Z.M. Adizova

DASTURLASH TEXNOLOGIYALARI

O‘QUV QO‘LLANMA

“KAMOLOT” nashriyoti
Buxoro 2023

UO‘K: 004 (075.8)
KBK: 32.973ya73
A 97

Z.M. Adizova Dasturlash texnologiyalari / [Matn]: o‘quv qo‘llanma / Buxoro : “BUXORO DETERMINANTI”MCHJning Kamolot nashriyoti, 2023. - 204 b.

Ushbu o‘quv qo‘llanma Python dasturlash tillaridagi imkoniyatlarini ochib berishga qaratilgan. Python dasturlash tillari hozirgi kunda ko‘pgina ta’lim muassasalarida o‘qitilayotganligi sababli, aynan shu dasturlash tillari tanlab olindi. O‘quv qo‘llanmada Python dasturlash tilini yaratilishi, imkoniyatlari, asosiy operatorlari, qo‘shimcha funksiyalari va darslikda keltirilgan oddiy dasturlashdan murakkab masalalar, ya’ni funksiya va protseduralarni qo‘llab dasturlash, shuningdek, grafik imkoniyatlari aniq misollar asosida ochib berilgan.

Taqrizchilar:

BuxDU Axborot tizimlari va raqamli texnologiyalar kafedrasi dotsenti f.-m.f.n **N.S.Sayidova**

BMTI Axborot kommunikatsiya texnologiyalari kafedrasi p.f.d. (DSc), professori **F.R.Muradova**

ISBN: 978-9943-9148-9-6

Oliy ta’lim, fan va innovatsiyalar vazirligi Buxoro davlat universitetining 2023 yil “23” martdagi “112”- sonli buyrug‘iga asosan nashr etishga ruxsat berildi.

MUNDARIJA

Kirish

Mamlakatimizdagi ta’lim tizimi, uni tashkil etish prinsiplari, mazmuni, ta’lim-tarbiya jarayonining shakl va usullarini yangi ta’lim texnologiyalari talablari daRejasida tubdan isloh qilishni taqozo etmoqda. Hozirgi davrga kelib zamonaviy axborot texnologiyalarining asosiy vakili bo‘lgan kompyuterlar hayotimizning barcha sohalariga jadallik bilan kirib bormoqda. Shunday ekan, kompyuter texnologiyasini o‘rganishga alohida e’tibor berish talab qilinmoqda.

Bu o‘rinda “Informatika va axborot texnologiyalar” fanining o‘rni alohida hisoblanadi. Informatika va axborot texnologiyalarni samarali o‘qitilishini yo‘lga qo‘yilishida o‘quv jarayonida amaliy dasturlardan foydalanish fanining ahamiyati juda kattadir.

Dasturlash-kompyuterlar va boshqa mikroprosessorli elektron mashinalar uchun dasturlar tuzish, sinash va o‘zgartirish jarayonidan iborat. Odatda dasturlash yuqori saviyali dasturlash tillari orqali amalga oshiriladi.

Dasturlash tillarining semantikasi odam tiliga yaqinligi tufayli dastur tuzish jarayoni ancha oson kechadi.

Algoritm so‘zi algoritmi so‘zidan olingan bo‘lib, u IX-asrning matematigi bobokalonimiz Muhammad Al-Xorazmiy nomining lotincha shaklidir.Informatika sohasida algoritm tushunchasi asosiy tushuncha hisoblanadi. Algoritm-bir masalani bajarish uchun bajarilishi zarur bo‘lgan buyruqlarning tartiblangan ketma-ketligi.Tuzilgan algoritmni uning yozilish qoidalarini tushunadigan va unda ko‘rsatilgan buyruqlarni bajarish imkoniyatiga ega bo‘lgan insonning o‘zi yoki texnik qurilma bajarishi mumkin.

Har qanday qo‘yilgan masalani kompyuterda yechish uchun oldin uning yechish usulini tanlab, keyin uning algoritmini ishlab chiqish kerak bo‘ladi. Dastur-bu berilgan algoritmga asoslangan biror bir algoritmik tilda yozilgan ko‘rsatmalar, ya’ni buyruqlar yoki operatorlar to‘plamidir. Dasturlash — esa bu dastur tuzish jarayonidir. Python dasturlash tillarini qiyoslasak, uch til universal til hisoblanadi, hamda ko‘plab imkoniyatlarga ega.

1-ma’ruza. Python tili va uning dasturlash muhiti

Reja:

1. Python dasturlash tili tarixi
2. Python dasturlash tili imkoniyatlari
3. Python dasturlash tilini o‘rnatish

Python dasturlash tilini yaratilishi 1990-yil boshlaridan boshlangan. O‘sha paytlarda uncha taniqli bo‘lmagan Gollandiyaning CWI institute xodimi Gvido van Rossum ABC tilini yaratilish proektida ishtirok etgan edi. ABCtili 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 MontiPayton 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 NRI koorparatsiyasiga ishlashga taklif etishdi. U o‘sha yerda ishladi va koorparatsiya 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 koorparatsiyani proektlarini yaratishgaketardi 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 CNRIdan ketdi, lekin shartnomaga 8 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 qatoriga kiradi. Python yuqori daRejadagi ma’lumotlar strukturasi va oddiy lekin samarador obyektga 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.

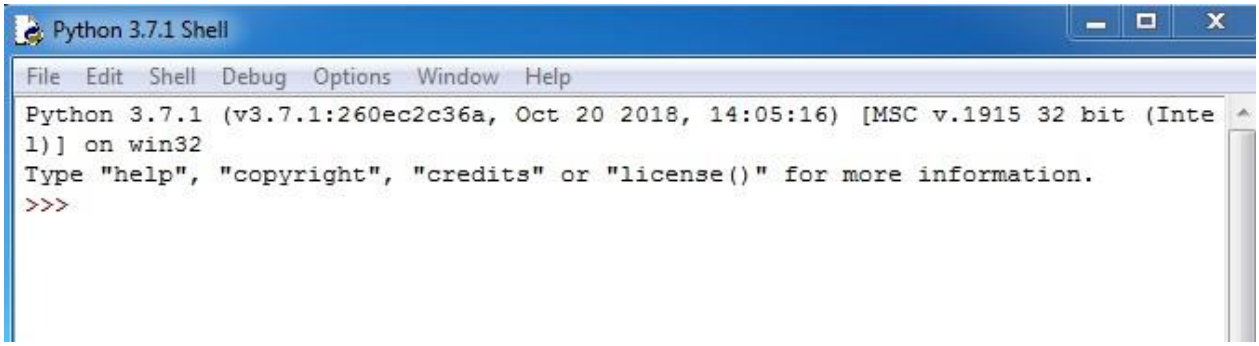
Dasturni yozishda vomidaquyidaRejadagidetallarni,misoluchun xotirani boshqarishni hisobga olish shart emas. Ko‘plab platformalarda hech qanday o‘zgartirishlarsiz ishlay oladi.Interpretatsiya qilinadigan til.

Kengayishga moyil til. Agar dasturni biror joyini tezroq ishlashini xohlasak shu qismni C yoki C++ dasturlash tillarida yozib keyin shu qismni python kodi orqali ishga tushirsa(chaqirsa) bo‘ladi.
Juda ham ko‘p xilma-xil kutubxonalarga ega.
7.xml/html fayllar bilan ishlash
http so‘rovlari bilan ishlash
GUI(grafik interfeys)
10.Veb saytlarni yaratish
11.FTP bilan ishlash
12. Rasmli audio video fayllar bilan ishlash13.Robot texnikada
14.Matematik va ilmiy hisoblashlarni dasturlash

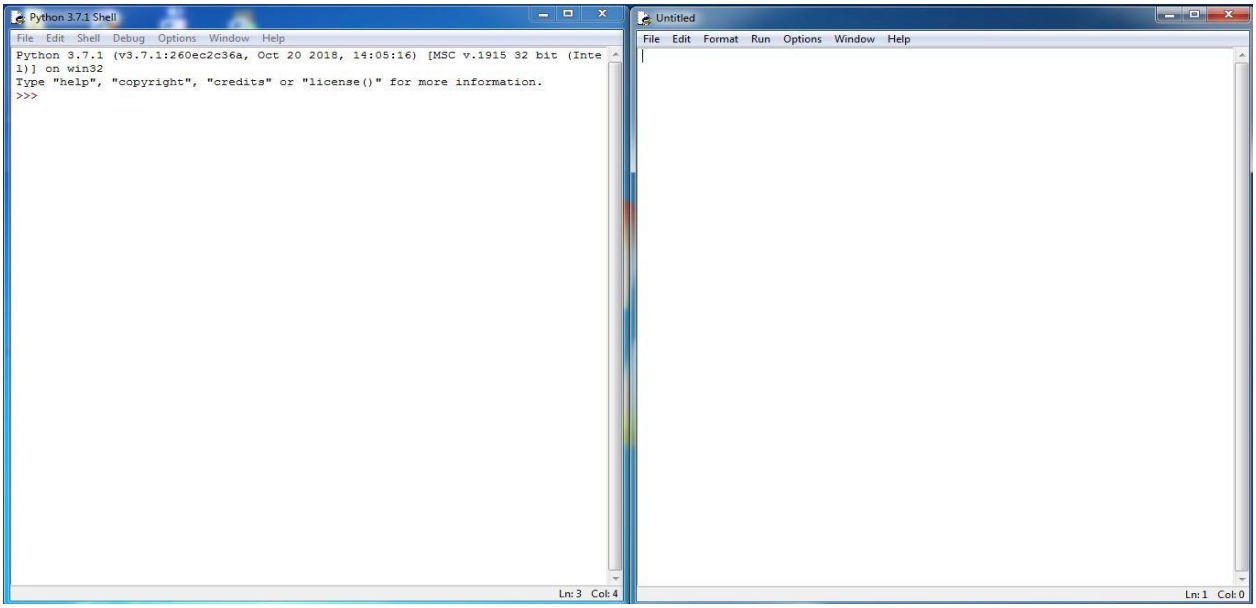
Pythonni katta proyektlarda ishlatish mumkin. Chunki, uni chegarasi yo‘q, imkoniyati yuqori. Shuningdek, u sodda va universalligi bilan dasturlash tillari orasida eng yaxshisidir.

IDLEni ishga tushirish tartibi

Har doim yangi dastur tuzishda IDLE alohida ishga tushiriladi, ishga tushirishtartibi esa doim bir xil ko‘rinishda bo‘ladi.



File bo‘limidan New Fileni tanlab (klaviaturadan Ctrl+N), yangi ikkinchi oqoynani hosil qilamiz.



Ikkinchi oyna kod yozish uchun, birinchi oyna esa dastur natijasini ko‘rish uchun ishlatiladi. Unutmang, ikkinchi oynada kodlarni yozib bo‘lgach uni saqlab olishimiz kerak, aks holda dastur ishlamaydi. Saqlash uchun ishchi stolidan “Dasturlar” nomli papka hosil qilib, hamma dasturlarni shu papkaga saqlaymiz. Hozir namuna sifatida biror dastur yozib, uni saqlab ishga tushirishni o‘rganamiz.

Nazorat savollari:

1. Python dasturlash tili qachon yaratila boshlangan?
2. IDLEni ishga tushirish tartibi qanday?
3. Pythonning o‘ziga xosligi nimalarda?
4. Python dasturlash tili imkoniyatlarini aytib bering?

2-ma’ruza. Python dasturlash tilida o‘zgaruvchilar.

Reja:

1. Python dasturlash tilida o‘zgaruvchilar.
2. Python dasturlash tilida o‘zgaruvchi tiplari.
3. Mantiqiy va haqiqiy qatorlar.

Literal konstantalar bilan ishlash tez orada sizni zeriktirishi mumkin. Biror ma’lumotni saqlash va uning ustida turli amallarni bajarish uchun bizga o‘zgaruvchilar yordam beradi. O‘zgaruvchining qiymati, o‘z nomi bilan aytib turibdiki, o‘zgarishi mumkin. Siz unda xohlagan qiymatni saqlashingiz mumkin.

O‘zgaruvchilar kompyuter xotirasidagi joy bo‘lib, u yerda siz biror ma’lumotni saqlaysiz. O‘zgaruvchining konstantadan farqi, o‘zgaruvchiga dastur ishlashi davomida (run time) murojaat qilib, uning qiymatini o‘zgartira olamiz. Konstantaga esa oldindan ma’lum bir qiymat beriladi va bu qiymatni o‘zgartirib bo‘lmaydi.

O‘zgaruvchilarni nomlashda quyidagi qoidalarga amal qilish kerak:

O‘zgaruvchining birinchi belgisi alifbo xarfi (ASCII simvollari katta va kichik registrda va Unicode) yoki “_" (underscore) simvoli bo‘lishi mumkin.

O‘zgaruvchilarning qolgan qismi xarflardan (ASCII simvollari katta va kichik registrda va Unicode), “_" (underscore) 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 va ixtiyoriy_simvol

O‘zgaruvchilarni noto‘g‘ri nomlashga misollar: 2things, ’ ’, my-name, >a1b2_c3 va “o‘zgaruvchi qo‘shtirnoqda"

O‘zgaruvchi va konstantalarni qo‘llanilishiga misol:

```
i = 5
print(i)
i = i + 1
print(i)
s = '''Bu ko'p qatorlik satr.
Bu uning ikkinchi qatori.'''
print(s)
```

Natija: 5 6
Bu ko‘p qatorlik satr.
Bu uning ikkinchi qatori.
Mantiqiy va haqiqiy qatorlar.
Haqiqiy qator – bu siz dastur yozayotganingizda ko‘rib turgan qatoringiz.
Mantiqiy qator – bu python biror amal bajarish uchun bir butun deb hisoblab ko‘radigan qator.
Mantiqiy qatorga misol:
print(’Hello World!’)
shu bilan birga u bir qatorda joylashgani uchun haqiqiy qatorga ham misol bo‘ladi.
Python o‘z holicha, har bir haqiqiy qatorni bir mantiqiy qator deb qaraydi.
i = 5
print(i)
yoki:
i = 5;
print(i);
Agar siz bir haqiqiy qatorda bir necha mantiqiy qator yozishni istasangiz u holda ’;’ bilan har bir mantiqiy qator tugash nuqtasini belgilashingiz lozim.
Misol uchun:
i = 5;
print(i);
yoki:
i = 5; print(i)

Bir mantiqiy qatorni bir haqiqiy qatorda yozishingiz maqsadga muvofiq. Bu bilan siz hech qanday nuqta vergulsiz dastur yozishni uddalashingiz mumkin.

Bittadan ko‘p haqiqiy qatorni bitta mantiqiy qator uchun ishlatish mumkin. Bu uzun satrlarni yozishda ishlatiladi. Quyida bir necha haqiqiy qatorni jamlagan bitta mantiqiy qator keltirilgan.
s = ’Bu satr. \
Bu uning davomi.’
print(s)

Natija:
Bu satr. Bu uning davomi.
Bo‘sh joy (Отступы)

Pythonda har bir qator boshidagi bo‘sh joy muhim ahamiyatga ega. Mantiqiy qator boshidagi bo‘sh joy(пробелы и табуляции) shu mantiqiy qatorning bajarilish daRejasini belgilaydi va amallarni guruhlashda katta ahamiyatga ega.

Bu shuni anglatadiki birgalikdagi amallar bir xil bo‘sh joyga ega bo‘lishlari kerak. Har bunday amallar to‘plami blok deb nomlanadi.

Keyingi darslarimizda bo‘sh joylar qanchalar muhimligini misollar orqali ko‘rishimiz mumkin bo‘ladi.

Yodingizda saqlang! Noto‘g‘ri qo‘yilgan bo‘sh joylar xatolik yuz berishiga olib kelishi mumkin!

Misol uchun quyidagi dasturni probel.py fayliga yozing va python yordamida ishga tushiring:

```
i = 5
print(’Qiymat ’, i) # Xatolik! Bo‘sh joy qator boshida
print(’Takroran, qiymat’, i)
```

Ishga tushirganingizda quyidagicha xatolik yuz beradi.

```
File “probel.py", line 2
print(’Qiymat’, i) # Xatolik! Bo‘sh joy qator boshida
IndentationError: unexpected indent
```

E’tibor bering 2-qator boshida bitta probel mavjud. Python chop etgan xatolik shuni anglatadiki dastur sintaksisi noto‘g‘ri va qoidalar bo‘yicha yozilmagan.

Bu shuni anglatadiki siz ixtiyoriy joyda amallar blokini boshlay olmaysiz (faqat asosiy dasturda ishlatiladigan blokdan tashqari).

Probel va tabulyatsiya belgilarini bo‘sh joyda aralashtirib yubormang. Bo‘sh joylar uchun bir tabulyatsiya yoki to‘rtta probel ishlatishga harakat qiling.

Nazorat savollari:

1. Python dasturlash tilida o‘zgaruvchilarni nomlashda qanday qoidalar mavjud?
2. Pythonda mantiqiy va haqiqiy qatorlar nima?
3. Python dasturlash tilida bo‘sh joy tashlash qanday amalga oshiriladi?

3-ma’ruza. Pythonda ma’lumot turlari.

Reja:

1. Python dasturlash tili ma’lumotlar turlari.
2. Python dasturlash tilida sonlar.

Python dasturlash tilida quyidagi ma’lumot turlari mavjud:

1	Matn turi	str
2	Son turlari	Int, float, complex
3	Tartib turlari	List, tuple, range
4	Xarita turi	Dict
5	Turlarni o‘rnatish	Set, frozenset
6	Boolean(mantiqiy) turi	Bool
7	Ikkilik turlari	Byte, bytearray,memoryview

Pythonda dasturlash tilida type() funksiyasi yordamida istalgan obyekt ma’lumotlarini olishingiz mumkin.

Masalan: print(type(attribute))

```
n = 99
print(type(n))
```

Natija:

```
<class ’int’>
```

Agar ma’lumotlar turini ko‘rsatmoqchi bo‘lsangiz, quyidagi konstruktor funktsiyalaridan foydalanishingiz mumkin:

x = str("Adizova Zuhro")	str
x = int(99)	int
x = float(99.09)	float
x = complex(99j)	complex
x = set(("Z", "ZA", "AZM-1996"))	set
x = frozenset(("Z", "ZA", "AZM-1996"))	frozenset
x = bool(99)	bool

x = bytes(99)	bytes
x = bytearray(99)	bytearray
x = memoryview(bytes(99))	memoryview
x = list(("Z", "ZA", "AZM-1996"))	list
x = tuple(("Z", "ZA", "AZM-1996"))	tuple
x = range(99)	range
x = dict(name="Adizova Zuhro", age=24)	dict

Pythonda sonlar

Python dasturlash tilida 3 xil sonlar turi mavjud va raqamli turiga kiruvchi o‘zgaruvchilar ularga qiymay belgilaganingizda yaratiladi.

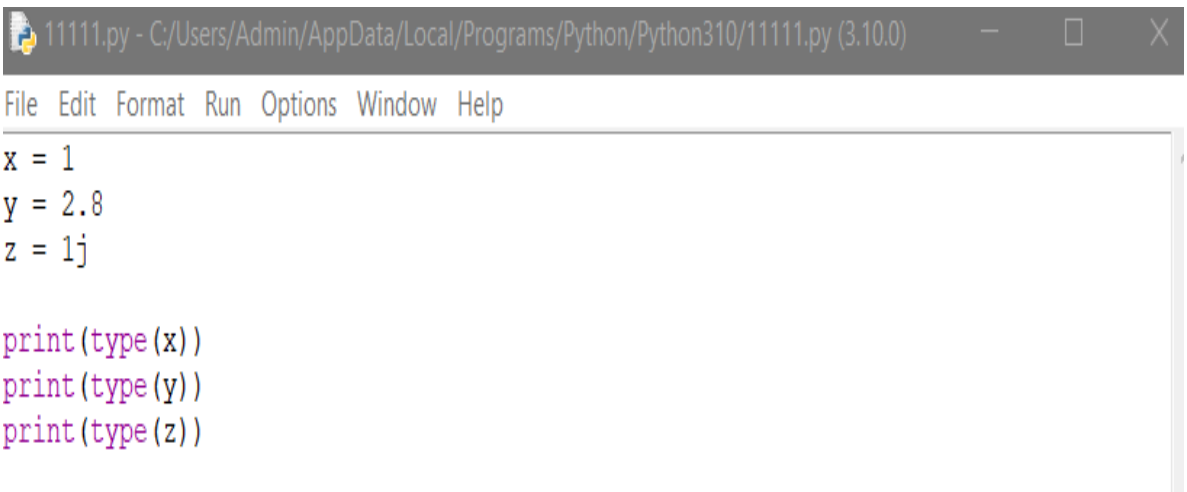
int

float

complex

```
x = 1    # int
y = 2.8  # float
z = 1j   # complex
```

Python dasturlash tilida har qanday o‘zgaruvchini turini aniqlash uchun type() funksiyasidan foydalangan holda keladigan o‘zgaruvchi yoki belgilangan o‘zgaruvchini turini aniqlash imkoniyati mavjud.



Natija:

```
=== RESTART: C:/Users/Admin/AppData/Local/Programs/Python/Python310/11111.py ===
<class 'int'>
<class 'float'>
<class 'complex'>
```

Int- bu turgan tegishli sonlar o‘nlik sonlar bo‘lmagan butun qiymatga ega qoldiq qismiga ega bo‘lmagan va manfiy yoki musbat sonlar kiradi.

```
x = 99
y = 549219161219562
z = -6544598464654654
print(type(x))
print(type(y))
print(type(z))
```

Natija:

<class’int’>
<class’int’>
<class’int’>

float— yuqoridagi qoidaga (int) muaffiq ammo tarkibiga o‘nlik sonlarni oladigan manfiy yoki musbat sonlar.

```
x = 99.656546
y = 5492191.61219562
z = -654459846465.4654
print(type(x))
print(type(y))
print(type(z))
```

Natija:

<class’float’>
<class’float’>
<class’float’>

complex — Kompleks sonlar xayoliy qismi sifatida "j" yoziladi.

```
x = 4+9j
y = 8j
z = -9j
print(type(x))
print(type(y))
```

```
print(type(z))
```
Natija:
```
<class’complex’>
<class’complex’>
<class’complex’>
```
Python dasturlash tilida boshqa dasturlash tillari kabi sonlar turini bir turdan boshqa bir turga o‘tkazish imkoniyati mavjud. Buning uchun int(), float()va complex() funksiyalardan foydalanishingiz mumkin.
```
# butun sondan float ga konvert qilish
x = float(1)
# float dan int ga o‘tkazish
y = int(2.8)
# int dan complex ga o‘tkazish
z = complex(1)
print(x)
print(y)
print(z)
```
Natija:
```
1.0
2
(1+0j)
```
Python random() tasodifiy raqamni hosil qilish funktsiyasiga ega emas , ammo Python random tasodifiy raqamlarni chiqarish uchun ishlatilishi mumkin bo‘lgan ichki modulga ega :
```
Import random
print(random.randrange(1, 10))
```

Nazorat savollari:

1. Python dasturlash tilida ma’lumotlar turlarini sanab bering?
2. Pythonda sonli ma’lumotlar?
3. Python dasturlash tilida bo‘sh joy tashlash qanday amalga oshiriladi?

4-ma’ruza.Pythonda operatorlar bilan ishlash.

Reja:

1. Python dasturlash tilida matematik funksiyalar.
2. Pythonda arifmetik operatorlar va ularning tadbiqi
3. Pythonda mantiqiy operatorlar va ularning tadbiqi
4. Pythonda matematik funksiyalar

Pythonning matematik funksiyalar kutubxonasi trigonometrik hisoblashlar, sonli shakl almashtirishlar va sonli almashtirishlarni bajaradi. Trigonometrik funksiyalar argumentlari radianlarda beriladi, hamda graduslarni radianga va aksincha almashtiruvchi funksiyalar ham mavjud. Matematik operatorlar bilan bir qatorda Pythonda ko‘p sonli matematik funksiyalar ham nazarda tutilgan.

Bular quyidagilar:

acos() - radianda ifodalangan arksinus.

import math x=float(input(’x=’))y=math.acos(x) print(’acos=’,y)

acosh() - radianda ifodalangan giperbolik arkkosinus.

import math x=int(input(’x=’))y=math.acosh(x)print(’acosh=’,y)

asin() - radianda ifodalangan arksinus.

import math x=int(input(’x=’))y=math.asin(x) print(’asin=’,y)

asinh() - giperbolik arksinus.

import math x=int(input(’x=’))y=math.asinh(x) print(’asinh=’,y)

atan() - radianda ifodalangan arktangis.

import math x=int(input(’x=’))y=math.atan(x) print(’atan=’,y)

atanh() - giperbolik arktanges.

import math x=float(input(’x=’))y=math.atanh(x) print(’atanh=’,y)

abs() - sonning absolyut qiymati.

acos() – radianda ifodalangan arkkosinus.

atan2() - arktangens y/x ni, y va x kvadratlar ishorasi bilan aniqlanuvchinatijaviy kvadrat bilan qaytariladi.

ceil() - sonni o‘zidan katta butun songa yaxlitlash.

cos() - radianda ifodalangan kosinus.

cosh() - giperbolik kosinus.

exp() - berilgan sonning eksponentasini hisoblash.

floor() - sonni o‘zidan kichik butun songa yaxlitlash.

fmod() -ikki son x ni y ga bo‘lgandagi qoldiqni hisoblaydi.

hypot() - to‘g‘ri burchakli uchburchakda ikki katet bo‘yicha gipotenuzanihisoblash.

log10() - o‘nlik logarifm.
log() - natural logarifm.
log1p() – log(1+x), bunda x ning qiymati nolga yaqin bo‘lganda ham natija aniq chiqadi. log() ning aniqligi etarli bo‘lmaganligi sababli, bu holda shunchaki log(1) ga qaytiladi.
pi() - π sonining qiymatini aniqlaydi.
pow() – x sonini y daRejaga ko‘tarish.
sin() - radianda ifodalangan sinus.
sqrt() – x sonining kvadrat ildizi.
sinh() - radianda ifodalangan geperbolik sinus.
tan() - radianda ifodalangan tangens
tanh() - radianda ifodalangan giperbolik tangens.
% - birinchi argumentni ikkinchi argumentga bo‘lgandagi qoldiq.
factorial(num) – sonning faktorialini hisoblaydi.
degrees(rad) – radiandan gradusga o‘tkazadi.
radians(grad) – gradusdan radianga o‘tkazadi;
Fabs(x) – x ning absolyut raqami
Frexp(x) — mantisa va tartibni (m, i) juftligi kabi qaytaradi, m- o‘zgaruvchan nuqtali son, i esa- x=m*2**i ga teng butun son bo‘ladi. Agarda 0-(0,0) qaytarsaboshqa paytda 0.5<=abs(m)<1.0 bo‘ladi.
Ldexp(m,i)=m*(2**i).- m ni, 2 ni i daragacha tartibda qaytaradi.
Modf(x)- (y,q) juftlikda x ning butun va q kasr qismini qaytaradi.
Trunc(x)-x haqiqiy sonning butun qismini qaytaradi.
tau – tau konstantasi
e – e konstantasi
gamma – x sonining gamma qiymati
int([object],[sanoq sistemasi asosi])- butun sonni berilgan sanoq sistemasidano‘nlik sanoq sistemasiga o‘tkazadi.
bin(x)- butun sonni ikkilik sanoq sistemasiga o‘tkazadi.
hex(x)- butun sonni o‘n oltilik sanoq sistemasiga o‘tkazadi.
oct(x)- butun sonni sakkizlik sanoq sistemasiga o‘tkazadi.
Python tilida turli tipdagi o‘zgaruvchilardan foydalanish mumkin, shu sababli, har bir tipdagi o‘zgaruvchilar qanday tavsiflanishini bilish zarur.
Python tilida bitta o‘zgaruvchini dastur bajarilishi davomida satr yoki son sifatida ishlatish mumkin. Shu bilan birga Python tilida o‘zgaruvchilar bilan ishlanganda oshkor ko‘rsatilishi mumkin bo‘lgan asosiy ma’lumotlar tiplari to‘plami mavjud.

Butun (integer) sonlar – Sonning kasr bo‘lmagan son bo‘lib, ularda sonning asosi (10 lik), o‘n oltilik (asosi 16-prefiksga ega) yoki sakkizlik (asosi 8- prefiksli) sanoq sistemalar ko‘rsatiladi.

Siljuvchi vergulli (float) sonlar – sonning kompyuterda amalga oshiriladigan eksponentsiol yozuvi. Xuddi shuningdek, “ikkilangan aniqlikga” ega bo‘lgan son ham mavjud.

Satr (string) – simvollar ketma – ketligidan iborat bo‘lib, unda har bir simvol bir bayt o‘lchamdan, toki maksimal uzunlik 216 gacha bo‘lgan joyni egallaydi. Yakka qavslarga olingan satrlar literallar sifatida qaraladi, ayni paytda ikkilangan qavslar ichidagi satrlar esa (maxsus belgilar, o‘zgaruvchilarning qiymatlari va shu kabilar) sifatida talqin qilinadi.

Bul (boolean) tipi — Mantiqiy ifoda bo‘lib, uning qiymati faqat rost (True) yokiyolg‘on (False) dan iborat.

Kompleks (complex) tipi – Sonning birinchi argument sifatida haqiqiy qism, ikkinchi argument sifatida mavhum qism uzatiladi.

Massiv (array) – bir nechta qiymatlarning tartiblashtirilgan xaritasi bo‘lib, undagi kalitlar qiymatlarga mos keladi. Kalitlar – bu indeks nomerlari (ular so‘zsiz tushuniladi) yoki aniq ko‘rsatilgan nishonlar.

Ob’ekt – bu berilganlarning xossalarini saqlovchi va berilganlarni qayta ishlash metodlaridan iborat bo‘lgan sinf.

Resurs – tashqi resursga havola bo‘lib, ular maxsus funksiyalar tomonidan yaratiladi va saqlanadi.

NULL – qiymatga ega bo‘lmagan o‘zgaruvchi. Bu o‘zgaruvchi, shakllantirilmagan bo‘ladi (unga hech bir qiymat berilmagan bo‘ladi), agar unga NULL o‘zgarmasi ta’minlangan bo‘lsa yoki unset() funksiyasi yordamidabekor qilinmagan bo‘lsa.

Pythonda arifmetik, mantiqiy operatorlar va ularning tadbiqi

Arifmetik amallar va qiymat berish operatori. Berilganlarni qayta ishlash uchun PYTHON tilida amallarning juda keng majmuasi aniqlangan. Amal — bu qandaydir harakat bo‘lib, u bitta (unar) yoki ikkita (binar) operandlar ustida bajariladi, hisob natijasi uning qaytariluvchi qiymati hisoblanadi. Tayanch arifmetik amallarga qo‘shish (+), ayirish (-), ko‘paytirish (*), bo‘lish (/), daRejaga ko‘tarish (**) va bo‘lish qoldig‘ini olish (%) amallarini keltirish mumkin. Amallarqaytaradigan qiymatlarni o‘zlashtirish uchun qiymat berish amali (=) va uning turli modifikatsiyalari ishlatiladi: qo‘shish,

qiymat berish bilan (+); ayirish, qiymatberish bilan (-); ko‘paytirish qiymat berish bilan (*); bo‘lish, qiymat berish bilan (/); bo‘lish qoldig‘ini olish, qiymat berish bilan (%) va boshqalar. Ularning umumiy ko‘rinishlariga to‘xtalamiz.

Razryadli mantiqiy amallar. Dastur tuzish tajribasi shuni ko‘rsatadiki, odatda qo‘yilgan masalani yechishda biror holat ro‘y berganligini yoki yo‘qligini ifodalash uchun 0 va 1 qiymat qabul qiluvchi bayroqlardan foydalaniladi. Shu maqsadda bir yoki undan ortiq baytli o‘zgaruvchilardan foydalanish mumkin. Masalan, bool (mantiqiy) tupdagi o‘zgaruvchini shu maqsadda ishlatsa bo‘ladi. Boshqa tomondan, bayroq sifatida baytning razryadlaridan foydalanish ham mumkin. Chunki razryadlar faqat ikkita qiymatni – 0 va 1 sonlarini qabul qiladi. Bir baytda 8 razryad bo‘lgani uchun unda 8 ta bayroqni kodlash imkoniyati mavjud. Quyidagi jadvalda Python tilida bayt razryadlari ustida mantiqiy amallarmajmuasi keltirilgan.

Bayt razryadlari ustida mantiqiy amallar

Amallar	Mazmuni
And yoki &	Mantiqiy VA (ko‘paytirish)
Xor yoki |	Mantiqiy yoki (qo‘shish))
Or yoki ^	Istisno qiluvchi YOKI

Razryadli mantiqiy amallarning bajarish natijalarini jadval ko‘rinishida ko‘rsatishmumkin.

A	B	A&B	A|B	A^B
0	0	0	0	0
0	1	0	1	1
1	0	0	1	1
1	1	1	1	0

Nazorat savollari:

1. Python matematik operatorlar bilan ishlash?
2. Pythonda mantiqiy operatorlar bilan ishlash?
3. Python dasturlash tilida razryadli mantiqiy amallar qanday bajariladi?
4. Bayt razryadlari ustida mantiqiy amallar qaysilar sanab bering?

5-ma’ruza. Pythonda tarmoqlanuvchi operatorlar.

Reja:

1. Python dasturlash tilida tarmoqlanuvchi operatorlar.
2. Python dasturlash tilida mantiqiy amallar.
3. Python dasturlash tilida PASS funksiyasi.

Python If … Else (shart amali)

Python dasturlash tili odatiy matematik mantiqiy amallarni qo‘llab quvvatlaydi. demak bo‘lar quyidagilar.

•Teng: a == b
•Teng emas: a! = B
•Kichik: a <b
•Kichik yoki teng: a <= b
•Katta: a> b
•Katta yoki teng: a> = b

Yuqorida keltirilgan shartlarni bir necha xil ko‘rinishda bir shart ichida keltirib o‘tsangiz bo‘ladi. Quyidagi ko‘rinishda If shart amaliga misol keltirib o‘taman.

```
a = 13
b = 15
if b > a:
  print("b son a dan katta")
>>>b son a dan katta
```

Yuqorida keltirilgan misolni tasnifi quyidagicha a va b o‘zgaruvchilar if amalini bajarish uchun kerakli bir qismi hisoblanadi. Shart bajarish jarayonida a bilan b ni solishtiradi agarda b soni a dan katta bo‘lsa shart bajarilib ekranga b son a dan katta so‘zi chiqadi.

Python dasturlash tilida if shart yoki boshqa amallarda uning tarkibiga kiruvchi qisim kodini yozish uchun berilayotgan operator oldidan bo‘sh qism qoldiriladi. Eslatma: Boshqa dasturlash tillarida(C++, C#) if, for va h.k amallar tarkibiga yozish uchun {} bilan tarkibiga olar edik!

Elif — "Aks holda agar" degan ma’nosi berib, undan oldingi keladigan if shart yolg‘on qiymat qabul qilsa ushbu shartga o‘tadi.
Unutmang!!! elif ishlatishdan oldin doimo elif yoki if keladi.

```
a = 9
b = 9
if b > a:
```

```
  print("b soni a dan katta")
elif a == b:
  print("a va b teng.")
```
>>>a va b teng.

else — "Aks Holda" degan ma’noni berib hech qanday shartni rost yolg‘onligini tekshirmaydi. Yuqoridagi shartlar barchasi yolg‘on qabul qilsa else ishlaydi. unutmang!! elif kabi yakka o‘zi kelmaydi.

```
a = 9
b = 11
if b == a:
  print("b soni  a tang")
else:
   print("b soni a ga teng emas!")
```
>>>b soni a ga teng emas!

IF qisqa shaklda

Agar siz if shart amalida qisqa shart tanasidan foydalanmoqchi bo‘lsangiz faqatgina bir qatorda yozishinigiz kerak bo‘ladi.

```
if a > b: print("a b dan katta")
```
>>>a b dan katta

IF...ELSE qisqa shakli

Qisqa shakli ya’ni if va else bir qatorda shakllantirish uchun shart amalimiz rost qiymat qabul qiladigan operatorlarni birinchi yozamiz so‘ngra else bajariladigan operatorlarni kiritamiz.

```
a = 18
b = 19
if a > b: print("A")
else: print("B")
```
>>>B

AND mantiqiy amali

and mantiqiy amalida har ikkala shart rost qiymat qabul qilgandagina rost bo‘ladi.

```
a = 19
b = 23
c = 29
if a < b and c > a:
   print("Ikkala shart rost qiymat qabul qildi!")
```
>>>Ikkala shart rost qiymat qabul qildi!

OR mantiqiy amali

Or mantiqiy amalida har ikkala shartdan hech bo‘lmaganda birisi rost qiymat qabul qilgandagina rost bo‘ladi.

```
a = 13
b = 15
c = 19
if a < b and c < a:
print("Ikkala shartdan birisi rost qiymat qabul qildi!")
>>>Ikkala shartdan birisi rost qiymat qabul qildi!
```

PASS

If shart tanasi bo‘lishi mumkin emas, agar siz shart operatori tarkibida hech qanday amal bajarmoqchi bo‘lmasangiz u holda siz pass qo‘llashingiz mumkin.

```
a = 13
b = 15
if b > a:
   pass
```

Masala: 1-999 gacha oraliqdagi sonlarni so‘zlarda ifodalovchi dastur tuzing.(masalan: 123-“bir yuz yigirma uch”).

```
y=int(input(’Son kiriting:’))
if y>=1000: print("1 dan 999 gacha bo‘lgan sonlarni kiriting!")
else: t1=int(y/100)
my_switch={ 1:"bir yuz",
2:"ikki yuz",
3:"uch yuz",
4:"to‘rt yuz",
5:"besh yuz",
6:"olti yuz",
7:"yetti yuz", 8:"sakkiz yuz", 9:"to‘qqiz yuz"
}
t2=y%100;
m=int(t2/10); switch={
1:"o‘n",
2:"yigirma",
3:"o‘ttiz",
4:"qirq",
5:"ellik",
6:"oltmish",
7:"yetmish",
```

```
8:"sakson",
9:"to‘qson"
}
t3=int(y/100); n=y%10;
myswitch={ 1:"bir",
2:"ikki",
3:"uch",
4:"to‘rt",
5:"besh",
6:"olti",
7:"yetti",
8:"sakkiz",
9:"to‘qqiz"
}
print(my_switch.get(t1,’’),switch.get(m,’’), myswitch.get(n,’’));
```

```
Python 3.10.1 (tags/v3.10.1:2cd268a, Dec  6 2021, 19:10:37) [MSC v.1929 64 bit (
AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.

===== RESTART: C:/Users/user/AppData/Local/Programs/Python/Python310/66.py =====
Son kiriting:913
to‘qqiz yuz o‘n uch
```

Nazorat savollari:

1. Pythoda tarmoqlanuvchi operatorlar qaysilar?
2. Pythonda or mantiqiy operatorlar bilan ishlash?
3. Pythonda and mantiqiy operatorlar bilan ishlash?
4. Python dasturlash tilida 2 sondan kattasini topuvchi dastur tuzing?

6-ma’ruza.Pythonda takrorlanish operatorlari.

Reja:

1. Python dasturlash tilida takrorlanish operatorlari.
2. Python dasturlash tilida range va xrange funksiyalari bilan ishlash.
3. Python dasturlash tilida break va continue.

Dasturlash davomida kodimizning biror qismini bir necha marta takrorlash talab etilishi mumkin. Misol uchun, ro‘yxat ichidagi har bir elementni alohida qatordan konsolga chiqarish, yoki bo‘lmasa har bir elementni kvdartaga oshirish va hokazo.

Mana shunday vaziyatlarda bizga **for** operatori yordam beradi. Dasturlashda bu **tsikl** (**loop**) deb ataladi.

Keling quyidagi misolni ko‘ramiz. Bizda mehmonlar ro‘yxati bor, biz har bir mehmonning ismini yangi qatordan chiqarmoqchimiz. Buning uchun quyidagi kodni yozamiz:

```
talabalar = [’Anvar’,’Ilhom’,’Malik’, ’Majid’,’Mamur’]
for talaba in talabalar:
print(talaba)
```

Natija:

```
Anvar
Ilhom
Malik
Majid
Mamur
```

Kodni tahlil qilamiz:

1-qatorda biz talabalar degan ro‘yxat yaratdik va uni mehmonlarning ismi bilan to‘ldirdik.

2-qatorda for tsiklini bohladik. Bu qator Pythonga mehmonlar degan ro‘yxatdan har bir elementini olib uni yangi, talaba degan o‘zgaruvchiga yuklashni buyuryapti (*o‘zgaruvchiga istalgan nom berishingiz mumkin. Biz tushunarli bo‘lishi uchun* talaba *deb atadik*)

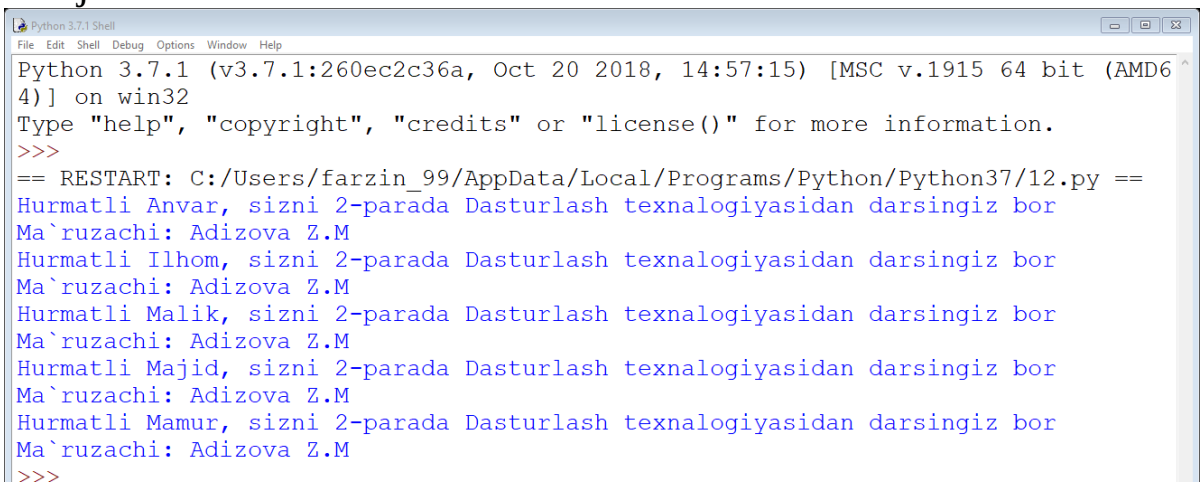
3-qatorda biz talaba degan o‘zgaruvchining qiymatini konsolga chiqardik. Bu tsikl to talabalar ro‘yxatida elementlar tugagunga qadar takrorlanadi.

"**For**" so‘zi ingliz tilidan "**uchun**" deb tarjima qilinadi.

Yuqoridagi kodni oddiy tilga tarjima qilsak *"talabalar ro‘yxatidagi har bir talaba* ***uchun*** *uning ismini konsolga chiqar"* degan ma’noni beradi.

for qanday ishlaydi
talabalar = [’Anvar’,’Ilhom’,’Malik’, ’Majid’,’Mamur’]
for talaba in talabalar:
  print(f"Hurmatli {talaba}, sizni 2-parada Dasturlash texnalogiyasidan darsingiz bor")
  print("Ma’ruzachi: Adizova Z.M")

Natija:



Yuqoridagi kodda 2-qator bu tsikl boshi deyiladi. Aynan shu qator kodimiz necha marta takrorlanishini aniqlaydi. Bizning holatimizda tsikl talabalar ro‘yxati ichidagi elementlar tugagunga qadar takrorlanadi. Tsikl boshlanishi ikki nuqta (:) bilan tugaydi. Undan keyingi 3 va 4-qatorlar bu tsiklning badani deyiladi.

