriting
Sonni kiriting:10
Sonni kiriting:15
Sonni kiriting:30
Sonni kiriting:stop
Sonlar yig’indisi: 55

Nazorat savollari

1. Pythonda sikl operatorlari.
2. For sikli.
3. range() va enumerate() funksiyalari
4. Pythonda while sikli.
5. Pythonda continue operatori.
6. Pythonda break operatori

6-ma’ruza. Sonlar. Sonlar bilan ishlashning tashqi funksiya va metodlari. *math* moduli. Matematik funksiyalar. *random* moduli. Tasodifiy son generasiyasi

Reja

1. Sonlar.
2. Sonlar bilan ishlashning tashqi funksiya va metodlari.
3. *math* moduli.
4. Matematik funksiyalar.
5. *random* moduli. Tasodifiy son generasiyasi

Python 3 dasturlash tili quyidagi sonli turlarni qo’llaydi:

- int - butun son (45, 6, -7, 0, 124). Son kattaligi faqat tezkor xotira sig’imiga bog’liq;
- float – haqiyqiy son (2.2101, 0.0025, 7.987456);
- complex – kompleks son (2+5j, 3+2j).

Butun son eng sodda tip, haqiyqiy son murakkabroq va eng murakkabi kompleks sondir. Shu tarzda agar amalda butun va haqiyqiy son ishtirok etsa, butun son avtomatik haqiyqiy songa o’tkaziladi so’ng haqiyqiy sonlar ustida amallar bajariladi. Bu amal natijasi haqiyqiy son bo’ladi.

Butun sonli obyektni oddiy usulda yarish mumkin:

```
>>> x = 0; y = 10; z = -80
>>> x, y, z
(0, 10, -80)
```

Bundan tashqari sonni ikkilik, sakkizlik yoki o’n oltilik shaklda ham ko’rsatish mumkin. Bunday sonlar avtomatik o’nlik butun songa o’zgartiriladi.

- 0b (yoki 0B) belgisidan boshlanuvchi hamda 0 va 1 raqamlari kombinatsiyasidan iborat ikkilik son:

```
>>> 0b11111111, 0b101101
(255, 45)
```

- 0 va o belgisidan boshlanuvchi hamda 0 dan 7 gacha raqamlar kombinatsiyasidan iborat sakkizlik son:

```
>>> 0o7 , 0o12, 0o777, 0o7 , 0o12, 0o777
(7, 10, 511, 7, 10, 51 1)
```

- 0x (yoki 0X) belgisidan boshlanuvchi hamda 0 dan 9 gacha raqamlar, A dan F gacha harflar kombinatsiyasidan iborat o’n oltilik son:

```
>>> 0x9, 0xA, 0x10, 0xFFF, 0xfff
(9, 10, 16, 4095, 40 95)
```

- Haqiyqiy son nuqta va (yoki) eksponsial ifodalash shaklidan iborat sonlar kombinasiyasi orqali ifodalanadi:

```
>>> 10., .14, 3.14, 11E20, 2.5e-12
```

(10.0, 0.14, 3.14, 1.le+21, 2.5e-12)

Haqiyqiy sonlar ustida amallarni bajarishda hisob aniqligi chegarasini e'tiborga olish kerak bo'ladi.

Masalan, keyingi amal natijasi g'ayritabiy ko'rinishi mumkin:

```
>>> 0.3 - 0.1  - 0.1  - 0.1
-2 . 77555 75615 62891 4e-17
```

Biz aslida 0.0 natijani kutgan edik, lekin biz kurib kurganimizdek boshqa natijaga ega boldik. Agar fiksirlangan natijaga erishmoqchi bo'lsak decimal modulidan foydalanishimiz mumkin:

```
>>> from decimal import  Decimal
>>>  Decimal("0.3")-Decimal("0.1")-Decimal("0.1")-Decimal("0.1")
Decimal('0.0')
```

Bundan tashqari fractions moduli yordamida kasr sonni ham ifodalash mumkin. Kasr soni yaratishda ikki sonni yani surat va mahrajni bitta son yoki satr sifatida ko'rsatish mumkin.

Misol sifatida 4/5 kasr sonini yaratamiz:

```
>>> from  fractions  import Fraction
>>> Fraction (4, 5)
Fraction (4,  5)
```

½ kasr sonini quyidagi uchta usul bilan yaratish mumkin:

```
>>> Fraction (l,  2)
Fraction (l,  2)
>>>  Fraction  ("0.5" )
Fraction (l,  2)
>>>  Fraction (0 .5 )
Fraction (l,  2)
```

Kasr sonlar ustida oddiy sonlar kabi arifmetik amallar bajarish mumkin:

```
>>> Fraction (9, 5) - Fraction (2,  3)
Fraction (l7,  15)
>>> Fraction ("0.3")  - Fraction ("0.1")  - Fraction (" 0.l ")  - Fraction (" 0.1 ")
Fraction (0,1)
>>>  float (Fraction (0,  1))
0.0
```

Kompleks son quyidagi formatda yoziladi:

<Haqiyqiy qism> + <Mavhum qism>J

Bu yerda J ixtiyoriy registrda (katta yoki kichik) bo'lishi mumkin. Kompleks sonlarga doir misollar:

```
>>>  2+ 5J,  8j
((2+ 5j),  8j )
```

Sonlar bilan ishlashning tashqi funksiya va metodlari

Sonlar bian ishlash uchun quyidagi tashqi funksiyalar mo'ljallangan:

- int ([<Obyekt>[, <Sanoq sistemasi>]]) – obyektni butun sonaga o'tkazish. Ikkinchi parametr sanoq sistemasini anglatib, ko'rsatish majburiy emas (odatiy holda 10 lik deb qabul qilinadi). Masalan:

```
>>>  int(7. 5),  int ("71",  10) ,  int ("0o71",  8) ,  int ("0xA",  16)
(7,  71,  57,  10)
>>>  int(),  int ("0b11111111",  2)
(0,  255 )
```

- float ([<Son yoki satr>]) - son yoki satrni haqiyqiy songa o'tkazish. Masalan:

```
>>> float (7),  float ("7.1") ,  float ("12.")
(7 .0,  7.1, 12.0)
>>>float ("inf") ,  float ("-Infinity" ),  float ("nan" )
(inf, -inf,  nan)
>>>  float  ()
0.0
```

- bin (<Son>) – o'nlik sonni ikkilikka o'tkazish. Sonlarni satr ko'rinishida taqdim qilinadi. Masalan:

```
>>> bin(255),  bin(l) ,  bin(- 45)
('0b1111111' ,  '0b1' ,  '-0b101101')
```

- oct (<Son>) - o'nlik sonni sakkizlik songa o'tkazish. Sonlarni satr ko'rinishida taqdim qilinadi. Masalan:

```
>>>  oct(7),  oct(8),  oct(64)
('0o7 '  ,  '0ol0 '  ,  '0ol00'  )
```

- hex (<Son>) - o'nlik sonni o'n oltilik songa o'tkazish. Sonlarni satr ko'rinishida taqdim qilinadi. Masalan:

```
>>>  hex (l0) ,  hex(l6) ,  hex (255)
('0xa' ,  '0xl0' ,  '0xff ')
```

- round (<Son> [, <Nuqtadan ketyingi belgilar soni>]) – kasr qismi 0.5 dan kichik bo'lsa sonni o'zi, katta bo'lsa eng yaqin katta songa aylantiriladi. Agar kasr qismi 0.5 ga teng bo'lsa, eng yaqin juft songa aylantiriladi. Masalan:

```
>>> round (0.49),  round (0.50),  round(0.51)
(0, 0, 1)
>>>  round (1.49), round(l.50),  round (l.51)
(1, 2,  2)
>>>  round  (2.49),  round(2.50),  round(2.51)
(2, 2, 3)
>>> round(3.49),  round(3.50),  round(3.51)
(3,  4,  4)
```

Ikkinchi parametrverguldan keying raqamlar sonini olish uchun ko'rsatiladi. Biroq majburiy emas, ko'rsatilmasa 0 deb qabul qilinadi (son butunga yahlitlanadi):

```
>>> round (l.524, 2),  round (l.525,  2),  round (l.5555,  3)
(1.52,  1.52,  1.556)
```

- abs (<Son>) - sonning absolyut qiymatini qaytaradi:

```
>>>  abs (-10) ,  abs (l0) ,  abs (-12.5 )
(1 0,  10,  12 .  5)
```

- pow (<Son>, <Daraja> [, <Bo'luvchi>]) - <Son>ni <Daraja>ga ko'tarish:

```
>>>  pow (l0,  2),  10**2,  pow (3, 3),  3**3
(100,  100,  27,  27 )
```

Agar vuchinchi parametr ko'rsatilsa, olingan natijani shu parametrga bo'lishda qoldiq qismi qaytariladi:

```
>>>  pow(l0, 2, 2) ,  (10 ** 2)%2,  pow(3, 3, 2),  (3**3) % 2
(0, 0, 1, 1)
```

- max (<Vergul bilan ajratilga sonlar ro'yxati>) - ro'yxatdagi sonlarning eng kattasini aniqlash:

```
>>>  max (l,  2,  3) ,  max (3,  2, 3,  1),  max (l,  1.0 ) ,  max (l.0,  1)
(3,  3,  1,  1.0 )
```

- min (<Vergul bilan ajratilga sonlar ro'yxati>) - ro'yxatdagi sonlarning eng kichigini aniqlash:

```
>>>  min(l,  2,  3) ,  min(3,  2,  3,  1) ,  min (l,  1.0) ,  min (l.0,  1)
(1,  1,  1,  1.0 )
```

- sum (<Ketma-ketlik> [, <Boshlang'ich qiymat>]) - ketma-ketlik (masalan: ro'yxat, kortej) elementlari yig'indisini aniqlash plyus <Boshlang'ich qiymat>. Agar ikkinchi parametr ko'rsatilmasa, u 0 ga teng deb qabul qilinadi. Agar ketma-ketlik bo'sh bo'lsa, ikkinchi parametr qiymati olinadi. Masalan:

```
>>>  sum ((10,  20,  30,  40 )),  sum ([10,  20,  30,  40])
(100 ,  100)
>>> sum ([10,  20,  30 ,  40] ,  2) ,  sum ([ ],  2)
(102 ,  2)
```

- divmod (x, y) - ikki qiymatli (birinchi qiymat –bo'linma, ikkinchi qiymat–qoldiq (x//y, x%y)) kortej qaytaradi:

```
>>>  divmod (l3,  2)              #  13  ==  6  *  2  +  1
(6,  1)
>>> 13//2,  13%2
(6, 1)
>>> divmod (13.5, 2.0)  #  13 .5 == 6.0  *  2.0  +  1.5
(6 .0,  1.5 )
>>> 13.5 // 2.0,  13 .5%2.0
```

(6.0 , 1.5)

math moduli. Matematik funksiyalar

math moduli sonlar bilan ishlash uchun qo'shimcha funksiyalarni o'z ichiga oladi. Bu modulan foydalanishdan oldin quyidagi instruksiya bo'yicha chaqirib olinadi:

import math

Kompleks sonlar bilan ishlashda cmath kutubxonasidan foydalanish zarur. Math moduli quyidagi standart konstantalarni taqdim qiladi:

- pi – pi sonini qaytaradi:

```
>>> import math
>>> math.pi
      3.141592653589793
```

- e – e konstantasi qiymatini qaytaradi:

```
>>> math.e
      2.718281828459045
```

Sonlar bilan ishlashning asosiy funksiyalarni sanab o'tamiz:

- sin(), cos(), tan() – standart trigonoometrik funksiyalar (sinus, kosinus, tangens). Qiymatlar radianda beriladi;
- asin (), acos (), atan () - teskari trigonometric funksiyalar (arksinus, arkkosinus,arktangens). Natija radianda qaytariladi;
- degrees () – radiandan gradusga o'tkazish:

```
>>>  math.degrees (math.pi)
180 .0
```

- radians () – gradusdan radianga o'tkazish:

```
>>> math.radians(l80.0)
3.141592653589793
```

- exp() - eksponenta;
- log (<Son> [, <Asos>]) - ko'rsatilga asos bo'yicha logarifm. Agar asos ko'rsatilmasa natural logarifm deb hisoblanadi (e asosga ko'ra);
- log10 () – o'nlik logarifm ;
- log2 () – 2 asosga ko'ra logarifm;
- sqrt () – kvadrat ildiz:

```
>>> math.sqrt(100),  math.sqrt(25)
(10.0, 5.0)
```

- ceil () – eng yaqin katta songa yaxlitlash:

```
>>> math.ceil(5.49),  math.ceil(5.50),  math.ceil(5.51)
(6, 6, 6)
```

- floor () - – eng yaqin kichik songa yaxlitlash:

```
>>> math.floor(5.49),  math.floor (5.50),  math.floor(5.51)
```

(5, 5, 5)

- pow (<Son>, <Daraja>) –<Sonni> ni <Daraja>ga ko'tarish:

```
>>> math.pow(10, 2), 10**2, math.pow(3, 3), 3**3
(100.0, 100, 27.0, 27 )
```

- fabs () – absolyut qiymat:

```
>>> math.fabs(10), math.fabs(-10) , math.fabs(-12.5)
(10.0, 10.0, 12.5)
```

- fmod() – qoldiqli bo'lish:

```
>>> math.fmod (l0, 5) , 10%5, math.fmod (l0, 3), 10%3
(0.0, 0, 1. 0, 1)
```

- factorial () – son faktoriali:

```
>>> math.factoria1(5), math.factorial(6)
(120, 720)
```

- fsum (<Sonlar ro'yxati>) - berilgan ro'yxatdagi barcha sonlar yig'indisini aniqlash:

```
> > > sum ([.1, .1, .1, .1, .1, .1, .1, .1, .1, .1])
0. 9999999999999999
>>> fsum ([ .1, .1 , .1 , .1 , .1, .1 , .1 , .1 , .1 , .1 ] )
1.0
```

random moduli. **random moduli. Tasodifiy sonlar generasiyasi**

random moduli tasodifiy sonlarni generasiya qilishga imkon beradi. Moduldan foydalanishdan oldin quyidagi instruksiya yordamida bog'lanish amalga oshiriladi:

```
import random
```

Uning asosiy funksiyalarini sanab o'tamiz:

- random() – 0.0 dan 1.0 gacha tasodifiy sonlarni qaytaradi:

```
>>> import random
>>> random.random ()
0. 9753144027290991
>>> random.random ()
0.5468390487484339
>>> random.random()
0. 13015058054767736
```

- seed ([<Parametr>] [, version=2]) - tasodifiy sonlar generatorini yangi ketmaketlikka sozlash. Agar birinchi element ko'rsatilmasa, tasodifiy sonlar bazasi sifatida tizim vaqti olinadi. Agar birinchi parametr qiymati bitta bo'lsa, bitta songacha generasiyalanadi:

```
>>> random.seed (10)
>>> random.random ()
0. 5714025946899135
>>> random.seed(l0)
```

```
>>>  random.random()
0. 5714025946899135
```

- uniform (<Boshlanish> , <Tugash>) - <Boshlanish> dan <Tugash> gacha oraliqdagi tasodifiy haqiyqiy sonni qaytaradi:

```
>>> random.uniform (0,  10)
9. 965569925394552
>>> random.uniform (0,  10)
0.4455638245043303
```

- randint (<Boshlanish> , <Tugash>) - <Boshlanish> dan <Tugash> gacha oraliqdagi tasodifiy butun sonni qaytaradi:

```
>>>  random.randint (0, 10)
4
>>>  random.randint (0,  10)
10
```

- randrange ([<Boshlanish>,] <Tugash> [, <Qadam>]) - sonli ketma-ketlikdan tasodifiy elementni qaytaradi. Parametrlari range() funksiyalari analogidir. Ya'ni, <Boshlanish> dan <Tugash>gacha bo'lgan sonlar orlaig'idan <Qadam> farqi bilan tasodifiy sonni qaytarardi:

```
>>> random. randrange(10)
5
>>> random.randrange (0, 10)
2
>>> random.randrange (0, 10, 2)
6
```

- choice (<Ketma-ketlik>) – berilgan ketma-ketlik (satr, ro'yxat, kortej) dagi tasodifiy elementni qaytaradi:

```
>>> random.choice ("string")                    #  Satr
'i'
>>> random.choice (["s", "t", "r"])             #  Ro'yxat
'r'
>>> random.choice((" s", "t", "r"))              #  Kortej
't '
```

- shuffle (<Ro'yxat> [, <0.0 dan 1.0 gacha sonlar>]) – ro'yxat elementlarini tasodifiy aralashtirish. Agar ikkinchi parametr ko'rsatilmasa, random() funksiyasi qiymatidan foydalaniladi. Bu holda hech qanday natija qaytarilmaydi. Faqat ro'yxatdagi elementlar tartibi o'zgaradi. Masalan:

```
>>> arr = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
>>> random.shuffle(arr)
>>> arr
```

[8, 6, 9, 5, 3, 7, 2, 4, 10, 1]

- sample (<Ketma-ketlik>, <Elementlar soni>) –berilgan ketma-ketlikdan ko'rsatilgan miqdordagi tasodifiy holdagi elementlar ro'yxatini qaytaradi. Bunda obyekt sifatida iterasiyani qo'llovchi ixtiyoriy obektni ko'rsatish mumkin. Masalan:

```
>>> random.sample ("string",  2)
['i',  'r' ]
>>> arr =  [1,  2,  3,  4,  5,  6,  7,  8,  9,  10]
>>> random.sample(arr,  2)
[7, 10]
>>>  arr                                  #  Ro'yxatning o'zi o'zgarmaydi
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
>>>  random.sample (( l,  2,  3,  4,  5,  6,  7) ,  3)
[6 ,  3,  5]
>>>  random .sample (range(300), 5)
[126, 194, 272, 46, 71]
```

Berilgan uzunlikdagi parol genaratorini yaratish misolini ko'rib chiqamiz (5.1-misol). Buning uchun arr ro'yxatiga barcha ruxsat etilgan belgilarni qo'shamiz. So'ng choice() funksiyasi yordamida tasodifiy elementni olamiz. Odatiy holda parol uzunligini 8 belgidan iborat deb olinadi.

5.1-misol. Parol generator

```
#-*-coding:utf-8-*-
import random                    # random modulini ulaymiz
def passw_generator(count_char=8):
  arr = ['a', 'b',  'c', 'd','e','f','g', 'A', 'B' , 'C', 'D', 'E', 'F','G', 'H', 'I','J', 'K', 'L', 'M',  'N',  'O',  'P',
'Q', 'R' ,  'S' ,  'T',  'U', 'V', 'W', 'X', 'Y',  'Z', '1', '2','3', '4', '5', '6', '7', '8', '9', '0']
  passw = []
  for i in range(count_char):
    passw.append(random.choice(arr))
  return "".join(passw)
      # funksiyani chaqiramiz
print (passw_generator(20))          #ngODHE8J8x ko'rinishada chiqishi mumkin
print (passw_generator())            #ZxcpkF50 ko'rinishada chiqishi mumkin
input ()
```

Nazorat savollari

1. Pythonda sonlar**.**
2. Pythonda sonlar bilan ishlashning tashqi funksiya va metodlari.Pythonda *math* moduli.
3. Pythonda matematik funksiyalar.
4. Pythonda *random* moduli. Tasodifiy son generasiyasi

7-ma’ruza. Satrlar va ular ustida amallar. Satr yaratish. Maxusu belgilar. Satrlar bilan ishlash amallari. Satrlarni formatlash. format() metodi. Satrlar bilan ishlash metod va funksiyalari. Localni sozlash.

Reja

1. Satrlar va ular ustida amallar.
2. Satr yaratish.
3. Maxusu belgilar.
4. Satrlar bilan ishlash amallari.
5. Satrlarni formatlash.
6. format() metodi.
7. Satrlar bilan ishlash metod va funksiyalari.
8. Localni sozlash.

Satr belgilarni ketma-ket berilishidan hosil bo’ladi. Satr uzunligi faqatgina konpyuter tezkor xotirasiga bog’liq cheklanadi.

Ketma-ketliklarga doir indeks bo’yicha murojaat, ulash(+operatori), kesish, takrorlash (* operatori), tegishlilikka tekshirish (in va not in operatori) kabi barcha amallar satrlar uchun ham amal qiladi.

Bundan tashqari, satrlar o’zgarmas ma’lumot tiplarga tegisli. Shuning uchun amaliy jihatdan barcha strlar bilan ishlash metodlari yangi satr sifatida qiymat qaytaradi.

Kichik satrlarni qayta ishlashda hech qanday muammo bo’lmasligi mumkin, biroq katta hajmli satrlar bilan ishlashda xotira yetishmovchiligi bilan bog’liq muammolar kelib chiqadi. Bir so’z bilan aytganda, belgini indeks bo’yicha o’qish mumkin lekin o’zgartirish mumkin emas(6.1-misol).

6.1-misol. Belgini indeks bo’yicha o’zgartirishga urinish

```
>>> s = "Python"
>>> s[0]              # Indeks bo’yicha belgini o’qib olish mumkin
'P'
>>> s[0] = "J"     # Indeks bo’yicha satrni o’zgartirish mumkin emas
Traceback (most recent call last ) :
File "<pyshell#2> ", line 1, in <module>
s[0] = "J"              # Indeks bo’yicha satrni o’zgartirish mumkin emas
TypeError : 'str' ob ject does not support item assignment
```

Python 3 dasturlash tilida quyidagi satr turlari qo’llaniladi:

- str – Unicode - satr. Diqqat qiling aniq ko’dirovkalar: UTF-8, UTF-16 yoki UTF-32 – buyerda ko’rsatilmaydi:

```
>>> type ("Satr" )
```

```
<class 'str' >
>>> "Satr".encode (encoding = "cpl251")
b' \xfl \xf2 \xf0\xee\xea\xe0 '
>>> "satr".encode(encoding="utf- 8")
b' \xdl\x81\xdl\x82\xdl\x80\xd0\xbe\xd0\xba \xd0\xb0'
```

- bytes – o'zgarmas baytlar ketma-ketligi. Ketma-ketliknign har bir elementida belgi kodini ifodalovchi 0 dan 255 gacha butun son saqlanishi mumkin. bytes obyekt turi katta miqdordagi satrlarga tegishli metodlarni go'yoki belgilar ketma-ketligi kabi qo'llashi mumkin. Biroq indeks murojaat belgi emas butun son qaytaradi. Masalan:

```
>>> s =  bytes ("стр str", "cp1251")
>>> s[0],  s[5],  s[0:3],  s[4:7]
(241, 116, b'\xfl\xf2\xf0', b'str')
>>> s
b' \xfl\xf2\xf0  str'


>>>  len ("строка ")
6
>>> len (bytes ("строка" ,  "ср125 1") )
6
>>>  len (bytes ("строка",  "utf-8") )
12
```

- bytearray – o'zgaruvchan baytlar ketma-ketligi. bytearray tipi bytes turi analogidir. Lekin elementni indeks bo'yicha o'zgartirish va qo'shimcha yaratish, o'chirish imkonini beradi. Maslan:

```
 >>> s = bytearray ("str", "cp1251")        # Belgini o'zgartirish mumkin
>>> s[0] = 49; s
bytearray (b'1tr')
>>> s. append(55) ;  s
bytea rray  (b'1tr7')                       #  Belgi qo'shish mumkin
```

Satr yaratish

Satr yaratish quyidagi usullarda amalga oshiriladi:

- str funksiyasi yordamida([<Obyekt> [, <Kodirovka> [, <Xatoliklarni qayta ishlash>]]]). Agar faqat birinchi parameter ko'rsatilsa, satr ko'rinishdagi ixtiyoriy obyektni qaytaradi. Agar parametr umuman ko'rsatilmasa, bo'sh satr qaytariladi. Masalan:

```
>>> str(),  str([l, 2]),  str((3, 4)),  str ({"x":1})
(' ', '[1, 2] ',  '(3, 4)',  "{'x':1}")
>>> str(b"\xfl \xf2\xf0\xee\xea\xe0")
"b'\\xfl\\xf2\\xf0\\xee\\xea\\xe0' "
```

- satrni apostrof yoki qo'shtirnoq orasida keltirish orqali:

```
>>> 'satr',  '"satr" ,  ' "x" :  5' ,  " 'x': 5"
('satr',  '"satr" ,  ' "x" :  5' ,  " 'x': 5" )
>>>  print ('1-satr\n2-satr ' )
1-satr
2-satr
>>> print ("1-satr\n2-satr " )
1-satr
2-satr
```

- satrni uchlangan apostrof yoki uchlangan qo'shtirnoq orasida keltirish orqali. Bunday obyektlar satrni bir nechta satrlarda joylashtirish va ular ichida qo'shtirnoq va apostroflardan foydalanish va chop etish imkonini beradi. Masalan:

```
>>>  print (' ' '1-satr
2-satr ' ' ')
1-satr
2-satr
>>>  print (" " "1-satr
2-satr " " ")
1-satr
2-satr
```

Maxusu belgilar

Maxsus belgilar – bu oddiy yo' bilan qo'yib bo'lmaydigan xizmatchi yoki chop etilmaydigan belgilar kombinasiayasidir:

- \n – satrga o'tkazish;
- \r – katetka qaytarish;
- \t – tabulyatsiya belgisi;
- \v - vertical tabulyatsiya;
- \a – qo'ng'iroq;
- \b - zaboy;
- \f - formatni surish;
- \0 – nolli belgi (satr ohiri hisoblanmaydi);
- \ "- qo'shtirnoq;
- \ ' -apostrof;
- \N –N sakkizlik qimat. Masalan, \74 < belgisiga mos keladi;
- \xN – N o'n oltililiq qiymat. Masalan, \x6a satri j belgisiga mos keladi;
- \ \ - teskari slesh;
- \uxxxx - Unicode -16 bitli belgi. Masalan, \u043a ga kiril xarfiga mos keladi;
- \Uxxxxxxxx – Unicode 32 -bitli belgi.

Satrlar bilan ishlash amallari

Satrlar ketma-ketliklar kabi belgilarig aindeks orqali murojat qilish, qirqib olish,

ulash, qayta takrorlash, tegishlilikka tekshirish kabi amallarni qo'llaydi. Bu amallarni batafsil ko'rib chiqamiz.

Satrning ixtiyoriy belgisiga murojat qilish ucun uning indeksini kvadrat qavs ichida keltirish etarli. Raqamlash noldan boshlanadi:

```
>>> s = "Python"
>>> s[0] , s[1] , s[2] , s[3] , s[4] , s[5]
('P', 'y', 't', 'h', 'o', 'n' )
```

Agar ko'rsatilgan indeksga mos keluvchi satrda bo'lmasa, IndexError istisnosi yuzaga keladi:

```
>>> s = "Python"
>>> s[10]
Traceback (most recent call last):
File "<pyshell#90 >", line 1, in <module>
s[l0]
IndexError: string index out of range
```

Indeks sifatida manfiy sonni ko'rsatish mumkin. Bu holda hisob teskari tartibdaya'ni satr oxiridan boshlanadi, ya'ni -1 elemen satr ohirga belgi bo'ladi. Masalan:

```
>>> s = "Python"
>>> s[-1] , s[len(s)-1]
        ('n', 'n' )
```

Bunday turdagi satrlar ozgarmas turga tegishli bo'lib, indeks bo'yicha belgini o'zgartirish mumkin emas:

```
>>> s = "Python"
>>> s[0] = "J"    # satrni o'zgartirish mumkin emas
Traceback (most recent call last):
File "<pyshell#94>" , line 1, in <module>
s [0] = "J"            # satrni o'zgartirish mumkin emas
TypeError: 'str' object does not support item assignment
```

Agar o'zgartirmoqchi bo'lsak, qatroni biror parchasini qirqib olishmiz mumkin. Amal formati:

[<Boshlanish>:<Tamom>:<Qadam>]

Bu yerda barcha parametrlar majburiy emas. Agar <Boshlanish> parametri ko'rsatilmasa, qiymat 0 deb qabul qilinadi. Agar <Tamom> ko'rsatilmasa, satr ohirigacha olinadi.

Agar <Qadam> ko'rsatilmasa, qiymat 1 deb olinadi. Parameter qiymati sifatida manfiy sonni ko'rsatish mumkin.

Misol orqali ko'rib chiqaylik. Boshlanishiga satrni nusxasini oalamiz:

```
>>> s = "Python"
>>> s[:]  # 0 pozitsiyadan satr ohirigacha satr qismini qaytaradi
'Python'
```

Endi belgilarni teskari tartibda chiqaramiz:

```
>>> s[: : -1] # <Qadam> parametri sifatida manfiy sonni ko’rsatamiz
'nohtyP '
```
Satrdagi birinchi belgini almashtiramiz:
```
>>>"J" + s[l:]        # 1 dan satr oxirigacha belgilarni olamiz va “J” ga ualaymiz
'Jython'
```
Oxirgi belgini o’chiramiz:
```
>>> s[:- 1]           # 0 dan len(s) -1 gacha satr qismini qaytaradi
'Pytho '
```
Satrdagi birinchi belgini olamiz:
```
>>> s[0:l ]           # indeksi 1 gacha satr chop etiladi
'P'
```
Endi ohirgi belgini olamiz:
```
>>> s[-1:]            # len(s)-1 dan satr ohirigacha qismni olamiz
'n'
```
Va nihoyat, 2,3,4 indeksdagi belgilarni chop etamiz:
```
>>> s[2:5]             #  2,  3 va  4 belgilar chop etiladi
'tho'
```
Satrdagi belgilar sonini aniqlashga len () funksiyasi yordam beradi:
```
>>> len("Python"),  len (" \r\n\t " ),  len (r"\r\n\t")
(6, 3, 6)
```
Endi, for sikli yordamida barcha belgilarni saralab chop etamiz:
```
>>> s = "Python"
>>> for i in range (len(s)): print (s [i], end=" ")
```
Natija:
```
P y t h o n
```
Satr iterasiyani qo’llashni nazarda tutib biz shunchaki satrni sikl paramitri sifatida ko’rsatishimiz mumkin:
```
>>> s = "Python"
>>> for i in s:  print (i,  end=" ")
```
Natija quyidagicha:
```
P y t h o n
```
+ operatoridan foydalanib ikki satrni bittaga ulaymiz:
```
>>> print("1-satr"  +  "2-satr")
1-satr2-satr
```
Bundan tashqari, oshkormas holda qotrni ulashimiz mumkin. Bu holda ikki satr ular orasidagi operatorsiz ko’rsatiladi:
```
>>> print ("1-satr" "2-satr")
1-satr2-satr
```
Diqqat qiling, biz agar satrlar orasida vergul keltirilasa, biz natija sifatida satr emas kotrej olamiz:
```
>>> s = "1-satr ",  "2-satr "
```

```
>>> type (s)  # satr emas kotrej olamiz
<class  'tuple '>
```

Agar o’zgaruvchi va satr ulanadigan bo’lsa, albatta ulash belgisini qo’llash kerak, aks holda xatolik kelib chiqadi:

```
>>> s = "1-satr"
>>> print (s  +  "2-satr") #  Xatosiz
1-satr2-satr
>>> print ("s 2-satr") #  Xato
SyntaxError:  invalid syntax
```

Agar satrni boshqa turdagi ma’lumot (amaslan, son) bilan ulash zarur bo’lsa str () funksiyasidan foydalanish va konketanasiya amalini qo’llash kerak bo’ladi:

```
>>> "string" + str(10)
'string10'
```

Ko’rib chiqilgan amallardan tashqari satrlar uchun tegishlilikka tekshirish (in va not in operatorlari), takrorlash (* operatori) amallarini ham qo’llash mumkin:

```
>>> "-" * 20
'--------------------'
>>> "yt" in "Python"
True
>>> "yt" in "Perl "
False
>>> "PHP" not in "Python"
True
```

Satrni formatlash

Satrlarni formatlashda quyidagi shablondan foydlaniladi:

<Maxsus formatdagi satr> % <Qiymat>

<Maxsus formatdagi satr> parametri ichida quyidagicha formatdagi sintaksis ishlatiladi:

%[(<KALIT>)][<bayroq>][<Kenglik>][.<Aniqlik>]<O’zgartirish turi>

Satrdagi tasniflash miqdori <Qiymat> parametridagi elementlar miqdoriga teng bo’lishi mumkin. Agar faqat bitta spesifikatordan foydalanilayotgan bo'lsa, <Qiymat> ham bitta bo’lishi kerak.

Misol:

```
>>> " %s" %  10                          #  Bitta element
'10'
>>> "%s - %s - %s" % (10, 20, 30)    #  Bir nechta element
'10 - 20 - 30'
```

Spesifikator ichoidagi parametrlar quyidagi ma’nolarga ega:

- <KALIT> - lug’at kaliti. Agar kalit berilsa, <Qiymat> kortejni emas lug’atni ko’rsatishi kerak. Masalan:

```
>>> "% (name)s  - % (year)s" % {"year":1978, "name": 'Nik'}
```

'Nik - 1978'

- <Bayroq> - o'zgartirish bayrog'i. Quyidagi qiymatlardan iborat bo'lishi mumkin:
  - # - sakkizlik qiymatlar oldidan 0o belgilar kombinasiyasi, sakkizlik qiymatlar oldidan 0x belgilar kombinasiyasi yoki ox(agar x tipidan foydalanilsa)

<Aniqlik>:

```
>>> print ("%#o %#o %#o" % (0o77, 10, 10.5))
0o77  0o12  0o12
>>>  print ("%#x %#x %#x" % (0xff,  10,  10 . 5) )
0xff 0xa 0xa
>>> print ("% #.0F %.0F" % (300, 300))
 %  (0o77 ,  10,  10 . 5) )
%  (Oxff,  10,  10 . 5) )
%  (Oxff ,  10,  10 .  5) )
(3 00,  300) )
```

0 – oldingi qismni belgilangan uzunlikka yetguncha nollarga to'ldirish:

```
>>> " ' %d '  - '%05d' % (3, 3)" # 5  - maydon kengligi
" '3' – '00003' "
```

- – maydon chap tomonidan tekislanishni belgilaydi. Odatiy holda o'ng tomondan tekislanadi. Agar bayroq bir vaqtda 0 bayroq bilan ko'rsatilsa, 0 bayrog'og'I bekor qilinadi. Masalan:

```
>>> " ' %5d ' – '%- 5d ' " % (3, 3)  #  5 -maydon kengligi
" '     3'  –  '3     ' "
>>>  " ' %05d '  - '% -05d' "  %  (3,  3)
" ' 00003 '  -  ' 3 ' "
```

• probel- musbat sondan oldin probel qo'yiladi. Manfiy sondan oldin minus qo'yiladi. Masalan:

```
>>> " ' % d ' - ' % d '  %  (-3, 3) "
" - 3 '  - 13 ' "
```

• + - agar musbat yoki manfiy ishoralar chop etilishi zarur bo'lsa qo'yiladi. Agar + boyrog'i probel bayrog'i bilan bir vaqtda ko'rsatilsa probel bekor qilinadi. Masalan:

```
>>> " ' % +d ' - ' %+d ' " %  (- 3,  3)
" ' -3 ' - ' +3 ' "
```

<Kenglik> - maydonning minimal kengligi. Agar satr belgilangan kenglikda jaoylashtirilmasa, satr to'liq yoyilib chop etiladi:

```
>>> '"%10d'-'%-10d'"%(3,3)
'"        3'-'3         '"
>>>'"%3s %10s'" %("string", "string")
" 'string     string' "
```

Qiymat o'rniga "*" belgisini ko'rsatish mumkin. Bu holda qiymat kortej ichida beriladi:

```
>>>  " '%*s' '% l0s' "  %  (10, "string",  "str" )
```

" ' string' ' str' "
<Aniqlik> - haqiyqiy sonda nuqtadan keying raqamlar soni. Bu parameter oldida albatta nuqta bo'lishi kerak. Masalan:

```
>>> import math
>>> " %s %f %.2f " % (math.pi, math.pi, math.pi)
'3.141592653589793 3.141593 3.14 '
```

Qiymat o'rniga «* » belgisini ko'rsatish mumkin. Bu holda qiymat kortej ichida beriladi:

```
>>> " ' %*.* f ' " % (8, 5, math .pi)
" ' 3.14159 ' "
```

- <Qayta tiplash> -O'zgartirish turini belgilaydi. Bu parameter majburiy hisoblanadi.

<Qayta tiplash> parametrida quyidagi belgilar ko'rsatilishi mumkin:

- s – hohlagan obyektni str() funksiyasi yordamida satrga o'zgartiradi:

```
>>> print ("%s" % ("Oddiy satr" ))
Oddiy satr
>>> print ("%s %s %s" % (10, 10.52, (1, 2, 3)))
10 10.52 (1, 2, 3)
```

- r – hohlagan obyektni repr() funksiyasi yordamida satrga o'zgartiradi:

```
>>> print (" %r " % ("Oddiy satr"))
'Oddiy satr'
```

- a – obyektni ascii() funksiyasi yordamida satrga o'zgartiradi:

```
>>> print ("%a"% ("satr" ) )
'\u0441\u0442\u04 40\u0 43e\u04 3a\u0430'
```

- c – bitta belgi yoki son qiymatini belgiga o'tkazamiz. Misol sifatida son qiymatini va bu qiymatga mos belgini chop etamiz:

```
>>> for i in range (33, 127): print ("%s => %c" % (i, i))
```

- d va i – sonning butun qismini qaytaradi:

```
>>> print ("%d %d %d" % (10, 25.6, - 80))
10 25 -80
>>> print("%i %i %i" %(10,25.6,-80))
10 25 -80
```

- o – sakkizlik qiymat:

```
>>> print ("%o %o %o" % (0o77, 10, 10.5))
77 12 12
>>> print ("%#0 %#0 %# 0" % (0o77, 10, 10.5))
Oo77 Oo12 Oo12
```

- x – quyi registrdagi o'n oltilik qiymat:

```
>>> print (" %x %x %x" % (0xff, 10, 10.5))
ffaa
```

```
>>> print (" %#x %#x %#x" % (Oxff, 10, 10 .5 ) )
0xff 0xa 0xa
```

• x – yuqori registrdagi o’n oltilik qiymat:

```
>>> print("%X %X %X"%(0xff, 10, 10.5))
FFAA
>>> print ("%#X %#X %#X" % (0xff, 10, 10.5))
OXFF OXA OXA
```

format() metodi

Bundan tashqari formatlash uchun format () operatoridan foydalanish mumkin. U quyidagi formatda beriladi:

<Satr> = <Maxsus formatdagi satr>. format (*args,**kwargs)

<Maxsus formatdagi satr> parametri { va } belgilari ichida ko’rsatiladi va quyidagi sintaksisga ega:

{[<Maydon>] [!<Funksiya>] [: <Format>] }

Qavs ichida joylashgan barcha belgilar probelsiz chiqariladi. Agar satr ichida { va } belgilaridan foydalanishga to’g’ri kelsa, ikki mattadan yozish kerak, aks holda ValueError hatolik kelib chiqadi:

```
>>> print ("{ { va } } - maxsus {0} ". format (" belgilar" ) )
{ va } - maxsus belgilar
```

<Maydon> parametrida pozitsiya indeksini(tartiblash noldan boshlanadi) yoki kalitni ko’rsatish mumkin. Aytaylik nomlanagan parametrlar yoki pozitsiyalar kombinasiyasi. Bu holda fomat() metodi nomlangan parametri o’zining oxirida ko’rsatiladi. Masalan:

```
>>> "{0} - {1} - {2}".format(10, 12.3, "string")
'10-12.3-string'
>>> arr = [10, 12 .3, "string"]          # Indeks
>>> "{0} - {1} - {2}".format (*arr)   # Indeks
'10 - 12.3 - string'
>>> "{model} - {color}".format(color = "red",  model = "BMW") #  kalit
'BMW - red'
>>> d = {"color": "red", "model": "BMW"}
>>> "{model}"
'BMW - red'
>>> "{color}"
'red - 2015 '
{color}".format (**d)                    #  Kalit
{0}".format(2015, color= "red" )       #  Kombinatsiya
```

Satrlar bilan ishlash metod va funksiyalari

Satrlar bilan ishlashda foydalaniladigan asosiy funksiyalarini ko’rib chgiqamiz:

- str([obyekt]) – ixtiyoriy obyektni satrga o'tkazadi. Agar paarmetr ko'rsatilmasa, bosh satrni qaytaradi. Obyektni chop etish uchun print() funksiyasidan foydalaniladi. Masalan:

```
>>> str( ) ,  str([l, 2]),  str((3, 4)),  str ({"x" : 1})
(' ', '[1,  2]',  '(3,  4)',  "{ 'x ' :  1}")
>>>  print ("1-satr\n2-satr")
1-satr
2-satr
```

- repr (<Obyekt>) – obyektni satr ko'rinishda qaytaradi. Python Shell IDLE redaktori oynasidan foydalanish mumkin. Masalan:

```
>>>  repr ("Satr"),  repr ([1, 2, 3]),  repr ({"x":5})
(" 'Satr' ",  ' [1, 2, 3]',  "{'x':5}")
>>>  repr (" 1-satr\n2-satr ")
" '1-satr\\2-satr' "
```

- ascii (<Obyekt>) – obyektni satr formatida qaytaradi. Satr faqat ASCII formatida beriladi. Masalan:

```
>>> ascii([l, 2, 3]), ascii ({"x": 5})
(' [1, 2, 3] ', "{'x': 5}")
>>> ascii ("satr")
'"\ \u04 41\\u04 42 \\u0440\\u043e\\u043a\\u04 30 '"
```

- len (<Satr>) - satrdagi belgilar sonini qaytaradi:

```
>>> len("Python"),  len("\r\n\t") ,  len (r"\r\n\t")
(6,  3,  6)
>>> len ("satr")
6
```

Satrlar bilan ishlashning asosiy metodlarni sanaymiz:

- strip([<belgilar>]) – qatar boshidagi va ohiridagi belgini o'chiradi. Agar parameter ko'rsatilmasa, probel belgilari o'chiriladi: probel, satrga o'tkazish belgisi (\n), karetka qaytarish belgisi (\r), gorizontal (\t) va vertikal (\v) tabulyatsiyalar:

```
>>>  sl,  s2 = "str\n\r\v\t",  "strstrstrokstrstrstr"
>>> " '%s'  - '% s' " % (sl.strip(),  s2.strip ("tsr" ))
" 'str' - 'ok' "
```

- lstrip ([<Belgi>]) – probel yoki satr boshidagi ko'rsatilgan belgilar o'chiriladi:

```
>>> sl,  s2  =  "  str  " ,  "strstrstrokstrstrstr"
>>> " '%s' – '%s' " % (sl.lstrip(), s2.lstrip("tsr" ))
" 'str ' – 'okstrstrstr' "
```

- rstrip([<Belgilar>]) – probel yoki satr ohiridagi ko'rsaatilgan belgini o'chiradi:

```
>>> sl, s2 = "  str          " ,  "strstrstrokstrstrstr"
```

```
>>> " '%s' - '% s' " % (sl.rstrip() ,  s2.rstrip ("tsr"))
" ' str ' - ' strstrstrok ' "
```

- split ([<Bo'luvchi> [, <Limit>]]) – satrni ko'rsatilgan satrlarga bo'ladi. Agar birinchi parametr ko'rsatilmasa yoki None bo'lsa,bo'lish sifatida probel belgisidan foydalaniladi. Masalan:

```
>>>  s  =  "word1 word2 word3"
>>>  s.split (),  s.split(None, 1)
(['word1', 'word2', 'word3'], ['wordl', 'word2 word3'])
>>> s = "wordl\nword2\nword3"
>>> s.split ("\n" )
[' wordl ', 'word2', 'word3 ']
```

Agar satrda birdaniga bir nechta probel bop'lsa va bo'luvchi ko'rsatilmasa, bo'sh element ro'yxatga kiritilmaydi:

```
>>> s = "wordl word2 word3 "
>>> s.split ()
 ['wordl', 'word2', 'word3']
```

Boshqa bo'luvchidan foydalanishda bo'sh elementlar paydo bo'lishi mumkin:

```
>>>  s  =  " ,   ,  wordl,  , word2, , word3, ,"
>>>  s.split (", ")
[' ', '  ', ' wordl', ' ', 'word2', '', 'word3', ',']
>>> "1,, 2, , 3".split (",")
['1', '', ' 2', ' ', ' 3']
```

Agar satrda bo'luvchi topilmasa, oldingi butun satrni o'zi qaytariladi:

```
>>> "wordl word2 word3".split ("\n")
['wordl word2 word3']
```

Localni sozlash

Lokal (tizimning barcha local sozlanmalini sozlash)ni o'rnatish uchun locale modulining set locale () funksiyasidan foydalaniladi.