Tsikl badani surilish (indentation) bilan ajratiladi, ya’ni tsiklning takrorlanuvchi qismi asosiy koddan bir muncha o‘ngroqqa surilgan bo‘ladi. Agar biz mana shu surilishni tark qilsak kodimiz xato beradi:

```
talabalar = [’Anvar’,’Ilhom’,’Malik’, ’Majid’,’Mamur’]
for talaba in talabalar:
print(f"Hurmatli {talaba}, sizni 2-parada Dasturlash texnalogiyasidan darsingiz bor")
print("Ma’ruzachi: Adizova Z.M")
```

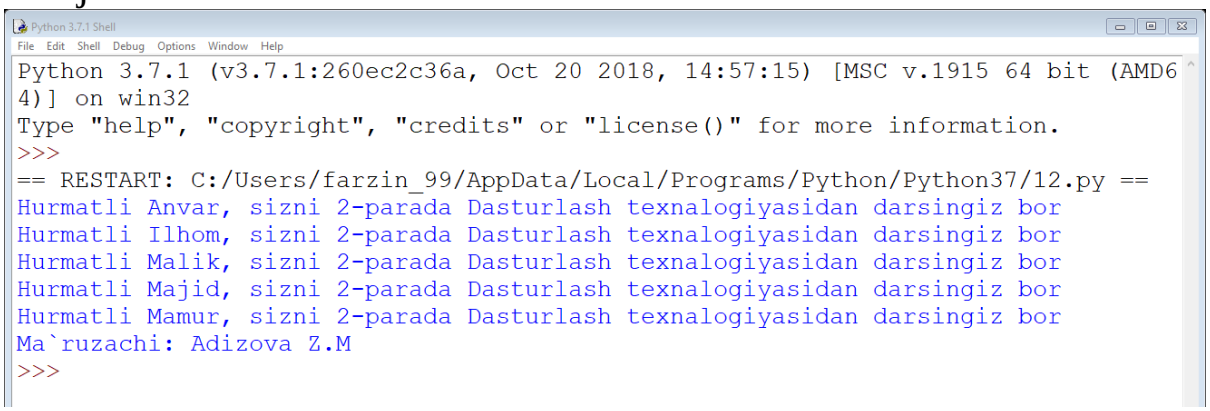
Natija:

IndentationError: expected an indented block

Ko‘rib turganingizdek for dan keyingi qatorni o‘ngga surmaganimiz uchun **indentation error** (surishda xatolik) degan xabarni oldik. Shunigdek, ko‘pchilik yo‘l qo‘yadigan yana bir xato, qo‘shimcha qatorlarni surish esdan chiqishi:

```
talabalar = [’Anvar’,’Ilhom’,’Malik’, ’Majid’,’Mamur’]
for talaba in talabalar:
  print(f"Hurmatli {talaba}, sizni 2-parada Dasturlash texnalogiyasidan darsingiz bor")
print("Ma’ruzachi: Adizova Z.M")
```

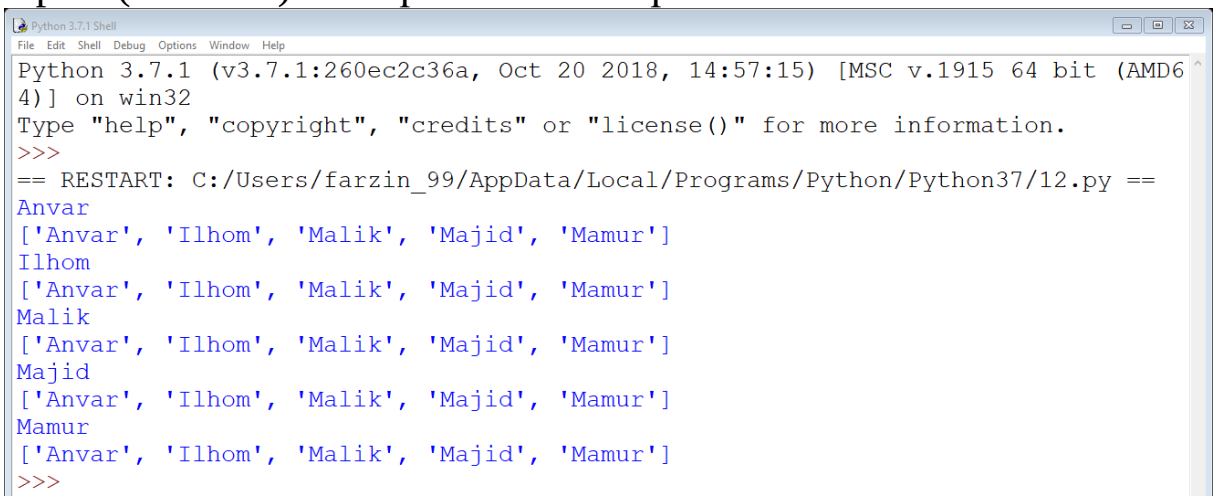
Natija:



Yuqoridagi kodimizda 4-qatorni o‘ngga surmaganimiz uchun, Python bu qatorni tsikl tashqarisida deb qabul qildi va faqatgina 1 marta, tsikl tugaganidan so‘ng bajardi.

Xuddi shu kabi agar takrorlanishi kerak bo‘lmagan kodni tsikldan so‘ng o‘ngga surib qo‘ysak Python bu qatorni tsiklning tarkibida deb hisoblab, qayta-qayta bajaradi:

```
talabalar = [’Anvar’,’Ilhom’,’Malik’, ’Majid’,’Mamur’]
for talaba in talabalar:
  print(talaba)
  print(talabalar) # bu qator tsikl tashqarisida bo‘lishi kerak
```



Yuoqirdagi misolda 5-qator o‘ngga surilib qolgani uchun Python bu qatorni ham bir necha bor takrorlab, konsolga chiqardi. To‘g‘ri kod quyidagicha bo‘ladi:

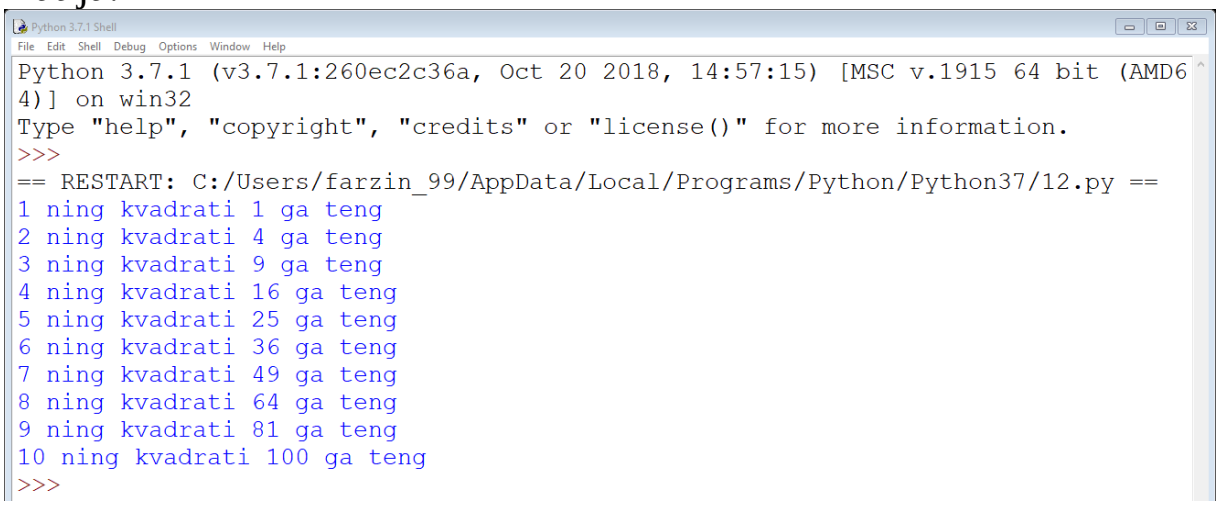
```
talabalar = [’Anvar’,’Ilhom’,’Malik’, ’Majid’,’Mamur’]
for talaba in talabalar:
   print(talaba)
print(talabalar)
```

for yordamida sonli ro‘yxatlar bilan ishlash

Misol:

```
sonlar = list(range(1,11))
for son in sonlar:
   print(f"{son} ning kvadrati {son**2} ga teng")
```

Natija:



for yordamida yangi ro‘yxat ham shakllantirish mumkin:

```
sonlar = list(range(11)) # 0 dan 10 gacha sonlar ro‘yxatini yaratamiz
sonlar_kvadrati =[] # bo‘sh ro‘yxat yaratamiz
for son in sonlar: # sonlar dagi har bir son uchun
   sonlar_kvadrati.append(son**2) # uning kv.ni hisoblab, sonlar_kvadrati ga yuklaymiz
print(sonlar)
print(sonlar_kvadrati)
```

Natija:

```
>>>[0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
[0, 1, 4, 9, 16, 25, 36, 49, 64, 81, 100]
```

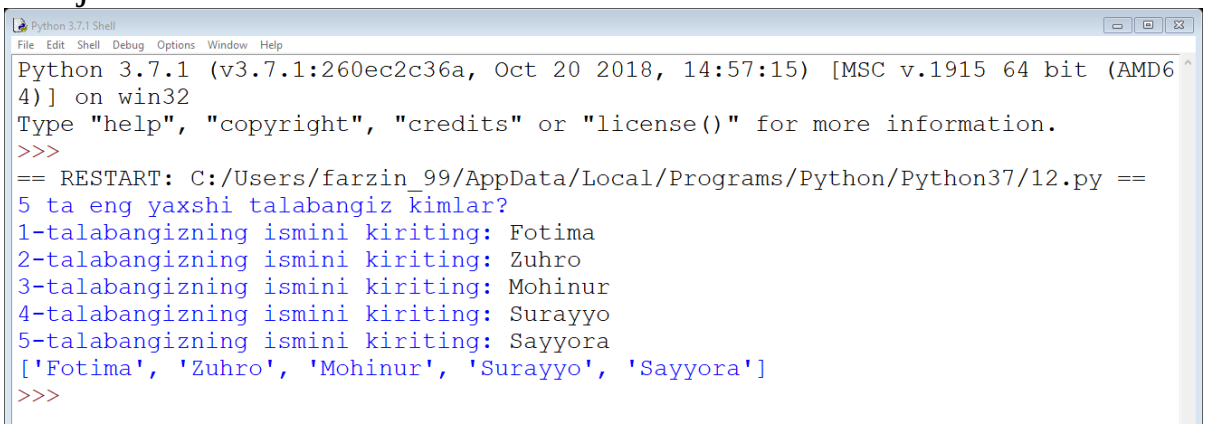
for va input()

for operatori va input() funktsiyasini jamlab, ro‘yxatni foydalanuvchidan olingan qiymatlar bilan to‘ldirish mumkin:

```
talabalar = [] # bo‘sh ro‘yxat
print("5 ta eng yaxshi talabangiz kimlar?")
for n in range(5): # n bu yerda 0 dan 4 gacha qiymatlar oladi
```

```
  talabalar.append(input(f"{n+1}-talabangizning ismini kiriting: "))
print(talabalar)
```

Natija:



Kodni tahlil qilamiz:

1-qatorda bo‘sh dostlar ro‘yxatini yaratdik

2-qatorda ekranga "5 ta eng yaxshi talabangiz kimlar?" degan xabarni chiqardik

3-qatorda tsiklni boshladik. range(5) funktsiyasi 0 dan 5 gacha sonlar ketma-ketligini yaratadi (0,1,2,3,4) tsikl esa n shularning har biriga teng bo‘lib chiqquncha davom etadi.

4-qatorda tsikl badani kelgan. Bu yerda biz foydalanuvchidan n+1 do‘stingizni kiriting deb so‘radik. Nima uchun n+1 (n emas)? Sababi n 0 dan 4 gacha qiymatlarni oladi, foydalanuvchiga tushunarli bo‘lishi uchun esa biz "0-talabangizni ismini kiriting:" deb emas, balki n+1 ya’ni 1-ismni kiriting deb murojat qilyapmiz.

5-qatorda shakllangan ro‘yxatni konsolga chiqardik.

for tsikli har qanday dasturlash tilining eng muhim qismlaridan hisoblanadi va biz bu operatoraga hali takror-takror qaytamiz.

While operator

Operator *while* shartli sikl operatori deyiladi, siklga kirishda oldin shartli ifoda hisoblanadi, agar uning qiymati noldan farqli bo‘lsa sikl tanasi bajariladi. Shundan so‘ng shartli ifodani hisoblash va sikl tanasi operatorlarini bajarish, shartli ifoda qiymati nolga teng bo‘lguncha davom etadi. Takrorlanishlar soni oldindan aniq bo‘lmaganda va qandaydir shartga bog‘liq bo‘lganda *while* operatoridan foydalanamiz.

While takrorlash operatorining sintaksisi quyidagicha:

while <shart>:

<operatorlar>

Bu yerda shart rost bo‘lganda operatorlar qismi bajariladi.
Quyidagi masalada 1 dan n gacha sonlarning yig‘indisini while da hisoblaymiz:

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

Nazorat savollari:

1. Pythoda takrorlanuvchi operatorlar qaysilar?
2. Pythonda for takrorlanish operatorlar bilan ishlash?
3. Pythonda while takrorlanuvchi operatorlar bilan ishlash?

7-ma’ruza. Pythonda satrlar.

Reja:

1. Python dasturlash tilida satrlar bilan ishlash.
2. Python dasturlash tilida satr metodlari.
3. Python dasturlash tilida satr funksiyalari.

Satrlar – bu belgilar ketma-ketligi. Ko‘p hollarda satrlar so‘zlar jamlanmasidan tashkil topadi. Pythonda satrlar bilan ishlash juda qulay. Bir qancha satr literallari mavjud. Ularni ko‘rib chiqamiz: Apostrof va qo‘shtirnoqdagi satrlar.

Apostrof va qo‘shtirnoqdagi satrlar bir narsa. Uni ikki xil variantda keltirilishiga sabab literallarga apostrof va qo‘shtirnoq belgilarini maxsus xizmatchi belgilardan foydalanmasdan kiritish mumkinligi deb hisoblanadi.

Ekran bilan ishlash ketma-ketliklari-xizmatchi belgilar

Ekran bilan ishlash ketma-ketliklari- klaviatura yodamida kiritish murakkab bo‘lgan belgilarni yozishga imkon beradi.

Xizmatchi belgilar	Vazifasi
\n	Keyingi qatorga o‘tish
\a	Qo‘ng‘iroq
\f	Keyingi betga o‘tish
\r	Koretkani qaytarish
\t	Gorizontal tabulatsiya
\v	Vertical tabulatsiya
\N{id}	Unicode ma’lumotlar bazasining ID identifikatori
\uhhhh	Unicode ning 16 lik ko‘rinishidagi 16 bitli belgisi
\Uhhhh. . .	Unicode ning 32 lik ko‘rinishidagi 32 bitli belgisi
\xhh	Belgining 16 lik kodi

\ooo	Belgining 8 lik kodi
\0	Null belgisi (satr oxiri belgisi emas)

Ko‘p qatorli satrlar

Pythonda satrlarni apostrof(’) va qo‘shtirnoqdan foydalanib hosil qilish mumkin. Apostrof (bir tirnoq(’)) yoki qo‘sh tirnoqni(") 3marta takrorlash orqali esa ko‘p qatorlik satrlarni xosil qilish mumkin. Milsol uchun:

Satr konstantalarini birlashtirish uchun ularni yonma-yon joylashtirishning o‘zi kifoya. Python avtomat ularni birlashtiradi. Misol uchun: "Ismingiz" "kim?" avtomat "Ismingiz kim?" ga aylanadi. Eslatma: Bir tirnoq va qo‘sh tirnoqdagi satrlar bir-biridan hech ham farq qilmaydi.

Satrlarning funksiya va metodlari

Shunday qilib satrlar bilan ishlash haqida gapirdik, endi satrlarning funksiyalari va metodlari haqida gapiramiz. Quyida satrlarning barcha funksiya va metodlari keltirilgan.

Asosiy operatsiyalar

Konkatenatsiyalash (qo‘shish)

Satrni takrorlash (dublikat qilish)

Satr uzunligi (len() funksiyasi)

Indeks bo‘yicha chiqarish

Misoldan ko‘rinib turibidiki Python manfiy indeks bo‘yicha chiqarishga ruxsat etadi, lekin hisoblash qator oxiridan boshlanadi.

Kesmani ajratib olish. Kesmani ajratib olish operatori:[X:Y]. X-kesmaning boshi, Y esa –oxiri. Y raqamli belgi kesmaga kirmaydi. Jimlik holatida birinchi indeks 0 ga teng, ikkinchi indeks esa qator uzunligiga teng bo‘ladi.

Bundan tashqari kesmani ajratib olishda qadamni belgilash mumkin. Satrlarning qo‘shimcha funksiya va metodlari. Metodlarni chaqirganga Pythondagi satrlar o‘zgarmaydigan ketma-ketliklar daRejasiga kirishini inobatga olishimiz kerak. Bu degani hamma funksiyalar va metodlar faqat yangi satrni tuzishi mumkin.

Shuning uchun hamma metodlar yangi satrni qaytaradilar, va u keyin boshqa nomga ega bo‘ladi.

S = ’str’; S = “str”; S = ’’’str’’’; S = “”“str”“”- Satrlarni literallari
S = “s\np\ta\nbbb”- ekran bilan ishlash ketma-ketliklari S = r”C:\temp\new”- Formatlashtirilmagan satrlar S = b”byte”- Baytlar qatori S1+S2- Konkatenatsiya(qo‘shish)
S1*3- Satrni takrorlash
S[i]- Indeks bo‘yicha murojaat
S[i:j:step]- Step qadamli i elementdan boshlab j elementgacha bo‘lgan kesmani ajratib olish.

Python dasturlash tilida satrlar bitta yoki ikki qo‘sh tirnoq ichida yoziladi. ’Salom’ bilan ’Salom’ ikkisi ham bir xil hisoblanadi. Satrni qora ekranga chiqarish uchun print() funksiyasidan foydalanishingiz mumkin.

Satr turiga ega o‘zgaruvchi yaratish.

Bir qatorli satrni o‘zgaruvchiga biriktirish uchun to‘g‘ridan to‘g‘ri python dasturlash tilida uchramaydigan kalit so‘zlarni ishlatmagan holda yozib ketsangiz bo‘ladi.

```
a=’salom dunyo‘
print(a)
```

Ko‘p qatorli satrni o‘zgaruvchiga biriktirish.
Ko‘p qatorli satrni biriktirish uchun 3 ta tirnoq tarkibida yozish imkoni mavjud. quyidagi misolga e’tibor bering.

```
b=’’’Python dasturlash tili
reytinngi baland dasturlash tillaridan
biri’’’
print(b)
```

Natija:

```
Python dasturlash tili
reytinngi baland dasturlash tillaridan
biri
```

Boshqa ko‘plab dasturlash tillari singari Python dasturlash tilida satrlar ifodalashda baytlardan iborat hisoblanadi. Biroq Python dasturlash tilida belgilar bo‘yicha ma’lumot turlari mavjud emas, bitta belgini uzunligini shunchaki 1 deb qabul qilasiz. [] kvadrat qavs ichida ma’lumotlar ifodalanadi.

```
z=’maktab’
print(z[1])
>>a
```

Satrni ma’lum bir qismini qirqib olish va uni biron bir o‘zgaruvchiga

yuklash uchun yoki bo‘lmasam ekranga chop qilish uchun kvadrat qavs ichiga boshlanishi va tugashini yozamiz va ikkitali nuqta orqali bir biridan ajratiladi.

[boslanish:tugatish]

```
 z=’maktab’
print(z[1:3])
>>ak
```

Teskari qirqish uchun - dan foydalanamiz. Bunda oxirgi simbol indeksi birinchi yoziladi.

```
z=’maktab’
print(z[-4:-1])
>>tab
```

Qator uzunligini olish uchun len() funktsiyadan foydalaning

```
z=’maktab’
print(len(z))
>>6
```

Satrni barcha elementlarini kichik qillish uchun lower() funksiyasidan foydalaniladi

```
z=’MaktaB’
print(z.lower())
>>maktab
```

Satrni barcha elementlarini katta qillish uchun **upper()** funksiyasidan foydalaniladi.

```
z=’MaktaB’
print(z.lower())
>>MAKTAB
```

Eslatma: Barcha satr funksiyalari yangi qiymatlarni qaytaradi. Ular asl qiymatini o‘zgartirmaydilar.

Nazorat savollari:

1. Pythoda starlar bilan ishlash metodlari qaysilar?
2. Pythonda ko‘p qatorli satrlar bilan ishlash?
3. Pythonda berilgan so‘zni kichkina harflar bilan ekranga chiqarish dasturini tuzing?
4. Pythonda satrlarni qirqishni misollar bilan tushuntiring.

8-ma’ruza.Pythonda ro‘yxatlar va to‘plamlar.

Reja:

1. Python dasturlash tilida ro‘yxatlar.
2. Python dasturlash tilida to‘plamlar.
3. Python dasturlash tilida ro‘yhat va to‘plam metodlari.

Python dasturlash tilida to‘rtta ma’lumotlar to‘plami mavjud. Set-tartibsiz va indekslanmagan to‘plam. Python-da to‘plamlar figurniy qavslar bilan yoziladi.

```
set99 = {"apricot", "banana", "apple"}
print(set99)
>>> {’apricot’, ’apple’, ’banana’}
>>> {’apple’, ’apricot’, ’banana’}
```

Eslatma: To‘plamlar tartibsiz, shuning uchun elementlarning qaysi tartibda paydo bo‘lishiga aniq bilmaysiz.

Set to‘plamda element mavjudligini tekshirish uchun quyidagicha yo‘l qo‘llasak bo‘ladi.

```
set99 = {"apricot", "banana", "apple"}
print("banana" in set99)
>>>True
```

To‘plam yaratilgandan so‘ng, siz uning elementlarini o‘zgartira olmaysiz, ammo yangi narsalarni qo‘shishingiz mumkin.

To‘plamga bitta element qo‘shish uchun add() usuldan foydalaning . Agar siz to‘plamga bir necha element qo‘shmoqchi bo‘lsangiz update() funksiyasidan foydalanishingiz mumkin.

```
set99 = {"apricot", "banana", "apple"}
set99.add("orange")
print(set99)
>>>>{’apricot’, ’orange’, ’apple’, ’banana’}
set99 = {"apricot", "banana", "apple"}
set99.update(["orange", "mango", "grapes"])
print(set99)
>>>{’orange’, ’apricot’, ’grapes’, ’apple’, ’mango‘, ’banana’}
```

To‘plamda nechta element borligini aniqlash uchun len() funksiyasidan foydalanamiz.

```
set99 = {"apricot", "banana", "apple"}
print(len(set99))
>>>3
```

To‘plamdagi elementni olib tashlash uchun remove()yoki discard()usulidan foydalaning.

```
set99 = {"apricot", "banana", "apple"}
set99.remove("banana")
print(set99)
>>> {’apple’, ’apricot’}
```

Eslatma: Agar olib tashlanadigan element mavjud bo‘lmasa, remove()xato yuzaga keladi. discard() funksiyasida xatolik yuzaga kelmaydi.

Quyidagi discard() usul yordamida "apple" ni olib tashlang :

```
set99 = {"apricot", "banana", "apple"}
set99.discard("apple")
print(set99)
>>> {’apricot’, ’banana’}
```

pop() Elementni olib tashlash uchun foydalanishingiz mumkin , ammo bu usul *oxirgi* elementni olib tashlaydi . Unutmangki, to‘plamlar tartibsiz, shuning uchun qaysi element o‘chirilishini bilmay qolasiz.pop()Usulning qaytish qiymati olib tashlangan element hisoblanadi.

```
set99 = {"apricot", "banana", "apple"}
x = set99.pop()
print(x) # uchirilan holda
print(set99) #oldingi holat
>>> apple
{’apricot’, ’banana’}
>>> banana
{’apricot’, ’apple’}
```

Python dasturlash tilida ikki yoki undan ortiq to‘plamlarni(set) ni qo‘shish uchun bir nechta usullari mavjud. union() va update() kabi funksiyalardan foydalanishingiz mumkin.

Eslatma: Har ikki union()va update() har qanday ikki nusxadagi ma’lumotlar istisno qiladi.

```
s1 = {"a", "b" , "c"}
s2 = {1, 2, 3}
s1.update(s2)
print(s1)
>>> {’a’, 1, ’c’, 2, 3, ’b’}
>>> {’c’, 1, 2, ’a’, 3, ’b’}
```

set tegishli to‘plamni yaratish uchun set() konstruktoridan foydalaniladi.

```
set99 = set(("apricot", "banana", "apple"))
print(set99)
```

{’banana’, ’apple’, ’apricot’}
>>> {’apple’, ’banana’, ’apricot’}
>>> {’apricot’, ’banana’, ’apple’}

Pythonda to‘plamlarda ishlatishingiz mumkin bo‘lgan o‘rnatilgan funksiyalar to‘plami mavjud.

add()	set ga element qo‘shish
clear()	Barcha elementni o‘chirib tashlash
copy()	to‘plandan nusxa olish
difference()	Ikki to‘plam orasidagi farqni qaytaradi.
difference_update()	Barcha kiritilgan elementlarni olib tashlaydi.
discard()	Belgilangan elementni olib tashlaydi
intersection()	To‘plam qaytaradi.
intersection_update()	Joriy to‘plamda ko‘rsatilmagan boshqa elementlarni olib tashlaydi.
isdisjoint()	Ikki to‘plamni kesishgan yoki yo‘qligini qaytaradi.
issubset()	Boshqa to‘plamda ushbu to‘plam bor yoki yo‘qligini qaytaradi.
issuperset()	Ushbu to‘plamda boshqa to‘plam mavjudmi yoki yo‘qliginiqaytaradi.
pop()	Bitta element o‘chirish
remove()	Belgilangan elementni o‘chirish
symmetric_difference()	Ikki to‘plamning nosimmetrik farqlari bilan to‘plamni qaytaradi
symmetric_difference_update()	ushbu to‘plamdan va boshqasidan nosimmetrik farqlarni kiritadi
union()	To‘plamlarning birlashishini o‘z ichiga olgan to‘plamni qaytaring
update()	Ushbu to‘plam va boshqalarni birlashishi bilan to‘plamni yangilang

Nazorat savollari:

1. Pythoda starlar bilan ishlash metodlari qaysilar?
2. Pythonda ko‘p qatorli satrlar bilan ishlash?
3. Pythonda berilgan so‘zni kichkina harflar bilan ekranga chiqarish dasturini tuzing?
4. Pythonda satrlarni qirqishni misollar bilan tushuntiring.

9-ma’ruza. Pythonda massivlar.

Reja:

1. Python dasturlash tilida bir o‘lchovli massivlar.
2. Python dasturlash tilida ko‘p o‘lchovli massivlar.
3. Massiv metod va funksiyalari.

Pythondagi massiv — bu bir xil turdagi ob’ektlarni saqlash uchun ishlatiladigan buyurtma qilingan ma’lumotlar tuzilishi. Funktsional imkoniyatlari jihatidan ular ro‘yxatlarga o‘xshashdir, ammo ularning kirish ma’lumotlari turiga, shuningdek o‘lchamlariga nisbatan ba’zi cheklovlar mavjud. Ushbu xususiyatga qaramay, massivlar Python dasturlash tilidagi ma’lumotlar to‘plamlari bilan ishlash uchun juda funktsional vosita hisoblanadi.

Massivlarni yaratish va to‘ldirish. Pythonda yangi qator qo‘shishdan (yaratishdan) oldin, bunday ob’ekt bilan ishlash uchun mas’ul bo‘lgan kutubxonani import qilishingiz kerak. Buning uchun dastur fayliga from array import * qator qo‘shilishi kerak. Massivlar bitta doimiy ma’lumotlar turi bilan o‘zaro aloqada bo‘lishga qaratilgan bo‘lib, natijada ularning barcha elementlari bir xil o‘lchamga ega. array funktsiyasidan foydalanib biz yangi ma’lumotlar to‘plamini yaratishimiz mumkin.

Massivlarni yaratishning umumiy sintaksisi quyidagicha:

array(massiv_turi, qiymatlar_ro‘yxati)

Quyidagi misol Python massivni qanday to‘ldirish kerakligini ko‘rsatib beradi: from array import *

massiv = array(’i’, [2, 5, 4, 0, 8])

Massiv funksiyasi ikkita argumentni oladi, birinchisi - bu yaratilgan massivning turi, ikkinchisi - uning qiymatlarining dastlabki ro‘yxati. Bu yerda massiv elementlarining ’i’ (2 baytli butun) tur. Buning o‘rniga 1 baytli belgi ’c’ (char turi)ni yoki 4 baytli ’f’ float turini kabi boshqa turlardan foydalanishimiz mumkin.

Quyidagi jadvalda massiv turlari keltirilgan:

Turning massivda yozilishi	C turi	Python turi	Minimal hajmi baytda
’b’	signed char	int	1
’B’	unsigned char	int	1
’u’	Py_UNICODE	unicode character	2
’h’	signed short	int	2
’H’	unsigned short	int	2
’i’	signed int	int	2
’I’	unsigned int	int	2
’l’	signed long	int	4
’L’	unsigned long	int	4
’q’	signed long long	int	8
’Q’	unsigned long long	int	8
’f’	Float	float	4
’d’	Double	float	8

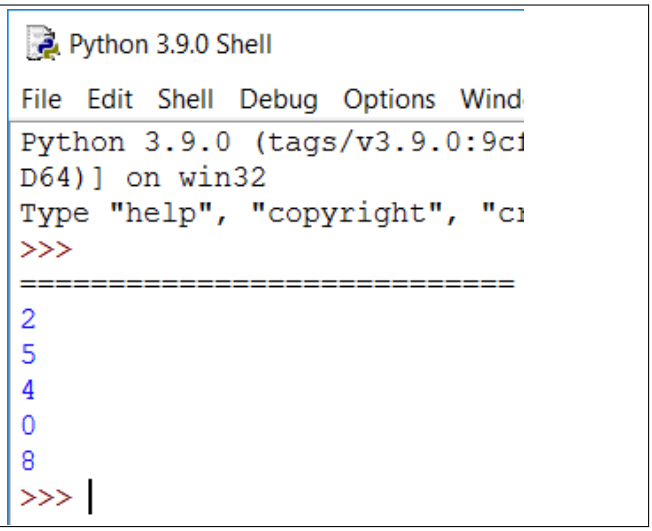
Shuni esda tutish kerakki, massiv faqat bitta turdagi ma’lumotlarni saqlashi mumkin, aks holda dasturni ishga tushirganimizda xatolik beradi va muvaffaqiyatsiz bo‘ladi.

Massiv elementiga murojaat qilish. Kvadrat qavs yordamida massiv elementiga murojaat qilishimiz mumkin. Masalan : massiv[2].

Massivlarni ekranga chiqarish. Dasturdagi har qanday ma’lumotlar bilan ishlashda vaqti-vaqti bilan ularni tekshirishga ehtiyoj bor. Bu ularni ekranda aks ettirish orqali osonlikcha amalga oshiriladi. Buni amalga oshirish uchun print deb nomlangan funktsiya yordam beradi.

Bu ilgari yaratilgan va to‘ldirilgan qator elementlaridan birini argument sifatida qabul qiladi. Quyidagi misolda for sikl operatori yordamida ma’lumotlar massivining har bir elementi vaqtinchalik identifikator i orqali chiqariladi:

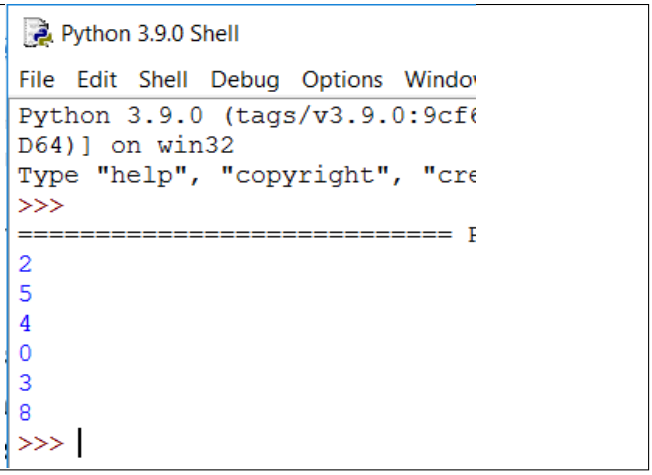
```
from array import *

massiv = array('i',
[2, 5, 4, 0, 8]) for
i in massiv:

  print(i)
```



Yuqoridagi kodning natijasida barcha element qiymatlari bo‘yicha takrorlanadiva ekranga chiqariladi.

Massivga element qo‘shish. Python qatoriga yangi element qo‘shish uchun insert metodidan foydalanish kerak. Buning uchun uni avval yaratilgan ob’ekt orqali chaqirish va ikkita qiymatni argument sifatida kiritish kerak. Birinchisi (4) massivdagi yangi elementning indeksiga, ya’ni uni joylashtirish kerak bo‘lgan joyga, ikkinchisi (3) qiymatning o‘zi uchun javobgardir.

```
from array import *

massiv = array('i', [2, 5,
4, 0, 8])

massiv.inser
t(4, 3) for
i in massiv:

  print(i)
```

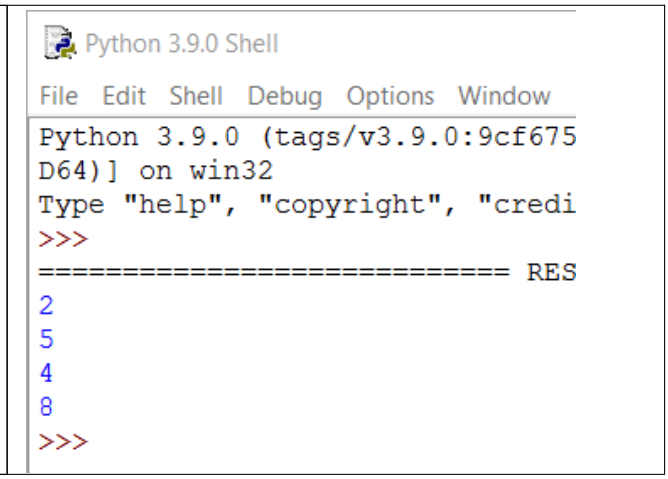


Shuni esda tutish kerakki, biz qatorga faqat ilgari yaratilgan ob’ekt tegishli bo‘lgan turdagi ma’lumotlarni qo‘shishimiz mumkin. Bunday operatsiyani bajarishda mavjud bo‘lgan elementlar soni dasturning ehtiyojlariga qarab ko‘payadi.

Elementni o‘chirish. Pythonda pop() metodi yordamida keraksiz elementlarni massivdan olib tashlash mumkin, uning argumenti

yacheka indeksi (3). Yangi element qo‘shilgandek bo‘lgani kabi, usulni misolda ko‘rsatilgandek, avval yaratilganob’ekt orqali chaqirish kerak.

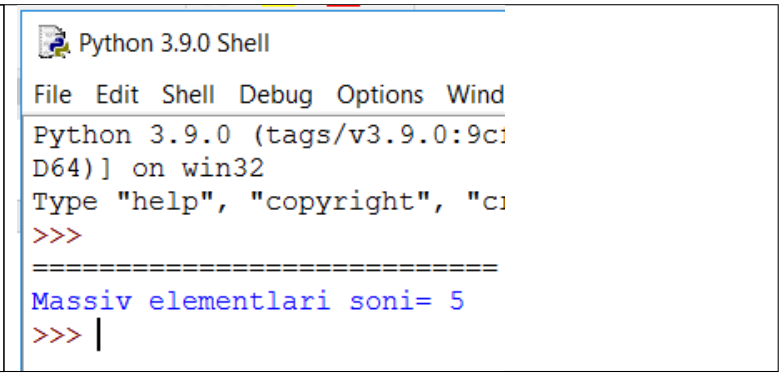
```
from array import *
massiv = array('i',
[2, 5, 4, 0, 8])
massiv.pop(3)

for i in
  massiv:
  print(i)
```



Ushbu operatsiyani bajargandan so‘ng, mavjud bo‘lgan xotira katakchalari soni elementlarning joriy soniga to‘g‘ri keladigan qilib massiv tarkibini o‘zgartiradi.

Massiv uzunligini olish. Dastur bajarilishida massivning uzunligi o‘zgarishi mumkinligi sababli, ba’zida uning tarkibidagi elementlarning hozirgi sonini bilish foydalidir. len() metodi Pythondagi massivning uzunligini (hajmini) butun son sifatida olish uchun ishlatiladi. Pythonda massiv elementlari sonini ekranga chiqarish uchun print()metodidan foydalanamiz:

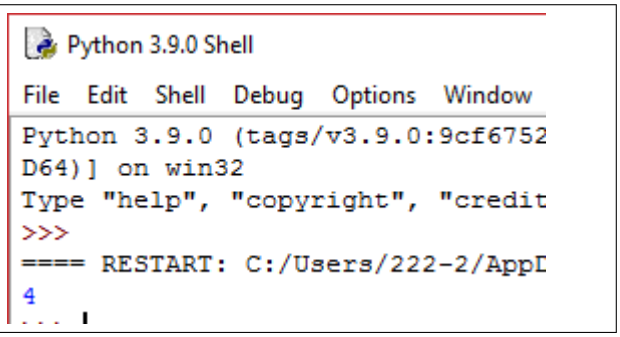
```
from array import *
massiv = array('i',
[2, 5, 4, 0, 8])
print("Massiv
              element
lari
soni=",len(massiv))
```



Yuqoridagi dastur kodidan ko‘rinib turibdiki, print() metodi argumenti sifatida len natijasini oladi, bu esa konsolga raqamli qiymatni chiqarishga imkon beradi.

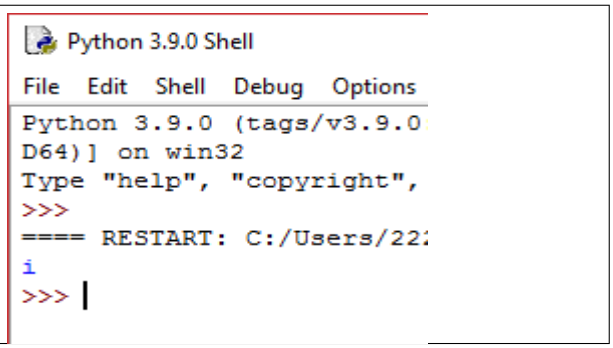
Pythonda massivlar bilan ishlashda qo‘llaniladigan funksiyalar va metodlar. Pythonda massivlar ishlashda qo‘llaniladigan bir nechta

```
from array import *

massiv = array('i',
[2, 5, 4, 0,8])
print(massiv.itemsize
)
```



metodlar mavjud bo‘lib, ularning eng asosiylari quyida keltirilgan:
array.typecode - Massivning elementlari turini aniqlash uchun ishlatiladi.Agar massivlar bir nechta bo‘lsa array.array(typecode) dan foydalaniladi.

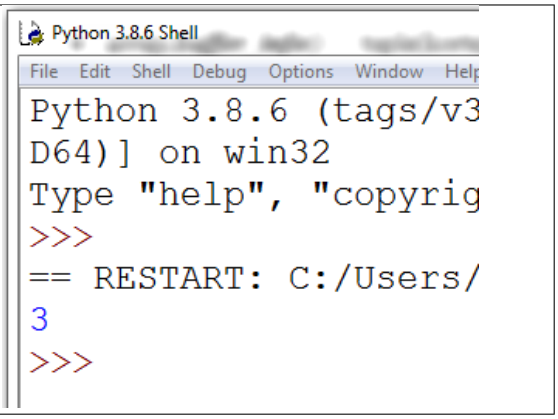
```
from array import *

massiv = array('i',
[2, 5, 4, 0, 8])
print(massiv.typecode)
```



array.itemsize- massivdagi bitta elementning baytdagi hajmini hisoblashuchun ishlatiladi.

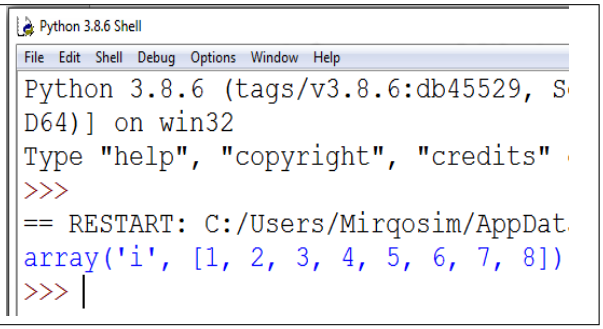
array.count(x) - masssivdagi *x* elementlar sonini qiymat sifatida qaytaradi ;

```
from array import *

massiv = array('i',
[2,2, 5, 4,4,4, 0,8])
print(massiv.count(4));
```



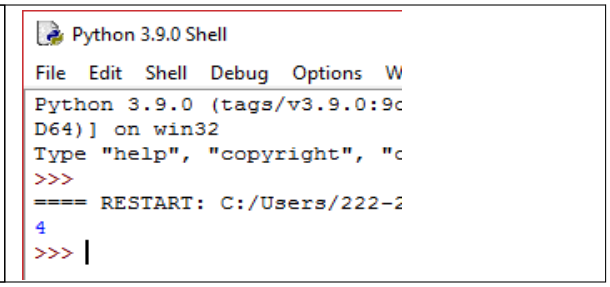
array.fromlist(ro‘yxat) – masssivga ro‘yxatdagi elementlarni qo‘shish uchunishlatiladi.

```
from array import *
massiv =
array('i',[1,2,3,4,
5]) list=[6,7,8]
massiv.fromlist(lis
t) print(massiv);
```



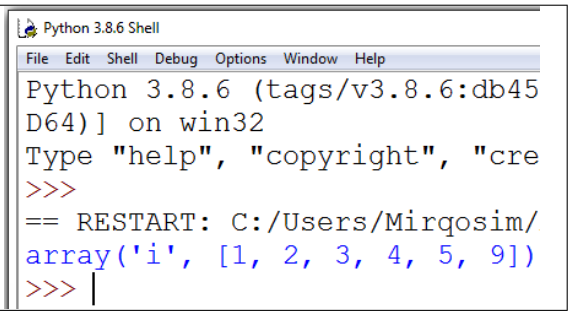
array.index(x) – massivdagi *x* elementining joylashgan indeksini qiymat
sifatida qaytaradi. Agar bunday element massivda mavjud bo‘lmasa, u holda
ValueError istisno holati ro‘y beradi;

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



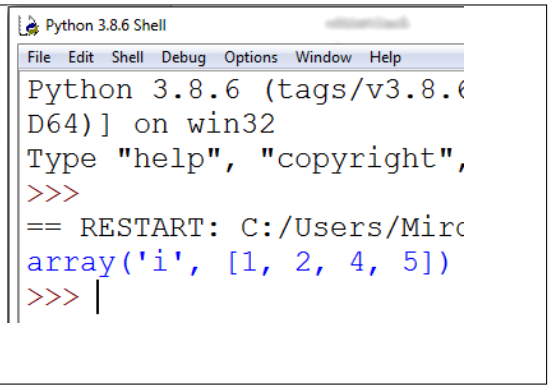
append()–metodimassivningoxirigayangielementqo‘shishuchun foydalaniladi.

```
from array import *
massiv = array('i',
[1,2,3,4,5])
massiv.append(9)
print(massiv)
```



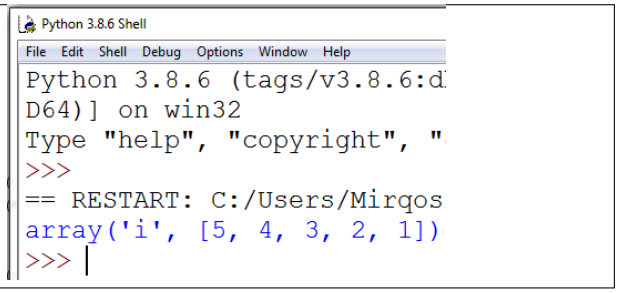
array.remove(x) — massivdan *x* elementini o‘chirish. Ushbu metod ro‘yxatdagi birinchi uchragan *x* elementini o‘chiradi. Agar bunday element ro‘yxatda mavjud bo‘lmasa *ValueError* istisno holati ro‘y beradi.

```
from array import *

massiv = array('i',
[1,2,3,4,5])
massiv.remove(3)
print(massiv)
```



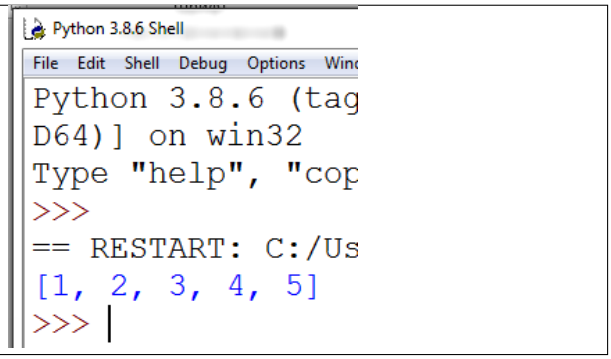
array.reverse() - massiv elementlarini teskari tartibda joylashtirish uchun qo‘llaniladi . Bundan tashqari, Python massiv bilan ishlashda qo‘llaniladigan bir nechta standart funksiyalarni ham o‘z ichiga qamrab olgan:

```
from array import *

massiv = array('i',
[1,2,3,4,5])
massiv.reverse()
print(massiv)
```



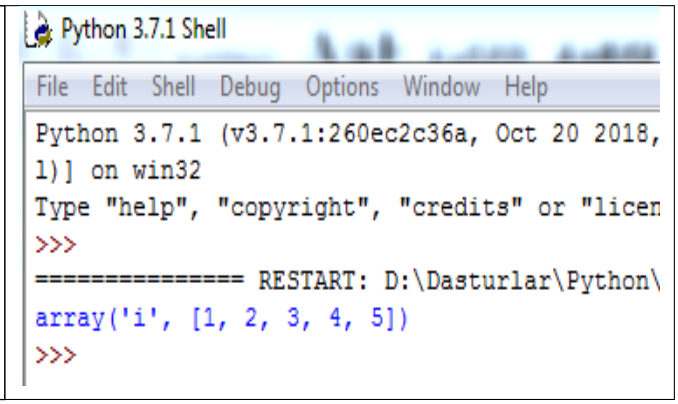
array.tolist() -massivni ro‘yxatga aylantirish uchun qo‘llaniladi.

```
from array import *

massiv = array('i',
[1,2,3,4,5])
list=massiv.tolist(
)

print(list)
```



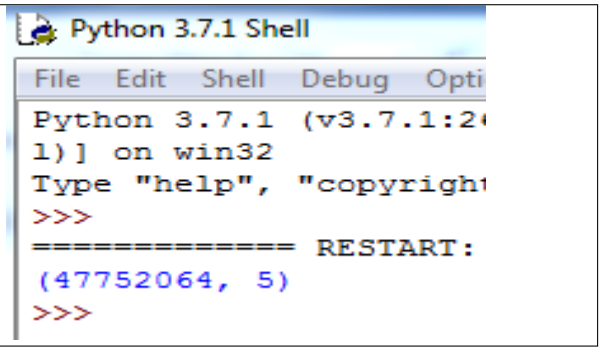
array.tofile(f) - massivni ochiq faylga yozish uchun ishlatiladi.
array.fromfile(F,N) — fayldan N elementni o‘qiydi va ularni massiv oxiriga qo‘shib qo‘yadi. Ikkilik o‘qish uchun fayl ochilishi kerak. Agar N dan kam element mavjud bo‘lsa, *ValueError* istisnoli tashlanadi, ammo mavjud bo‘lgan elementlar qatorga qo‘shiladi.

```
import array
f=open("array.bin","wb"
)
massiv=array.array("i",
[1,2,3,4,5])
massiv.tofile(f)

f.close()
```



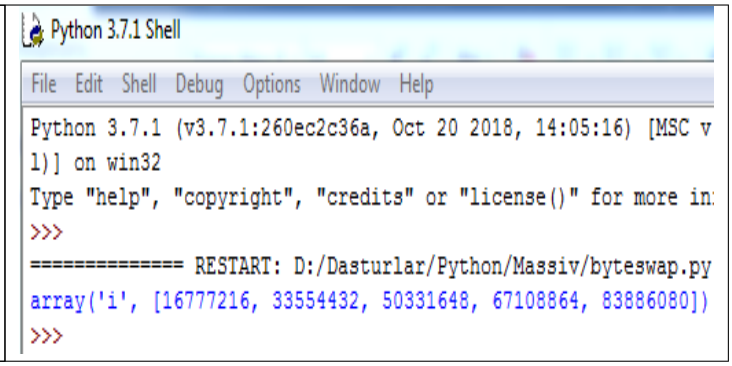
array.buffer_info() - tuple(kortej) xotiraning joylashuvi, uzunligini aniqlaydi.Past daRejadagi operatsiyalar uchun foydalidir.

```
import array
massiv=array.array('i',[1,
2,3,4,5])
print(massiv.buffer_info()
)
```



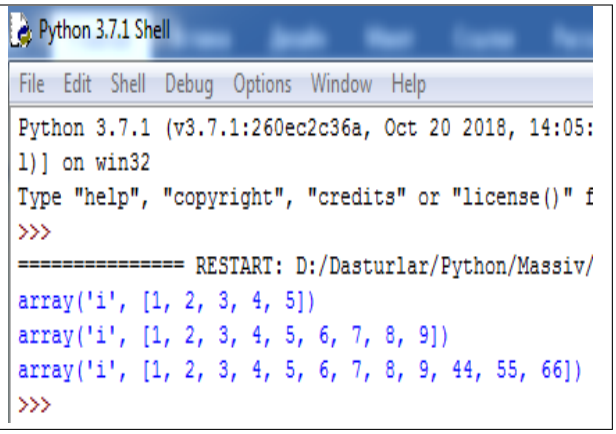
array.byteswap() - massivning har bir elementida baytlarning tartibini o‘zgartirish. Chunki boshqa bayt tartibida mashinada yozilgan fayldan ma’lumotlarni o‘qishda foydalidir;

```
from array import
array
my_array=array('i',[1
,2,3,4,5])
my_array.byteswap()
print(my_array)
```



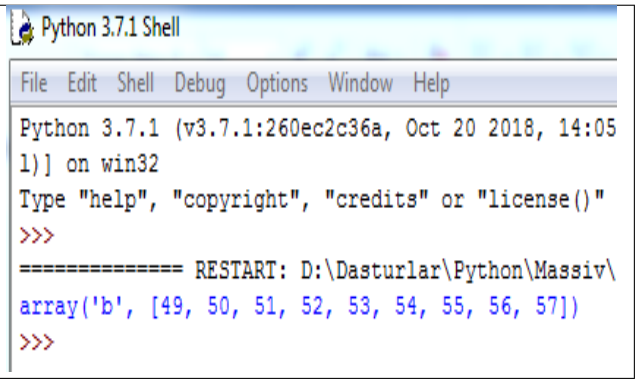
array.extend(iter)-massivgaob’ektdanelementlarniqo‘shishuchun foydalanadi.

```
from array import array
my_array=array('i')
my_array.extend([1,2,3,4,5
]) print(my_array)
my_array.extend(range(6,10
)) print(my_array)
my_array.extend(array('i',
[44,55,66]))
print(my_array)
```



array.frombytes (b) - bir qator baytlardan massiv hosil qiladi. Baytlar soni massivdagi bitta element kattaligining ko‘paytmasi bo‘lishi kerak.

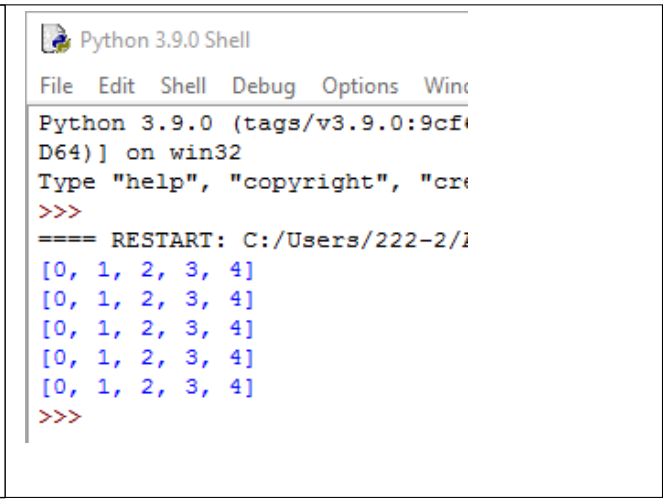
```
from array
import array
my_array =
array('b')

my_array.frombytes(b'1234
56789') print(my_array)
```



Pythonda ikki va ko‘p o‘lchovli massivlar

Ikki o‘lchovli massiv. Ba’zi hollarda oddiy bir o‘lchovli massiv ma’lum bir ma’lumot to‘plamini to‘g‘ri ko ’rsatish uchun etarli emas. Python dasturlash tilida ikki o‘lchovli va ko‘p o‘lchovli massivlar mavjud emas, ammo ushbu platformaning asosiy imkoniyatlari ikki o‘lchovli ro‘yxatni tuzishni osonlashtiradi. Ushbu dizayn elementlari quyidagi misolda ko‘rsatilgandek to‘ldirilib, ustunlar va qatorlarga joylashtirilgan.

```
from array
import * d1 =
[]

for j in
  range(5):
  d2 = []

  for i in
    range(5):
    d2.append
    (i)
```

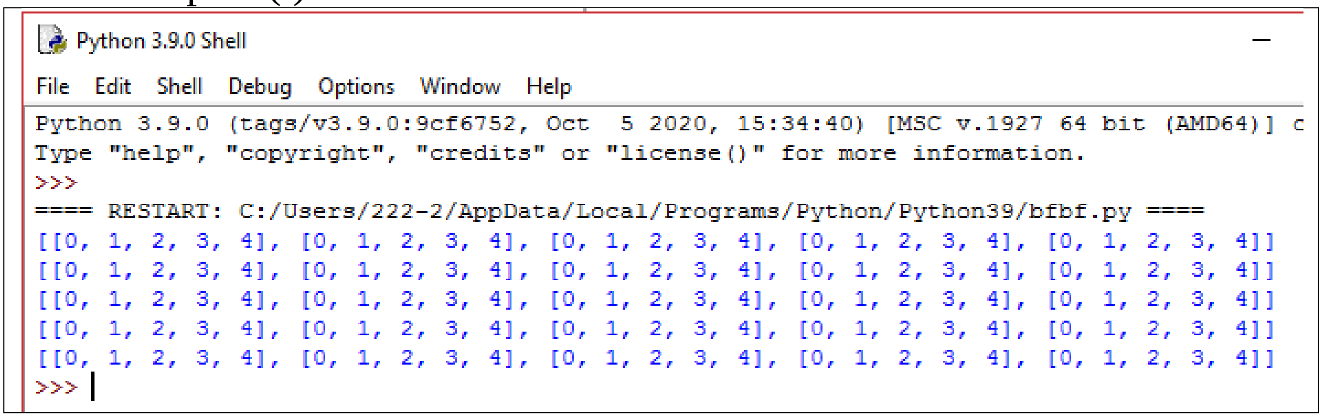


Bu yerda biz ikki o‘lchovli ma’lumotlar to‘plamini amalga oshirishning asosiy g‘oyasi bitta katta d1 ro‘yxati ichida bir nechta d2 ro‘yxatlarini yaratish ekanligini ko‘rishimiz mumkin. Ikki o‘lchamli 5 × 5 matritsani nol bilan avtomatik to‘ldirish uchun ishlatiladi. Qo‘shish va diapazon usullari ushbu vazifani yengishga yordam beradi, ularning birinchisi ro‘yxatga yangi element qo‘shadi (0), ikkinchisi esa uning qiymatini (5) o‘rnatishga imkon beradi. Shuni ta’kidlash kerakki, for uchun har bir yangi tashqi (j) yoki ichki (i) ro‘yxatlarning joriy elementini ifodalovchi o‘z vaqtinchalik o‘zgaruvchisidan foydalanadi.

Ko‘p o‘lchovli ro‘yxatning kerakli katakchasiga uning koordinatalarini to‘rtburchaklar ichida ko‘rsatib, satrlar va ustunlarga e’tibor qaratishingiz mumkin: d1 [1] [2].

Ko‘p o‘lchovli massiv. Murakkab ro‘yxat sifatida ko‘rsatilgan ikki o‘lchovli qatorda bo‘lgani kabi, ko‘p o‘lchovli qator ham ro ’yxat ichida ro‘yxat tarzida amalga oshiriladi. Quyidagi misolda uch o‘lchamli(5x5x5) massiv yaratishni ko‘rib chiqamiz:

```
from array import *d1 = []
for k in range(5):d2 = []
for j in range(5):d3 = []
for i in range(5):d3.append(i)
d2.append(d3)d1.append(d2)
for i in d1:print(i)
```



Ikki o‘lchovli massivga o‘xshab, to‘rtburchaklar ichidagi ko‘rsatkichlar yordamida yuqorida qurilgan ob’ekt katakchasiga murojaat qilishimiz mumkin.

Masalan, d1 [4] [2] [3].

Massivlar odatda Python dasturlash tilidagi bir xil turdagi ma’lumotlar to‘plamlari bilan o‘zaro aloqada bo‘lish uchun ishlatiladi. Platformaning standart kutubxonasi sizga tegishli funktsiyalar

yordamida uning tarkibini boshqarish qobiliyatini ta’minlaydigan bunday tuzilma bilan samarali ishlashga imkon beradi. Bundan tashqari, Python sathlar soniga cheklovlarsiz ro‘yxatlarning ko‘p o‘lchovli namoyishini qo‘llab-quvvatlaydi.

Nazorat savollari:

1. Massiv deb nimaga aytiladi?
2. Massiv nechta turga bo‘linadi?
3. Bir o‘lchovli massivning umumiy ko‘rinishi?
4. Ikki o‘lchovli massivning umumiy ko‘rinishi?
5. Massivning faollashtirish usullari?

10-ma’ruza.Pythonda funksiyalar.

Reja:

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

Funksiya – bu ko‘p marta ishlatiladigan dastur bo‘lagi. Funksiyalar ma’lum buyruqlar blokini ko‘rsatilgan nom bilan saqlash va shu blokni dasturning istalgan joyida, istalgan miqdorda bajarish imkonini beradi. Funksiyalar def zahira so‘zi orqali aniqlanadi. Bu so‘zdan so‘ng funksiya nomi, undan so‘ng qavs va shu qavs ichida bir necha o‘zgaruvchilarni ko‘rsatish mumkin bo‘ladi va oxirida ikki nuqta (:) yoziladi. Shulardan so‘ng funksiyani tashkil qiluvchi buyruqlar bloki yoziladi. Quyidagi misolda buning oson ekanligini ko‘rish mumkin.

Misol:

```
def funksiya1():
print(’Salom, Dunyo!’) # funksiyaga tegishli blok
funksiya1() # funksiyani chaqirish
funksiya1() # ya’na bir marta funksiyani chaqirish
```

Natija:

```
>>>Salom, Dunyo!
Salom, Dunyo!
```

Bu qanday ishlaydi:

Biz funksiya1 funksiyasini yuqorida aytib o‘tilgan qoida bo‘yicha aniqladik. Bu funksiya xech qanday paramert qabul qilmaydi shuning uchun qavs ichida xech qanday parametr yozilmadi. Funksiya parametri – bu qandaydir kiruvchi qiymatlar bo‘lib, tegishli natija olish uchun biz uni funksiyaga berishimiz mumkin.

E’tibor bering, bitta funksiyani ko‘p marta chaqirishimiz mumkin, demak, aynan bir xil dastur kodini qayta-qayta yozishga hojat yo‘q.

Funksiya parametrlari

Funksiyalar parametrlar, ya’ni funksiyaga berilishi mumkin bo‘lgan qiymatlar qabul qila oladi va ular ustuda biror amal bajarishi mumkin. Bu parametrlar o‘zgaruvchilarga o‘xshaydi. Faqat ulardan farqi bu o‘zgaruvchilarning qiymati funksiyani chaqirish vaqtida o‘rnatiladi. Funksiya ish boshlagan vaqtda bularga qiymat biriktirilgan bo‘ladi.

Parametrlar funksiya aniqlanayotgan vaqtda qavs ishida vergul bilan ajratilgasn holda ko‘rsatiladi. Ularga qiymatni funksiyani chaqirganimizda biriktiramiz. Ushbu atamalarga e’tibor bering: funksiya e’lon qilinayotgan vaqtda ko‘rsatilgan nomlar parametrlar, funksiyani chaqirayotganimizda unga berilgan qiymatlar esa argumentlar deyiladi.

Misol:

```
def maximum(a, b):
if a > b:
print(a, ’katta’)
elif a == b:
print(a, ’teng‘, b)
else:
print(b, ’katta’)
maximum(9, 11) # qiymatlarni to‘g‘ridan-to‘g‘ri berish
x = 99
y = 77
maximum(x, y) # o‘zgaruvchilarni argument sifatida uzatish.
```

Natija:

```
>>>11 katta
99 katta
```

Bu qanday ishlaydi:

Bu yerda biz a va b parametrlardan foydalanadigan maximum funksiyasini e’lon qildik. Bu funksiyada oddiygina if..else operatoridan foydalangan holda sonlarning kattasini aniqlaymiz va uni ekranga chop etamiz.

Maximum funksiyasini birinchi marta chaqirganimizda sonlarni to‘g‘ridan-to‘gri argument sifatida beramiz. Ikkinchi marta chaqirganimizda esa o‘zgaruvchilarni argument sifatida beramiz. Maximum (x, y) x argument qiymatini a parametrga, y argument qiymatini esa b parametrga biriktiradi. Yuqoridagi ikkala holatda ham funksiya bir xilda ishlaydi.

Mahalliy(локальные) parametrlar.

Funksiyaning ichida e’lon qilingan o‘zgaruvchilar xuddi shu nomdagi funksiya tashqarisida e’lon qilingan o‘zgaruvchilar bilan xech qanday bog‘liklikka ega emas, ya’ni bu o‘zgaruvchilar mahalliy o‘zgaruvchilar hisoblanadi. Bu o‘zgaruvchining ko‘rinish maydoni(область видимости) deyiladi. Har bir o‘zgaruvchining

koʻrinish maydoni oʻzgaruvchi aniqlangan amallar bloki va shu oʻzgaruvchi e’lon qilingan nuqta bilan chegaralangan.
Misol:

```
x = 99
def func(x):
    print("x teng", x)
    x = 9
    print("Mahalliy x qiymatini", x, "ga oʻzgartiramiz")
func(x)
print("x qiymati qanday boʻlsa, shunday turibdi", x)
```

Natija:

```
>>> x teng 99
Mahalliy x qiymatini 9 ga oʻzgartiramiz
x qiymati qanday boʻlsa, shunday turibdi 99
```

Bu qanday ishlaydi:

Funksiyaning birinchi qatorida, x oʻzgaruvchisiga biriktirilgan qiymatni birinchi bor chop etishda Python funksiyaning yuqorisida, ya’ni asosiy blokda e’lon qilingan parametrning qiymatidan foydalanadi.

Soʻng x oʻzgaruvchiga 2 qiymatni biriktiramiz. x oʻzgaruvchisi bizning funksiyamiz uchun mahalliy hisoblanadi. Shuning uchun x oʻzgaruvchining qiymatini funksiya ichida oʻzgartirganimizda, funksiya tashqarisida aniqlangan x oʻzgaruvchi qiymati xech qanday oʻzgarishsiz qoladi.

Dastur oxirida print funksiyasi yordamida asosiy blokda aniqlangan x oʻzgaruvchisi qiymatini chop etamiz va funksiya ichida x oʻzgaruvchisi qiymatining oʻzgartirilishi xech qanday ta’sir qilmaganligini koʻrishimiz mumkin boʻladi.

"global" zahira soʻzi (зарезервированное слово)

Funksiya ichidagi biror oʻzgaruvchiga yuqori daRejadagi biror qiymatni (funksiya yoki klassning koʻrinish maydoni emas) biriktirish uchun Pythonga uning nomi mahalliy emasglobal ekanligini koʻrsatish darkor. Buni "global" zahira soʻzi yordamida amalga oshirish mumkin. "global" zahira soʻzini koʻrsatmasdan funksiya tashqarisida aniqlangan biror oʻzgaruvchi qiymatini biriktirib boʻlmaydi.

Funksiya tashqarisida aniqlangan biror oʻzgaruvchi qiymatini funksiya ichida ishlatish mumkin (faqat sharti funksiya ichida xuddi shu nom bilan biror oʻzgaruvchi aniqlanmagan boʻlishi kerak). Lekin

bunday holatlardan qochish kerak. Sababi dastur kodini o‘qiyotgan odamga o‘zgaruvchi qayerda aniqlanganligi tushunarsiz bo‘lishi mumkin.
"global" zahira so‘zini ishlatish esa o‘zgaruvchi eng yuqori blokda aniqlanganini aniq ko‘rsatadi.
Misol:

```
x = 99
def func():
    global x
    print("x teng", x)
    x = 9
    print("Global x qiymatini", x, "ga o‘zgartiramiz")
func()
print("x ning qiymati", x)
```

Natija:

```
>>>x teng 99
Global x qiymatini 9 ga o‘zgartiramiz
x ning qiymati 9
```

Bu qanday ishlaydi:
"global" zahira so‘zi funksiya ichidagi o‘zgaruvchini global o‘zgaruvchi ekanligini bildiradi va bu shuni anglatadiki qachonki biz bu o‘zgaruvchi qiymatini funksiya ichida o‘zgartirsak, bu o‘zgarish asosiy blokdagi o‘zgaruvchi qiymatida xam aks etadi.
Bitta "global" zahira so‘zi yordamida bir necha o‘zgaruvchini aniqlash mumkin:

```
global x, y, z
```

"nonlocal" zahira so‘zi

Biz qanday qilib global va mahalliy o‘zgaruvchilarga murojat qilishni ko‘rib chiqdik. Ya’na bir korinish maydoni borki, bu maydon global va mahalliy maydonlar o‘rtasini bildiruvchi "mahalliy emas" (нелокальной) ko‘rinish maydoni mavjud. Bu ko‘rinish maydoni siz funksiya ichida funksiya aniqlaganingizda uchraydi.

Pythonda funksiyani xohlagan joyingizda aniqlashingiz mumkin.
Misol:

```
def func_outer():
    x = 2
    print("x teng", x)
    def func_inner():
```

```
        nonlocal x
        x = 5
    func_inner()
    print("maxalliy x", x, "ga o‘zgardi")
func_outer()
```

Natija:

```
>>>x teng 2
maxalliy x 5 ga o‘zgardi
```

Bu qanday ishlaydi:

func_inner funksiyasi birinchi qatorida aniqlangan x o‘zgaruvchi mahalliy ko‘rinish maydonida emas (o‘zgaruvchini aniqlash func_inner blokiga kirmaydi), global ko‘rinish maydonida ham emas (o‘zgaruvchi dasturning asosiy blokida ham emas). Biz aynan x o‘zgaruvchini ishlatish uchun uni quyidagich e’lon qilamiz: nonlocal x.

"nonlocal x" yozuvini "global x" yozuviga, keyin esa "global" zahira so‘zini o‘chirib tashlang va bu ikki holdagi o‘zgarishlarni kuzating.

Argumentlarning qiymat ko‘rsatilmaganda ishlatiladigan qiymati (Значения аргументов по умолчанию)

Ko‘p hollarda funksiyaning ba’zi parametrlari majburiy bo‘lmasligi mumkin. Agar dasturchi bunday parametrlar uchun qiymat bermasa, u holda funksiya aniqlanishida ko‘rsatilgan qiymat (значение по умолчанию) ishlatiladi. Buning uchun funksiya e’lon qilinish joyida parametr nomidan keyin o‘zlashtirish operatori (=) va undan so‘ng biror qiymat beriladi.

E’tibor bering, funksiya aniqlanishida ko‘rsatilgan qiymat (значение по умолчанию) konstanta bo‘lishi kerak. Yoki aniqroq qilib aytganda qiymati o‘zgarmas bo‘lishi kerak.

Misol:

```
def say(xabar, qiymat = 1):
    print(xabar * qiymat)
say(’Salom’)
say(’Dunyo‘, 5)
```

Natija:

```
>>>Salom
DunyoDunyoDunyoDunyoDunyo
```

Bu qanday ishlaydi:say funksiyasi satrni ko‘rsatilgan miqdorda ekranga chop etish uchun foydalaniladi. Agar biz qiymat ko‘rsatmasak

satr bir marta chop etiladi. Biz bunga qiymat parametriga boshlang‘ich 1 qiymatini berib erishishimiz mumkin.

Say funksiyasini birinchi marta chaqirganimizda biz unga faqat satr qiymatini beramiz va funksiya uni bir marta chop etadi. Ikkinchi marta chaqirganimizda esa berilgan satrni 5 marta takrorlashni ko‘zda tutgan holda unga 5 qiymatli argument beramiz. Boshlang‘ich qiymat bilan parametrlar ro‘yxati oxiridagi parametrlar qiymatlanishi mumkin. Shunday qilib boshlang‘ich qiymatli parametrlar qiymatsiz parametrlardan oldin kelmasligi kerak.

Bu parametrlarga qiymatlar ularning joylashuviga qarab biriktirilishi bilan bog‘liq. Misol uchun, def func(a, b=5) mumkin, lekin def func(a=5, b) kabi e’lon qilish mumkin emas.

Kalit argumentlar (Ключевые аргументы)

Agar biror funksiya ko‘p parametrga ega bo‘lsa va uni chaqirish joyida bu parametrlardan faqat ayrimlariga qiymat ko‘rsatilishi kerak bo‘lsa, u holda bu parametrlar qiymatlari ularning nomi bo‘yicha berilishi mumkin – bu kalit parametrlar deyiladi. Bu holda argumentlarni berish uchun ularning pozitsiyasi emas, nomi (kalit) ishlatiladi.

Bunday uslubni ikki xil afzalligi mavjud: birinchidan, funksiyadan foydalanish oson bo‘ladi. Sababi argumentlarning ketma-ketligini rioya qilish zarur bo‘lmaydi; ikkinchidan, faqat tanlangan argumentlarga qiymat berish mumkin bo‘ladi va qolgan argumentlar boshlang‘ich qiymatlarga ega bo‘ladi.

Misol:

```
def func(a, b=5, c=10):
   print(’a teng‘, a, ’, b teng‘, b, ’, c teng‘, c)
func(3, 7)
func(25, c=24)
func(c=50, a=100)
```

Natija:

```
>>>a teng 3, b teng 7, c teng 10
a teng 25, b teng 5, c teng 24
a teng 100, b teng 5, c teng 50
```

Bu qanday ishlaydi:

func nomli funksiya bitta boshlang‘ich qiymatsiz parametr va ikkita boshlang‘ich qiymatli parametrga ega.

func(3, 7) funksiyaning birinchi chaqirilishida a parametr

3, b parametr 7, c parametr esa boshlang‘ich qiymat 10 qabul qiladi.
func(25, c=24) funksiyaning ikkinchi chaqirilishida a parametr argument pozitsiyasi bo‘yicha 25 qiymat qabul qiladi. Shundan song c parametr nom ya’ni kalit parametr bo‘yicha 24 qiymat qabul qiladi. b parametr esa boshlang‘ich qiymatga ko‘ra 5 qiymat qabul qiladi.
func(c=50, a=100) funksiyaning uchunchi chaqirilishida biz barcha qiymatlar uchun kalit argumentlardan foydalanamiz. E’tibor bering, funksiyani e’lon qilishda c parametr aparametrdan keyin ko‘rsatilgan bo‘lsa ham, qiymat biriktirish vaqtida c parametrga aparametrdan oldin qiymat biriktiryapmiz.
Ixtiyoriy miqdordagi parametrlar (Переменное число параметров)
Ba’zan funksiyani ixtiyoriy miqdordagi parametr qabul qila oladigan holda e’lon qilish kerak bo‘lishi mimkin. Bunga yulduzchalar yordamida erishishimiz mumkin.
Misol:

```
def total(initial=5, *numbers, **keywords):
count = initial
for number in numbers:
count += number
for key in keywords:
count += keywords[key]
return count
print(total(10, 1, 2, 3, vegetables=50, fruits=100))
```

Natija:
>>>166
Bu qanday ishlaydi:
Agar biz parametrni yulduzcha (*) bilan e’lon qiladigan (misol uchun param) bo‘lsak, shu pozitsiyadan boshlab oxirigacha bo‘lgan barcha pozitsiya argumentlari param nomlik kortejga yig‘iladi. Bizning holatda numbers kortejida (1, 2, 3) qiymat mavjud bo‘ladi.
Shunga o‘xshash agar biz ikkita yulduzcha (**) bilan parametrni e’lon qiladigan (misol uchunparam) bo‘lsak shu pozitsiyadan boshlab oxirigacha bo‘lgan kalit argumentlar param nomli lug‘at(словарь)ga yig‘iladi. Bizning holatda keywords lug‘atida {’vegetables’: 50,’fruits’: 100} qiymat mavjud bo‘ladi.
Biz kortej va lug‘atlarni imkon qadar keyingi darslarimizda o‘rganamiz.

Faqat kalit argumentlar
Agar ba’zi kalit argumentlarga faqat kalit bo‘yicha murojaat qilish kerak bo‘lsa, u holda yulduzchali parametrdan so‘ng bu argumentni e’lon qilish mumkin.
Misol:
def total(initial=5, *numbers, extra_number):
count = initial
for number in numbers:
count += number
count += extra_number
print(count)
total(10, 1, 2, 3, extra_number=50)
total(10, 1, 2, 3)
# Xatolik yuz beradi sababi biz ’extra_number’ uchun qiymat bermadik.
Natija:
>>>66
Traceback (most recent call last):
File "keyword_only.py", line 12, in <module>
total(10, 1, 2, 3)
TypeError: total() needs keyword-only argument extra_number
Bu qanday ishlaydi:
Yulduzchali parametrdan keyin faqat kalitl parametrlar e’lon qilinadi. Agar bunday argumentlar uchun boshlang‘ich qiymat berilmagan bo‘lsa va funksiyani chaqirilish joyida qiymat berilmasa, xatolik yuz beradi.
Agar sizga faqat kalit argumentlar kerak bo‘lsa, lekin yulduzchali parametr kerak bo‘lmasa, u xolda shunchaki bitta nomsiz yulduzcha ko‘rsatishingiz mumkin:
def total(initial=5, *, number).
return operatori. return operatori funksiyani to‘xtatish va undan chiqish uchun ishlatiladi. Shu bilan birga funksiyadan biror qiymat qaytarish uchun ham xizmat qiladi.
Misol:
def maximum(x, y):
if x > y:
return x
elif x == y:

return ’Sonlar teng.’
else:
return y
print(maximum(2, 3))
Natija:
>>>3
Bu qanday ishlaydi:
maximum funksiyasi berilgan ikki parametrdan kattasini aniqlaydi. Bu funksiya oddiy if..elseshart operatorini sonlarning kattasini aniqlashda foydalanadi va aniqlangan katta sonni qaytaradi.
Shunga e’tibor beringki, return operatori qaytarish qiymatisiz return None ifodasiga teng kuchli hisoblanadi.

None Pythondagi xech narsani ifodalovchi maxsus ma’lumot turi hisoblanadi. Misol uchun o‘zgaruvchi qiymatiga None biriktirilgan bo‘lsa, unga xech qanday qiymat biriktirilmaganiligini bildiradi.
Agar siz funksiyada return operatorini ishlatmagan bo‘lsangiz, u holda har bir shunday funksiya tugash joyida oshkormas holda (в неявной форме) return None ifodasi mavjud bo‘ladi. Buni amalda ko‘rish uchun quyidagi dastur kodini ishga tushirib ko‘ring.
def someFunction():
pass
print(def someFunction())
pass operatori Pythonda bo‘sh buyruqlar blokini ifodalash uchun ishlatiladi.
Xujjatlash satrlari(Строки документации)
Python xujjatlash satrlari, qisqa ifodalaganda docstrings deb nomlanuvchi o‘ziga xosligi mavjud. Bu juda muhim instrument bo‘lib siz undan albatta foydalanishingiz kerak. Sababi bu sizning dasturingizni yaxshi xujjatlash(документировать) va oson tushunishga yordam beradi. Xujjatlash satrini dastur bajarilish jarayonida funksiyadan olish mumkin.
Misol:
def maximum(x, y):
"""Ikki sondan kattasini chop etadi.
Ikkala qiymat ham butun son bo‘lishi kerak."""
x = int(x) # agar iloji bo‘lsa, butun songa konvertatsiya qilamiz
y = int(y)
if x > y :

print(x, ’katta’)
else:
print(y, ’katta’)
maximum(3, 5)
print(maximum.doc)
Natija:
>>>5 katta
Ikki sondan kattasini chop etadi.
Ikkala qiymat ham butun son bo‘lishi kerak.
Bu qanday ishlaydi:
Birinchi mantiqiy qatordagi satr funksiya uchun docstring hisoblanadi.
Docstiring modul va klasslar bilan ham qo‘llaniladi.Funksiya docstiringini ko‘p qatorli satr ko‘rinishida yozish qabul qilingan. Bu satrning birinchi qatori bosh xarf bilan boshlanadi va nuqta bilan tugaydi. Ikkinchi qator bo‘sh qoldiriladi va funksiya haqidagi to‘liq ma’lumot uchunchi qatordan boshlab yoziladi.
Shunday uslubda docstring yozish tavsiya qilinadi.
maximum funksiyasining docstringiga shu funksiyaning __doc__ atributi orqali murojat qilish mumkin (Ikkita belgilash simvoliga e’tibor bering). Shuni e’tiborga olingki, python "hamma" narsani ob’yekt ko‘rinishida tasavvur qiladi.
Agar siz Pythonda help() funksiyasini ishlatgan bo‘lsangiz, demak, siz docstringni ko‘rgansiz. Bu funksiya tegishli funksiyaning doc atributini hisoblaydi va ifodali holda ekranga chop etadi. Buni yuqoridagi funksiya misolida ko‘rishingiz mumkin.
help(maximum) kodini dastur kodiga qo‘shing.

Biz funksiyalarning ko‘p qirralarini ko‘rib chiqdik, lekin bu hammasi emas. Ko‘rib chiqqanlarimiz pythonda funksiyalar bilan ishlashda duch kelishingiz mumkin bo‘lgan ko‘p hollarni qamrab olgan.

Nazorat savollari.

1. Qism dastur deganda nimani tushunasiz?
2. Funksiyalarni shakllantirish usullarini tushuntirib bering?
3. Funksiyalarni shakllantirishni umumiy ko‘rinishi?
4. Global va lokal o‘zgaruvchilar deganda nimani tushunasiz?
5. Funksiyaga argument berishni soddalashtirish jarayonini tushuntirib bering?

11-ma’ruza.Pythonda fayllar bilan ishlash.

Reja:

1. Fayllarni faollashtirish, Fayllar ustida amallar bajarish.
2. Fayldan ma’lumot o‘qish.
3. Fayl tarkibiga ma’lumotlarni yozish.
4. Fayl tarkibidagi ma’lumotlarni o‘chirish.

Pythonda turli xil fayl turlari bilan ishlash imkoniyati mavjud bo‘lib, shartli ravishda ularni ikki turga bo‘lish mumkin: matn va binar fayllar. Matn fayllari, masalan, kengaytmasi cvs, txt, html, umuman, matn shaklida ma’lumot saqlaydigan barcha fayllarlarni o‘z ichiga oladi. Binar fayllar tasvirlar, audio va video fayllar va boshqalardan iborat. Fayl turiga qarab u bilan ishlash biroz farq qilishi mumkin.

Fayllar bilan ishlaganda, quyidagi tartibdagi operatsiyalar ketma-ketligini amalga oshirish talab etiladi:

open() metodi yordamida faylni ochiladi;

read() metodi yordamida faylni o‘qish yoki write() metodi yordamida faylga yozish amalga oshiriladi;

close() metodi faylni yopadi.

1.1. Fayllarni ochish va yopish

Fayllar bilan ishlash uchun avval faylni *open*() metodi yordamida ochish zarur. *open*() metodidan quyidagi ko‘rinishda foydalaniladi:

open(file, mode)

Funksiyaning birinchi parametri faylning yo‘lini ifodalaydi. Fayl yo‘li absolyut bo‘lishi mumkin, ya’ni disk harfidan boshlanadi, masalan, C: //qandaydirpapka/somefile.txt. Yoki nisbiy bo‘lishi mumkin, masalan, qandaydirpapka/ somefile.txt — bu holda, fayl Python ishlaydigan skript joylashgan katologda hosil qilinadi. Ikkinchi argument *mode* — bu faylni ochish rejimi bo‘lib, fayl bilan qanday ish bajarilishiga qarab, 4 turdagi fayllar bilan ishlash rejimidan birini qo‘llash mumkin:

r (*Read*) — Fayl o‘qish uchun ochadi. Fayl topilmasa, *FileNotFoundError* xatolik qaytaradi;

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

a (*Append*). Faylni qayta yozish uchun fayl ochiladi. Agar fayl yo‘q bo‘lsa, u hosil bo‘ladi. Bunday fayl allaqachon mavjud bo‘lsa,

ma’lumotlar oxiridan yozish davom ettiriladi.
b (*Binary*). Binar fayllar bilan ishlash uchun foydalaniladi. *w* va r kabi rejimlar kombinatsiyasi bilan birgalikda ishlatiladi.

Fayl bilan ishlashni tugatgandan so‘ng uni *close()* metodi bilan yopish kerak bo‘ladi. Ushbu metod fayl bilan bog‘liq barcha resurslarni bo‘shatadi.

Misol uchun, "salom.txt" matnli faylini yozish uchun ochamiz:

```
meningfaylim = open("salom.txt", "w")
meningfaylim.close()
```

Faylni ochishda yoki u bilan ishlashda turli xil istisno holatlarga duch kelish mumkin, masalan, unga ruxsat yo‘q bo‘lishi mumkin. Bunday holatlarda, dastur ishlash jarayonida xatolik yuz beradi va dastur bajarilishi *close()* metodi chaqirilishiga yetib bormaydi va shunga muvofiq fayl yopilmaydi. Bu kabi holatlarni oldini olish uchun istisnolardan foydalaniladi:

```
try:
    somefile = open("salom.txt", "w")     try:
        somefile.write("Salom olam")     except Exception as e:
        print(e)     finally:
        somefile.close() except Exception as ex:
    print(ex)
```

Bu erda, fayl bilan bajariladigan barcha amallar ketma-ketligi *try* blokida yoziladi. Agar biror bir istisno to‘satdan kelib chiqsa, u holda *finally* blokida fayl blokirovka qilinadi.

Fayllar bilan ishlashning yanada qulayroq *with* konstruktsiyasi mavjud:

```
with open(file, mode) as file_obj:
    #buyruqlar
```

Bu konstruktsiya ochiq fayl uchun *file_obj* o‘zgaruvchi aniqlanadi va buyruqlar ketma-ketligi bajariladi. Ular bajarilgandan so‘ng, fayl avtomatik ravishda yopiladi. Blokda amallar ketma-ketligini bajarishda istisnolar yuzaga kelsa ham, fayl avtomatik ravishda yopiladi.

with konstruktsiyasi yordamida, yuqoridagi misolni quyidagicha qayta yozish mumkin:

```
with open("salom.txt", "w") as somefile:
    somefile.write("Salom Python")
```

1.2. Matn fayllari. Matn faylga yozish

Matn faylini yozish uchun ochishda *w* (qayta yozish) yoki *a* (yozuv qo‘shish) rejimini qo‘llaniladi. So‘ngra, faylga yozish uchun *write(str)* metodidan foydalanilib, *str* parametriga yozilishi kerak bo‘lgan satr uzatiladi. Shuni eslatib o‘tish joizki, bu parametr satr bo‘lishi shart, shuning uchun raqamlar yoki boshqa turdagi ma’lumotlarni yozish zarur bo‘lsa, dastlab ularni satr turiga keltirish talab qilinadi.
"salom.txt" fayliga ba’zi ma’lumotlarni yozamiz:

```
1  with open("salom.txt", "w") as fayl:
2      fayl.write("salom olam")
```

Joriy Python skriptining joylashgan papkasini ochsak, u yerda salom.txt faylini ko‘rish mumkin. Ushbu fayl har qanday matn muharriridan ochilishi mumkin va agar kerak bo‘lsa o‘zgartirilishi ham mumkin. Keling, ushbu faylga yana bitta qator qo‘shamiz:

```
1  with open("salom.txt", "a") as fayl:
2      fayl.write("\nHayr, olam")
```

Agar satrni yangi qatorga yozish zarur bo‘lsa, u holda, yuqoridagi kabi, yozilayotgan satr oldidan "\n"(yangi satrga o‘tish belgisi) qo‘yish yetarli bo‘ladi.
Yukunida salom.txt faylida quyidagi matn hosil bo‘ladi: *salom olam*
Hayr, olam
Faylga yozishning yana bir usuli — ma’lumotlarni konsolga chiqarish uchun ishlatiladigan standart *print()* metodan foydalanish orqali amalga oshiriladi:

```
1  with open("salom.txt", "a") as salom_fayl:    print("Salom, olam",
2  file=salom_fayl)
```

print metodi yordamida ma’lumotlarni faylga chiqarish uchun faylning nomi *file* parametri orqali beriladi va birinchi parametr faylga yoziladigan ma’lumotni ifodalaydi.
2. Fayldan o‘qish
Fayldan o‘qish uchun *r* (*Read*) rejimida ochiladi va uning mazmunini turli usullar bilan o‘qish mumkin:
readline() — fayldan bir qator o‘qiydi;
read() — faylning butun tarkibini bir qatorga o‘qiydi;
readlines() - faylning barcha satrlarini ro‘yxatga oladi.

Masalan, biz yuqorida yozilgan fayllarni satrlar bo‘yicha ko‘rib chiqamiz:

```
1  with open("salom.txt", "r") as fayl:     for satr in fayl:
2          print(satr, end="")
3
```

Biz, albatta, har bir qatorni o‘qish uchun *readline()* metidini ishlatmasak ham, har bir yangi satrni olish uchun ushbu metod avtomatik ravishda chaqiriladi. Shuning uchun ham, *readline()* metodini siklda chaqirishdan ma’no yo‘q va satrlar yangi satr "\n" belgisi bilan ajratilganligi uchun, yangi satrga chop qilish zaruryati qolmaydi va *end=""* qiymati *print* metodining ikkinchi parametri sifatida uzatiladi.

Endi satrlarni alohida o‘qish uchun *readline()* metodini to‘g‘iridan-to‘g‘ri chaqiramiz:

```
1  with open("salom.txt", "r") as fayl:
2     str1 = fayl.readline()     print(str1, end="")
3
```

```
4     str2 = fayl.readline()     print(str2)
5
```

Konsol ektaniga quyudagi natijalar chiqariladi:
salom olam hayr, olam *readline()* metodini alohida qatordagi satrlarni o‘qish uchun *while* siklida ham foydalanish mumkin:

```
1  with open("salom.txt", "r") as fayl:
2     satr = fayl.readline()     while satr:
3        print(satr, end="")        satr = fayl.readline()
4
5
```

Fayl kichik bo‘lsa, *read*() metodidan foydalanib, uni birdan o‘qishingiz mumkin:

```
1  with open("salom.txt", "r") as fayl:
2     mazmun = fayl.read()     print(mazmun)
3
```

Hamda, *readlines()* metodi yordamida fayldagi barcha satrlar

ro‘yxatga o‘qib olinadi, ya’ni elementlari fayldagi satrlardan tashkil topgan ro‘yxat hosil qilinadi:

```
1  with open("salom.txt", "r") as faly:
2      mazmun = fayl.readlines()       str1 = mazmun [0]       str2 =
3  mazmun [1]     print(str1, end="")     print(str2)
4
5
6
```

Ba’zida fayldagi ma’lumotlar ASCIIdagi belgilardan farqlanishi mumkin. Ushbu holatda fayldan berilganlarni o‘qish to‘g’ri bo‘lishi uchun kodlash parametrini ishlatib kodlashni aniq belgilab olishimiz mumkin:

```
1  faylnomi = "salom.txt" with open(faylnomi, encoding="utf8") as
2  file:
3      matn = file.read()
```

Quyidagi dastur orqali foydalanuvchi tomonidan kiritilgan satrlar massivi dastlab faylga yozish amalga oshirilgan, so‘ngra ularni fayldan konsolga qayta o‘qib, chop qilish amalga oshirilgan:

```
1    # fayl nomi
2    FILENAME = "habarlar.txt" # bo‘sh ro‘yxat aniqlaymiz xabarlar =
3 4  list()
5     for i in range(4):
6        xabar = input("Satrni kiriting " + str(i + 1) + ": ")
7    xabarlar.append(xabar + "\n")
8
9    # ro‘yxatni faylga yozish with open(FILENAME, "a") as fayl:    for
10   xabar in xabarlars:         fayl.write(xabar)
11
12   # xabarlarni fayldan o‘qiymiz  print("Xabarlarni o‘qish") with
13   open(FILENAME, "r") as fayl:     for xabar in fayl:
14         print(xabar, end="")
15
16
17
18
19
```

Dastur ishlashining namunasi:

```
1  Satrni kiriting 1: salom
2  Satrni kiriting 2: tinchlik so‘zi
3  Satrni kiriting 3: buyuk ish
4  Satrni kiriting 4: Python
5  Xabarlarni o‘qish Salom tinchlik so‘zi buyuk ish
6  Python
7
8
9
```

Ma’lumotni qulay shaklda saqlashning keng tarqalgan fayl formatlaridan biri csv formatidir. CSV faylidagi har bir satr vergul bilan ajratilgan alohida ustunlardan iborat bo‘lgan yozuv yoki satrni aks ettiradi. Aslida, bu format "Vergul bilan ajratilgan qiymatlar (Comma Separated Values)" deb nomlanadi. CSV formati matnli fayl

formati bo‘lsa-da, Python u bilan ishlashni soddalashtirish uchun maxsus ajralmas CSV modulini taqdim etadi. Quyidagi misolda modulning ishini ko‘rib chiqamiz:

```
import csv
FILENAME = "users.csv"
 users = [
   ["Ali", 25],
   ["Sobir", 32],
   ["Dilnoza", 14]
]  with open(FILENAME, "w", newline="") as fayl:
   writer = csv.writer(fayl)     writer.writerows(users)
 with open(FILENAME, "a", newline="") as fayl:
   user = ["Shaxnoza", 18]              writer = csv.writer(fayl)
writer.writerow(user)
```

Faylga ikki o‘lchovli ro‘yxat yoziladi – har bir satr bitta foydalanuvchini ifodalaydigan jadval. Har bir foydalanuvchi esa ikkita maydon — ism va yoshni o‘z ichiga oladi. Ya’ni, uchta satr va ikki ustunli jadvalni ifodalaydi.

Yozish uchun fayl ochilganda, uchinchi parametr sifatida *newline=""* qiymati ko‘satildi — bo‘sh satr operatsion tizimidan qat’i nazar, fayllardan to‘g‘ri satrlarni o‘qishga imkon beradi.

Yozish uchun *csv.writer(file)* funktsiyasi tomonidan qaytariladigan *writer* obyektini olishimiz kerak. Ushbu funktsiyaga ochiq fayl topshiriladi. Hamda, mos ravishda yozish *writer.writerows(users)* metodi yordamida amalga oshiriladi. Bu usul qatorlar to‘plamini parametr sifatida oladi. Bizning holatimizda bu ikki o‘lchovli ro‘yxat hisoblanadi.

Agar bitta yozuv qo‘shish zarur bo‘lsa, ya’ni, bir o‘lchamli ro‘yxat, masalan, ["Shaxnoza", 18], bu holda *writer.writerow(user)* metodidan foydalaniladi. Natijada, skriptni ishga tushirgandan so‘ng, quyidagi tarkibga ega bo‘lgan *users.csv* fayli shu papkada paydo bo‘ladi:

```
Ali,25
Sobir,32
Dilnoza,14
Shaxnoza,18
```

Fayldan o‘qish uchun, aksincha, *reader* obyektini yaratishimiz kerak:

```
import csv
FILENAME = "users.csv"
 with open(FILENAME, "r", newline="") as fayl:
   reader = csv.reader(fayl)     for row in reader:
       print(row[0], " — ", row[1])
```

reader obyektini olayotganda, biz uning barcha satrlarini ko‘chirib olishimiz mumkin:

```
Ali — 25
Sobir — 32
Dilnoza — 14
Shaxnoza — 18
```

Yuqoridagi misolda har bir yozuv yoki satr alohida ro‘yxat bo‘lib, masalan, ["Shaxnoza", 18]. Bundan tashqari, CSV modullari lug‘atlar bilan ishlash uchun maxsus qo‘shimcha xususiyatlarga ega. Xususan, *csv.DictWriter()* funktsiyasi faylga yozish imkonini beruvchi *writer* obyektni qaytaradi va *csv.DictReader()* funktsiyasi esa fayldan o‘qish uchun *reader* obyektini qaytaradi.

Masalan:

Qatorlarni *writerow()* va *writerows()* metodlari yordamida ham yozish mumkin. Ammo endi har bir satr alohida lug‘at, hamda, ustun sarlavhalari *writeheader()* metodidan foydalanib yoziladi va ikkinchi parametr sifatida *csv.DictWriter* metodiga ustunlar to‘plami uzatiladi.

Ustunlardan foydalangan holda satrdan berilganlarni o‘qishda satr ichidagi alohida qiymatlarga *row["ism"]* ko‘rinishida murojaat qilishimiz mumkin.

Binar fayllarda, matnli fayllardan farqli o‘laroq, ma’lumotlar

baytlar majmui sifatida saqlanadi. Pythonda ular bilan ishlash uchun *pickle* – ichki modul kerak bo‘ladi. Ushbu modul ikkita metodni taqdim etadi:

dump(obj, file) — obyektni ikkilik faylga yozadi;

load(file) — binar fayldan obyektga ma’lumotlarni o‘qiydi.