Funksiyadan foydalanishdan oldin modulni quyidagicha chaqirib olish kerak bo'ladi:

irnport locale

set locale ( ) funksiyasi quyidagi formatga ega:

setlocale (<kategoriya> [, <Lokal>));

<kategoriya> parametri quyidagi qiymatlarni qabul qiladi:

- locale.LC_ ALL -barch arejim uchun lokalni o'rnatish;
- locale.LC_ COLLATE – qatirni taqqoslash uchun;
- locale.LC_ CTYPE – belgiini quyi yoki yuqori registrga o'tkazish uchun;
- locale.LC_MONETARY – pul birligini aks ettirish uchun;
- locale.LC_NUMERIC – sonni formatlash uchun;
- locale.LC_ TIME – sana va vaqtni formatlash uchun.

Lokalning joriy qiymatini olish uchun getlocale ([<категория>)) funksiyasi qo’llaniladi. Misol sifatida Windows tizimida dastlab Windows-1251 kodirovkasi keyin utf-8 va ihtiyoriy hol sozlashni ko’rib chiqamiz. So’ng joriy sozlanma holatini barcha kategoriya va locale.LC_COLLATE uchun aniqlaymiz(6.3-misol).

6.3-misol. lokalni sozlash

```
>>> import locale
>>> # windows -1251 kodirovkasi uchun
>>> locale.setlocale(locale.LC_ALL,  "Russian_Russia .1251" )
'Russian_Russia.1251'
>>> # Lokalni oddiy hol uchun o’rnatamiz
>>> locale.setlocale(locale.LC_ ALL ,  "")
' Russian__Russia.1251 '
>>> # Barcha kategoriya uchun joriy qiymatni olamiz
>>> locale.getloca le ()
(' Russian_Russia' ,  '1251')
>>>  # locale.LC  COLLATE kategoriyasi uchun joriy qiymatni olamiz
>>> locale.getlocale(locale.LC_COLLATE )
(' Russian_Russia', '1251')
```

Lokal sozlanmasini olish uchun localeconv () funksiyasi imkon beradi. Funksiya sozlanma lug’atini qaytaradi.

```
>>>  locale . localeconv ()
{'int_curr_symbol': 'RUB', 'currency_symbol': '?', 'mon_decimal_point': ',', 'mon_thousands_sep': '\xa0', 'mon_grouping': [3, 0], 'positive_sign': '', 'negative_sign': '-', 'int_frac_digits': 2, 'frac_digits': 2, 'p_cs_precedes': 0, 'p_sep_by_space': 1, 'n_cs_precedes': 0, 'n_sep_by_space': 1, 'p_sign_posn': 1, 'n_sign_posn': 1, 'decimal_point': ',', 'thousands_sep': '\xa0', 'grouping': [3, 0]}
```

Nazorat savollari

1. Pythonda satrlar va ular ustida amallar.
2. Pythonda satr yaratish.
3. Pythonda maxusu belgilar.
4. Pythonda satrlar bilan ishlash amallari.
5. Pythonda satrlarni formatlash.
6. Pythonda format() metodi.

8-ma’ruza. Ro’yxatlar, Kortejlar, To’plamlar va Diapazonlar. Ro’yxat yaratish. Ro’yxatlar ustida amallar.

Reja

9. Ro’yxatlar, kortejlar, to’palamlar va diapazonlar.
10. Ro’yxat yaratish
11. Ro’yxatlar ustida amallar

Ro’yxatlar, kortejlar, to’palamlar va diapazonlar

Ro’yxatlar, kortejlar, to’palamlar va diapazonlar – bu obyektlarning tartiblangan to’plamidir. To’plamning har bir elementi faqatgini ixtiyoriy turdagi obyektga havola saqlay olishi sabab o’zida cheklanmagan darajadagi imkoniyat taqdim qiladi.

To’plamdagi element pozitsiayasi indeks orqali aniqlanadi. Elementlarni tartiblash 0 dan boshlanadi.

Ro’yxatlar va kortejlar shunchaki elementlarning tartiblangan ketma-ketligidir. Barcha ketma-ketliklar singari ular elementga indeks bo’yicha murojaat qilish, qirqim olish, konkatenasiya (+ operatori), takrorlash (* operatori), tegishlilikka (in operatori) yoki tegishli emaslikka (not in operatori) tekshirish amallarini qo’llaydi.

- ✓ Ro’yxatlar o’zgaruvchan tiplar toifasiga kiradi. Biz elementni nagfaqat indeks bo’yicha chop qilishimiz, balki uni o’zartira oilishimiz ham mumkinligini anglatadi:

```
>>> arr = [1, 2, 3]  # Ro’yxat yaratamiz
>>> arr [0]          # indeks bo’yicha elementni amiqlaymiz
1
>>> arr [0] = 50  #  indeks bo’yicha elementni o’zgartiramiz
>>> arr
[50, 2, 3]
```

Kortejlar o’zgarmas tiplar toifasiga mansub. Kortej elementlarini indeks bo’yicha aniqlash (olish, chop etish) mukin, biroq o’zgartirish mumkin emas:

```
>>> t = (1, 2, 3)            #  Kortej yaratamiz
>>> t [0]                    #  Indeks bo’yicha elementni olamiz
1
>>>  t[0]  =  50  #  Indeks bo’yicha elementni o’zgartirish mumkin emas
Traceback (most recent  call last ) :
File  "<pyshell#7 >",  line  1,  in  <module>
t[0]  =  50    #  Indeks bo’yicha elementni olamiz
TypeError:  'tuple '  obj ect  does  not  support  item  assignment
```

To’plam o’zgaruvchan kabi bo’lishi ham mumkin, o’zgarmas kabi ham mumkin. Uning asosiy farqi o’zida unikal qiymatlarni saqlashidadir (bir hil qiymatlar avtomatik yo’qotiladi). Maslan:

```
>>> set([0,  1,  1,  2,  3,  3,  4])
{0, 1, 2, 3, 4}
```

✓ Diapazonlar o’zida boshlang’ich va ohirgi qiymatlari hamda qadam kattaligi berilgan sonlar to’plamini aks ettiradi. Ular ning oldingi obyektlar to’plamidan muhim ustunligi tezkor xotiradan kam joy egallaydi. Masalan:

```
>>> r =  range (0, 101, 10)
>>> for i in r: print (i, end = "  ")
0  10  20  30  40  50  60  70  80  90  100
```

Yuqoridagi turlarni batafsil ko’rib chiqamiz.

Ro’yxat yaratish

Ro’yxat yaratish quyidagi usullarda amalga oshirilishi mumkin:

➢ list ([<Ketma-ketlik>]) funksiyasi yordamida. Funksiya ixtiyoriy ketma-ketlikni ro’yxatga o’tkazadi. Agar paraetr ko’rsatilmasa bo’sh ro’yxat yaratiladi. Masalan:

```
>>> list ()  #  Bo’sh ro’yxat yaratamiz
[  ]
>>> list ("String" )  #  Satrni ro’yxtga o’tkazamiz
[ 's' , 't',  'r', 'i',  'n ',  'g' ]
>>> list  ( (1,  2,  3,  4, 5))  #  Kortejni ro’yxatga o’giramiz
[1,  2,  3,  4,  5]
```

➢ barcha royxat elementlarini kvadrat qavs ichida keltirish orqali:

```
>>> arr = [1, "str", 3, "4"]
>>> arr
[1, 'str', 3, '4 ']
```

➢ append () metodi yordamida element bo’yicha to’ldirish:

```
>>> arr =  []  #  Bo’sh ro’yxat yaratamiz
>>> arr.append (l)  # Element yaratamiz  (0- indeks)
>>> arr.append (" str") # 2-element yaratamiz  (1- indeks)
>>>  arr
[1, 'str ']
```

Ro’yxat yaratishda o’zgaruvchida obyekt emas, obyektga havola saqlanadi. Buni guruhli o’zlashtirishda hisobga olish kerak. Gurugli o’zlashtirishdan sonlsr uchun, qatoralar uchun foydalanish mumkin. Ro’yxat uchun esa mumkin emas. Misol ko’rib chiqamiz:

```
>>>  x  =  y  =  [1,  2]                    #  ikkita obyekt yaratildi
>>>  x,  y
( [ 1,  2] ,  [ 1,  2] )
```

Biz bu misolda ikki elementli ro’yxat yaratdik va uni x va y o’zgaruvchilariga o’zlashtirdik. Endi y o’zgaruvchidagi ikkinchi element qiymatini o’zgartiramiz:

```
>>>  y[l]  =  100  #  ikkinchi elementni o’zgartiramiz
>>>  x,  y #  birdnaiga ikkita element qiymati o’zgarmoqda
([1,  100] ,  [1,  100])
```

Ko’rinib turibdiki har ikkala o’zgaruvchi bitta obyektni ko’rsatyapdi, alohida obyektni ko’rsatishi uchun boshqacharoq o’zlashtiramiz:

```
>>> x,  y= [1,  2],  [1,  2]
```

```
>>> y[l] = 100 #  Ikkinchi elenetni o’zgartirmiz
>>> x,  y
([1, 2], [1,  100] )
```

Ikkita o’zgaruvhci bitta obyektga havolanai ko’rsatayotganini aniqlash uchun is operatoridan foydalaniladi. Agar o’zgaruvchilar bitta obyektni ko’rsatsa, is operatori True qiymat qaytaradi:

```
>>> x  =  y  =  [1,  2]  # Bu tavsiya etilmaydi
>>> x  is  y          #  o’zgaruvchilar havolasini tekshirish
True                  # demak bitta obyekni ko’rsatayapdi
>>>  x,  y  = [1, 2], [1, 2]         #  to’g’ri
>>>  x  is  y  #  bular farqli obyekt

False
```

Ro’yxatlar ustida amallar

Ro’yxat elementiga murojaat element indeksini kvadrat qavs orasiada ko’rsatish orqali amalga oshiriladi. Elementlar tartibi noldan boshalanadi. Masalan:

```
>>>  arr  =  [1,  "str",  3.2,  "4 "]
>>>  arr[0] ,  arr[1] ,  arr[2] ,  arr[3]
(1, 'str', 3.2, '4')
```

Pozitsion o’zlashtirishda elementlar qiymatini ihtiyoriy o’zgaruvchilarga o’zlashtirish mumkin. Bunda chap tomondagi o’zgaruvchilar soni o'ng tomondagi elementlar soniga o’zaro mos bo’lishi kerak, aks holda xatolik yuz beradi:

```
>>>  x,  y,  z  =  [1,  2,  3]  #  Pozitsion o’zlashtirish
>>>  x,  y,  z
(1,  2,  3)
>>>  x,  y  =  [1,  2,  3]  # Elementlar soni mos bo’lishi kerak
Traceback (most recent call last):
Fi le  "<pyshell#86>" ,  line  1,  in  <module>
x,  y  =  [1,  2,  3]  # Elementlar soni mos bo’lishi kerak
ValueError:  too many values to unpack (expected  2)
```

Python 3 da pozitsion o’zlashtirishda chap tomondagi bitta o’zgaruvchi oldidan yulduzcha yozish va bu orqali ortiqcha elementlarni ro’yxat sifatida unga o’zlashtirish mumkin. Agar ortiqcha element bo’lmasa, bu o’zgaruvchi o’zida bosh ro’yxatni saqlaydi:

```
>>>  x,  y,  *z  =  [1,  2,  3] ;  x,  y,
(1,  2,  [3]  )
>>> x,  y,  *z  =  [1,  2,  3,  4,  5] ;  x, y, z
(1,  2,  [3,  4,  5])
>>> x, y, *z  =  [1,  2] ;    x,  y,  z
( 1,  2,  [  ] )
>>> *x, y, z = [1,  2] ;        x,  y,  z
( [ ],  1,  2 )
```

```
>>> x, *y, z = [1,  2,  3,  4,  5] ;  x,  y,  z
(1,  [2,  3,  4] ,  5)
>>>  *z  =  [1,  2,  3,  4,  5] ;  z
[1, 2, 3,  4,  5]
```

Ro’yxat elementlarni tartiblash nol dan boshlangani uchun ohirgi element tartibi elementlar sonidan bitta kam bo’ladi. Elementlar sonini aniqlash uchun len() funksiyasidan foydalaniladi:

```
>>> arr = [1,  2,  3,  4,  5]
>>> len (arr)  #  Elementlar sonini aniqlaymiz
5
>>>  arr [len (arr ) -1] #  Ohirgi element tartibini aniqlaymiz
5
```

Agar ko’rsatilgan indeksdagi element mavjud bo’lmasa, IndexError xatoligi yuz beradi:

```
>>>  arr  =  [1,  2,  3,  4,  5]
>>>  arr [5]  # Mavjud bo’lmagan elementga murojaat
Traceback (most recent call last):
File "<pyshell#99>", line 1, in <module>
arr [5]  # Mavjud bo’lmagan elementga murojaat
IndexError:  list  index  out  of  range
```

Indeks sifatida manfiy qiymatdan foydalanih mumkin. Bu holda tartiblanish ro’yxat oxiridan boshlanadi. Agar indeksni musbatlashtirmoqchi bo’lsak, elementlar sonidan manfiy indeksni ayrladi. Masalan:

```
>>>  arr  =  [1,  2,  3,  4,  5]
>>>  arr [- 1] ,  arr [len (arr) -1]  # Oxirgi elementga murojaat
(5,  5)
```

Ro’yxatlar o’zgaruvchi turlaga kirgani uchun indekslar yordamida ularning elementlarini o’zgartirish mumkin:

```
>>>  arr  =  [1,  2,  3,  4,  5]
>>>  arr [0]  =  600 #  indeks bo’yicha elementni o’zgartirish
>>>  arr
[600, 2, 3, 4, 5]
```

Bundan tashqari ro’yxatlarda ko’rsatilgan qismdagi ro’yxat elementlarini nusxalab olish imkoniayati mavjud. Amal formati:

[<Boshlanish> :<Ohiri> :<Qadam>]

Barcha barametrlar majburiy emas. Agar <Boshlanish> ko’rsatilmasa qiymat siaftida 0 olinadi, <Ohiri> ko’rsatilmasa ohirigacha olinadi.

Agar :<Qadam> parametri kiritilmasa 1 qiymatdan foydlaniladi. Parametrlar qiymati sifatida manfiy son olinishi mumkin. Misol ko’rib chiqamiz. Dastlab ro’yxat yuzaki nusxasini olamiz:

```
>>>  arr  =  [1,  2,  3,  4,  5 ]
>>>  m  =  arr[ : ];  m  # Yuzaki nusxa yaratamiz va qiymatni chop etamiz
```

[1, 2, 3, 4, 5]
>>> m is arr # is operatori turli obyekt ekanligini ko'rsatyapdi
False
Endi belgilarni teskari tartibda chop etamiz:
>>> arr = [1, 2, 3, 4, 5]
>>> arr [: : -1] # - 1 qadam
[5, 4, 3, 2, 1]
Birinchi va ohirgi elementsiz chop etamiz:
>>> arr [1:] # birinchi elementsiz
[2, 3, 4, 5]
>>> arr [:- 1] # Ohirgi elementsiz
[1, 2, 3, 4]
Ro'yxatning birinchi ikki elementini olamiz:
>>> arr [0:2] # 2 indeksli belgi chop etilmaydi
[1, 2]
Endi ohirgi elementni chop etamiz
>>> arr[- 1:] # ro'yxatning ohirgi elementi
[5]
Va nihoyat ikkinchidan to'rtinchigacha elemetlarni chop etamiz:
>>> arr[1:4] # 1, 2 va 3 indeksli elementlar olinadi
[2, 3, 4]
Qirqim bo'yicha ro'yxat bo'lagini o'zgartirish mumkin. Agar qirqimda bo'sh ro'yxat o'zlashtirilsa, ushbu bo'lak asosiy ro'yxatdan o'chiriladi:
>>> arr = [1, 2, 3, 4, 5]
>>> arr[l:3] = (6, 7) # 1 va 2 indeksli elementni o'zgartiramiz
>>> arr
[1, 6, 7, 4, 5]
>>> arr[l:3] = [ ] # 1 va 2 indeksli element o'chiriladi
>>> arr
(1, 4, 5]

Ikkita ro'yxatni bittaga birlashtirishda + operatoridan foydalaniladi:
>>> arrl = [1, 2, 3, 4, 5]
>>> arr2 = [6, 7, 8, 9]
>>> arr3 = arrl + arr2
>>> arr3
[1, 2, 3, 4, 5, 6, 7, 8, 9]
+ operatori o'rniga += operatoridan foydalanish mumkin. Bu holda elementlar joriy ro'yxatga qo'shiladi:
>>> arr = [1 , 2, 3, 4, 5]
>>> arr += [6 , 7, 8, 9]

```
>>> arr
[1, 2, 3, 4, 5, 6, 7, 8, 9 ]
```
Ko’rib chiqilgan amallardan tashqari ro’yxatlar uchun takrorlash(* operatori yordamida), tegishlilikka tekshirish (in operatori yordamida) amallarini qo’llash mumkin:
```
>>> [ 1, 2, 3] * 3 # takrorlash amali
[1, 2, 3, 1, 2, 3, 1, 2, 3]
>>> 2 in [1, 2, 3, 4, 5], 6 in [1, 2, 3, 4, 5] #Tegishlilikka tekshirish
(True, False)
```

Ro’yaxatga elementlar qo’shish va o’chirish .Ro’yxat elementlarini qidirish va ro’yxatga kirivchi qiymatlari haqida ma’lumot olish. Ro’yxatni teskarilash va aralshtirish. Tasodifiy elementni tanlash

Reja

1. Ro’yaxatga elementlar qo’shish va o’chirish.
2. Elementni ro'yxatda mavjudligini tekshirish.

Ro'yxatga elementlarni qo'shish va o'chirish.

Ro'yxatga elementlarni qo'shish uchun *append*, *insert* metodlari, elementlarni o'chirishda esa *remove, clear, pop* metodlaridan foydalaniladi, masalan:

```
1  users = ["Tom", "Bob"]
9
2  # ro'yxat oxiriga element qo’shish
3
4  users.append( "Alice")    # ["Tom", "Bob", "Alice"]
5  # ro'yxatdagi ikkinchi o'ringa element qo's'hish
6  users.insert(1, "Bill")    # ["Tom", "Bill", "Bob", "Alice"]
7
8  # element indeksini olish

9  i = users.index( "Tom")
10 # indeks bo'yicha elementni ro'yxatdan o'chirish
11 removed item = users.pop(i)    # ["Bill", "Bob", "Alice"]
12
13 # ro’yxatdagi oxirgi qiymatni olish
14 last_user = users[-1]
15 # ro'yxatdan oxirgi elementni o’chirish
16 users.remove(last user)    # ["Bill", "Bob"]
17
18 print(users)
```

```
19
20 # ro'yxatning barcha elementlarini o'chirish
21 users.clear()
```

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:

```
1   companies = ["Microsoft", "Google", "Oracle", "Apple"]
2   item = "Oracle"  # o'chirish kerak bo'lgan element
3   if item in companies:
4   companies.remove(item)
5 6 print(companies)  # ['Microsoft', 'Google', 'Apple']
```

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:

```
1   nomlar = ["Anvar","Sobir","Sobir","Qosim"]
2   nom soni = nomlar.count("Sobir")
3   print(nom soni)  #2
```

Ro’yxatni saralash. Ro’yxatni sonlar bilan to’ldirish. Ro’yxatni satrga o’tkazish

Reja:

1. Ro’yxatni teskarilash va aralshtirish.
2. Ro'yxatlarni ko'chirish.
3. Ro'yxatlarni birlashtirish.
4. Ichma - ich joylashgan ro'yxatlar.

Ro’yxatni teskarilash va aralshtirish.

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

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

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

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

Shuni alohida takidlash lozimki, tartiblashda ob'ektlar taqqoslanadi. Obekt 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":

```
1  nomlar =  ["anvar","Sobir","sobir","Qosim","tolib"]
2  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.

```
1  nomlar =  ["anvar","Sobir","sobir","Qosim","tolib"]
2  sorted_nomlar = sorted(nomlar,  key = str.lower)
3  print(sorted nomlar)    # ['anvar', 'Qosim', 'Sobir', 'sobir',
4  #  '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.

```
1  sonlar = [12, 45, 23, -35, 2]
2  print(min(sonlar))  #  -35
3  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 tasir qiladi:

```
1  vil1 = ["Toshkent", "Xorazm", ]
2  vil2 = vil1
3  vil2.append('Buxoro'
4  print(vil1)  # ['Toshkent', 'Xorazm', 'Buxoro']
5  print(vil2)  # ['Toshkent', 'Xorazm', 'Buxoro']
```

Bu misolda har ikkala *vil1* va *vil2* o'zgaruvchilar yordamida ayni bir ro'yxatga murojaat bo'lgan. Shuning uchun *vil2* o'zgaruvchisi 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:

```
1  import copy
2  vil1 = ["Toshkent", "Xorazm", ]
3  vil2 = copy.deepcopy(vil1)
4  vil2.append('Buxoro')
5  # ikkita alohida ro'yxat hosil bo'ldi
6  print(vil1)  # ['Toshkent', 'Xorazm']
7  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 sistaksis quyidagi shakllarda qo'llaniladi:

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

list(start:end) - *start, end* parametrlar orqali ro'yxatning *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.

```
1  vil = ["Toshkent", "Xorazm", 'Buxoro', 'Navoi', 'Jizzax']
2  vil1 = vil[:2]
3  print(vil1)    # ['Toshkent', 'Xorazm']
4  vil2 = vil[2:4]
5  print(vil2)    # ['Buxoro', 'Navoi']
6  vil3 = vil[1:5:2]
7  print(vil3)    # ['Xorazm', 'Navoi']
```

Ro'yxatlarni birlashtirish.

Ro yxatlarni birlashtirish uchun (+) amali qo llaniladi:

```
1  vil1 = ["Toshkent", "Xorazm", 'Buxoro']
2  vil2 = ['Navoi', 'Jizzax']
3  vil = vil1 + vil2
4  print(vil)
```

Ichma - ich joylashgan ro'yxatlar.

Ro'yxat elementlari son, satr kabi oddiy turdagi qiymatlargina bo'lib qolmay, balki ro'yxatni ham ifodalashi mumkin. Odatda bunday ro'yxatlardan jadvallar bilan ishlashda ko'p foydalaniladi. Ushbu holatda tashqi ro'yxatning har bir elementi jadvaldagi bitta qatorni ifodalovchi ro'yxatdan iborat bo'ladi:

```
1  ishchilar =  [
2  [ "Tolib" , 33],
3  [ "Akmal", 30],
4  [ "Botir", 27] ]
5
6  print(ishchilar[0])    #          ["Tolib", 33]
7  print(ishchilar[0][0])           # Tolib
8  print(ishchilar[0][1])           # 33
```

Bu erda ichki ro yxatni elementiga murojaat qilish uchun [][] indekslar juftligidan foydalanilgan. Xususan, *ishchilar[0][1]* - birinchi ichki ro'yxatning ikkinchi elementiga murojaat bo lgan.

Ro'yxatga elementlarni qo'shish, o'chirish, o'zgartirish kabi jarayonlar oddiy ro

yxatlardagi kabi amalga oshiriladi:

```
1  ishchilar =   [
2    [ "Tolib" , 33],
3    [ "Akmal", 30],
4    [ "Botir", 27] ]
5
6  print(ishchilar[0])     # ["Tolib", 33]
7  print(ishchilar[0][0])     # Tolib
8  print(ishchilar[0][1])     # 33
9  # ro^yxat yaratish
10 ishchi = list()
11 ishchi.append("Rustam")
12 ishchi.append(21)
13 # Tashqi roʻyxatga yaratilgan roʻyxatni qoʻshish
14 ishchilar.append(ishchi)
15 print(ishchilar[-1])     # ["Rustam", 21]
16 # Tashqi roʻyxatning          oxirgi elementiga element qoʻshish
17 ishchilar[-1].append("+998909737066")
18 print(ishchilar[-1])          #  ['Rustam', 21,  '+998909737066']
19 # tashqi ro^yxatning          oxirgi elementining oxirgi elementini
20 # oʻchirish
21 ishchilar[-1].pop()
22 print(ishchilar[-1])          #  ["Rustam", 21]
23 # tashqi ro^yxatning          oxirgi elementini oʻchirish
24 ishchilar.pop(-1)
25 # birinchi elementni          oʻzgartirish
26 ishchilar[0] = [ "Sobir", 18]
27 print(ishchilar)   #  [['Sam',  18],['Akmal',  30],['Botir', 27]]
```

Murakkab ro'yxatlar elementlariga murojaat ichma - ich joylashgan takrorlash operatorlari orqali amalga oshirilishi mumkin:

```
1  ishchilar =   [
2  [ "Tolib" , 33],
3  [ "Akmal", 30],
4  [ "Botir", 27] ]
5
6  for ishchi in ishchilar:
7  for i in ishchi:
```

```
8  print(i,  end="|")
9  # Konsolga quyidagi ma^lumotlar chiqariladi
10 # Tolib\33\Akmal\30\Botir\27\
```

To’plamlar.

To'plamlar elementlar majmuini ifodalashning yana bir ko'rinishi hisoblanadi. To'plamlarni aniqlash uchun figurali qavs ('{','}') dan foydalanilib, elementlar unda ketma-ket sanaladi:

```
1  talabalar = {"Bobur",  "Zafar",  "Alisher"}
2  print(talabalar)    # {'Bobur',  'Zafar',  'Alisher'}
```

To'plamni tashkil qiluvchi elementlar qiymatlari unikal bo'lishi kerak, agar elementlar qiymatlari ayni bir xil bo'lsa, ya'ni bir xil element takrorlansa, u holda barcha takrorlanuvchi qiymatlar bitta deb hisoblanadi:

```
1  son = {"1",  "1",  "2","2",  "2"}
2  print(son)  # {'2',  '1'}
```

Bu erda to'plam elementlari ikkita "1" va va uchta "2" qiymatlar orqali hosil qilingan. Lekin ekranga to'plam elementlari chop qilinganda to'plam faqatgina ikki elementdan tashkil topganligini ko'rish mumkin.

To'plamni yaratish uchun *set()* funksiyasidan ham foydalanish mumkin. Ushbu funksiyadan foydalanib to'plam yaratilganda parametriga qiymat sifatida ro'yxat yoki kortej ham berilishi mumkin:

```
1  tubSonlar =  [2,3,5,7,11]
2  tubSonlarTuplami = set(tubSonlar)
3  print(tubSonlarTuplami)  # {2r 3r 5r  7r 11}
```

Ayniqsa set() funksiyasi bo'sh to'plam hosil qilish uchun juda qulay hisobladi:

```
1  son = set()
2  print(son)  # set()
```

To'plam uzunligi (to'plam elementlari soni) ni toppish uchun len() funksiyasidan foydalaniladi:

```
1  son =  {3,4,5,6}
2  print(len(son))  # 4
```

To'plamga element qo'shish.

To'plamga element qo'shish uchun add() metodidan foydalaniladi:

```
1  son = set()
```

```
2  son.add(2)
3  son.add(4)
4  print(son)   # {2, 4}
```

To'plamdan elementni o'chirish.

To'plamdan elementni o'chirish uchun *remove()* metodi qo'llanilib, uning argumentiga o'chirilishi kerak bo'lgan element beriladi. Agar o'chirilishi kerak bo'lgan element to'plamda mavjud bo'lmasa, u holda *KeyError* istisno xatoligi ro'y beradi. Shuning uchun to'plamdan elementni o'chirishdan oldin *in* operatori orqali shu elementning lug'atda mavjud yoki yo'qligini tekshirish tavsiya qilinadi:

```
1  ismlar = {"Anvar", "Abbos", "Abror"}
2  ism = " Abror"
3  if ism in ismlar:
4  ismlar.remove(ism)
5  print(ismlar)    # {'Anvar',          'Abbos'}
```

To'plamdan elementni o'chirishning boshqa usuli ham mavjud bo'lib, *discard()* metodi orqali amalga oshiriladi. Usbu usulda element to'plamdan o'chirilganda, agar o'chirilayotgan element to'plamda mavjud bo'lmasa ham istisno xatoligi ro'y bermaydi:

```
1  ismlar = {"Anvar", "Abbos", "Abror"}
2  ism = "Abbos"
3  ismlar.discard(ism)
4  print(ismlar)    # {'Anvar', 'Abror'}
```

To'plamning barcha elementlarini birdaniga o'chirish uchun yani to'plamni tozalash uchun *clear()* metodidan foydalaniladi:

```
1  ismlar = { "Anvar", "Abbos", "Abror"}
2  ismlar.clear()
3  print(ismlar)    # set()
```

To'plam elementlariga *for* operatori orqali murojaatni (perebor) amalga oshirish mumkin:

```
1  ismlar = {"Anvar", "Abbos", "Abror"}
2  for ism in ismlar:
3  print(ism)
```

bu erda to'plamni har bir elementi *ism* o'zgaruvchisiga ketma-ket yuklanadi va keyingi hisoblashlarda ishlatilishi mumkin.

To'plamlar ustuda amallar.

To'plamlar ustuda turli xil amallarni bajarish mumkin bo'lib, ular metod va funksiyalar orqali amalga oshiriladi. Quyida ulardan eng ko'p qo'llaniladiganlarini qarab chiqamiz: *copy()* metodi biror bir to'plamdan nusxa olish uchun ishlatiladi, masalan:

```
1  ismlar = {"Anvar", "Abbos", "Abror"}
2  ismlar2 = ismlar.copy()
3  print(ismlar)   # {'Abbos', 'Anvar', 'Abror'}
4  print(ismlar2)  # {'Abbos', 'Anvar', 'Abror'}
```

union() metodi ikkita to'plamni birlashtiradi va qiymat sifatida yangi to'plamni qaytaradi:

```
1  famil = {"Axmedov", "Niyazov"}
2  ism = {"Sardor", "Tohir"}
3  FIO = famil.union(ism)
4  print(FIO)  # {'Axmedov', 'Tohir', 'Sardor', 'Niyazov'}
```

intersection() metodi ikkita to plamni kesishmasmi olish uchun ishlatib, qiymat sifatida yangi to'plam qaytaradi. Yani ikkita to'plam uchun umumiy bo'lgan elementlarni olish uchun *intersection()* metodi qo'llaniladi:**63**

```
1  famil = {"Axmad", "Sardor", "Ikrom"}
2  ism = { "Sardor", "Tohir", "Ikrom"}
3  ism2 = famil.intersection(ism)
4  print(ism2)  # {'Sardor', 'Ikrom'}
```

intersection() metodi o'rniga unga ekvivalent bo'lgan & (mantiqiy ko'paytirish) amalini ham qo'llash mumkin:

```
1  famil = {"Axmad", "Sardor", "Ikrom"}
2  ism = { "Sardor", "Tohir", "Ikrom"}
3  ism2 = famil & ism
4  print(ism2)  # {'Sardor', 'Ikrom'}
```

Difference() metodi to'plamlar ayirmasini topish uchun qo'llanilib, qiymat sifatida yangi to'plam qaytaradi. Yani birinchi to'plamda mavjud va ikkinchi to'plamda yo'q bo'lgan elementlarni topishda ishlatish mumkin. *difference()* metodiga ekvivalent amal bu amalidir:

```
1  famil = {"Axmad", "Sardor", "Ikrom"}
2  ism = { "Sardor", "Tohir", "Ikrom"}
3  ism2 = famil.difference(ism)
4  ism3 = famil - ism
5  print(ism2)  # {'Axmad'}
6  print(ism3)  # {'Axmad'}
```

issubset() metodi qaralayotgan to'plam boshqa to'plam (argumentida berilgan) ning qism to'plami yoki yo'qligini tekshirish uchun ishlatiladi:

```
famil = {"Axmad", "Sardor", "Ikrom" }
ism = { "Sardor", "Ikrom"}
print(ism.issubset(famil))          # True
print(famil.issubset(ism))          # False
```

issuperset() metodi qaralayotgan to'plam boshqa to'plamni (argumentida berilgan) o'z ichiga olishi yoki olmasligini tekshirish uchun ishlatiladi:

```

famil = {"Axmad", "Sardor", "Ikrom"}
ism = {"Sardor", "Ikrom"}
print(ism.issuperset(famil))    # False
print(famil.issuperset(ism))    # True
```

frozenset. *frozenset* - o'zgartirib bo'lmaydigan to'plamlarni yaratish uchun ishlatiladi. Ushbu turdagi to'plamga yangi element qo'shish, o'chirish yoki element qiymatini o'zgartirishga ruxsat berilmaydi. *frozenset* turidagi to'plam odatda ro'yxat, kortej yoki oddiy to'plam (set) orqali hosil qilinadi:

```
famil = {"Axmad", "Sardor", "Ikrom"}
fam = frozenset(famil)
print(fam)    # frozenset({'Sardor', 'Ikrom', 'Axmad'})
```

frozenset turidagi to'plamlar ustuda quyidagi amallarni bajarish mumkin: len(s) - *s* to'plam uzunligi (elementlari soni)ni qaytaradi; *x in s* - *True* qiymat qaytaradi, agar *x* element s to'plamning tarkibida mavjud bo'lsa;
x not in s - *True* qiymat qaytaradi, agar *x* element s to'plamning tarkibida mavjud bo'lmasa;
s.issubset(t) - *True* qiymat qaytaradi, agar *t* to'plam *s* to'plamni o'z ichiga olsa;
s.issuperset(t) - *True* qiymat qaytaradi, agar s to'plam t to'plamni o'z ichiga olsa;
s.union(t) - *s* va *t* to'plamlarning birlashmasidan tashkil topgan yangi to'plamni qaytaradi;
s.intersection(t) - *s* va *t* to'plamlarning kesishmasidan tashkil topgan yangi to'plamni qaytaradi;
s.difference(t) - *s* to'plamdan *t* to'plamni ayirishdan hosil bo'lgan yangi to'plamni qaytaradi;
s.copy() - *s* to'plamning nusxasini qaytaradi.

Nazorat savollari

1. Pythonda ro’yxatlar
2. Pythonda Ro’yxat yaratish.
3. Pythonda Ro’yxatlar ustida amallar.
4. Pythonda Ko’p o’lchamli ro’yxatlar.
5. Pythonda Ro’yxat elemantlarini saralash.
6. Pythonda Ro’yxat generatorlari
7. Pythonda ro’yxat elementlarini qidirish
8. Pythonda ro’yxatga kirivchi qiymatlari haqida ma’lumot olish.
9. Pythonda ro’yxatni teskarilash va aralshtirish.
10. Pythonda tasodifiy elementni tanlash
11. Pythonda ro’yxatni saralash.
12. Pythonda ro’yxatni sonlar bilan to’ldirish.
13. Pythonda ro’yxatni satrga o’tkazish
14. Pythonda *itertools* moduli qanday maqsadda ishlatiladi.
15. Pythonda noaniq qiymatlar generatsiyasi qanday hosil qilinadi.
16. Pythonda qiymatlar kombinasiyasi generasiyasi qanday hosil qilinadi.
17. Pythonda elementlar izchilligi filtri deganda nimani tishunasiz?

9-ma’ruza. Lug’atlar. Lug’at yaratish. Lug’atlar ustida amallar. Lug’at elementlarini saralash. Lug’atlar bilan ishlash metodlari. Lug’atlar generatori

Reja:

1. Lug'atlar
2. Ro'yxatlar yordamida lug'at xosil qilish.
3. Lug'at elementini o'zgartirish.
4. Lug'atdan elementni o'chirish.
5. Lug'atlarni ko'chirish va birlashtirish.
6. Kompleks (murakkab) lug'atlar.

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:

```
19  users = {1: "Tom", 2: "Bob", 3: "Bill"}
23  elements = {"Au": "Oltin", "Fe": "Temir", "H": "Vodorod",
```

```
4   "O":  "Kislorod"}
```

Bu erda *users* ro'yxatida kalit sifatida son, qiymat sifatida satr qo'llanilgan. *element* ro'yxatida esa qiymat sifatida ham kalit sifatida ham satr ishlatilgan.
Lekin kalitlar va qiymatlar bir turga mansub bo lishi shart emas. Ular har xil turdagi qiymatlar bo lishi mumkin:

```
    objects = {1:  "Tom",  "2": True,  3:  10 0.6}
```

Bundan tashqari bo sh lug atlarni ham yaratish mumkin:

```
1   object1 =
2   # yoki
3   object2 =      dict()
```

Ro'yxatlar yordamida lug'at xosil qilish.

Lug'atlar tuzilmaviy jihatidan ro'yxatlarga o'xshamasada, lekin bazi 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:

```
1   users list =  [["909837022",  "Tolib"],
2   ["909939343",  "Bobur"],
3   ["903943493",  "Alibek"]  ]
4   users dict = dict(users list)
5   print(users dict)   # {'909837022':  'Tolib',  '909939343':
67  'Bobur',  '903943493':  'Alibek'}
```

Xuddi shu tarzda kortejlarni ham lug'atlarga aylantirish mumkin. Buning uchun ikki o'lchamli kortejning ichki kortejlari o'z navbatida ikkitadan elementdan tashkil topgan bo'lishi shart:

```
1   users tuple =  (  ("909837022",  "Tolib"),
2   ( "909939343", "Bobur"),
3   ( "903943493", "Alibek") )
4   users dict = dict(users tuple)
5   print(users dict)   # {'909837022':  'Tolib',  '909939343':
6   #  'Bobur',  '903943493':  '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 quyidagicha amalga oshiriladi:

```
1  users = {
2  "Bir": "Tolib",
3  "Ikki":  "Bobur",
4  "Uch":  "Alisher"   }
5  # Lug'atning "Bir" kalitli elementiga murojaat uchun
6  print(users["Bir"])    # Tolib
7  # Lug'atdagi  "Uch" kalitli element qiymatini o'zgartiramiz
8  users["Uch"]  = "Baxtiyor"
9  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:

```
1  bahoDict = {"A":  5,  "B":  4,  "C":  3}
2  key = "D"
3  if key in bahoDict:
4  baho = bahoDict[key]
5  print(baho)
6  else:
7  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:

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

Shuni alohida ta'kidlash lozimki, agar lug'atda bunday kalit mavjud bolmasa *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 usuli -pop() metodi orqali amalga oshiriladi. U ikki xil 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.

```
1  bahoDict = {"A": 5, "B": 4, "C": 3, "D": 2}
2  key = "A"
3  baho = bahoDict.pop(key)
4  print(baho)    # 5
5  # ikkinchi marta yana shu kalit bo'yicha o'chirishga urinamiz
6  baho2 = bahoDict.pop(key, "Bunday baho mavjud emas!")
7  print(baho2)    # Bunday baho mavjud emas!
```

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

```
1  bahoDict = {"A": 5, "B": 4, "C": 3, "D": 2}
2  # Lug'atning barcha elementlarini ekranga chiqaramiz
3  print(bahoDict)    # {'A': 5, 'B': 4, 'C: 3, 'D': 2}
4  bahoDict.clear()
5  # clear metodini qo'llagandan so'ng yana
6  # lug'atning barcha elementlarini ekranga chiqaramiz
7  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:

```
1  l = {"ismi": "Sardor", "yoshi": 34}
2  l2 = l.copy()
3  print(l)    # {'ismi': 'Sardor', 'yoshi': 34}
4  print(l2)    # {'ismi': 'Sardor', 'yoshi': 34}
```

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

```
1  l1 = {"ismi": "Sardor", "yoshi": 34}
2  l2 = {"kursi": 1, "yo'nalishi": "IAT"}
3  l1.update(l2)
4  print(l1)    # {'ismi': 'Sardor', 'yoshi': 34, 'kursi': 1,
5  #  'yo^nalishi': 'IAT'}
6  print(l2)    # {'ismi': 'Sardor', 'yoshi': 34}
```

Yuqoridagi holatda *l2* tarkibi o'zgarishsiz qoladi va *ll* 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:

```
1  talabalar = {
2  "+99890123":  "Tolmas",
3  "+99890124":  "Bobur",
4  "+99890125":  "Alisher"  }
5  for tal in talabalar:
6  print(tal,  " - ",  talabalar[tal])
```

Javobga quyidagi natija chiqariladi:

+99890123 - Tolmas

+99890124 - Bobur

+99890125 - Alisher bu erda *for* operatoridagi *t* o'zgaruvchiga 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 usuli *items()* metodini qo'llash orqali amalga oshiriladi. Yuqoridagi dastur kodi *items()* metodi orqali quyidagicha yoziladi va ayni bir xil natijaga erishiladi:

```
1  talabalar = {
2  "+99890123":  "Tolmas",
3  "+99890124":  "Bobur",
4  "+99890125":  "Alisher"  }
5  for nomer,  ism in talabalar.items():
6  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:

```
1  talabalar = {
2  "+99890123":  "Tolmas",
3  "+99890124":  "Bobur",
4  "+99890125":  "Alisher"  }
5  # lug'atning kalitlariga murojaat
6  print("Kalitlar:")
7  for kalit in talabalar.keys():
8  print(kalit,  end='; ')
```

```
9   # lug'atning qiymatlariga murojaat
10  print("\n Qiymatlar:")
11  for qiymat in talabalar.values():
12  print(qiymat,  end='; ')
```

Ushbu dastur ishga tushirilganda quyidagi javob ekranga chiqariladi:

Kalitlar:
+99890123; +99890124; +99890125; Qiymatlar:
Tolmas; Bobur; Alisher;

Kompleks (murakkab) lug'atlar.

Lug'atlar faqatgina int, str, *float, bool* kabi oddiy turlarga oid berilganlardangina emas, balki *list, tuple, set, dict* kabi murakkab tuzulmaviy berilganlardan ham tashkil topishi mumkin:

```
1   loginData =  {
2   "Zafar":
3   {
4   "email":  "zafar@nuu.uz",
5   "tel":   "+99890933",
6   "manzil":   "Univer ko'chasi 4"
7   },
8   "Rustam":
9   {
10  "email":  "rustam@nuu.uz",
11  "tel":   "+998902222",
12  "manzil":   "Dekanat ko'chasi 105"
13  }  }
14  print(loginData)
```

Yuqorida keltirilgan misolda *loginData* lug'ati (tashqi lug'at) o'z navbatida boshqa lug'atlar (ichki lug'atlar) dan tashkil topgan. Buday hollarda ichki lug'atni elementlariga quyidagi tarzda murojaat qilinadi:

```
1   lData = loginData["Zafar"]["tel"]
2   print(lData)
```

lug'atda mavjud bo'lmagan kalit orqali uning elementiga murojaat amalga oshirilganda Python interpretatori *KeyError* turidagi istisno xatoligini yuzaga keltiradi:

```
1   lData = loginData["Zafar"]["telegram"]   #KeyError
```

bu erda "telegram" kalit so'zi mavjud emas. Shuning uchun istisno xatoligi ro'y berdi. Bunday xatoliklarning oldini olish uchun dastlab kalitning lug'atda bor yoki yo'qligini tekshirish tavsiya qilinadi:

```
1  key = "telegram"
2  if key in loginData[ "Zafar"]:
3  print(loginData["Tom"]["telegram"])
4  else:
5  print( "telegram kaliti topilmadi")
```

Umuman olganda, murakkab lug'atlar (ichma - ich joylashgan) ustuda amallar oddiy lug'atlardagi kabi amalga oshiriladi.

Nazorat savollari

1. Pythonda lug’atlar**.**
2. Pythonda lug’at yaratish.
3. Pythonda lug’atlar ustida amallar.
4. Pythonda lug’at elementlarini saralash.
5. Pythonda lug’atlar bilan ishlash metodlari.
6. Pythonda lug’atlar generatori

10-ma’ruza. Sana va vaqt bilan ishlash. Joriy sana va vaqtni chop etish. Sana va vaqt formati.

Reja

1. *datetime* moduli
2. *date* sinfi
3. ***time* sinfi**
4. ***datetime* sinfi**

datetime moduli

Sana va vaqt bilan ishlaydigan quyidagi asosiy funksiyalar *datetime* modulida jamlangan:

• date;
• time;
• datetime.

date sinfi

Sana bilan ishlash uchun *datetime* modulida aniqlangan *date* sinfidan foydalaniladi. *date* sinfi obyektini yaratish uchun uchta parameter: yil, oy va kun parametrlarini qabul qiladigan *date* sinfi konstruktoridan foydalaniladi:

```
1  date(year, month,  day)
```

Masalan, qandaydir sana yaratamiz:

```
1  import datetime
2
3  yesterday = datetime.date(2 017,  5,  2)
4
5  print(yesterday)     # 2017-05-02
```

today() metodidan foydalanish orqali joriy sanani olish mumkin:

```
1  from datetime import date
2
3  today = date.today()
4
5  print(today)     # 2017-05-03
6  print("{}.{}.{}".format(today.day,  today.month,
7  today.year))     # 2.5.2017
```

day, *month* va *year* xususiyatlari orqali mos ravishda kun, oy va sanani olish mumkin.

time sinfi

Vaqt bilan ishlash uchun time sinfidan quydagicha foydalaniladi:

```
1  time([hour]   [, min]   [,  sec]   [, microsec])
```

Konstruktor soat, minut, sekund va mikro sekundlarni ketma-ket ravishda qabul qiladi va bu parametrlar shart bo'lmagan parametrlardir. Agarda birorta parameter konstruktorda berilmasa, u holda kelishuv bo'yicha nol qiymat olinadi.