O‘qish yoki yozish uchun binar faylni ochishda yozishni ("w") yoki o‘qish ("r") rejimidan tashqari "b" rejimidan foydalanish kerakligini ham hisobga olish kerak. Masalan, ikkita obyektni saqlash kerak bo‘lsin:

```
import csv
FILENAME = "users.csv"
 users = [
    {"ism": "Ali", "yosh": 25},
    {"ism": "Sobir", "yosh": 32},
    {"ism": "Dilnoza", "yosh": 14}
]  with open(FILENAME, "w", newline="") as fayl:
    ustunlar = ["ism", "yosh"]
    writer       =       csv.DictWriter(fayl,       fieldnames=ustunlar)
writer.writeheader()

    # bir qancha qatorlarni yozish     writer.writerows(users)

    user = {"ism": "Shaxnoza", "yosh": 18}
    # bitta qatorni yozish     writer.writerow(user)
 with open(FILENAME, "r", newline="") as fayl:
    reader = csv.DictReader(fayl)     for row in reader:
        print(row["ism"], "-", row["yosh"])
```

```
import pickle
FILENAME  =  "user.dat"  ism  =  "ali"  yosh  =  25     with
open(FILENAME, "wb") as file:
    pickle.dump(ism, file)     pickle.dump(yosh, file)
 with open(FILENAME, "rb") as file:
    ism = pickle.load(file)     yosh = pickle.load(file)
    print("Ism: ", ism, "\tYosh: ", yosh)
```

dump() funksiyasi bilan ikki obyekt ketma-ket yoziladi. Shu sababli, fayldan ketma-ketlikda *load()* funksiyasi yordamida o‘qiganimizda mos ravishda yozilgan obyektlarni olamiz. Dastur ishlaganda konsol ekraniga quyidagilar chiqariladi:

```
1  Ism: Ali Yosh: 25
```

Shu tarzda, faylga obyektlar to‘plamini saqlashimiz va undan o‘qib olishimiz mumkin:

```
import pickle
FILENAME = "users.dat"
 users = [
   ["Ali", 25, True],
   ["Sobir", 32, False],
   ["Dilnoza", 14, False]
]  with open(FILENAME, "wb") as file:
   pickle.dump(users, file)
 with open(FILENAME, "rb") as file:
   users_from_file = pickle.load(file)
for user in users_from_file:
       print("Ism: ", user[0], "\tYosh: ", user[1],
"\tUylangan(turmushga chiqan): ", user[2])
```

Faylga *dump()* funktsiyasidan foydalanib qanday obyekt yozilsa, fayldan *load()* funksiyasi orqali xuddi shu obyekt o‘qiladi.
Natijaning konsolga chiqarilishi:

```
1  Ism: Ali Yosh: 25 Uylangan(turmushga chiqan): True
2  Ism: Sobir Yosh: 32 Uylangan(turmushga chiqan): False
3  Ism: Dilnoza Yosh: 14 Uylangan(turmushga chiqan): False
```

Binar fayllar bilan ishlash uchun Pythonda yana bitta modul — shelve modulidan foydalanish mumkin. Obyektlarni maxsus kalitga ega faylga saqlaydi.

Keyinchalik, bu kalit yordamida avvaldan saqlangan obyektni fayldan oqib olinadi. *shelve* modul orqali ma’lumotlarni ishlash jarayoni lug‘atlarga o‘xshash, shuning uchun obyektlarni saqlash va o‘qib olish uchun kalitlardan foydalaniladi.

Faylni ochish uchun *shelve* moduli *open()* funktsiyasidan foydalanadi:

```
1  open(fayl_manzili[, flag="c"[, protocol=None[, writeback=False]]])
```

flag parametri quyidagi qiymatlarni qabul qilishi mumkin:
c — faylni o‘qish va yozish uchun ochadi (*flag=c*, *flag* parametrining kelishuv bo‘yicha qiymati aynan *c* ga teng). Agar fayl mavjud bo‘lmasa, u yaratiladi;
r — fayl faqat o‘qish uchun ochiladi;
w — fayl faqat yozish uchun ochiladi;
n — fayl yozish uchun ochilgan. Agar fayl mavjud bo‘lmasa, fayl yaratiladi.
Agar mavjud bo‘lsa, u holda fayl qayta yozish uchun ochiladi.
Faylga ulanishni yopish uchun *close*() metodidan foydalaniladi:

```
1  import shelve
2  d = shelve.open(faylnomi)
3  d.close()
```

Yoki faylni *with* operatoridan foydalanib ochishingiz mumkin. Quyidagi dasturda faylga bir nechta obyektni saqlash va keyin undan o‘qib olish ko‘rsatilgan:

```
import shelve
FILENAME = "davlatlar" with shelve.open(FILENAME) as states:
states["Toshkent"] = "O\’zbekiston" states["Moskva"] =
"Rossiya" states["Pekin"] = "Xitoy" states["Seul"] = "Korea"
 with shelve.open(FILENAME) as states:
   print(states["Toshkent"]) print(states["Pekin"])
```

Berilganlarni faylga yozish – bu ma’lum bir kalitga qiymatni yuklash tarzida amalga oshiriladi:

```
1  states["Toshkent"] = "O\’beksiton"
```

Fayldan ma’lumotlarni o‘qib olish kalit bo‘yicha qiymatni olishga ekvivalent tarzda amalga oshiriladi:

```
1  print(states["Toshkent"])
```

Kalit sifatida satrlar ishlatiladi.

Ma’lumotni o‘qiyotganda so‘ralgan kalit mavjud bo‘lmasa, u holda istisno holati yuzaga keladi. Shu kabi vaziyatlarda, odatda, kalitni mavjud yoki yo‘qligini tekshirish tavsiya etiladi va buning uchun *in* operatoridan foydalaniladi, masalan:

```
1  with shelve.open(FILENAME) as states:
2     key = "Ostana"     if key in states:
3        print(states[key])
4
```

Fayldan ma’lumotlarni o‘qib olish uchun *get()* metodidan ham foydalanish mumkin. Ushbu metodning birinchi parametri – kalit bo‘lib, u orqali qiymat olinadi, ikkinchi parametr esa, birinchi parametrda keltirilgan kalit topilmasa qaytariladigan kelishuv bo‘yicha qiymatni ifodalaydi.

```
1  with shelve.open(FILENAME) as states:
2     state = states.get("Ostana", "Aniqlanmagan")     print(state)
3
```

for operatoridan foydalanib, fayldagi barcha qiymatlarni ketma-ket ko‘rib chiqish mumkin:

```
1  with shelve.open(FILENAME) as states:     for key in states:
2        print(key," — ", states[key])
3
```

keys() metodi fayldan natija sifatida barcha kalitlarni, *values()* metodi esa barcha qiymatlarni qaytaradi:

```
with shelve.open(FILENAME) as states:
   for city in states.keys():
      print(city, end=" ")  # Toshkent Moskva Pekin Seul      print()
for country in states.values():
      print(country, end=" ")  # O‘zbekiston Rossiya
Xitoy Korea
```

Yana bir metod — *items()* metodi har bir elementi kalit va qiymat

juftligini tashkil qiluvchi kortejlardan iborat kortejlar to‘plamini qaytaradi:

```
1  with shelve.open(FILENAME) as states:          for state in
2  states.items():
3        print(state)
```

Natija:

```
1  ("Toshkent", "O‘zbekiston")
2  ("Moskva", "Rossiya")
3  ("Pekin", "Xitoy")
4  ("Seul", "Korea")
```

Ma’lumotlarni o‘zgartirish uchun kalitga yangi qiymat o‘zlashtirish va yangi berilganlarni qo‘shish uchun yangi kalit bo‘yicha qiymat berish kifoya:

```
import shelve
FILENAME = "states2" with shelve.open(FILENAME) as states:
states["Tashkent"] = "O‘zbekiston"      states["Moskva"] = "Rossiya"
states["Pekin"] = "Xitoy"     states["Seul"] = "Korea"
 with shelve.open(FILENAME) as states:        states["Tashkent"] =
"O‘zbekston Respublikasi"     states["Ostona"] = "Kozog‘iston"     for
key in states:
      print(key, " — ", states[key])
```

Faydan elementni o‘chirish va qiymat sifatida o‘chirilgan elementni qaytarish uchun *pop*() metodidan foydalanilib, unga birinchi parametr sifatida o‘chirilishi kerak bo‘lgan element kaliti hamda ikkinchi parametriga faylda birinchi parametrda ko‘rsatilgan kalit mavjud bo‘lmasa natija sifatida qaytariladigan kelishuv bo‘yicha qiymat beriladi:

```
with shelve.open(FILENAME) as states:
    state = states.pop("Buxoro", "Topilmadi")     print(state)

```

del operatorini ham o‘chirish uchun ishlatilishi mumkin:

```
with shelve.open(FILENAME) as states:
    del states["Pekin"]  # Pekin kalit bilan obyektni o‘chirish

```

clear() metodi yordamida hamma elementlarni o‘chirish mumkin:

```
with shelve.open(FILENAME) as states:
    states.clear()
```

Kataloglar va fayllar bilan ishlash uchun bir qancha imkoniyatlar ichki *os* moduli tomonidan ta’minlanadi. Ushbu modul ko‘p funktsiyani o‘z ichiga olgan bo‘lsada, faqat asosiylarini ko‘rib chiqamiz:
mkdir() – yangi papka yaratadi;
rmdir() – papkani o‘chiradi;
rename() – fayl nomini o‘zgartiradi; remove() – faylni o‘chiradi.
6. Papkalarni yaratish va o‘chirish
Papka yaratish uchun *mkdir()* funksiyasidan foydalaniladi, unga yaratilayotgan papkaga yo‘li beriladi:

```
import os
#Skirpt joylashgan adresga nisbiy yo‘l os.mkdir("hello") # absolyut adres os.mkdir("c://adizova")
os.mkdir("c://adizova/hello")
```

Papkani o‘chirish uchun *rmdir()* funksiyasidan foydalanilib, o‘chirilishi kerak bo‘lgan papakanining yo‘li beriladi:

```
import os
#Skirpt joylashgan adresga nisbiy yo‘l os.rmdir("hello") # absolyut adres
os.rmdir("c://somedir/hello")
```

Fayllarni qayta nomlash uchun *rename(source, target)* funksiyasidan

ishlatiladi, birinchi parametrda manba fayl yo‘li, ikkinchisida esa yangi fayl nomi beriladi. Fayl joylashgan yo‘lini berishda ham mutlaq va ham nisbiy yo‘llar berilishi mumkin. Misol uchun, “*somefile.txt*” fayli “*C://SomeDir/*” papkasida joylashganligini tasavvur qiling. Uni "*hello.txt*" fayliga o‘zgartirish:

```
1  import os
2  os.rename("C://SomeDir/somefile.txt",
3  "C://SomeDir/hello.txt")
4
```

Biror faylni o‘chirish uchun *remove()* funktsiyasida foydalaniladi. Ushbu funksiyaga parametr sifatida o‘chirilishi lozim bo‘lgan fayl yo‘li beriladi, masalan:

```
1  import os
2  os.remove("C://SomeDir/hello.txt")
3
```

Agar mavjud bo‘lmagan faylni ochishga harakat qilinsa, u holda Python
FileNotFoundError istisno holati ro‘y berganligi to‘g’risida xatolik xabarini qaytaradi. Bunday istisno holatini uchun *try...except* konstruksiyasidan foydalanishimiz mumkin. Biroq, faylni ochishdan avval *os.path.exists(path)* metodi yordamida fayl mavjudligini tekshirish lozim. Tekshirilishi kerak bo‘lgan yo‘l bu metodga uzatiladi:

```
filename = input ("Faylning yo‘lini kiriting:")
if os.path.exists(filename):
    print("ko‘rsatilgan fayl mavjud")
else:
    print("Fayl mavjud emas")
```

Nazorat savollari.

1. Fayl deb nimaga aytiladi?
2. Dasturlashda qanday kengaytmali fayllardan foydalaniladi?
3. Dasturlashda fayl nomi qanday qismlardan tashkil topgan?
4. Fayllarni fizik va mantiqiy nomlarini bog‘lashning umumiy ko‘rinishi?
5. Fayllarni o‘qish uchun faollashtirishning umumiy ko‘rinishini?

12-ma’ruza.Istisnolar (Exceptions).

Reja:

1. Python dasturlash tilida xatoliklar bilan ishlash.
2. Xatoliklar turlari.
3. Ma’lum turdagi xatolarni ushlash

Ba’zan dastur ishlashi davomida istisno holatlar yuzaga kelishi mumkin. Misol uchun, mavjud bo‘lmagan faylni o‘qishga bo‘lgan harakat yoki dasturdagi no‘malum buyruqlar. Bunday holatlar exception’lar yordamida hal qilinadi.

Agar biz print funksiyasini Print shaklida chaqiradigan bo‘lsak, python interpretatori bizga sintaksis xatolikni ko‘rsatadi.

```
>>> Print("Salom dunyo")
Traceback (most recent call last):
File "<stdin>", line 1, in <module>
NameError: name ’Print’ is not defined
>>> print("Salom dunyo")
Salom dunyo
```

E’tibor bergan bo‘lsangiz, NameError xatoligi chiqarildi hamda qayerda shu xatolik qayd qilingani ham ko‘rsatildi. Bu holda xatoliklarni qayta ishlovchisi harakatga tushadi.

Exceptionlar

Foydalanuvchidan nimadir kiritishini so‘raymiz, so‘ngra Ctrl + d tugmalarini bosamiz va nima bo‘lishini kuzatamiz.

```
>>> s = input("Nimadir kiriting –> ")
Nimadir kiriting –> Traceback (most recent call last):
File "<stdin>", line 1, in <module>
EOFError
```

Python EOFError nomli xatolikni chiqardi. Bu xatolik kutilmagan joyda (Ctrl + dtugmalari yordamida kiritiladigan) fayl oxiri (end of file) belgisi qayd qilinganini bildiradi.

Exception – istisnolarni qayta ishlash

Exception’larni try . . except operatori yordamida qayta ishlash mumkin. Bunda hamma odatiy buyruqlar try blokiga joylashtiriladi, istisnolarni qayta ishlovchilari esa except blokiga joylashtiriladi.

Misol: (try_except.py nomi bilan saqlang.)

```
try:
    text = input(’Nimadir kiriting --> ’)
```

```
except EOFError:
    print('Nega EOF qildigiz?')
except KeyboardInterrupt:
    print('Siz amallarni bekor qildingiz.')
else:
    print('Siz {0} kiritdingiz.'.format(text))
```

Natija:

```
$ python try_except.py
Nimadir kiriting → # Ctrl + d ni bosing
Nega EOF qildingiz?
$ python try_except.py
Nimadir kiriting → # Ctrl + c ni bosing
Siz amallarni bekor qildingiz.
$ python try_except.py
Nimadir kiriting → xatosiz
Siz xatosiz kiritdingiz.
```

Bu qanday ishlaydi:

Bu yerda biz istisno/xatolik chaqirishi mumkin bo‘lgan barcha buyruqlarni trybloki ichiga joylashtirdik. So‘ng except bloki ichiga tegishli istisno/xatolikka mos keluvchi qayta ishlovchilarni joylashtirdik. except ifodasi bitta istisno/xatolikni yoki qavs ichida ko‘rsatilgan bir nechta istisno/xatoliklarni qayta ishlashi mumkin.

Agar xatolik yoki istisno nomi ko‘rsatilmagan bo‘lsa, u holda barcha xatolik va istisnolar qayta ishlanadi.

Yodingizda saqlang, try ifodasi uchun hech bo‘lmaganda bitta except ifodasi bo‘lishi kerak. Aks holda try ishlatishning hech qanday ma’nosi qolmaydi.

Agar xatolik yoki istisno biror except blokida ushlab qolinmasa, u holda pythonning istisnolarni qayta ishlovchisi chaqiriladi va oynaga xatolik haqidagi xabarni chop etadi. Yuqorida bunga misol ko‘rdik.

try . . except blokiga else blokini ham qo‘shish mumkin. Bu blok hech qanday istisno yuz bermaganda ishga tushadi.

Exception – istino chaqirish

Istisnolarni raise operatori yordamida tegishli xatolik/istisno nomini bergan holda chaqirish mumkin.

Misol: (raising.py nomi bilan saqlang)

```
class ShortInputException(Exception):
```

```
    '''Foydalanuvchi klass exceptioni.'''
    def __init__(self, length, atleast):
        Exception.__init__(self)
        self.length = length
        self.atleast = atleast
try:
    text = input("Nimadir kiriting --> ")
    if len(text) < 3:
        raise ShortInputException(len(text), 3)
    # Bu yerda kerakli amallar bajarilishi mumkin.
except EOFError:
    print("Nega EOF qildingiz?")
except ShortInputException as ex:
    print("ShortInputException: Kiritilgan satr uzunligi -- {0}; minimum {1} kutilgan edi".format(ex.length, ex.atleast))
else:
    print("Exception bo‘lmadi.")
```

Natija:

```
$ python raising.py
Nimadir kiriting –> a
ShortInputException: Kiritilgan satr uzunligi — 1; minimum 3 uzunlik kutilgan edi
```

Bu qanday ishlaydi:

Bu yerda biz ShortInputException nomli o‘zimizning exception turimizni xosil qildik. U ikkita xususiyatdan (maydon) tashkil topgan: length — kiritilgan satr uzunligini saqlash uchun, atleast — dastur kutgan minimal satr uzunligi.

except blokida biz ShortInputException klasini ko‘rsatamiz u o‘z navbatidaex o‘zgaruvchisiga saqlanadi. ex o‘zgaruvchisi esa tegishlik xatolik/exception ob’yektidan tashkil topgan. except bloki ichida length va atleastxususiyatlarini ishlatib foydalanuvchiga tegishli xabarni ko‘rsatamiz.

try . . finally

Tasavvur qilaylik, dasturda fayldan o‘qib olish amali bor va dastur oxirida shu fayl to‘g‘ri yopilishi kerak. Bunga finally blokini qo‘llash bilan erishish mumkin.

Misol: (finally.py nomi bilan saqlang)

```
import time
try:
    f = open('poem.txt')

    while True: # Odatiy fayldan o‘qish usuli

        line = f.readline()

        if len(line) == 0:

            break

        print(line, end='')

        time.sleep(2) # Bir qancha vaqt kutish

except KeyboardInterrupt:

    print("!! Siz fayldan o‘qishni bekor qildingiz.")

finally:

    f.close()

    print('(Tozalash: Faylni yopish)')
```

Natija:

```
$ python finally.py
Dasturlash qiziqarli.
Agar ish zerikarli bo‘lsa,
Unga quvnoq tus berish uchun –
^C!! Siz fayldan o‘qishni bekor qildingiz.
(Tozalash: Faylni yopish)
```

Bu qanday ishlaydi:

Bu yerda biz oddiy fayldan o‘qib olish amallarini bajaryapmiz. Dastur sekin ishlashi uchun har bir chop qilingan satrdan so‘ng dastur 2 sekund uyquga ketadi (python judayam tez ishlaydi). Dasturni to‘xtatish yoki bekor qilish uchun dastur bajarilishi jarayonida Ctrl +

c bosing.

Kuzatgan bo‘lsangiz KeyboardInterrupt istisnosi yuz berdi va dastur bajarilishdan to‘xtadi. Ammo dastur bajarilishdan to‘xtashidan oldin finally bloki bajarildi va o‘qish uchun ochilgan fayl yopildi.

with operatori

Biror bir resursga try blokida murojaat qilib, so‘ngra bu resursni finally blokida bo‘shatib yuborish odatiy yo‘l sifatida ko‘riladi. Lekin bu amalni nisbatan qulayroq bajarish uchun with operatoridan foydalanish mumkin.

Misol: (using_with.py nomi bilan saqlang)

```
with open("poem.txt") as f:
    for line in f:
        print(line, end='')
```

Natija:

```
$ python using_with.py
Dasturlash qiziqarli.
Agar ish zerikarli bo‘lsa,
Unga quvnoq tus berish uchun –
Pythonni ishlating!
```

Bu qanday ishlaydi:

Natija bundan oldingi misoldagi kabi bo‘lishi kerak. Farqi shundaki, biz bu yerda open funksiyasini with operatori bilan ishlatyapmiz va shu bilan faylni avtomat yopishni with operatori zimmasiga yuklayapmiz.

poem.txt fayli teksti.

```
Dasturlash qiziqarli.
Agar ish zerikarli bo‘lsa,
Unga quvnoq tus berish uchun –
Pythonni ishlating!
```

except kalit so‘zini ishlatganimizda Python istalgan turdagi istisno holat uchun amal qiladi. Ammo biz qaysi turdagi istisno holatini tekshirishni o‘zimiz belgilashimiz mumkin. Bunday holatlarning turlari ko‘p, biz esa asosiylarini sanab o‘tamiz:

- NameError – murojaat qilinayotgan obyekt topilmasa, ishga tushadi.
- ValueError – o‘zgaruvchining qiymati unga mos bo‘lmagan turda bo‘lsa ishga tushadi. Masalan, biror harfli qiymatni son deb qabul qilmoqchi bo‘lsak, shunday bo‘ladi.

❖ TypeError - o‘zaro nomutanosib qiymatlar bilan amallar bajarilsa ishga tushadi. Masalan harfga son qo‘shmoqchi bo‘lganimizda.

❖ ZeroDivisionError – Istalgan sonni nolga bo‘lish holati bo‘lganda ishga tushadi.

Yuqorida "Run time error" xatoliklari bilan tanishgan edik. Bunday xatolar dastur bajarish jarayonida kelib chiqadi va dasturning ishlashini to‘xtatadi. Sintaks xatolikdan farqli ravishda Python bunday xatolarni dasturni bajarishdan avval aniqlay olmaydi.

Ushbu darsimizda qanday qilib mana shunday xatoliklarni jilovlashni o‘rganamiz. Maqsadimiz xatolik yuz berganda dastur to‘xtab qolishining oldini olish. Gap shundaki, dastur davomida xato yuz berganda Python maxsus exception (istisno) obyektini yaratadi. Agar bu obyekt "tutib" olinmasa, dastur bajarilishdan to‘xtaydi.

try-except

Istisno obyektlarni tutib olish uchun Pythonda maxsus try-except operatorlari bor. Bu operatorlar quyidagicha ishlaydi, try operatori badanida bajarish kerak bo‘lgan kod yoziladi, except operatori badanida esa xatolik yuz berganda bajarilishi kerak bo‘lgan kod yoziladi. Ya’ni dasturimiz to‘xtab qolmasdan bajarilaveradi.

Tushunarli bo‘lishi uchun quyidagi misolni ko‘ramiz.

```
yosh = input("Yoshingizni kiriting: ")
yosh = int(yosh)
print(f"Siz {2021-yosh} yilda tug‘ilgansiz")
```

Yuqoridagi misolning 1-qatorida biz foydalanuvchidan yoshini kiritishni so‘rayabmiz. Navbatdagi qatorda esa foydalanuvchi kiritgan qiymatni int() yordamida butun songa o‘tkazayapmiz. Agar foydalanuvchi yoshini kiritganda, butun emas, o‘nlik son kiritsa bu ValueError xatoligiga olib keladi, va dastur bajarilishdan to‘xtaydi.

Keling, yuqoridagi kodni try-except yordamida yozamiz:

```
yosh = input("Yoshingizni kiriting: ")
try:
    yosh = int(yosh)
    print(f"Siz {2021-yosh} yilda tug‘ilgansiz")
except:
    print("Butun son kiritmadingiz")
print("Dastur Tugadi!")
```

.

Bu yerda ham dastavval foydalanuvchi yoshini so‘radik. int() finksiyasini esa try badani ichida yozdik, agar foydalanuvchi to‘gri qiymat kiritgan bo‘lsa kodimiz foydalanuvchi tug‘ilgan yilini hisoblab ko‘rsatadi, exception (istisno) yuz berganda esa "Butun son kiritmadingiz" xabarini konsolga chiqaradi. Lekin dastur bajarilishdan to‘xtamaydi, va try-except blokidan keyingi qatorlar ham bajarilaveradi (print("Dastur Tugadi!")). Buni quyidagi natijadan ham ko‘rishimiz mumkin:

try-except operatorining afzalliklaridan biri, foydalanuvchiga tushunarsiz xatolar o‘rniga, o‘zimiz istagan, tushunarliroq matnni ko‘rsatishimiz mumkin. Shuningdek, kompleks tizimlarda arzimagan xatoni deb dasturimiz to‘xtab qolmaydi.

try-except-else

Yuqoridagi kodimizda biz try moduli ichida xato qaytarishi mumkin bo‘lgan ifodani ham (tyil = int(tyil)), xato qaytmaganda bajarilishi kerak bo‘lgan ifodani ham (print(f"Siz {2021-tyil} yoshdasiz")) birdan yozib ketayapmiz. Aslida, bunday qilishimiz to‘g‘ri emas.

To‘g‘ri usuli, bu avval xatoga tekshirish va xato yuz bermaganda bajariladigan ifodani alohida, else blokida yozish:

```
yosh = input("Yoshingizni kiriting: ")
try:
   yosh = int(yosh)
except:
   print("Butun son kiritmadingiz")
else:
   print(f"Siz {2021-yosh} yilda tug‘ilgansiz")
```

Albatta, yuqoridagi usul har doim ham qo‘l kelavermaydi.

Ma’lum turdagi xatolarni ushlash. Xatolarning turlari ko‘p, except operatori yordamida esa biz aynan qaysi xatolarni ushlamoqchi ekanimizni ham ko‘rsatishimiz mumkin. Misol uchun yuqoridagi misolda int() funksiyasi ValueError xatosini qaytardi. Agar biz faqatgina shu turdagi xatolarni ushlamoqchi bo‘lsak, kodimizni quyidagicha o‘zgartiramiz:

```
yosh = input("Yoshingizni kiriting: ")
try:
   yosh = int(yosh)
except ValueError:
```

```
    print("Butun son kiritmadingiz")
else:
    print(f"Siz {2021-yosh} yilda tug‘ilgansiz")
```

ZeroDivisionError

Ba’zi dastur davomida arifmetik amallar bajarilishi natijasida 0 ga bo‘lish xatoligi (ZeroDivisionError) yuzaga kelishi mumkin. Aynan shu xatoni jilovlash uchun, except ZeroDivisionError ifodasidan foydalanamiz:

```
x,y=5,10
try:
   y/(x-5)
except ZeroDivisionError:
    print("0 ga bo‘lib bo‘lmaydi")
```

Natija: 0 ga bo‘lib bo‘lmaydi

IndexError

Bu xatolik odatda ro‘yxatda mavjud bo‘lmagan indeksga murojat qilishda chiqib keladi.

```
mevalar = [’olma’,’anor’,’anjir’,’uzum’]
try:
    print(mevalar[7])
except IndexError:
    print(f"Ro‘yxatda {len(mevalar)} ta meva bor xolos")
```

Natija: Ro‘yxatda 4 ta meva bor xolos

KeyError

Bu xatolik lug‘atda mavjud bo‘lmagan kalitga murojat qilishda kelib chiqadi:

```
user = {"username":"sariqdev",
        "status":"admin",
        "email":"admin@sariq.dev",
        "phone":"99897123456"}

key="tel"
try:
    print(f’Foydalanuvchi: {user[key]}’)
except KeyError:
    print("Bunday kalit mavjud emas")
```

Natija: Bunday kalit mavjud emas
FileNotFoundError
Avvalgi darsimizda fayllar bilan ishlashni o‘rgangan edik. Fayllarni biz o‘qish (open(filename,’r’)) yoki yozish (open(filename,’w’)) uchun ochishimiz mumkin. Agar faylga ma’lumot yozish uchun ochishda, mavjud bo‘lmagan faylga murojat qilsak, Python yangi fayl yaratadi. Lekin, faylni o‘qish uchun ochishda fayl nomini xato yozsak, yoki mavjud bo‘lmagan faylni ochmoqchi bo‘lsak FileNotFoundError (fayl topilmadi) xatoligi yuzaga keladi.

```
filename = "data.txt" # bunday fayl mavjud emas
with open(filename) as f:
    text = f.read()
```

Natija: FileNotFoundError: [Errno 2] No such file or directory: ’data.txt’
Demak, bu xatolikni ham ushlab qolish uchun except FileNotFoundError ifodasidan foydalanamiz:

```
filename = "data.txt" # bunday fayl mavjud emas
try:
    with open(filename) as f:
        text = f.read()
except FileNotFoundError:
    print(f"Kechirasiz, {filename} fayli mavjud emas. Bosh fayl tanlang.")
```

Natija: Kechirasiz, data.txt fayli mavjud emas. Bosh fayl tanlang.
Pythonda bundan boshqa xatolar ham ko‘p uchraydi, ularning ba’zilari 12-darsda tanishgan edik.

.

BIR NECHTA XATOLARNI USHLASH

try-except ketma-ketligida bir nechta except operatorlari ham bo‘lishi mumkin. Ularning har biri ma’lum turdagi xatolik uchun javobgar bo‘ladi:

```
n = input("Butun son kiriting: ")
try:
    n = int(n)
    x=20/n
except ValueError: # agar foydalanuvchi butun son kiritmasa
    print("Butun son kiritmadingiz")
except ZeroDivisionError: # agar foydalanuvchi 0 kiritsa
```

```
    print("0 ga bo‘lib bo‘lmaydi")
else:
print(f"x={x}")
```

XATOLARNI KO‘RSATMAY O‘TISH

Yuqoridagi misollarda kodimiz xato qaytarganda, dasturimiz foydalanuvchiga qandaydur ma’lumotni ko‘rsatayapti:

```
import json
files = [’talaba1.json’,’talaba2.json’,’talaba3.json’,’talaba4.json’]
for filename in files:
    try:
        with open(filename) as f:
            talaba = json.load(f)
    except FileNotFoundError:
        print(f"{filename} mavjud emas")
    else:
        print(talaba[’ism’])
        # fayl ustida turli amallar
```

Hech qanday ma’lumot ko‘rsatmay, dasturni davom etish uchun pass operatoridan foydalanamiz. Odatda pass operatoridan funksiyalar yoki operatorlarning badanini "to‘ldirib" ketish uchun ishlatiladi.

Ya’ni, agar biz except operatorini yozsagu, uning badanida hech narsa bajarilishini istmasak, pass operatorini ishlatamiz.

```
import json
files = [’talaba1.json’,’talaba2.json’,’talaba3.json’,’talaba4.json’]
for filename in files:
    try:
        with open(filename) as f:
            talaba = json.load(f)
    except FileNotFoundError:
        pass
    else:
        print(talaba[’ism’])
```

Xatolarning oldini olish. Yuqorida biz xatolar yuz berganda, ularni ushlash va dastur to‘xtashining oldini olishni ko‘rdik. Ya’ni, try-except ketma-ketligi xatolarning oldini olishga yordam bera olmaydi. Xatolarning oldini olish uchun if-else va while tsikllaridan foydalanganimiz afzal.

Mavzu boshidagi misolimizga qaytsak. Biz foydalanuvchi yoshini so‘radik, va foydalanuvchi butun son kiritmagani sababli dasturni to‘xtatdik. Aslida, while tsikli yordamida toki foydalanuvchi to‘g‘ri qiymat kritgunga qadar uning yoshini qayta-qayta talab qilishimiz mumkin:

```
while True:
    yosh = input("Yoshingizni kiriting: ")
    if yosh.isdigit():
        yosh = int(yosh)
        break

print(f"Siz {2021-yosh} yilda tug‘ilgansiz")
```

Albatta yuqordagi usul barcha xatolar uchun ishlamaydi. Shunday xatolar bo‘lishi mumkinki, biz ularni oldindan ko‘ra olmasligimiz yoki, oldindan to‘g‘rila olmasligimiz, yoki xato foydalanuvchiga bog‘liq bo‘lmasligi mumkin. Shunday holatlarda, try-except operatorlari bizning xaloskorimiz bo‘ladi.

Nazorat savollari.

1. Xatoliklar turlarini sanab bering?
2. Ma’lum turdagi xatoliklar qanday ushlanadi?
3. Xatoliklarni oldini olish usullari?

13-14-ma’ruza.Pythonda sinflar va obyektlar

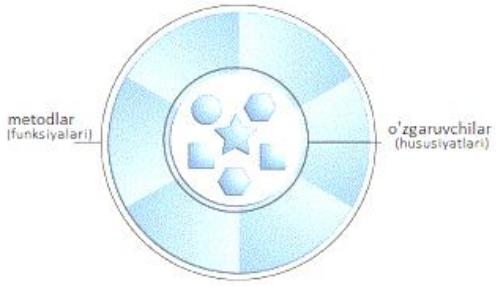
Reja:

1. Pythonda obyektlar?
2. Pythonda sinflar?
3. Klassning xususiyatlari va metodlari.

Obyekt Obyektga yo‘naltirilgan dasturlash (OYD) texnologiyasining eng asosiy kalit tushunchasidir. Atrofga qarang, haqiqiy hayotdagi bir necha obyektlarni ko‘rishingiz mumkin: stol, uy, it, mushuk, televizor va h.k.

Ularning barchasining albatta hususiyatlari va bajaradigan vazifalari (funktsiyalari) bor. Masalan, Mushuk hususiyatlari: rangi, qorni to‘qligi, yoshi, jinsi; funktsiyalari: ovqat yeyishi, myovlashi, yurishi, sichqon tutishi. Mashina, hususiyatlari: tezligi, rangi, nomi, narxi; funktsiyalari: yurishi, to‘xtashi, oyna artgichlarining ishlashi, eshiklarning ochilib yopilishi v.h.k. Bu kabi hayotiy misollarning hususiyatlari va funktsiyalarini aniqlash OYD nuqtai nazaridan fikrlashning eng zo‘r ko‘rinishidir.

Kompyuter indikatorining 2 ta xususiyati bor o‘chiq va yoniq; funktsiyalari esa yonish va o‘chish. Bu barcha kuzatishlar OYD dunyosiga o‘tkazish mumkin.



Dasturlashdagi obyekt.

Dasturlashdagi obyekt(bundan keyin oddiygina obyekt deb ketiladi) ham haqiqiy hayotdagi obyektlarga o‘xshash: Ular ham qandaydir hususiyatlar va bajaradigan funktsiyalardan iborat bo‘ladi. Obyektning hususiyatlari har xil dasturiy o‘zgaruvchilardan iborat bo‘ladi va ularning o‘zgartirish uchun qandaydir funktsiyalar bajariladi. Bunday funktsiyalar bilan o‘zgaruvchilarning holatini berkitish mumkin ya’ni aynan o‘sha o‘zgaruvchini tashqaridan o‘zgartirish uchun albatta maxsus funktsiyadan foydalanish kerak bo‘ladi. Bu jarayon "Enkapsulatsiya" deb atalib, OYDning eng muxim

tushunchalaradian biridir. Hech e’tibor berganmisiz dorilarda ham shu termin ishalitladi ya’ni kapsula(ustidan maxsus modda bilan o‘ralgan dorilar), buni misolni Enkapsulatsiya jarayoni esda yaxshi qolishi va tushunarli bo‘lishi uchun keltirdik.

Obyektga yo‘naltirilgan dasturlash.

Shu paytgacha python bo‘yicha darslarimizda keltirilgan dasturlar faqat funksiyalardan tashkil topgan edi. Ya’ni ma’lum bir ma’lumotlarni qayta ishlaydigan ifodalar blokidan iborat bo‘ldi. Bu prosedura ko‘rinishidagi dastrulash uslubi hisoblanadi. Dasturlarni tashkil qilishning boshqa ko‘rinishi ham mavjud: ma’lumotlar va funksiyalarni bir ob’ekt ostiga jamlash. Bu dasturlashning ob’ektga yo‘naltirilgan modeli hisoblanadi.

Klass va ob’yektlar – ob’yektga yo‘naltirilgan dasturlashning ikkita asosiy aspekti. Klass yangi tur (тип) hosil qiladi, ob’ektlar esa klassning nusxasi hisoblanadi. Masalan, “int turidagi o‘zgaruvchi” deganda, butun son qiymatlarni saqlovchi o‘zgaruvchilar int klassining nusxasi – ob’yekti ekanligini tushunishimiz kerak.

Obyektlar ma’lumotlarni o‘ziga tegishli bo‘lgan o‘zgaruvchilarda saqlaydi. Obyekt yoki klassga tegishli bo‘lgan o‘zgaruvchilar maydonlar deyiladi. Shu bilan birga obyektlar klasslarga tegishli bo‘lgan funksiyalarni ham o‘zlida jamlaydi. Bunday funksiyalar klass metodlari deb ataladi. Maydon va metodlarni umumiy qilib klass attributlari deb atash mumkin.

Maydonlar ikki turda bo‘ladi: ular aloxida ob’yektga tegishli bo‘lishi mumkin yoki butun bir klassga tegishlik bo‘lishi mumkin.

Ular o‘z navbatida ob’yekt o‘zgaruvchilari va klass o‘zgaruvchilari deb nomlanadi.

Klasslar class kalit so‘zi bilan xosil qilinadi.

```
class Person:
pass # Bo‘sh blok
```

Klass ob’yekti quyidagicha xosil qilinadi:

```
p = Person()
```

Misol:

```
class Person:
 def sayHi(self):
 print(’Salom! Ishlar qanday?’)
 p = Person()
 p.sayHi()
```

Natja:
>>>Salom! Ishlar qanday?
Bu qanday ishlaydi:

Klass metodlari oddiy funksiyardan parametrlari boshiga qo‘shiladigan self qo‘shimcha parametri bilan farqlanadi.

Person klassining sayHi metodida bu narsa ko‘rsatilgan.

Lekin bu metodni ob’yekt orqali chaqirganimizda xech qanday qiymat bu parametr uchun berilmasligi kerak. self o‘zgaruvchisi klass ob’yektiga murojat qilish uchun ishlatiladi.

__init__ metodi

__init__ (boshida va oxirida ikkita “_” belgisi) metodi klass ob’yekti xosil qilinayotgan vaqtda chaqiriladi. Bu metod ob’yekt uchun kerak bo‘ladigan boshlang‘ich o‘zlashtirishlar uchun foydali.

Misol: (class_init.py nomi bilan saqlang)

```
class Person:
 def __init__(self, name):
 self.name = name
 def sayHi(self):
 print(’Salom! Mening ismim’, self.name)
  p = Person(’Swaroop’)
 p.sayHi()
```

Natija:

>>>**Salom! Mening ismim Swaroop**

Bu qanday ishlaydi:

Bu yerda biz __init__ metodini self parametri yonida name parametrini qabul qiladigan qilib aniqladik. So‘ngra name (self.name) maydonini aniqladik. Bu yerda self.name ob’yektga tegishli maydon name esa mahalliy o‘zgaruvchi.

Ahamiyat berish kerakki, klass ob’yektini xosil qilishda __init__ metodini ochiq chaqirganimiz yo‘q, uning o‘rniga klass nomidan so‘ng argumentlarni qavs ichida berdik xolos. Shu holat bilan bu metodning roli bog‘liq.

Shundan so‘ng biz self.name maydonini metodlarimizda ishlatish imkoniga ega bo‘lamiz. Buni sayHi metodi ko‘rsatib turibdi.

Obyektga yo‘naltirilgan dasturlashning tamoyillaridan biri bu inkapsulyatsiya, ya’ni obyektning xususiyatlarga to‘g‘ridan-to‘g‘ri (nuqta orqali) murojat qilishni va ularning qiymatini o‘zgartirishni taqiqlab qo‘yish. Pythonda bunday yopiq xususiyatlarning nomi ikki

pastki chiziq bilan boshlanadi:

```
from uuid import uuid4
class Avto:
    """Avtomobil klassi"""
    def __init__(self,make,model,rang,yil,narh,km=0):
        """Avtomobilning xususiyatlari"""
        self.make = make
        self.model = model
        self.rang = rang
        self.yil = yil
        self.narh = narh
        self.__km = km
        self.__id = uuid4()

    def get_km(self):
        return self.__km
    def get_id(self):
        return self.__id
```

Yuqoridagi kodimizning 11-qatorida __km xususiyati avtomobilning necha km yurgani haqida ma’lumot saqlaydi va bu ma’lumotni tashqaridan o‘zgartirib bo‘lmaydi. Kodimizning 12-qatorida esa har bir yangi yaratilgan avtomobilga yangi, noyob va takrorlanmas ID generasiya qilish uchun uuid4() funksiyasidan foydalanayapmiz. Deylik biz mashinalar sotish uchun onlayn bozor yaratsak, bozorimizga qo‘shilgan har bir moshina endi o‘zining ID raqamiga ega bo‘ladi va bu ID raqamni to‘g‘ridan-to‘g‘ri (nuqta orqali) ko‘rib bo‘lmaydi.

```
avto1 = Avto("GM","Malibu","Qora",2020,40000,100000)
avto1.__km
```

Natija: AttributeError: ’Avto‘ object has no attribute ’__km’

Yopiq xususiyatlarni ko‘rish uchun esa alohida metodlar yozish maqsadga muvofiq bo‘ladi (get_km()) va get_id()):

```
print(f"ID: {avto1.get_id()}")
```

Natija: ID: 1d4f39a4-3222-4682-9231-6275ca5e1bff

Bunday yopiq xususiyatlarni o‘zgartirish ham metodlar orqali amalga oshirilishi kerak. Misol uchun mashinaning necha km

yurganini o‘zgartirish uchun klassimizga quyidagi metodni qo‘shamiz:

```
def add_km(self,km):
        """Mashinaning km ga yana km qo‘shish"""
        if km>=0:
            self.__km += km
        else:
            print("Mashina km kamaytirib bo‘lmaydi")
#########################
avto1.add_km(1500)
print(avto1.get_km())
```

Natija: 101500

Inkapsulyatsiyaning maqsadi obyektning ma’lum xususiyatlarini tashqi ta’sirdan himoya qilish. Misol uchun yuqoridagi misolimizda mashinaning qancha yurganini faqat musbat tarafga o‘zgartirish mumkin, noyob ID raqamini esa umuman o‘zgartirib bo‘lmaydi.

Klassning xususiyatlari va metodlari. Avvalgi darslarimizda biror klassdan yaratilgan har bir obyektning xususiyatlarini klass ichidagidef __init__() medoti yordamida berayotgan edik. Bu metod qanday ishlaydi? Biz har gal klassga murojat qilganimizda klass aynan shu __init__metodini ishga tushiradi va biz bergan xususiyatlar bilan yangi obyekt yaratadi.

Pythonda xususiyatlarni nafaqat obyektga balki to‘g‘ridan-to‘g‘ri klassga ham yuklash imkoniyati mavjud. Bunda klassning xususiyatiga berilgan qiymat barcha obyektlar uchun umumiy bo‘ladi.

Bu xususiyatni bir obyekt orqali o‘zgartirilganda shu klassga oid barcha obyektlarda ham uning qiymati o‘zgaradi.

Klassga oid xususiyatlar def __init__() metodidan avval e’lon qilinadi. Keling tushunarli bo‘lishi uchun quyidagi misolni ko‘ramiz:

```
class Avto:
    """Avtomobil klassi"""
    num_avto = 0
    def __init__(self,make,model,rang,yil,narh,km=0):
        """Avtomobilning xususiyatlari"""
        self.make = make
        self.model = model
        self.rang = rang
        self.yil = yil
```

```
        self.narh = narh
        self.__km = km
        self.__id = uuid4()
        Avto.num_avto += 1
```

Kodni tahlil qilamiz:

1-qatroda Avto klassini e'lo qildik

3-qatorda Avto klassiga oid bo'lgan xususiyat num_avto yaratdik va unga 0 qiymatini berdik

Keyingi qatorlarda __init__ metodi yordamida klassdan yaratiladigan obyektlarning xususiyatlarini e'lon qildik va har gal bu metdoga murojat qilingandan num_avto ning qiymatini 1 ga oshradigan qilib qo'ydik (13-qator).

Yuqoridagi usul bilan endi biz dastur davomida Avto klassidan jami nechta obyektlar yaratilganini kuzatib borishimiz mumkin bo'ladi. Bunda num_avto o'zgaruvchisiga istalgan obyekt orqali yoki klass nomi orqali murojat qilish mumkin:

```
avto1 = Avto("GM","Malibu","Qora",2020,40000)
avto2 = Avto("GM","Lacetti","Oq",2020,20000)
print(avto1.num_avto)
```

Copied!

Natija: 2

```
avto3 = Avto("Toyota",'Carolla',"Silver",2018, 45000)
print(Avto.num_avto)
```

Copied!

Natija: 3

Klassning xususiyatlarini inkapsulyatsiya qilish. Klassga oid xususiyatlar ham huddi yuoqridagi kabi nomidan avval ikki pastki chiziq qo'shish bilan inkapsulyatsiya qilinadi:

```
class Avto:
    """Avtomobil klassi"""
    __num_avto = 0 # klassga oid xususiyat
    def __init__(self,make,model,rang,yil,narh):
        """Avtomobilning xususiyatlari"""
        self.make = make
        self.model = model
        self.rang = rang
```

```
        self.yil = yil
        self.narh = narh
        Avto.__num_avto += 1
```

Klassga oid metodlar. Klasslarning o‘ziga xos metodlari ham bo‘lishi mumkin. Misol uchun yuqoridagi num_avto xususiyatini ko‘rish uchun alohida metod yozishimiz mumkin. Klassga oid metodlar @classmethod dekoratori bilan boshlanadi va obyektga oid metodlardan farqli ravishda *self* emas *cls* (class) argumentini qabul qiladi.

```
class Avto:
    """Avtomobil klassi"""
    __num_avto = 0
    def __init__(self,make,model,rang,yil,narh,km=0):
        """Avtomobilning xususiyatlari"""
        self.make = make
        self.model = model
        self.rang = rang
        self.yil = yil
        self.narh = narh
        self.__km = km
        self.__id = uuid4()
        Avto.__num_avto += 1

    @classmethod
    def get_num_avto(cls):
        return cls.__num_avto
```

Copied!

Klass metodlarga klassning nomi orqali murojat qilinadi:

```
avto1 = Avto("GM","Malibu","Qora",2020,40000)
avto2 = Avto("GM","Lacetti","Oq",2020,20000)
avto3 = Avto("Toyota",’Carolla’,"Silver",2018, 45000)
print(Avto.get_num_avto())
```

Natija: 3

@classmethod bu maxsus dekorator. Dekoratorlar bu o‘z ichiga funksiya oluvchi funksiyalar. Dekoratorlar haqida keyingi darslarimizning birida batafsil to‘xtalamiz.

KLASSLARNI MODULGA AJRATISH

Vaqt o‘tishi bilan dasturimizda klasslar ko‘payib borishi tabiiy.

Bizning asosiy dasturimiz uzun va chigal bo‘lmasligi uchun klasslarni ham huddi funksiyalar kabi alohida modullarga ajratish maqsadga muvofiq bo‘ladi. Dastur davomida kerak bo‘ladigan klasslarga esa modulni chaqirish (import) orqali murojat qilishimiz mumkin. Bunda, bir-biriga bog‘liq klasslarni bitta faylga joylashimiz mumkin.

Misol uchun, biz Talaba, Professor, Foydalanuvchi va Shaxs degan klasslarni bitta odamlar.py moduliga, Avto, Bus, Train degan klasslarni esa boshqa transport.py moduliga joyladik. Kelajakda biz bu klasslarga import orqali murjat qilishimiz mumkin.