```
1  from datetime import time
2
3  current_time = time()
4  print(current time)     # 00:00:00
5
6  current time = time(16,   25)
7  print(current time)     # 16:25:00
8
9  current_time = time(16,   25,   45)
10 print(current time)     # 16:25:45
```

datetime sinfi

datetime sinfi bir vaqtning o'zida sana va vaqt bilan ishlash imkoniyatini yaratadi. *datetime* sinfi obyektini yaratish uchun quyidagi konstruktor ishlatiladi:

1	datetime(year, month, day [, hour] [, min] [, sec] [, microsec])

Birinchi uchta parametr yil, oy va kunlar zaruriy parametrlar hisoblanadi, qolgan uchtasi esa shart bo'lmagan, hamda, kelishuv bo'yicha ular nol qiymat oladi.

```
1  from datetime import datetime
2  deadline = datetime(2017,  5,  10)
3
4  print(deadline)    # 2017-05-10 00:00:00
5
6  deadline = datetime(2017, 5, 10, 4, 30)
7  print(deadline)    # 2017-05-10 04:30:00
```

Joriy vaqtni va sanani olish uchun *now()* metodidan foydalaniladi.

```
1  from datetime import datetime
2
3  now = datetime.now()
4  print(now)    # 2017-05-03 11:18:56.239443
5
6  print("{}.{}.{}   {}:{}".format(now.day, now.month, now.year, now.hour,
7  now.minute))    # 3.5.2017   11:21
8
9  print(now.date()) print(now.time())
10
```

```
1  from datetime import datetime
2 3
4  now = datetime.now()
   print(now)    # 2017-05-03 11:18:56.239443
5 6
7 8
9  print("{}.{}.{}   {}:{}".format(now.day, now.month, now.year, now.hour,
10 now.minute))    # 3.5.2017   11:21


   print(now.date()) print(now.time())
```

day, *month*, *year*, *hour*, *minute*, *second* parametrlari orqali sana va vaqtning alohida qiymatlarini olish mumkin. *date()* va *time()* metodlari orqali esa mos ravishda alohida sana va vaqtni olish mumkin.

Satrdan sanaga o'tkazish

datetime sinfida *strptime()* metodi mavjud bo'lib, u satr ko'rinishidagi berilgani vaqtga o'tkazadi. Bu metod ikkita parametr qabul qiladi:

1	strptime(str, format)

Birinchi *str* parametri sana va vaqtning satr ko'rinishi, hamda ikkinchi *format* parametri satrdagi sana va vaqt orasi qanday ajratilganligi formati. Formatni aniqlash uchun quyidagi kodlarni ishlatamiz:

- %d - oy kuni son ko'rinishida;
- %m - oyning tartib raqami;
- %y - yil ikkita raqamdan iborat;
- %Y - yil to'rta raqamdan iborat;
- %H - soat 24 soatlik formatda;
- %M - minut;
- %S - sekund.

Har xil formatlarga misol:

```
1  from datetime import datetime
2
3  deadline = datetime.strptime("22/05/2017", "%d/%m/%Y")
4  print(deadline)    # 2017-05-22 00:00:00
5
6  deadline = datetime.strptime("22/05/2017 12:30", "%d/%m/%Y
7  %H:%M"
8  print(deadline)    # 2017-05-22 12:30:00
Q
9  deadline = datetime.strptime( "05-22-2017 12:30", "%m-%d-%Y
10
11 %H:%M" )
12 print(deadline)    # 2017-05-22 12:30:00
```

Nazorat savollari

1. Pythonda sana va vaqt bilan ishlash ucbun qaysi moduldan foydalaniladi**.**

2. Pythonda joriy sana va vaqtni chop etish qanday amalga oshiriladi.

3. Pythonda sana va vaqt formati qanday bo’ladi.
4. Pythonda «Uxlash» skriptini tushuntiring

Sana ustida bajariladigan asosiy amallar. Sana va vaqtni formatlash. Sana va vaqtlarni qo'shish va ayirish. *timedelta* xususiyatlari.

Reja:

1. Sana ustuda bajariladigan asosiy amallar
2. Sana va vaqtni formatlash.
3. Sana va vaqtlarni qo'shish va ayirish.
4. *timedelta* xususiyatlari.
5. Sanalarni solishtirish

Sana ustida bajariladigan asosiy amallar. Sana va vaqtni formatlash

Ushbu sinflar doirasida sana va vaqt obyektlarini formatlash uchun *strftime(format)* metodi mavjud. Bu metod formatlashni ko'rsatuvchi bitta parametr qabul qiladi.

Formatlashni amalga oshirishimiz uchun quyida aniqlangan formatlash kodlaridan birini ishlatish mumkin:

• %a - hafta kuni uchun abbreviatoriya. Masalan, Wed - Wednesday so'zidan (kelishuv bo'yicha ingiliz tilidagi so'zlar ishlatiladi);
• %A - hafta kun to'liq, masalan, Wednesday;
• %b - oy kuni uchun abbreviatoriya. Masalan, Oct (October so'zining qisqartmasi);
• %B - oy nomi to'liq, masalan, October;
• %d - oy kuni, nol qo'shilgan, masalan, 01;
• %m - oy raqami, nol qo'shilgan, masalan, 05;
• %y - yil ikkita raqamdan iborat;
• %Y - yil to'rta raqamdan iborat;
• %H - soat 24 soatlik formatda, masalan, 13
• %I - soat 12 soatlik formatda, masalan, 01
• %M - Minut;
• %S - sekund.
• %f - mikrosekund;
• %p - AM/PM ko'rsatgich;
• %c - sana va vaqt, joriy mahalliy bo'yicha formatlangan;
• %x - sana, joriy mahalliy bo'yicha formatlangan;
• %X - vaqt, joriy mahalliy bo'yicha formatlangan. Har xil formatlar:

```
1  from datetime import datetime
2  now = datetime.now()
3  print(now.strftime("%Y-%m-%d"))          #  2017-05-03
4  print(now.strftime("%d/%m/%Y"))          #  03/05/2017
5  print(now.strftime("%d/%m/%y"))          #  03/05/17
6  print(now.strftime("%d %B %Y  (%A)"))    #  03 May 2017
7  (Wednesday)
```

```
8  print(now.strftime("%d/%m/%y %I:%M"))      # 03/05/17
9  01:36
```

Oy va kunlarning nomini kiritishda kelishuv bo'yicha ingliz tilidagi nomlar ishlatiladi. Agarda joriy mahalliy formatlarni o'rnatish zarur bo'lsa, u holda *locale* modulidan foydalniladi:

```
1  from datetime import datetime
2  import locale
3
4  locale.setlocale(locale.LC ALL,  "Uz")
5
6  now = datetime.now()
7  print(now.strftime("%d %B %Y  (%A)"))    # 08 avgust 2019
8  (payshanba)
```

Sana va vaqtlarni qo'shish va ayirish

Sana va vaqt bilan ishlashda ma'lum bir vaqt oraliqdagi sanani qo'shish yoki ayirish zarurati tug'uladi. *datetime* modulida bu ishlarni amalga oshirish uchun maxsus *timedelta* sinfi aniqlangan. Ushbu sinf ma'lum bir vaqt oralig'dani aniqlaydi.
Vaqt oralig'ini aniqlash uchun *timedelta* sinfi konstruktori quyidagicha ishlatiladi:

```
1  timedelta([days] [, seconds] [, microseconds] [, milliseconds] [, minutes] [,
   hours] [, weeks])
```

Konstrutorga mos ketma-ketlikda kunlar, sekundlar, mikrosekundlar, milisekundlar, minutlar, soatlar va haftalarni berish mumkin. Bir qancha oraliqlarni aniqlaymiz:

```
1  from datetime import timedelta
2
3  three_hours = timedelta(hours=3)
4  print(three hours)     # 3:00:00
5  three_hours_thirty_minutes = timedelta(hours=3,  minutes=30) # 3:30:00
6
7
8  two days = timedelta(2)     # 2 days,  0:00:00
9
10 two_days_three_hours_thirty_minutes = timedelta(days=2, hours=3,
11 minutes=30)     # 2 days,  3:30:00
```

timedelta obyektini ishlatish orqali qo'shish va ayirish amallarini bajarishimiz mumkin. Masalan, ikki kundan keyingi sanani olamiz:

```
1  from datetime import timedelta,  datetime
2
3  now = datetime.now()
4  print(now)     # 2019-08-08 19:23:41.774384
5  two_days = timedelta(2)
6  in_two_days = now + two_days
7  print(in two days)     # 2019-08-10 19:23:41.774384
```

Yoki 10 soatu 15 minut oldin qanday vaqt bo'lganini aniqlaymiz, buning uchun joriy vaqtdan 10 soatu 15 minutni ayrish kerak:

```
1  from datetime import timedelta,  datetime
2
3  now = datetime.now()
4  till_ten_hours_fifteen_minutes = now - timedelta(hours=10,
5  minutes=15)
6  print(till_ten_hours_fifteen_minutes)
```

timedelta xususiyatlari

timedelta sinfi bir qancha xususiyatlarga ega bo'lib, ular orqali vaqt oraliqlarini olish mumkin:

• days - kunlar miqdorini qaytaradi;
• seconds - sekundlar miqdorini qaytaradi;
• microseconds - mikrosekundlar miqdorini qaaytaradi.

Bundan tashqari umimiy sekundlar miqdorini qaytaruvchi *total_seconds()* metodi ham mavjud, hamda bunga kunlar, sekundlar va mikrosekundlar kiradi. Masalan, ikki sana oralig'idagi vaqtni aniqlaymiz:

```
1  from datetime import timedelta,  datetime
9 2
3  now = datetime.now()
4  twenty_two_may = datetime(2019,   12,   22)
5  period = twenty_two_may - now
6  print("{} kun    {} sekund      {}
```

```
7  mikrosekund".format(period.days,  period.seconds,
8  period.microseconds))
9  # 135 kun    16024 sekund      750740 mikrosekund
10
11 print( "Hammasi:   {} sekund".format(period.total_seconds()))
12 # Hammasi: 11680024.75074 sekund
```

Sanalarni solishtirish

Satrlar va sonlar kabi sanani ham standart taqqoslash amallari yordamida solishtirish mumkin:

```
1  from datetime import datetime
2  now = datetime.now()
3  deadline = datetime(2019,   5,   22)
4  if now > deadline:
5  print("Dasturni topshirish muddati o'tdi")
6  elif now.day == deadline.day and now.month ==
7  deadline.month and now.year == deadline.year:
8  print( "Bugun dasturni topshirish muddati")
9  else:
10 period = deadline - now
11 print( "Qoldi {} kun".format(period.days))
```

Nazorat savollari

1. Pythonda datetime moduli qo’llanilishiin tushintiring.
2. Pythonda sana va vaqt monipulyatsiyasi qanday bo’ladi?
3. Pythonda calendar moduli foydalanish tartibini tushintiring.
4. Pythonda calendarni ko’rsatish uchun qaydi metod ishlatiladi?

11-ma’ruza. Foydalanuvchi funksiyalari

O`quv modullari

Funksiya, anonim funksiyalar, funksiya-generatorlar, dekorator, rekursiya, faktorial, global va local o’zgaruvchilar, ilova funksiyalar, funksiya annotasiyasi, parametrsiz funksiya, paramertli funksiya.

Funksiya - bu dasturning istalgan joyidan chaqirib ishlatish mumkin bo’lgan, biror muayyan vazifani bajaradigan qism dasturidir. Funksiyalar dasturni ixchamlashtiradi va ishlash tezligini oshirishga hizmat qiladi. Oldingi boblarda biz Python tilining ichki yaratilgan funktsiyalaridan bir necha bor foydalanganmiz - masalan, len() funktsiyasidan foydalanib, biz ketma-ketlik elementlari sonini aniqladik.

Ushbu bobda dasturning maxsus funksiyalarni yaratishni ko’rib chiqamiz.

*12.1. **Funksiyai aniqlanishi va uni chaqirish***

Funktsiya **def** kalit so’zidan foydalanib quyidagi ko’rsatma bo’yicha yaratiladi (yoki dasturchilar tili bilan aytganidek, aniqlanadi):

def <Funktsiya nomi> ([<Parametrlar>]):

[“” “Hujjat qatori” “”]

<Funktsiya tanasi>

[return <Natija>]

Funksiya nomi lotin harflari, raqamlar va pastki chiziqlardan tashkil topgan, raqam bilan boshlanmaydigan, takrorlanmas identifikator bo’lishi kerak. Kalit so’zlar va o’rnatilgan identifikatorlarning nomlari funksiya nomi sifatida ishlatilmasligi kerak. Funksiya nomidagi belgilarning registri ham muhimdir.

Funktsiya nomidan keyin qavslar ichida vergul bilan ajratilgan bir yoki bir nechta parametrlarni belgilash mumkin, agar funksiya parametrlarni qabul qilmasa, faqat qavslar ko'rsatiladi. Qavslardan keyin ikki nuqta qo‘yiladi.

Funktsiya tanasida asosiy amallar ko’rsatiladi. Har qanday birikma dastur konstruktsiyasida bo'lgani kabi, funktsiya ichidagi qatorlarda ham chap tomonda bir xil sonli bo'shliqlar bilan ajralib turadi. Funktsiyaning oxiri oldinroq kamroq bo'shliqlar qo'yilgan gap hisoblanadi. Agar funktsiya tanasida ko'rsatmalar mavjud bo'lmasa, unda uning ichiga biron bir amalni bajarmaydigan pass bayonoti joylashtirilishi kerak.

Ushbu operator dasturni nosozliklarini tuzatish bosqichida, funktsiyani aniqlab,

tanani keyinroq qo'shishga qaror qilganimizda foydalanish uchun qulaydir. Hech narsa bajarmaydigan funksiyaga misol:

```
def func() :
pass
```

Majburiy bo’lmagan return operatori funktsiyadan qiymatni funksiya tashqarisiga qaytarishga imkon beradi. Ushbu ko'rsatmani bajargandan so'ng, funktsiyani bajarish to'xtatiladi. Bu shuni anglatadiki, return operatoridan keyingi operatorilar hech qachon bajarilmaydi. Misol:

```
def  func() :
print ("return operatorigacha bo’lgan matn" )
return "Qaytariluvchi qiymat"
print ("Bu operatorlar hech qachon bajarilmaydi" )
print (func ())  #  Funksiyani chaqiramiz
```

Dastru natiajasi quyidagicha:

```
return operatorigacha bo’lgan matn
Qaytariluvchi qiymat
```

return operatori funksiya ichida umuman bo'lmasligi mumkin. Bunday holda, funksiya ichidagi barcha ko'rsatmalar bajariladi va natijada None qaytariladi.

Masalan, uchta funksiyani yarataylik (11.1-misol)

12.1-misol. Funksiya aniqlanishi

```
def print_ok () :
"""Parametrsiz funksiyaga misol"""
print ("Amalni muvaffaqiyatli bajarilishi haqida xabar")
def echo (m) :
"""Paramertli funksiyaga misol"""
print (m)
def summa (x,  y) :
""" Ikki son yig’indisini hisoblovchi
parametrli funklsiayaga doir misol """
return x  +  y
```

Funksiyani chaqirishda qiymatlar vergul bilan ajratilgan holda qavs ichida beriladi. Agar funksiya parametrlarni qabul qilmasa, faqat qavslar ko'rsatiladi. Shuni ham ta'kidlash

kerakki, funksiya ta'rifidagi parametrlarning soni chaqirilganda parametrlar soniga mos kelishi kerak, aks holda xato xabari ko'rsatiladi. 11.1-misolda ko'rsatilgan funktsiyalardan 11.2-misoldagi ko'rinishda chaqirib foydalanish mumkin.

12.2-misol. Funksiyani chaqirish

```
print_ok ()          # Parametrsiz funksiyani chaqirish
echo ("Habar")             # Funksiya habar matnini chop etadi
x = summa (5,  2)          #  x o'zgaruvchisi 7 qiymatini o'zlashtiradi
a, b = 10, 50
y= summa(a,  b)    #  y o'zgaruvchisi 60 qiymatini o'zlashtiradi
```

Oxirgi misoldan ko'rinib turibdiki, funksiya chaqiruvidagi o'zgaruvchi nomi funksiya ta'rifidagi o'zgaruvchiga mos kelmasligi mumkin. Bundan tashqari, global o'zgaruvchilar x va y funksiya ta'rifida bir xil nomdagi o'zgaruvchilarga zid kelmaydi, chunki ular har xil ko'rinish sohasida joylashgan. Funktsiya ta'rifida ko'rsatilgan o'zgaruvchilar lokal va faqat funktsiya ichida amal qilinadi. Biz keyiongi bo'limda buni batafsilroq ko'rib chiqamiz.

summa() funktsiyasida ishlatiladigan + operatori nafaqat sonlarni qo'shish uchun, balki ketma-ketlikni birlashtirish uchun ham ishlatiladi. Shunday bo'lsada, summa() funksiyasidan faqat sonlarni qo'shishda ko'proq foydalanish mumkin. Misol tariqasida satrlarni ulash va ro'yxat birlashtirishni ko'rib chiqamiz:

```
def  summ a (x,  y ) :
return  x  +  y
print (summa("ona",  "jon") )          # Natija:  onajon
print (summa ([1, 2],  [3, 4]))        #  Natija:   [1,  2,  3,  4  ]
```

Siz allaqachon bilganingizdek, Pythonda hamma narsa: satrlar, ro'yxatlar va hatto ma'lumotlar turlari ham ob'ektlar sifatida tasvirlanadi. Funktsiyalar ham bundan istisno emas. Def bayonoti function turidagi ob'ektni yaratadi va unga havolani def bayonotidan keyin ko'rsatilgan identifikatorda saqlaydi. Shunday qilib, biz funktsiyaga havolani boshqa o'zgaruvchida saqlashimiz mumkin - buning uchun funktsiya nomi qavssiz ko'rsatiladi. Ma'lumotni o'zgaruvchiga saqlaylik va u orqali funktsiyani chaqiramiz (11.3-misol).

12.3-misol. Funktsiya ma'lumotlarini o'zgaruvchida saqlash

```
def  summa (x,  y)  :
```

```
return  x  +  y
f  =  summa          #  f o’zgaruvchida havolani saqlaymiz
v  =  f (10,  20)     #  f o’zgaruvchi orqali funksiyani chaqiramiz
```

Bundan tashqari, funksiya nomini boshqa funksiyada parametr sifatida parametrlar ro’yxatida berish ham mumkin. Havola orqali uzatiladigan funktsiyalar odatda qayta chaqiriluvchi funktsiyalari deb ataladi (11.4-misol).

12.4-misol. Qayta chaqiriluvchi funksiya

```
def  summ a (x,  y) :
return  x  +  y
def  func (f ,  a,  b) :
"""f o’zgaruvchisi orqali
summa( )  funksiyasiga havola """
return f(a,  b)                  #  summa() funksiyasin ichaqiramiz
#  funksiyaga parameter sifatiada havola beramiz
v  =  func(summa,  10,  20 )
```

12.2. ***Funktsiya ta'riflarining joylashishi***

Dasturdagi barcha ko'rsatmalar yuqoridan pastgacha ketma-ket bajariladi. Bu shuni anglatadiki, dasturda identifikatorni ishlatishdan oldin, avval unga qiymat berish orqali aniqlanishi kerak. Shuning uchun funktsiya ta'rifi funktsiya chaqirig'idan oldin joylashtirilishi kerak.

To'g'ri:

```
def  s uппn a( x,  y ):
return  x  +  y
v = summa(10,  20 )  #  Aniqlanishdan keyin chaqirilyapdi.  Hammasi joyida
```

Noto’g;ri:

```
v  =  summa(10,  20)  #  Identifikator hali aniqlanmagan.  Bu xato!!!
def  summa (x,  y )  :
return  x  +  y
```

Muhokama uchun savollar va topshiriqlar

1. Funksiya yaratishdan ko’zlangan asosiy maqsad nima?
2. Pythonda funksiyaning aniqlanishi qanday amalga oshiriladi?
3. Funktsiya ma'lumotlarini o'zgaruvchida saqlash qanday amalga oshiriladi?
4. Pythonda tiplarni o’zgartiruvchi standart funksiyalarga misollar keltiring?
5. Funksiyaga qiymat uzatish qanday amalga oshiriladi?
6. Funksiyaga murojaat qilishda qanday xatoliklar yuzaga kelishi mumkin?
7. Foydalanuvchi ismi va yoshini so'rab, uning tug'ilgan yilini hisoblaydigan funksiya yozing?
8. Foydalanuvchidan son olib, son juft yoki toqligini konsolga chiqaruvchi funksiya yozing?
9. Foydalanuvchidan son qabul qilib, sonni 2 dan 10 gacha bo'lgan sonlarga qoldiqsiz bo'linishini tekshiruvchi funksiya yozing. Natijalarni konsolga chiqaring?
10. Foydalanuvchidan son olib, uning kvadrati va kubini konsolga chiqaruvchi funksiya yozing?

12-ma’ruza. Fayl va kataloglar bilan ishlash

O`quv modullari

Faylni ochish, buffering, Os moduli, Fayl tarkibini o’qish, StringIO va BytesIO sinfi, fayl va kataloglarga kirish huquqi, faylarni monipulyasiyalash funksiyasi, fayl yoki kataloglarga yo’lni o’zgartirish, kiritish/chiqarishni qayta yo'naltirish

Biz tez-tez ma'lumotlarni xotirada saqlashimizga to’g’ri keladi. Buning ikkita usuli bor: faylga yozish va ma'lumotlar bazasiga saqlash. Birinchi usul oz miqdordagi ma'lumotni saqlashda qo'llaniladi. Agar ma’lumot hajmi katta bo'lsa, ma'lumotlar bazasidan foydalanish yaxshiroq va qulayroq.

10.1. 13.1. Fayllarni ochish.

Fayl ustida ishlashdan oldin open() funktsiyasidan foydalangan holda fayl ob'ekti yaratishingiz kerak. Funktsiya quyidagi formatga ega:

open (<Faylga yo’l> [, mode= ' r '] [, buffering=- 1] [, encoding=None] [, errors=None] [, newline=None] [, closefd=True])

Birinchi parametr faylga yo'lni belgilaydi. Yo'l absolyut yoki nisbiy bo'lishi mumkin. Windows OTda absolyut yo'lni belgilashda, Pythonda chiziq(slesh)ning maxsus belgi ekanligini unutmang. Shu sababli, yotiq chiziq(slesh) ikki baravarga ko'paytirilishi yoki oddiy satrlar o'rniga formatlanmagan satrlardan foydalanish kerak. Misol:

```
>>> "C:\\temp\\new\\file.txt"          #To’g’ri
'C:\\temp\\new\\file.txt'
>>> r"C:\temp\new\file.txt"            #To’g’ri
'C:\\temp\\new\\file.txt'
>>> "C : \temp\new\file.txt"           #Noto’g’ri
'C: \temp\new\xOcile.txt'
```

Oxirgi misolga e’tibor qarating. Shu tarzda, chiziqlar ikki baravar ko'paymaganligi sababli, uchta maxsus belgining mavjudligi darhol paydo bo'ldi: \ t, \ n va \ f (\ xOc sifatida ko'rsatiladi). Ushbu maxsus belgilarni o'zgartirgandan so'ng, yo'l quyidagicha ko'rinadi:

C : <Tab>emp<Satrga o’tkazish>ew<Formatga o’tkazish>ile.txt

Bunday satrni open () funktsiyasiga o'tkazishda OSError istisnoga olib keladi:

>>> open ("C : \temp\new\file . txt")

Traceback (most recent call last) :

File "<pyshell#O> " , line 1 , in <module>

open ("C : \temp\new\file . txt ")

OSError: [Errno 22] Invalid argument : ' C :\temp\new\xOcile . txt '

Absolyut fayl yo'li o'rniga nisbiy yo'lni belgilashingiz mumkin. Bunday holda, yo'l joriy ishlaydigan katalog joylashgan joyga qarab belgilanadi. Nisbiy yo'l os.path modulidagi abspath () funktsiyasi yordamida avtomatik ravishda absolyut yo'lga aylantiriladi. Natijalar quyidagicha bo’lishi mumkin:

- agar ochilayotgan fayl joriy ishchi katalogda bo'lsa, unda faqat fayl nomi ko'rsatilishi mumkin. Misol:

  >>> import os.path # Modulni bog’laymiz

  >>> # Joriy ish katalogidagi fayl (C : \book\ )

  >>> os.path . abspath (r"file.txt")

  'C: \\book\\file.txt'

- agar ochiladigan fayl ichki papkada joylashgan bo'lsa, fayl nomidan oldin chiziq bilan ajratilgan pastki papkalarning nomlari berilgan. Misollar:

  >>> # C:\book\folderl\ da ochiladigan fayl

  >>> os.path.abspath(r" folderl/file.txt")

  'C:\\ book\\ folderl\\file.txt'

  >>> # C:\book\folderl\folder2\ da ochiladigan fayl

  >>> os.path.abspath(r" folderl/folder2/file.txt")

  'C:\\book\\folderl\\folder2\\file.txt '

- agar faylga ega papka darajadan pastroq bo'lsa, unda fayl nomidan oldin ikkita nuqta va chiziq (slesh) ko'rsatiladi. (" .. /"). Misol:

  >>> # C:\ da ochiladigan fayl

  >>> os.path.abspath (r" .. /file.txt ")

  'C: \\file. txt

- agar yo'lning boshida yotiq chiziq (slesh) bo'lsa, u holda yo'l diskning ildizidan boshlab hisoblanadi. Bunday holda, joriy ishlaydigan katalogning joylashuvi muhim emas. Misollar:

```
>>> # C:\book\ folderl\ da ochiladigan fayl
>>> os.path.abspath ( r" /book/folderl/file.txt")
'C:\\ book\\folderl\\file.txt '
>>> # C:\book\folderl\folder2\ da ochiladigan fayl
>>> os.path.abspath ( r" /book/folderl/folder2 /file.txt" )
'C: \\ book\\ folderl\\folder2\\file.txt '
```

Ko'rib turganimizdek, oldinga va orqaga yotiq chiziqlar (sleshlar) absolyut va nisbiy yo'llarda belgilanishi mumkin. Ularning barchasi os.path modulidagi sep atributi qiymati asosida avtomatik ravishda o'zgartiriladi. Ushbu atributning qiymati ishlatilayotgan operatsion tizimga bog'liq. Windows operatsion tizimida sep atributining qiymatini ko'rsatib o’tamiz:

```
>>> os.path.sep
'\\'
>>> os.path.abspath( r"C: /book/folderl /file.txt ")
'C: \\book\\folderl\\file.txt '
```

Nisbiy yo'ldan foydalanganda, amaldagi ishchi katalog joylashgan joydan xabardor bo'ling, chunki ishchi katalog har doim ham bajariladigan dastur joylashgan katalog bilan bir xil bo’lmaydi. Agar fayl uning belgisini ikki marta bosish orqali ishga tushirilsa, u holda kataloglar mos keladi. Agar fayl buyruq satridan ishga tushirilsa, u holda ishchi katalog fayl ishga tushiriladigan katalog bo'ladi.

Keling, bularning barchasini misol bilan ko'rib chiqamiz, buning uchun C: \ book katalogida quyidagi fayl tuzilishini yaratamiz:

```
C: \book\
    test .py
    folderl\
        __init__ . py
        Module1.py
```

C:\book\test.py faylining tarkibi quyidagi ro'yxatda keltiriladi:

```
# -*- coding : utf-8 -*
import os , sys
print ( "%-25s%s" % ( "Fayl : ", os . path . abspath( file )))
```

```
print ( " %-25s%s" % ( "Joriy ishchi katalog : ", os . getcw d() ))
print ( " %-25s%s" % ( " Katalogni import qilish : ", sys . path [O ] ))
print ( " %-25s%s" % ( "Faylga yo’l: "; os . path . abspath ( "file.txt " )))
print ( "-" * 40)
import folderl . module1 as m
m. get_cwd ()
```

C: \ book \ folder1 \ __ init__ .py fayli bo'sh holatda yaratiladi. Siz avval bilganingizdek, ushbu fayl Python tarjimoniga ushbu katalog modullar to'plami ekanligini aytadi.

C:\book\folder1\ module1 .py faylining tarkibi quyidagi ro'yxatda keltiriladi:

```
# -*- coding : utf-8
import os , sys
def get_cwd ( ) :
    print ( "%-25s%s" % ( "Fayl : " , os.path.abspath (__file__)))
    print ( "%-25s%s" % (" Joriy ishchi katalog:" , os.getcwd ()))
    print ( "%-25s%s" % ("Katalogni import qilish :" , sys.path [O] ))
    print ( "%-25s%s " % (" Faylga yo’l:", os.path.abspath ("file.txt ")))
```

Buyruqning satrini ishga tushiramiz va undan C:\book katalogiga o'tamiz va test.py faylini ishga tushiramiz:

```
C:\>cd C:\book
C:\book>test.py
Fayl:                   C:\book\test.py
Joriy ishchi katalog:   C:\book
Katalogni import qilish: C:\book
Faylga yo’l:            C:\book\file.txt
---------------------------------------
Fayl:                   C:\book\folder1\module1.py
Joriy ishchi katalog:   C:\book
Katalogni import qilish: C:\book
Faylga yo’l:            C:\book\file.txt
```

Ushbu misolda joriy ishchi katalog test.py fayli joylashgan katalog bilan bir xil. Biroq, module1.py moduli ichidagi joriy ishchi katalogga e'tibor qarating. Agar siz ushbu modul ichida open () funktsiyasida fayl nomini yo'lsiz ko'rsatadigan bo'lsangiz, u holda fayl

C:\book\folder1 katalogidan emas, balki C:\book katalogidan qidiriladi.

Endi C: diskining ildiziga o'tamiz va yana test.py faylini ishga tushiramiz:

```
C:\book>cd C:\
C:\>C:\book\test.py
Fayl:                    C:\book\test.py
Joriy ishchi katalog:    C:\
Katalogni import qilish: C:\book
Faylga yo’l:             C:\file.txt
---------------------------------------
Fayl:                    C:\book\folder1\module1.py
Joriy ishchi katalog:    C:\
Katalogni import qilish: C:\book
Faylga yo’l:             C:\file.txt
```

Bunday holda, joriy ishchi katalog test.py fayli joylashgan katalog bilan bir xil emas. Agar open () funktsiyasida test.py va module1.py fayllari ichida siz fayl nomini yo'lsiz ko'rsatsangiz, u holda fayl ushbu fayllar katalogidan emas, balki C: diskining ildizidan qidiriladi. Ishga tushiriladigan fayl bilan katalogdagi faylni har doim qidirish uchun siz os modulidagi chdir () funktsiyasidan foydalanib ushbu katalogni joriy qilishingiz kerak. Masalan, test.py fayli tarkibini o'zgartiraylik:

```
# - * - coding : utf-8 -*
import os, sys
# Ishga tushiriladigan fayl bilan katalogni joriy qilish
os.chdir (os.path.dirnarne(os.path.abspath (file)))
print ( "% -25s%s" % ( "Fayl:", file))
print ( " %-25s%s" % ( "Joriy ishchi katalog:", os.getcwd() ))
print ( " %-25s%s" % ( "Katalogni import qilish: ", sys.path [ O ]))
print ( " %-25s%s" % ( "Faylga yo’l: ", os.path. abspath ( "file.txt " )))
```

To'rtinchi qatorga e'tibor bering. _File_ atributidan foydalanib, fayl nomi bilan birga bajariladigan faylga yo'l olamiz. _File_ atributi har doim ham faylga to'liq yo'lni o'z ichiga olmaydi. Masalan, agar faylni ishga tushirish quyidagi tarzda amalga oshirilsa: C:\book>C : \Python37\python test.py, unda atributda yo'lsiz faqat fayl nomi bo'ladi.

Har doim faylga to'liq yo'l olish uchun atribut qiymatini os.path modulidan abspath ()

funktsiyasiga o'tkazishimiz kerak bo’ladi. Keyinchalik, dirname() funktsiyasi yordamida yo'lni (fayl nomisiz) chiqaramiz va uni chdir() funktsiyasiga o'tkazamiz. Endi open () funktsiyasida yo'lsiz fayl nomini ko'rsatsangiz, qidiruv ushbu fayl bilan katalogda amalga oshiriladi. Buyruqlar satri yordamida test.py faylini ishga tushiramiz:

C : \>C : \book\test. py

Fayl:	C:\book\test.py
Joriy ishchi katalog	C:\book
Katalogni import qilish:	C:\book
Faylga yo’l:	C:\book\file.txt

Kataloglar bilan ishlash funktsiyalarini keyingi bo'limlarda batafsil muhokama qilib chiqamiz. Hozircha esda tutish kerakki, bizni misollarda joriy katalog bajarilayotgan fayl joylashgan katalog emas, balki fayl ishga tushirilgan katalog bo'ladi. Bundan tashqari, fayllarni qidirish yo'llari va modul qidirish yo'llari bir biri bilan hech qanday bog’liq emas. Faylni ochishda Open() funktsiyasidagi ixtiyoriy rejim parametri quyidagi qiymatlarni qabul qilishi mumkin:

- r – faqat o’qish uchun ochish(standart xolda foydalaniladi). Faylni ochgandan so'ng, ko'rsatkich faylning boshiga o'rnatiladi. Agar fayl mavjud bo'lmasa, FileNotFoundError xatolik qaytaradi.
- r+ – o’qish va yozish uchun ochish. Faylni ochgandan so'ng, ko'rsatkich faylning boshiga o'rnatiladi. Agar fayl mavjud bo'lmasa, FileNotFoundError istisno qaytaradi.
- w – yozish uchun ochish. Agar fayl mavjud bo'lmasa, u yaratiladi. Agar fayl mavjud bo'lsa, uning ustiga yoziladi. Faylni ochgandan so'ng, ko'rsatkich faylning boshiga o'rnatiladi;
- w+ – yozish va o’qish uchun ochish. Agar fayl mavjud bo'lmasa, u yaratiladi. Agar fayl mavjud bo'lsa, uning ustiga yoziladi. Faylni ochgandan so'ng, ko'rsatkich faylning boshiga o'rnatiladi;
- a – yozish uchun ochish. Agar fayl mavjud bo'lmasa, u yaratiladi. Yozish faylning oxiridan amalga oshiriladi. Avvalgi faylning tarkibi o'chirilmaydi;

- a+ – yozish uchun fayl yaratish. o’qish va yozish uchun ochish. Agar fayl mavjud bo'lmasa, u yaratiladi. Yozish faylning oxiridan amalga oshiriladi. Avvalgi faylning tarkibi o'chirilmaydi;
- x – yozish uchun fayl yaratish. Agar fayl mavjud bo’lsa, FileExistsError xatolik qaytaradi;
- x+ – o’qish va yozish uchun fayl yaratish. Agar fayl mavjud bo’lsa, FileExistsError xatolik qaytaradi.

Eslatma:

Oxirgi ikkita rejimni qo'llab-quvvatlash Python 3.3 dan keyingi versiyalardan boshlab taqdim etilgan.

Rejimni ko'rsatgandan so'ng, modifikator quyidagilarni bajarishi mumkin:

- b – fayl ikkilik(binar) rejimda ochiladi. Fayl metodlari ***bayt*** tipidagi obyektlarni qabul qiladi va qaytaradi.
- t – fayl matn rejimida ochiladi (Windows OTda standart xolda shunday). Fayl metodlari ***str*** tipidagi ob'ektlarni qabul qiladi va qaytaradi. Ushbu rejimda satr oxiridagi belgini qayta ishlash avtomatik ravishda amalga oshiriladi - masalan, Windows OTda o'qish paytida \ r \ n belgilar \ n belgisi bilan almashtiriladi. Masalan, file.txt faylini yaratib, unga ikkita qator yozamiz:

  ```
  >>> f = open ( r"file . txt", "w")        # Yozish uchun faylni ochamiz
  >>> f.write( "Stringl\nString2")          # Faylga ikkita satr yozamiz
  15
  >>> f.close()                             # Faylni yopamiz
  ```

  w rejimini belgilaganimiz uchun, agar fayl mavjud bo'lmasa, u yaratiladi va agar mavjud bo'lsa, uning ustiga yoziladi.

  Endi faylning tarkibini ikkilik(binar) va matnli rejimlarda ko’rib chiqamiz:

  ```
  >>> # Ikkilik(binar) rejimi (\ r belgisi qoladi)
  >>> with open (r"file . txt", "rb") as f:
          for line in f:
              print( repr ( line ) )
  b ' Stringl\r\n '
  b ' String2 '
  ```

```
>>> # Matn rejimi ( \ r belgisi o’chiriladi)
>>> with open ( r"file . txt", "r") as f:
        for line in f:
            print (repr ( line ) )
' Stringl\n '
' String2 '
```

Ishni tezlashtirish uchun yozilgan ma'lumotlar buferlanadi. Buferdan olingan ma'lumotlar faylga faqat faylni yopish paytida yoki funktsiya yoki flush() metodi chaqirilgandan so'ng to'liq yoziladi. Bufer hajmining ixtiyoriy parametrlarini ***buffering***dan foydalanib belgilash mumkin. Agar 0 qiymat sifatida ko'rsatilgan bo'lsa, ma'lumotlar darhol faylga yoziladi (qiymat faqat ikkilik rejimda amal qiladi). Faylga satrma-satr yozishda 1 qiymati ishlatiladi (qiymat faqat matn rejimida amal qiladi), boshqa musbat raqam buferning taxminiy hajmini belgilaydi va manfiy qiymat (yoki qiymat yo'q) tizimning standart o'lchamini belgilaydi. Odatda, matnli fayllar satrma-satr buferlanadi va ikkilik fayllar qismlarga bo'linadi, ularning o'lchamlarini interpretator o'zi 4096 dan 8192 baytgacha diapozoni bo’yicha tanlaydi. Matn rejimidan foydalanganda (standart ravishda), faylni o'qiyotganda ma'lumotlarni Unicode kodlashiga o'tkazishga harakat qilinadi va yozishda qarama-qarshi operatsiya amalga oshiriladi - satr ma'lum bir kodlashda baytlar ketma-ketligiga aylantiriladi. Odatda, tizimda ishlatiladigan kodlash standart belgilanadi. Agar o’tkazishning iloji bo'lmasa, xatolik qaytariladi. ***encoding*** parametri sizga faylni yozishda va o'qishda foydalaniladigan kodlarni belgilashga imkon beradi. Masalan, ma'lumotlarni UTF-8 da kodlashni ko’raylik:

```
>>> f = open(r"file.txt" , "w", encoding="utf -8 ")
>>> f.write( "Satr")                    # Satrni faylga yozamiz
6
>>> f.close ()                          # Faylni yopamiz.
```

Ushbu faylni o'qish uchun faylni ochishda kodlashni aniq belgilashingiz kerak:

```
>>> with open ( r"file . tx t ", "r", encoding="utf-8 ") as f:
        for line in f:
            print ( line)
Satr
```

UTF-8, UTF-16 va UTF-32 kodlashlaridagi fayllar bilan ishlashda, faylning boshida qisqacha BOM (Byte Order Mark – baytlar tartibi belgisi) deb nomlangan xizmat belgilari bo'lishi mumkin. Ushbu belgilar UTF-8 da kodlash uchun ixtiyoriydir va oldingi misolda ular yozish paytida faylga qo'shilmagan. Belgilar qo'shilishi uchun ***encoding*** parametrida utf-8-sig ni ko'rsatishingiz kerak. Qatorni BOM bilan UTF-8 kodlashdagi faylga yozamiz:

```
>>> f = open (r"file . txt " , "w" , encoding="utf-8-sig" )
>>> f.write( "Satr" )                    # Satrni faylga yozamiz
6
>>> f.close() # Faylni yopamiz.
```

Endi encoding parametridagi har xil qiymatdagi faylni o'qiylik:

```
>>> with open ( r"file . txt" , "r" , encoding="utf-8" ) as f :
        for line in f :
                print(repr(line) )
' \u feffSatr'
>>> with open (r"file . tx t", "r" , encoding="utf-8-sig") as f:
        for line in f:
                print(repr(line) )
‘Satr’
```

Birinchi misolda biz utf-8 qiymatini ko'rsatdik, shuning uchun ma'lumotlar bilan birga BOM markeri fayldan o'qildi. Ikkinchi misolda utf-8-sig qiymati ko'rsatilgan, shuning uchun BOM markeri natijaga kiritilmagan. Agar siz faylda marker mavjudligini bilmasangiz va siz markersiz ma'lumotlarni olishingiz kerak bo'lsa, u holda faylni UTF-8 kodlashda o'qiyotganda har doim utf-8-sig qiymatini belgilashingiz kerak. UTF-16 va UTF-32 kodlashlari uchun BOM markeri albatta kerak. ***encoding*** parametrida utf -16 va utf-32 qiymatlarini belgilashda belgi avtomatik ravishda qayta ishlanadi. Ma'lumotlarni yozishda marker avtomatik ravishda faylning boshiga kiritiladi va o'qiyotganda natijada ko'rinmaydi. Satrni faylga yozamiz, keyin uni fayldan o'qiymiz:

```
>>> with open ( r"file.txt ", "w" , encoding="utf-16" ) as f:
        f. write ( "Satr" )
6
>>> with open (r"file.txt", "r", encoding="utf-16") as f:
        for line in f:
```