Bitta klassni import qilish. Moduldan bitta klass import qilish uchun from modul import klass ifodasidan foydalanamiz:

```
from odamlar import Talaba
from transport import Avto
talaba = Talaba("Alijon","Valiyev","FA010101","N00022")
avto = Avto("GM","Malibu","Qora",2020,40000)
```

Bir nechta klasslarni import qilish. Moduldan bir nechta klass chaqirish talab qilinsa, import so‘zidan so‘ng klasslar ketma-ket vergul bilan ajratib yoziladi:

```
from odamlar import Talab, Shaxs
talaba = Talaba("Alijon","Valiyev","FA010101","N00022")
shaxs = Shaxs("Hasan","Husanov","FB0011223")
```

Copied!

Modulni o‘zini chaqirish. Modulni to‘liq import qilish uchun import modul ifodasidan foydalanamiz. Bunda modul ichidagi klasslarga modul nomi va nuqta orqali murojat qilinadi:

```
import odamlar
talaba = odamlar.Talaba("Alijon","Valiyev","FA010101","N00022")
shaxs = odamlar.Shaxs("Hasan","Husanov","FB0011223")
```

Moduldagi barcha klasslarni import qilish. Moduldagi barcha klasslar quyidagicha import qilinadi: from modul import *. Bunda modul ichidagi istalgan klassga to‘g‘ridan-to‘g‘ri uning nomi bilan murojat qilinadi.

```
from odamlar import *
talaba = Talaba("Alijon","Valiyev","FA010101","N00022")
shaxs = Shaxs("Hasan","Husanov","FB0011223")
```

Bu usul 2 sababga ko‘ra tavsiya qilinmaydi:

Dasturni kelajakda qayta ochganimizda, dastur davomida moduldagi

aynan qaysi klasslardan foydalanganimizni bir qarashda bilib bo‘lmaydi
Agar modul juda katta bo‘lsa, uning ichidagi ba’zi klasslar biz o‘zimiz yaratgan klasslar bilan nomi bir hil bo‘lib qolish ehtimoli bor. Bu esa o‘z navbatida dastrumiz xato ishlashiga olib keladi.
! Modul ichiga boshqa modulni ham import qilish mumkin. Masalan modul ichidagi ba’zi klasslar to‘g‘ri ishlashi uchun boshqa modul talab qilingan vaqtlarda.

Nazorat savollari.

1. Obyekt nima?
2. Sinflar qanday yaratiladi?
3. Sinflar ustida metodlar?

15-ma’ruza. Vorislik tushunchasi

Reja:

1. Voris class yaratish.
2. Voris klassga xos xususiyatlar va metodlar.
3. Obyekt ichida obyekt.

Obyektga yo‘naltirilgan dasturlashning qulayliklaridan biri bu klasslardan boshqa klass yaratish imkoniyati. Bizga kerak bo‘lgan yangi klass, avval yaratilgan boshqa klass bilan o‘xshashlik joylari bo‘lsa, biz bu klassdan voris klass yaratishimiz mumkin. Bunda asl klass — ota, yoki super klass deb ataladi.

Super klassdan yaratilgan voris klass otaning barcha yoki tanlangan xususiyatlari va metodlarini meros olish bilan birga, o‘ziga xos xususiyat va metodlariga ega bo‘ladi.

Keling boshlanishiga Shaxs klassini yaratamiz, bu klassimiz keyinchalik boshqa klasslar uchun super klass vazifasini bajaradi:

```
class Shaxs:
    """Shaxslar haqida ma’lumot"""
    def __init__(self,ism,familiya,passport,tyil):
        self.ism = ism
        self.familiya = familiya
        self.passport = passport
        self.tyil = tyil

    def get_info(self):
        """Shaxs haqida ma’lumot"""
        info = f"{self.ism} {self.familiya}. "
        info += f"Passport:{self.passport}, {self.tyil}-yilda tug‘ilgan"
        return info
    def get_age(self,yil):
        """Shaxsning yoshini qaytaruvchi metod"""
        return yil — self.tyil
```

Klassimizni tekshirib ko‘ramiz:

```
inson = Shaxs("Hasan","Alimov","FB001122",1995)
print(f"{inson.get_info()}. {inson.get_age(2021)} yoshda.")
```

Natija: Hasan Alimov. Passport:FB001122, 1995-yilda tug‘ilgan. 26 yoshda.

VORIS KLASS YARATISH

Endi avvalgi darsimizda yaratgan Talaba klassimizni qaytadan yaratamiz. Bu safar, avvalgidan farqli ravishda, Talaba ni yaratishda, Shaxs dan super klass sifatida foydalanamiz:

```
class Talaba(Shaxs):
    """Talaba klassi"""
    def __init__(self, ism, familiya, passport, tyil):
        """Talabaning xususiyatlari"""
        super().__init__(ism, familiya, passport, tyil)
```

Kodimizni tahlil qilaylik:

1-qatorda klass nomidan so‘ng, qavs ichida super klass nomini berdik

5-qatorda (def __init__ ichida) klassimiz super klassning xususiyatlarini meros olishini ko‘rsatdik

Yangi yaratgan Talaba klassimiz Shaxsning barcha xususiyatlari va metodlariga ega bo‘ladi.

```
talaba = Talaba("Valijon","Aliyev","FA112299",2000)
print(talaba.get_info())
```

Natija: Valijon Aliyev. Passport:FA112299, 2000-yilda tug‘ilgan

Talaba klassi uchun alohida get_info() metodini yozmagan bo‘lsakda, bu metod Talabaga Shaxsdan meros o‘tdi.

Huddi shu kabi get_age() metodiga ham murojat qilishimiz mumkin:

```
>>>print(talaba.get_age(2021))
21
```

Dastur davomida super klass voris klasslardan avval yozilgan (chaqirilgan) bo‘lishi kerak.

VORIS KLASSGA XOS XUSUSIYATLAR VA METODLAR

Hozirgi ko‘rinishda Talaba va Shaxs klasslari o‘rtasida hech qanday farq yo‘q. Keling Talaba klassimizga o‘ziga xos xususiyatlar va metodlar yarataylik. Avvalosiga, talabaning bosqichi va ID raqamini xususiyat sifatida qo‘shamiz. Bunda ID raqami obyekt yaratilishida parameter sifatida uzatiladi, bosqich esa standart qiymatga ega.

```
class Talaba(Shaxs):
    """Talaba klassi"""
    def __init__(self, ism, familiya, passport, tyil,idraqam):
        """Talabaning xususiyatlari"""
        super().__init__(ism, familiya, passport, tyil)
        self.idraqam = idraqam
        self.bosqich = 1
```

Endi yangi, Talaba obyektini yaratishda qo‘shimcha idraqam

parametrini ham kiritish talab qilinadi:
talaba = Talaba("Valijon","Aliyev","FA112299",2000,"0000012")
So‘ngra, bu qiymatlarni qaytaruvchi alohida metodlar yozamiz:

```
class Talaba(Shaxs):
    """Talaba klassi"""
    def __init__(self, ism, familiya, passport, tyil,idraqam):
        """Talabaning xususiyatlari"""
        super().__init__(ism, familiya, passport, tyil)
        self.idraqam = idraqam
        self.bosqich = 1
     def get_id(self):
        """Talabaning ID raqami"""
        return self.idraqam
      def get_bosqich(self):
        """Talabaning o‘qish bosqichi"""
        return self.bosqich
```

Metodlarni tekshirib ko‘ramiz:

```
>>>print(f"{talaba.get_info()}. ID raqami:{talaba.get_id()}")
Valijon Aliyev. Passport:FA112299, 2000-yilda tug‘ilgan. ID raqami:0000012
>>>print(f"{talaba.get_bosqich()}-bosqich talabasi")
1-bosqich talabasi
```

Shu zayilda yangi klassimizga istalgancha yangi xususiyatlar va metodlar qo‘shishimiz mumkin. Bunda, agar yangi xususiyat yoki metod super klassga ham aloqador bo‘lsa uni birdan super klassga qo‘shish tavsiya qilinadi.
Voris klass boshqa klass uchun super klass bo‘lishi mumkin.
POLIMORFIZM — SUPER KLASS METODLARINI QAYTA YOZISH
Voris klassga super klassdan meros qolgan istalgan metodni qayta talqin qilish mumkin. Avvalgi misolimizdagi get_info() super metodini ko‘raylik, bu metod talabaning ismi, familiyasi, passport raqami va tug‘ilgan yilini qaytaradi:

```
>>> print(talaba.get_info())
Valijon Aliyev. Passport:FA112299, 2000-yilda tug‘ilgan
```

Endiget_info() metodi talabaga mos ma’lumotlar qaytarishi uchun, Talaba klassi ichida huddi shu nomli metodni qayta yozamiz:

```
class Talaba(Shaxs):
```

```
    """Talaba klassi"""
    def __init__(self,ism,familiya,passport,tyil,idraqam):
        """Talabaning xususiyatlari"""
        super().__init__(ism, familiya, passport, tyil)
        self.idraqam = idraqam
        self.bosqich = 1

    def get_id(self):
        """Talabaning ID raqami"""
        return self.idraqam

    def get_bosqich(self):
        """Talabaning o‘qish bosqichi"""
        return self.bosqich

    def get_info(self):
        """Talaba haqida ma’lumot"""
        info = f"{self.ism} {self.familiya}. "
        info += f"{self.get_bosqich()}-bosqich. ID raqami: {self.idraqam}"
        return info
```

Metodni tekshirib ko‘ramiz:

```
>>> print(talaba.get_info())
Valijon Aliyev. 1-bosqich. ID raqami: 0000012
```

OBYEKT ICHIDA OBYEKT

Ba’zida klassimiz xususiyatlar va ular bilan ishlaydigan metodlarga to‘lib ketishi, bu esa o‘z navbatida obyektga murojat qilishni qiyinlashitirishi mumkin. Shunday holatlarda ba’zi xususiyatlarni alohida klass ko‘rinishida yozish va keyinchalik bu klassdan yaratilgan obyektni boshqa obyektning xususiyati sifatida foydalanish mumkin.

Misol uchun, yuqoridagi Shaxs klassimizga yana bir manzil degan xususiyat qo‘shaylik. Odatda manzil bir nechta qismlardan iborat bo‘ladi (xonadon, ko‘cha, mahalla, tuman/shahar, viloyat, indeks va hokazo) va ularning har biri uchun Shaxs ichida alohida xususiyat yaratmasdan, alohida manzil degan klassga yuklash maqsadga muvofiq bo‘ladi.

```
class Manzil:
```

```
    """Manzil saqlash uchun klass"""
    def __init__(self,uy,kocha,tuman,viloyat):
        """Manzil xususiyatlari"""
        self.uy = uy
        self.kocha = kocha
        self.tuman = tuman
        self.viloyat = viloyat

    def get_manzil(self):
        """Manzilni ko‘rish"""
        manzil = f"{self.viloyat} viloyati, {self.tuman} tumani, "
        manzil += f"{self.kocha} ko‘chasi, {self.uy}-uy"
        return manzil
```

Talaba klassimizga ham qo‘shimcha manzil xususiyatini qo‘shamiz:

```
class Talaba(Shaxs):
    """Talaba klassi"""
    def __init__(self,ism,familiya,passport,tyil,idraqam,manzil):
        """Talabaning xususiyatlari"""
        super().__init__(ism, familiya, passport, tyil)
        self.idraqam = idraqam
        self.bosqich = 1
        self.manzil = manzil

    def get_id(self):
        """Talabaning ID raqami"""
        return self.idraqam

    def get_bosqich(self):
        """Talabaning o‘qish bosqichi"""
        return self.bosqich

    def get_info(self):
        """Talaba haqida ma’lumot"""
        info = f"{self.ism} {self.familiya}. "
        info += f"{self.get_bosqich()}-bosqich. ID raqami:
{self.idraqam}"
        return info
```

Keling endi talaba obyektini qayta yaratamiz. Bu safar talabaning

manzili ham alohida obyekt sifatida talaba ga uzatiladi:

```
talaba_manzil = Manzil(12,’Olmazor’,"Bog‘bon","Samarqand")
talaba = Talaba("Valijon","Aliyev","FA112299",2000,"0000012",talaba_manzil)
```

Obyekt ichidagi obyektning xususiyatlari va metodlariga ham avvalgidek nuqta orqali murojat qilishimiz mumkin:

```
>>> print(talaba.manzil.get_manzil())
Samarqand viloyati, Bog‘bon tumani, Olmazor ko‘chasi, 12-uy
>>> print(talaba.manzil.tuman)
Bog‘bon
```

Nazorat savollari.

1. Obyekt ichida obyekt qanday hosil qilinadi?
2. Polimorfizm nima?
3. Voris classga xos xususiyatlar qaysilar?

16-17-ma’ruza.Grafik modular bilan ishlash.

Reja:

1. Python dasturlash tilida grafik modullar.
2. Matematik grafiklar chizish.

Python grafik foydalanuvchi interfeyslarini (GUI) dasturlash uchun turli xil xususiyatlar ca imkoniyatlarni taqdim etadi. Eng muhimlari:

Tkinter — Tkinter Python bilan ta’minlangan Python Tk GUI interfeysidir.

wxPython — wxWindows uchun http://wxpython.org ochiq manba kodli Python interfeysi.

JPython — JPython Python skriptlarini mahalliy kompyuterdagi Java sinf kutubxonalariga osonlik bilan kirish imkonini beruvchi Java porti uchun Pythondir.

Tkinter standart Python GUI kutubxonasidir. Python Tkinter bilan birgalikda GUI ilovalarini yaratish uchun tezkor va qulay usulni ta’minlaydi. Tkinter Tk GUI asboblar to‘plami uchun kuchli ob’ektga asoslangan interfeysni ta’minlaydi.

Tkinter bilan GUI dasturini yaratish oddiy vazifadir. Buning uchun quyidagi bosqichlarni bajarish kerak:

Tkinter modulini import qiling.

Grafik interfeysli asosiy dastur oynasini yaratish.

Kerakli vidjetlardan bir yoki bir nechtasini GUI ilovasiga qo‘shing.

Foydalanuvchi tomonidan boshlangan har bir tadbirga qarshi harakat qilish uchun asosiy hodisani ko‘chasini kiriting.

Graph moduli

from graph import *-graph modulning barcha funksiyalarini ulash.

Chiziq rangi:

penColor("red")

white, black, gray, navy, blue,cyan, green, yellow, red, orange,brown, maroon, violet, purple, ...

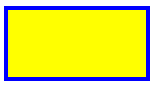
Chiziq kengligi:

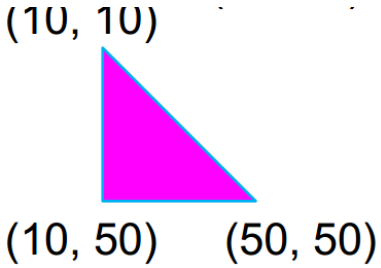
penSize(2)

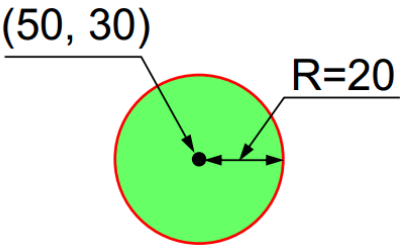
To‘ldirish rangi:

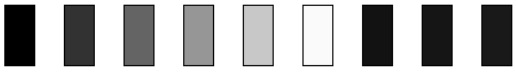
brushColor(“green”)

RGB(Red, Green, Blue) rangi:
penColor(255,255,0)
point(x, y)-x,y kordinatada nuqta chizish
from graph import*
penColor(0, 0, 255)
point(200, 300)
line(x1,y1,x2,y2)-chiziq chizish
from graph import*
penColor(0, 0, 255)
line(100,100, 400,400)
Bir nechta chiziqlar chizish:
penColor(255, 0, 0)
moveTo(x1, y1)
lineTo(x2, y2)
lineTo(x3, y3)
lineTo(x4, y4)
lineTo(x5, y5)
Primitivlar(eng oddiy shakllar):
To‘rtburchak chizish:
penColor("blue") #chiziq rangi
brushColor("yellow") #to‘ldirish rangi
rectangle(10, 20, 50, 40) #shakl nomi
#faqat shakl nomini yozsa oddiy rangsiz konturga ega #shakl chizadi
Uchburchak chizish:
penColor("cyan")
brushColor("magenta")
polygon([(10, 10), (50, 50),
(10, 50), (10, 10)])
Aylana chizish:
penColor("red")
brushColor("green")
circle(50, 30, 20)
Turli xil ranglar bilan to‘ldirish(gradian):
from graph import*
x = 50
c = 0
for i in range(20):
    brushColor(c, c, c)





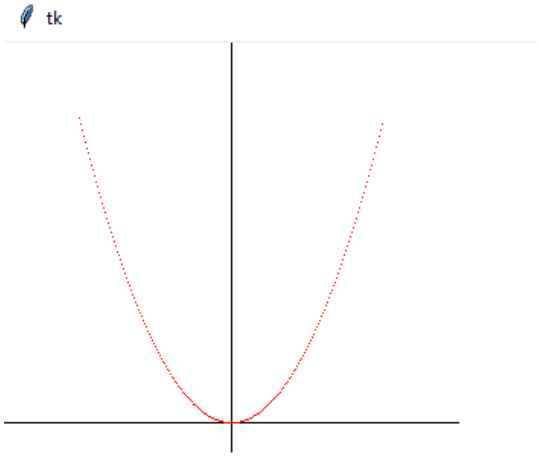




```
    rectangle(x, 50, x+25, 100)
    x += 50
    c += 50
```

$y=x^2$ funksiya grafigini chizish:

```
from graph import *
x0 = 150
y0 = 250
k = 50
xmin = -2; xmax = 2
line(0, y0, x0+150, y0)
line(x0, 0, x0, y0+20)
x = xmin
h = 0.02
penColor("red")
while x <= xmax:
  y = x*x # funsiya
  xe = x0 + k*x
  ye = y0 — k*y
  point(xe, ye)
  x += h
run()
```



17-ma’ruza.Turtle moduli

Ushbu ma’ruzada biz Turtle (Turtle) grafik moduli yordamida Pythondagi grafik dasturlarni tuzamiz. Tirtle moduli Python uchun maxsus oynada grafik ob’ektlar, chizmalar yaratishga imkon beruvchi moduldir.

Turtle grafik modulidan pythonda o‘yinlarini yaratish uchun foydalanish mumkin.

Tirtle grafik modulini biriktirish (import qilish) uchun siz quyidagi usullardan birini tanlashingiz kerak:

1. import turtle
2. from turtle import*
3. from turtle import open as t

Birinchi usul modulni import qilish, ammo buyruqlarni modulga havola bilan yozish kerak (havola sifatida) atribut) OOP uslubida.

Ikkinchi usul sizga modul nomlarini eslatmasdan to‘g‘ridan-

to‘g‘ri modul funktsiyalariga kirishga imkon beradi.
Uchinchi usul sizga modul nomini o‘zingizning ismingiz bilan, ya’ni yozish o‘rniga o‘zgartirishga imkon beradi to‘liq modul nomi biz faqat bitta belgidan foydalanamiz,
Masalan:
t.reset ()
t.fd (100)

Tirtle moduli bilan ishlashning o‘ziga xos xususiyatlari quyidagilarni o‘z ichiga oladi grafik muhit -maydon deb nomlangan, ya’ni ijrochi harakatlanadigan maydon. Buning uchun parametrsiz reset()buyrug‘idan foydalaning. Bundan tashqari, u barcha sozlamalarni standart ko‘rinishiga keltirib, ijrochini dastlabki holatiga qaytaradi.

Ijrochiining tashqi ko‘rinishi (standart uslubi classic qora rangda) — har doim o‘q uchi u xarakatlanadagan tomon yo‘naltirilgan

Ijrochining ko‘rinishini belgilaydigan bir nechta buyruqlar mavjud:
1.turtle.shape ("Style")- tashqi ko‘rinishini o‘zgartiradi
2.turtle.shapesize (m, n) — ijrochining o‘lchamini boshqaradi
3.turtle.tilt (alfa) — ijrochining yo‘nalishini o‘zgartiradi
4.turtle.hideturtle ()- Turtleni yashiradi
5.turtle.showturtle ()- Turtleni ko‘rsatadi.

Shape() funktsiyasidagi "Style" parametri ijrochining ko‘rinishini o‘zgartiradi.
Quyidagi stillar mavjud:
• arrow
• turtle
• circle
• square
• triangle
• classic

Python-da tirtle moduli bo‘lgan dastur to‘g‘ri ishlashi uchun, dastur oxirida har doimikkita buyruqni ro‘yxatdan o‘tkazishingiz kerak:
1.t.screen.exitonclick ()
2.t.screen.mainloop ()
t.screen.exitonclick () buyrug‘i sichqoncha tugmasi bosilganda Python dasturi bunga javob beradi. Agar foydalanuvchi kursor paytida

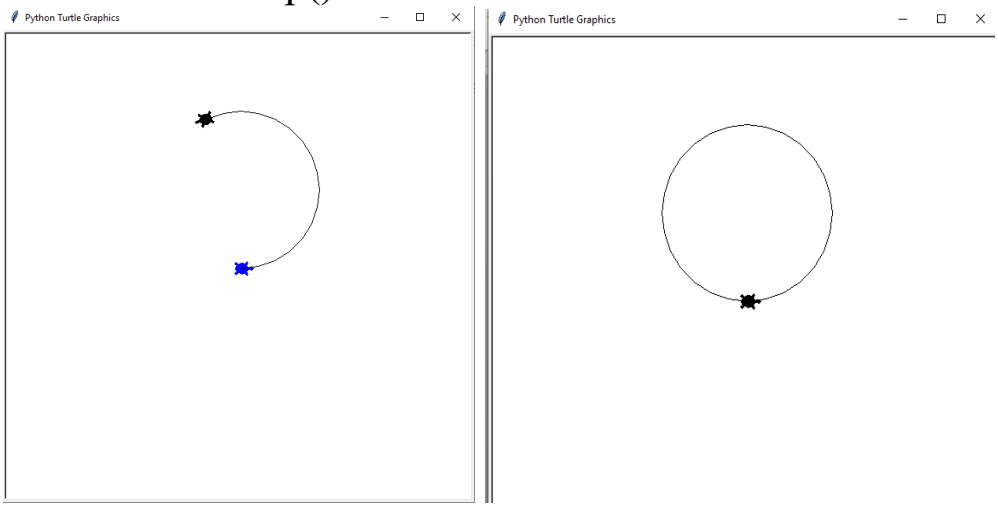
sichqonchaning chap tugmachasini bossa tirtlr modulining grafik oynasida bo‘lsa, oyna yopiladi.
t.screen.mainloop () buyrug‘i dasturning bajarilishini to‘xtatadi.

Dasturni ishga tushirgandan so‘ng, markazda "Turtle" va grafikalar uchun oynani ko‘rasiz

Grafik oynasini ishga tushirganda tirtle koordinatasi oyna markazida joylashganligini ko‘ramiz.Musbat X o‘qi yo‘nalishi chapdan o‘ngga, musbat Y o‘qi yo‘nalishi pastdan yuqoriga aniqlanadi.

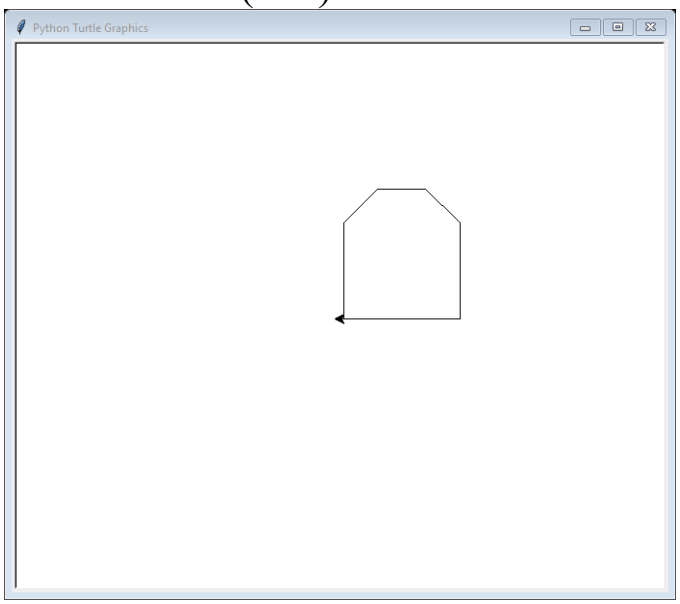
Python-da Turtle modulining grafikasi uchun oynada Turtleni oldinga siljitish uchun t.fd (x) buyrug‘idan foydalaning, bu erda x — Turtle harakatlanadigan piksellar soni. Orqaga harakatlanish uchun t.bk (x) buyrug‘i ishlatiladi. Toshbaqnii ma’lum joyga ko‘chirish uchun koordinatalari t.goto (x, y) foydalaning, bu erda x va y — Turtle harakatlanishi kerak bo‘lgan nuqtaning koordinatalari.

```
from turtle import *
t=Turtle()
t.screen.setup(600,600)
t.color("red")
t.stamp()
t.color("blue")
t.circle(100)
t.screen.exitonclick()
t.screen.mainloop()
```




```
import turtle
big_line = 100
little_line = 50
```

```
angle = 90
turtle.left(angle)
turtle.forward(big_line)
count = 0
while count < 4:
    turtle.right(angle//2)
    if count != 3:
        turtle.forward(little_line)
    else:
        turtle.forward(big_line)
    count = count + 1
turtle.right(90)
turtle.forward(130)
```



Turtlening asosiy buyruqlari ro‘yxati

	Turtlening ko‘chirish buyruqlari
forward(n)/fd(n)	oldinga siljish n qadam (piksel)
backward(n)	orqaga qarab n qadam (piksel)
left(angle)	chap burchak gradusi bo‘yicha burish
right(angle)	o‘ng burchak gradusl bo‘ycha burish
circle(r)	| r | radiusli aylana chizish markazi Turtle chap tomonida joylashgan bo‘ladi agar r> 0 bo‘lsa, va r <0 bo‘lsa markaz o‘ngda joylashadi
circle(r,angle)	| r | radiusli yoyni chizamiz va burchak gradus o‘lchoviga ko‘ra. Yoy chizilganr> 0 bo‘lsa, soat

	strelkasiga teskari , r <0 bo‘lsa, soat strelkasi bo‘ylab
goto(x, y)	”Turtleni” koordinatalari (x, y) nuqtaga joylashtiriladi
	Chizish buyruqlari
down()	qalamni pastga tushiring. Ushbu buyruqdan keyin Turtle har qanday harakat bilaniz qoldirishni boshlaydi
up()	qalamni ko‘taring
width(n)	Turtle izining kengligini n pikselga sozlang
color(s)	Turtle izi ning rangi
Bgcolor(s)	fon rangini s qilib belgilayadi.
fill(f)	bo‘yalgan joylarga chizish uchun ishlatiladi
	Boshqa buyruqlar
reset()	Turtleni asl holiga qaytaradi: ekranni tozalaydi
write(s)	Turtle joylashgan joyda s satrini chop eting
dot(r, color)	nuqta chizadi, bu erda r nuqta radiusi va rang nuqta rangidir
radians()	burchak o‘lchovi (barcha Turtle buyruqlarida) radianda qo‘yiladi
degrees()	burchaklar o‘lchovini (barcha Turtle buyruqlarida) gradusda qo‘ying

Nazorat savollari.

1. Pythonda grafik modullarni snab o‘ting?
2. Grafik modular qanday o‘rnatiladi?
3. Turtle modulining buyruqlar ro‘yhati?

18-19-ma’ruza. Tkinter moduli

Reja:

1. Tkinter modulini o‘rnatish.
2. Tkinter modulining buyruqlar ro‘yxati.
3. Tkinterda vidjetlar.

Tkinter — Python uchun standart GUI kutubxonasi. Python Tkinter bilan birgalikda GUI dasturlarini yaratishning tez va oson usulini taqdim etadi. Tkinter Tk GUI asboblar to‘plamiga kuchli obyektga yo‘naltirilgan interfeysni taqdim etadi.

Pythonda Tkinter modulini 2 usulda import qilish mumkin !!! 1-usul

```
import tkinter
top = tkinter.Tk()
# Vidjetlarni qo‘shish uchun kod bu yerga yoziladi ... top.mainloop ()
```

Bu ikkila yozgan kodimiz quyidagi oynani yaratadi:

2-usul

```
from tkinter import *
top = Tk()
# Vidjetlarni qo‘shish uchun kod bu yerga yoziladi ...
top.mainloop ()
```

Tkinterni import qilishda bu ikki usulning bir-biridan farqi shundaki, agar biz 1-usul ko‘rinishida Tkinter modulini import qilsak: biz dastur kodini yozayotganimizda har bir tkinter metodi oldidan tkinter so‘zini yozishga majburmiz. Ikkinchi usulda esa bunday majburiyatdan halos bo‘lamiz va kodimiz qisqa va aniq ko‘rinishdan iborat bo‘ladi.

TKINTER WIDGETS (“Tkinter vidjetlari”)

Tkinter GUI dasturida ishlatiladigan tugmalar, yorliqlar va matn qutilari kabi turli xil boshqaruv elementlarini taqdim etadi. Ushbu boshqaruv elementlari odatda vidjetlar deb nomlanadi. Hozirda Tkinterda 15 turdagi vidjet mavjud.

№	Operatorlar	Tavsif
1	Button (“Tugma”)	*Button* vidjeti sizning ilovangizdagi tugmalarni ko‘rsatish uchun ishlatiladi.
2	Canvas (“Kanvas”)	*Canvas* vidjeti sizning ilovangizda chiziqlar, tasvirlar, ko‘pburchaklar va to‘rtburchaklar kabi shakllarni chizish uchun ishlatiladi.

3	Checkbutton (“Tekshirish tugmasi”)	*Checkbutton* vidjeti bir qator parametrlarni tasdiqlash qutisi sifatida ko‘rsatish uchun ishlatiladi. Foydalanuvchi bir vaqtning o‘zida bir nechta variantni tanlashi mumkin.
4	Entry (“Kirish)	*Entry* vidjeti foydalanuvchidan qiymatlarni qabul qilish uchun bitta qatorli matn maydonini ko‘rsatish uchun ishlatiladi.
5	Frame (“Kvadrat”)	*Frame* vidjeti boshqa vidjetlarni tartibga solish uchun konteyner vidjeti sifatida ishlatiladi.
6	Label (“Yorliq”)	*Label* vidjeti boshqa vidjetlar uchun bitta qatorli sarlavha bilan ta’minlash uchun ishlatiladi. Unda tasvirlar ham bo‘lishi mumkin.
7	Listbox	*Listbox* vidjeti foydalanuvchiga imkoniyatlar ro‘yxatini taqdim etish uchun ishlatiladi.
8	Menubutton (“Menyu tugmasi”)	*Menubutton* vidjeti sizning ilovangizda menyularni ko‘rsatish uchun ishlatiladi.
9	Menu (“Menyu”)	*Menu* vidjeti foydalanuvchiga turli xil buyruqlar berish uchun ishlatiladi. Ushbu buyruqlar Menubutton-da joylashgan bo‘ladi.
10	Message (“Xabar”)	*Message* vidjeti foydalanuvchidan qiymatlarni qabul qilish uchun ko‘p satrli matn maydonlarini ko‘rsatish uchun ishlatiladi.
11	Radiobutton (“Radion tugmasi”)	*Radiobutton* vidjeti bir qator parametrlarni radio tugmalari sifatida ko‘rsatish uchun ishlatiladi. Bunda foydalanuvchi bir vaqtning o‘zida faqat bitta variantni tanlashi mumkin bo‘ladi.
12	Scale (“Miqyosi”)	*Scale* vidjeti slayder vidjetini ta’minlash uchun ishlatiladi.
13	Scrollbar (“Otkazish paneli”)	*Scrollbar* vidjeti turli xil vidjetlarga, masalan, ro‘yxat qutilariga o‘tish imkoniyatini qo‘shish uchun ishlatiladi.
14	Text (“Matn”)	*Text* vidjeti matnni bir necha qatorda aks ettirish uchun ishlatiladi.
15	Toplevel (“Uchinchi daReja”)	*Toplevel* vidjeti alohida oyna idishini ta’minlash uchun ishlatiladi.
16	Spinbox	*Spinbox* vidjeti - bu standart Tkinter Entry vidjetining bir varianti bo‘lib, u belgilangan qiymatlar orasidan tanlash uchun ishlatilishi mumkin.

17	PanedWindow	*PanedWindow* - bu gorizontal yoki vertikal holda joylashtirilgan har qanday oynani o‘z ichiga oladigan konteyner vidjeti.
18	Labelframe	*Labelframe* - bu oddiy konteyner vidjeti. Uning asosiy maqsadi - oynalarning murakkab joylashuvi uchun oraliq yoki konteyner vazifasini bajarish.
19	Messagebox	*tkMessageBox* — Ushbu modul sizning ilovalaringizda xabarlar qutilarini ko‘rsatish uchun ishlatiladi.

Tkinter Button (“Tugma”)

Button vidjeti Python dasturida tugmachalarni qo‘shish uchun ishlatiladi. Ushbu tugmachalar tugmachalarning maqsadini anglatadigan matn yoki rasmlarni aks ettirishi mumkin. Tugmani bosganingizda avtomatik ravishda chaqiriladigan tugmachaga funktsiya yoki metod biriktirishingiz mumkin.

Sintaksis

Ushbu vidjetni yaratish uchun oddiy sintaksis :

w = Button (master, xossa = qiymat, ...)

Parametrlar

• master — Bu ota-ona oynasini aks ettiradi.

• xossa — Mana bu vidjet uchun eng ko‘p ishlatiladigan xossalarining ro‘yxati. Ushbu parametrlar vergul bilan ajratilgan kalit-qiymat juftlari sifatida ishlatilishi mumkin.

№	*Parametr*	*Tavsif*
1	activebackground (“Faol zamin”)	Tugma kursor ostida bo‘lganda (*tugma bosilganda*) orqa fon rangi.
2	activeforeground (“Faol maydon”)	Tugma kursor ostida bo‘lganida (*tugma bosilganda*) matn rangi.
3	Bd	Chegaraning kengligi piksellarda. Standart 2.
4	Bg	Oddiy fon rangi.
5	Command	Tugma bosilganda chaqiriladigan funktsiya yoki usul.
6	Fg	Oddiy oldingi (matn) rang.
7	font (“shrift”)	Tugma yorlig‘i uchun ishlatiladigan matn shrifti.

8	height (“balandlik”)	Matn satrlaridagi tugmachaning balandligi (matnli tugmalar uchun) yokipiksellar (rasmlar uchun).
9	highlightcolor (“ochrang”)	Vidjet fokusga ega bo‘lganda fokusning rangi ta’kidlanadi.
10	image (“rasm”)	Tugmachada ko‘rsatiladigan rasm (matn o‘rniga).
11	justify (“oqlash”)	Bir nechta matn satrlarini qanday ko‘rsatish kerak. Bu justify parametric orqali amalga oshiriladi: har bir satrni chap tomonga o‘rnatish uchun LEFT; Ularni markazlashtirish uchun CENTER; yoki o‘ngga joylashtirish uchun RIGHT.
12	padx	Matnning chap va o‘ng tomonlariga qo‘shimcha to‘ldirish.
13	pady	Matnning yuqorisida va ostida qo‘shimcha to‘ldirish.
14	relief	Relief chegara turini belgilaydi. Ba’zi qiymatlari SUNKEN, RAISED, GROOVE va RIDGE.
15	state (“davlat”)	Ushbu parametr tugmani activ holatda ishlashni boshlashi va unga bo‘lgan murojaatni cheklash uchun ishlatiladi. state = DISABLED holatida tugmaga murojaat qilib bo‘lmaydi. state = ACTIVE holatida esa tugma unga murojaat etilgan holatida bo‘ladi. Tugmaga yana bir bor bosilganda esa tugma yana o‘zining standart holatiga o‘tadi. Standart holatda esa state=NORMAL bo‘ladi.
16	underline (“tagiga chizish”)	Odatiy qiymati -1, ya’ni tugmachadagi matnning biron bir belgisi ostiga chizilmaydi. Agar manfiy bo‘lmasa, tegishli matn belgisi ostiga chiziladi.
17	width (“kenglik”)	Tugmaning kengligi harflar bilan (agar matn ko‘rsatilsa) yoki piksel (agar rasmko‘rsatilsa).
18	wraplength (“o‘rash uzunligi”)	Agar bu qiymat ijobiy raqamga o‘rnatilsa, matn satrlari shu uzunlikka mosravishda o‘raladi.

Metodlari:

Ushbu vidjet uchun quyidagi keng tarqalgan metodlar qo‘llaniladi :

№	*Metod*	*Tavsif*
1	flash (“chaqmoq”)	Tugmani faol va normal ranglar o‘rtasida bir necha marta yonib-o‘chishiga olib keladi. Tugmani asl holatida qoldiradi. Agar tugma o‘chirilgan bo‘lsa, e’tiborga olinmaydi.
2	invoke() (“chaqirish”)	Tugmachani qayta chaqirishni chaqiradi va bu funktsiya qaytib kelgan narsani qaytaradi. Agar tugma o‘chirilgan bo‘lsa yoki qayta murojaat bo‘lmasa, ta’sir qilmaydi.

Tkinter Canvas

Canvas - bu rasmlar yoki boshqa murakkab sxemalarni chizish uchun mo‘ljallangan to‘rtburchak maydon. Siz canvasga grafikalar, matnlar, vidjetlar yoki ramkalarni joylashtirishingiz mumkin.

Sintaksis

Ushbu vidjetni yaratish uchun oddiy sintaksis:

w = Canvas (master, xossa = qiymat, ...)

Parametrlar

master - Bu ota-ona oynasini aks ettiradi.

xossa - Mana bu vidjet uchun eng ko‘p ishlatiladigan xossalarining ro‘yxati. Ushbuparametrlar vergul bilan ajratilgan kalit-qiymat juftlari sifatida ishlatilishi mumkin.

№	*Parametr*	*Tavsif*
1	bd	Chegaraning kengligi piksellarda. Standart 2.
2	bg	Oddiy fon rangi.
3	confine	Agar True (rost) bo‘lsa (standart), canvasni scrollregion tashqarisiga o‘tkazib bo‘lmaydi.
4	cursor	Canvasda o‘q, aylana, nuqta va hokazo kabi ishlatiladigan kursor.

5	height	Y o‘lchovidagi *canvas* ning o‘lchami.
6	higlightcolor	Fokusni ta’kidlashda ko‘rsatilgan rang.
7	Relief	Relief chegara turini belgilaydi. Ba’zi qiymatlar SUNKEN, RAISED,GROOVE va RIDGE.
8	scrollregion	*Canvas* ni qanchalik katta maydonga siljitish mumkinligini belgilaydigangrafika (w, n, e, s), bu erda w chap tomon, n yuqori, e o‘ng tomon va s pastki qism.
9	Width	X o‘lchamdagi *canvas* ning o‘lchami.
	xscrollincrement	Agar siz ushbu parametrni biron bir ijobiy o‘lchamga o‘rnatgan bo‘lsangiz,*canvas* faqat shu masofaning ko‘paytmalariga joylashtirilishi mumkin va bu qiymat aylantirish birliklari orqali o‘tish uchun ishlatiladi, masalan, foydalanuvchi aylantirish satrining uchlaridagi o‘qlarni chertganida.
	xscrollcommand	Agar *canvas* o‘ralgan bo‘lsa, bu atribut gorizontal aylantirish panelining .set() usuli bo‘lishi kerak.
	yscrollincrement	Xscrollincrement kabi ishlaydi, lekin vertikal harakatni boshqaradi.
	yscrollcommand	Agar canvas o‘ralgan bo‘lsa, bu atribut vertikal aylantirish panelining .set ()usuli bo‘lishi kerak.

Canvas vidjeti quyidagi standart elementlarni qo‘llab-quvvatlashi mumkin:
arc - akkord, pieslice yoki oddiy kamon bo‘lishi mumkin bo‘lgan yoy elementini yaratadi.

```
coord = 10, 50, 240, 210

arc = canvas.create_arc (coord, start = 0, extend = 150, fill = "blue")
```

image - BitmapImage yoki PhotoImage sinflarining misoli bo‘lishi mumkin bo‘lgan rasmelementini yaratadi.

```
fayl nomi = PhotoImage (file = "sunshine.gif")

image = canvas.create_image (50, 50, anchor = NE, image = fayl nomi)
```

line - satr elementini yaratadi.
line = canvas.create_line (x0, y0, x1, y1, ..., xn, yn, parametrlar)
oval - berilgan koordinatalarda aylana yoki ellips hosil qiladi. Buning uchun ikki juft koordinatalar kerak; oval uchun cheklovchi to‘rtburchakning yuqori chap va pastki o‘ng burchaklari.
oval = canvas.create_oval (x0, y0, x1, y1, parametrlar)
polygon - Kamida uchta tepalikka ega bo‘lishi kerak bo‘lgan ko‘pburchak elementni yaratadi.
oval = canvas.create_polygon (x0, y0, x1, y1, ... xn, yn, parametrlar)
Tkinter Checkbutton

Checkbutton vidjeti foydalanuvchiga o‘tish tugmachalari sifatida bir qator variantlarni ko‘rsatish uchun ishlatiladi. Keyin foydalanuvchi har bir parametrga mos keladigan tugmani bosish orqali biryoki bir nechta variantni tanlashi mumkin. Matn o‘rniga rasmlarni ham ko‘rsatishingiz mumkin.

Sintaksis

Ushbu vidjetni yaratish uchun oddiy sintaksis :
w = Checkbutton (master, xossa=qiymat, ...)Parametrlar
master - Bu ota-ona oynasini aks ettiradi.
xossa - Mana bu vidjet uchun eng ko‘p ishlatiladigan xossalarining ro‘yxati. Ushbuparametrlar vergul bilan ajratilgan kalit-qiymat juftlari sifatida ishlatilishi mumkin.

№	*Option*	*Tavsif*
1	activebackground (“Faol zamin”)	Tekshirish tugmasi kursor ostida bo‘lganda fon rangi.
2	activeforeground (“Faol maydon”)	Tekshirish tugmasi kursor ostida bo‘lganidan oldingi rangi
3	Bg	Oddiy fon rangi label va indikator orqasida ko‘rsatiladi.
4	Bitmap	Bitta rangli tasvirni tugmachada aks ettirish uchun.
5	Bd	Ko‘rsatkich atrofidagi chegara kattaligi. Standart - 2 piksel.

6	command	Foydalanuvchi har safar ushbu tugmachaning holatini o‘zgartirganda chaqiriladiganprotsedura.
7	Cursor	Agar siz ushbu parametrni kursor nomiga o‘rnatgan bo‘lsangiz (o‘q, nuqta va boshqalar), sichqoncha kursori tugma tugagandan so‘ng shu naqshga o‘zgaradi.
8	disabledforegroun d	O‘chirilgan tugmachani ko‘rsatish uchun ishlatiladigan oldingi rang. Sukut bo‘yicha oldingi rangning stippled versiyasi.
9	Font	Matn uchun ishlatiladigan shrift.
10	Fg	Matnni ko‘rsatish uchun ishlatiladigan rang.
11	Height	Tekshirish tugmachasidagi satrlar soni. Standart - 1.
12	highlightcolor	Tugma fokusga ega bo‘lganda fokusning rangi ta’kidlanadi.
13	Image	Tugmachada grafik tasvirni ko‘rsatish uchun ishlatiladi.
14	Justify	Agar matnda bir nechta satr mavjud bo‘lsa, ushbu parametr matnning qanday asoslanishini boshqaradi: CENTER, LEFT yoki RIGHT.
15	Offvalue	Odatda, tekshiruv tugmasi bilan bog‘liq boshqaruv o‘zgaruvchisi o‘chirilganda (o‘chirilganda) 0 ga o‘rnatiladi. O‘chirish holati uchun muqobil qiymatni ushbu qiymatga qiymatni belgilash orqali taqdim etishingiz mumkin.
16	Onvalue	Odatda, tasdiqlash tugmachasi bilan bog‘liq boshqaruv o‘zgaruvchisi o‘rnatilganda(yoqilganda) 1 ga o‘rnatiladi. On holatini ushbu qiymatga o‘rnatib, muqobil qiymatni taqdim etishingiz mumkin.
17	Padx	Tekshirish tugmasi va matnning chap va o‘ng tomoniga qancha joy qoldirish kerak.Odatiy - 1 piksel.
18	Pady	Tekshirish tugmasi va matnning yuqorisida va pastida qancha joy qoldirish kerak.Odatiy - 1 piksel.
19	Relief	Odatiy qiymati, relief = FLAT bilan, tugma uning fonidan ajralib turmaydi. Ushbu parametrni boshqa har qanday uslubga o‘rnatishingiz mumkin.
20	Selectcolor	Belgilangan tugmachaning rangi. Odatiy qiymati selectcolor = "red".

21	Selectimage	Agar siz ushbu parametrni rasmga o‘rnatgan bo‘lsangiz, u o‘rnatilgandan so‘ngushbu rasm tasdiqlash tugmachasida paydo bo‘ladi.
22	State	Odatiy holat state = NORMAL, ammo boshqaruvni kul rangga aylantirish va uni javobsiz holatga keltirish uchun state = DISABLED dan foydalanishingiz mumkin. Agar kursor hozirda tasdiqlash tugmachasi ustida bo‘lsa, holat ACTIVE.
23	Text	Belgilash tugmasi yonida ko‘rsatilgan yorliq. Bir nechta matn satrlarini ko‘rsatishuchun yangi qatorlardan ("\ n") foydalaning.
24	Underline	Standart qiymat -1 bo‘lsa, matn yorlig‘ining biron bir belgisi ostiga chizilmaydi. Ushbu belgini tagiga chizish uchun ushbu parametrni matndagi belgi indeksiga o‘rnating (noldan hisoblang).
25	Variable	Tekshirish tugmachasining joriy holatini kuzatadigan boshqaruv o‘zgaruvchisi. Odatda bu o‘zgaruvchi IntVar, 0 esa tozalangan va 1 degani o‘rnatilgan degan ma’noni anglatadi, lekin yuqoridagi qiymat va qiymat parametrlariga qarang.
26	Width	Tekshirish tugmachasining standart kengligi ko‘rsatilgan rasm yoki matnning o‘lchamiga qarab belgilanadi. Ushbu parametrni bir qator belgilarga o‘rnatishingiz mumkin, shunda tasdiqlash tugmachasida har doim shuncha belgilar uchun joy bo‘ladi.
27	Wraplength	Odatda, chiziqlar o‘ralgan emas. Siz ushbu parametrni bir qator belgilarga o‘rnatishingiz mumkin va barcha satrlar bu raqamdan oshib ketmaydigan qismlarga bo‘linadi.