```
print (repr(line))
```

‘Satr’

Utf-16-le, Utf-16-be, Utf-32-le va Utf-32-be qiymatlaridan foydalanganda faylning boshiga BOM markerini qo'shish kerak va o'qiyotganda uni o'chirish kerak.

errors parametrida siz xatolar bilan ishlash darajasini belgilashingiz mumkin. Bu parametr qo’llashi mumkin bo'lgan qiymatlar:

- "strict" (xatolik ro’y berganda ValueError istisnosini qaytaradi - standart xolda)
- "replace" (noma'lum belgi so’roq belgisi bilan yoki \ufffd kodli belgi bilan almashtiriladi)
- " ignore" (noma'lum belgilar e'tiborga olinmaydi)
- "xmlcharrefreplace" (noma'lum belgi &#xxxx ketma-ketlik bilan almashtiriladi)
- "backslashreplace" (noma'lum belgi \uxxxx ketma-ketligi bilan almashtiriladi)

newline parametri satr oxiridagi belgilar uchun ishlov berish rejimini o'rnatadi. Bu parametr qo’llashi mumkin bo'lgan qiymatlar:

- None (standart xolda) – satr oxiridagi belgilar uchun standart ishlov berishni amalga oshiradi. Masalan, Windows OTda faylni o'qish paytida \r\n belgisi \n ga aylanadi, teskarisi esa yozishda amalga oshiriladi;
- “” (bo’sh satr) – satr oxirigacha ishlov berish amalga oshirilmaydi;
- "<Maxsus belgi>" – ko’rsatilgan maxsus belgi satr oxirini belgilash uchun ishlatiladi va qo'shimcha ishlov berish amalga oshirilmaydi. Faqat \r\n, \r va \n lar maxsus belgilar sifatida ko'rsatilishi mumkin.

13.2. Fayllar bilan ishlash metodlari

Faylni ochganimizdan so'ng open() funktsiyasi fayl bilan keyingi ish olib boriladigan ob'ektni qaytaradi. Ob'ekt turi faylni ochish va buferlash rejimiga bog'liq bo’ladi. Asosiy metodlar bilan tanishtirib o’tamiz:

- ✓ close() – faylni yopadi. Interpretator ob'ektni havolasi bo'lmaganida avtomatik ravishda o'chirib tashlaganligi sababli, kichik dasturlarda siz faylni aniq yopolmaysiz. Biroq, faylni aniq yopish dasturlash uslubining yaxshi belgilaridan biri hisoblanadi. Bundan tashqari, agar fayl yopilmay qolgan bo'lsa, quyidagicha ogohlantirish xabari hosil bo'ladi:
  "Resourcewarning : unclosed file".

Python tili kontekst menejeri protokolini qo'llab-quvvatlaydi. Ushbu protokol kod blokida istisno ro'y berganidan yoki hech qanday istisno yo'qligidan qat'i nazar, fayl yopilishini kafolatlaydi. Misol:

```
with open ( r"file.txt", "w", encoding="cpl251 ") as f:
        f.write ( "Satr")                    # Satrni faylga yozamiz
# Bu yerda fayl avtomatik ravishda yopiladi.
```

- ✓ write(<Ma’lumotlar>) – satrlarni yoki baytlar ketma-ketligini faylga yozadi. Agar parametr sifatida satr ko'rsatilgan bo'lsa, unda fayl matn rejimida ochilishi kerak. Baytlar ketma-ketligini yozish uchun esa faylni ikkilik(binar) rejimda ochish kerak. Ikkilik(binar) rejimda qatorni va matn rejimida baytlar ketma-ketligini yozish mumkin emasligini unutmaslik kerak. Bu metod yozilgan belgilar yoki baytlar sonini qaytaradi. Faylga yozishga misol ko’rib chiqaylik:

```
>>> # Matn rejimi
>>> f = open (r"file.txt", "w", encoding="cp1251 ")
>>> f.write ( "Satrl\n Satr2 ")                    # Satrni faylga yozamiz
15
>>> f . close () # Faylni yopamiz
>>> # Ikkilik (binar) rejimi
>>> f = open (r" file.txt", "wb")
>>> f.write (bytes( "Satrl\nSatr2 ", "cp1251") )
15
>>> f.write (bytearray ( " \nSatr3", "cp1251") )
8
>>> f.close ()
```

- ✓ writelines(<Ketma-ketlik>) – ketma-ketlikni faylga yozadi. Agar ketma-ketlikning barcha elementlari satrlar bo'lsa, unda fayl matn rejimida ochilishi kerak. Agar barcha elementlar baytlar ketma-ketligi bo'lsa, unda faylni ikkilik (binar) rejimda ochish kerak. Ro'yxat elementlarini yozishga misol:

```
>>> # Matn rejimi
>>> f = open (r"file.txt ", "w", encoding="cp 1251" )
>>> f.writelines ( ["Satrl\n", "Satr2 " ] )
```

```
>>> f. close ()
>>> # Ikkilik (binar) rejimi
>>> f = open (r"file . tx t", "wb")
>>> arr = [bytes ( "Satrl \n" , "cp1251" ) , bytes ( "Satr2 ", " cp125 1 " )]
>>> f .writelines(arr)
>>> f.close ()
```

- ✓ writable() – agar faylga yozish mumkin bo'lsa, True (rost), aks holda False (yolg'on) qiymat qaytaradi:

```
>>> f = open ( r"file.txt", "r")          # Faylni o'qish uchun ochamiz
>>> f.writable ()
False
>>> f = open ( r"file.txt", "w" )         # Faylga yozish uchun ochamiz
>>> f .writable ()
True
```

- ✓ read ([<miqdor>]) – fayldan ma'lumotlarni o'qiydi. Agar fayl matn rejimida ochilsa, u holda satr, ikkilik (binar) rejimida esa baytlar ketma-ketligi qaytariladi. Agar parametr ko'rsatilmagan bo'lsa, faylning tarkibi ko'rsatgichning joriy holatidan fayl oxiriga qaytariladi:

```
>>> # Matn rejimi
>>> with open ( r"file.txt" , "r", encoding="cp l251") as f:
f.read ()
'Satrl\nSatr2 '
>>> # Ikkilik (binar) rejimi
>>> with open ( r"file.txt", " rb") as f:
f. read ()
b ' \xdl \xf2\xf0\xee\xea\xe01\n\xdl\xf2\xf0\xee\xea\xe02 '
```

Agar siz parametr sifatida raqamni ko'rsatsangiz, unda har bir murojaatda ko'rsatilgan belgilar yoki baytlar soni qaytariladi. Fayl tugagandan so'ng, metod bo'sh satrni qaytaradi. Misol:

```
>>> # Matn rejimi
>>> f = open (r"file.txt", "r", encoding="cp1251")
```

```
>>> f . read (8)                    # 8 belgisini sanaymiz
' Satr1\n '
>>> f.read (8)                      # 8 belgisini sanaymiz
'Satr2 '
>>> f.read (8)                      # Faylni tugatish
''
>>> f.close ()
```

✓ readline ([<miqdor>]) – har bir murojaatda fayldan bitta qatorni o'qiydi. Agar fayl matn rejimida ochilsa, u holda satr, ikkilik (binar) rejimida esa baytlar ketma-ketligi qaytariladi. Qaytarilayotgan satr uzatilayotgan satr belgilarini o'z ichiga oladi. Oxirgi satr istisno hisoblanadi, agar u satrga o'tkazish belgisi bilan tugamasa, u holda qo'shilmaydi. Fayl tugagandan so'ng, bo'sh satr qaytariladi. Misol:

```
>>> # Matn rejimi
>>> f = open (r"file.tx t", "r", encoding="cp l251")
>>> f . readline ( ) , f . readline ()
( ' Satrl \n ' , 'Satr2 ' )
>>> f.readline ()                   # Faylni tugatish
' '
>>> f . close ()
>>> # -Ikkilik (binar) rejimi
>>> f = open (r"file.tx t ", "rb" )
>>> f . readline ( ) , f.readline ()
(b ' \xdl\xf2\xf0\xee\xea\xe01\n ' , b ' \xdl\xf2\xf0\xee\xea\xe02 ')
>>> f.readline ()                   # Faylni tugatish
b ' '
>>> f.close ()
```

Agar ixtiyoriy parametrda raqam ko'rsatilgan bo'lsa, yangi satrli belgi (\ n)ni o'qish, fayl oxiridagi belgi yoki ko'rsatilgan sonli belgi fayldan o'qilguncha amalga oshiriladi. Boshqacha qilib aytadigan bo'lsak, agar satrdagi belgilar soni parametr qiymatidan kam bo'lsa, unda ko'rsatilgan belgilar soni emas,

balki bitta satr o'qiladi va agar satrdagi belgilar soni ko'proq bo'lsa, unda ko’rsatilgan belgilar soni qaytariladi. Misol:

```
>>> f = open (r"file.txt ", "r", encoding="cpl251 ")
>>> f . readline (2), f.readline (2)
( ‘S’ , ‘a’ )
>>> f . readline (100) # Bu 100 ta belgini emas, balki bitta satrni qaytaradi.
‘trl\n’
>>> f . close ()
```

- ✓ readlines () – ro'yxatdagi fayl tarkibini to'liq o'qiydi. Ro’yxatning har bir elementi bitta satrni, shu jumladan satrga o’tkazish belgisini ham o'z ichiga oladi. Oxirgi satr esa istisno hisoblanadi. Agar satrni oxiri satrga o’tkazish belgisi bilan tugamagan bo’lsa, u belgi avtomatik qo’shilib qolmaydi. Agar fayl matn rejimida ochilgan bo'lsa, unda satrlar ro'yxati, ikkilik rejimda esa baytlar ro'yxati qaytariladi. Misol:

```
>>> # Matn rejimi
>>> with op en ( r" file . tx t" , " r", encoding="cpl251") as f:
f. readlines ()
[ ' Satrl \n ' , ' Satr2 ' ]
>>> # Ikkilik rejimi
>>> with open ( r" file.tx t" , "rb") as f:
f. readlines ()
[b ' \xdl \xf2\xf0\xee\xea\xe01\n ' , b ' \xdl\xf2\xf0\xee\xea\xe02 ']
```

- ✓ _next_ ( ) – har bir murojaatda bitta qatorni o'qiydi. Agar fayl matn rejimida ochilgan bo'lsa, unda satr, ikkilik rejimda esa baytlar ketma-ketligi qaytariladi. Faylni oxiriga yetkanda Stopiteration istisnosini qaytaradi. Misol:

```
>>> # Matn rejimi
>>> f = open(r"file.txt", "r", encoding="cp1251")
>>> f.__next__(), f.__next__()
('Satr1\n', 'Satr2')
>>> f.__next__()                    # Faylni tugatish
Traceback (most recent call last):
 File "<pyshell#26>", line 1, in <module>
```

```
f.__next__()                    # Faylni tugatish
StopIteration
>>> f.close()
```

__Next__ () metodi tufayli biz for operatori yordamida fayl satrlari bo'yicha satrlarni takrorlashimiz mumkin. For operatori avtomatik ravishda har bir takrorlashda __next__ () metodini chaqiradi. Barcha satrlarni qabul qilishi uchun satrga o'tkazish belgisi (\n)ni olib tashlashimiz kerak bo'ladi:

```
>>> f = open (r"file . txt ", "r", encoding="cp1251 ")
>>> for line in f : print ( line.rstrip ( " \n" ) , end=" " )
Satr1 Satr2
>>> f.close ()
```

- ✓ flush () – buferdan ma'lumotlarni diskka majburiy ravishda yozadi.
- ✓ fileno () – faylning butun deskriptorini qaytaradi. Qaytaradigan qiymati har doim 2 dan katta bo'ladi, chunki 0 *stdin* kiritish standartiga bog'langan, 1 *stdout* chiqarish standartiga bog'langan va 2 *stderr* xatolik haqida habar chiqarish standartiga o'rnatiladi. Misol:

```
>>> f = open ( r" file . txt ", "r", encoding="cp l251" )
>>> f. fileno ()              # Fayl deskriptori
3
>>> f . close ()
```

- ✓ truncate ([<Miqdor>]) - faylni ko'rsatilgan miqdordagi sonli belgilarga (agar matn rejimi ko'rsatilgan bo'lsa) yoki baytlarga (ikkilik rejimda) qisqartiradi. Ushbu metod yangi fayl hajmini qaytaradi. Misol:

```
>>> f = open (r"file . txt ", "r+", encoding="cpl251 ")
>>> f.read ()
'Satrl\nSatr2 '
>>> f . truncate (4)
4
>>> f.close ()
>>> with open ( r"file . txt", " r", encoding="cp1251") as f:
f. read ( )
'Sat'
```

- ✓ tell () – faylning boshiga nisbatan ko'rsatgich o'rnini butun son sifatida qaytaradi. Shuni e’tiborgaolish kerakki, Windows OTda tell () metodi \ r belgisini qo'shimcha bayt sifatida qabul qiladi, agar bu fayl matn rejimida ochilgan bo’lsa o'chiriladi. Misol:

```
>>> with open ( r"file . tx t", "w", encoding="cpl251 ") as f:
f.write ( "Stringl\nString2")
15
>>> f = open (r"file . txt", "r", encoding="cp1251")
>>> f.tell () # Ko'rsatkich faylning boshida joylashgan
0
>>> f.readline () # Ko'rsatkichni joyini o’zgartirish
' Stringl\n '
>>> f.tell () # Qaytgan natija 9 ( 8 + ' \ r' ) ,  8 emas ! ! !
9
>>> f close ()
```

Ushbu nomuvofiqlikni oldini olish uchun faylni matnda emas, ikkilik rejimda oching:

```
>>> f = open ( r"file . txt ", "rb")
>>> f. readline () # Ko'rsatkichni joyini o’zgartirish
b ' Stringl\r\n '
>>> f. tell () # Endi qiymat mos keladi
9
>>> f . close ()
```

Fayl ob'ektlari metodlardan tashqari bir nechta atributlarni qo'llab-quvvatlaydi:

- ✓ name – faylning nomi
- ✓ mode – faylni ochilgan rejimi
- ✓ closed – agar fayl yopilgan bo’lsa True(rost) qiymat, aks holda False(yolg’on) qiymat qaytaradi. Misollar:

```
>>> f = open (r"file . txt ", "r+b")
>>> f.name, f.mode, f.closed
('file.txt', 'rb+' , False)      # faylni yopmaganimiz uchun False qiymat qaytardi
```

```
>>> f.close ()                # faylni yopamiz
>>> f.closed
True
```

- ✓ encoding - faylga yozishdan oldin yoki o'qiyotganda satrlarni konvertatsiya qilish uchun ishlatiladigan kodlash nomi. Atribut faqat matn rejimida ishlaydi. Shuni ham unutmangki, atributning qiymatini o'zgartira olmaysiz, chunki uni faqat o'qish mumkin. Misol:

```
>>> f = open (r"file . txt", "a", encoding="cpl25 1")
>>> f . encoding
' cp1251 '
>>> f . close ()
```

- ✓ buffer – buferga kirishga imkon beradi. Atribut faqat matn rejimida ishlaydi. Ushbu ob'ekt yordamida siz matn oqimiga baytlar ketma-ketligini yozishingiz mumkin. Misol:

```
>>> f = open (r"file . txt ", "w", encoding="cp1251")
>>> f.buffer . write (bytes ( "С трока", "cp125 1") )
6
>>> f . close ()
```

«Python dasturlash tili»
fanidan amaliy mashg’ulotlarini o’tkazish bo’yicha
USLUBIY KO’RSATMA

1-amaliy *ish*

Python dasturlash tili bilan tanishish. Pythonni o’rnatish. Dastur natijasini chop etish. Ma’lumotlarni kiritish

Ishdan maqsad: Python dasturlash tili tuzilishi bilan tanishtirish va unda dastur sodda tuzish ko’nikmalarini shakllantirish. Python ni o’rnatish va Python shell muhitiada ishlash bo’yicha amaliy malakalarni oshirish. Python dasturlash muhitida dastur natijasini chop etish va ma’lumot kiritish bo’yicha amaliy ko’nikmalarni shakllantirish.

Masalaning qo’yilishi: Talaba variant bo’yicha berilgan masalani Python dasturlash tilida dasturini tuzishi va Python shell yordamida kerakli natija olishi lozim.

Ishni bajarish uchun namuna

1-misol: Aylana uzunligi *l=24* berilgan. Bu aylananing radiusini va aylana chegaralagan doiraning yuzasini hisoblash dasturi tuzilsin. π=3,14 ga teng deb olinsin.

Matematik ifodalanishi (matematik modeli): $r = \frac{l}{2\pi}; \quad s = \pi r^2$

Bunda r – aylana radiusi, s – doira yuzasi

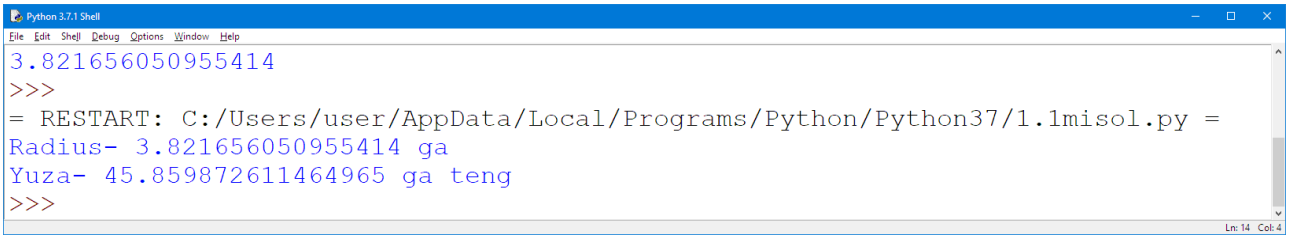
Python shell muhitini ishga tushuramiz. NewFile buyrug’ini beramiz. Yangi sahifaga quyidagi dastur kodini yozamiz va 1.1-misol.py nomiu bilan saqlaymiz.

Dastur kodi:

1.1-misol.py fayli:

```
l=24
r=l/(2*3.14)
s=3.14*r*r
print("Radius-",r, "ga")
print("Yuza-",s, "ga teng")
```

Dastur ishlashi natijasi:



2-misol: Uchburchakning *a*, *b*, *c* tomonlari berilgan. Uning yuzasi va perimetrini topish dasturi tuzilsin. a=7; b=6; c=9

Matematik ifodalanishi (matematik modeli): $p = \frac{a+b+c}{2}; s = \sqrt{p(p-a)(p-b)(p-c)}$

p – uchburchak yarim pirimetri; s – yuzasi;

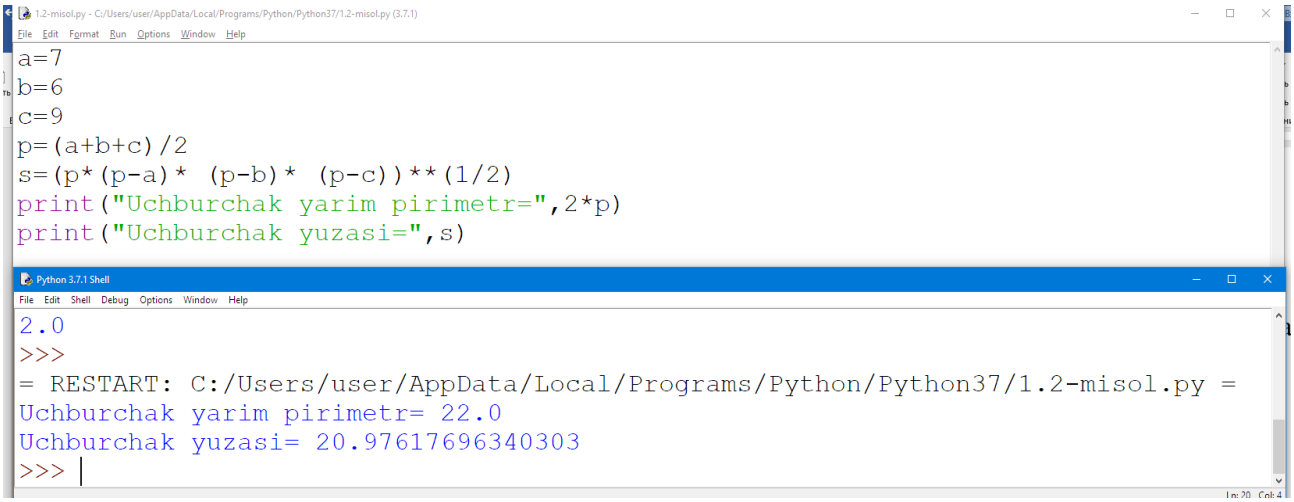
Python shell muhitini ishga tushuramiz. NewFile buyrug'ini beramiz. Yangi sahifaga quyidagi dastur kodini yozamiz va 1.1-misol.py nomiu bilan saqlaymiz.

Dastur kodi:

1.2-misol.py fayli:

```
a=7
b=6
c=9
p=(a+b+c)/2
s=(p*(p-a)* (p-b)* (p-c))**(1/2)
print("Uchburchak yarim pirimetr=",2*p)
print("Uchburchak yuzasi=",s)
```

Dastur ishlashi natijasi:



1-amaliy ish topshiriq variantlari

1. Kvadratning tomoni *a* berilgan. Uning perimetrini va yuzasini topish dasturi tuzilsin.
2. Temperaturaning Farengeyt gradusidagi qiymati berilgan. Shu temperaturaning Selsiydagi qiymati hisoblansin. $T_C = (T_F - 32) \cdot \frac{5}{9}$
3. *l* santimetr uzunlik berilgan. Uning necha metr ekanligi aniqlansin.
4. Temperaturaning Selsiy gradusidagi qiymati berilgan. Shu temperaturaning Farengeytdagi qiymati hisoblansin. $T_F = T_C \cdot \frac{9}{5} + 32$
5. *m* massa kilogrammlarda berilgan. Uning tonnalardagi qiymati topilsin.
6. *x* kg mahsulotning narxi *a* so'm. Shu mahsulotning 1 kgi va *y* kgi necha so'm ekanligi aniqlansin.
7. *f* fayl o'lchami bitlarda berilgan. Uning necha kilobayt ekanligi topilsin.
8. *s* yuza sm^2 larda berilgan. Uni dm^2 lardagi qiymati hisoblansin.
9. Kubning qirrasining uzunligi berilgan. Kubning hajmini va yon sirtining yuzini hisoblash algoritmini tuzing.
10. To'g'ri to'rtburchakning *a* va *b* tomonlari berilgan. Uning perimetri va yuzasini topish dasturi tuzilsin.
11. Ikkita haqiqiy musbat son berilgan. Ularning o'rta arifmetigi va o'rta geometrigini hisoblash dasturini tuzing.
12. Aylananing diametri *d* berilgan. Uning uzunligini topish dasturi tuzilsin. π=3,14 ga teng **deb olinsin**.
13. To'g'ri burchakli uchburchak katetlarining uzunligi berilgan. Uning gipotenuzasi va yuzasini topish dasturini tuzing.
14. Kubning qirrasi *a* berilgan. Uning hajmini va sirti yuzasini topish dasturi tuzilsin.
15. To'g'ri burchakli parallelipipedning *a*, *b*, va *c* qirralari berilgan. Uning hajmini va to'la sirtini topish dasturi tuzilsin.
16. *r* radiusli aylanaga tashqi chizilgan muntazam *n* burchakning peremetrini hisoblash dasturini tuzing.
17. *a* va *b* nomanfiy sonlari berilgan. Bu sonlarning o'rta arifmetigini va o'rta geometrigini topish dasturi tuzilsin.
18. Ikkita *a* va *b* nomanfiy sonlari berilgan. Ularning yig'indisi, ayirmasi, ko'paytmasi va bo'linmasini topish dasturi topilsin.
19. *h* balandlikdan tushayotgan *m* massali toshning tushish vaqtini hisoblash dasturini tuzing.
20. To'g'ri burchakli uchburchakning *a* va *b* katetlari berilga. Uning gipotenuzasi va perimetruini topish dasturi tuzilsin.
21. Umumiy markazli r_1 va r_2 radiusli ikkita doira berilgan. Har ikkala doira va bu doiralar orqali hosil bo'lgan xalqaning yuzalarini hisoblash dasturi tuzilsin.
22. Aylana uzunligi *l* berilgan. Bu aylananing radiusini va aylana chegaralagan doiraning yuzasini hisoblash dasturi tuzilsin. π=3,14 ga teng deb olinsin.
23. Doiraning yuzasi *S* berilgan. Uning diametri *d* va doirani chegaralab turuvchi aylana uzunligi hisoblansin. π=3,14 ga teng deb olinsin.
24. Koordinatalar tekisligida $A(x_1, y_1)$ va $B(x_2, y_2)$ nuqtalar berilgan. *AB* kesma uzunligini hisoblash dasturi tuzilsin.
25. *a* va *b* o'zgaruvchilar berilgan. Ularning qiymatlari o'zaro almashtirilsin. *a* va *b* larning yangi qiymatlari chop etilsin.

26. *a*, *b* va *c* o’zgaruvchilar berilgan. *a* ning qiymati *b* ga, *b* ning qiymati *c* ga, *c* ning qiymati *a* ga almashtirilsin. *a*, *b* va *c* larning yangi qiymatlari chop etilsin.
27. Sutka boshlanganiga *n* soniya bo’ldi. Sutka tugashiga necha daqiqa qolganligi topilsin.
28. *a* va *b* sonlari berilgan. $ax+b=0$ chiziqli tenglamaning yechimini topish dastuiri tuzilsin.
29. *l* santimetr uzunlik berilgan. Uning necha metr ekanligi aniqlansin.
30. *s* yuza sm^2 larda berilgan. Uni dm^2 lardagi qiymati hisoblansin.

2-amaliy *ish*

O’zgaruvchilar. O’zgaruvchilar ustida amallar. Ma’lumot turlari. Ma’lumot turlarini aniqlash, o’zgartirish

Ishdan maqsad: Python dasturlash tilida o’zgaruchi tushunchasi bian tanishish, ma’lumot turlarini o’rganish, ulardan foydalana olish. Python dasturlash tilida o’zgaruvchilarni ishlatish, va turlarini aniqlash va o’zgartirishni hamda turli ifodalarni yozishni o’rganish. Murakkab ifodalarni dasturini tuzish va hisoblash.

Masalaning qo’yilishi: Talaba variant bo’yicha berilgan masalani Python dasturlash tilida dasturini tuzishi va kerakli natija olishi lozim.

Ishni bajarish uchun namuna

1-misol: Koordinatalar tekisligida $A(x_1, y_1)$ va $B(x_2, y_2)$ nuqtalar berilgan. *AB* kesma uzunligini hisoblash dasturi tuzilsin.

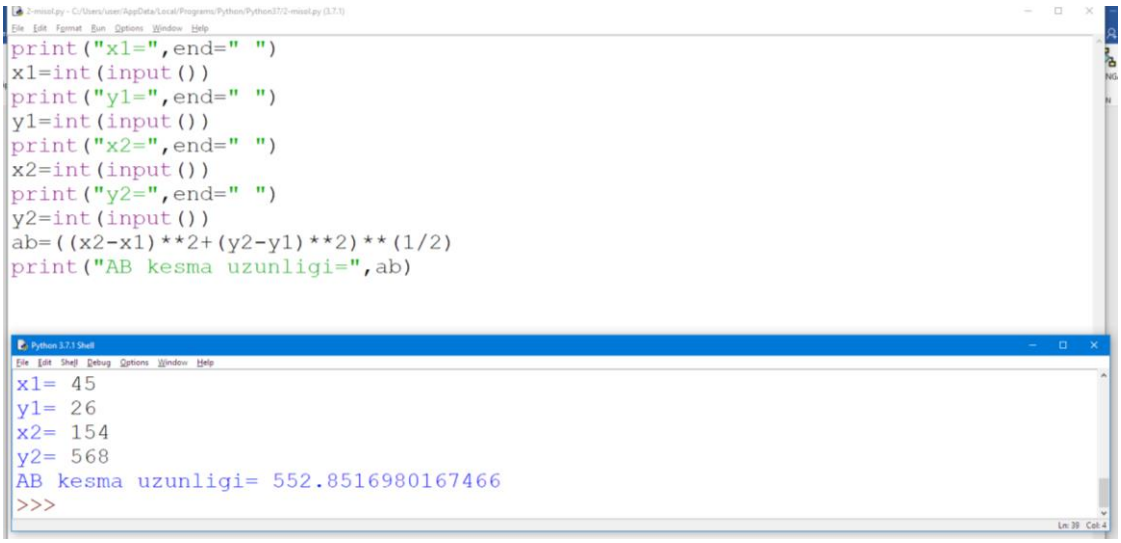
Matematik ifodalanishi: $AB = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$

Dastur kodi:

2.1-misol.py fayli:

```
print("x1=",end=" ")
x1=int(input())
print("y1=",end=" ")
y1=int(input())
print("x2=",end=" ")
x2=int(input())
print("y2=",end=" ")
y2=int(input())
ab=((x2-x1)**2+(y2-y1)**2)**(1/2)
print("AB kesma uzunligi=",ab)
```

Dastur ishlashi natijasi:



2-amaliy ish topshiriq variantlari

1.	Turg'un suvdagi qayiq tezligi V km/s. Daryo suvi oqimining tezligi U km/s (U<V). Qayiq ko'lda T_1 soat, daryoda esa (oqimga qarshi) T_2 soat harakat qilgan. Qayiq suzgan umumiy S masofa topilsin.
2.	Birinchi avtomobil tezligi V_1 km/s, ikkinchisiniki - V_2 km/s, ular orasidagi masofa - S km. Avtomobillar bir-biridan uzoqlashsa (bir-biriga qarab harakat qilganda), T soatdan keyin ular orasidagi masofa qanday bo'ladi?
3.	Asoslari a va b (a>b), katta asosdagi burchagi α bo'lgan teng yonli trapetsiyaning perimetri hamda yuzasi topilsin (burchak radianda beriladi).
4.	Noldan farqli berilgan R_1, R_2, R_3 elektr qarshiliklari uchun R hisoblansin. Bunda: $\frac{1}{R}=\frac{1}{R_1}+\frac{1}{R_2}+\frac{1}{R_3}$.
5.	Xodimning oylik ish haqiga 45% mukofot puli qo'shilsin. Hosil bo'lgan miqdordan 17% daromad solig'i, 1,5% kasaba uyushmasi va 1% nafaqa solig'i ushlab qolinsin. Qo'lga tegadigan pul miqdori chop etilsin.
6.	Teng tomonli uchburchak tomoni berilgan, uchburchak yuzasi topilsin.
7.	Uchta musbat son berilgan. Sonlar o'rta geometrigining kasr qismi topilsin.
8.	Berilgan katetlari bo'yicha to'g'ri burchakli uchburchakning perimetri va yuzasi hisoblansin.
9.	Berilgan ikki tomoni va ular orasidagi burchak (gradusda) asosida uchburchakning uchinchi tomoni va yuzasi topilsin.
10.	Og'irligi bir kilogramm bo'lgan mahsulotning narxi berilgan. Uning og'irligi grammlarda kiritilsin va to'lash zarur bo'lgan pul miqdori chop etilsin.
11.	10 metr radiusli silindrik shaklga ega bo'lgan suv bosimi minorasidagi suv sathining balandligi berilgan bo'lsa, suvning hajmi hisoblansin.
12.	Bolalar bog'chasiga bir oylik to'lov 70000 so'm (bir oy - 22 kun). Agar bola bog'chaga N (0<N<23) kun kelmagan bo'lsa, bir oy uchun qancha to'lash kerak bo'ladi?
13.	R radiusli doiraga ichki chizilgan muntazam n-burchakning perimetri va yuzasi hisoblansin.
14.	Aylana diametri *d* berilgan. Uning uzunligini toping $L=\pi \cdot d$.
15.	Kubning tomonlari *a* berilgan. Uning hajmini $V=a^3$ va sirti maydonini toping $P=6 \cdot a^2$.
16.	To'g'ri burchakli paralelopipedning *a, b, s* tomonlari berilgan. Uning hajmini va sirti maydonini toping: $V=a \cdot b \cdot s;\quad S=2 \cdot (a \cdot b+ b \cdot s+ a \cdot s)$.
17.	Aylana radiusi *R* beilgan. Aylana uzunligi *L* va uning ichi maydoni *S* ni toping: $L=2 \cdot \pi \cdot R,\ S=\pi \cdot R^2$. Bu erda π sonini 3.14 ga teng qilib oling.

18.	Ikkita *a* va *b* haqiqiy sonlari berilgan. Ularning o'rta arifmetik qiymatini toping: *(a+b)/2.*
19.	Ikkita *a* va *b* haqiqiy musbat sonlari berilgan. Ularning o'rta geometrik qiymatini toping: $\sqrt{a \cdot b}$.
20.	Doira yuzi *S* berilgan. Uning diametri *D* va aylanasi uzunligi *L* ni $L = 2 \cdot \pi \cdot R,\ \ S = \pi \cdot R^2$ formulalardan foydalanib toping.
21.	Sonlar koordinata o'qida joylashgan ikkita nuqta x_1 va x_2 orasidagi masofani toping: $d=\lvert x_2 - x_1 \rvert$.
22.	Sonlar koordinata o'qida uchta nuqta berilgan A, B, S. AS va BS kesmalari uzunliklari va ular yig'indisini toping.
23.	Uchburchakning uchta uchi koordinatalari berilgan: (x_1,y_1), (x_2,y_2), (x_3,y_3). Uning perimetrini va maydonini toring. Uchburchakning maydonini topishda Geron formulasidan foydalaning $S = \sqrt{p \cdot (p-a) \cdot (p-b) \cdot (p-c)}$. Bu erda *a, b, s* - uchburchak tomonlari, *p=(a+b+s)/2* – yarim perimetr.
24.	*V=S/T* tezlikni hisoblash formulasi orqali, masofa va vaqtni berib jism tezigini hisoblang.
25.	Uchta *a, b, s* o'zgaruvchilar qiymatlarini ketma-ket almashtiring (*a* qiymatini *b* ga, *b* qiymatini *s* ga va *s* qiymatini *a* ga).
26.	Uchta *a, b, s* o'zgaruvchilar qiymatlarini ketma-ket almashtiring (*a* qiymatini *s* ga, *s* qiymatini *b* ga va *b* qiymatini *a* ga).
27.	Kvadratning tomonlari *a* berilgan. Uning perimetrini toping $P=4 \cdot a$.
28.	Kvadratning tomonlari *a* berilgan. Uning yuzasini toping $P=a^2$.

3-amaliy *ish*.

Operatorlar. Shartli operatorlar

Ishdan maqsad: Python dasturlash tilida operatorlar bilan ishlash, ulardan foydalanish ko'nikmalariga ega bo'lish. Dasturda inkrement, dekrement, mantiqiy, razryadli, taqqoslash amallaridan foydalana olish. Python dasturlash tilida shartlar bilan ishlash, tarmoqlanuvhci jarayonlar uchun dastur yozish ko'nikmalariga ega bo'lish.

Masalaning qo'yilishi: Talaba variant bo'yicha berilgan masalani Python dasturlash tilida ishlashi va kerakli natija olishi lozim.

Ishni bajarish uchun namuna

1-misol: Berilgan uch xonali butun sonning raqamlari o'zaro teng yoki teng emasligi aniqlansin.

Echish usuli. Masala Python dasturlash tilining butun sonlar ustidagi arifmetik amallardan foydalangan holda yechiladi. Berilgan butun a va b sonlar uchun '/' amali a/b bo'linmaning butun qismini, '%' amali a%b bo'linmaning butun qoldiqini beradi. Bu bo'lishlardan foydalanib, berilgan sonning raqamlarini ajratib olish va ularni o'zaro solishtirish mumkin.

Dastur kodi

3.1-misol.py fayli:

```
n=int(input("n - qiymatini kiriting: "))
if (n<100 or n>999):
   print("Kiritilgan son 3 xonali emas!")
else:
  a2=n//100
  a1=(n%100)//10
  a0=n%10
  print("Berilgan son raqamlari o'zaro teng",end="")
  if(a2!=a1 and a1!=a0 and a2!=a0): print(" emas!")
  else: print("!")
```

Dastur ishlashi natijasidan namunalar:

```
n=int(input("n - qiymatini kiriting: "))
if (n<100 or n>999):
   print("Kiritilgan son 3 xonali emas!")
else:
  a2=n//100
  a1=(n%100)//10
  a0=n%10
  print("Berilgan son raqamlari o'zaro teng",end="")
  if(a2!=a1 and a1!=a0 and a2!=a0): print(" emas!")
  else: print("!")
```

```
n - qiymatini kiriting: 555
Berilgan son raqamlari o'zaro teng!
>>>
==== RESTART: C:/Users/user/AppData/Local/Programs/Python/Python37/3.3.py ====
n - qiymatini kiriting: 12
Kiritilgan son 3 xonali emas!
>>>
==== RESTART: C:/Users/user/AppData/Local/Programs/Python/Python37/3.3.py ====
n - qiymatini kiriting: 456
Berilgan son raqamlari o'zaro teng emas!
```

3.1-amaliy ish topshiriqlari

1.	*a, b, c* butun sonlar berilgan bo'lsa, ularning $a \le b \le c$ shartni qanoatlantirishi aniqlansin.
2.	*a, b, c* butun sonlar berilgan. *b* sonining, *a* va *c* sonlar orasida yotishi aniqlansin.
3.	*a* va *b* butun sonlar berilgan. Ularning bir vaqtda toq bo'lmasligi aniqlansin.
4.	*a* va *b* butun sonlar berilgan bo'lsa, ularning hech bo'lmaganda bittasi toq ekanligi aniqlansin.
5.	*a* va *b* butun sonlar berilgan bo'lsa, bu sonlardan biri toq ekanligi aniqlansin.
6.	*a* va *b* butun sonlar berilgan bo'lsa, ularning bir xil juftlikka ega ekanligi aniqlansin.
7.	*a, b, c* butun sonlar berilgan. Ularning har biri musbat ekanligi tekshirilsin.
8.	*a, b, c* butun sonlar berilgan bo'lsa, ularning hech bo'lmaganda bittasi musbat ekanligi aniqlansin.
9.	*a, b, c* butun sonlar berilgan bo'lsa, ulardan faqat bittasi musbat bo'lishi aniqlansin.
10.	*a, b, c* butun sonlar berilgan bo'lsa, ulardan faqat ikkitasi bir vaqtda musbat ekanligi aniqlansin.
11.	Butun musbat son berilgan bo'lsa, uning bir vaqtda juft va ikki xonali ekanligi aniqlansin.
12.	Butun musbat son berilgan bo'lsa, uning bir vaqtda toqligi va uch xonali ekanligi aniqlansin.
13.	*a, b, c* butun sonlar berilgan. *b* sonining, *a* va *c* sonlar orasida yotishi aniqlansin.
14.	*a* va *b* butun sonlar berilgan. Ularning bir vaqtda toq bo'lmasligi aniqlansin.
15.	*a* va *b* butun sonlar berilgan bo'lsa, ularning hech bo'lmaganda bittasi toq ekanligi aniqlansin.
16.	Berilgan uchta sondan juftliklar hosil qilingan. Shu juftliklarning hech bo'lmaganda bittasidagi sonlar o'zaro teng bo'lishi aniqlansin.
17.	Berilgan uchta butun sonlar orasidan olingan juftliklardan hech bo'lmaganda bittasidagi sonlar ishoralari bilan farq qilishi aniqlansin.
18.	Uch xonali son berilgan. Uning raqamlari o'suvchi ketma-ketlik tashkil etishi

	aniqlansin.
19.	Uch xonali son berilgan. Uning raqamlari o‘suvchi yoki kamayuvchi ketma-ketlik tashkil etishi aniqlansin.
20.	To‘rt xonali son berilgan. Uni chapdan o‘ngga va o‘ngdan chapga o‘qiganda bir xil o‘qilishi aniqlansin.
21.	*a, b, c* sonlar berilgan (*a≠0*). Bu sonlarni kvadrat tenglama koeffisientlari deb hisoblab, shu kvadrat tenglamaning haqiqiy yechimga ega ekanligi aniqlansin.
22.	*x, y* sonlari berilgan. Ularni tekislikdagi nuqta koordinatalari deb hisoblab, shu nuqtaning 2-chorakda yotishi aniqlansin.
23.	*x, y* sonlari berilgan. Ularni tekislikdagi nuqta koordinatalari deb hisoblab, ularning 2- yoki 3-chorakda yotishi aniqlansin.
24.	Tekislikda nuqta *x* va *y* koordinatalari bilan berilgan. Shu nuqta (yuqori chap burchagi (x_1,y_1), quyi o‘ng burchagi (x_3,y_3) bo‘lgan, hamda tomonlari koordinata o‘qlariga parallel) to‘g‘ri burchakli to‘rtburchakning ichida yotishi yoki yotmasligi aniqlansin.
25.	Uchta butun son berilgan bo’lsa, shu sonlarning uchburchakning tomonlarini tashkil etishi aniqlansin.
26.	*a,b,c* butun sonlar berilgan bo‘lib, ular uchburchakning tomonlarini tashkil etsa, shu uchburchakning teng yonli ekanligi aniqlansin.
27.	*a,b,c* butun sonlar berilgan bo‘lib, ular uchburchakning tomonlarini tashkil etsa, shu uchburchakning to‘g‘ri burchakli ekanligi aniqlansin.
28.	Uch xonali son berilgan. Bu son raqamlarining har xil ekanligi aniqlansin.

4-amaliy *ish*.

Sikl operatorlari. For va While sikli opertorlari bilan ishlash

Ishdan maqsad: Python dasturlash tilida sikllar bilan ishlash, uning turli ko'rinishlaridan foydalanish ko'nikmalariga ega bo'lish. Dasturda for va while operatorlaridan foydalana olish.

Masalaning qo'yilishi: Talaba variant bo'yicha berilgan masalani Python dasturlash tilida ishlashi va kerakli natijalarni olishi lozim.

Ishni bajarish uchun namuna

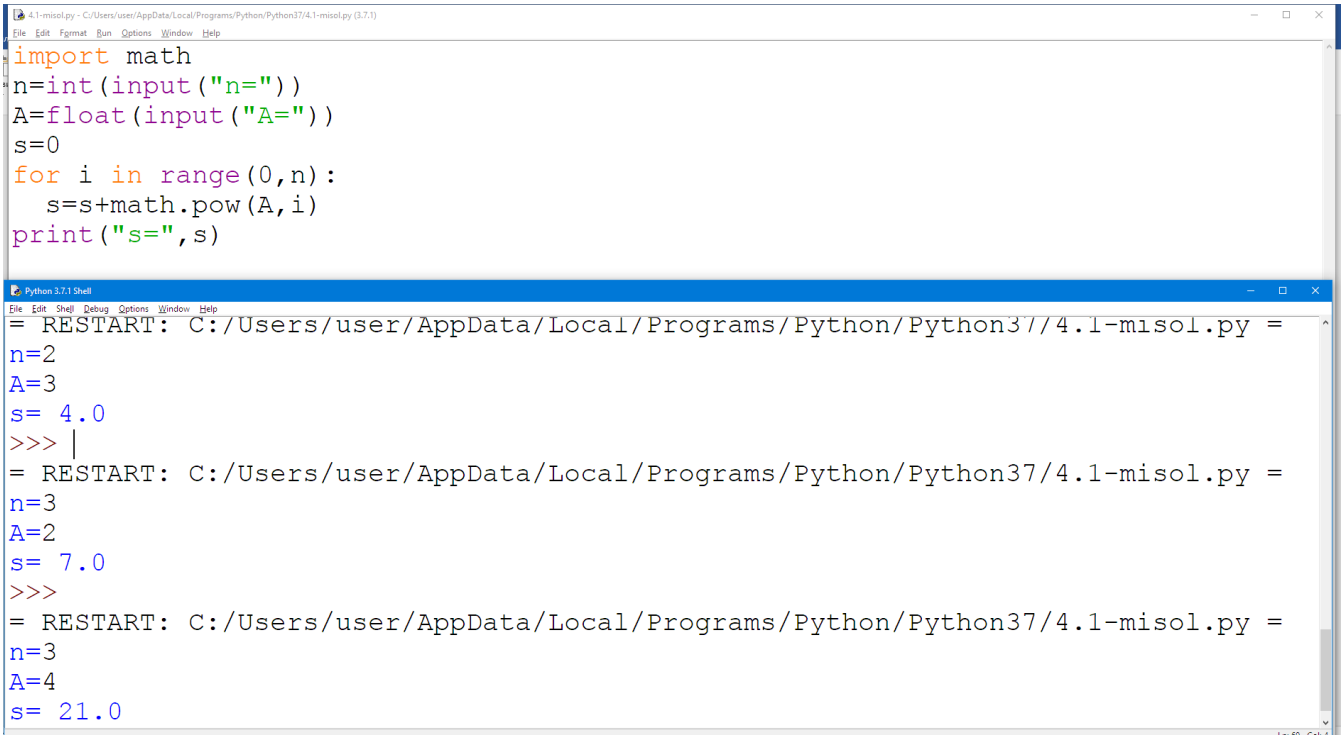
Misol: Butun N>0 soni va haqiyqiy A son berilgan. Quyidagi yig'indini hisoblang. Faqat bitta sikl ishlating. $1+A+A^2+A^3+\ldots+A^N$

Dastur kodi:

4-misol.py fayli:

```
import math
n=int(input("n="))
A=float(input("A="))
s=0
for i in range(0,n):
  s=s+math.pow(A,i)
print("s=",s)
```

Dastur ishlashi natijasi:



4-amaliy ish topshiriq variantlari

1. K va N (N>0) butun sonlari berilgan. K sonini N marta ekranga chiqaring.

2. Ikkita **A va B** (**A<B**) butun sonlari berilgan. **A va B** oralig'ida joylashgan sonlarni o'sish tartibida (**A va B** larni ham hisobga olgan holda) ekranga chiqaring, hamda ular sonini hisoblang.

3. Ikkita **A va B** (**A<B**) butun sonlari berilgan. **A va B** oralig'ida joylashgan sonlarni kamayish tartibida (**A va B** larni ham hisobga olgan holda) ekranga chiqaring, hamda ular sonini hisoblang.

4. Bir kilogramm konfet narxini ifodalovchi haqiqiy son berilgan. 1 kg, 2 kg,...10 kg konfetlar baholarini hisoblang.
   A) 0.1, 0.2,...1 kg. konfetlar narxlarini hisoblab chiqaring.
   B) 1.2, 1.4, ...2 kg konfetlar narxlarini hisoblab chiqaring.

5. Ikkita **A va B** (**A<B**) butut sonlari berilgan. **A va B** sonlar orasidagi barcha butun sonlar yig'indisini toping(**A va B** larni ham hisobga olgan holda).

6. Ikkita **A va B** (**A<B**) butun sonlari berilgan. **A va B** sonlar orasidagi barcha butun sonlar kvadrati yig'indisini toping(**A va B** larni ham hisobga olgan holda).

7. Butun N va K sonlari berilgan. Quyidagi yig'indini toping
   $$N^2 + (N+1)^2 + (N+2)^2 + \ldots\ldots + (N+K)^2$$
8. Butun N>0 soni berilgan. Quyidagi N ta ko'paytuvchilar ko'paytmasini toping.
   $$1.1*1.2*1.3*.....$$
9. Butun N>0 soni berilgan. N ta ishorasi almashuvchi yig'indilardan iborat ifoda qiymatini toping (Shartli o'tish operatori ishlatilmasin).
   $$1.1-1.2+1.3-...$$
10. Butun N>0 toq soni berilgan. Quyidagi formula orqali hisoblanuvchi sonning kvadratini toping.
    $$1+3+5+\ldots\ldots N$$
11. Har bir yig'indi joriy qiymati ekranga chiqarilsin (natijada 1 dan N gacha toq butun sonlar kvadratlari chiqarilsin).
12. Haqiqiy A va butun N>0 sonlari berilgan. A ning N darajasini toping (A soni N marta ko'paytirilsin)
    $$A^N = A*A*\ldots..*A$$
13. Butun N>0 soni berilgan. Ifoda qiymatini toping, bunda Shartli operator foydalanilmasin.
    $$1-A+A^2-A^3+....+(-1)^N*A^N$$
14. Butun N>0 soni berilgan. Ko'paytmani toping (N-faktorial). O'zgaruvchilar haqiqiy tipli deb olinsin.
    $$N!=1*2*\ldots. *N$$

15. Butun N>0 soni berilgan. Bir sikldan foydalanib yig'indini toping. O'zgaruvchilar haqiqiy tipli deb olinsin.

$$1!+2!+3!+\ldots+N!$$

16. Butun N>0 soni berilgan. Bir sikldan foydalanib yig'indini toping.

$$1+1/(1!)+1/(2!)+1/(3!)+\ldots.+1/(N!)$$

O'zgaruvchilar haqiqiy tipli deb olinsin. (N!-faktorial. N!=1*2*.. N.)
Topilgan yig'indi e=exr(1) o'zgarmasning taqribiy qiymati bo'ladi.

17. Haqiqiy X va butun N>0 sonlari berilgan. Ifoda qiymatini toping.
$1+X+X^2/(2)+\ldots X^N/(N!)$
18.
19. (N!=1*2*.. N) hisoblangan ifoda qiymati $u=e^x$ funktsiyaning x nuqtadagi taqribiy qiymati bo'ladi.
20. Haqiqiy X va butun N>0 sonlari berilgan. Ifoda qiymatini toping.
$X-X^3/(3!)+X^5/(5!)-.....+(-1)^N*X^{2*N+1}/((2*N)!)$
21. (N!=1*2*.. N) hisoblangan ifoda qiymati sinx funktsiyaning x nuqtadagi taqribiy qiymati bo'ladi.
22. Haqiqiy X (|X|<1) va butun N>0 sonlari berilgan. Ifoda qiymatini toping.
$1-X^2/(2!)+X^4/(4!)-.....+(-1)^N*X^{2*N+1}/((2*N)!)$
23. (N!=1*2*.. N) hisoblangan ifoda qiymati sos(x) funktsiyaning x nuqtadagi taqribiy qiymati bo'ladi.
24. Haqiqiy X (|X|<1) va butun N>0 sonlari berilgan. Ifoda qiymatini toping.
$X-X^2/2+X^3/4-.....+(-1)^{N-1}*X^{N+1}/N$

25. (N!=1*2*.. N) hisoblangan ifoda qiymati ln(x) funktsiyaning x nuqtadagi taqribiy qiymati bo'ladi.
26. Haqiqiy X (|X|<1) va butun N>0 sonlari berilgan. Ifoda qiymatini toping.
$X-X^3/3+X^5/5-.....+(-1)^N*X^{2*N+1}/(2*N+1)$
27. (N!=1*2*.. N) hisoblangan ifoda qiymati arstg(x) funktsiyaning x nuqtadagi taqribiy qiymati bo'ladi.
28. Haqiqiy X (|X|<1) va butun N>0 sonlari berilgan. Ifoda qiymatini toping.
$X+1*X^3/(2*3)+1*3*X^5/(2*4*5)+.....+1*3*\ldots.*(2*N-1)*$
$X^{2*N+1}/(2*4*\ldots.*(2*N)*(2*N+1))$
29. (N!=1*2*.. N) hisoblangan ifoda qiymati arssin(x) funktsiyaning x nuqtadagi taqribiy qiymati bo'ladi.

30. Haqiqiy X (|X|<1) va butun N>0 sonlari berilgan. Ifoda qiymatini toping.

$$1+X/2-1*X^2/(2*4)+1*3*X^3/(2*4*6)-.....+(-1)^{N-1}*1*3*\ldots(2*N-3)*X^N/(2*4*\ldots*(2*N))$$

31. (N!=1*2*.. N) hisoblangan ifoda qiymati $\sqrt{1+x}$ funktsiyaning x nuqtadagi taqribiy qiymati bo'ladi.

5-amaliy *ish*

Sonlar. Sonlar. Sonlar bilan ishlashning tashqi funksiya va metodlari. math moduli. random moduli.

Ishdan maqsad: Talabalarda Python dasturlash tilida sonlar bilan ishash funksiyalarni bilan tanishish, funksiyalarni qo'llab dasturlar tuzish malakalarini shakllantirish. Math va random modulidan foydalanib dasturlar tuzish amaliy ko'nikmalariga ega bo'lishlariga erishish

Masalaning qo'yilishi: Talaba variant bo'yicha berilgan masalani Python dasturlash tilida dasturini tuzishi va kerakli natija olishi lozim.

Ishni bajarish uchun namuna

Misol: 1-misol: x ning berilgan haqiyqiy qiymatida formula bo'yich u funksiya hisonblansin.

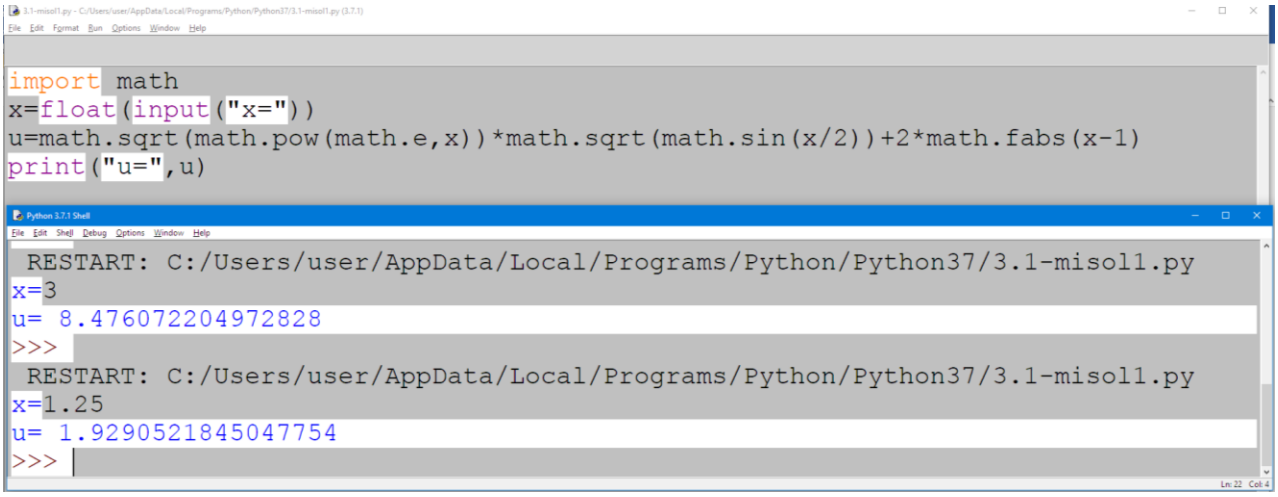
$$u = \sqrt{e^x}\sqrt{Sin(\frac{x}{2})} + 2\,|\,x-1\,|$$

Dastur kodi

5.1-misol.py fayli:

```
import math
x=float(input("x="))
u=math.sqrt(math.pow(math.e,x))*math.sqrt(math.sin(x/2))+2*math.fa
bs(x-1)
print("u=",u)
```

Dastur ishlashi natijasidan namunalar:



5-amaliy ish topshiriqlari

Berilgan x, y va z sonlari uchun formulaning natijasi topilsin (1-10 - misollar).

Berilgan x, y va z sonlarining berilgan qiymatlari uchun formulaning natijasi topilsin (11-32- misollar).

1.	x=14.26, y=-1.22, z=0.5×10^{-2}, $t=\frac{2\cos\left(x-\frac{\pi}{6}\right)}{0.5+\sin^2 y}\left(1+\frac{z^2}{3-z^2/5}\right)$ Natija: t=0.564849
2.	x=-4.5, y=$0.75\text{x}10^{-4}$, z=$0.845\text{x}10^{2}$, $u=\frac{\sqrt[3]{8+\lvert x-y\rvert^2+1}}{x^2+y^2+2}-e^{\lvert x-y\rvert}\left(tg^2 z+1\right)^x.$ Natija: u=-55.6848.
3.	x=-15.246, y=$4.642\text{x}10^{-2}$, z=$20.001\text{x}10^{2}$, $\alpha=\ln\left(y^{-\sqrt{\lvert x\rvert}}\right)\left(x-\frac{y}{2}\right)+\sin^2 arctg(z).$ Natija: α=-182.036
4.	x=0.1722, y=6.33, z=$3.25\text{x}10^{-4}$, $\gamma=5arctgx-\frac{1}{4}\arccos x\frac{x+3\lvert x-y\rvert+x^2}{\lvert x-y\rvert z+x^2}.$ Natija: γ=-172.025
5.	x=$1.825\text{x}10^{2}$, y=18.225, z=$-3.298\text{x}10^{-2}$, $\psi=\left\lvert x^{\frac{y}{x}}-\sqrt[3]{\frac{y}{x}}\right\rvert+(y-x)\frac{\cos y-\frac{z}{(y-x)}}{1+(y-x)^2}.$ Natija: ψ=1.2131
6.	x=6.251, y=0.827, z=25.001, $b=y^{\sqrt[3]{\lvert x\rvert}}+\cos^3 y\frac{\lvert x-y\rvert\left(1+\frac{\sin^2 z}{\sqrt{x+y}}\right)}{e^{\lvert x-y\rvert}+x/2}.$ Natija: b=0.7121
7.	X=17.421, y=$10.365\text{x}10^{-3}$, z=$0.828\text{x}10^{5}$, $f=\frac{\sqrt[4]{y+\sqrt[3]{x-1}}}{\lvert x-y\rvert(\sin^2 z+tgz)}.$

	Natija: f=0.33056
8.	X=2.444, y=0.869x10^{-2}, z=-0.13x10^{3}, $h=\frac{x^{y+1}+e^{y-1}}{1+x\lvert y-tgz\rvert}\left(1+\lvert y-x\rvert\right)+\frac{\lvert y-x\rvert^{2}}{2}-\frac{\lvert y-x\rvert^{3}}{3}.$ Natija: h=-0.49871
9.	X=1, y=1, z=3 $a=(1+y)\frac{x+y/(x^{2}+4)}{e^{-x-2}+1/(x^{2}+4)};$ $b=\frac{1+\cos(y-2)}{x^{4}/2+\sin^{2}z}.$ Natija: a=9.608184; b=2.962605
10.	X=3, y=4, z=5, $a=\frac{1+\sin^{2}(x+y)}{2+\lvert x+2x/(1+x^{2}y^{2})\rvert}+x;$ $b=\cos^{2}(arctg\frac{1}{z}).$ Natija: a=3.288716; b=0.9615385
11.	$\gamma=5arctgx-\frac{1}{4}\arccos x\frac{x+3\lvert x-y\rvert+x^{2}}{\lvert x-y\rvert z+x^{2}}.$
12.	$\psi=\left\lvert x^{\frac{y}{x}}-\sqrt[3]{\frac{y}{x}}\right\rvert+(y-x)\frac{\cos y-\frac{z}{(y-x)}}{1+(y-x)^{2}}.$
13.	$b=y^{\sqrt[3]{\lvert x\rvert}}+\cos^{3}y\frac{\lvert x-y\rvert\left(1+\frac{\sin^{2}z}{\sqrt{x+y}}\right)}{e^{\lvert x-y\rvert}+x/2}.$
14.	$f=\frac{\sqrt[4]{y+\sqrt[3]{x-1}}}{\lvert x-y\rvert(\sin^{2}z+tgz)}.$
15.	$h=\frac{x^{y+1}+e^{y-1}}{1+x\lvert y-tgz\rvert}\left(1+\lvert y-x\rvert\right)+\frac{\lvert y-x\rvert^{2}}{2}-\frac{\lvert y-x\rvert^{3}}{3}.$
16.	$u=\sqrt{e^{x}}\sqrt{\cos^{2}(\frac{x}{4})}+2\ln x$
17.	$u=\sqrt{tg^{2}(\frac{x}{5})}+2e^{x}$

18.	$y = \log_4 x + \sqrt{e^3} + \cos\frac{x}{2}$
19.	$y = \sec x + \sqrt[4]{e^3} + \frac{x}{2}$
20.	$u = \sqrt{\ln x} + 2\,\lvert e^{x+2} - 5x \rvert$
21.	$u = \sqrt{\ln x} + 2\,\lvert e^{x+2} - 15 \rvert$
22.	$y = 2{,}5\sqrt{x^4} + Sin\frac{4}{x}$
23.	$y = e^{\sin x^2} + \frac{x}{2e^{x-1}}$
24.	$u = \sqrt{e^x}\sqrt{Sin(\frac{x}{2})} + 2\,\lvert x-1 \rvert$
25.	$u = \sqrt{\ln x} + 2\,\lvert e^{x+2} - 5x \rvert$
26.	$u = \sqrt{Sin\,\lvert x-8 \rvert} + e^{x+5} + x^4$
27.	$u = \sqrt{\lvert 18-x \rvert} + e^{x+2} - 5x$
28.	$u = \sqrt{Sin\,\lvert x-8 \rvert} + e^{x+5} + x^4$
29.	$a = \frac{\lvert 4-13x \rvert + e^x}{1+\ln x}$
30.	$u = \sqrt{Cos^2(\frac{x}{\lvert 2-x \rvert})} + 2e$
31.	$u = \sqrt{\ln x}\sqrt{Sin(\frac{x}{2})} + 2\,\lvert e^x - 1 \rvert$
32.	$u = \sqrt{e^x}\sqrt{Sin(\frac{x}{2})} + 2\,\lvert x-1 \rvert$

6-amaliy *ish*.

Satrlar va ular ustida amallar

Ishdan maqsad: Python dasturlash tilida satr tushunchasi, ular bian ishlovchi funksiyalar bilan tanishtirish va satrlar ustida turli amallar bajarish ko'nikmalarini shakllantirish. Satrlar bilan ishlashga doir maslalaalr uchun dastur yozish amaliy malakalarni paydo qilish.

Masalaning qo'yilishi: Talaba variant bo'yicha berilgan masalani Python dasturlash tilida dasturini tuzishi va kerakli natija olishi lozim.

Ishni bajarish uchun namuna

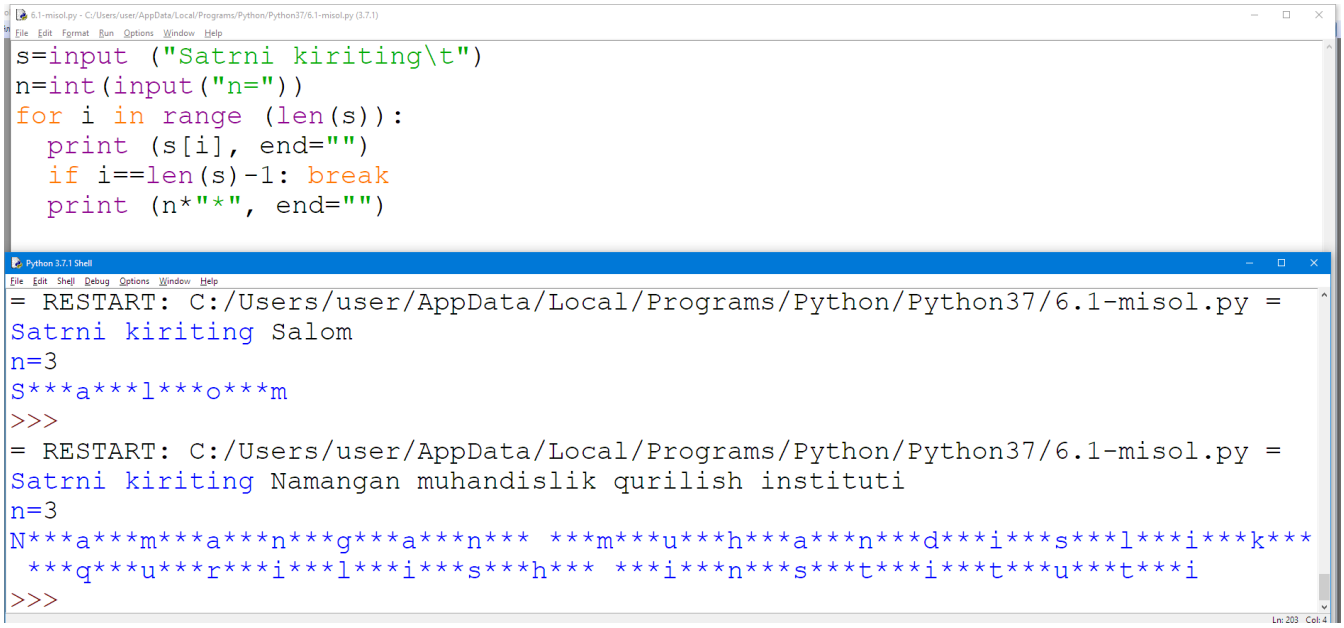
Misol: Bo'sh bo'lmagan satr va *n(n>0)* butun soni berilgan. Satrdagi belgilar orasiga *n* tadan "*" qo`yib, satr chop etilsin.

Dastur kodi:

dastur.cpp fayli:

```
s=input ("Satrni kiriting\t")
n=int(input("n="))
for i in range (len(s)):
  print (s[i], end="")
  if i==len(s)-1: break
  print (n*"*", end="")
```

Dastur ishlashi natijasi:



6.amaliy ish topshiriq variantlari

1.	Satr berilgan. Undagi raqamlar soni hisoblansin.
2.	Satr berilgan. Uning belgilari teskari tartibda chop etilsin (telefon -> nofelet).
3.	Bo‘sh bo‘lmagan satr berilgan. Satrda joylashgan belgilarning orasiga bittadan bo‘sh joy qo‘yishdan hosil bo’lgan satr chop etilsin.
4.	Satr berilgan. Undagi lotin alfavitining bosh harflari soni hisoblansin.
5.	Satr berilgan. Satrga kirmagan barcha lotin harflarining soni hisoblansin.
6.	Satr berilgan. Berilgan satrdagi barcha bosh harflar kichik harflarga aylantirilsin.
7.	Satr berilgan. Berilgan satrdagi barcha kichik harflar bosh harflarga almashtirilsin.
8.	Satr berilgan. Berilgan satrdagi barcha bosh harflar kichik harflarga, kichik harflar esa bosh harflarga almashtirilsin.
9.	Satr berilgan. Agar satr butun sondan iborat bo‘lsa *1*, haqiqiy sondan iborat bo‘lsa *2*, satrni son ko‘rinishiga o‘tkazib bo‘lmasa *0* chop etilsin.
10.	Butun musbat son berilgan. Bu sonni tasvirlovchi raqamlardan iborat belgilar o‘ngdan chapga qaragan tartibda chop etilsin.
11.	Butun musbat sonni tasvirlovchi satr berilgan. Bu sonning raqamlari yig‘indisi hisoblansin.
12.	“*<raqam1>*±*<raqam2>*±...±*<raqamN>*” ushbu arifmetik ifodani tasvirlovchi satr berilgan. Satrdagi “±”belgilar o‘rniga, “+” yoki “-” amali qo’yilsin va ifodaning qiymati chiqarilsin.
13.	Satrda butun musbat sonning ikkilik sanoq sistemasidagi ko‘rinishi tasvirlangan. Bu sonning 10 lik sanoq sistemasidagi ko‘rinishi chop etilsin.
14.	Satrda butun musbat sonning 10 liksanoq sistemasidagi ko‘rinishi tasvirlangan. Bu sonning 2 lik sanoq sistemasidagi ko‘rinishi chop etilsin.
15.	*n(n>0)* butun soni va satr berilgan. *n* uzunlikka teng bo‘lgan satr quyidagi ko‘rinishda aniqlanadi: satr uzunligi *n* dan katta bo‘lsa, uning o`ng tomonidan ortiqcha belgilar olib tashlansin, agar satr uzunligi *n* dan kichik bo‘lsa, uning o`ng tomoniga nuqtalar qo‘shilsin. Hosil qilingan satr chop etilsin.
16.	Butun musbat n_1, n_2 sonlar va 2 ta satr berilgan. Bu satrlardan foydalanib yangi satr hosil qilinsin: satrning dastlabki n_1 ta belgi ikkinchi satrning bosh qismidan, oxirgi n_2 ta belgi birinchi satrning oxirgi qismidan (chapdan o’ngga qarab) iborat bo‘lsin.
17.	Satr va ixtiyoriy belgi berilgan. Agar satrda berilgan belgi uchrasa, u ikkilantirilib satr chop etilsin.
18.	Ixtiyoriy belgi va 2 ta satr berilgan. Ikkinchi satrda berilgan belgi uchrasa uning oldiga birinchi satr joylashtirilsin.
19.	Ixtiyoriy belgi va 2 ta satr berilgan. Birinchi satrda belgi uchrasa, shu belgi ortidan ikkinchi satr joylashtirilsin.
20.	2 ta satr berilgan. Agar birinchi satr ikkinchi satrda mavjud bo‘lsa *1 (rost)* aks holda *0 (yolg’on)* qiymat chiqarilsin.
21.	2 ta satr berilgan. Ikkinchi satrda birinchi satrning necha marta uchrashi aniqlansin.
22.	2 ta satr berilgan. Ikkinchi satrdan birinchi satr bilan ustma-ust tushuvchi 1-qism satr o‘chirilsin. Agar ikkinchi satrda birinchi satr topilmasa satr o‘zgarishsiz chop etilsin.
23.	2 ta satr berilgan. Ikkinchi satrdan birinchi satr bilan ustma-ust tushuvchi oxirgi

	qism satr o‘chirilsin. Agar *s* satrda *Birinchi* satr topilmasa *s* satr o‘zgarishsiz chop etilsin.
24.	2 ta satr berilgan. Ikkinchi satrdan birinchi satr bilan ustma-ust tushuvchi barcha qism satrlar o‘chirilsin. Agar ikkinchi satrda birinchi satr topilmasa ikkinchi satr o‘zgarishsiz chop etilsin.
25.	3 ta satr berilgan. Birinchi satrdagi dastlabki uchragan ikkinchi qism satr uchinchi qism satr bilan almashtirilsin. Agar birinchi satrda ikkinchi satr topilmasa birinchi satr o‘zgarishsiz chop etilsin.
26.	3 ta satr berilgan. Birinchi satrdagi oxirgi uchragan ikkinchi qism satr uchinchi qism satr bilan almashtirilsin. Agar birinchi satrda ikkinchi satr topilmasa birinchi satr o‘zgarishsiz chop etilsin.
27.	3 ta satr berilgan. Birinchi satrda uchragan barcha ikkinchi qism satrlaruchinchi qism satr bilan almashtirilsin. Agar birinchi satrda ikkinchi satr topilmasa birinchi satr o‘zgarishsiz chop etilsin.
28.	Kamida 2 ta bo‘sh joyga ega satr berilgan. Berilgan satrdagi 1- va 2- bo‘sh joylar orasida joylashgan qism satr chiqarilsin.
29.	Kamida 2 ta bo‘sh joyga ega satr berilgan. Berilgan satrdagi 1- va oxirgi bo‘sh joylar orasida joylashgan qism satr chiqarilsin.

Foydalanilgan asosiy darsliklar va o'quv qo'llanmalar ro'yxati

1. Н. А. Прохоренок, В. А. Дронов. “Python 3 и PyQt 5. Разработка приложений”. БХВ-Петербург, 2016. — 832 с.: ил.
2. Д.Ю.Федоров. “Основы программирования на примере языка Python” Учебное пособие. Санкт-Петербург. 2-е изд., перераб. и доп. — М.: Издательство Юрайт, 2019. — 161 с.
3. Любанович Билл. “Простой Python. Современный стиль программирования”. — СПб.: Питер, 2016. — 480 с.: ил. — (Серия «Бестселлеры O’Reilly»).
4. Васильев А. Н. “Python на примерах. Практический курс по программированию”. - СПб.: Наука и Техника, 2016. - 432 с.: ил.
5. Рашка С. “Python и машинное обучение”/ пер.с англ. А. В. Логунова. - М.: ДМК Пресс, 2017. - 418 с.: ил.
6. Шолле Франсуа. “Глубокое обучение на Python”. — СПб.: Питер, 2018. — 400 с.: ил.
7. Мэтиз Эрик. “Изучаем Python. Программирование игр, визуализация данных, веб-приложения”. СПб.: Питер, 2017. — 496 с.: ил.
8. JOHNE. GRAYSON. “Python and TkinterProgramming”. MANNING Greenwich (74° w. long.) ©2000 by Manning Publications Co.

Internet ma'lumotlarini olish mumkin bo'lgan saytlar:

www.ziyonet.uz;

www.google.com;

www.python.org;

www.qt.com;

www.intuit.ru.

ADABIYOTLAR RO’YXATI

Foydalanilgan asosiy darsliklar va o'quv qo'llanmalar ro'yxati

Asoiy darsliklar va o'quv qo'llanmalar

9. Н. А. Прохоренок, В. А. Дронов. “Python 3 и PyQt 5. Разработка приложений”. БХВ-Петербург, 2016. — 832 с.: ил.
10. Д.Ю.Федоров. “Основы программирования на примере языка Python” Учебное пособие. Санкт-Петербург. 2-е изд., перераб. и доп. — М.: Издательство Юрайт, 2019. — 161 с.
11. Любанович Билл. “Простой Python. Современный стиль программирования”. — СПб.: Питер, 2016. — 480 с.: ил. — (Серия «Бестселлеры O’Reilly»).
12. Васильев А. Н. “Python на примерах. Практический курс по программированию”. - СПб.: Наука и Техника, 2016. - 432 с.: ил.
13. Рашка С. “Python и машинное обучение”/ пер.с англ. А. В. Логунова. - М.: ДМК Пресс, 2017. - 418 с.: ил.
14. Шолле Франсуа. “Глубокое обучение на Python”. — СПб.: Питер, 2018. — 400 с.: ил.
15. Мэтиз Эрик. “Изучаем Python. Программирование игр, визуализация данных, веб-приложения”. СПб.: Питер, 2017. — 496 с.: ил.
16. JOHNE. GRAYSON. “Python and TkinterProgramming”. MANNING Greenwich (74° w. long.) ©2000 by Manning Publications Co.

Qo'shimcha adabiyotlar

Internet ma'lumotlarini olish mumkin bo'lgan saytlar:

www.ziyonet.uz;

www.google.com;

www.python.org;

www.qt.com;

www.intuit.ru.

MUSTAQIL TA`LIM MASHG'ULOTLARI

1. Pythonda na'lumotlar bazasi bilan ishash
2. PyQT da ishlash
3. Pythonda Tkinter moduli bilan ishlash
4. Pythonga grafika bilan ishlash
5. Pythonda modullarni o'rnatish
6. Pythonda web sayt uchun dastur yaratish
7. Pythonda tarmoq bilan ishlash
8. Python tilida mobil dasturlash
9. Python freymworklari
10. Django freymworki bilan ishlash
11. Flask freymworki bilan ishlash
12. Python da o'yinlarni dasturlash
13. Pygame modulini o'rnatish
14. Python dasturlash tilida ma'lumotlarni vizuallashtirish
15. Matplotlib kutubxonasi bilan tanishish
16. Python dasturlash tilada API lar bilan ishlash

GLOSSARIY

Python

bu o'rganishga oson va shu bilan birga imkoniyatlari yuqori bo'lgan oz sonlik zamonaviy dasturlash tilli. **Python** yuqori darajadagi ma'lumotlar strukturasi va oddiy lekin samarador ob`yektga yo'naltirilgan dasturlash uslublarini taqdim etadi.

Izoh
dastur matnini tushintirish uchun foydalaniladi va dasturni izohlaydi

print()
obyektni satrga o’tkazadi va uni stdout standart chiqarishga uzatish funksiyasi

input ()
ma’lumotlarni kiritish uchun stdin standartidan ma’lumot oluvchi funksiya

int
butun son tipi

float
haqiyqiy son tipi

list
ro’yxat ko’rinishdagi tur

True
Mantiqiy ifodaning rost (1) qiymatli natijasi

False
Mantiqiy ifodaning yolg’on (0) qiymatli natijasi

in
ketma-ketlikka tegishlilikka tekshirish operatori

not in
ketma-ketlikka tegishli emaslikka tekshirish operatori

if...else
mantiqiy ifoda natijasi bog’liq holda dastur qismining bajarilishi yoki bajarilmasligini

ta’minlaydigan tarmoqlanish operatori

for
takrorlanishlarni hisoblash va ketma-ketliklar bilan ishlash uchun foydalaniladigan sikl operatori

while
instruksiyalar mantiqiy ifoda rost bo’lganda bajariladigan sikl operatori

continue
sikldagi barcha instruksiyalar bajarilmay, siklning navbatdagi iterasiyasiga o’tish imkonini beradigan operator

break
sikl bajarilishini zudlik bilan to’xtatish imkonini beradigan operatori

Internet
–dunyodagi turli xil kompyuter tarmoqlari bilan aloqa bog`lab turishni ta`minlovchi texnik vositalar, programma ta`minoti, standart va kelishuvlar yig`indisi.

HTTP (Hyper Text Transfer Protocol)
–bu internet protokoli hisoblanib, uning yordamida bir formatdagi ikki kompyuter o`zaro bog`lanib, muloqot olib borish imkoniyatiga ega bo`ladi.

PPP (Post office protocol)
–oddiy modem liniyalarini internetga kirishda ishlatiladigan kanal darajasidagi protokol,(Analog Slip).

Telnet
–uzoqda turib tarmoqdagi istalgan kompyuterni boshqarish rejimi.

Usenet (Usenet Wewsq roupe)
– tarmoq yangiliklari va tarmoqdagi elektron elonlar doskasini olish.

SLIP (Serial Line Internet Protocol)
–oddiy modem liniyalarini internetga kirishida ishlatiladigan jahon darajasidagi protokol.

Elektron tarjimon
–o`ziga yuborilgan matnni bir tildan ikkinchi tilga tarjima qilib beradi.

UUCP

–bir Unix-xoctdan boshqasiga axborotlarni nusxalash protokoli. Ko`plab pochta almashuv sistemalari shu protokolga asoslanib tuzilgan.

PAP (Password authentication protocol)
–serverga ulovchi parollar sistemasi.

NNTP (Net News Transfor Protocol)
–tarmoq yangiliklarini uzatuvchi protokol.

Servis markazi
–internetga ulangan ko`plab kompyuter sistemalarini quvvatlovchi markaz.

Clarinet
–foydalanish uchun ko`pchilik servis markazlari bilan imzolanadigan katta yangiliklar xizmati.

FTP (Fili Transfer Protocol)
–fayllarni uzatuv protokoli; kompyuterlararo axborot almashuvining standart usuli.

Veronica (Very Easy Rodent–Oriented Vetwide Index to Computer Archives)
–kalit so`zlar bo`yicha internet tarmog`ining ommaviy arxivida axborotlarni qidirish sistemasi.

WWW (World Wide Web)
–hujjatlararo gipermatn aloqa bog`lash qobiliyatiga ega bo`lgan tarqoq ma`lumotlar bazasi sistemasi.

Netscape Communication
–bu dunyodagi eng ommabop va eng ko`p ishlatiladigan brauzer hisoblanadi.

Linked files
–sayt asosiy sahifasi bilan bog`langan fayllar miqdori.

All files
–sayt barcha fayllarining miqdori va umumiy o`lchami.

Pictures
–sayt grafik fayllarining miqdori va umumiy hajmi.

Slow pages
–30 sekundan ortiq yuklanadigan HTML–fayllar miqdori («sust sahifalar.

Whois
–Internet tarmog`ining adres kitobi.

WAIS (Wide Arle Information Service)
–kalit so`zlar bo`yicha internet tarmog`ining ma`lumotlar bazasida kuchli axborotlar qidiruv sistemasi.

Gopher
–Internet zaxira va imkoniyatlarni qidirish, ularga bog`lanish va ulardan foydalanish uchun mo`ljallangan interaktiv obolochka (qobig`) foydalanuvchi bilan interfeys menyu sistemasi orqali olib boriladi.

Telnet
–uzoqdan kirish. abonentga Internet tarmog`idagi istalgan EHMda ishlash imkonini beradi.

LAN (local area NetWork)
–geografik bir joydagi lokal tarmoq.

WAN (wide Area NetWork)
–katta hududda joylashgan global tarmoq.

NSFNET–IP
–texnologiyasida tashkil qilingan milliy − ilmiy fondning xususiy tarmog`i.

IP (Internet Protocol)
–tarmoqdagi paketlarni marshrutlashni ta`minlovchi tarmoqlararo o`zaro harakat protokoli.

TCP (Transmission Control Protocol)
–tarmoqdagi axborot uzatuvini nazorat qilib turuvchi protokol; katta hajmdagi axborotlarning jo`natish muammolarini xal qiladi.

DOMEN (DNS–DOMAIN NAME SYSTEM)
–normalarning domen sistemasi; internet tarmog`idagi kompyuter nomlarini IP-adreslariga o`tkazib beruvchi ma`lumotlar bazasining tarmoq sistemasi.

FRAME
–jadval ramkasining qaysidir tashqi qismi chizmasini aniqlaydi.

CELLPADING

–yacheyka ichidagi narsalar bilan ramka orasidagi bo`sh oraliq razmerini piksellarda beradi.

CELLSPACING
–jadval yacheykalari orasidagi bo`sh oraliqni piksellarda ko`rsatadi.

Void
–tashqi ramkani butunlay yo`qotadi.

Above
–tashqi ramkaning faqat yuqori chizig`i chiziladi.

Below
–tashqi ramkaning faqat pastki chizig`i chiziladi.

RULES
–jadval ramkasini ichki qismining qanday chizilishini ko`rsatadi.

None
–hech qanday ichki ramkalar bo`lmaydi.

Rows
–faqat gorizontal chiziqlar chiziladi, (satrlar orasidagi).

Cols
–faqat vertikal chiziqlar chiziladi, (ustunlar orasidagi).

Ale
–barcha ichki ramkalar chiziladi.

TEXT
–matnli maydon

RADIO
–selektor tugmasi

CHECKBOX
–nazorat indikatori

SUBMIT

–anketani jo`natish tugmasi

RESET
–anketani tozalash tugmasi.

Page
–loyihalash rejimi.

Folders
–sayt strukturasini aks ettirish rejimi.

Reports
–sayt to`g`risidagi zaruriy axborotlarni aks ettirish rejimi.

Navigation
–sayt navigasiyasi rejimi.

Hyperlinks
–ichki va tashqi aloqalar strukturasi rejimi.

Task
–topshiriq va masalalarni boshqarish rejimi.

NAME
–qatorida fayl yoki papka nomi ko`rsatiladi.

Title
–sahifa sarlavhasi yoki sayt qolgan elementlari nomini ko`rsatadi.

Modified Datea
–soni va qaysidir saytning oxirgi o`zgarish vaqtini ko`rsatadi.

Modified
–oxirgi fayl nomi.

All files
–sayt barcha fayllarining miqdori va umumiy o`lchami.

Pictures
–sayt grafik fayllarining miqdori va umumiy hajmi.

Linked files
–sayt asosiy sahifasi bilan bog`langan fayllar miqdori.

Slow pages
–30 sekundan ortiq yuklanadigan HTML–fayllar miqdori («sust sahifalar»).

Marshrutizator-(roater)
–tarmoq paketlarini marshrutlash bilan shug`ullanadigan kompyuter tarmog`i, ya`ni paketlarning tarmoq bo`ylab eng qisqa harakat marshrutlari tanlab beriladi.

Protokol
–ikki va undan ortiq mustaqil qurilma yoki prosessorlar o`rtasida forma va proseduralarga reklama qiluvchi qoida va kelishuvlar yig`indisi.

Resurs
–foydalanuvchi ixtiyoriga berilish imkoniyati bor bo`lgan sistemaning mantiqiy yoki fizikaviy qismi.

Server-kompyuter
–boshqalarga o`z xizmatini tavsiya qiluvchi tarmoq kompyuteri, ya`ni foydalanuvchilarning talablari, (savollari) bilan shug`ullanadi.

Server-programma
–bitta kompyuter xizmatini boshqa kompyuterga taqdim etish imkonini yaratuvchi tarmoq kompyuter dasturi.

Uzel
–tarmoqning asosiy vazifalarini bajaruvchi tarmoq kompyuteri.

Xost
–tarmoq vazifalaridan tashqari foydalanuvchilarning topshiriqlarini, (programmalar, hisoblash ishlari va b.q.) bajaruvchi tarmoqning ishchi mashinasi ya`ni, bosh EHM.

Shlyuz
–tarmoqni har xal kompyuter sistemalari bilan bog`lab turuvchi o`zaro harakatdagi tarmoqlararo vosita.

PPP (Post office protocol)
–oddiy modem liniyalarini internetga kirishda ishlatiladigan kanal darajasidagi protokol, (Analog Slip).

SLIP (Serial Line Internet Protocol)
–oddiy modem liniyalarini internetga kirishda ishlatiladigan jaxon darajasidagi protokol.

UUCP
–bir Unix-xoctdan boshqasiga axborotlarni nusxalash protokoli. Ko`plab pochta almashuv sistemalari shu protokolga asoslanib tuzilgan.

PAP (Password authentication protocol)
–serverga ulovchi parollar sistemasi.

Clarinet
–foydalanish uchun ko`pchilik servis markazlari bilan imzolanadigan katta yangiliklar xizmati.

FTP (Fili Transfer Protocol)
–foydali uzatuvlar protokoli; kompyuterlararo Axborot almashuvining standart usuli.

E-mail
–Internet ning istagan abonenti bilan pochta xabarlarini almashtirish va xabarlarni uzatish servisi.

Fayl–servis
–boshqa kompyuterga o`z fayliga kirish imkonini beruvchi kompyuter.

Klient
–server zaxiralaridan foydalanuvchi kompyuter yoki programma.

Programma-server
–o`z mijozidan buyurtma qabul qiladi, unga ishlov beradi va mijozga kerakli axborotni qaytaradi.

Portlar
–har xil ilova va qo`shimchalar bilan aloqani qilovchi server programma nomer, (yoki port nomeri).

POP (Post Office Protocol)
–protokol «pochtali ofis». Xost va abonent o`rtasida pochta almashuvi uchun ishlatiladi. Abonent talabi bo`yicha ham almashuv ishlari bajariladi.

Xost-kompyuter
–internetga mustaqil ravishda ulanish xuquqiga ega bo`lgan kompyuterlar.

SMTP (Simple Mail Transfer Protocol)
–xabarlarni jo`natish uchun ishlatiladigan oddiy pochta uzatuv protokoli.

ASCII (American Standart Cade for Information inferchange)
–matnli axborotlarni almashtirish uchun ishlatiladigan amerika standart kodi.

ILOVALAR

Fan dasturi

O`ZBEKISTON RESPUBLIKASI
OLIY VA O`RTA MAXSUS TA`LIM VAZIRLIGI
NAMANGAN MUHANDISLIK-QURILISHINSTITUTI

NamMQI o'quv-uslubiy bo'limi
Ro'yhatga olindi:
№ ____
2020y. «__» 02
"29" 05 20 20y.

"TASDIQLAYMAN"
Namangan muhandislik – qurilish instituti o'quv-uslubiy kengash raisi
«29» 05 2020 y.

PYTHON DASTURLASH TILI
fanining

O`QUV DASTURI

Bilim sohasi: 300000 - Ishlab chiqarish – texnik soha

Ta'lim sohasi: 330 000 - Kompyuter texnologiyalari va informatika

Ta'lim yo'nalishi: 5330200 – Informatika va axborot texnologiyalari (tarmoqlar va sohalar bo'yicha)

Namangan – 2020 y.

Fan dasturi Namangan muhandislik qurilish instituti Ilmiy kengashining 29 08 2020 yildagi № 1 - sonli bayoni bilan tasdiqlangan 5330200 - Informatika va axborot texnologiyalari (sanoat ishlab chiqarishida) ta'lim yo'nalishining ishchi o'quv rejasiga asosan ishlab chiqildi.

Fan dasturi Namangan muhandislik-qurilish institutining Informatika va AT kafedrasida ishlab chiqildi.

Tuzuvchilar:

O.Jakbarov. – NamMQI, Informatika va AT kafedrasi dotsenti

U.Goyipov – NamMQI, Informatika va AT kafedrasio'qituvchisi

Taqrizchilar:

Imomov A. – Namangan Davlat Univesiteti "Amaliy matematika va axborot texnologiyalari" kafedrasi dotsenti, f-m.f.n.

Isomiddinov A –Namangan muhandislik-qurilish instituti Texnik tizimlarda axborot texnologiyalari kafedrasi mudiri, t.f.n.

Fan dasturi Namangan muhandislik-qurilish institutiIlmiy-uslubiy Kengashida ko'rib chiqilgan va tavsiya qilingan(201_ yil "29" 08 dagi 1 - sonli bayonnoma).

Kirish

Hozirgi kunda Davlatimiz siyosatining ustuvor yo'nalishlaridan biri bu – zamonaviy axborot texnologiyalarini ishlab chiqarishga hamda jamiyatimizning deyarli barcha jabhalariga keng tatbiq qilish, kompyuterlashtirish va avtomatlashtirish tizimlaridan yanada samarali foydalanishdir. Shu ma'noda, kadrlar tayyorlash tizimini takomillashtirish masalasi dolzarb vazifalardan biridir.

Shuning uchun ham, o'quv dasturlarini zamon talablari asosida takomillashtirib borish o'ta zarur masalalardan biri bo'lib hisoblanadi.

Ayniqsa, zamonaviy axborot texnologiyalaridan foydalanish, ularni boshqarish, dasturlashtirish kabi eng zamonaviy ilmlarni o'zlartirish hozirgi kun talabalari uchun muhim vazifadir.

Ushbu fan dasturi dasturlash tillari, xususan Python dasturlash tilini talabalarga mukammal o'rgatish maqsadida zamonaviy talablar asosida ishlab chiqildi.

O'quv fanining maqsadi va vazifalari

Ushbu fanning maqsadi zamonaviy texnika va texnologiyalarni ishlatish, ularga dasturiy ilovalar yaratish, yosh avlodga milliy istiqlol g'oyasini yetkazish uchun jaxon standartlarini darajasida fan texnika hamda ilg'or tajriba va texnologiyalarning eng so'nggi yutuqlaridan xabardor bo'lgan raqobatbardosh, o'z sohasining ham ilmiy, ham amaliy bilgan mutaxassislar tayyorlashdir.

Fan bo'yicha talabalarning bilimiga, ko'nikma va malakasiga qo'yiladigan talablar

Python dasturlash tili fani bo'yicha talabalar bilimi va ko'nikmalariga qo'yiladigan talablar quyidagilardan iborat:

- dasturlash tillarida chiziqli algoritmlarni va dasturlarini tuzishni biladi;
- tarmoqlanuvchi jarayonlarni algoritmlarni va dasturlarini tuzishni biladi;
- takrorlanuvchi jarayonlarni algoritmlarni va dasturlarini tuzishni biladi;
- massivlarga doir algoritm va dasturlarni tuzishni biladi va qo'llay oladi;
- xotira modellaridan foydalana oladi.

Fanning o'quv rejadagi boshqa fanlar bilan o'zaro bog'liqligi va uslubiy jihatdan uzviy ketma-ketligi

«Python dasturlash tili» fani bir qator fanlar bilan uzviy bog'liqdir. Shu jumladan «Dasturlash tillari(C++)», «Ob'ektga yo'naltirilgan dasturlash tillari», «Dasturlash texnologiyasi» va «Tizimli dasturiy ta'minot» fanlarini bilish juda muhimdir. Fanni o'rganish natijasida ega bo'lingan bilimlar ixtisoslik fanlarini muvaffaqqiyatli o'zlashtirishda va bitiruv malakaviy ishini bajarishda o'z ifodasini topadi.

Fanning ishlab chiqarishdagi o'rni

Hozirgi kunda kompyuter va kompyuter tarmoqlari yordamida talabalarga ta'lim berish borasida davlatimiz tomonidan muayyan ishlar olib borilmoqda. O'qitishning kompyuterlashgan, axborot va boshqa zamonaviy texnologik vositalardan (turli animatsiyalar, virtual laboratoriyalar va boshqalar) foydalanish hozirgi zamon talabidir.

Tajriba mashg'ulotlarini olib borishda, «Python dasturlash tili» fanidan tayyorlangan elektron darslik, elektron qo'llanmalar va Python dasturlash tili dasturiy ta'minotidan foydalaniladi.

Bundan tashqari, amaliy ishlarni bajarishda Pythondan foydalanib, dasturlar tuzish, har bir topshiriqni amaliy dasturlar bog'lami ko'rinishiga keltirish va kelajakda foydalanish uchun yo'riqnomalar tayyorlanadi.

Fanni o'qitishda zamonaviy axborot va pedagogik texnologiyalar

Ta'lim tizimini isloh qilish – bu ta'limning mazmunini yangilash, o'qitishning zamonaviy uslublarini tadbiq qilish, o'quv jarayonida yangi ta'lim vositalaridan foydalanishni yo'lga qo'yish demakdir. SHu bois bugungi kunda ilg'or pedagogik texnologiyalarni qo'llash zaruriyati vujudga keldi. Biz orzu qilayotgan ilg'or pedagogik texnologiyalar rivojlantiruvchi ta'lim tamoyillariga ega bo'lib, birinchi navbatda talaba shaxsiga yo'naltirilgan bo'lmog'i lozim.

Ilg'or pedagogik texnologiyalar – bu Kadrlar tayyorlash milliy dasturida ko'zda tutilgan vazifalarni bajarishda qo'llanadigan asosiy pedagogik usuldir. Bu usulni qo'llash uchun quyidagi muammolarni hal qilish lozim:

- talabalarni mustaqil ta'lim olishlari uchun alohida sharoitlar yaratish;
- dars mashg'ulotlarida talabaning faolligini orttirish, buning uchun ko'proq interfaol uslublardan foydalanish;
- ta'lim jarayonida ta'limiy va tarbiyaviy maqsadlarni amalga oshirishda tarixiy an'analarga, halqning boy ma'naviy me'rosiga tayanib umuminsoniy qadriyatlarni hisobga olgan holda ish yuritish;
- ta'lim jarayonida noan'anaviy dars shakllaridan (musobaqa, konkurs, amaliy o'yinlar va h.k.) samarali foydalanish;
- dars jarayonida talabani o'qitish emas, ko'proq o'qishga bilim olishga o'rgatish;
- reproduktiv bilim berish usullaridan evristik uslublarga o'tish, olinayotgan natijalarni o'quvchining o'zi «ixtiro» qilishiga erishish.

Asosiy qism

Fanning nazariy mashg'ulotlari mazmuni

Python dasturlash tili bilan tanishish. Python tarixi va imkoniyatlari. Pythonni o’rnatish. Dastur tuzilishi. Izoxlar. Dastur natijasini chop etish. Ma’lumotlarni kiritish.

O’zgaruvchilar. O’zgaruvchini nomlash. Ma’lumot turlari. O’zgaruvchiqa qiymat o’zlashtirish. Ma’lumot tipini aniqlash. Ma’lumot tipini o’zgartirish. O’zgaruvchini o’chirish.

Operatorlar. Matematik operatorlar. Munosabat operatorlari. Ketma-ketliklar bilan ishlash operatorlari. O’zlashtirish operatorlari. Operatorlarni bajarish ketma-ketligi.

Shartli operatorlar. Taqqoslash operatorlari.if...else operatori.

Sikl operatorlari. For sikli. range() va enumerate() funksiyalari. While sikli. continue operatori. break operatori.

Sonlar. Sonlar bilan ishlashning tashqi funksiya va metodlari.*math* moduli. Matematik funksiyalar. *random* moduli. Tasodifiy son generasiyasi.

Qatorlar va ular ustida amalllar. Qator yaratish. Maxusu belgilar. Qatorlar bilan ishlash amallari. Qatorlarni formatlash. format() metodi. Qatorlar bilan ishlash metod va funksiyalari. Localni sozlash. Belgila registrini o'zgartirish. Belgilar bilan ishlash funksiyalari. Qatorni qidirish va almashtirish. Qator turlarini tekshirish.

Muntazam ifodalar. Muntazam ifoda sintaksisi. Shablonga birinchi moslikni qidirish. Shablonga barcha moslikni qidirish. Qatorni almashtirish.

Ro’yxatlar, kortejlar, to’palamlar va diapazonlar. Ro’yxat yaratish. Ro’yxatlar ustida amallar. Ko’p o’lchamli ro’yxatlar. Ro’yxat elemantlarini saralash. Ro’yxat generatorlari. map(), zip(), filter() va reduce() funksiyalari. Ro’yaxatga elementlar qo’shish va o’chirish. Ro’yxatni tezkarilash va aralshtirish. Tasodifiy elementni tanlash. Ro’yxatni saralash. Ro’yxatni sonlar bilan to’ldirish. Ro’yxatni satrga o’tkazish. Kortejlar. To’plamlar. Diapazonlar. Itertools moduli. Noaniq qiymatlar generatsiyasi. Qiymatlar kombinasiyasi generasiyasi. Elementlar izchilligi filtrasiyasi.

Lug’atlar. Lug’at yaratish. Lug’atlar ustida amallar. Lug’at elementlarini saralash. Lug’atlar bilan ishlash metodlari. Lug’atlar generatori.

Sana va vaqt bilan ishlash. Joriy sana va vaqtni chop etish. Sana va vaqt formati. «Uxlash» skripti. Datetime moduli. Sana va vaqt monipulyatsiyasi. Calendar moduli. Calendarni ko’rsatish.

Foydalanuvchi funksiyalari. Funksiyai aniqlanishi va uni chaqirish. Fiunksiyani aniqlanishi joylashuvi. Anonim funksiyalar. Funksiya-generatorlar. Rekursiya. Faktorialni hisoblash. Global va local o’zgaruvchilar. Ilova funksiyalar. Funksiya annotasiyasi.

Pythonda modullar va paketlar. *import* ko’rsatmasi. *from* ko’rsatmasi. Moduldan qidirish yo’li. Modullarini qayta yuklash. Paketlar.

Fayl va kataloglar bilan ishlash. Faylni ochish. Fayllarbilan ishlash metodlari. Os moduli yordamida fayllarga kirish. StringIO va BytesIO sinfi. Fayl va kataloglarga kirish huquqi. Faylarni monipulyasiyalsh funksiyasi. Fayl yoki kataloglaraga yo’lni o’zgartirish.

Kiritish/chiqarishni qayata yo'naltirish. Fayllarda obyektni saqlash. Kataloglar bilan ishlash funksiyalari.

Pythonda ba’zi algoritmlar.

Pythonda istisnolar va xatolarni qayta ishlash.

Pythonada obyektga yo’naltirilgan dasturlash. Obyektga yo’naltirilgan yondashuv asoslari. Pythonda merosxo’rlik. Pythonda merosxo’rlik iyerarxiyasi.

Grafik interfeysli ilovalarni qayata ishlash. tkinter moduli bilan ishlash asoslari. Tkinter modulida “model-ko’rinish-nazorat” shabloni.

Python da kliyent-serverli dastirlash. Arduino kontrollerini dasturlash.

Amaliy mashg’ulotlarini tashkil etish bo’yicha ko’rsatma va tavsiyalar

Amaliy mashg’ulotlar o’tkazilishidan maqsad dasturlash bo’yicha olingan nazariy bilimlarni amalda mustahkamlash va turli toifadagi masalalarni yechishga qo’llashdan iborat. Amaliy mashg’ulotlarni bir qismi auditoriyada doskada yechilishi bilan o’tkazilsa, uning katta qismi bevosita kompyuterda amalga oshirilishi kerak.

Amaliy mashg’ulotlarining taxminiy tavsiya etiladigan mavzulari:

1. Python dasturlash tili bilan tanishish
2. O’zgaruvchilar
3. Operatorlar
4. Shartli operatorlar
5. Sikl operatorlari
6. Sonlar
7. Qatorlar va ular ustida amalllar
8. Muntazam ifodalar
9. Ro’yxatlar, kortejlar, to’palamlar va diapazonlar
10. Lug’atlar
11. Sana va vaqt bilan ishlash
12. Foydalanuvchi funksiyalari
13. Pythonda modullar va paketlar
14. Fayl va kataloglar bilan ishlash
15. Pythonda ba’zi algoritmlar
16. Pythonda istisnolar va xatolarni qayta ishlash
17. Pythonada obyektga yo’naltirilgan dasturlash
18. Grafik interfeysli ilovalarni qayata ishlash
19. Pythonda kliyent-serverli dastirlash)
20. Arduino kontrollerini dasturlash

Izoh: Amaliy mashg’ulot soatlari hajmlaridan kelib chiqqan holda ishchi dasturda mazkur mavzular ichidan amaliy mashg’ulot mavzulari shakllantiriladi.

Mustaqil ta`limni tashkil etishning shakli va mazmuni

Talaba mustaqil ta`limning asosiy maqsadi – o'qituvchining rahbarligi va nazoratida muayyan o'quv ishlarini mustaqil ravishda bajarish uchun bilim va ko'nikmalarini shakllantirish va rivojlantirish.

Mustaqil ishlarni bajarish jarayonida talabalar quyidagi ishlarni bajaradidar:

- darslik va o'quv qo'llanmalar asosida fan mavzulari bo'yicha nazariy tayyorgarlik ko'rish, amaliy va laboratoriya mashg'ulotlariga tayyorlanish;
- tarqatma materiallar bo'yicha ma`ruzalarni chuqur o'zlashtirish;
- fan mazmunida ko'rsatilmagan dasturlash tillari va muhitlari bilan tanishish va qiyosiy tahlil qilish;
- masofaviy ta`lim orqali dasturlash bilan turdosh fanlar bo'yicha o'quv kurslarida qatnashish va mos sertifikatlarga ega bo'lish tavsiya qilinadi.

Talaba mustaqil ishini tashkil etishda quyidagi shakllardan foydalanadi:

- berilgan mavzular bo'yicha axborot (referat) tayyorlash;
- nazariy bilimlarni amaliyotda qo'llash;
- maket, model va namunalar yaratish va h.k.

Tavsiya etilayotgan mustaqil ishlarning mavzulari

17. Pythonda na'lumotlar bazasi bilan ishash
18. PyQT da ishlash
19. Pythonda Tkinter moduli bilan ishlash
20. Pythonga grafika bilan ishlash
21. Pythonda modullarni o'rnatish
22. Pythonda web sayt uchun dastur yaratish
23. Pythonda tarmoq bilan ishlash
24. Python tilida mobil dasturlash
25. Python freymworklari
26. Django freymworki bilan ishlash
27. Flask freymworki bilan ishlash
28. Python da o'yinlarni dasturlash
29. Pygame modulini o'rnatish
30. Python dasturlash tilida ma'lumotlarni vizuallashtirish
31. Matplotlib kutubxonasi bilan tanishish
32. Python dasturlash tilada API lar bilan ishlash

Izoh: Mustaqil ta`lim soatlari hajmlaridan kelib chiqqan holda ishchi dasturda mazkur mavzular ichidan mustaqil ta`lim mavzulari shakllantiriladi.

Dasturning informatsion-uslubiy ta'minoti

Mazkur fanni o'qitish jarayonida ta'limning zamonaviy metodlari, pedagogik va axborot-kommunikatsiya texnologiyalari qo'llanilishi nazarda tutilgan.

- tizimli dasturlash tillari bo'limiga tegishli ma'ruza darslarida zamonaviy kompyuter

texnologiyalari yordamida prezentatsion va elektron-didaktik texnologiyalaridan;

- tip o'zgartiruvchi operatorlar va ularni formatlari. Ro’yxatlar. Ro’yxatlar ustida turli amallar bajarish mavzularida o'tkaziladigan amaliy mashg'ulotlarda aqliy xujum, guruxli fikrlash pedagogik texnologiyalaridan;

- tarmoqlanuvchi jarayonlar uchun dasturlar yaratish mavzularida o'tkaziladigan tajriba mashg'ulotlarida kichik guruxlar musobaqalari, guruxli fikrlash pedagogik texnologiyalarini qo'llash nazarda tutiladi.

Foydalanilgan asosiy darsliklar va o'quv qo'llanmalar ro'yxati

17. Н. А. Прохоренок, В. А. Дронов. “Python 3 и PyQt 5. Разработка приложений”. БХВ-Петербург, 2016. — 832 с.: ил.
18. Д.Ю.Федоров. “Основы программирования на примере языка Python” Учебное пособие. Санкт-Петербург. 2-е изд., перераб. и доп. — М.: Издательство Юрайт, 2019. — 161 с.
19. Любанович Билл. “Простой Python. Современный стиль программирования”. — СПб.: Питер, 2016. — 480 с.: ил. — (Серия «Бестселлеры O’Reilly»).
20. Васильев А. Н. “Python на примерах. Практический курс по программированию”. - СПб.: Наука и Техника, 2016. - 432 с.: ил.
21. Рашка С. “Python и машинное обучение”/ пер.с англ. А. В. Логунова. - М.: ДМК Пресс, 2017. - 418 с.: ил.
22. Шолле Франсуа. “Глубокое обучение на Python”. — СПб.: Питер, 2018. — 400 с.: ил.
23. Мэтиз Эрик. “Изучаем Python. Программирование игр, визуализация данных, веб-приложения”. СПб.: Питер, 2017. — 496 с.: ил.
24. JOHNE. GRAYSON. “Python and TkinterProgramming”. MANNING Greenwich (74° w. long.) ©2000 by Manning Publications Co.

Internet ma'lumotlarini olish mumkin bo'lgan saytlar:

www.ziyonet.uz;
www.google.com;
www.python.org;
www.qt.com;
www.intuit.ru.

O`ZBEKISTON RESPUBLIKASI
OLIY VA O`RTA MAXSUS TA`LIM VAZIRLIGI
NAMANGAN MUHANDISLIK-QURILISH INSTITUTI

Ro’yhatga olindi:
№_____

202_ y. «___» _____

"Tasdiqlayman"
O’quv ishlari bo’yicha prorektor
_____ prof.M.Dadamirzayev
«___» _____202_ y.

PYTHON DASTURLASH TILI
fanining
ISHCHI FAN DASTURI

Bilim sohasi: 300 000 - Ishlab chiqarish – texnik soha

Ta’lim sohasi: 320 000 - Ishlab chiqarish texnologiyalari

Ta’lim yo’nalishi: 5321700-Texnologik jarayonlarni boshqarishning axborot-kommunikasiya tizimlari

Semestr	Fan tarkibi						Nazorat turi	Ja'mi o'quv soati
	Ma'ruza	Amaliy mashg'ulot	Labora-toriya ishlari	Seminar mashg'ulot	Mustaqil ta'lim	Kurs ishi (loyihasi)		
Kunduzgi bo'lim								
VI	**24**	**24**			**52**		**+**	**100**

Namangan – 202_ y.

Fanning ishchi o’quv dasturi o’quv, ishchi o’quv reja va NamMQI ning _____ dagi ____ - sonli buyrug’i va №_____ raqamli “Pythonda dasturlash” fanining o’quv dasturiga muvofiq ishlab chiqildi.

Tuzuvchilar:

Goyipov U.G. – NamMQI, «Informatika va AT» kafedrasi oq’ituvchisi.

Taqrizchilar:

Imomov A. – Namangan Davlat Universiteti "Amaliy matematika va axborot texnologiyalari" kafedrasi dotsenti, f-m.f.n.
Isomiddinov A. – NamMQI, Texnik tizimlarda AT kafedrasi mudiri, PhD.

Fanning ishchi o’quv dasturi «Informatika va AT» kafedrasining kafedrasining 202_ yil «___» _____dagi «_____» -son yig’ilishida muhokamadan o’tgan va fakultet kengashida muhokama qilish uchun tavsiya etilgan.

Kafedra mudiri: _____ T.Jo’rayev

Fanning ishchi o’quv dasturi «Sanoatni axborotlashtirish» fakultetining kengashida muhokamadan o’tgan va foydalanishga tavsiya etilgan.
(202_ yil «___» _____dagi «_____» -sonli bayonnoma).

Fakultet kengashi raisi: _____ A.Qaxxorov

KELISHILDI:

Mutaxassislik kafedralari:

_____	_____	_____
Kafedra nomi	Imzo	Kafedra mudiri F.I.Sh

O’quv-uslubiy boshqarma boshlig’i: _____ Q.Inoyatov

Namangan muhandislik–qurilish instituti o’quv-uslubiy kengashida ko'rib chiqilgan va tavsiya qilingan. «__»_____202_ y.dagi ___ sonli majlis bayoni.
(__ - son bilan ro'yhatga olingan).

Kirish

Hozirgi kunda davlatimiz siyosatining ustuvor yo'nalishlaridan biri bu – zamonaviy axborot texnologiyalarini ishlab chiqarishga hamda jamiyatimizning deyarli barcha jabhalariga keng tatbiq qilish, kompyuterlashtirish va avtomatlashtirish tizimlaridan yanada samarali foydalanishdir. Shu ma'noda, kadrlar tayyorlash tizimini takomillashtirish masalasi dolzarb vazifalardan biridir. Zero buyuk mutafakkir bobomiz Muhammad Al-Xorazmiyning o'lmas me'rosi butun dunyo olimlarining e'tirofida bo'lgani holda, yoshlarimizning ham ushbu yo'nalishda katta yutuqlarga erishishi lozimdir.

Shuning uchun ham, o'quv dasturlarini zamon talablari asosida takomillashtirib borish o'ta zarur masalalardan biri bo'lib hisoblanadi.

Ayniqsa, zamonaviy axborot texnologiyalaridan foydalanish, ularni boshqarish, dasturlashtirish kabi eng zamonaviy ilmlarni o'zlartirish hozirgi kun talabalari uchun muhim vazifadir.

Mazkur ishchi dastur «Python dasturlash tili» fanidan yaratilgan namunaviy o`quv dasturi asosida ishlab chiqildi.

Fanning maqsad va vazifalari

«Python dasturlash tili» fanini talabalarga o'qitishdan **maqsad** quyidagilardan iborat:

- zamonaviy dasturlash tillaridan biri–Pythonni mukammal o'rgatish;
- Python dasturlash tilining asosiy tushunchalarini o'rgatish;
- Python dasturlash tilining kutubxona fayllaridan foydalanishni o'rgatish;
- Python dasturlash tilida funktsiya grafiklarini yasash dasturlarini yaratish;
- Python dasturlash tili tizimlar ishini boshqarish dasturlarini yaratish.

Fanning vazifasi - talabani ushbu fan bo'yicha olgan nazariy va amaliy bilimlarini kurs loyihasi va bitiruv ishlarini bajarish bilan real sharoitga qo'llash bo'yicha ko'nikmalar hosil qilishdan iborat.

Fan bo'yicha talabalarning bilimiga, ko'nikma va malakasiga qo'yiladigan talablar

"Python dasturlash tili" o'quv fanini o'zlashtirish jarayonida amalga oshiriladigan masalalar doirasida bakalavr:

- Python dasturlash tili asosiy tushunchalari va metodlarini;
- tabiatshunoslik va texnikadagi eng oddiy tizimlar va jarayonlarning matematik modellarini;

muayyan jarayonlar uchun algoritmlarni yaratish va tahlil qilishni ***bilishi kerak***;

- Murakkab jarayonlarni ifodalash uchun matematik simvollardan foydalanish;
- Tasvirlashni hisobga olgan holda modellarni tadqiq qilish ***ko'nikmalariga ega bo'lshii*** kerak.

– talaba fanning turli bo'limlariga xarakterli bo'lgan qiymatlarning sonli tartibini o'lchash va baholash ***malakalariga ega bo'lshii kerak***.

Fanning o`quv rejadagi boshqa fanlar bilan o`zaro bog'liqligi va uslubiy jihatdan uzviy ketma-ketligi

«Python dasturlash tili» fani bir qator fanlar bilan uzviy bogliqdir. Shu jumladan «Dasturlash texnologiyasi»? «Dasturlash asoslari» va «Texnik tizimlarida AT» fanlarini bilish juda muhimdir. Fanni o'rganish natijasida ega bo'lingan bilim va malakalar ixtisoslik fanlarini muvaffaqqiyatli o'zlashtirishda va magistrlik dissertatsiyalarini bajarishda o'z ifodasini topadi.

"Python dasturlash tili" o'quv fanini o'zlashtirish jarayonida amalga oshiriladigan masalalar doirasida bakalavr:

- Python dasturlash tilining asosiy tushunchalari va metodlarini;
- tabiatshunoslik va texnikadagi eng oddiy tizimlar va jarayonlarning matematik modellarini;

muayyan jarayonlar uchun algoritmlarni yaratish va tahlil qilishni ***bilishi kerak***;

- operatorlari va ularni tasvirlashni hisobga olgan holda modellarni tadqiq qilish;
- eksperimental ma`lumotlarga ishlov berishning asosiy metodlari va usullaridan foydalanish ***ko'nikmalariga ega bo'lshii*** kerak.

– talaba fanning turli bo'limlariga xarakterli bo'lgan qiymatlarning sonli tartibini o'lchash va baholash ***malakalariga ega bo'lshii kerak***.

Fanni o`qitishda zamonaviy axborot va pedagogik texnologi lar

O`quv jarayoni bilan bog`liq ta`lim sifatini belgilovchi holatlar quyidagilar: yuqori ilmiy-pedagogik darajada dars berish, muammoli ma`ruzalar o`qish, darslarni savol-javob tarzida qiziqarli tashkil qilish, ilg`or pedagogik texnologiyalardan va mul`timedia vositalaridan foydalanish, tinglovchilarni undaydigan, o`ylantiradigan muammolarni ular oldiga qo`yish, talabchanlik, tinglovchilar bilan individual ishlash, erkin muloqot yuritishga, ilmiy izlanishga jalb qilish.

"Python dasturlash tili" kursini loyihalashtirishda quyidagi asosiy kontseptual yondoshuvlardan foydalaniladi:

Shaxsga yo`naltirilgan ta`lim. Bu ta`lim o`z mohiyatiga ko`ra ta`lim jarayonining barcha ishtirokchilarini to`laqonli rivojlanishlarini ko`zda tutadi. Bu esa ta`limni loyihalashtirilayotganda, albatta, ma`lum bir ta`lim oluvchining shaxsini emas, avvalo, kelgusidagi mutaxassislik faoliyati bilan bog`liq o`qish maqsadlaridan kelib chiqqan holda yondoshilishni nazarda tutadi.

Tizimli yondoshuv. Ta`lim texnologiyasi tizimning barcha belgilarini o`zida mujassam etmog`i lozim: jarayonning mantiqiyligi, uning barcha bo`g`inlarini o`zaro bog`langanligi, yaxlitligi.

**Faoli tga yo`naltirilgan yondoshuv.** Shaxsning jarayonli sifatlarini shakllantirishga, ta`lim oluvchining faoliyatni aktivlashtirish va intensivlashtirish, o`quv jarayonida uning barcha qobiliyati va imkoniyatlari, tashabbuskorligini ochishga yo`naltirilgan ta`limni ifodalaydi.

Dialogik yondoshuv. Bu yondoshuv o`quv munosabatlarini yaratish zaruriyatini bildiradi. Uning natijasida shaxsning o`z-o`zini faollashtirishi va o`z-o`zini ko`rsata olishi kabi ijodiy faoliyati kuchayadi.

**Hamkorlikdagi ta`limni tashkil etish.** Demokratik, tenglik, ta`lim beruvchi va

ta`lim oluvchi faoliyat mazmunini shakllantirishda va erishilgan natijalarni baholashda birgalikda ishlashni joriy etishga e`tiborni qaratish zarurligini bildiradi.

**Muammoli ta`lim.** Ta`lim mazmunini muammoli tarzda taqdim qilish orqali ta`lim oluvchi faoliyatini aktivlashtirish usullaridan biri. Bunda ilmiy bilimni ob`ektiv qarama-qarshiligi va uni hal etish usullarini, dialektik mushohadani shakllantirish va rivojlantirishni, amaliy faoliyatga ularni ijodiy tarzda qo`llashni mustaqil ijodiy faoliyati ta`minlanadi.

**Axborotni taqdim qilishning zamonaviy vositalari va usullarini qo`llash** - yangi kompyuter va axborot texnologiyalarini o`quv jarayoniga qo`llash.

**O`qitishning usullari va texnikasi.** Ma`ruza (kirish, mavzuga oid, vizuallash), muammoli ta`lim, keys-stadi, pinbord, paradoks va loyihalash usullari, amaliy ishlar.

**O`qitishni tashkil etish shakllari:** dialog, polilog, muloqot hamkorlik va o`zaro o`rganishga asoslangan frontal, kollektiv va guruh.

**O`qitish vositalari:** o`qitishning an`anaviy shakllari (darslik, ma`ruza matni) bilan bir qatorda – kompyuter va axborot texnologiyalari.

Kommunikatsi usullari: tinglovchilar bilan operativ teskari aloqaga asoslangan bevosita o`zaro munosabatlar.

Teskari aloqa usullari va vositalari: kuzatish, blits-so`rov, oraliq va joriy va yakunlovchi nazorat natijalarini tahlili asosida o`qitish diagnostikasi.

Boshqarish usullari va vositalari: o`quv mashg`uloti bosqichlarini belgilab beruvchi texnologik karta ko`rinishidagi o`quv mashg`ulotlarini rejalashtirish, qo`yilgan maqsadga erishishda o`qituvchi va tinglovchining birgalikdagi harakati, nafaqat auditoriya mashg`ulotlari, balki auditoriyadan tashqari mustaqil ishlarning nazorati.

Monitoring va baholash: o`quv mashg`ulotida ham butun kurs davomida ham o`qitishning natijalarini rejali tarzda kuzatib borish. Kurs oxirida test topshiriqlari yoki yozma ish variantlari yordamida tinglovchilarning bilimlari baholanadi.

“Python dasturlash tili” fanini o`qitish jarayonida kompyuter texnologiyasidan foydalaniladi. Ayrim mavzular bo`yicha talabalar bilimini baholash test asosida va kompyuter yordamida bajariladi. “Internet” tarmog`idagi rasmiy iqtisodiy ko`rsatkichlaridan foydalaniladi, tarqatma materiallar tayyorlanadi, test tizimi hamda tayanch so`z va iboralar asosida oraliq va yakuniy nazoratlar o`tkaziladi.

ASOSIY QISM

“Python dasturlash tili” fani bo’yicha ma’ruza mashg’ulotlarining kalendar tematik rejasi

T.R	Ma’ruza mashg’ulotning tematik rejasi	Soati
1.	**Python dasturlash tili.** Python dasturlash tili yaratilishi tarixi. Python dasturlsh tili imkoniyatlari. Pythonni o’rnatish. Dastur tuzilishi. Izoxlar. Dastur natijasini chop etish. Ma’lumotlarni kiritish	2