Metodlari:

№	*Metod*	*Tavsif*
1	deselect() (“bekor qilish”)	Tekshirish tugmachasini tozalaydi (o‘chiradi).
2	flash() (“chaqmoq”)	Faol va normal ranglar o‘rtasida bir necha marta yonib-o‘chib turadi, lekin uni qanday boshlagan bo‘lsa, shunday qoldiradi.
3	invoke() (“chaqirish”)	Xuddi shu amallarni bajarish uchun ushbu usulni chaqirishingiz mumkin agar foydalanuvchi o‘z holatini o‘zgartirish uchun tugmani bosgan bo‘lsa paydobo‘ladi.

4	select() (“tanlash”)	Tekshirish tugmachasini o‘rnatadi (yoqadi).
5	toggle() (“almashtirish”)	O‘rnatilgan bo‘lsa, tugmachani tozalaydi, agar o‘chirilgan bo‘lsa, uni o‘rnatadi.

Tkinter Entry (“Kirish”)

Entry vidjeti foydalanuvchidan bitta qatorli matn satrlarini qabul qilish uchun ishlatiladi.

Agar tahrirlash mumkin bo‘lgan bir nechta matn satrlarini namoyish qilmoqchi bo‘lsangiz, u holda *Textwidget*-dan foydalanishingiz kerak.

Agar siz foydalanuvchi tomonidan o‘zgartirilishi mumkin bo‘lmagan bir yoki bir nechta matnsatrini ko‘rsatishni xohlasangiz, u holda *Label* vidjetidan foydalanishingiz kerak.

Sintaksis

Ushbu vidjetni yaratish uchun oddiy sintaksis:

w = Entry (master, xossa=qiymat, ...)

Parametrlar

master - Bu ota-ona oynasini aks ettiradi.

xossa - Mana bu vidjet uchun eng ko‘p ishlatiladigan xossalarining ro‘yxati. Ushbuparametrlar vergul bilan ajratilgan kalit-qiymat juftlari sifatida ishlatilishi mumkin.

№	*Option*	*Tavsif*
1	Bg	Oddiy fon rangi yorliq va indikator orqasida ko‘rsatiladi.
2	Bd	Ko‘rsatkich atrofidagi chegara kattaligi. Standart - 2 piksel.
3	Command	Foydalanuvchi har safar ushbu tugmachaning holatini o‘zgartirgandachaqiriladigan protsedura.
4	Cursor	Agar siz ushbu parametrni kursor nomiga o‘rnatgan bo‘lsangiz (arrow – “o‘q”,dot – “nuqta” va boshqalar), sichqoncha kursori tugma tugagandan so‘ng shu naqshga o‘zgaradi.
5	Font	Matn uchun ishlatiladigan shrift.
6	exportselection	Odatiy bo‘lib, agar siz *Entry* vidjeti ichidagi matnni tanlasangiz, u avtomatik ravishda buferga eksport qilinadi. Ushbu eksportni oldini olish uchun

		exportelection = 0 dan foydalaning.
7	Fg	Matnni ko‘rsatish uchun ishlatiladigan rang.
8	highlightcolor	Tugma fokusga ega bo‘lganda fokusning rangi ta’kidlanadi.
9	Justify	Agar matnda bir nechta satr mavjud bo‘lsa, ushbu parametr matnning qanday asoslanishini boshqaradi: CENTER, LEFT yoki RIGHT.
10	Relief	Odatiy qiymati, yordam = FLAT bilan, tugma uning fonidan ajralib turmaydi. Ushbu parametrni boshqa har qanday uslubga o‘rnatishingiz mumkin
11	selectbackground	Tanlangan matnni aks ettirish uchun fon rangi.
12	selectborderwidth	Tanlangan matn atrofida foydalaniladigan chegara kengligi. Odatiy bo‘lib, bittapiksel.
13	selectforeground	Tanlangan matnning oldingi (matn) rangi.
14	show (“korsatish”)	Odatda foydalanuvchi kiritgan belgilar yozuvda paydo bo‘ladi. Siz bu belgilarni*password* bilan yashirishingiz mumkin. Buning uchun show = "*" dan foydalaning. Bu parameter sizga har bir belgini yashirgan holda yulduzcha ko‘rinishida aks ettirib beradi.
15	State	Odatiy holat state = NORMAL, ammo boshqaruvni kul rangga aylantirish va uni javobsiz holatga keltirish uchun state = DISABLED dan foydalanishingiz mumkin. Agar kursor hozirda tasdiqlash tugmachasi ustida bo‘lsa, state=ACTIVE.
16	textvariable	*Entr*y vidjetidan joriy matnni olish uchun ushbu parametrni StringVar sinfining misoliga o‘rnatishingiz kerak.
17	Width	Tekshirish tugmachasining standart kengligi ko‘rsatilgan rasm yoki matnning o‘lchamiga qarab belgilanadi. Ushbu parametrni bir qator belgilarga o‘rnatishingiz mumkin, shunda tasdiqlash tugmachasida har doim shuncha belgilar uchun joy bo‘ladi.
18	xscrollcommand	Agar foydalanuvchilar tez-tez vidjetning ekrandagi hajmidan ko‘proq matn kiritadilar deb o‘ylasangiz, kirish vidjetingizni aylantirish paneliga bog‘lashingiz mumkin.

Metodlari

№	*Metodlar*	*Tavsif*
1	delete (first, last=None)	Vidjetdagi belgilar birinchi indeksdagi belgidan boshlanib, oxirgi pozitsiyadagi belgini qo‘shmasdan o‘chiriladi. Agar ikkinchi argument tashlansa, faqat birinchi pozitsiyadagi bitta belgi o‘chiriladi.
2	get() (“olish”)	Yozuvning joriy matnini satr sifatida qaytaradi.
3	icunsor (index)	Kursorni berilgan indeksdagi belgi oldidan o‘rnating.
4	Index	Belgilangan indeksdagi belgi chap tomondagi ko‘rinadigan belgi bo‘lishi uchun yozuv tarkibini o‘zgartiring. Agar matn to‘liq yozuvgato‘g‘ri keladigan bo‘lsa, ta’sir qilmaydi.
5	insert (index, s)	Belgidan oldin s qatorini berilgan indeksga kiritadi.
6	select_adjust (index)	Ushbu usul tanlovning belgilangan indeksdagi belgini o‘z ichiga olganligiga ishonch hosil qilish uchun ishlatiladi.
7	select_clear()	Tanlovni tozalaydi. Agar hozirda tanlov bo‘lmasa, ta’sir qilmaydi.
8	select_form (index)	*ANCHOR* indeks o‘rnini indeks bo‘yicha tanlangan belgiga o‘rnatadiva shu belgini tanlaydi.
9	select_present ()	Agar tanlov bo‘lsa, true qiymatini qaytaradi, aks holda falseqiymatini qaytaradi.
10	select_range (start, end)	Tanlovni dastur nazorati ostida o‘rnatadi. Matnni boshlang‘ich indeksidan boshlab, indeksdagi belgini qo‘shmasdan tanlaydi. Boshlanish holati oxirgi pozitsiyadan oldin bo‘lishi kerak.
11	select_to (index)	Barcha matnni ANCHOR pozitsiyasidan shu indeksdagi belginiqo‘shmasdan tanlaydi.
12	xview (index)	Ushbu usul *Entry* vidjetini gorizontal aylantirish paneliga bog‘lashdafoydalidir.
13	xview_scroll (number, what)	*Entry* ni gorizontal ravishda aylantirish uchun foydalaniladi. Belgilar kengligi bo‘yicha o‘tish uchun UNITS yoki PAGES, *Entry* vidjetining kattaligi bo‘yicha qismlarga o‘tish uchun nima argumentbo‘lishi kerak. Chapdan o‘ngga siljitish uchun raqam ijobiy, o‘ngdanchapga o‘tish uchun salbiy.

Tkinter Frame

Frame vidjeti boshqa vidjetlarni qandaydir tarzda do‘stona tarzda guruhlash va tartibga solish jarayonida juda muhimdir. U boshqa vidjetlarning joylashishini tartibga solish uchun javob beradigan konteyner kabi ishlaydi. U tartibni tashkil qilish va ushbu vidjetlarning to‘ldirilishini ta’minlash uchun ekrandagi to‘rtburchak maydonlardan foydalanadi. Kadr, shuningdek, murakkabvidjetlarni amalga oshirish uchun poydevor sinfi sifatida ham foydalanish mumkin.

Sintaksis

Ushbu vidjetni yaratish uchun oddiy sintaksis :
w = Frame (master, xossa=qiymat, ...)

Parametrlar

master - Bu ota-ona oynasini aks ettiradi.
xossa - Mana bu vidjet uchun eng ko‘p ishlatiladigan xossalarining ro‘yxati. Ushbuparametrlar vergul bilan ajratilgan kalit-qiymat juftlari sifatida ishlatilishi mumkin.

№	*Parametrlar*	*Tavsif*
1	Bg	Oddiy fon rangi yorliq va indikator orqasida ko‘rsatiladi.
2	Bd	Ko‘rsatkich atrofidagi chegara kattaligi. Standart - 2 piksel.
3	Cursor	Agar siz ushbu parametrni kursor nomiga o‘rnatgan bo‘lsangiz (arrow – “o‘q”, dot – “nuqta” va boshqalar), sichqoncha kursori tugma tugagandan so‘ng shu naqshga o‘zgaradi.
4	Height	Yangi ramkaning vertikal o‘lchamlari.
5	highlightbackground	Fokus bo‘lmasa, fokusning rangi ta’kidlanadi.
6	Highlightcolor	Agar ramka fokusga ega bo‘lsa, u fokusda ko‘rsatilgan rang.
7	highlightthickness	Fokusning qalinligi.

8	Relief	Odatiy qiymati, relief = FLAT bilan, tugma uning fonidan ajralib turmaydi. Ushbu parametrni boshqa har qanday uslubga o‘rnatishingiz mumkin
9	Width	Tekshirish tugmachasining standart kengligi ko‘rsatilgan rasm yoki matnning o‘lchamiga qarab belgilanadi. Ushbu parametrni bir qator belgilarga o‘rnatishingiz mumkin, shunda tasdiqlash tugmachasida har doim shuncha belgilar uchun joy bo‘ladi.

Tkinter Label (“Yorliq”)

Ushbu vidjet matn yoki rasmlarni joylashtirishingiz mumkin bo‘lgan ekran oynasini amalga oshiradi. Ushbu vidjet ko‘rsatadigan matnni xohlagan vaqtda yangilash mumkin.

Bundan tashqari, matnning bir qismini ostiga chizish (masalan, klaviatura yorlig‘ini aniqlash) va matnni bir nechta satrlar bo‘ylab uzatish mumkin.

Sintaksis

Ushbu vidjetni yaratish uchun oddiy sintaksis -
w = Label (master, xossa=qiymat, ...)

Parametrlar

master - Bu ota-ona oynasini aks ettiradi.
xossa - Mana bu vidjet uchun eng ko‘p ishlatiladigan xossalarining ro‘yxati. Ushbuparametrlar vergul bilan ajratilgan kalit-qiymat juftlari sifatida ishlatilishi mumkin.

№	*Option*	*Tavsif*
1	anchor (“langar”)	Ushbu parametr, agar vidjetda matn ehtiyojidan ko‘proq joy bo‘lsa, matn qayerda joylashishini boshqaradi. Odatiy bo‘lib, matnni mavjud bo‘shliqda markazlashtiradigan anchor = CENTER.
2	Bg	Oddiy fon rangi yorliq va indikator orqasida ko‘rsatiladi.

3	Bitmap	Ushbu parametrni bitmap yoki rasm ob’ektiga tenglashtiring, shunda yorliq o‘sha grafikniaks ettiradi.
4	Bd	Ko‘rsatkich atrofidagi chegara kattaligi. Standart - 2 piksel.
5	Cursor	Agar siz ushbu parametrni kursor nomiga o‘rnatgan bo‘lsangiz (arrow – “o‘q”, dot –“nuqta” va boshqalar), sichqoncha kursori tugma tugagandan so‘ng shu naqshga o‘zgaradi.
6	Font	Agar siz ushbu yorliqda matnni namoyish qilsangiz (matn yoki matn o‘zgaruvchan variant bilan birga, shrift opsiyasi ushbu matn qaysi shriftda ko‘rsatilishini belgilaydi.
7	Fg	Agar siz ushbu yorliqda matn yoki bitmap ko‘rsatayotgan bo‘lsangiz, ushbu parametrmatn rangini belgilaydi. Agar siz bitmap ko‘rsatayotgan bo‘lsangiz, bu bitmapdagi 1- bitlar holatida paydo bo‘ladigan rang.
8	Height	Yangi ramkaning vertikal o‘lchamlari.
9	Image	Yorliq vidjetida statik tasvirni ko‘rsatish uchun ushbu parametrni rasm ob’ektigao‘rnating.
10	Justify	Matnning bir nechta satrlari bir-biriga nisbatan qanday tekislanishini belgilaydi: chap tomonga *LEF*T, markazlashtirilgan uchun markaziy (standart) uchun *CENTER* yokio‘ngga tartiblanishi uchun *RIGHT*.
11	Padx	Vidjet ichidagi matnning chap va o‘ng qismiga qo‘shimcha joy qo‘shildi. Standart - 1.
12	Pady	Vidjet ichidagi matnning yuqorisida va ostiga qo‘shimcha joy qo‘shildi. Standart - 1.
13	Relief	*Label* atrofidagi dekorativ chegara ko‘rinishini belgilaydi. Odatiy qiymati FLAT;
14	Text	*Label* vidjetida bir yoki bir nechta satrlarni ko‘rsatish uchun ushbu parametrni matn o‘z ichiga olgan qatorga o‘rnating. Ichki yangi qatorlar ("\ n") qatorni to‘xtatishga majbur qiladi.
15	textvariable	*Label* vidjetida ko‘rsatilgan matnni StringVar sinfidagi boshqaruv o‘zgaruvchisiga saqlash uchun ushbu parametrni ushbu o‘zgaruvchiga o‘rnating.
16	underline	Ushbu parametrni n ga o‘rnatib, matnning n harfi ostida 0 dan boshlab, pastki chizig‘ini (_) ko‘rsatishingiz mumkin. Sukut bo‘yicha chiziq underline= -1, ya’ni pastki chiziqchizilmaydi.

17	width	Belgilardagi *Label* ning kengligi (piksel emas!). Agar ushbu parametr o‘rnatilmagan bo‘lsa, *Label* uning tarkibiga mos keladigan hajmga ega bo‘ladi.
18	wraplength	Ushbu parametrni kerakli raqamga o‘rnatib, har bir satrdagi belgilar sonini cheklashingiz mumkin. Standart qiymat 0, chiziqlar faqat yangi satrlarda buzilishini anglatadi.

Tkinter listbox (“Listlar qutisi”)

Listbox vidjeti foydalanuvchi bir nechta narsalarni tanlashi mumkin bo‘lgan elementlarningro‘yxatini ko‘rsatish uchun ishlatiladi.

Sintaksis

Ushbu vidjetni yaratish uchun oddiy sintaksis:
w = Listbox (master, xossa=qiymat, ...)

Parametrlar

master - Bu ota-ona oynasini aks ettiradi.
xossa - Mana bu vidjet uchun eng ko‘p ishlatiladigan xossalarining ro‘yxati. Ushbuparametrlar vergul bilan ajratilgan kalit-qiymat juftlari sifatida ishlatilishi mumkin.

№	*Option*	*Tavsif*
1	Bg	Oddiy fon rangi yorliq va indikator orqasida ko‘rsatiladi.
2	Bd	Ko‘rsatkich atrofidagi chegara kattaligi. Standart - 2 piksel.
3	Cursor	Sichqoncha ro‘yxat qutisi ustida turganida paydo bo‘ladigan kursor.
4	Font	Ro‘yxat qutisidagi matn uchun ishlatiladigan shrift.
5	Fg	Ro‘yxat qutisidagi matn uchun ishlatiladigan rang.
6	Height	Qatorlar soni (piksel emas!) Ro‘yxat oynasida ko‘rsatilgan. Standart 10 gateng.
7	highlightcolor	Vidjet fokusga ega bo‘lganda, diqqat markazida ko‘rsatilgan rang.

8	highlightthickness	Fokusning qalinligi.
9	Relief	Uch o‘lchovli chegara soyalash effektlarini tanlaydi. Odatiy holatdaSUNKEN bo‘ladi.
10	selectbackground	Tanlangan matnni aks ettirish uchun fon rangi.
11	selectmode	Qancha elementni tanlash mumkinligini va sichqoncha sudrab chiqarilishitanlovga qanday ta’sir qilishini aniqlaydi -
		BROWSE — Odatda, ro‘yxat qutisidan faqat bitta qatorni tanlashingizmumkin. Agar siz biror elementni bosib, keyin boshqa qatorga tortibqo‘ysangiz, tanlov sichqonchani kuzatib boradi. Bu sukut bo‘yicha. SINGLE - Siz faqat bitta qatorni tanlashingiz mumkin va sichqonchanisudrab olib borolmaysiz. MULTIPLE — Siz bir vaqtning o‘zida istalgan qatorni tanlashingiz mumkin. Har qanday satrni bosish tanlangan yoki tanlanmaganliginio‘zgartiradi. EXTENDED - birinchi qatorni bosish va oxirgi qatorga tortish orqali birvaqtning o‘zida har qanday qo‘shni qator guruhini tanlashingiz mumkin.
12	Width	Belgilarda vidjetning kengligi. Sukut bo‘yicha 20 ga teng.
13	xscrollcommand	Agar siz foydalanuvchiga ro‘yxat qutisini gorizontal ravishda aylantirishga ruxsat berishni xohlasangiz, siz ro‘yxat qutisi vidjetini gorizontal o‘tish satriga bog‘lashingiz mumkin.
14	yscrollcommand	Agar siz foydalanuvchiga ro‘yxat qutisini vertikal ravishda aylantirishga ruxsat berishni xohlasangiz, ro‘yxat qutisi vidjetini vertikal o‘tish satrigabog‘lashingiz mumkin.

Metodlari:

Listbox ob’ektlaridagi metodlarga quyidagilar kiradi:

№	*Option*	*Tavsif*
1	active (index)	Berilgan indeks bo‘yicha chiziqni tanlaydi.
2	curselection()	Tanlangan element yoki elementlarning satr raqamlarini o‘z ichiga olgan katakchani qaytaradi, 0 dan sanaydi. Agar hech narsa tanlanmasa, bo‘sh katakkaqaytadi.
3	delete (first, last=None)	Indekslari [birinchi, oxirgi] oralig‘ida bo‘lgan qatorlarni o‘chiradi. Agar ikkinchi argument tashlansa, birinchi indeksli bitta satr o‘chiriladi.
4	get (first, last=None)	Ilkdan oxirigacha, shu jumladan, indekslari bo‘lgan satrlar matni o‘z ichiga olgan karnayni qaytaradi. Agar ikkinchi argument o‘tkazib yuborilgan bo‘lsa, birinchi qatorga eng yaqin satr matni qaytariladi.
5	index (i)	Iloji bo‘lsa, ro‘yxat oynasining ko‘rinadigan qismini shunday joylashtiring,shunda indeks o‘z ichiga olgan satr vidjetning yuqori qismida joylashgan bo‘ladi.
6	insert (index, *elements)	Ro‘yxat katakchasiga indeks bilan belgilangan qatordan oldin bir yoki bir nechta yangi qatorlarni kiriting. Ro‘yxat oxiriga yangi qatorlar qo‘shishni istasangiz, birinchi dalil sifatida END dan foydalaning.
7	nearest (“eng yaqin”)	Y-koordinatali y ga eng yaqin ko‘rinadigan chiziq indeksini ro‘yxat qutisividjetiga nisbatan qaytaring.
8	see (index)	Ro‘yxat maydonining o‘rnini indeks bo‘yicha ko‘rsatilgan satr ko‘rinadigan qilibsozlang.
9	size()	Ro‘yxat qatoridagi qatorlar sonini qaytaradi.
10	xview()	Ro‘yxat qutisini gorizontal ravishda aylantiriladigan qilish uchun, ushbu usul bilan bog‘liq gorizontal aylantirish panelining buyruq parametrini o‘rnating.
11	xview_moveto (fraction)	Ro‘yxat qutisini shunday siljiting, shunda uning eng uzun satrining kengligining chap qismi ro‘yxat qutisining chap tomonidan tashqarida bo‘ladi. Fraktsiya [0,1]oralig‘ida joylashgan.

12	xview_scroll (number,what)	Ro‘yxat qutisini gorizontal ravishda aylantiradi. Qaysi argument uchun belgilar bilan siljitish uchun UNITS yoki sahifalar bo‘yicha, ya’ni ro‘yxat qutisi kengligi bo‘yicha PAGES-dan foydalaning. Raqam argumenti nechta aylantirish kerakligini aytadi.
13	yview	Ro‘yxat qutisini vertikal ravishda aylantiriladigan qilish uchun, ushbu usul bilan bog‘liq vertikal aylantirish panelining buyruq parametrini o‘rnating.
14	yview_moveto (fraction)	Ro‘yxat oynasini eng uzun chiziq kengligining yuqori qismi ro‘yxat qutisining chap qismidan tashqarida bo‘lishi uchun aylantiring. Fraktsiya [0,1] oralig‘idajoylashgan.
15	yview_scroll (number,what)	Ro‘yxat qutisini vertikal ravishda aylantiradi. Qaysi dalil uchun satrlar bo‘yichao‘tish uchun UNITS-dan foydalaning yoki sahifalar bo‘yicha, ya’ni ro‘yxat qutisining balandligi bo‘yicha PAGES-dan foydalaning. Raqam argumenti nechta aylantirish kerakligini aytadi.

Tkinter Menubutton (“Menyu tugmasi”)

Menubutton - bu ekranda doimo saqlanib turadigan ochiladigan menyu qismidir. Har bir *menubutton* Menyu vidjeti bilan bog‘langan bo‘lib, foydalanuvchi uni bosganida ushbu menyutugmachasi uchun tanlovni aks ettirishi mumkin.

Sintaksis

Ushbu vidjetni yaratish uchun oddiy sintaksis :
w = Menubutton (master, xossa=qiymat, ...)

Parametrlar

master - Bu ota-ona oynasini aks ettiradi.
xossa - Mana bu vidjet uchun eng ko‘p ishlatiladigan xossalarining ro‘yxati. Ushbuparametrlar vergul bilan ajratilgan kalit-qiymat juftlari sifatida ishlatilishi mumkin.

№	*Option*	*Tavsif*
1	activebackground	Sichqoncha menyu tugmasi ustiga qo‘yilganda fon rangi.
2	activeforeground	Sichqoncha menyu tugmachasi ustida turganida oldingi rang.
3	anchor	Ushbu parametr, agar vidjetda matn ehtiyojidan ko‘proq joy bo‘lsa, matn qayerdajoylashishini boshqaradi. Odatiy bo‘lib, matnni markazlashtiradigan anchor = CENTER.
4	bg	Oddiy fon rangi yorliq va indikator orqasida ko‘rsatiladi.
5	bitmap	Menyu tugmachasida bitmapni ko‘rsatish uchun ushbu parametrni bitmap nomigao‘rnating
6	bd	Ko‘rsatkich atrofidagi chegara kattaligi. Standart - 2 piksel.
7	cursor	Sichqoncha ushbu menyu tugmasi ustida turganida paydo bo‘ladigan kursor.
8	direction	Menyuni tugmachaning chap tomonida ko‘rsatish uchun yo‘nalishni = LEFT-ni o‘rnating; tugmachaning o‘ng tomonidagi menyuni aks ettirish uchun yo‘nalish =RIGHT dan foydalaning; yoki menyuni tugmachaning yuqorisiga qo‘yish uchundirection = ’above’ dan foydalaning.
9	disabledforegroun d	O‘chirilganda ushbu menyu tugmachasida oldingi rang ko‘rsatilgan.
10	Fg	Sichqoncha menyu tugmachasi yonida bo‘lmaganida oldingi rang.
11	height	Matn satrlaridagi menyu tugmachasining balandligi (piksel emas!). Odatiy bo‘lib menyu tugmachasining o‘lchamiga uning tarkibiga mos keladi.
12	highlightcolor	Vidjet fokusga ega bo‘lganda, diqqat markazida ko‘rsatilgan rang.
13	image	Ushbu menyu tugmachasida rasmni ko‘rsatish uchun
14	justify	Ushbu parametr, matn menyu tugmachasini to‘ldirmasa, matn qaerda joylashganligini boshqaradi: matnni chap tomonida oqlash uchun justify = LEFTdan foydalaning (bu asl qiymati); uni markazlashtirish uchun justify = CENTER-dan foydalaning yoki o‘ng-oqlash uchun justify = RIGHT-dan foydalaning.
15	Menu	Menyu tugmachasini bir qator tanlov bilan bog‘lash uchun ushbu parametrni o‘z ichiga olgan Menyu ob’ektiga

		o‘rnating. Ushbu menyu ob’ekti konstruktorga birinchi argument sifatida bog‘langan menyu tugmachasini berish orqali yaratilganbo‘lishi kerak.
16	Padx	Menyu tugmachasi matnining chap va o‘ng tomoniga qancha joy qoldirish kerak.Standart - 1.
17	Pady	Menyu tugmasi matni ustida va pastda qancha joy qoldirish kerak. Standart - 1.
18	Relief	Uch o‘lchovli chegara soyalash effektlarini tanlaydi. Sukut bo‘yicha RAISED.
19	State	Odatda menyu tugmachalari sichqonchaga javob beradi. Menyu tugmachasini kulrangga aylantirish va uni javob bermaslik uchun state = DISABLED.
20	Text	Menyu tugmachasida matnni ko‘rsatish uchun ushbu parametrni kerakli matnni o‘z ichiga olgan qatorga o‘rnating. Satr ichidagi yangi qatorlar ("\ n") qatorlarning uzilishiga olib keladi.
21	textvariable	Siz ushbu menyu tugmasi bilan StringVar sinfidagi boshqaruv o‘zgaruvchisini bog‘lashingiz mumkin. Ushbu boshqaruv o‘zgaruvchisini o‘rnatish, ko‘rsatilgan matnni o‘zgartiradi.
22	Underline	Odatda menyu tugmachasida matn ostida hech qanday chiziqcha ko‘rinmaydi. Belgilarning birining ostiga chizish uchun ushbu parametrni ushbu belgi indeksigao‘rnating.
23	Width	Belgilarda vidjetning kengligi. Sukut bo‘yicha 20 ga teng.
24	wraplength	Odatda, chiziqlar o‘ralgan emas. Siz ushbu parametrni bir qator belgilarga o‘rnatishingiz mumkin va barcha satrlar bu raqamdan oshib ketmaydigan qismlargabo‘linadi.

Ushbu vidjetning maqsadi bizning dasturimiz tomonidan ishlatilishi mumkin bo‘lgan barcha turdagi menyularni yaratishga imkon berishdir. Asosiy funktsiya uchta menyu turini yaratishusullarini taqdim etadi: pop-up, toplevel va pull-down.

Menyularning yangi turlarini amalga oshirish uchun boshqa kengaytirilgan vidjetlardan ham foydalanish mumkin, masalan *OptionMenu* vidjeti, bu tanlov doirasida elementlarning ochiladigan ro‘yxatini yaratadigan maxsus turini amalga oshiradi.

Sintaksis

Ushbu vidjetni yaratish uchun oddiy sintaksis :
w = Menu (master, xossa=qiymat, ...)

Parametrlar

master - Bu ota-ona oynasini aks ettiradi.
xossa - Mana bu vidjet uchun eng ko‘p ishlatiladigan xossalarining ro‘yxati. Ushbuparametrlar vergul bilan ajratilgan kalit-qiymat juftlari sifatida ishlatilishi mumkin.

№	*Option*	*Tavsif*
1	activebackground	Sichqoncha ostida bo‘lganida tanlovda paydo bo‘ladigan fon rangi.
2	activeborderwidth	Tanlov atrofida chizilgan sichqoncha ostida bo‘lgan chegaraning kengliginibelgilaydi. Odatiy - 1 piksel.
3	activeforeground	Sichqoncha ostida bo‘lganida tanlovda paydo bo‘ladigan oldingi rang.
4	bg	Sichqoncha ostida bo‘lmagan tanlov uchun fon rangi.
5	bd	Barcha tanlov atrofida chegara kengligi. Standart - 1.
6	cursor	Sichqoncha tanlov tugashi bilan paydo bo‘ladigan kursor, lekin faqat menyuo‘chirilganida.
7	disabledforeground	Holati O‘CHIRILGAN elementlar uchun matnning rangi.
8	font	Matn tanlovi uchun standart shrift.
9	fg	Sichqoncha ostida bo‘lmagan tanlov uchun ishlatiladigan oldingi rang.
10	postcommand	Siz ushbu parametrni protseduraga o‘rnatishingiz mumkin va ushbu protsedurahar safar kimdir ushbu menyuni ochganida chaqiriladi.
11	relief	Standart menyular uchun 3 o‘lchovli effekt - bu yordam = RAISED.
12	image	Ushbu menyu tugmachasida rasmni ko‘rsatish uchun.

13	selector	Tanlanganida tugmalar va radio tugmalarida ko‘rsatilgan rangni belgilaydi.
14	tearoff	Odatda menyu o‘chirilishi mumkin, tanlov ro‘yxatidagi birinchi pozitsiyani (0- pozitsiyani) tearoff elementi egallaydi va qo‘shimcha tanlovlar 1-pozitsiyadan
		boshlab qo‘shiladi. Agar siz tearoff = 0 ni o‘rnatgan bo‘lsangiz, menyu tearoff xususiyatiga ega bo‘lmaydi va tanlovlar 0 pozitsiyasidan boshlab qo‘shiladi.
15	title	Odatda, tearoff menyu oynasining sarlavhasi ushbu menyuga olib boradigan menyu tugmasi yoki kaskad matni bilan bir xil bo‘ladi. Agar siz ushbu oynaningsarlavhasini o‘zgartirmoqchi bo‘lsangiz, sarlavha parametrini shu qatorga o‘rnating.

Metodlari

Ushbu usullar Menyu ob’ektlarida mavjud :

№	*Option*	*Tavsif*
1	add_command (options)	Menyuga menyu elementini qo‘shadi.
2	add_radiobutton(options)	Radio tugmasi menyusi elementini yaratadi.
3	add_chekbutton(options)	Tekshirish tugmachasi menyusi bandini yaratadi.
4	add_cascade(options)	Berilgan menyuni asosiy menyuga bog‘lash orqali yangi ierarxik menyuyaratadi.
5	add_separator()	Menyuga ajratuvchi qator qo‘shadi.
6	add(type, options)	Menyuga ma’lum bir menyu elementini qo‘shadi.
7	delete (startindex, [,endindex])	Startindex dan endindexgacha bo‘lgan menyu elementlarini o‘chiradi.
8	entryconfig (index,options)	Indeks bilan aniqlangan menyu bandini o‘zgartirish va uning parametrlarinio‘zgartirishga imkon beradi.
9	index (item)	Berilgan menyu elementi yorlig‘ining indeks raqamini qaytaradi.
10	insert_separator	Indeks bilan belgilangan joyga yangi ajratgich joylashtiring.

11	invoke(index)	Pozitsiya indeksidagi tanlov bilan bog‘liq bo‘lgan qayta qo‘ng‘iroqni chaqiradi. Agar tugma bo‘lsa, uning holati o‘rnatilgan va tozalangan o‘rtasida almashtiriladi; agar radio tugmasi bo‘lsa, bu tanlov o‘rnatiladi.
12	type(index)	Indeks bo‘yicha belgilangan tanlov turini qaytaradi: "cascade", "checkbutton", "command", "radiobutton", "separator", yoki "tearoff".

Tkinter Message (“Xabar”)

Ushbu vidjet ko‘p satrli va o‘zgartirilmaydigan ob’ektni taqdim etadi, u matnlarni aks ettiradi,satrlarni avtomatik ravishda buzadi va tarkibini oqlaydi. Uning funktsionalligi *Label* vidjeti tomonidan taqdim etilganga juda o‘xshash, faqat u matnni avtomatik ravishda o‘rab, berilgankenglik yoki tomonlar nisbatini saqlab turishi mumkin.

Sintaksis

Ushbu vidjetni yaratish uchun oddiy sintaksis :

w = Message (master, xossa=qiymat, ...)

Parametrlar

master - Bu ota-ona oynasini aks ettiradi.

xossa - Mana bu vidjet uchun eng ko‘p ishlatiladigan xossalarining ro‘yxati. Ushbuparametrlar vergul bilan ajratilgan kalit-qiymat juftlari sifatida ishlatilishi mumkin.

№	*Option*	*Tavsif*
1	anchor	Ushbu parametr, agar vidjetda matn ehtiyojidan ko‘proq joy bo‘lsa, matn qayerda joylashishini boshqaradi. Odatiy bo‘lib, matnni mavjud bo‘shliqda markazlashtiradigan anchor = CENTER
2	bg	Oddiy fon rangi yorliq va indikator orqasida ko‘rsatiladi.
3	bitmap	Ushbu parametrni bitmap yoki rasm ob’ektiga tenglashtiring, shunda yorliqo‘sha grafikni aks ettiradi.
4	bd	Ko‘rsatkich atrofidagi chegara kattaligi. Standart - 2 piksel.
5	cursor	Agar siz ushbu parametrni kursor nomiga o‘rnatgan bo‘lsangiz (o‘q, nuqta va boshqalar), sichqoncha kursori tugma tugagandan so‘ng shu naqshga o‘zgaradi.

6	font	Agar siz ushbu yorliqda matnni namoyish qilsangiz (matn yoki matn o‘zgaruvchan variant bilan birga, shrift opsiyasi ushbu matn qaysi shriftdako‘rsatilishini belgilaydi.
7	fg	Agar siz ushbu yorliqda matn yoki bitmap ko‘rsatayotgan bo‘lsangiz, ushbu parametr matn rangini belgilaydi. Agar siz bitmap ko‘rsatayotgan bo‘lsangiz, bu bitmapdagi 1-bitlar holatida paydo bo‘ladigan rang.
8	height	Yangi ramkaning vertikal o‘lchamlari.
9	image	Yorliq vidjetida statik tasvirni ko‘rsatish uchun ushbu parametrni rasm ob’ektigao‘rnating.
10	justify	Matnning bir nechta satrlari bir-biriga nisbatan qanday tekislanishini belgilaydi: chap tomonga *LEF*T, markazlashtirilgan uchun markaziy (standart) uchun *CENTER* yoki o‘ngga tartiblanishi uchun *RIGHT*.
11	padx	Vidjet ichidagi matnning chap va o‘ng qismiga qo‘shimcha joy qo‘shildi.Standart - 1.
12	pady	Vidjet ichidagi matnning yuqorisida va ostiga qo‘shimcha joy qo‘shildi. Standart - 1.
13	relief	Yorliq atrofidagi dekorativ chegara ko‘rinishini belgilaydi. Odatiy qiymatiFLAT;
14	text	Yorliq vidjetida bir yoki bir nechta satrlarni ko‘rsatish uchun ushbu parametrnimatn o‘z ichiga olgan qatorga o‘rnating. Ichki yangi qatorlar ("\ n") qatorni to‘xtatishga majbur qiladi.
15	textvariable	Yorliq vidjetida ko‘rsatilgan matnni StringVar sinfidagi boshqaruv o‘zgaruvchisiga saqlash uchun ushbu parametrni ushbu o‘zgaruvchiga o‘rnating.
16	underline	Ushbu parametrni n ga o‘rnatib, matnning n harfi ostida 0 dan boshlab, pastki chizig‘ini (_) ko‘rsatishingiz mumkin. Sukut bo‘yicha chiziq underline= -1, ya’nipastki chiziq chizilmaydi.
17	width	Belgilardagi label ning kengligi (piksel emas!). Agar ushbu parametr o‘rnatilmagan bo‘lsa, Label uning tarkibiga mos keladigan hajmga ega bo‘ladi.
18	wraplength	Ushbu parametrni kerakli raqamga o‘rnatib, har bir satrdagi belgilar sonini cheklashingiz mumkin. Standart qiymat 0, chiziqlar faqat yangi satrlarda buzilishini anglatadi.

Tkinter Radiobutton (“Radio tugma”)

Ushbu vidjet ko‘p variantli tugmachani amalga oshiradi, bu foydalanuvchiga mumkin bo‘lgan tanlovlarni taqdim etishning bir usuli va foydalanuvchiga ulardan faqat bittasini tanlashga imkon beradi. Ushbu funktsiyani amalga oshirish uchun radio tugmalarining har bir guruhi bir xil o‘zgaruvchiga bog‘langan bo‘lishi kerak va tugmachalarning har biri bitta qiymatni ramziy qilishi kerak. Bir tugmachadan ikkinchisiga o‘tish uchun Tab tugmachasidan foydalanishingiz mumkin.

Sintaksis

Ushbu vidjetni yaratish uchun oddiy sintaksis -

w = Radiobutton (master, xossa=qiymat, ...)

Parametrlar

master - Bu ota-ona oynasini aks ettiradi.

xossa - Mana bu vidjet uchun eng ko‘p ishlatiladigan xossalarining ro‘yxati. Ushbuparametrlar vergul bilan ajratilgan kalit-qiymat juftlari sifatida ishlatilishi mumkin.

№	*Option*	*Tavsif*
1	activebackground	Sichqoncha radio tugmachasini bosganda fon rangi.
2	activeforeground	Sichqoncha radio tugmasi ustida turganida oldingi rang.
3	anchor	Agar vidjet kerakli miqdordan kattaroq bo‘shliqda yashasa, ushbu parametr ushbu tugmachada radio tugmachaning qaerda joylashganligini aniqlaydi. Odatiy bo‘lib anker = CENTER.
4	bg	Ko‘rsatkich va yorliq ortidagi normal fon rangi.
5	bitmap	Monoxrom tasvirni radio tugmasida ko‘rsatish uchun ushbu parametrni bitmap-ga o‘rnating.
6	borderwidth	Ko‘rsatkich qismining o‘zi atrofidagi chegara kattaligi. Standart - 2 piksel.
7	command	Foydalanuvchi har safar ushbu radio tugmachasining holatini o‘zgartirgandachaqiriladigan protsedura.
8	cursor	Agar siz ushbu parametrni kursor nomiga o‘rnatgan bo‘lsangiz (o‘q, nuqta va boshqalar), sichqoncha kursori radio tugmachasi tugagandan so‘ng shu naqshgao‘zgaradi.

9	font	Matn uchun ishlatiladigan shrift.
10	fg	Matnni ko‘rsatish uchun ishlatiladigan rang.
11	height	Radio tugmasidagi satrlar soni (piksel emas). Standart - 1.
12	highlightbackground	Radio tugmasi fokusga ega bo‘lmaganida fokusning rangi ta’kidlanadi.
13	highlightcolor	Radio tugmasi fokusga ega bo‘lganda fokusning rangi ta’kidlanadi.
14	image	Ushbu radio tugmasi uchun matn o‘rniga grafik tasvirni ko‘rsatish uchun ushbu parametrni rasm ob’ektiga o‘rnating.
15	justify	Agar matnda bir nechta satr bo‘lsa, ushbu parametr matnning qanday asoslanishini boshqaradi: CENTER (standart), LEFT yoki RIGHT.
16	padx	Radio tugmasi va matnning chap va o‘ng tomoniga qancha joy qoldirish kerak. Standart - 1.
17	pady	Radio tugmasi va matni ustida va pastda qancha joy qoldirish kerak. Standart -1.
18	relief	Yorliq atrofidagi dekorativ chegara ko‘rinishini belgilaydi. Odatiy qiymati FLAT;
19	selector	O‘rnatilganda radio tugmachasining rangi. Standart qizil rang.
20	selectimage	Agar siz radio tugmachasi o‘chirilganida rasm o‘rniga matn o‘rniga grafik tasvirni ko‘rsatish uchun foydalanayotgan bo‘lsangiz, tanlash tugmachasi parametrini radio tugmasi o‘rnatilganda ko‘rsatiladigan boshqa rasmga o‘rnatishingiz mumkin.
21	state	Odatiy holat state = NORMAL, ammo siz boshqaruvni kul rangga aylantirish va uni javob bermaslik uchun state = DISABLED-ni o‘rnatishingiz mumkin. Agar kursor hozirda radio tugmachasi ustida bo‘lsa, state ACTIVE.
22	text	Radio tugmasi yonida ko‘rsatilgan yorliq. Bir nechta matn satrlarini ko‘rsatish uchun yangi qatorlardan ("\n") foydalaning.
23	textvariable	Yorliq vidjetida ko‘rsatilgan matnni StringVar sinfidagi boshqaruv o‘zgaruvchisiga saqlash uchun

		ushbu parametrni ushbu o‘zgaruvchiga o‘rnating.
24	underline	Ushbu parametrni n ga o‘rnatib, matnning n harfi ostida 0 dan boshlab, pastki chizig‘ini (_) ko‘rsatishingiz mumkin. Sukut bo‘yicha chiziq underline = -1, ya’nipastki chiziq chizilmaydi.
25	value	Radio tugmasi foydalanuvchi tomonidan yoqilganda uning boshqaruv o‘zgaruvchisi joriy qiymat parametriga o‘rnatiladi. Agar boshqaruv o‘zgaruvchisi IntVar bo‘lsa, guruhdagi har bir radio tugmachasiga boshqa tamsayı qiymatining variantini bering. Agar boshqaruv o‘zgaruvchisi StringVar bo‘lsa, har bir radio
		tugmachasiga turli xil satr qiymati variantini bering.
26	variable	Ushbu radio tugmachani guruhdagi boshqa radio tugmalar bilan baham ko‘radigan boshqaruv o‘zgaruvchisi. Bu IntVar yoki StringVar bo‘lishi mumkin.
27	width	Belgilardagi yorliqning kengligi (piksel emas!). Agar ushbu parametr o‘rnatilmagan bo‘lsa, yorliq uning tarkibiga mos keladigan hajmga ega bo‘ladi.
28	wraplength	Ushbu parametrni kerakli raqamga o‘rnatib, har bir satrdagi belgilar sonini cheklashingiz mumkin. Odatiy qiymat 0, chiziqlar b bo‘lishini anglatadi faqat yangi qatorlarda.

Metodlari:

№	*Option*	*Tavsif*
1	deselect()	Radio tugmachasini tozalaydi (o‘chiradi).
2	flash()	Radio tugmachasini faol va normal ranglar orasida bir necha marta yondiradi, lekin uni qanday boshlagan bo‘lsa, shunday qoldiradi.
3	invoke()	Agar foydalanuvchi o‘z holatini o‘zgartirish uchun radio tugmachasini bosgan bo‘lsa, sodir bo‘ladigan amallarni bajarish uchun ushbu usulni chaqirishingiz mumkin.
4	select()	Radio tugmachasini o‘rnatadi (yoqadi).

Nazorat savollari.

1. Buttonning parametrlari va metodlari qaysilar?
2. Tkinter oynasiga 3 ta radiobutton hosil qiling?
3. Vidjet xossalari?

20-21-ma’ruza.Python standart kutubxonasi.

Reja:

1. Python standatr kutubxonalari.
2. Standart kutubxonalarga misollar.

Python dasturlash tilining kutubxonasida 200dan ortiq standart modullari bo‘lib, ularning ortiqcha yoki keraksizi yo‘q. Dastur tuzish jarayonida vazifasiga qarab modullardan foydalanamiz. Masalan, matematik ifodalarni hisoblash uchun math, grafik interfeysli dasturlar yaratish uchun tkinter, fayl va papkalar bilan ishlashda os va boshqa standart modullar bor.

Modullar bu o‘zida funksiyalar, klasslar, ro‘yxatlar, o‘zgaruvchilarni saqlovchi python fayllaridir. Modullar ikkiga bo‘linadi: standart modullar va foydalanuvchilar tomonidan yaratilgan modullar. Standart modullar python dasturlash muhiti bilan birga o‘rnatilgan bo‘ladi. Modullardan foydalanish uchun ularni quyidagicha dastur kodlariga bog‘lashimiz kerak.

Import yordamida:

```
import math
print(math.pi)
```

Barcha funksiyalarini olish uchun:

```
from math import *
print(pi)
```

Modulni nomini o‘zgartirib olish:

```
import math as m
print(m.pi)
```

Modullardagi funksiya, o‘zgaruvchilarni ko‘rish uchun dir() funksiyasidan foydalanamiz. dir() funksiyasi barcha modullar uchun amal qiladi. Masalan,

```
import math
print(dir(math))
```

ushbu dastur math modulini funksiya va o‘zgaruvchilarni chiqarib beradi.

Standart modullar.

1.Math-matematik ifodalarni, masalalarni hisoblashda foydalaniladi.

Funksiya	Vazifasi
sqrt(x)	x ni kvadrat ildizi.
pow(x,n)	x ning n chi daRejadi.
factorial(x)	x ning faktorialini hisoblaydi.
abs(x)	x ning modulini hisoblaydi.
ceil(x)	x ni katta songacha yaxlitlaydi.
round(x,n)	x sonini nuqtadan keyin n ta belgigacha yaxlitlaydi.
floor(x)	x ni kichik songacha yaxlitlaydi.
log(a,b)	b asosga koʻra a logarifmni hisoblash.
e	e oʻzgarmas kattalik
pi	Pi oʻzgarmas kattalik
log10(x)	x ning 10 lik logarifmini hisoblaydi.
sin(x)	x ning sinusini hisoblaydi.
cos(x)	x ning kosinusini hisoblaydi.
tan(x)	x ning tangesini hisoblaydi.
asin(x)	x ning arksinusini hisoblaydi.
acos(x)	x ning arkkosinusini hisoblaydi.
atan(x)	x ning arktangesini hisoblaydi.
degrees(x)	Radiandan gradusga oʻtkazish.
radians(x)	Gradusdan radianga oʻtkazish.