2.	**O’zgaruvchilar.** O’zgaruvchini nomlash. Ma’lumot turlari. O’zgaruvchiqa qiymat o’zlashtirish. Ma’lumot tipini aniqlash. Ma’lumot tipini o’zgartirish. O’zgaruvchini o’chirish.	2
3.	**Operatorlar.** Matematik operatorlar. Munosabat operatorlari. Ketma-ketliklar bilan ishlash operatorlari. O’zlashtirish operatorlari.	2
4.	**Shartli operatorlar.** Taqqoslash operatorlari. if...else operatori	2
5.	**Sikl operatorlari.** For sikli. range() va enumerate() funksiyalari. While sikli. continue operatori. break operatori	2
6.	**Sonlar.** Sonlar bilan ishlashning tashqi funksiya va metodlari. *math* moduli. Matematik funksiyalar. *random* moduli. Tasodifiy son generasiyasi	2
7.	**Qatorlar va ular ustida amallar.** Qator yaratish. Maxusu belgilar. Qatorlar bilan ishlash amallari. Qatorlarni formatlash. format () metodi. Qatorlar bilan ishlash metod va funksiyalari.	2
8.	**Ro’yxatlar, Kortejlar, To’plamlar va Diapazonlar.** Ro’yxat yaratish. Ro’yxatlar ustida amallar. Ko’p o’lchamli ro’yxatlar. Ro’yxat elemantlarini saralash. Ro’yxat generatorlari. Ro’yaxatga elementlar qo’shish va o’chirish. Ro`yxat elementlarini qidirish va ro’yxatga kirivchi qiymatlari haqida ma’lumot olish. Ro’yxatni teskarilash va aralshtirish. Tasodifiy elementni tanlash. Ro’yxatni saralash. Ro’yxatni sonlar bilan to’ldirish. Ro’yxatni satrga o’tkazish. Kortejlar. To’plamlar.	2
9.	**Lug’atlar.** Lug’at yaratish. Lug’atlar ustida amallar. Lug’at elementlarini saralash. Lug’atlar bilan ishlash metodlari. Lug’atlar generatori	2
10.	**Sana va vaqt bilan ishlash.** Joriy sana va vaqtni chop etish. Sana va vaqt formati. Sana ustida bajariladigan asosiy amallar. Sana va vaqtni formatlash. Sana va vaqtlarni qo'shish va ayirish. *timedelta* xususiyatlari.	2
11.	**Foydalanuvchi funksiyalari.** Funksiyai aniqlanishi va uni chaqirish. **Pythonda modullar va paketlar**. *import* va *from* ko’rsatmasi. Moduldan qidirish yo’li. Modullarini qayta yuklash	2
12.	**Fayl va kataloglar bilan ishlash.** Faylni ochish. Fayllarbilan ishlash metodlari. **Grafik interfeysli ilovalarni qayata ishlash.** Tkinter moduli bilan ishlash asoslari.	2
JAMI		**24**

“Python dasturlash tili” fani bo’yicha amaliy mashg’ulotlarning kalendar tematik rejasi

T.R	Amaliy mashg’ulotning tematik rejasi	Soati
1.	**Python dasturlash tili bilan tanishish.** Python muhitini o’rnatish. Dastlabki dasturni tuzish. Ma’lumotlarni kiritish, dastur natijasini chop etish.	2
2.	**O’zgaruvchilar** bilan ishlash. O’zgaruvchiqa qiymat o’zlashtirish. Ma’lumot tipini aniqlash, o’zgartirish. O’zgaruvchini o’chirish.	2
3.	**Dasturda operatorlardan foydalanish.** Matematik, munosabat va ketma-ketliklar bilan ishlash operatorlari. O’zlashtirish operatorlari. Operatorlarni bajarish ketma-ketligi.	2
4.	**Shartli operatorlarni qo’llash.** Taqqoslash operatorlari (if...else) dan foydalanib masalalarni dasturlash.	2
5.	**Sikl operatorlaridan foydalanib dasturlash.** For, While sikl operatorlari, range() va enumerate() funksiyalarinidan foydalanishni o’rganish. Continue va break operatorlarini qo’llash.	2
6.	**Sonlar** bilan ishlashning tashqi funksiya va metodlaridan foydalanib dasturlash. *math* modulini qo’llash. Matematik funksiyalar. *random* moduli. Tasodifiy son generasiyasi dasturi	2
7.	**Qatorlar va ular ustida amallar bajarish.** Qator yaratish. Qatorlar bilan ishlash amallari. Qatorlarni formatlash. Belgilar registrini o'zgartirish. Belgilar bilan ishlash funksiyalarini qo’llash.	2
8.	**Ro’yxatlar, Kortejlar, To’plamlar va Diapazonlar bilan tanishish.** Ro’yxat yaratish. Ro’yxatlar ustida amallar bajarish. Ro’yxat elementlarini saralash. Ro’yaxatga elementlar qo’shish va o’chirish. Ro`yxat elementlarini qidirish va ro’yxatga kirivchi qiymatlari haqida ma’lumot olish. Ro’yxatni teskarilash va aralshtirish. Tasodifiy elementni tanlash. Ro’yxatni saralash. Ro’yxatni sonlar bilan to’ldirish. Ro’yxatni satrga o’tkazish. Kortejlar va To’plamlar bilan tanishish.	2
9.	**Lug’at yaratish.** Lug’atlar ustida amallar bajarish. Lug’at elementlarini saralash.	2
10.	**Sana va vaqt bilan ishlash funksiyalarini o’rganish.** Joriy sana va vaqtni chop etish. Sana va vaqtni formatlash.	2
11.	**Foydalanuvchi funksiyalarini yaratish.** Funksiyai aniqlanishi va uni chaqirish. **Pythonda modullar va paketlardan foydalanish.**	2
12.	**Fayl va kataloglar bilan ishlash. Faylni ochish. Fayllar bilan ishlash metodlarini o'rganish.**	2
JAMI		**24**

Mustaqil ta`limni tashkil etishning shakli va mavzulari

“Python dasturlash tili” fani bo`yicha talabaning mustaqil ta`limi shu fanni o`rganish jarayonining tarkibiy qismi bo`lib, uslubiy va axborot resurslari bilan to`la ta`minlangan. Talabalar auditoriya mashg`ulotlarida professor-o`qituvchilarning ma`ruzasini tinglaydilar, misol va masalalar yechadilar. Auditoriyadan tashqarida talaba darslarga tayyorlanadi, adabiyotlarni konspekt qiladi, uy vazifa sifatida berilgan misol va masalalarni yechadi.

Bundan tashqari ayrim mavzularni kengroq o`rganish maqsadida qo`shimcha adabiyotlarni o`qib referatlar tayyorlaydi hamda mavzu bo`yicha testlar yechadi.

Mustaqil ta`lim natijalari reyting tizimi asosida baholanadi.

Uyga vazifalarni bajarish, qo`shimcha darslik va adabiyotlardan yangi bilimlarni

mustaqil o`rganish, kerakli ma`lumotlarni izlash va ularni topish yo`llarini aniqlash, internet tarmoqlaridan foydalanib ma`lumotlar to`plash va ilmiy izlanishlar olib borish, ilmiy to`garak doirasida yoki mustaqil ravishda ilmiy manbalardan foydalanib ilmiy maqola va ma`ruzalar tayyorlash kabilar talabalarning darsda olgan bilimlarini chuqurlashtiradi, ularning mustaqil fikrlash va ijodiy qobiliyatini rivojlantiradi. Shuning uchun ham mustaqil ta`limsiz o`quv faoliyati samarali bo`lishi mumkin emas. Uy vazifalarini tekshirish va baholash amaliy mashg`ulot olib boruvchi o`qituvchi tomonidan, konspektlarni va mavzuni o`zlashtirish darajasini tekshirish va baholash esa ma`ruza darslarini olib boruvchi o`qituvchi tomonidan har darsda amalga oshiriladi.

"Python dasturlash tili" fanidan mustaqil ish majmuasi fanning barcha mavzularini qamrab olgan va quyidagi 4 ta katta mavzu ko`rinishida shakllantirilgan.

Talabalar mustaqil ta'limining mazmuni va hajmi
Tavsiya etilayotgan mustaqil ishlarning mavzulari:

33. Pythonda na'lumotlar bazasi bilan ishash
34. PyQT da ishlash
35. Pythonda Tkinter moduli bilan ishlash
36. Pythonga grafika bilan ishlash
37. Pythonda modullarni o'rnatish
38. Pythonda web sayt uchun dastur yaratish
39. Pythonda tarmoq bilan ishlash
40. Python tilida mobil dasturlash
41. Python freymworklari
42. Django freymworki bilan ishlash
43. Flask freymworki bilan ishlash
44. Python da o'yinlarni dasturlash
45. Pygame modulini o'rnatish
46. Python dasturlash tilida ma'lumotlarni vizuallashtirish
47. Matplotlib kutubxonasi bilan tanishish
48. Python dasturlash tilada API lar bilan ishlash

Dasturning informasion-uslubiy ta`minoti

Mazkur fanni o`qitish jarayonida ta`limning zamonaviy metodlari, pedagogik va ahborot-kommunikasiy tehnologiylari qo`llanilishi nazarda tutilgan.

- fanni bulimlariga tegishli ma`ruza darslarida zamonaviy kompyter tehnologiylari yordamida prezentasion va elektron-didaktik tehnologiyalaridan;
- "Python dasturlash tili" tarkib toptirish mavzularida o`tkaziladigan mashg`ulotlarda aqliy hujum, guruhli fikrlash pedagogik tehnologiyalaridan;
- Amaliy, mashg`ulotlar mabzusi va mazmuni, asboblarni tarirovka qilish, haroratni o`lchash usullari, asbob yrdamida darajalashni o`rganish suyqlik va gazlarni tezliklarini hisoblash mavzularida utkaziladigan tajriba mashg`ulotlarida kichik guruhlar musobaqalari, guruhli fikrlash, pedagogik tehnologiylarni qullash nazarda tutiladi.

“Python dasturlash tili” fanidan

BAHOLASH MEZONI

Talabalar bilimini nazorat qilish uchun ishlab chiqilgan ushbu baholash mezoni O'zbekiston Respublikasi Oliy va o'rta maxsus ta'lim vazirining “Oliy ta'lim muassasalarida talabalar bilimini nazorat qilish va baholash tizimi to'g'risidagi nizomni tasdiqlash haqida” 2018 yil 9 avgustdagi 19-2018-sonli buyrug'i asosida ishlab chiqildi.

Talabalar bilimini nazorat qilish oraliq va yakuniy nazorat turlarini o'tkazish orqali amalga oshiriladi.

Oraliq nazorat (ON) semestr davomida ishchi fan dasturining tegishli bo'limi tugagandan keyin talabaning bilim va amaliy ko'nikmalarini baholash maqsadida o'quv mashg'ulotlari davomida o'tkaziladi.

Ushbu fan bo'yicha oraliq nazorat turi ikki marta o'tkaziladi. Oraliq nazorat yozma shaklda o'tkaziladi. 1-ON ma'ruzalarning birinchi yarmi o’tilgandan keyin, 2-ON esa, qolgan ikkinchi yarmi o'tilgandan keyin, yakuniy nazorat (YN) esa, o'z navbatida semestr yakunida o'tkaziladi.

Shu bilan birga, ushbu fan bo'yicha talabaning amaliy, tajriba va mustaqil ta'lim topshiriqlarini bajarishi, shuningdek uning ushbu mashg'ulotlardagi faolligi ham baholab boriladi.

Talabalar bilimini baholash 5 baholik tizimda amalga oshiriladi.

Talabalarning bilimi quyidagi mezonlar asosida:

- talaba mustaqil xulosa va qaror qabul qiladi, ijodiy fikrlay oladi, mustaqil mushohada yuritadi, olgan bilimini amalda qo'llay oladi, fanning (mavzuning) mohiyatini tushunadi, biladi, ifodalay oladi, aytib beradi hamda fan (mavzu) bo'yicha tasavvurga ega deb topilganda — 5 (a'lo) baho;
- talaba mustaqil mushohada yuritadi, olgan bilimini amalda qo'llay oladi, fanning (mavzuning) mohiyatni tushunadi, biladi, ifodalay oladi, aytib beradi hamda fan (mavzu) bo'yicha tasavvurga ega deb topilganda — 4 (yaxshi) baho;
- talaba olgan bilimini amalda qo'llay oladi, fanning (mavzuning) mohiyatni tushunadi, biladi, ifodalay oladi, aytib beradi hamda fan (mavzu) bo'yicha tasavvurga ega deb topilganda — 3 (qoniqarli) baho;
- talaba fan dasturini o'zlashtirmagan, fanning (mavzuning) mohiyatini tushunmaydi hamda fan (mavzu) bo'yicha tasavvurga ega emas deb topilganda — 2 (qoniqarsiz) baho bilan baholanadi.

Nizomga ko'ra, yakuniy nazorat turini o'tkazish va mazkur nazorat turi bo'yicha talabaning bilimini baholash o'quv mashg'ulotlarini olib bormagan professor-o'qituvchi tomonidan amalga oshiriladi. Shu sababli, ushbu fandan YN o'tkazish uchun kafedra o’qituvchilari hamda NamDU “Amaliy matematika” kafedrasining turdosh fan o'qituvchi-professorlarini YN ga jalb etish ko'zda tutilgan.

Oraliq nazorat turini topshirmagan, shuningdek ushbu nazorat turi bo'yicha «2» (qoniqarsiz) baho bilan baholangan talaba yakuniy nazorat turiga kiritilmaydi.

Ushbu fan bo'yicha quyidagicha baholash tizimini tashkil qilinadi:

Fan bo'yicha 6 ta tajriba ishi (***A***) o'tkaziladi. Amaliy ishlar bo'yicha baholash quyidagicha hisoblanadi:

$$\boldsymbol{A} = \frac{A_1 + A_2 + A_3 + A_4 + A_5 + A_6}{6}.$$

Amaliy mashg'ulot o'tish davomida oraliq nazorat o'tkaziladigan kunga qadar xar bir talaba 6 ta amaliy ish topshiradi va baholanadi. Bu baholash **amaliy** (***A***) topshiriqlar asosida amalga oshiriladi. Talabaning **o'zlashtirishini** (***O'***) hisoblashda shuningdek **mustaqil ta'lim** (***M***) hamda talabaning **faolligini** (***F***) ham xisobga olinadi. Umumiy holda **o'zlashtirish**ni qiyidagicha hisoblanadi:

$$\boldsymbol{O'} = \frac{A + M + F}{3}.$$

Ijobiy o'zlashtirgan talabalar **oraliq narorat** (***ON***) ga qo'yiladi. Ushbu fan bo'yicha oraliq nazorat yozma ish shaklida o'tkaziladi.

Oraliq nazoratdan ijobiy baholangan talabalar **yakuniy nazorat** (***Yn***) ga qo'yiladi.

Qo'yilgan baholar hammasi guruh jurnalida qayd qilib boriladi.

Talabaning YN da olgan bahosi yakuniy baho hisoblanadi va u qaydnomaga hamda talabaning baholash daftarchasiga qo'yiladi.

Topshiriqlar banki

1-amaliy ish topshiriq variantlari

31. Kvadratning tomoni *a* berilgan. Uning perimetrini va yuzasini topish dasturi tuzilsin.
32. Temperaturaning Farengeyt gradusidagi qiymati berilgan. Shu temperaturaning Selsiydagi qiymati hisoblansin. $T_C = (T_F - 32) \cdot \frac{5}{9}$
33. *l* santimetr uzunlik berilgan. Uning necha metr ekanligi aniqlansin.
34. Temperaturaning Selsiy gradusidagi qiymati berilgan. Shu temperaturaning Farengeytdagi qiymati hisoblansin. $T_F = T_C \cdot \frac{9}{5} + 32$
35. *m* massa kilogrammlarda berilgan. Uning tonnalardagi qiymati topilsin.
36. *x* kg mahsulotning narxi *a* so'm. Shu mahsulotning 1 kgi va *y* kgi necha so'm ekanligi aniqlansin.
37. *f* fayl o'lchami bitlarda berilgan. Uning necha kilobayt ekanligi topilsin.
38. *s* yuza sm^2 larda berilgan. Uni dm^2 lardagi qiymati hisoblansin.
39. Kubning qirrasining uzunligi berilgan. Kubning hajmini va yon sirtining yuzini hisoblash algoritmini tuzing.
40. To'g'ri to'rtburchakning *a* va *b* tomonlari berilgan. Uning perimetri va yuzasini topish dasturi tuzilsin.
41. Ikkita haqiqiy musbat son berilgan. Ularning o'rta arifmetigi va o'rta geometrigini hisoblash dasturini tuzing.

42. Aylananing diametri *d* berilgan. Uning uzunligini topish dasturi tuzilsin. π=3,14 ga teng **deb olinsin**.
43. To'g'ri burchakli uchburchak katetlarining uzunligi berilgan. Uning gipotenuzasi va yuzasini topish dasturini tuzing.
44. Kubning qirrasi *a* berilgan. Uning hajmini va sirti yuzasini topish dasturi tuzilsin.
45. To'g'ri burchakli parallelipipedning *a*, *b*, va *c* qirralari berilgan. Uning hajmini va to'la sirtini topish dasturi tuzilsin.
46. *r* radiusli aylanaga tashqi chizilgan muntazam *n* burchakning peremetrini hisoblash dasturini tuzing.
47. *a* va *b* nomanfiy sonlari berilgan. Bu sonlarning o'rta arifmetigini va o'rta geometrigini topish dasturi tuzilsin.
48. Ikkita *a* va *b* nomanfiy sonlari berilgan. Ularning yig'indisi, ayirmasi, ko'paytmasi va bo'linmasini topish dasturi topilsin.
49. *h* balandlikdan tushayotgan *m* massali toshning tushish vaqtini hisoblash dasturini tuzing.
50. To'g'ri burchakli uchburchakning *a* va *b* katetlari berilga. Uning gipotenuzasi va perimetruini topish dasturi tuzilsin.
51. Umumiy markazli r_1 va r_2 radiusli ikkita doira berilgan. Har ikkala doira va bu doiralar orqali hosil bo'lgan xalqaning yuzalarini hisoblash dasturi tuzilsin.
52. Aylana uzunligi *l* berilgan. Bu aylananing radiusini va aylana chegaralagan doiraning yuzasini hisoblash dasturi tuzilsin. π=3,14 ga teng deb olinsin.
53. Doiraning yuzasi *S* berilgan. Uning diametri *d* va doirani chegaralab turuvchi aylana uzunligi hisoblansin. π=3,14 ga teng deb olinsin.
54. Koordinatalar tekisligida $A(x_1, y_1)$ va $B(x_2, y_2)$ nuqtalar berilgan. *AB* kesma uzunligini hisoblash dasturi tuzilsin.
55. *a* va *b* o'zgaruvchilar berilgan. Ularning qiymatlari o'zaro almashtirilsin. *a* va *b* larning yangi qiymatlari chop etilsin.
56. *a*, *b* va *c* o'zgaruvchilar berilgan. *a* ning qiymati *b* ga, *b* ning qiymati *c* ga, *c* ning qiymati *a* ga almashtirilsin. *a*, *b* va *c* larning yangi qiymatlari chop etilsin.
57. Sutka boshlanganiga *n* soniya bo'ldi. Sutka tugashiga necha daqiqa qolganligi topilsin.
58. *a* va *b* sonlari berilgan. $ax+b=0$ chiziqli tenglamaning yechimini topish dastuiri tuzilsin.
59. *l* santimetr uzunlik berilgan. Uning necha metr ekanligi aniqlansin.
60. *s* yuza sm^2 larda berilgan. Uni dm^2 lardagi qiymati hisoblansin.

2-amaliy ish topshiriq variantlari

29. Turg'un suvdagi qayiq tezligi V km/s. Daryo suvi oqimining tezligi U km/s (U<V). Qayiq ko'lda T_1 soat, daryoda esa (oqimga qarshi) T_2 soat harakat qilgan. Qayiq suzgan umumiy S masofa topilsin.

30. Birinchi avtomobil tezligi V_1 km/s, ikkinchisiniki - V_2 km/s, ular orasidagi masofa - S km. Avtomobillar bir-biridan uzoqlashsa (bir-biriga qarab harakat qilganda), T soatdan keyin ular orasidagi masofa qanday bo'ladi?
31. Asoslari a va b (a>b), katta asosdagi burchagi α bo'lgan teng yonli trapetsiyaning perimetri hamda yuzasi topilsin (burchak radianda beriladi).
32. Noldan farqli berilgan R_1, R_2, R_3 elektr qarshiliklari uchun R hisoblansin. Bunda: $\frac{1}{R}=\frac{1}{R_1}+\frac{1}{R_2}+\frac{1}{R_3}$.
33. Xodimning oylik ish haqiga 45% mukofot puli qo'shilsin. Hosil bo'lgan miqdordan 17% daromad solig'i, 1,5% kasaba uyushmasi va 1% nafaqa solig'i ushlab qolinsin. Qo'lga tegadigan pul miqdori chop etilsin.
34. Teng tomonli uchburchak tomoni berilgan, uchburchak yuzasi topilsin.
35. Uchta musbat son berilgan. Sonlar o'rta geometrigining kasr qismi topilsin.
36. Berilgan katetlari bo'yicha to'g'ri burchakli uchburchakning perimetri va yuzasi hisoblansin.
37. Berilgan ikki tomoni va ular orasidagi burchak (gradusda) asosida uchburchakning uchinchi tomoni va yuzasi topilsin.
38. Og'irligi bir kilogramm bo'lgan mahsulotning narxi berilgan. Uning og'irligi grammlarda kiritilsin va to'lash zarur bo'lgan pul miqdori chop etilsin.
39. 10 metr radiusli silindrik shaklga ega bo'lgan suv bosimi minorasidagi suv sathining balandligi berilgan bo'lsa, suvning hajmi hisoblansin.
40. Bolalar bog'chasiga bir oylik to'lov 70000 so'm (bir oy - 22 kun). Agar bola bog'chaga N (0<N<23) kun kelmagan bo'lsa, bir oy uchun qancha to'lash kerak bo'ladi?
41. R radiusli doiraga ichki chizilgan muntazam n-burchakning perimetri va yuzasi hisoblansin.
42. Aylana diametri *d* berilgan. Uning uzunligini toping $L=\pi\cdot d$.
43. Kubning tomonlari *a* berilgan. Uning hajmini $V=a^3$ va sirti maydonini toping $P=6\cdot a^2$.
44. To'g'ri burchakli paralelopipedning *a, b, s* tomonlari berilgan. Uning hajmini va sirti maydonini toping: $V=a\cdot b\cdot s;\quad S=2\cdot(a\cdot b+ b\cdot s+ a\cdot s)$.
45. Aylana radiusi *R* beilgan. Aylana uzunligi *L* va uning ichi maydoni *S* ni toping: $L=2\cdot\pi\cdot R,\ S=\pi\cdot R^2$. Bu erda π sonini 3.14 ga teng qilib oling.
46. Ikkita *a* va *b* haqiqiy sonlari berilgan. Ularning o'rta arifmetik qiymatini toping: $(a+b)/2$.
47. Ikkita *a* va *b* haqiqiy musbat sonlari berilgan. Ularning o'rta geometrik qiymatini toping: $\sqrt{a\cdot b}$.
48. Doira yuzi *S* berilgan. Uning diametri *D* va aylanasi uzunligi *L* ni

$L = 2 \cdot \pi \cdot R, \quad S = \pi \cdot R^2$ formulalardan foydalanib toping.

49. Sonlar koordinata o'qida joylashgan ikkita nuqta x_1 va x_2 orasidagi masofani toping: $d=|x_2 - x_1|$.
50. Sonlar koordinata o'qida uchta nuqta berilgan A, B, S. AS va BS kesmalari uzunliklari va ular yig'indisini toping.
51. Uchburchakning uchta uchi koordinatalari berilgan: (x_1,y_1), (x_2,y_2), (x_3,y_3). Uning perimetrini va maydonini toring. Uchburchakning maydonini topishda Geron formulasidan foydalaning

$$S = \sqrt{p \cdot (p-a) \cdot (p-b) \cdot (p-c)}.$$

Bu erda *a, b, s* - uchburchak tomonlari, $p=(a+b+s)/2$ – yarim perimetr.

52. $V=S/T$ tezlikni hisoblash formulasi orqali, masofa va vaqtni berib jism tezigini hisoblang.
53. Uchta *a, b, s* o'zgaruvchilar qiymatlarini ketma-ket almashtiring (*a* qiymatini *b* ga, *b* qiymatini *s* ga va *s* qiymatini *a* ga).
54. Uchta *a, b, s* o'zgaruvchilar qiymatlarini ketma-ket almashtiring (*a* qiymatini *s* ga, *s* qiymatini *b* ga va *b* qiymatini *a* ga).
55. Kvadratning tomonlari *a* berilgan. Uning perimetrini toping $P=4 \cdot a$.
56. Kvadratning tomonlari *a* berilgan. Uning yuzasini toping $P=a^2$.

3-amaliy ish topshiriqlari

29. *a, b, c* butun sonlar berilgan bo'lsa, ularning $a \le b \le c$ shartni qanoatlantirishi aniqlansin.
30. *a, b, c* butun sonlar berilgan. *b* sonining, *a* va *c* sonlar orasida yotishi aniqlansin.
31. *a* va *b* butun sonlar berilgan. Ularning bir vaqtda toq bo'lmasligi aniqlansin.
32. *a* va *b* butun sonlar berilgan bo'lsa, ularning hech bo'lmaganda bittasi toq ekanligi aniqlansin.
33. *a* va *b* butun sonlar berilgan bo'lsa, bu sonlardan biri toq ekanligi aniqlansin.
34. *a* va *b* butun sonlar berilgan bo'lsa, ularning bir xil juftlikka ega ekanligi aniqlansin.
35. *a, b, c* butun sonlar berilgan. Ularning har biri musbat ekanligi tekshirilsin.
36. *a, b, c* butun sonlar berilgan bo'lsa, ularning hech bo'lmaganda bittasi musbat ekanligi aniqlansin.
37. *a, b, c* butun sonlar berilgan bo'lsa, ulardan faqat bittasi musbat bo'lishi aniqlansin.
38. *a, b, c* butun sonlar berilgan bo'lsa, ulardan faqat ikkitasi bir vaqtda musbat ekanligi aniqlansin.
39. Butun musbat son berilgan bo'lsa, uning bir vaqtda juft va ikki xonali ekanligi aniqlansin.
40. Butun musbat son berilgan bo'lsa, uning bir vaqtda toqligi va uch xonali ekanligi aniqlansin.
41. *a, b, c* butun sonlar berilgan. *b* sonining, *a* va *c* sonlar orasida yotishi aniqlansin.

42. *a* va *b* butun sonlar berilgan. Ularning bir vaqtda toq bo‘lmasligi aniqlansin.
43. *a* va *b* butun sonlar berilgan bo’lsa, ularning hech bo‘lmaganda bittasi toq ekanligi aniqlansin.
44. Berilgan uchta sondan juftliklar hosil qilingan. Shu juftliklarning hech bo‘lmaganda bittasidagi sonlar o‘zaro teng bo‘lishi aniqlansin.
45. Berilgan uchta butun sonlar orasidan olingan juftliklardan hech bo’lmaganda bittasidagi sonlar ishoralari bilan farq qilishi aniqlansin.
46. Uch xonali son berilgan. Uning raqamlari o‘suvchi ketma-ketlik tashkil etishi aniqlansin.
47. Uch xonali son berilgan. Uning raqamlari o‘suvchi yoki kamayuvchi ketma-ketlik tashkil etishi aniqlansin.
48. To‘rt xonali son berilgan. Uni chapdan o‘ngga va o‘ngdan chapga o‘qiganda bir xil o‘qilishi aniqlansin.
49. *a, b, c* sonlar berilgan (*a≠0*). Bu sonlarni kvadrat tenglama koeffisientlari deb hisoblab, shu kvadrat tenglamaning haqiqiy yechimga ega ekanligi aniqlansin.
50. *x, y* sonlari berilgan. Ularni tekislikdagi nuqta koordinatalari deb hisoblab, shu nuqtaning 2-chorakda yotishi aniqlansin.
51. *x, y* sonlari berilgan. Ularni tekislikdagi nuqta koordinatalari deb hisoblab, ularning 2- yoki 3-chorakda yotishi aniqlansin.
52. Tekislikda nuqta *x* va *y* koordinatalari bilan berilgan. Shu nuqta (yuqori chap burchagi (x_1,y_1), quyi o‘ng burchagi (x_3,y_3) bo‘lgan, hamda tomonlari koordinata o‘qlariga parallel) to‘g‘ri burchakli to‘rtburchakning ichida yotishi yoki yotmasligi aniqlansin.
53. Uchta butun son berilgan bo’lsa, shu sonlarning uchburchakning tomonlarini tashkil etishi aniqlansin.
54. *a,b,c* butun sonlar berilgan bo‘lib, ular uchburchakning tomonlarini tashkil etsa, shu uchburchakning teng yonli ekanligi aniqlansin.
55. *a,b,c* butun sonlar berilgan bo‘lib, ular uchburchakning tomonlarini tashkil etsa, shu uchburchakning to‘g‘ri burchakli ekanligi aniqlansin.
56. Uch xonali son berilgan. Bu son raqamlarining har xil ekanligi aniqlansin.

4-amaliy ish topshiriq variantlari

32. K va N (N>0) butun sonlari berilgan. K sonini N marta ekranga chiqaring.

33. Ikkita **A va B** (**A<B**) butun sonlari berilgan. **A va B** oralig’ida joylashgan sonlarni o’sish tartibida (**A va B** larni ham hisobga olgan holda) ekranga chiqaring, hamda ular sonini hisoblang.

34. Ikkita **A va B** (**A<B**) butun sonlari berilgan. **A va B** oralig'ida joylashgan sonlarni kamayish tartibida (**A va B** larni ham hisobga olgan holda) ekranga chiqaring, hamda ular sonini hisoblang.
35. Bir kilogramm konfet narxini ifodalovchi haqiqiy son berilgan. 1 kg, 2 kg,...10 kg konfetlar baholarini hisoblang.
    A) 0.1, 0.2,...1 kg. konfetlar narxlarini hisoblab chiqaring.
    B) 1.2, 1.4, ...2 kg konfetlar narxlarini hisoblab chiqaring.
36. Ikkita **A va B** (**A<B**) butut sonlari berilgan. **A va B** sonlar orasidagi barcha butun sonlar yig'indisini toping(**A va B** larni ham hisobga olgan holda).
37. Ikkita **A va B** (**A<B**) butun sonlari berilgan. **A va B** sonlar orasidagi barcha butun sonlar kvadrati yig'indisini toping(**A va B** larni ham hisobga olgan holda).
38. Butun N va K sonlari berilgan. Quyidagi yig'indini toping

$$N^2 + (N+1)^2 + (N+2)^2 + \ldots\ldots + (N+K)^2$$

39. Butun N>0 soni berilgan. Quyidagi N ta ko'paytuvchilar ko'paytmasini toping.

$$1.1*1.2*1.3*\ldots\ldots$$

40. Butun N>0 soni berilgan. N ta ishorasi almashuvchi yig'indilardan iborat ifoda qiymatini toping (Shartli o'tish operatori ishlatilmasin).

$$1.1-1.2+1.3-\ldots$$

41. Butun N>0 toq soni berilgan. Quyidagi formula orqali hisoblanuvchi sonning kvadratini toping.

$$1+3+5+\ldots\ldots N$$

42. Har bir yig'indi joriy qiymati ekranga chiqarilsin (natijada 1 dan N gacha toq butun sonlar kvadratlari chiqarilsin).
43. Haqiqiy A va butun N>0 sonlari berilgan. A ning N darajasini toping (A soni N marta ko'paytirilsin)

$$A^N = A*A*\ldots\ldots*A$$

44. Butun N>0 soni berilgan. Ifoda qiymatini toping, bunda Shartli operator foydalanilmasin.

$$1-A+A^2-A^3+\ldots.+(-1)^N*A^N$$

45. Butun N>0 soni berilgan. Ko'paytmani toping (N-faktorial). O'zgaruvchilar haqiqiy tipli deb olinsin.

$$N!=1*2*\ldots. *N$$

46. Butun N>0 soni berilgan. Bir sikldan foydalanib yig'indini toping. O'zgaruvchilar haqiqiy tipli deb olinsin.

$$1!+2!+3!+\ldots+ N!$$

47. Butun N>0 soni berilgan. Bir sikldan foydalanib yig'indini toping.

$$1+1/(1!)+1/(2!)+1/(3!)+\ldots.+1/(\ N!)$$

O'zgaruvchilar haqiqiy tipli deb olinsin. (N!-faktorial. N!=1*2*.. N.)
Topilgan yig'indi e=exr(1) o'zgarmasning taqribiy qiymati bo'ladi.

48. Haqiqiy X va butun N>0 sonlari berilgan. Ifoda qiymatini toping.
$1+X+X^2/(2)+\ldots X^N/(N!)$
49. (N!=1*2*.. N) hisoblangan ifoda qiymati $u=e^x$ funktsiyaning x nuqtadagi taqribiy qiymati bo'ladi.
50. Haqiqiy X va butun N>0 sonlari berilgan. Ifoda qiymatini toping.
$X-X^3/(3!)+X^5/(5!)-.....+(-1)^N*X^{2*N+1}/((2*N)!)$
51. (N!=1*2*.. N) hisoblangan ifoda qiymati sinx funktsiyaning x nuqtadagi taqribiy qiymati bo'ladi.
52. Haqiqiy X (|X|<1) va butun N>0 sonlari berilgan. Ifoda qiymatini toping.
$1-X^2/(2!)+X^4/(4!)-.....+(-1)^N*X^{2*N+1}/((2*N)!)$
53. (N!=1*2*.. N) hisoblangan ifoda qiymati sos(x) funktsiyaning x nuqtadagi taqribiy qiymati bo'ladi.

54. Haqiqiy X (|X|<1) va butun N>0 sonlari berilgan. Ifoda qiymatini toping.
$X-X^2/2+X^3/4-.....+(-1)^{N-1}*X^{N+1}/N$
55. (N!=1*2*.. N) hisoblangan ifoda qiymati ln(x) funktsiyaning x nuqtadagi taqribiy qiymati bo'ladi.
56. Haqiqiy X (|X|<1) va butun N>0 sonlari berilgan. Ifoda qiymatini toping.
$X-X^3/3+X^5/5-.....+(-1)^N*X^{2*N+1}/(2*N+1)$
57. (N!=1*2*.. N) hisoblangan ifoda qiymati arstg(x) funktsiyaning x nuqtadagi taqribiy qiymati bo'ladi.
58. Haqiqiy X (|X|<1) va butun N>0 sonlari berilgan. Ifoda qiymatini toping.
$X+1*X^3/(2*3)+1*3*X^5/(2*4*5)+.....+1*3*\ldots.*(2*N-1)*$
$X^{2*N+1}/(2*4*\ldots.*(2*N)*(2*N+1))$
59. (N!=1*2*.. N) hisoblangan ifoda qiymati arssin(x) funktsiyaning x nuqtadagi taqribiy qiymati bo'ladi.
60. Haqiqiy X (|X|<1) va butun N>0 sonlari berilgan. Ifoda qiymatini toping.
$1+X/2-1*X^2/(2*4)+1*3*X^3/(2*4*6)-.....+(-1)^{N-1}*1*3*\ldots(2*N-3)*X^N/(2*4*\ldots*(2*N))$
61. (N!=1*2*.. N) hisoblangan ifoda qiymati $\sqrt{1+x}$ funktsiyaning x nuqtadagi taqribiy qiymati bo'ladi.

5-amaliy ish topshiriqlari

Berilgan x, y va z sonlari uchun formulaning natijasi topilsin (1-10 - misollar).
Berilgan x, y va z sonlarining berilgan qiymatlari uchun formulaning natijasi topilsin (11-32- misollar).

33.	x=14.26, y=-1.22, z=0.5×10^{-2}, $t=\frac{2\cos\left(x-\frac{\pi}{6}\right)}{0.5+\sin^2 y}\left(1+\frac{z^2}{3-z^2/5}\right)$ Natija: t=0.564849
34.	x=-4.5, y=0.75x10^{-4}, z=0.845x10^{2}, $u=\frac{\sqrt[3]{8+\lvert x-y\rvert^2+1}}{x^2+y^2+2}-e^{\lvert x-y\rvert}\left(tg^2 z+1\right)^x.$ Natija: u=-55.6848.
35.	x=-15.246, y=4.642x10^{-2}, z=20.001x10^{2}, $\alpha=\ln\left(y^{-\sqrt{\lvert x\rvert}}\right)\left(x-\frac{y}{2}\right)+\sin^2 arctg(z).$ Natija: α=-182.036
36.	x=0.1722, y=6.33, z=3.25x10^{-4} , $\gamma=5arctgx-\frac{1}{4}\arccos x\frac{x+3\lvert x-y\rvert+x^2}{\lvert x-y\rvert z+x^2}.$ Natija: γ=-172.025
37.	x=1.825x10^{2}, y=18.225, z=-3.298x10^{-2}, $\psi=\left\lvert x^{\frac{y}{x}}-\sqrt[3]{\frac{y}{x}}\right\rvert+(y-x)\frac{\cos y-\frac{z}{(y-x)}}{1+(y-x)^2}.$ Natija: ψ=1.2131
38.	x=6.251, y=0.827, z=25.001, $b=y^{\sqrt[3]{\lvert x\rvert}}+\cos^3 y\frac{\lvert x-y\rvert\left(1+\frac{\sin^2 z}{\sqrt{x+y}}\right)}{e^{\lvert x-y\rvert}+x/2}.$ Natija: b=0.7121
39.	X=17.421, y=10.365x10^{-3}, z=0.828x10^{5}, $f=\frac{\sqrt[4]{y+\sqrt[3]{x-1}}}{\lvert x-y\rvert(\sin^2 z+tgz)}.$ Natija: f=0.33056
40.	X=2.444, y=0.869x10^{-2}, z=-0.13x10^{3}, $h=\frac{x^{y+1}+e^{y-1}}{1+x\lvert y-tgz\rvert}(1+\lvert y-x\rvert)+\frac{\lvert y-x\rvert^2}{2}-\frac{\lvert y-x\rvert^3}{3}.$

	Natija: h=-0.49871
41.	X=1, y=1, z=3 $a=(1+y)\dfrac{x+y/(x^2+4)}{e^{-x-2}+1/(x^2+4)};$ $b=\dfrac{1+\cos(y-2)}{x^4/2+\sin^2 z}.$ Natija: a=9.608184; b=2.962605
42.	X=3, y=4, z=5, $a=\dfrac{1+\sin^2(x+y)}{2+\left\|x+2x/(1+x^2y^2)\right\|}+x;$ $b=\cos^2(arctg\dfrac{1}{z}).$ Natija: a=3.288716; b=0.9615385
43.	$\gamma=5arctgx-\dfrac{1}{4}\arccos x\dfrac{x+3\left\|x-y\right\|+x^2}{\left\|x-y\right\|z+x^2}.$
44.	$\psi=\left\|x^{\frac{y}{x}}-\sqrt[3]{\dfrac{y}{x}}\right\|+(y-x)\dfrac{\cos y-\dfrac{z}{(y-x)}}{1+(y-x)^2}.$
45.	$b=y^{\sqrt[3]{\left\|x\right\|}}+\cos^3 y\dfrac{\left\|x-y\right\|\left(1+\dfrac{\sin^2 z}{\sqrt{x+y}}\right)}{e^{\left\|x-y\right\|}+x/2}.$
46.	$f=\dfrac{\sqrt[4]{y+\sqrt[3]{x-1}}}{\left\|x-y\right\|(\sin^2 z+tgz)}.$
47.	$h=\dfrac{x^{y+1}+e^{y-1}}{1+x\left\|y-tgz\right\|}(1+\left\|y-x\right\|)+\dfrac{\left\|y-x\right\|^2}{2}-\dfrac{\left\|y-x\right\|^3}{3}.$
48.	$u=\sqrt{e^x}\sqrt{\cos^2(\frac{x}{4})}+2\ln x$
49.	$u=\sqrt{tg^2(\frac{x}{5})}+2e^x$
50.	$y=\log_4 x+\sqrt{e^3}+\cos\dfrac{x}{2}$
51.	$y=\sec x+\sqrt[4]{e^3}+\dfrac{x}{2}$
52.	$u=\sqrt{\ln x}+2\left\|e^{x+2}-5x\right\|$

53.	$u = \sqrt{\ln x} + 2\,\lvert e^{x+2} - 15\rvert$
54.	$y = 2{,}5\sqrt{x^4} + Sin\dfrac{4}{x}$
55.	$y = e^{\sin x^2} + \dfrac{x}{2e^{x-1}}$
56.	$u = \sqrt{e^x}\sqrt{Sin(\dfrac{x}{2})} + 2\,\lvert x-1\rvert$
57.	$u = \sqrt{\ln x} + 2\,\lvert e^{x+2} - 5x\rvert$
58.	$u = \sqrt{Sin\,\lvert x-8\rvert} + e^{x+5} + x^4$
59.	$u = \sqrt{\lvert 18-x\rvert} + e^{x+2} - 5x$
60.	$u = \sqrt{Sin\,\lvert x-8\rvert} + e^{x+5} + x^4$
61.	$a = \dfrac{\lvert 4-13x\rvert + e^x}{1+\ln x}$
62.	$u = \sqrt{Cos^2(\dfrac{x}{\lvert 2-x\rvert})} + 2e$
63.	$u = \sqrt{\ln x}\sqrt{Sin(\dfrac{x}{2})} + 2\,\lvert e^x - 1\rvert$
64.	$u = \sqrt{e^x}\sqrt{Sin(\dfrac{x}{2})} + 2\,\lvert x-1\rvert$

6-amaliy ish topshiriq variantlari

30. Satr berilgan. Undagi raqamlar soni hisoblansin.
31. Satr berilgan. Uning belgilari teskari tartibda chop etilsin (telefon -> nofelet).
32. Bo‘sh bo‘lmagan satr berilgan. Satrda joylashgan belgilarning orasiga bittadan bo‘sh joy qo‘yishdan hosil bo’lgan satr chop etilsin.
33. Satr berilgan. Undagi lotin alfavitining bosh harflari soni hisoblansin.
34. Satr berilgan. Satrga kirmagan barcha lotin harflarining soni hisoblansin.
35. Satr berilgan. Berilgan satrdagi barcha bosh harflar kichik harflarga aylantirilsin.
36. Satr berilgan. Berilgan satrdagi barcha kichik harflar bosh harflarga almashtirilsin.
37. Satr berilgan. Berilgan satrdagi barcha bosh harflar kichik harflarga, kichik harflar esa bosh harflarga almashtirilsin.
38. Satr berilgan. Agar satr butun sondan iborat bo‘lsa *1*, haqiqiy sondan iborat bo‘lsa *2*, satrni son ko‘rinishiga o‘tkazib bo‘lmasa *0* chop etilsin.
39. Butun musbat son berilgan. Bu sonni tasvirlovchi raqamlardan iborat belgilar o‘ngdan chapga qaragan tartibda chop etilsin.
40. Butun musbat sonni tasvirlovchi satr berilgan. Bu sonning raqamlari yig‘indisi hisoblansin.
41. “<*raqam1*>±<*raqam2*>±...±<*raqamN*>” ushbu arifmetik ifodani tasvirlovchi satr berilgan. Satrdagi “±”belgilar o‘rniga, “+” yoki “-” amali qo’yilsin va ifodaning qiymati chiqarilsin.
42. Satrda butun musbat sonning ikkilik sanoq sistemasidagi ko‘rinishi tasvirlangan. Bu sonning 10 lik sanoq sistemasidagi ko‘rinishi chop etilsin.
43. Satrda butun musbat sonning 10 liksanoq sistemasidagi ko‘rinishi tasvirlangan. Bu sonning 2 lik sanoq sistemasidagi ko‘rinishi chop etilsin.
44. *n(n>0)* butun soni va satr berilgan. *n* uzunlikka teng bo‘lgan satr quyidagi ko‘rinishda aniqlanadi: satr uzunligi *n* dan katta bo‘lsa, uning o`ng tomonidan ortiqcha belgilar olib tashlansin, agar satr uzunligi *n* dan kichik bo‘lsa, uning o`ng tomoniga nuqtalar qo‘shilsin. Hosil qilingan satr chop etilsin.
45. Butun musbat n_1, n_2 sonlar va 2 ta satr berilgan. Bu satrlardan foydalanib yangi satr hosil qilinsin: satrning dastlabki n_1 ta belgi ikkinchi satrning bosh qismidan, oxirgi n_2 ta belgi birinchi satrning oxirgi qismidan (chapdan o’ngga qarab) iborat bo‘lsin.
46. Satr va ixtiyoriy belgi berilgan. Agar satrda berilgan belgi uchrasa, u ikkilantirilib satr chop etilsin.
47. Ixtiyoriy belgi va 2 ta satr berilgan. Ikkinchi satrda berilgan belgi uchrasa uning oldiga birinchi satr joylashtirilsin.
48. Ixtiyoriy belgi va 2 ta satr berilgan. Birinchi satrda belgi uchrasa, shu belgi ortidan ikkinchi satr joylashtirilsin.
49. 2 ta satr berilgan. Agar birinchi satr ikkinchi satrda mavjud bo‘lsa *1 (rost)* aks holda *0 (yolg’on)* qiymat chiqarilsin.
50. 2 ta satr berilgan. Ikkinchi satrda birinchi satrning necha marta uchrashi aniqlansin.
51. 2 ta satr berilgan. Ikkinchi satrdan birinchi satr bilan ustma-ust tushuvchi 1-qism satr o‘chirilsin. Agar ikkinchi satrda birinchi satr topilmasa satr

o‘zgarishsiz chop etilsin.

52. 2 ta satr berilgan. Ikkinchi satrdan birinchi satr bilan ustma-ust tushuvchi oxirgi qism satr o‘chirilsin. Agar *s* satrda *Birinchi* satr topilmasa *s* satr o‘zgarishsiz chop etilsin.
53. 2 ta satr berilgan. Ikkinchi satrdan birinchi satr bilan ustma-ust tushuvchi barcha qism satrlar o‘chirilsin. Agar ikkinchi satrda birinchi satr topilmasa ikkinchi satr o‘zgarishsiz chop etilsin.
54. 3 ta satr berilgan. Birinchi satrdagi dastlabki uchragan ikkinchi qism satr uchinchi qism satr bilan almashtirilsin. Agar birinchi satrda ikkinchi satr topilmasa birinchi satr o‘zgarishsiz chop etilsin.
55. 3 ta satr berilgan. Birinchi satrdagi oxirgi uchragan ikkinchi qism satr uchinchi qism satr bilan almashtirilsin. Agar birinchi satrda ikkinchi satr topilmasa birinchi satr o‘zgarishsiz chop etilsin.
56. 3 ta satr berilgan. Birinchi satrda uchragan barcha ikkinchi qism satrlaruchinchi qism satr bilan almashtirilsin. Agar birinchi satrda ikkinchi satr topilmasa birinchi satr o‘zgarishsiz chop etilsin.
57. Kamida 2 ta bo‘sh joyga ega satr berilgan. Berilgan satrdagi 1- va 2- bo‘sh joylar orasida joylashgan qism satr chiqarilsin.
58. Kamida 2 ta bo‘sh joyga ega satr berilgan. Berilgan satrdagi 1- va oxirgi bo‘sh joylar orasida joylashgan qism satr chiqarilsin.
59. Bo‘sh joylar bilan ajratilgan, so‘zlardan tuzilgan satr berilgan. Satrdagi so‘zlarning har biri teskari tartibda joylashtirilib, chop etilsin.

Foydalanilgan asosiy darsliklar va o'quv qo'llanmalar ro'yxati

Asoiy darsliklar va o'quv qo'llanmalar

25. Н. А. Прохоренок, В. А. Дронов. “Python 3 и PyQt 5. Разработка приложений”. БХВ-Петербург, 2016. — 832 с.: ил.
26. Д.Ю.Федоров. “Основы программирования на примере языка Python” Учебное пособие. Санкт-Петербург. 2-е изд., перераб. и доп. — М.: Издательство Юрайт, 2019. — 161 с.
27. Любанович Билл. “Простой Python. Современный стиль программирования”. — СПб.: Питер, 2016. — 480 с.: ил. — (Серия «Бестселлеры O’Reilly»).
28. Васильев А. Н. “Python на примерах. Практический курс по программированию”. - СПб.: Наука и Техника, 2016. - 432 с.: ил.
29. Рашка С. “Python и машинное обучение”/ пер.с англ. А. В. Логунова. - М.: ДМК Пресс, 2017. - 418 с.: ил.
30. Шолле Франсуа. “Глубокое обучение на Python”. — СПб.: Питер, 2018. — 400 с.: ил.
31. Мэтиз Эрик. “Изучаем Python. Программирование игр, визуализация данных, веб-приложения”. СПб.: Питер, 2017. — 496 с.: ил.
32. JOHNE. GRAYSON. “Python and TkinterProgramming”. MANNING Greenwich (74° w. long.) ©2000 by Manning Publications Co.

Qo'shimcha adabiyotlar

Internet ma'lumotlarini olish mumkin bo'lgan saytlar:

www.ziyonet.uz;

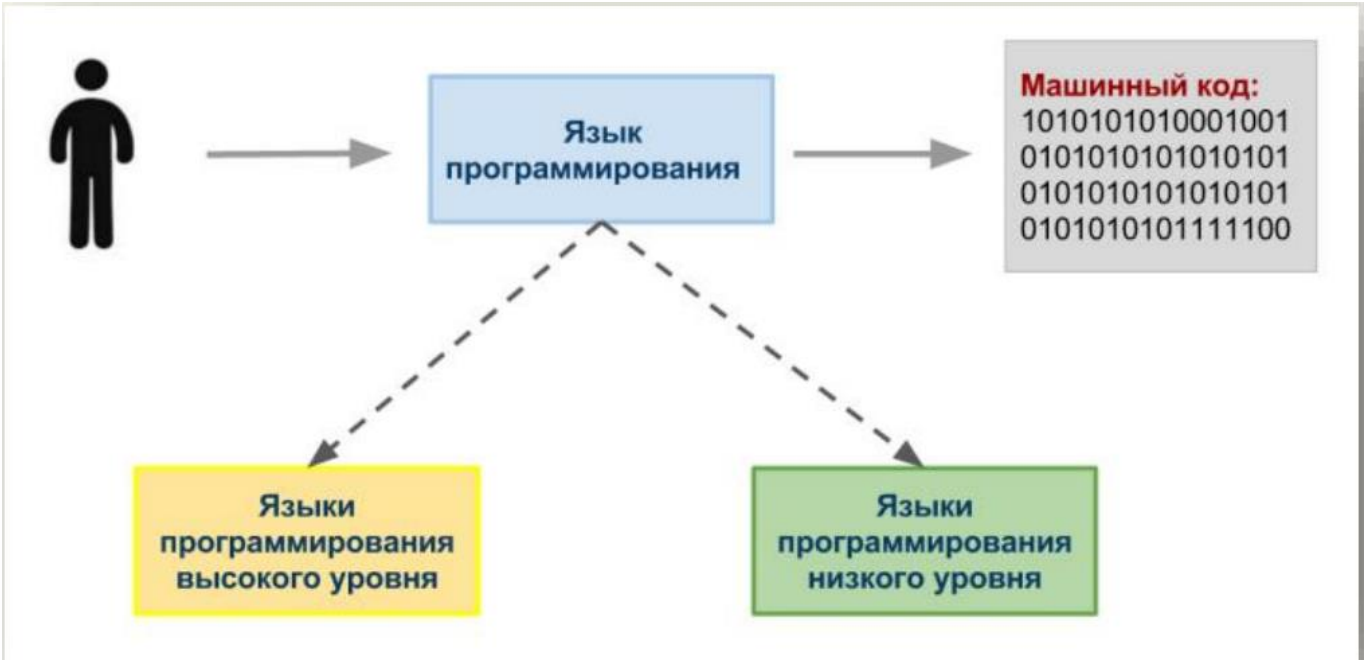
www.google.com;

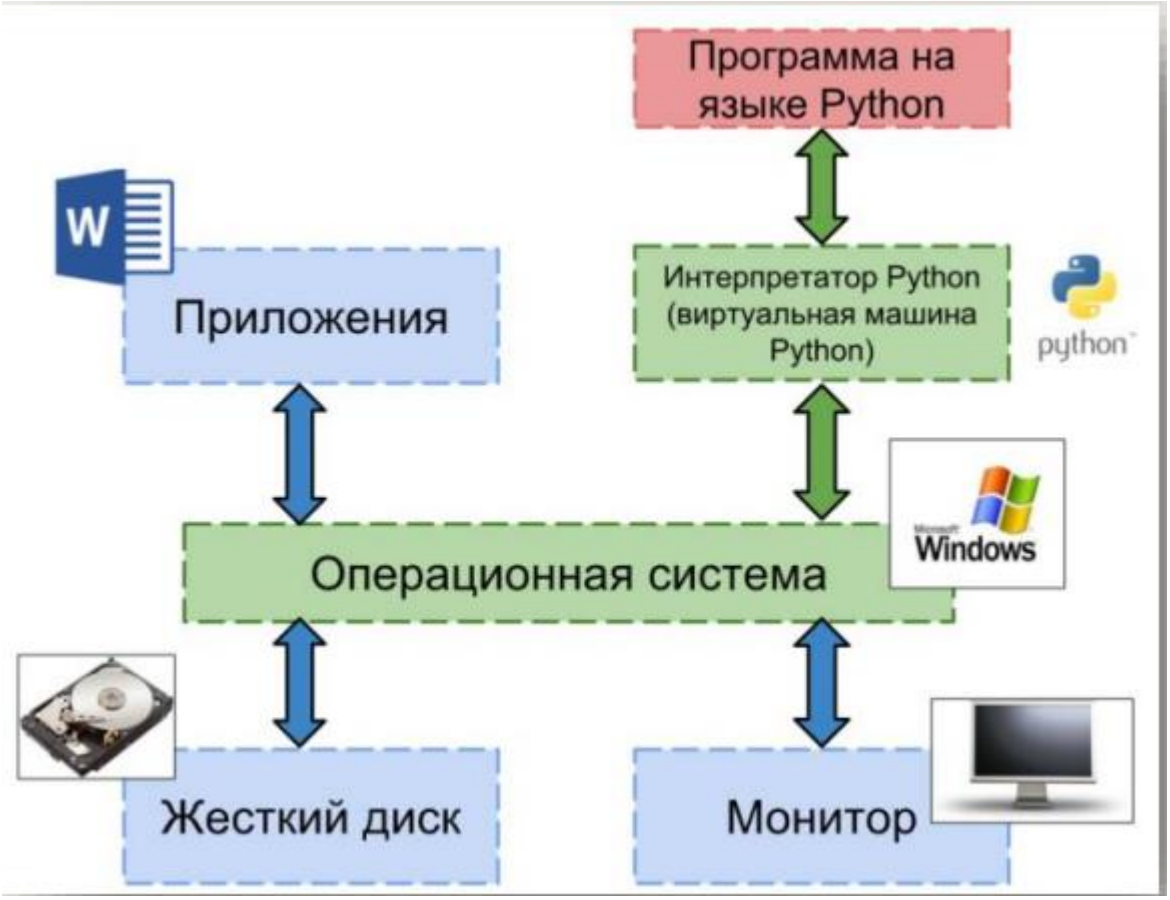
www.python.org;

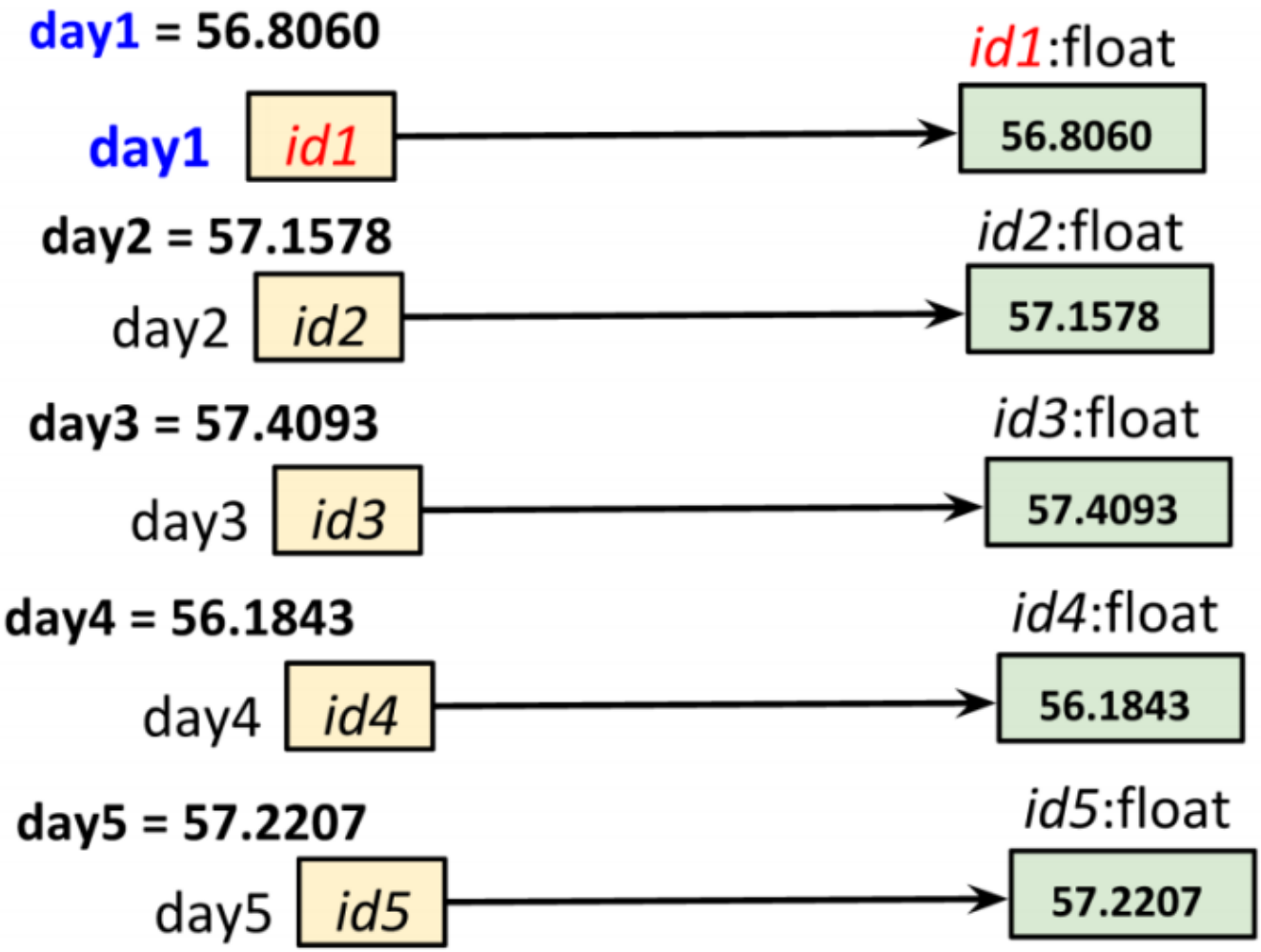
www.qt.com;

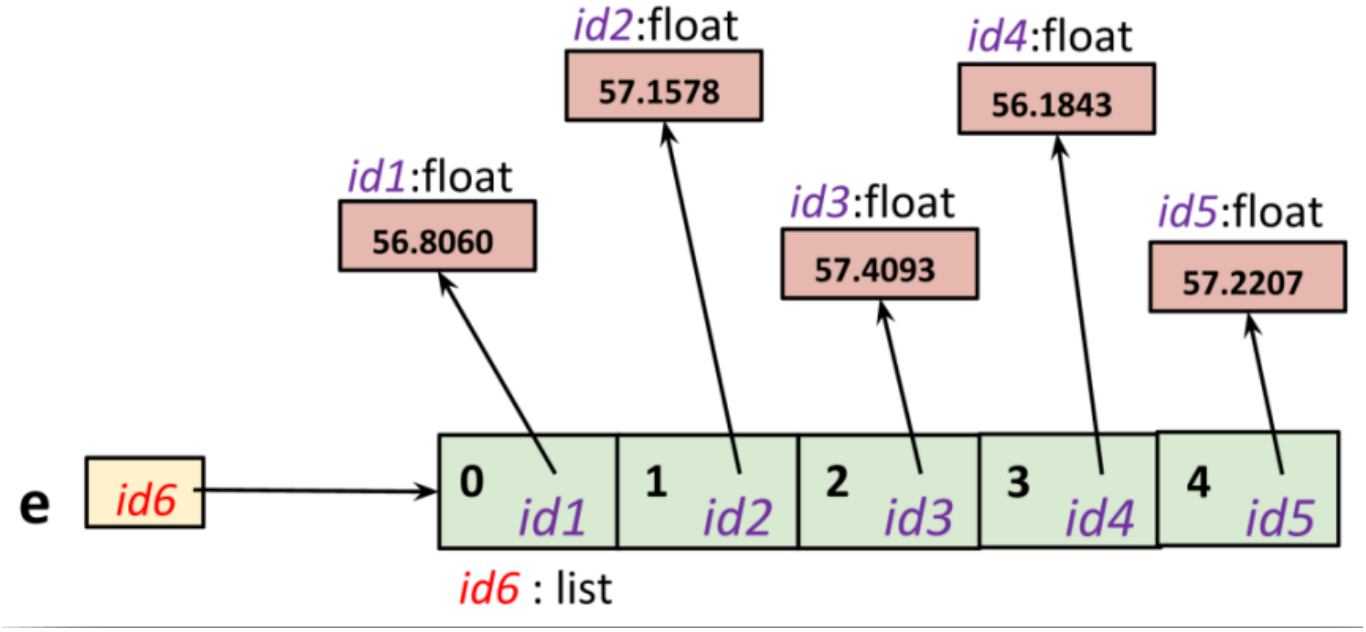
www.intuit.ru.

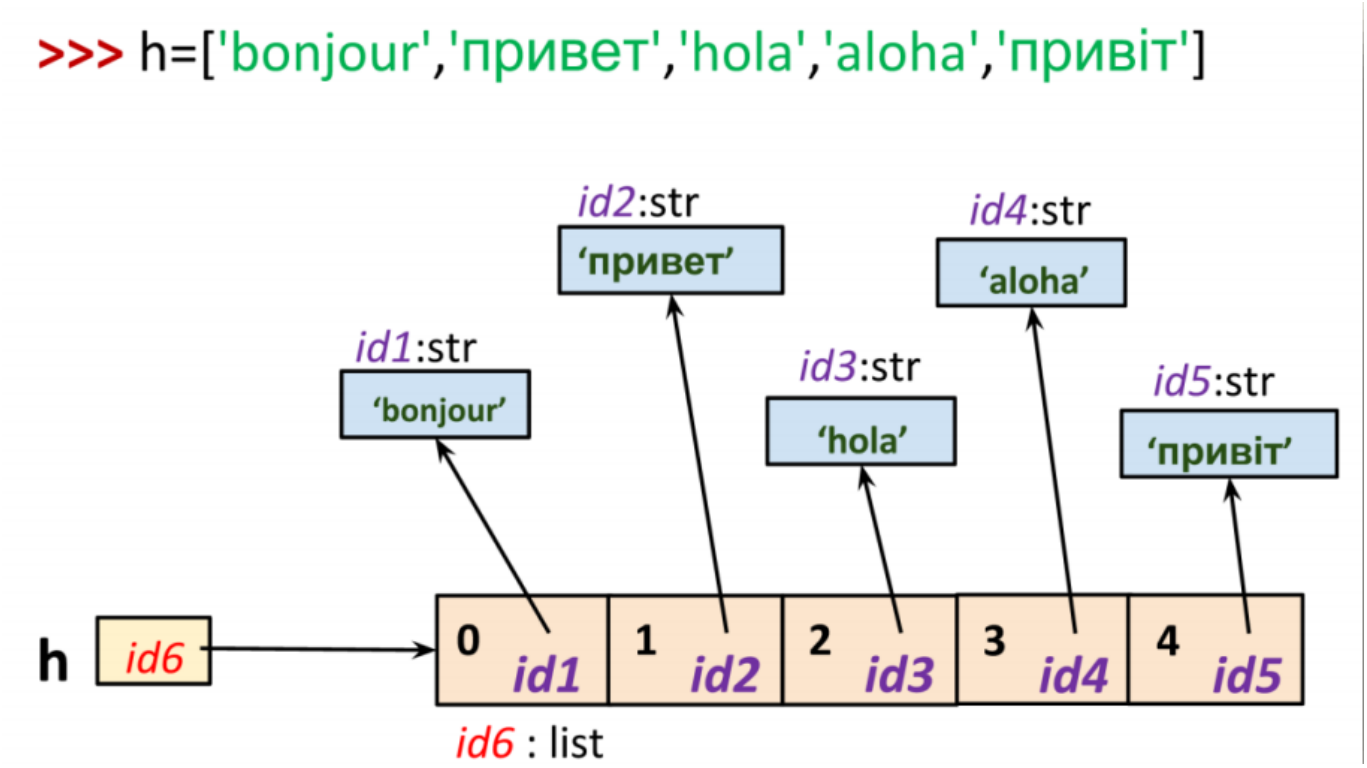
Tarqatma materiallar.

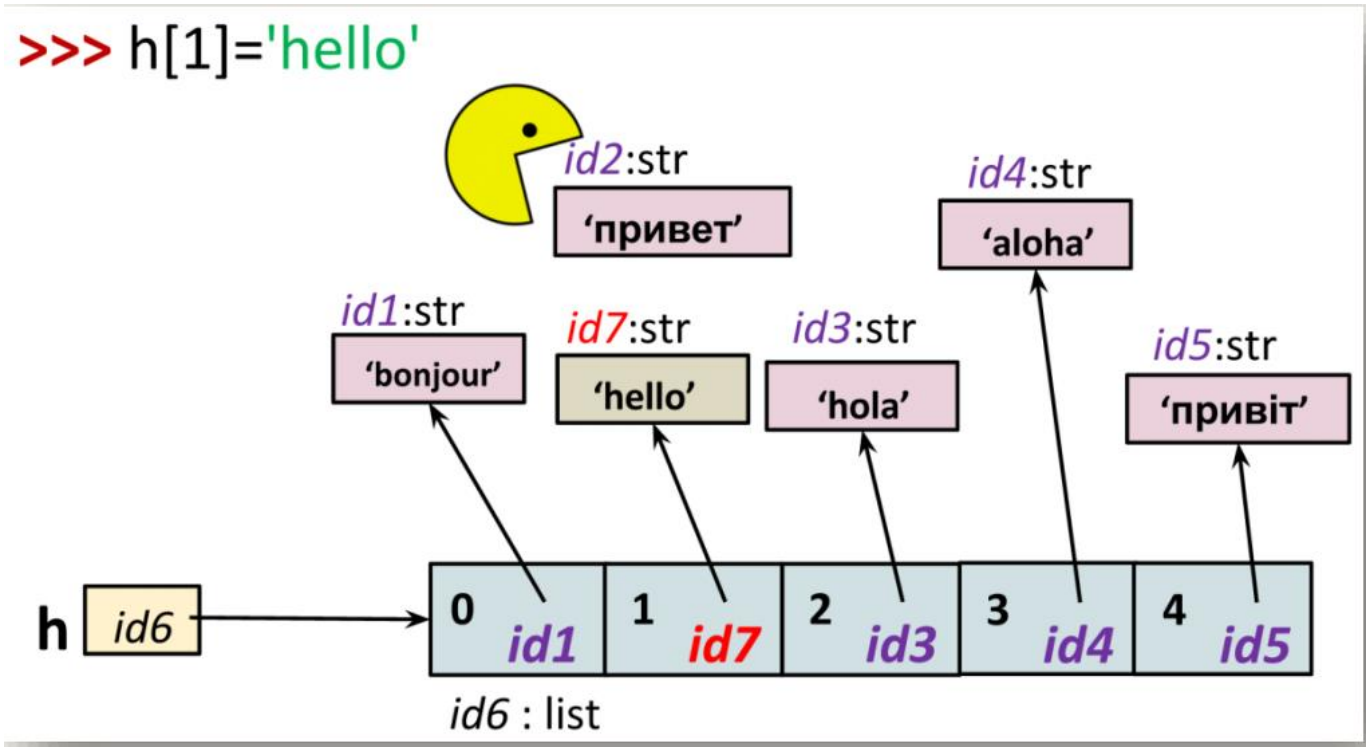


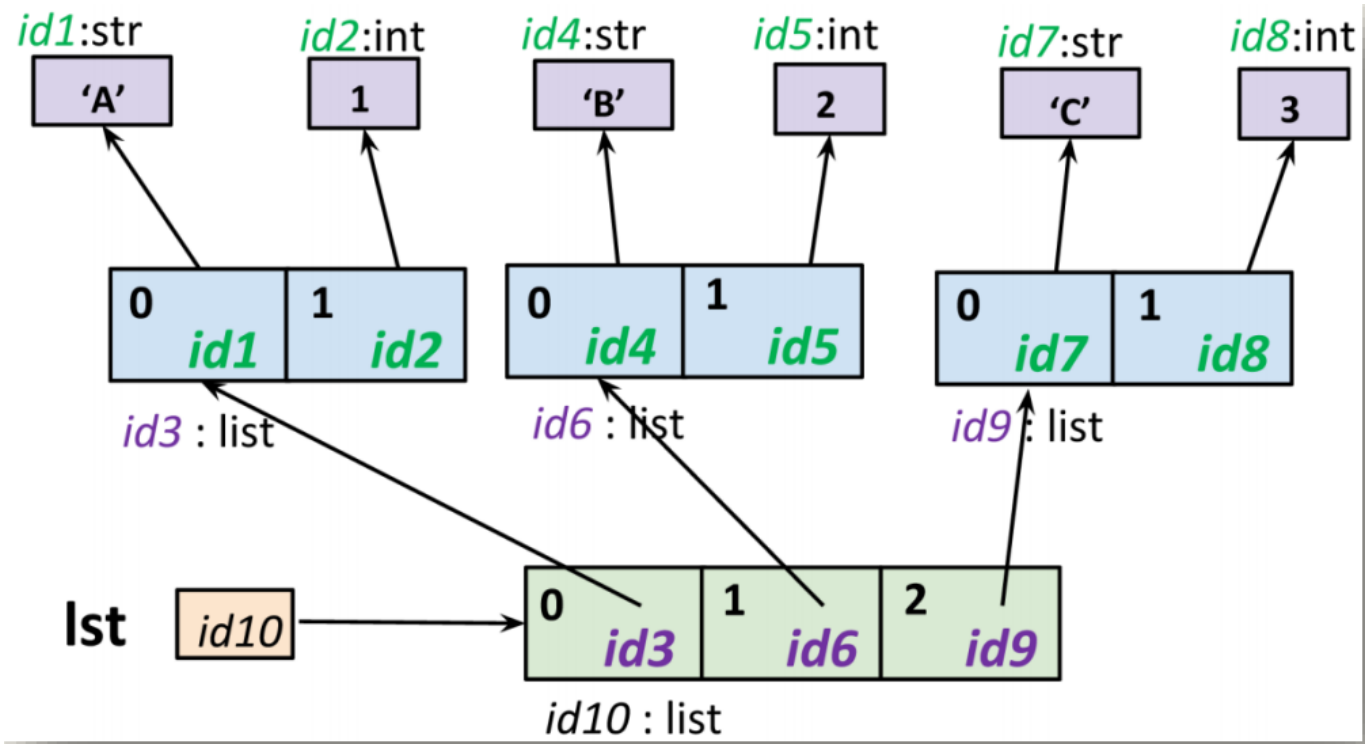

Программа на языке Python
Приложения
Интерпретатор Python (виртуальная машина Python)
python
Операционная система
Windows
Жесткий диск
Монитор

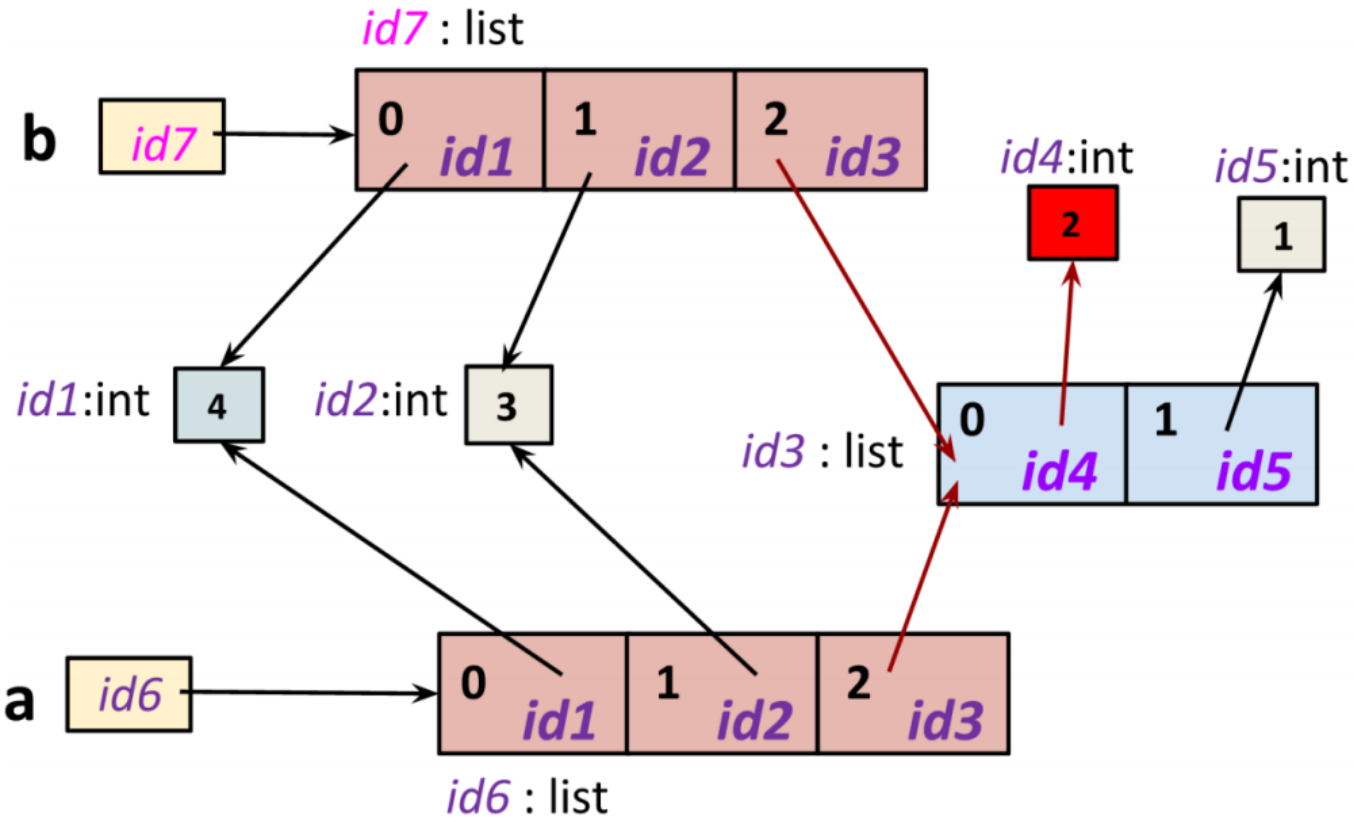

day1 = 56.8060
id1:float
day1
id1
56.8060
day2 = 57.1578
id2:float
day2
id2
57.1578
day3 = 57.4093
id3:float
day3
id3
57.4093
day4 = 56.1843
id4:float
day4
id4
56.1843
day5 = 57.2207
id5:float
day5
id5
57.2207

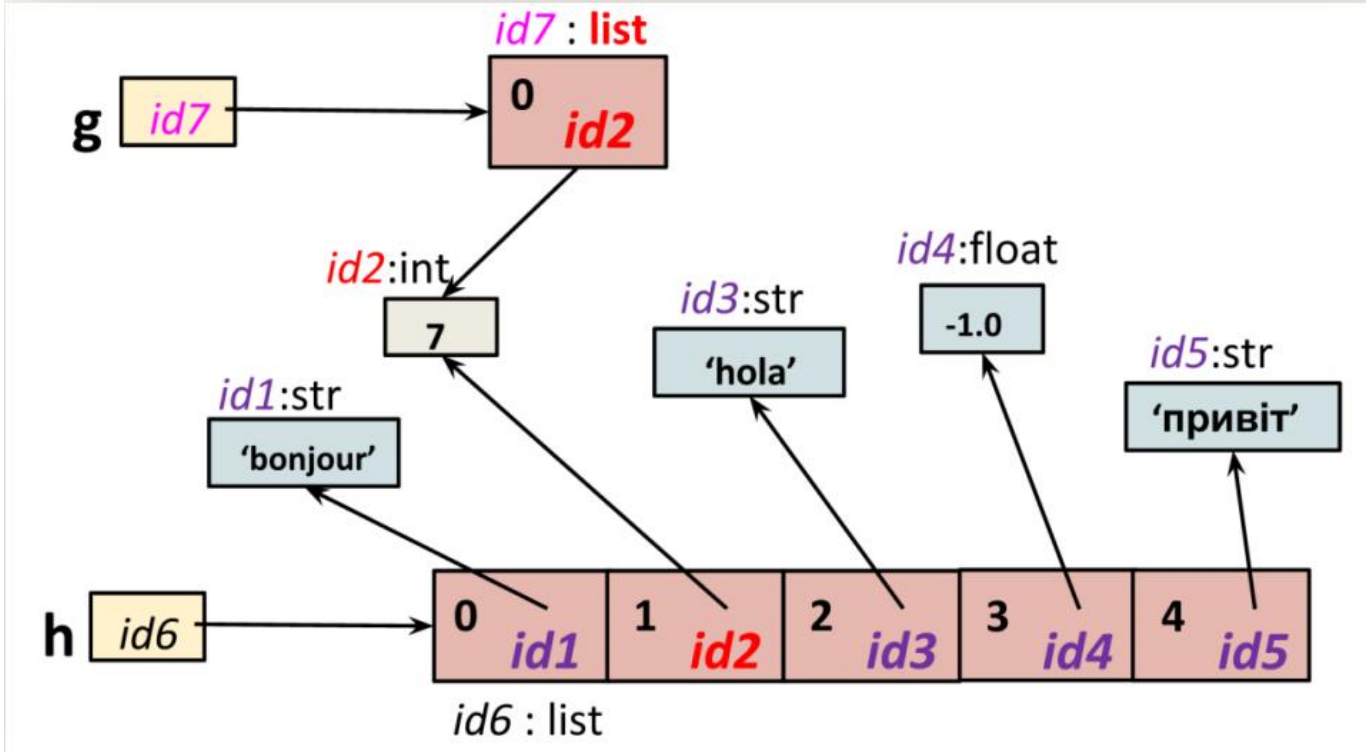

id2:float
57.1578
id4:float
56.1843
id1:float
56.8060
id3:float
57.4093
id5:float
57.2207
e
id6
0
id1
1
id2
2
id3
3
id4
4
id5
id6 : list

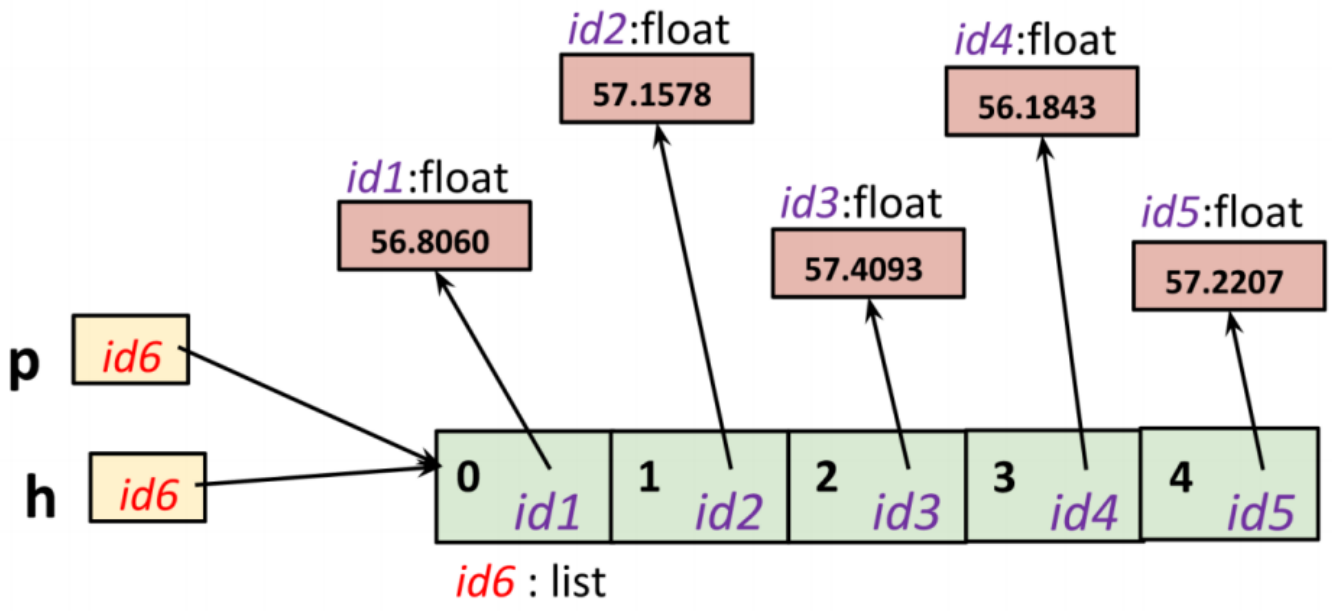

>>> h=['bonjour','привет','hola','aloha','привіт']
id2:str
'привет'
id4:str
'aloha'
id1:str
'bonjour'
id3:str
'hola'
id5:str
'привіт'
h
id6
0
id1
1
id2
2
id3
3
id4
4
id5
id6 : list

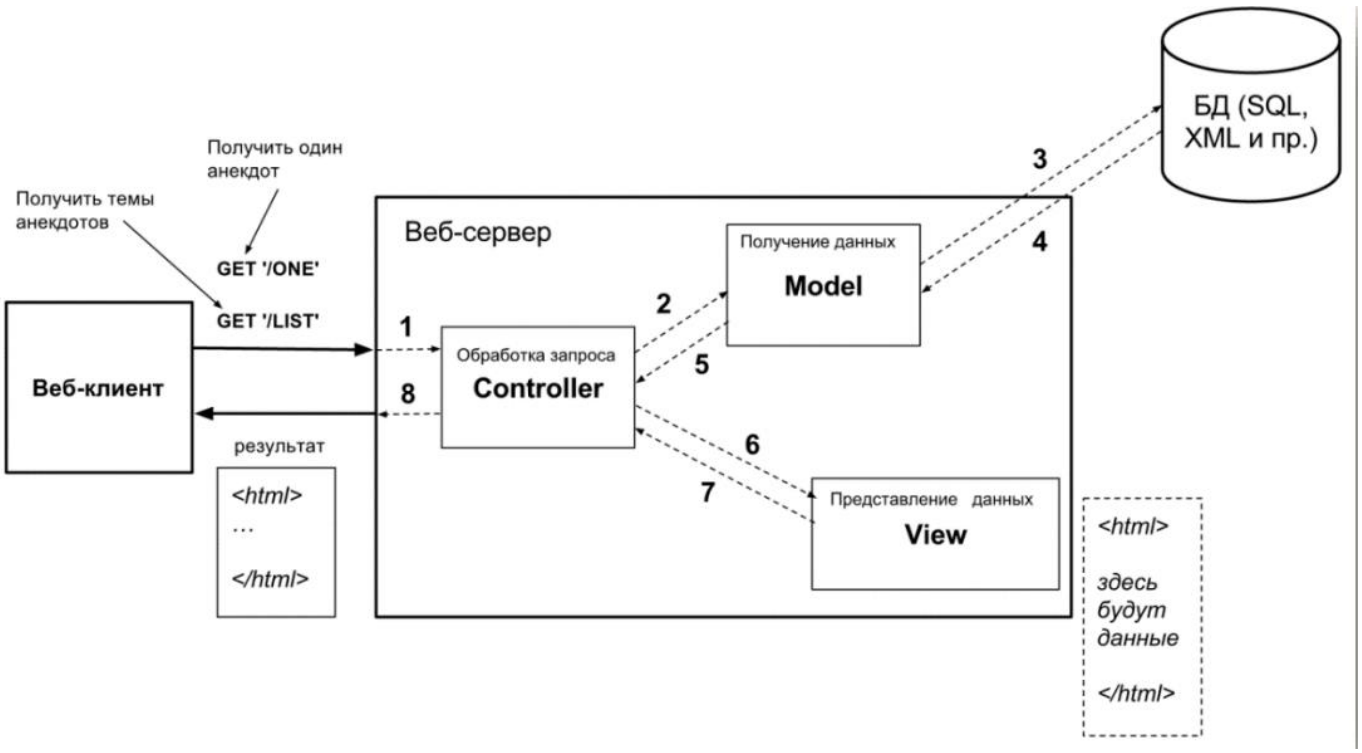

>>> h[1]='hello'
id2:str
'привет'
id4:str
'aloha'
id1:str
'bonjour'
id7:str
'hello'
id3:str
'hola'
id5:str
'привіт'
h
id6
0 id1
1 id7
2 id3
3 id4
4 id5
id6 : list


id1:str
'A'
id2:int
1
id4:str
'B'
id5:int
2
id7:str
'C'
id8:int
3
0 id1
1 id2
0 id4
1 id5
0 id7
1 id8
id3 : list
id6 : list
id9 : list
lst
id10
0 id3
1 id6
2 id9
id10 : list


id7 : list
b
id7
0 id1
1 id2
2 id3
id4:int
2
id5:int
1
id1:int
4
id2:int
3
id3 : list
0 id4
1 id5
a
id6
0 id1
1 id2
2 id3
id6 : list


id7 : list
g
id7
0 id2
id2:int
7
id4:float
-1.0
id3:str
'hola'
id5:str
'привіт'
id1:str
'bonjour'
h
id6
0 id1
1 id2
2 id3
3 id4
4 id5
id6 : list


id2:float
57.1578
id4:float
56.1843
id1:float
56.8060
id3:float
57.4093
id5:float
57.2207
p
id6
h
id6
0
id1
1
id2
2
id3
3
id4
4
id5
id6 : list


БД (SQL, XML и пр.)
Получить один анекдот
Получить темы анекдотов
GET '/ONE'
GET '/LIST'
Веб-сервер
Получение данных
Model
Обработка запроса
Controller
Веб-клиент
результат
<html>
…
</html>
Представление данных
View
<html>
здесь будут данные
</html>
1
2
3
4
5
6
7
8

Testlar

Python 3.0 versiayasi qaysi yilda chiqarilgan? {
= 2008
~ 2004
~ 1996
~ 2005}

Python 1.0 versiayasi qaysi yilda chiqarilgan? {
=1990
~2008
~1996
~2005}

Python yaratuvchisi kim? {
=Gvido
~Mante Python
~Tim Berners Li
~Yan Guk}

Pythonga hos bo’lmagan hususiyat qaysi? {

=Komplyatsiya qilinadi.

~Ko'plab platformalarda hech qanday o'zgartirishlarsiz ishlay oladi.

~Dasturni yozish davomida quyi darajadagi detallarni, misol uchun xotirani boshqarishni hisobga olish shart emas.

~Interpretatsiya (Интерпретируемый) qilinadigan til.}

Pythonda bloklarni belgilashda gulli (figurali) qavslardan foydalanish mumkinmi?{
=Mumkin, lekin tavsiya qilinmaydi
~Ha, mumkin
~Mumkin emas
~Bazan mumkin, bazan mumkin emas}

Pythonda bir qatorli izoh to’g’ri berilgan javobni aniqlang.{
=# bu izox;
~% bu izox;
~* bu izox;
~$ bu izox;}

Pythonda dastur natijasini chop etish uchun qaysi operator ishlatiladi?{
=print()
~cout()
~echo()
~writeln() }

Pythonda ma’lumotlarni kiritish uchun qaysi operator ishlatiladi?{
=input ()
~print()
~cin()
~edit() }

Pythonda o’zgaruvchilar qaysi kalit so’z bilan e’lon qilinadi?{
=Pythonda o’zgaruvchi e’lon qilinmaydi
~var
~let
~input}

Pythonda mantiqiy amal natijasi qanday boladi?{
=True yoki False(1 yoki 0)
~Begin yoki end
~int
~float}

Pythonda suzuvchi vergulli son turi qaysi so’z bilan ifoda qilinadi?{
=float
~decimal
~int
~bool }

Pythonda ro’yxat turi qaysi so’z bilan ifoda qilinadi?{
=list
~dict
~tuple
~range }

Pythonda lug’at turi qaysi so’z bilan ifoda qilinadi?{
=dict
~list
~tuple

~range }

Pythonda o’zgaruvchida qanday qiymat saqlanadi?{
=Obyektga havola
~Son
~Yacheyka
~Satr }

Pythonda ma’lumotni butun son tipiga o’zgartirish tug’ri ko’rsatilag javobni aniqalang. {
=int(<ma’lumot>)
~float(ma’lumot)
~str(ma’lumot)
~bool(ma’lumot)}

Pythonda o’zgaruvchini o’chirish konstruksiyasi qanday bo’ladi? {
=del <o’zgaruvchi>[, …, <o’zgaruvchi_N>]
~delete <o’zgaruvchi>[, …, <o’zgaruvchi_N>]
~rd <o’zgaruvchi>[, …, <o’zgaruvchi_N>]
~clear <o’zgaruvchi>[, …, <o’zgaruvchi_N>] }

Pythonda “ / / ” amali qanaday ma’no anglatadi?{
=kami bilan yaxlitlanadigan bo’lish
~qoldiqli bo’lish
~darajaga ko’tarish
~bo’lish}

Pythonda “%” amali qanaday ma’no anglatadi?{
=qoldiqli bo’lish
~kami bilan yaxlitlanadigan bo’lish
~darajaga ko’tarish
~bo’lish}

Quyidagilardan qaysi biri o’z;ashtirish amali bo’lolmaydi?{
= **
~ +=
~ -=
~ /=}

Mantiqiy (ikkilik) VA amali qaysi belgi bilan abelgilanadi?{
= &

~ |
~ ^
~ << }

Pythonda >>> True + 2 amal natijasi nimaga teng?{
=3
~“True+2”
~2
~0}

Pythonda bool (20) funksiya qanday qiymat qaytaradi?{
=True
~0
~False
~20}

Pythonda bool (“0”) funksiya qanday qiymat qaytaradi?{
=True
~False
~0
~20}

Pythonda qaysi operator ikkita o’zgaruchi bir obyektga havola qilinganini tekshiradi?{
=is
~in
~if
~not}

Pythonda qaysi operator qiymatni qator yoki ketma-ketlikka tegishlilikka tekshiradi?{
=in
~is
~if
~not}

Pythonda “VA” mantiqiy amalai qaysi so’z bilan ifoda qilinadi?{
=AND
~OR
~NOT
~IF}

Pythonda “YOKI” mantiqiy amalai qaysi so’z bilan ifoda qilinadi?{
=OR
~AND
~NOT
~IF}

Javoblar orasidan qaysi operator Pythonda takrorlanishni ta’minlaydi? {
=while
~if
~switch
~numerate}

Pythonda range funksiyasi formati to’g’ri berilgan javobni belgilang?{
=range ([<boshlang’ich qiymat>,] <Tugash> [, <Qadam>])
~range ([<boshlang’ich qiymat>,] [<Qadam>], <Tugash>)
~range (<Qadam>)
~range ([<Qadam>,] [<boshlang’ich qiymat>,] <Tugash>)}

Pythonda range() funksiyasi qanday vaziafa bajaradi?{
=Ketma-ketlikni ifodalaydi
~Sonlarni yigindisini qayataradi
~Darajaga ko’taradi
~Kvadrat ildiz chiqaradi}

Pythonda for operatori qanaday maqsadda qo’llaniladi?{
=takrorlanishlarni hisoblash va ketma-ketliklar bilan ishlash uchun foydalaniladi;
~tarmoqlanishlarni amaliga oshirishda va sonlar bilan ishlashda qo’llaniladi;
~murakkab ifodalarni yozishda foydalaniladi;
~ketma-ketliklarni ifodalsh uchun va tarmoqlarga ajratsish uchun ishlatiladi;}

Pythonda ushbu dastur natijasi nimaga teng?