Masala. Kiritilgan sonni sinusi, kosinusi, tangensini hisoblovchi dastur tuzing.

```
import math
n=int(input("Son kiriting: "))
print("Sinusi", math.sin(n))
print("Kosinusi", math.cos(n))
print("Tangensi", math.tan(n))
```

2.Turtle-bu modul yordamida ekranda turli shakl va chiziqlarni chizishimiz mumkin. Pythonni oʻrganuvchilar ham birinchi shu moduldan oʻrganishni boshlashadi.

Funksiya	Vazifasi
forward()	Oldinga yurgizdirish.
backward()	Orqaga yurgizdirish.
right()	Oʻnga burish.
left()	Chapga burish.
bgcolor()	Fon rangi.

title()	Sarlavha nomi.
speed()	0..10 oraliqdagi tezligi.
pendown()	Qalamni tushirib chizish mumkin.
penup()	Qalamni ko‘tarish.
pensize()	Chiziq qalinligi.
pencolor()	Chiziq rangi.
clearscreen()	Ekranni tozalash.

Turtle modulining boshqa ko‘plab funksiyalari bor.
Masala. To‘rtburchak chizuvchi dastur tuzing.

```
import turtle
t=turtle.Turtle()
t.forward(100)
t.right(90)
t.forward(100)
t.right(90)
t.forward(100)
t.right(90)
t.forward(100)
```

3.Random-bu modul tasodifiy sonlarni olishda yoki ro‘yxat elementlarini tasodifiy aralashtirishda foydalaniladi.

Funksiya	**Vazifasi**
random()	0 va 1 dan birini oladi.
randint(a,b)	a va b oraliqdan tasodifiy sonni olish
randrange(a,b,c)	a va b oraliqdan tasodifiy sonni c qadam bo‘yicha olish
choice()	Ro‘yxatdan tasodifiy elementni olish
shuffle()	Ro‘yxatni aralashtirish
uniform(a,b)	a va b oraliqdan tasodifiy sonni olish

Masala. 1dan 100 gacha sonlar orasidan tasodifiy sonni ekranga chiqaruvchi dastur tuzing.

```
import random
print("Tasodifiy son:", random.randint(1,100))
```

4.Datetime-sana va vaqtni aniqlash ular ustida amallar bajarish uchun mo‘ljallangan ko‘plab funksiyalarga ega.

Joriy sana va vaqtni olish uchun s=**datetime. datetime.now**()

funksiyasidan foydalanamiz. **strftime**(metod) funksiyasi yordamida sana va vaqtni formatlab kerakli ko‘rinishda olishimiz mumkin. Quyida metodlar jadvali berilgan.

Metod	**Vazifasi**
%a	Hafta kuni (qisqa)
%A	Hafta kuni (to‘liq)
%w	Hafta kuni (raqam shaklida)
%d	Oyning sanasi
%b	Oy nomi (qisqa)
%B	Oy nomi (to‘liq)
%m	Oy (raqam ko‘rinishida)
%y	Yil (qisqa)
%Y	Yil (to‘liq)
%H	Soat (00-23)
%I	Soat (00-12)
%p	Kun vaqti (AM/PM)
%M	Minut (00-59)
%S	Sekund (00-59)
%j	Yildagi kun raqami (001-366)
%U	Yildagi hafta raqami, Yakshanba birinchi kun sifatida (00-53)
%W	Yildagi hafta raqami, Dushanba birinchi kun sifatida (00-53)
%c	Mahalliy sana va vaqt
%x	Mahalliy sana
%X	Mahalliy vaqt

Masala. Joriy vaqtni chiqaruvchi dastur tuzing.

```
from datetime import *
s=datetime.now()
print("Yil:", s.strftime("%Y"))
print("Oy:", s.strftime("%B"))
print("Kun:", s.strftime("%d"))
print("Soat:", s.strftime("%H"))
print("Minut:", s.strftime("%M"))
print("Sekund:", s.strftime("%S"))
```

5.OS-moduli fayl va papkalar bilan ishlash uchun mo‘ljallangan. Operatsion tizimda fayl va papkalarni boshqarish uchun juda ko‘plab

funksiyalarga ega bo‘lib, quyida ayrimlari keltirilgan.

Funksiya	**Vazifasi**
mkdir()	Yangi papka yaratish.
rmdir()	Papkani o‘chirib tashlash.
listdir()	Ko‘rsatilgan papkadagi barcha papka va fayllar ro‘yxatini aniqlaydi.
close()	Ulangan fayllarni yopish.
open()	Faylni ochish.
read()	Fayldan o‘qish.
write()	Faylga yozish.
link()	Yorliq yaratish.
remove()	Faylni o‘chirish.
rename()	Fayl yoki papkaning nomini o‘zgartirish
system()	Tizm buyruqlaridan foydalanish.
cpu_count()	Protsesor sonini aniqlaydi.

Masala. Joriy papkada "Salom" nomli yangi papka yaratuvchi dastur tuzing.

```
import os
os.mkdir("Salom")
```

6.SYS-sys moduli python interpretatori bilan ishlashga mo‘ljallangan.

Funksiya	**Vazifasi**
version()	Python versiyasini qaytaradi.
copyright()	Pythonga tegishli mualliflik huquqlarini o‘z ichiga oladi.
_clear_type_cache()	Ichki keshni tozalash.
exc_info()	Xatolik xaqida ma’lumot olish.
exit()	Pythondan chiqish.
implementation()	Python haqida ma’lumotlar qaytaradi.
maxsize()	O‘zgaruvchiga beriladigan maksimal son uzunligini aniqlaydi
modules()	O‘rnatilgan modullar ro‘yxatini qaytaradi.
path()	Modullar joylashgan papkalarni ro‘yxatini qaytaradi.
platform()	Tizimni aniqlash.

Masala. O‘rnatilgan python dasturlash tili versiyasini aniqlovchi dastur tuzing.

```
import sys
```

print("Python versiyasi:", sys.version)

7.Socket-tarmoq bilan ishlovchi dasturlar yaratishda foydalaniladi. Tarmoq bilan ishlovchi dasturlar server va klient qismga bo‘linadi. Socket yordamida ular o‘zaro ma’lumot almashadi.

Funksiya	**Vazifasi**
socket()	Berilgan parametr bo‘yicha tarmoq yaratadi.
error()	Xatolikni qaytaradi.
sockettype()	Sokket turini aniqlash.
close()	Sokketni yopish
send()	Tarmoqdan ma’lumot jo‘natish
recv()	Tarmoqdan ma’lumotni olish
create_connection()	Tarmoqga ulanish
create_server()	Server yaratish
accept()	Ulanishni qabul qilish.
connect()	Tarmoqqa ulanish
sendfile()	Tarmoqdan fayl jo‘natish
shutdown()	Tarmoqdan uzish

Masala. Socket yordamida server yaratuvchi dastur tuzing.

```
import socket
addr = ("", 8080)
if socket.has_dualstack_ipv6():
   s = socket.create_server(addr, family=socket.AF_INET6, dualstack_ipv6=True)
else:
   s = socket.create_server(addr)
```

8.Tkinter-GUI yani grafik interfeysli dasturlar tuzish uchun foydalaniladi. Python dasturlash tilida faqat buyruqlar satrida natija chiqaruvchi dasturlar tuzilmaydi. Foydalanishga qulay interfeysli dasturlar yaratsa ham bo‘ladi. Buning uchun tkinter modulining vidjetlaridan foydalanamiz.

Vidjet	**Vazifasi**
Label	Ma’lumot chiqarish uchun matnli maydon hosil qiladi.
Combobox	Ochiluvchi ro‘yxat yaratadi.
Spinbox	Ko‘paytiruvchi va kamaytiruvchi tugmalarga ega kiritish qatorini hosil qiladi.

Progressbar	Jarayonni ko‘rsatuvchi vidjet yaratadi.
checkbutton	Ro‘yxatdan bir nechtasini tanlashda ishlatiladigan vidjet.
PhotoImage	Dastur oynasiga rasm joylashtirish uchun vidjet
Text	Ma’lumot chiqarish uchun matnli maydon hosil qiladi.
Entry	Matn kiritadigan maydon hosil qiladi.
OptionMenu()	Tanlanadigan menyu hosil qiladi.
Radiobutton	Ro‘yxatdan bittasini tanlashda ishlatiladigan vidjet.
Button	Boshqarish tugmasi.

Masala. Tkinterdan foydalanib "Hello World" dasturini tuzing.

```
import tkinter
oyna=tkinter.Tk()
l=tkinter.Label(oyna, text="Hello World!")
l.pack()
oyna.mainloop()
```

Pythonda F1 klavishini bosib qolgan funksiyalar haqida ma’lumot olish mukin. Dastur tuzish jarayonida ko‘p vaqt sarflamaslik va ish sifati, unumdorligini oshirish uchun standart modul, funksiyalar haqida batafsil o‘rganish muhim ahamiyatga ega.

21-ma’ruza.Python tashqi kutubxonasi

Oldingi darsimizda Python bilan birga o‘rnatluvchi, standart kutubxona va undagi ba’zi foydali modullar bilan tanishdik. Ushbu darsimizda esa tashqi kutubxona bilan tanishamiz. Bu kutubxonalar yillar davomida turli foydalanuvchilar tarafidan yaratilib, yangilanib kelinadi. Bunday kutubxonalarda boshqa dasturchilar o‘zlari yaratgan turli paketlarni (package) boshqalar bilan ulashadi.

Paket (package) — modullar yig‘indisi.

Tashqi kutubxonalar va ular ichidagi paketlar shunchalik ko‘pki, deyarli istalgan vazifa yoki xizmat uchun katta ehtimollik bilan kerakli dasturlar allaqachon bir nechtadan yaratilgan. Bugungi kunda Python uchun eng katta tashqi kutubxonalardan biri bu PyPi.org sahifasi.

PIP

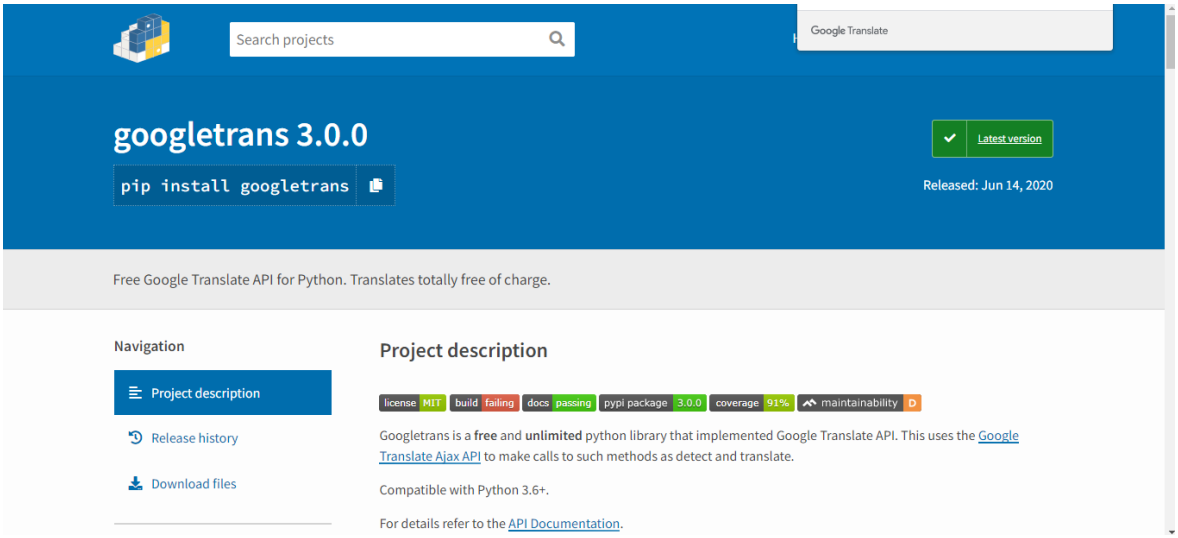
Tashqi paketlarni o‘rnatish uchun Pythonda maxsus pip paket

menejeri mavjud. pip odatda Python bilan birga o‘rnatiladi, lekin turli sabablarga ko‘ra kompyuteringizda pip o‘rnatilmagan bo‘lsa, uni o‘rnatish lozim.

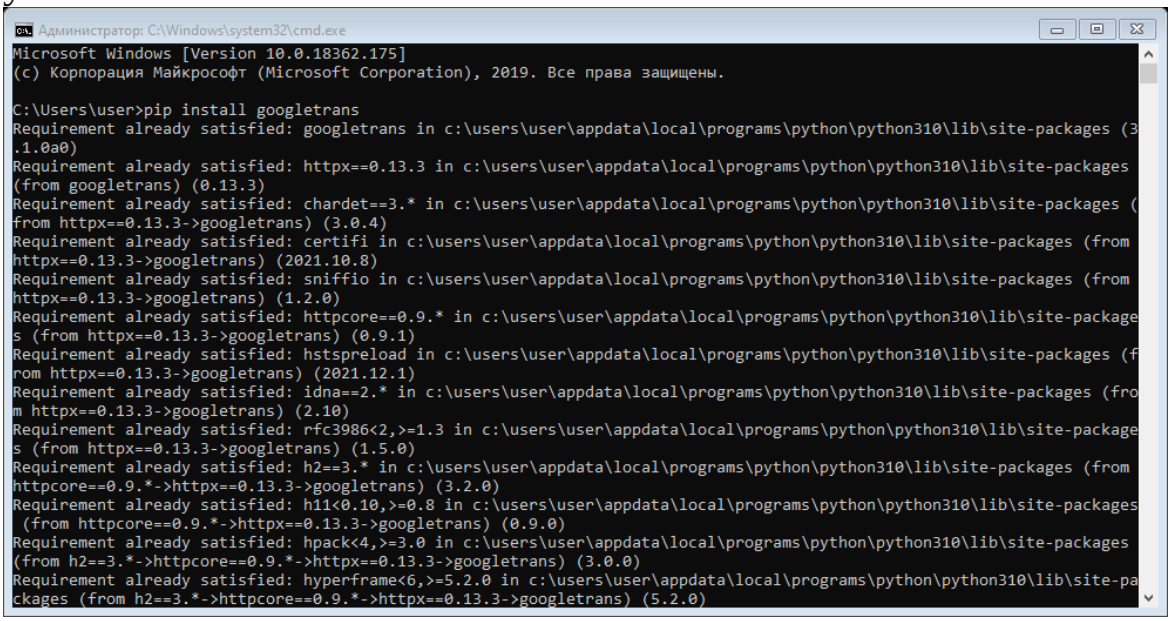
Paket menejer yordamida tashqi paketlarni o‘rnatish juda oson, buning uchun Windows terminalda (cmd) (yoki Spyder konsolida, yoki PyCharm konsolida va hokazo) *pip install paket_nomi* komandasidan foydalanasiz.

Python tashqi kuyubxonasining rasmiy sayti: PyPi.org

Paket nomi qanday yozilishini paketning rasmiy sahifasidan ko‘rib olishingiz mumkin. Misol uchun, quyidagi rasmda googletrans paketinig sahifasi va pip komandasi qanday yozilishi ko‘rsatilgan:



Demak ushbu paketni o‘rnatish uchun ’pip install gooletrans’ deb yozamiz.



Biror paketni o‘chirib tashlash uchun esa pip uninstall paket_nomi deb yozamiz.

Ushbu darsimizning maqsadi tashqi paktelar va modullar bilan tanishish orqali, Pythonning naqadar keng qamrovli til ekanini ko‘rsatish. Aslida, har bir modul haqida, ularning imkoniyatlari haqida soatlab gapirish mumkin, lekin biz vaqtni tejash maqsadida turli yo‘nalishlardagi ba’zi modullar bilan tanishamiz va ularning ishlashiga sodda misollar ko‘rish bilan chegaralanamiz.

Har bir modul haqida batafsil ma’lumot olish uchun modulning sahifasiga murojat qilish kerak.

Requests (pip install requests) paketi.

Bu paket yordamida Pythonda veb sahifalarga murojat qilishimiz (so‘rov yuborishimiz) va ulardan qaytgan ma’lumotlar ustida turli amallar bajarishimiz mumkin. Misol uchun quyida requests yordamida daryo.uz sahifasini to‘liqligicha toritb olamiz:

```
import requests
from pprint import pprint
manzil= "https://daryo.uz/category/texnologiyalar/"
r = requests.get(manzil)
pprint(r.text)
```

Ko‘pincha requests paketidan APIlar bilan ishlashda foydalaniladi. API bu ma’lum bir veb hizmatga so‘rov yuborish orqali undan foydalanish. Misol uchun yandex tarjimonga yoki google haritalari xizmatiga requests paketi yordamida API so‘rov yuborish va o‘zimizga kerakli ma’lumotlarni olishimiz mumkin. API haqida kelgusida batafsil dars qilamiz. Hozir esa sodda misollar bilan cheklanamiz.

Internetda restcountries.eu sahifasi mavjud. Bu sahifa orqali dunyodagi davlatlar haqida turli ma’lumotlarni olishingiz mumkin. Sahifadan foydalanish qulay bo‘lishi uchun esa, sahifa yaratuvchilari bir nechta tayyor API lar e’lon qilishgan. Misol uchun O‘zbekiston haqida ma’lumot olish uchun quyidagi manzilga so‘rov yuborasiz: https://restcountries.eu/rest/v2/name/Uzbekistan.

API dan qaytgan natija JSON (lug‘at) ko‘rinishda bo‘ladi va biz bu lug‘atdan o‘zimizga kerakli ma’lumotni sug‘urib olishimiz mumkin. Misol uchun quyidagi kodimiz APIga yuborilgan davlatning poytaxtini ko‘rsatadi:

```
import requests
country = "Uzbekistan"
url = f"https://restcountries.eu/rest/v2/name/{country}"
r = requests.get(url)
r_json = r.json()[0]
print(r_json[’capital’])
>>>Tashkent
```

BeautifulSoup4(pip install beautifulsoup4) paketi

BeautifulSoup juda kuchli modullardan biri bo‘lib, bu modul yordamida turli veb sahifalardan istalgan ma’lumotlarni yig‘ishtirib (scarpping) olish mumkin. Biror kishining instagram sahifasidagi barcha rasmlar deysizmi, Facebook guruhidagi barcha postlar va izohlar deysizmi, oldi-sotdi bozoridagi e’lonlar deysizmi, marhamat, bs4 moduli yordamida buni bemalol avtomatlashtirish mumkin.

Odatda bs4 moduli requests moduli bilan hamkorlikda ishlaydi. Keling, sodda misol kor’amiz. Avvalgi bo‘limda, requests yordamida kun.uz sahifasining html kodini olgan edik. Endi esa bs4 yordamida html sahifadan oxirgi yangiliklarning ma’ruzasini ajratib olamiz.

```
import requests
from bs4 import BeautifulSoup
```

```
sahifa = "https://kun.uz/news/main"
r = requests.get(sahifa)
soup = BeautifulSoup(r.text, 'html.parser')
news = soup.find_all(class_="news-title") # yangiliklarning ma'ruzasini ajratib olamiz
print(news[0].text) # eng birinchi yangilikni konsolga chiqaramiz
```

```
Python 3.10.1 (tags/v3.10.1:2cd268a, Dec  6 2021, 19:10:37) [MSC v.1929 64 bit (
AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.

= RESTART: C:/Users/user/AppData/Local/Programs/Python/Python310/BeautifulSoup.p
y
«Сиз бизга душман эмассиз». Байден россияликларга мурожаат қилди
```

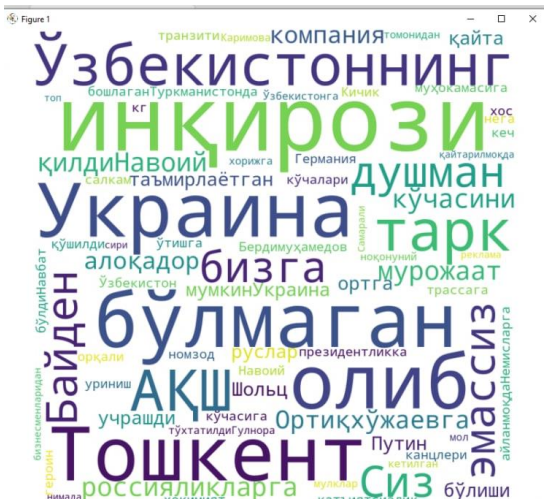
wordcloud va matplotlib (pip install wordcloud va pip install matplotlib) paketlari.

Wordcloud moduli yordamida katta matnlarda eng ko‘p uchraydigan so‘zlarni chiroyli qilib, so‘zlarni buluti chiqarish mumkin.

Wordcloud moduli grafiklarni chizishga mo‘ljallangan matplotlib moduli bilan hamkorlikda ishlaydi. Quyida kun.uz sahifasidan olingan yangiliklar uchun so‘zlar bulutini yaratishni ko‘ramiz:

```
import requests
from bs4 import BeautifulSoup
from wordcloud import WordCloud
import matplotlib.pyplot as plt
sahifa = "https://kun.uz/news/main"
r = requests.get(sahifa)
soup = BeautifulSoup(r.text, 'html.parser')
news = soup.find_all(class_="news-title")
matn=""
for n in news:
    matn += n.text
# kerakmas so‘zlar
stopwords = ["учун","бўйича","лекин","билан","ва","бор","ҳам","хил","йил"]
# bulutni yaratamiz
wordcloud = WordCloud(width = 1000, height = 1000,
                background_color ='white',
                stopwords = stopwords,
                min_font_size = 20).generate(matn)
```

```
# plot the WordCloud image
plt.figure(figsize = (8, 8), facecolor = None)
plt.imshow(wordcloud)
plt.axis("off")
plt.tight_layout(pad = 0)
plt.show()
```



Nazorat savollari.

1. Socket nima uchun ishlatiladi?
2. Requests moduli nima uchun ishlatiladi?
3. Kun.uz saytidagi yozuvlarni chaqirib olish?

22-ma’ruza.Pythonda rasmlar bilan ishlash.

Reja:

1. Rasmlar bilan ishlash modullari
2. Rasmlarni .pdf formatga o‘giruvchi ilova yaratish.

Ko‘pgina ilovalar raqamli tasvirlardan foydalanadi va bu bilan odatda ishlatiladigan tasvirlarni qayta ishlashga ehtiyoj bor. Agar siz ilovangizni Python bilan qurayotgan bo‘lsangiz va unga tasvirni qayta ishlash funksiyalarini qo‘shishingiz kerak bo‘lsa, siz foydalanishingiz mumkin bo‘lgan turli kutubxonalar mavjud.

Bu yerda qaysi kutubxona eng yaxshi ekanligi haqida bahslashmaymiz; ularning barchasining o‘ziga xos fazilatlari bor. Ushbu ma’ruza tasvirni qayta ishlashning keng doirasini ta’minlovchi va ulardan foydalanish oson bo‘lgan kuchli kutubxona Pillowga qaratiladi.

PIL — bu tasvirlarni manipulyatsiya qilish uchun bir nechta standart protseduralarni taklif qiluvchi kutubxona. Bu kuchli kutubxona, lekin 2009-yildan beri yangilanmagan va Python 3-ni qo‘llab-quvvatlamaydi. Pillow bunga asoslanib, Python 3-ga qo‘shimcha funksiyalar va qo‘llab-quvvatlaydi. U PNG, JPEG, PPM kabi bir qator tasvir fayl formatlarini qo‘llab-quvvatlaydi. GIF, TIFF va BMP. Ushbu kutubxonadan foydalanib, tasvirlar ustida kesish, o‘lchamini o‘zgartirish, tasvirga matn qo‘shish, aylantirish, kulrang rangni o‘zgartirish va hokazo kabi turli amallarni qanday bajarishni ko‘rib chiqamiz.

O‘rnatish va loyihani sozlash

Pillowni o‘rnatishdan oldin siz quyidagilarni bilishingiz kerak:

Pillow va PIL bir muhitda birga bo‘lolmaydi, shuning uchun sizda PIL o‘rnatilgan bo‘lsa, davom etishdan oldin uni o‘chirib tashlang.

Biz ushbu ma’ruzada Pillowning joriy barqaror versiyasidan foydalanamiz (yozish vaqtida 8.0.1 versiyasi). Ushbu versiya Python 3.6 va undan yuqori versiyalarini talab qiladi.

Pillowni pip ko‘rsatilganidek o‘rnatishingiz mumkin :

```
python3 -m pip install --upgrade pip
python3 -m pip install --upgrade Pillow
```

Barcha misollar kerakli tasvirlar ishga tushirilayotgan python skript fayli bilan bir xil katalogda bo‘lishini nazarda tutadi.

Tasvir ob’ekti

Python Imaging kutubxonasidagi muhim sinf bu Image sinfdir. U Image modulda aniqlangan va manipulyatsiya operatsiyalarini amalga oshirish mumkin bo‘lgan PIL tasvirini taqdim etadi. Ushbu klassning namunasi bir necha usulda yaratilishi mumkin: fayldan rasmlarni yuklash, noldan tasvirlarni yaratish yoki boshqa tasvirlarni qayta ishlash natijasida. Biz bularning barchasini ishlatilayotganini ko‘ramiz. *(pip install Image yoki pip install Pillow)*

Fayldan rasmni yuklash uchun biz moduldagi funksiyadan foydalanamiz, unga tasvirga yo‘l o‘tkazamiz .

```
from PIL import Image
image = Image.open(’22.jpg‘)
image.show()
```

Muvaffaqiyatli bo‘lsa, yuqoridagi Imageo byektni qaytaradi. Agar faylni ochishda muammo yuzaga kelsa, OSErroristisno paydo bo‘ladi. Image Obyektni olganingizdan so‘ng , siz uni qayta ishlash va boshqarish uchun sinf tomonidan belgilangan usullar va atributlardan foydalanishingiz mumkin. Rasmni ko‘rsatishdan boshlaylik.

Buni undagi usulni chaqirish orqali qilishingiz mumkin.

Bu tasvirni tashqi ko‘rish vositasida ko‘rsatadi (odatda macOS da oldindan ko‘rish, Unix da xv va Windows da Paint dasturi).show ()

Obyektning atributlari yordamida tasvir haqida ba’zi tafsilotlarni olishingiz mumkin.

```
from PIL import Image
image = Image.open(’22.jpg‘)
# rasm formati.
print(image.format) # Output: JPEG
# Rasm tomonidan ishlatiladigan piksel formati. Odatdagi qiymatlar "1", "L", "RGB" yoki "CMYK"dir."
print(image.mode) # Output: RGB
# Rasm hajmi, piksellarda. Hajmi (kenglik, balandlik) sifatida berilgan.
print(image.size) # Output: (883, 540)
# Ranglar palitrasi jadvali, agar mavjud bo‘lsa.
print(image.palette) # Output: None
```

Rasm turini o‘zgartirish. Tasvirni qayta ishlashni tugatganingizdan so‘ng, uni faylga yorliqlash uchun ishlatiladigan

nomni kiritish usuli bilan faylga saqlashingiz mumkin . Rasmni saqlashda siz uning asl nusxasidan boshqa kengaytmani belgilashingiz mumkin va saqlangan rasm belgilangan formatga o‘zgartiriladi.

```
image = Image.open(’22.jpg‘)
image.save(’22-yangi.png‘)
```

Yuqoridagi rasm bilan yuklangan Tasvir ob’ektini yaratadi va uni yangi faylga saqlaydi, Pillow fayl kengaytmasi PNG sifatida belgilanganligini ko‘radi va faylga saqlashdan oldin uni PNGga o‘zgartiradi. Fayl formatini aniq belgilash uchun ikkinchi argumentni taqdim etishingiz mumkin . Bu avvalgidek xuddi shunday qiladi . Odatda, bu ikkinchi dalilni keltirishning hojati yo‘q, chunki Pillow fayl nomi kengaytmasidan foydalanish uchun faylni saqlash formatini aniqlaydi, lekin agar siz nostandart kengaytmalardan foydalanayotgan bo‘lsangiz, formatni har doim shu tarzda belgilashingiz kerak.

Rasmlar hajmini o‘zgartirish.Tasvirning o‘lchamini o‘zgartirish uchun siz o‘lchami o‘zgartirilgan tasvirning kengligi va balandligini ifodalovchi ikki butun sonli kortej argumentiga o‘tib, undagi usulni chaqirasiz . Funktsiya ishlatilgan tasvirni o‘zgartirmaydi; buning o‘rniga yangi o‘lchamli boshqa Rasmni qaytaradi.

```
image = Image.open(’22.jpg‘)
new_image = image.resize((400, 400))
new_image.save(’22-yangi.jpg‘)
print(image.size) # Output: (1920, 1280)
print(new_image.size) # Output: (400, 400)
```

Python yordamida rasmlarni PDF-ga aylantirish (Image->PDF).

Dastlab, Python yordamida “png” tasvirini (JPEG uchun "jpg" fayl kengaytmasidan foydalaniladi) PDF- ga aylantirish uchun foydalanishimiz mumkin bo‘lgan dastur kodi(asosiy kod):

```
from PIL import Image
ima1 = Image.open(r’rasm saqlanadigan yo‘l\fayl nomi.png‘)
im1 = ima1.convert(’RGB’)
im1.save(r’pdf saqlanadigan yo‘l\fayl nomi.pdf’)
```

Python yordamida rasmlarni PDF-ga aylantirish uchun qadamlar ketma-ketligi:

1-qadam: PIL paketini o‘rnatish.

Birinchi bo‘lib quyidagi buyruq yordamida PIL paketini o‘rnatiladi:

pip install Pillow
(pip install qilingan bo‘lishi kerak)
2-qadam: Tasvir saqlangan yo‘lni olish.

Keyin rasm saqlanadigan yo‘l olinadi(agar rasm va python fayl bitta katalogda joylashgan bo‘lsa yo‘l shart emas. Masalan: Image.open(r’fayl nomi.png‘).
Masalan: C:\Users\user\Desktop
3-qadam: Python yordamida rasmni PDF-ga aylantirish
Yakuniy bosqichda tasvirni PDF-ga aylantirish uchun quyidagi dastur kodidan foydalanishimiz mumkin:

```
from PIL import Image
image1 = Image.open(r’ C:\Users\user\Desktop\2.png‘)
im1 = image1.convert(’RGB’)
im1.save(r’ C:\Users\user\Desktop\55.pdf’)
```

Demak, bizning dasturimizda rasm nomi 2.png va PDF faylizmiz nomi 55.pdf
Dastur kodi ishga tushirilgach C:\Users\user\Desktop\55.pdf nomli PDF faylimiz hosil bo‘ladi.

Python yordamida rasmlar ro‘yxatini *PDF-ga aylantirish:* Agar sizda rasmlar ro‘yxati bo‘lsa va ularning barchasini bitta PDF faylida saqlamoqchi bo‘lsangiz quydagi dastur kodi yordamida bajarish mumkin:

```
image1 = Image.open(r’1.jpg‘)
image2 = Image.open(r’2.jpg‘)
image3 = Image.open(r’3.jpg‘)
image4 = Image.open(r’4.jpg‘)
```

Konvertatsiya qilinadi:

```
im1 = image1.convert(’RGB’)
im2 = image2.convert(’RGB’)
im3 = image3.convert(’RGB’)
im4 = image4.convert(’RGB’)
```

Keyin yangi tasvirlar ro‘yxati yaratiladi (birinchi rasmdan tashqari im1):

```
imagelist99 = [im2,im3,im4]
```

Dastur oxirida PDFga saqlash kodi yoziladi(im1ga e’tibor bering):

```
im1.save(r’55.pdf’,save_all=True, append_images=imagelist99)
```

To‘liq dastur kodi:

```
from PIL import Image
```

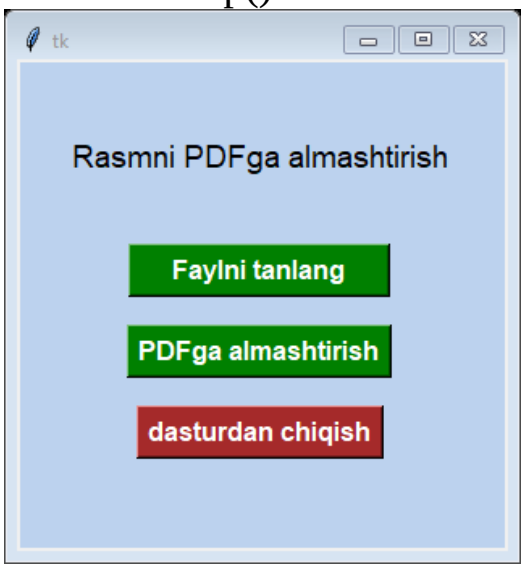
```
image1 = Image.open(r'1.jpg')
image2 = Image.open(r'2.jpg')
image3 = Image.open(r'3.jpg')
image4 = Image.open(r'4.jpg')
im1 = image1.convert('RGB')
im2 = image2.convert('RGB')
im3 = image3.convert('RGB')
im4 = image4.convert('RGB')
imagelist99 = [im2,im3,im4]
im1.save(r'55.pdf',save_all=True, append_images=imagelist99)
```

PDF faylimiz barcha rasmlarni o'z ichiga oladi.

Rasmlarni PDF ga aylantirish uchun vosita: Pytonning grafik interfeysi va tkinter paketiga asoslanib, rasmlarni PDFga almashtirish uchun vosita kodi:

```
from PIL import Image
import tkinter as tk
from tkinter import filedialog
from tkinter import messagebox
root= tk.Tk()
canvas1 = tk.Canvas(root, width = 300, height = 300, bg = 'lightsteelblue2', relief = 'raised')
canvas1.pack()
label1 = tk.Label(root, text='File Conversion Tool', bg = 'lightsteelblue2')
label1.config(font=('helvetica', 20))
canvas1.create_window(150, 60, window=label1)
def getFile ():
    global im1
    import_file_path = filedialog.askopenfilename()
    image1 = Image.open(import_file_path)
    im1 = image1.convert('RGB')
browseButton = tk.Button(text="      Select File     ", command=getFile, bg='green', fg='white', font=('helvetica', 12, 'bold'))
canvas1.create_window(150, 130, window=browseButton)
def convertToPdf ():
    global im1
    export_file_path =
filedialog.asksaveasfilename(defaultextension='.pdf')
```

```
    im1.save(export_file_path)
saveAsButton = tk.Button(text='Convert to PDF', command=convertToPdf, bg='green', fg='white', font=('helvetica', 12, 'bold'))
canvas1.create_window(150, 180, window=saveAsButton)
def exitApplication():
    MsgBox = tk.messagebox.askquestion ('Exit Application','Are you sure you want to exit the application',icon = 'warning')
    if MsgBox == 'yes':
       root.destroy()
exitButton = tk.Button (root, text='Exit Application', command=exit Application, bg='brown', fg='white', font=('helvetica', 12, 'bold'))
canvas1.create_window(150, 230, window=exitButton)
root.mainloop()
```



Nazorat savollari.

1. Rasmlarni import qilish qanday amalga oshiriladi?
2. Rasmlarni .pdf formatga o‘giring.

23-ma’ruza.Pythonda animatsiya.

Reja:

1. Pythonda animatsiya yaratish usullari va turlari.
2. Rasmlardan .gif yaratish.

Animatsiyalar vizualizatsiyani yanada jozibador qilish va foydalanuvchini jalb qilishning ajoyib usuli hisoblanadi. Bu bizga ma’lumotlar vizualizatsiyasini mazmunli tarzda namoyish qilishimizga yordam beradi. Python bizga mavjud kuchli Python kutubxonalaridan foydalangan holda animatsiya vizualizatsiyasini yaratishda yordam beradi. Matplotlib — bu juda mashhur ma’lumotlarni vizualizatsiya qilish kutubxonasi va odatda ma’lumotlarni grafik tasvirlash uchun, shuningdek, o‘rnatilgan funktsiyalardan foydalangan holda animatsiyalar uchun ishlatiladi.

Matplotlib yordamida animatsiya yaratishning ikki yo‘li mavjud:
Pause() funksiyasidan foydalanish
FuncAnimation() funksiyasidan foydalanish

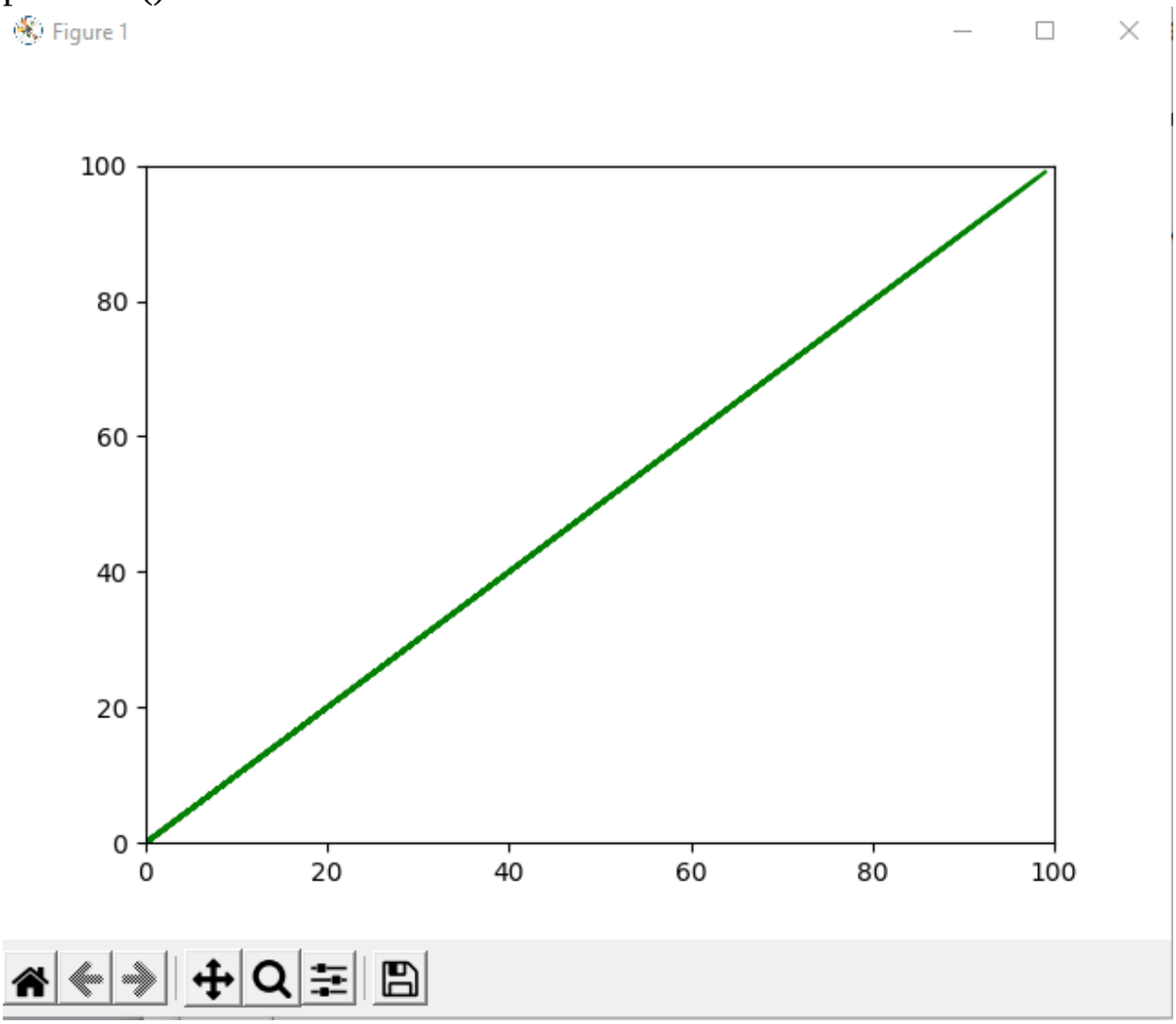
1-usul: pause() funksiyasidan foydalanish

Matplotlib kutubxonasining pyplot modulidagi pause() funksiyasi argumentda qayd etilgan oraliq soniyalar uchun pauza qilish uchun ishlatiladi. Quyidagi misolni ko‘rib chiqing, unda biz matplotlib yordamida oddiy chiziqli grafik yaratamiz va unda Animatsiyani ko‘rsatamiz:

X va Y 2 ta massiv yarating va 1 dan 100 gacha qiymatlarni saqlang.
plot() funksiyasidan foydalanib X va Y ni chizing.
Tegishli vaqt oralig‘i bilan pause() funksiyasini qo‘shing
Dasturni ishga tushiring va siz animatsiyani ko‘rasiz.
Misol:

```
from matplotlib import pyplot as plt
x = []
y = []
for i in range(100):
x.append(i)
y.append(i)
# Ularning diapazonini aniqlash uchun x va y chegaralarini eslatib o‘ting
plt.xlim(0, 100)
```

```
plt.ylim(0, 100)
# Grafik chizish
plt.plot(x, y, color = ’green’)
plt.pause(0.01)
plt.show()
```



Pillow Image Processing Library (PIL) yordamida siz Python-da jonlantirilgan GIF tasvirlarini yaratishingiz mumkin. Har bir freymni ochish va uning ob’ektini maxsus freymlar ro‘yxatiga qo‘shish uchun for va range tsiklidan foydalanamiz.

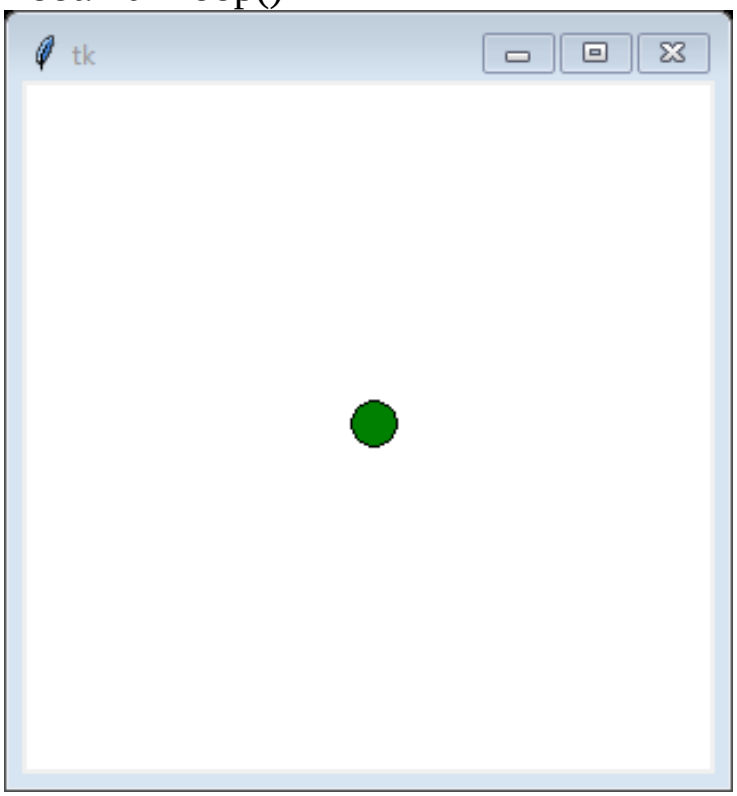
```
from PIL import Image
# Kadrlarni saqlash uchun ro‘yxat
frames = []
for frame_number in range(1, 11):
    # Har bir ramkaning rasmini oching
    frame = Image.open(f’homer-{frame_number}.jpg‘)
```

```
    # Kadrlar ro‘yxatiga ramka qo‘shing
    frames.append(frame)
# Birinchi ramkani oling va unga qolgan ramkalarni qo‘shing
frames[0].save(
    ’homer.gif’,
    save_all=True,
    append_images=frames[1:],        # Birinchi kadrni e’tiborsiz
qoldiradigan bo‘lak
    optimize=True,
    duration=100,
    loop=0
)
```

Tkinter moduli orqali animatsiya yaratish:

```
from tkinter import *
root = Tk()
c = Canvas(width=300, height=300,
           bg=’white’)
c.focus_set()
c.pack()
 ball = c.create_oval(140, 140, 160, 160,
                      fill=’green’)
c.bind(’<Up>’,
       lambda event: c.move(ball, 0, -2))
c.bind(’<Down>’,
       lambda event: c.move(ball, 0, 2))
```

```
c.bind('<Left>',
       lambda event: c.move(ball, -2, 0))
c.bind('<Right>',
       lambda event: c.move(ball, 2, 0))
 root.mainloop()
```



Nazorat savollari.

1. Pythonda animatsiya yaratish usullarini sanab o‘ting?
2. Rasmlardan .gif yarating?

24-25-ma’ruza. PyGame moduli.

Reja:

1. Pygame modulini o‘rnatish.
2. Pygame moduli yordamida o‘yin dasturlarini yaratish.

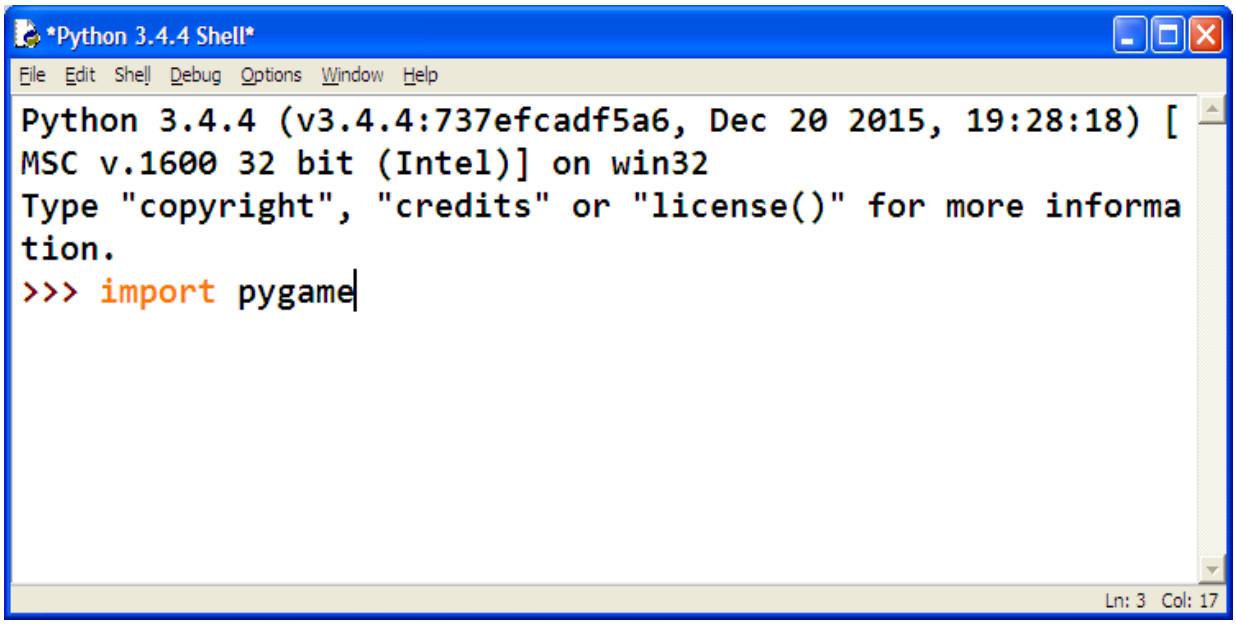
PyGame moduli. Bu *Python* dasturlash tilining interfeysli o‘yinlarni yaratish uchun maxsus mo‘ljallangan kutubxona . Biz Python tilida (3 versiya) oddiy 2D o‘yinlarni qanday yaratishni bilib olamiz. Biz o‘yinni animatsiya, sprites (rasmlar), funksionallik va grafik interfeys bilan yaratamiz. Ushbu kutubxona asosida ko‘plab o‘yinlar va ilovalar allaqachon yaratilgan. Ommabop o‘yinlarning katta ro‘yxatini ularning rasmiy veb-saytida ko‘rishingiz mumkin.