```
s=0
for i in range(1,10,2):
        s=s+i
print(s) {
```

=25
~10
~45
~0}

Pythonda qaysi operator sikldagi barcha instruksiyalar bajarilmay, siklning navbatdagi iterasiyasiga o’tish imkonini beradi{
=continue
~break
~for
~while}

Pythonda qaysi operator sikl bajarilishini zudlik bilan to’xtatish imkonini beradi {
=break
~continue
~for
~while}

Pythonda while operatori qanday maqsadda qo’llaniladi?{
=takrorlanishlarni tashkil etish uchun
~tarmoqlanishi tashkil etish uchun
~ichma-ich takrorlanishni tashkil etish uchun
~shartsiz takrorlanishlarni tashkil etish uchun}

Pythonda ushbu dastur natijasi nimaga teng?

```
i=1
s=1
while i<10:
        s+=i
        i+=1
print(s){
```

=46
~45
~1
~0}

Pythonda ushbu dastur natijasi nimaga teng?

```
i=1
s=0
while i<10:
  if i==5: break
  s+=i
  i+=1
print(s) {
```

=10
~45

~46
~0}

Python 3 butun son tipi qaysi so’z bilan belgilanadi?{
=int
~float
~decimal
~complex}

Python 3 kasr sonni qanday ifodalash mumkin?{
=Fraction() funksiyasi yordamida
~complex() funksiyasi yordamida
~float xizmatchi so’zi yordamida
~ifodalab bo’lmaydi}

Pythonda kompleks son yozilish formati to’g’ri berilgan javobni tping?{
=<Haqiyqiy qism> + <Mavhum qism>J
~<Haqiyqiy qism>J + <Mavhum qism>
~<Haqiyqiy qism>+J + <Mavhum qism>
~<Haqiyqiy qism>+ <Mavhum qism>+J}

Pythonda o’nlik sonni ikkilik sanoq sistemasiga o’tkazish uchun qayis funksiya ishlatiladi?{
=bin (<Son>)
~oct (<Son>)
~hex (<Son>)
~abs (<Son>)}

Pythonda o’nlik sonni sakkizlik sanoq sistemasiga o’tkazish uchun qayis funksiya ishlatiladi?{
=oct (<Son>)
~bin (<Son>)
~hex (<Son>)
~abs (<Son>)}

Pythonda o’nlik sonni o’n oltilik sanoq sistemasiga o’tkazish uchun qayis funksiya ishlatiladi?{
=hex (<Son>)
~oct (<Son>)
~bin (<Son>)
~abs (<Son>)}

Pythonda divmod (x, y) funksiyasi qanday amal bajkaradi?{
=Sonning butun va qoldiq qismini qaytaradi
~Sonning butun qismini qaytaradi
~Sonning qoldiq qismini qaytaradi
~Sonni yaxlitlaydi}

Pythonda sonni radiandan gradusga o'tkazish uchunqaysi funksiya qo'llaniladi?{
=degrees()
~radians()
~exp()
~ceil()}

Pythonda sonni eng yaqin katta songa yaxlitlash funksiyasi qaysi?{
=ceil ()
~floor ()
~fabs ()
~round ()}
Pythonda random() funksiyasining vazasi nima?{
=0.0 dan 1.0 gacha tasodifiy sonlarni qaytaradi
~Tasodifiy sonlar diapazonini aniqlaydi
~Tasodifiy sonlar modulini chaqirib beradi
~1 dan 10 gacha tasodifiy sonlarni qaytaradi}
Pythonda uniform () funksiyasining vazasi nima?{
=Ko'rsatilgan diapazondagi haqiyqiy tasodifiy sonlar qaytaradi
~0.0 dan 1.0 gacha tasodifiy sonni qaytaradi
~Tasodifiy sonlar modulini chaqirib beradi
~1 dan 10 gacha tasodifiy sonlarni qaytaradi}
Pythonda randint () funksiyasining vazasi nima?{
=Ko'rsatilgan diapazondagi butun tasodifiy sonni qaytaradi
~0.0 dan 1.0 gacha tasodifiy sonni qaytaradi
~Tasodifiy sonlar modulini chaqirib beradi
~1 dan 10 gacha tasodifiy sonlarni qaytaradi}
Pythonda qaysi funksiya berilgan ketma-ketlik (satr, ro'yxat, kortej) dagi tasodifiy elementni qaytaradi?{
=choice (<Ketma-ketlik>)
~shuffle (<Ro'yxat> [, <0.0 dan 1.0 gacha sonlar>])
~sample (<Ketma-ketlik>, <Elementlar soni>)
~randrange ([<Boshlanish>,] <Tugash> [, <Qadam>]) **}**

Ishchi fan dasturiga muvofiq baholash mezonlarini qo’llash bo’yicha uslubiy ko’rsatmalar

“Python dasturlash tili” fanidan

BAHOLASH MEZONI

Talabalar bilimini nazorat qilish uchun ishlab chiqilgan ushbu baholash mezoni O'zbekiston Respublikasi Oliy va o'rta maxsus ta'lim vazirining “Oliy ta'lim muassasalarida talabalar bilimini nazorat qilish va baholash tizimi to'g'risidagi nizomni tasdiqlash haqida” 2018 yil 9 avgustdagi 19-2018-sonli buyrug'i asosida ishlab chiqildi.

Talabalar bilimini nazorat qilish oraliq va yakuniy nazorat turlarini o'tkazish orqali amalga oshiriladi.

Oraliq nazorat (ON) semestr davomida ishchi fan dasturining tegishli bo'limi tugagandan keyin talabaning bilim va amaliy ko'nikmalarini baholash maqsadida o'quv mashg'ulotlari davomida o'tkaziladi.

Ushbu fan bo'yicha oraliq nazorat turi ikki marta o'tkaziladi. Oraliq nazorat yozma shaklda o'tkaziladi. 1-ON ma'ruzalarning birinchi yarmi o’tilgandan keyin, 2-ON esa, qolgan ikkinchi yarmi o'tilgandan keyin, yakuniy nazorat (YN) esa, o'z navbatida semestr yakunida o'tkaziladi.

Shu bilan birga, ushbu fan bo'yicha talabaning amaliy, tajriba va mustaqil ta'lim topshiriqlarini bajarishi, shuningdek uning ushbu mashg'ulotlardagi faolligi ham baholab boriladi.

Talabalar bilimini baholash 5 baholik tizimda amalga oshiriladi.

Talabalarning bilimi quyidagi mezonlar asosida:

- talaba mustaqil xulosa va qaror qabul qiladi, ijodiy fikrlay oladi, mustaqil mushohada yuritadi, olgan bilimini amalda qo'llay oladi, fanning (mavzuning) mohiyatini tushunadi, biladi, ifodalay oladi, aytib beradi hamda fan (mavzu) bo'yicha tasavvurga ega deb topilganda — 5 (a'lo) baho;
- talaba mustaqil mushohada yuritadi, olgan bilimini amalda qo'llay oladi, fanning (mavzuning) mohiyatni tushunadi, biladi, ifodalay oladi, aytib beradi hamda fan (mavzu) bo'yicha tasavvurga ega deb topilganda — 4 (yaxshi) baho;
- talaba olgan bilimini amalda qo'llay oladi, fanning (mavzuning) mohiyatni tushunadi, biladi, ifodalay oladi, aytib beradi hamda fan (mavzu) bo'yicha tasavvurga ega deb topilganda — 3 (qoniqarli) baho;
- talaba fan dasturini o'zlashtirmagan, fanning (mavzuning) mohiyatini tushunmaydi hamda fan (mavzu) bo'yicha tasavvurga ega emas deb topilganda — 2 (qoniqarsiz) baho bilan baholanadi.

Nizomga ko'ra, yakuniy nazorat turini o'tkazish va mazkur nazorat turi bo'yicha talabaning bilimini baholash o'quv mashg'ulotlarini olib bormagan professor-o'qituvchi tomonidan amalga oshiriladi. Shu sababli, ushbu fandan YN

o'tkazish uchun kafedra o’qituvchilari hamda NamDU “Amaliy matematika” kafedrasining turdosh fan o'qituvchi-professorlarini YN ga jalb etish ko'zda tutilgan.

Oraliq nazorat turini topshirmagan, shuningdek ushbu nazorat turi bo'yicha «2» (qoniqarsiz) baho bilan baholangan talaba yakuniy nazorat turiga kiritilmaydi.

Ushbu fan bo’yicha quyidagicha baholash tizimini tashkil qilinadi:

Fan bo’yicha 6 ta tajriba ishi (***T***) o’tkaziladi. Tajriba ishlari bo’yicha baholash quyidagicha hisoblanadi:

$$T = \frac{T_1 + T_2 + T_3 + T_4 + T_5 + T_6}{6}.$$

Amaliy mashg’ulot o’tish davomida oraliq nazorat o’tkaziladigan kunga qadar xar bir talaba 4 ta amaliy ish topshiradi va baholanadi. Bu baholash **amaliy** (***A***) topshiriqlar asosida amalga oshiriladi. Talabaning **o’zlashtirishini** (***O’***) hisoblashda shuningdek **mustaqil ta’lim** (***M***) hamda talabaning **faolligini** (***F***) ham xisobga olinadi. Umumiy holda **o’zlashtirish**ni qiyidagicha hisoblanadi:

$$O' = \frac{T + \frac{A_1 + A_2 + A_3 + A_4 + M + F}{6}}{2}.$$

Ijobiy o’zlashtirgan talabalar **oraliq narorat** (***ON***) ga qo’yiladi. Ushbu fan bo’yicha oraliq nazorat yozma ish shaklida o’tkaziladi.

Oraliq nazoratdan ijobiy baholangan talabalar **yakuniy nazorat** (***Yn***) ga qo’yiladi.

Qo’yilgan baholar hammasi guruh jurnalida qayd qilib boriladi.

Talabaning YN da olgan bahosi yakuniy baho hisoblanadi va u qaydnomaga hamda talabaning baholash daftarchasiga qo’yiladi.