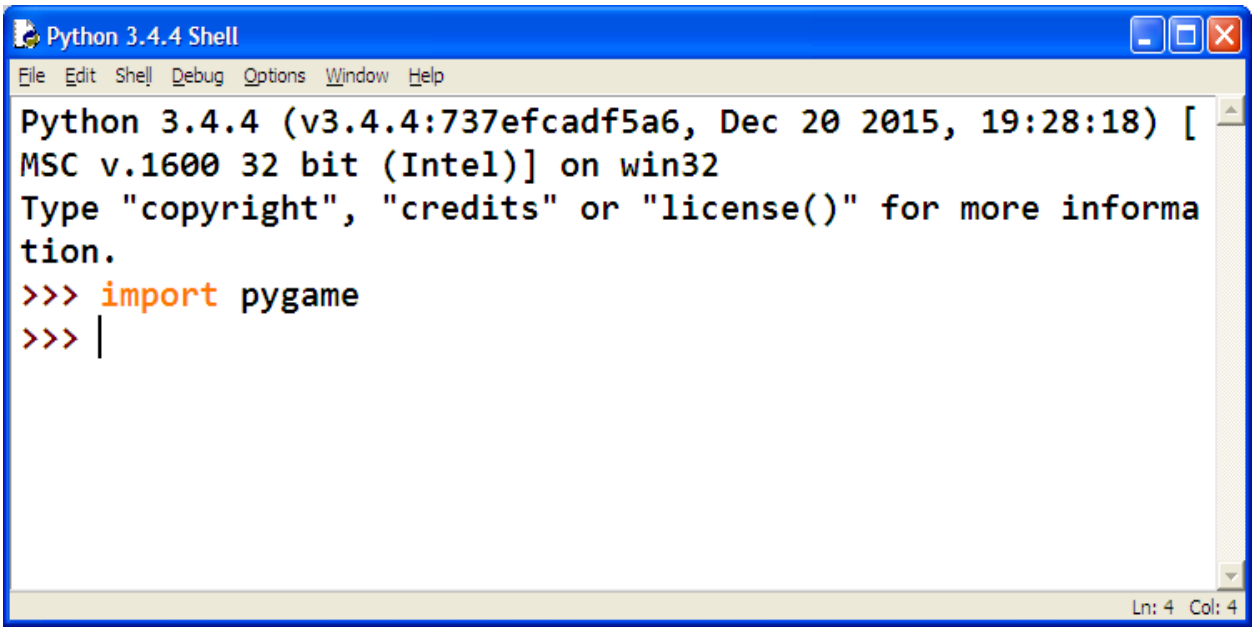
Pygame- ni o‘rnatish

Pygame Python bilan birga dasturda o‘rnatilmaydi. Python dastur kabi, Pygame ham bepul. Bu dasturda Pygame ni o‘rnatish shart bo‘ladi va uni yuklab olib o‘rnatamiz. Pygamani yuklab olish va o‘rnatish, Pythonni yuklab olish va o‘rnatish kabi juda oson. Veb-brauzerda http://pygame.org URL manziliga o‘tamizva "Downloads" ni bosamiz veb-saytning chap tomonidagi havola. Ushbu kitob sizga Windows operatsion tizimiga ega,ammo Pygame har bir operatsion tizim uchun bir xil ishlaydi. Pygame-ni yuklab olishingiz kerak operatsion tizimingiz uchun o‘rnatuvchi va o‘rnatgan Python versiyasi.

Siz Pygame uchun "Source" ni emas, balki sizning Pygame-“binary” ni yuklab olishnixohlamaysiz operatsiontizim. Windows uchun pygame-1.9.1.win32- py3.2.msi faylini yuklab oling. (Bu Windows uchun Python 3.2 uchun Pygame. Pythonning boshqa versiyasini (masalan, 2.7 yoki 2.6) Python versiyasi uchun .msifaylini yuklab oling.) Pygame ning joriy versiyasi Bu kitob 1.9.1 da yozilgan. Agar veb-saytda yangi versiyani ko‘rsangiz, yuklab oling va yangi Pygame-ni o‘rnating. Windowsda Pygame-ni o‘rnatish uchun yuklab olingan faylga ikki marta bosing.

Pygame-ni tekshirish uchun to‘g‘ri o‘rnatish, interaktiv qobiqga quyidagilarni yozing:





Salom dunyo uchun Pygame bilan Source Code

Pygame bilan tuzilgan birinchi dasturimiz - bu derazani ochadigan kichik dastur

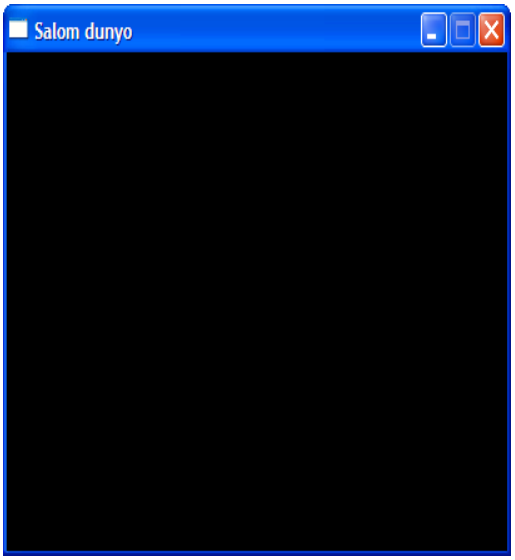
Dunyo!! "Ekranda paydo bo‘ladi. IDLE ning File menyusiga bosish orqali yangi fayl muharriri oynasini oching.

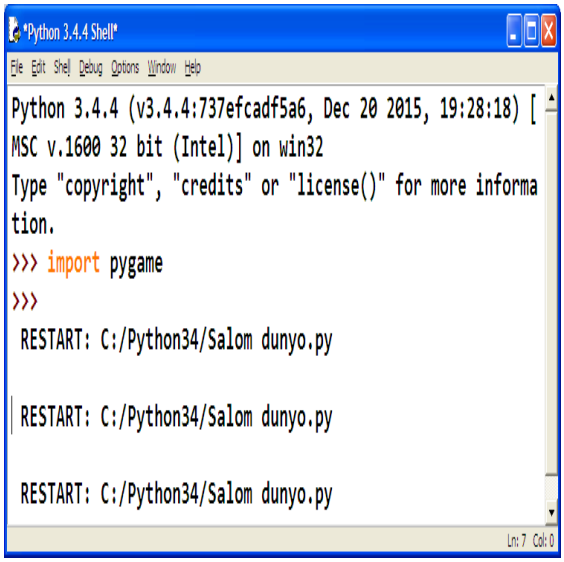
Keyin yangi oyna. Quyidagi kodni IDLE-ning fayl muharriri ichiga yozing va yozib oling blankpygame.py. Keyin dasturni F5 tugmasini bosib yoki Run dan=> Run Modulni tanlang. Fayl muharriri ustidagi menyu.

Esda tuting, har bir satrning boshida raqamlar yoki davrlarni yozmang (faqatginabu Ushbu kitobda namunalar).

```
import pygame, sys
from pygame.locals import*
pygame.init()
DISPLAYSURF=pygame.display.set_mode((400,300))
pygame.display.set_caption("Salom dunyo")
while True:
    for event in pygame.event.get():
        if event.type==QUIT:
            pygame.quit()
            sys.exit()
        pygame.display.update()
```

Ushbu dasturni ishga tushirganda, shu kabi qora oyna paydo bo‘ladi:





Endi biz dunyoning eng zerikarli video o‘yinini yaratdik! Bu shunchaki bo‘sh oyna, xolos ("Dunyo!") Oynasining yuqori qismida ("window" ning nom satri deb nomlanadi) sarlavhali matn). Biroq oyna yaratish — bu grafik o‘yinlarni amalga oshirish uchun birinchi qadamdir. Chertganingizda oynaning burchagidagi X tugmachasida dastur tugaydi va oyna yo‘qoladi.

Matnni matn oynasida ko‘rsatish uchun print () funktsiyasini chaqirish, chunki ishlamaydi print () - CLI dasturlari uchun funktsiya. Klaviatura kiritish uchun kirish () ga ham xuddi shundayfoydalanuvchidan. Pygame kirish va chiqish uchun boshqa funktsiyalardan foydalanadi bo‘limiga qarang. Hozirda, "Dunyo Dunyosi" dasturining har bir satrini batafsil ko‘rib chiqaylik.

Pygame dasturini o‘rnatish

"Salom dunyo" dasturidagi dastlabki kodlar soni deyarli har biriga boshlanadiPygame ishlatadigan dasturni yozing.

```
1. import pygame, sys
```

1-satr bizning pygame va sys modullarini import qiladigan sodda import

ko‘rsatgichidir dastur o‘zida bu funktsiyalardan foydalanishlari mumkin. Barcha pygame vazifalari grafik, ovoz,va pygame taqdim etgan boshqa xususiyatlar pygamemodulida mavjud. pygame modulini import qilganingizda siz avtomatik ravishda barcha modullarni import qilishni unutmang pygame.images va pygame.mixer.music kabi pygame modulida ham mavjud.Ushbu modullarni importmodullarini import qilishning hojati yo‘q.

```
2. from pygame.locals import *
```

2-satr, shuningdek, *import* bayonnomasi. Biroq, *import modulename* formati

o‘rniga, u *modulename import* * formatidan foydalanadi. Odatda agar siz bu funksiyani chaqirishni istasangiz modulda bo‘lsa, importdan so‘ng *modulename.functionname* () formatini ishlatishingiz kerak moduli. Biroq, *modulename import* * dan *modulename* nomlayabsiz.qismini ishlatish va shunchaki *functionname* () funksiyasidan foydalaning (xuddi Pythonning o‘rnatilgan funksiyalari singari). Biz *pygame.locals* uchun *import* so‘rovi bu shaklini ishlatish sababi shudir *pygame.locals* bir qator doimiy parametrlarga ega bo‘lib, ularni aniqlash oson *pygame.locals* holda *pygame.locals* moduli. oldida. Boshqa barcha modullar uchun, odatda muntazam *import modulename* formatidan foydalanishni xohlaysiz.

```
4. pygame.init()
```

satr *pygame.init* () funktsiyasi chaqiruvi bo‘lib, uni importdan so‘ng har doim

chaqirish kerak *pygame* moduli va boshqa *pygame* funksiyasini chaqirishdan oldin.Nimani bilishingiz kerak emas bu vazifani bajaradi, shuning uchun ko‘pgina *pygame*uchun avvalo uni chaqirish kerakligini bilishingiz kerak ishlash uchun vazifalar. Agar *pygame.error* kabi xato xabari ko‘rinsa: shrift yo‘q boshlanganida, sizning boshingizdagi

pygame.init () ga qo‘ng‘iroq qilishni unutdingizmi yoki yo‘qligini tekshiring dasturi.

```
5. DISPLAYSURF = pygame.display.set_mode((400, 300))
```

qator - bu *pygame.display.set_mode* () funktsiyasiga chaqiruv window uchun pygame.Surface obyekti. (Er yuzidagi ob’ektlar ushbu bobningoxirida tasvirlangan.). Shuni e’tiborga olish kerakki, biz funksiya uchun ikkitatamsayıning bir tayanch qiymatini o‘tkazamiz: (400, 300). Bu tuple aytadi *set_mode* () funksiyasi oynani piksellarda qanchalik keng va qanchalik balandqilishini aniqlang. (400, 300) kengligi 400 piksel va balandligi 300 pikselli deraza yaratadi. Set_mode () funktsiyasiga ikkita tamsaytning ikkita sonini emas, balkiikkita tamsayını kiritishni unutmang. Funktsiyani chaqirishning to‘g‘ri usuli quyidagicha: *pygame.display.set_mode ((400, 300)). Pygame.display.set_mode (400, 300)* kabi funksiya chaqiruvi bu xatoga olib keladi quyidagicha ko‘rinadi: *TypeError: argument 1 int*-emas, 2-bandli tartib bo‘lishi kerak. *Pygame.Surface* obyekti (biz ularni faqat ularni "Surface" ob’ektlarini qisqacha deb atashadi) qaytib keldi *DISPLAYSURF* nomli o‘zgaruvchida saqlanadi.

```
6. pygame.display.set_caption(’Hello World!’)
```

satrda derazaning yuqori qismida paydo bo‘ladigan matn matni chaqiriladi *pygame.display.set_caption* () funktsiyasi. "Salom dunyo!" bo‘ladi ushbu matnni matn nomi sifatida ko‘rsatish uchun ushbu funktsiya chaqiruvidano‘tgan:

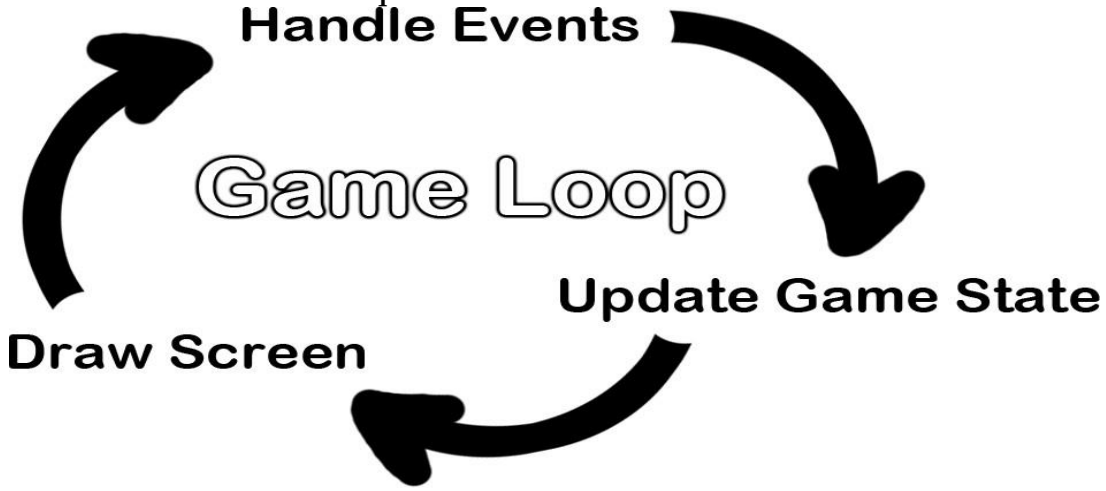
Game Loops and Game States

```
7. while True: # main game
8.     for event in
```

satr oddiy qiymatli "True" shartiga ega bo‘lgan vaqtli pastadir. Bu hech qachon demaydi uning holati shubhali holatlarga qarab baholanadi. Dasturni bajarishning yagona usuli *loop* dan chiqib ketish, agar buzilish bayonoti amalga oshirilsa (ijrodan keyingi birinchi qatorga o‘tishni amalga oshiradi) loop) yoki *sys.exit* () dasturini tugatadi. Agar bunday funktsiyaning bir

funktsiyasi ichida bo‘lsa, orqaga qaytish iborasi ijroni ko‘chadan chiqarib tashlaydi (shuningdek, funktsiya ham). Bu kitobdagi o‘yinlarning barchasi bularning barchasini o‘z ichiga oladi bu — o‘yinning aylanishi ". O‘yin loopi (asosiy pastadir deb ham ataladi) kodning bajarilishida pastadir uch narsa:
Voqealarni ushlaydi.
O‘yin holati yangilanadi.
O‘yin holatini ekranga tortadi.

O‘yin holati oddiygina o‘yindagi barcha o‘zgaruvchilar uchun bir qator qadrlarni nazarda tutadi dasturi. Ko‘pgina o‘yinlarda o‘yin davlati kuzatuvchi o‘zgaruvchan qiymatlarni o‘z ichiga oladi o‘yinchilarning salomatligi va mavqei, har qanday dushmanning sog‘lig‘i va mavqei, qaysi belgilar bo‘lganligi bortda, skorda yoki kimga aylanadi. O‘yinchilar kabi biror narsa sodir bo‘lganda (ularning sog‘lig‘ini pasaytiradigan) yoki dushman biror joyga yoki biror narsaga harakat qiladi o‘yin dunyosida sodir bo‘layotgani, o‘yin davlati o‘zgardi. Agar siz hech qachon saqlab qo‘ygan o‘yinni o‘ynab ketsangiz, "o‘ynash holati" — bu o‘yinning holati uni saqlab qoldingiz. Ko‘pgina o‘yinlarda o‘yinni pauza qilish o‘yin holatining o‘zgarishiga to‘sqinlik qiladi. O‘yin holati odatda voqealarga (masalan, sichqoncha yoki klaviatura) javoban yangilanadi presslar) yoki vaqt o‘tishi bilan, o‘yin aylanishi doimo tekshiriladi va ko‘p marta qayta tekshiriladi boshlangan har qanday yangi voqealar uchun ikkinchi o‘rinda turadi. Asosiy pastadir ichida qaysi kodga qarash kodi voqealar yaratilgan (Pygame bilan, bu *pygame.event.get* () funktsiyasi). Asosiy ko‘chadan, shuningdek, o‘yin holatini qaysi voqealarga asoslanganligini yangilaydigan kodi ham bor yaratilgan. Bu odatda tadbirni boshqarish deb nomlanadi.



pygame.event.Event obyektlari

Foydalanuvchilar bir necha harakatlardan birini (ular ushbu bobning keyingi qismida keltirilgan) bajarishi bilan har qanday vaqtda klaviatura tugmachasinibosish yoki dasturning oynasiga sichqonchani suring, apygame.event.Event obyekti"-event" ni yozish uchun Pygame kutubxonasi tomonidan yaratilgan. (Bu Voqealardeb ataladigan voqea deb nomlanadigan ob'ekt turi, u o'zi pygame ichida bo'lgan voqea modulida mavjudmoduli). Quyidagi voqealarni topish mumkin: *pygame.event.get* () funktsiya, pygame.event.Event obyektlari ro'yxatini qaytaradi qisqa).Voqealar moslamalari ro'yxati oxirgi marta sodir bo'lgan har bir voqea uchun bo'ladi *pygame.event.get* () funktsiyasi chaqirildi. (Yoki agar pygame.event.get () hech qachon bo'lmaganida) Dastur boshlanganidan buyon sodir bo'lgan voqealar.)

```
7. while True: # main game loop
8.     for event in
```

satr qaytarilgan Voqealar moslamalari ro'yhatining ustida yineluvchi loop uchun *pygame.event.get* (). Har bir iteratsiya uchun for loop orqali voqea deb nomlanuvchio'zgaruvchi ushbu ro'yxatdagi keyingi voqea obyektining qiymati tayinlanadi. Voqealar moslamalari ro'yxati qaytdi *pygame.event.get* () dan voqealar yuz bergan tartibda bo'ladi. Agar foydalanuvchi chertgan bo'lsa sichqoncha va keyin klaviatura tugmachasini bosgan holda, Sichqoncha tugmasini bosish uchun Event ob'ekti bo'ladi ro'yxatdagi birinchi element va klaviatura matni uchun Voqealar obyekti ikkinchi bo'ladi. Voqealar bo'lmasa *pygame.event.get* () bo'sh ro'yxatni qaytaradi.

QUIT voqea va pygame.quit () funktsiyasi

```
9.  if event.type == QUIT:
10.     pygame.quit()
11.     sys.exit()
```

Voqealar moslamalarni turi o'zgaruvchiga (shuningdek, atributlar yoki xususiyatlar deb ataladigan) ega bu bizga ob'ekt nimani ifodalaydi. Pygame har biruchun doimiy o'zgaruvchiga ega pygame.locals modullarida mumkin bo'lgan turlari. 9-satrda Voqealar

obyektining turi tekshiriladi doimiy QUITga teng. Esingizda bo‘lsa, biz *pygame.locals* dan foydalanganimizdan beri *import * import* bayonnomasi shakli o‘rniga, faqat *QUIT* yozing *pygame.locals.QUIT*.

Voqealar obyekti yopilish hodisasi bo‘lsa, *pygame.quit* () va *sys.exit* () funktsiyalari chaqirdi. *pygame.quit* () funktsiyasi *pygame.init* () ning teskarisidir.vazifasi: Pygame kutubxonasini o‘chirib qo‘yadigan kod ishlaydi. Sizning dasturlari doimo qo‘ng‘iroq qilish kerak Dasturni tugatish uchun *sys.exit* ()ni chaqirishdan oldin *pygame.quit* () funktsiyasini ishlatadi. Odatda bunday emas Python dasturi har qanday holatda chiqqandan keyin yopilgandan beri juda muhimdir. Biroq, IDLE da xato bor Pygame dasturi *pygame.quit* () funktsiyasidan oldin bekor qilinsa, to‘xtatishga sababbo‘ladi.

Bizda boshqa ob’ektlar uchun kodni ishlatadigan so‘zlar yo‘q ekan, foydalanuvchi sichqonchani bosganida, klaviatura tugmacha-larini bosib yoki boshqabiror narsaga olib kelishi uchun hech qanday voqea kodi mavjud emas. yaratiladiganvoqealar moslamalarni turini. Foydalanuvchining, bu voqealar moslamalarni yaratish uchun narsalarni qila oladi, lekin bu dasturda hech narsa o‘zgarmaydi, chunki dasturda hech qanday hodisa sodir bo‘lmaydi Ushbu turdagi Voqealar moslamalarni uchun kod. 8-satrda forma aylantirilgandan so‘ng barcha tadbirlarni ko‘rib chiqiladi *pygame.event.get* () tomonidan qaytarilgan ob’ektlar, dasturni bajarish davom etadi 12-qatorga o‘tish.

```
12.  pygame.display.update()
```

12-satr Surface ob’ektini jalb qiluvchi pygame.display.update () funktsiyasini
chaqiradi *pygame.display.set_mode* () yordamida ekranga qaytdi (bu ob’ektni saqlaymiz.

DISPLAYSURF o‘zgaruvchisida). Surface ob’ekti o‘zgarmasligi sababli (masalan, ba’zi tomonidan Bu bobda keyinroq tushuntirilgan chizilgan funktsiyalari), xuddi shu qora tasvir qayta ishlanadi *pygame.display.update* () funktsiyasini har safar ekranda ko‘rsatishga harakat qiladi. Bu butun dastur. 12-satr tugagandan so‘ng, cheksiz tugma loop yana qaytadan boshlanadi boshlanishi. Ushbu dastur ekranda qora oyna paydo bo‘lishidan tashqari hech narsa qilmaydi, muntazamQUIT hodisasini tekshiring va keyin o‘zgarmas qora oynani ekranga qaytaring qayta-qayta. Keling, bu oynada qiziqarli narsalar qanday paydo bo‘lishini bilib olaylikpiksellar, yuza moslamalari, rang

moslamalari, Rect obyektlari va Pygamanichizish funktsiyalari.

Pygame kutubxonasining asosiy modullari

Modul	Vazifasi
pyame.cdrom	CD-disklarga kirish va boshqarish
pyame.cursors	Kursor rasmlarini yuklaydi
pygame.display	Displeyga kirish
pygame.draw	Shakllar, chiziqlar va nuqta chizadi.
pygame.event	Tashqi hodisalarni boshqarish
pygame.font	Tizim shriftlaridan foydalanadi
pygame.image	Tasvirni yuklaydi va saqlaydi
pygame.joystick	Joystiklar va shunga o‘xshash qurilmalardan foydalanadi
pygame.key	Klaviaturadan tugmalarni o‘qiydi
pygame.mixer	Musiqalarni yuklab oling va ijro eting
pygame.mouse	Sichqoncha elementlari
pygame.movie	Video fayllarni o‘ynatish
pygame.music	Musiqa va audio oqimlari bilan ishlaydi
pygame.overlay	Kengaytirilgan videoga kirish
Pigame	Yuqori daRejadagi Pygame funktsiyalarini o‘z ichiga oladi
pygame.rect	To‘rtburchaklar maydonlarni boshqaradi
pygame.sndarray	Ovoz ma’lumotlarini boshqaradi
pygame.sprite	Kinofilmlarni boshqarish
pygame.surface	Rasmlar va ekranni boshqaradi
pygame.surfarray	Rasm pikseli ma’lumotlarini boshqaradi
pygame.time	vaqt va kadrlar tezligini boshqarish uchun pygame moduli
pygame.transform	Tasvirlarning o‘lchamlari va o‘lchamlari

Rasm yuklash. Pygame.image moduli sizga fayldan rasmni yuklash va Surface turidagi ob’ektni qaytarishga imkon beradi.
pygame.image.load ("fayl yo‘li"): fayldan yangi rasmni yuklash
Rect usullari

pyame.Rect.copy	—	Asl nusxasi bilan bir xil pozitsiyaga va hajmga ega bo‘lgan yangi to‘rtburchakni qaytaradi.
pygame.Rect.move	—	Ushbu ofset tomonidan ko‘chirilgan yangi to‘rtburchakni qaytaradi. X va y argumentlar har qanday musbat yoki musbat son bo‘lishi mumkin.
pygame.Rect.move_ip	—	Rect.move () usuli bilan bir xil, ammo joyida ishlaydi.
pygame.Rect.inflate	—	joyida to‘rtburchaklar hajmini oshirish yoki kamaytirish
pygame.Rect.inflate_ip	—	joyida to‘rtburchaklar hajmini oshirish yoki kamaytirish
pygame.Rect.clamp	—	to‘rtburchakni boshqasi ichida harakatlantiradi
pygame.Rect.clamp_ip	—	to‘rtburchakni boshqa joyga, joyiga ko‘chiradi
pygame.Rect.clip	—	to‘rtburchagini boshqasi ichida kesib tashlang
pygame.Rect.union	—	ikkita to‘rtburchaklar bog‘laydi
pyame.Rect.union_ip	—	ikkita to‘rtburchaklar bir-biriga joyida bog‘laydi
pygame.Rect.unionall	—	ko‘plab to‘rtburchaklar birligi

pyame.Rect.unionall_ip	—	joyida koʻplab toʻrtburchaklar birligi
pygame.Rect.fit	—	qirralarning nisbatlarini hisobga olgan holda toʻrtburchaklar hajmini oʻzgartiring va siljiting
pygame.Rect.normalize	—	salbiy oʻlchamlarni oʻrnating
pyame.Rect.contains	—	bitta toʻrtburchaklar ikkinchisining ichida yoki yoʻqligini tekshiring
pygame.Rect.collidepoint	—	nuqta toʻrtburchaklar ichida yoki yoʻqligini tekshiring
pygame.Rect.colliderect	—	Agar ikkita toʻrtburchaklar kesishgan boʻlsa, sinov qiling
pyame.Rect.collidelist	—	roʻyxatdagi kamida bitta toʻrtburchaklar kesishganligini tekshiring
pyame.Rect.collidelistall	—	roʻyxatdagi barcha toʻrtburchaklar kesishadi
pygame.Rect.collidedict	—	lugʻatda bitta toʻrtburchak kesishganligini tekshiring
pygame.Rect.collidedictall	—	lugʻat bilan kesishishda barcha toʻrtburchaklar qiling

Hodisalarni boshqarish. Voqea Pygame dastur kodidan tashqarida biron bir narsa yuz berganligi haqida xabar beradi. Hodisalar, masalan, klaviatura, sichqonchani bosib, ishlov berilishini kutib, navbatga qoʻyiladi. Pygame.event modulidagi get funktsiyasi navbatda turgan oxirgi voqeani qaytaradi va uni navbatdan olib tashlaydi.

Hodisa obʼekti. Voqealar navbatini qayta ishlash uchun modul

pygame.event.pump	—	Agar siz oʻz oʻyiningizda boshqa voqea funktsiyalaridan foydalanmasangiz, pygame-ga

		ichki amallarni bajarishga ruxsat berish uchun pygame.event.pump () ni chaqirishingiz kerak.
pyame.event.get	—	navbatdan voqealarni qabul qiladi
pyame.event.poll	—	bitta voqeani navbatdan oling
pygame.event.wait	—	navbatdan bitta tadbirni kutish
pygame.event.peek	—	navbatlar ma’lum bir turdagi voqealarni kutayotganligini tekshiring
pygame.event.clear	—	barcha voqealarni navbatdan olib tashlash
pyame.event.event_name	—	voqea turi uchun nomni qaytaradi. Satr WordCap uslubida.
pygame.event.set_blocked	—	navbatda qaysi voqealarga ruxsat berilmaganligini tekshiradi
pygame.event.set_allowed	—	navbatda qanday voqealarga ruxsat berilganligini tekshiradi
pygame.event.get_blocked	—	voqea turi navbatdan bloklanganligini tekshiring
pyame.event.set_grab	—	kirish qurilmalarini boshqa dasturlar bilan almashishini tekshiradi
pyame.event.get_grab	—	dastur ma’lumot kiritish qurilmalarida ishlayotganligini tekshiring
pyame.event.post	—	yangi tadbirni navbatga qo‘yish

pygame.event.Event	—	yangi voqea ob’ekti yaratish
pygame.event.EventType	—	SDL hodisasini bildiruvchi Python ob’ekti. Maxsus voqea misollari "Voqealar" funktsiyasini chaqirish orqali yaratiladi. EventType turini to‘g‘ridan-to‘g‘ri chaqirib bo‘lmaydi. EventType misollari atributlarni tayinlash va yo‘q qilishni qo‘llab-quvvatlaydi.

Pygame voqea haqidagi barcha xabarlarni voqea navbatida kuzatib boradi. Ushbu moduldagi muolajalar ushbu voqea navbatini boshqarishga yordam beradi. Kirish kuchi pygame displey moduliga juda bog‘liq. Agar displey ishga tushirilmagan bo‘lsa va video rejimi o‘rnatilmagan bo‘lsa, voqealar navbati ishlamaydi. Hodisalar navbatiga kirishning ko‘plab usullari mavjud.

pygame.mouse.get_pressed	—	sichqonchaning tugmachalari holatini oling
pygame.mouse.get_pos	—	sichqoncha kursorini olish
pygame.mouse.get_rel	—	sichqoncha harakatlarining sonini oling
pygame.mouse.set_pos	—	sichqoncha kursorini o‘rnating
pygame.mouse.set_visible	—	kursorni yashirish yoki ko‘rsatish
pygame.mouse.get_focused	—	displey sichqoncha kiritishni qabul qiladimi yoki yo‘qligini tekshiradi
pygame.mouse.set_cursor	—	sichqoncha kursoriga rasm o‘rnatish
pygame.mouse.get_cursor	—	sichqoncha kursoriga rasm olish

Sichqoncha funktsiyalaridan sichqoncha qurilmasining joriy holatini olish mumkin. Ushbu funktsiyalar sichqoncha kursorini ham

o‘zgartirishi mumkin. Displey rejimi o‘rnatilganda, voqealar navbatida sichqonchani tadbirlari qabul qilinadi. Sichqoncha tugmachalari pygame. MOUSEBUTTONDOWN va pygame. MOUSEBUTTONUP voqealarini, ular bosilganda va chiqarilganda yaratadi. Ushbu hodisalar qaysi tugmachani bosganligini ko‘rsatuvchi tugmaviy atributdan iborat. Sichqoncha g‘ildiragi pygame hosil qiladi. MOUSEBUTTONDOWN va pygame.MOUSEBUTTONUP harakat-lanayotganda.

G‘ildirak aylantirilganda, tugma 4 ga, pastga -5 ga o‘rnatiladi. Sichqoncha har safar harakat qilganda, pygame.MOUSEMOTION hodisasi o‘chiriladi. Sichqoncha harakati kichik va aniq harakat hodisalariga bo‘linadi. Sichqoncha harakatlantirilganda, ko‘plab harakat voqealari navbatga qo‘yiladi. Hodisalar navbatini noto‘g‘ri olib tashlangan sichqonchani harakatga keltirish voqealar navbatining to‘lishiga asosiy sababdir.

Klaviatura. Ushbu modul klaviatura bilan ishlash uchun funktsiyalarni o‘z ichiga oladi va klaviatura tugmachalarini bosganingizda va bo‘shatganingizda voqealar navbatiga pygame. KEYDOWN va pygame.KEYUP hodisalari kiradi. Ikkala hodisada ham klaviaturadagi har bir tugmachani ifodalovchi butun identifikator bo‘lgan kalit atributi mavjud.Pygame.KEYDOWN hodisasi qo‘shimcha atributlarga ega: unicode va scancode. unicode bu kiritilgan belgilarga mos keladigan bitta belgi satri.

Ko‘p klaviatura konstantalari mavjud, ular klaviaturadagi tugmalarni ifodalash uchun ishlatiladi.

Quyidagi barcha klaviatura doimiylarining ro‘yxati:

Keyastri	Ascii	Umumiy nom
K_BACKSPACE	b	backspace
K_TAB	T	Yorliq
K_CLEAR	Ravshan	
K_RETURN	R	Qaytish
K_PAUSE	Pauza	
K_ESCAPE	^[	Qochish

K_SPACE	bo‘sh joy	
K_EXCLAIM	!	deb xitob qilmoq
K_QUOTEDBL	"	Tirnoq
K_HASH	#	Xesh
K_DOLLAR	$	Dollar
K_AMPERSAND	&	ampersand
K_QUOTE	Kotirovka	
K_LEFTPAREN	(	chapparentez
K_RIGHTPAREN	)	o‘ngparentez
K_ASTERISK	*	yulduzcha
K_PLUS	+	Plussign
K_COMMA	,	Vergul
K_MINUS	-	minussign
K_PERIOD	.	Davri
K_SLASH	/	oldinga siljish
K_0	0	0
K_1	1	1
K_2	2	2
K_3	3	3
K_4	4	4
K_5	5	5
K_6	6	6
K_7	7	7

K_8	8	8
K_9	9	9
K_COLON	:	yo‘g‘on ichak
K_SEMICOLON	,	nuqta-vergul
K_LESS	tejamkorlikdan kam	
K_EQUALS	=	tenglashtirish
K_GREATER	>	katta qiymat
K_QUESTION	?	savol belgisi
K_AT	@	Da
K_LEFTBRACKET	[	chap qo‘ltiqcha
K_BACKSLASH		teskari chiziq
K_RIGHTBRACKET	]	o‘ng qator
K_CARET	^	Karet
K_UNDERSCORE	_	pastki chiziq
K_BACKQUOTE	’	Qabr
K_a	A	A
K_b	B	B
K_c	V	V
K_d	D	D
K_e	E	E
K_f	F	F
K_g	G	G
K_h	H	H

K_i	I	I
K_j	J	J
K_k	K	K
K_l	L	L
K_m	M	M
K_n	N	N
K_o	O	O
K_p	P	P
K_q	Q	Q
K_r	R	R
K_s	S	S
K_t	T	T
K_u	U	U
K_v	V	V
K_w	W	W
K_x	X	X
K_y	Y	Y
K_z	Z	Z
K_DELETE	o‘chirish	
K_KP0	klaviatura0	
K_KP1	klaviatura1	
K_KP2	klaviatura2	
K_KP3	klaviatura3	

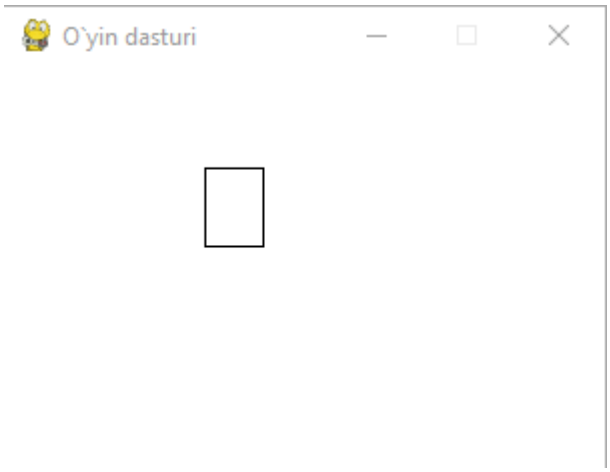
K_KP4	klaviatura4	
K_KP5	klaviatura5	
K_KP6	klaviatura6	
K_KP7	klaviatura7	
K_KP8	klaviatura8	
K_KP9	klaviatura9	
K_KP_PERIOD	.	klaviatura davri
K_KP_DIVIDE	/	keypaddivide
K_KP_MULTIPLY	*	klaviatura paneli
K_KP_MINUS	-	klaviatura
K_KP_PLUS	+	klaviatura
K_KP_ENTER	R	klaviatura
K_KP_EQUALS	=	klaviaturalar
K_UP	ko‘tarilish	
K_DOWN	pastga tushish	
K_RIGHT	o‘ng tomonda	
K_LEFT	Chapakcha	
K_INSERT	Joylashtiring	
K_HOME	Uy	
K_END	Oxiri	
K_PAGEUP	Pageup	
K_PAGEDOWN	Pagedown	
K_F1	F1	
K_F2	F2	

K_F3	F3
K_F4	F4
K_F5	F5
K_F6	F6
K_F7	F7
K_F8	F8
K_F9	F9
K_F10	F10
K_F11	F11
K_F12	F12
K_F13	F13
K_F14	F14
K_F15	F15
K_NUMLOCK	Numlock
K_CAPSLOCK	Capslock
K_SCROLLOCK	Scrollock
K_RSHIFT	Rightshift
K_LSHIFT	Leftshift
K_RCTRL	Rightcontrol
K_LCTRL	Leftcontrol
K_RALT	Rightalt
K_LALT	Leftalt
K_RMETA	Rightmeta
K_LMETA	Leftmeta

K_LSUPER	leftWindowskey
K_RSUPER	rightWindowskey
K_MODE	Modeshift
K_HELP	Help
K_PRINT	Printscreen
K_SYSREQ	Sysrq
K_BREAK	Break
K_MENU	Menu
K_POWER	Power
K_EURO	Euro

Pygame oynasini hosil qilish va unga to‘rtburchak chizish:

```
import pygame
background_colour = (255,255,255)
(width, height) = (300, 200)
color=(0,0,0)
screen = pygame.display.set_mode((width, height))
pygame.display.set_caption(’O‘yin dasturi’)
screen.fill(background_colour)
pygame.draw.rect(screen, color, (100,50,30,40), 1)
pygame.display.update()
running = True
while running:
  for event in pygame.event.get():
    if event.type == pygame.QUIT:
      running = False
        pygame.quit()
```

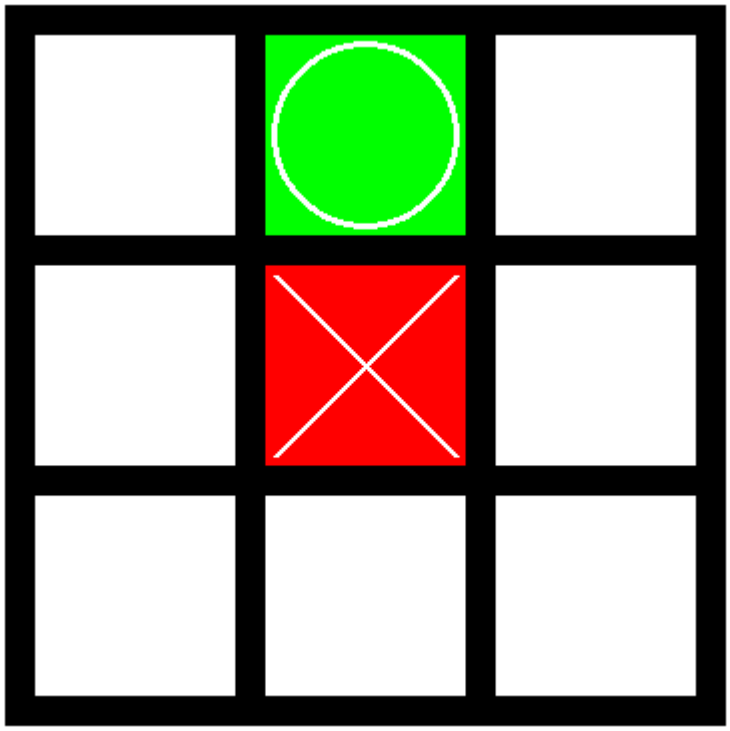


Tez top o‘yini: import pygame
import sys

```
def check_win(mas, sign):
    zeros = 0
    for row in mas:
        zeros+=row.count(0)
        if row.count(sign) == 3:
            return sign

    for col in range(3):
        if mas[0][col] == sign and mas[1][col] == sign and mas[2][col]
== sign:
            return sign
    if mas[0][0] == sign and mas[1][1] == sign and mas[2][2] == sign:
        return sign
    if mas[0][2] == sign and mas[1][1] == sign and mas[2][0] == sign:
        return sign
    if zeros == 0:
        return "Prise"
    return False
pygame.init()
size_block = 100
margin = 15
width = height = 3*size_block + 4*margin

size_window = (width, height)
screen = pygame.display.set_mode(size_window)
```

```
pygame.display.set_caption("Tez top")

black =(0, 0, 0)
red =(255, 0, 0)
green = (0, 255, 0)
white = (255, 255, 255)
mas = [[0]*3 for i in range(3)]
query = 0
game_over = False
while True:
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            pygame.quit()
            sys.exit(0)
        if event.type == pygame.MOUSEBUTTONDOWN:
            x_mouse, y_mouse = pygame.mouse.get_pos()
            col = x_mouse // (size_block+margin)
            row = y_mouse // (size_block+margin)
            if mas[row][col] == 0:
                if query%2 == 0:
                    mas[row][col] = 'x'
                else:
                    mas[row][col] = 'o'
                query+=1
        if event.type == pygame.KEYDOWN and event.key ==
pygame.K_SPACE:
            game_over = False
            mas = [[0]*3 for i in range(3)]
            query = 0
            screen.fill(black)
    if not game_over:
        for row in range(3):
            for col in range(3):
                if mas[row][col] == 'x':
                    color = red
                elif mas[row][col] == 'o':
                    color = green
                else:
```

```
                color = white
            x = col * size_block + (col + 1) * margin
            y = row * size_block + (row + 1) * margin
            pygame.draw.rect(screen, color, (x, y, size_block,
size_block))
            if color == red:
                pygame.draw.line(screen, white, (x+5, y+5),
(x+size_block-5, y+size_block-5), 3)
                pygame.draw.line(screen, white, (x + size_block — 5,
y+5), (x+5, y+size_block-5), 3)
            elif color == green:
                pygame.draw.circle(screen, white, (x+size_block//2,
y+size_block//2), size_block//2-3, 3)
    if(query-1)%2==0:
        game_over = check_win(mas, "x")
    else:
        game_over = check_win(mas, "o")
    if game_over:
        screen.fill(black)
        font = pygame.font.SysFont("arial", 80)
        text1 = font.render(game_over, True, white)
        text_rect = text1.get_rect()
        text_x = screen.get_width()/2 — text_rect.width/2
        text_y = screen.get_height()/2 — text_rect.height/2
        screen.blit(text1, [text_x, text_y])
      pygame.display.update()
```



Nazorat savollari.

1. Pygame moduli metodlari qaysilar?
2. Pygame moduli yordamida o‘yin yarating?

26-ma’ruza. Pythonda OS moduli.

Reja:

1. Pythonda OS moduli bilan ishlovchi konstantalar.
2. OS moduli bilan ishlashga misollar.

Python-dagi OS moduli operatsion tizim bilan o‘zaro ishlash funktsiyalarini ta’minlaydi. OT Python standart yordamchi modullari ostida keladi. Ushbu modul operatsion tizimga bog‘liq funksiyalardan foydalanishning ko‘chma usulini taqdim etadi. *os* va *os.path* modullari fayl tizimi bilan o‘zaro ishlash uchun ko‘plab funktsiyalarni o‘z ichiga oladi.

Os moduli-har xil operatsion sistemalarning o‘ziga xos xususiyatlari bilan ishlovchi kategoriyadagi asosiy modul hisoblanadi. Bu modul funksiyalari ko‘plab operatsion sistemalarda ishlaydilar. Kataloglarni bo‘luvchi os moduli va u bilan bog’liq bo‘lgan ifodalar konstanta ko‘rinishida berilgan.

Konstanta	Vazifasi
Os.curdir	Joriy katalog
Os.pardir	Bosh katalog
Os.sep	Yo‘lning elementlarini taqsimlovchi
Os.altsep	Boshqayo‘lningelementlarini taqsimlovchi
Os.pathsep	Yo‘llarro‘yxatidagiyo‘llarni taqsimlovchi
Os.defpath	Yashirin yo‘llar ro‘yxati
Os.linesep	Satrni yakunlovchi belgi

Pythondagi dastur operatsion tizimda alohida jarayon ko‘rinishida ishlaydi. Os modulining funksiyalari protsesda, muhitda bajariladigan turli xildagi ahamiyatga ega bo‘lgan irishlarga ruxsatetadilar. Osmodulining eng muhimruxsat etuvchi obyektlaridan biri deb environ o‘rab oluvchi muhiti o‘zgaruvchilarning lug‘ati hisoblanadi. Masalan o‘rab oluvchi muhit o‘zgaruvchilar yordamida web server CGI-ssenariyga bir qancha parametrlarni o‘tkazadi. Quyidagi misolda PATH

o‘rab oluvchi muhiti o‘zgaruvchini olish mumkin:

import os PATH=os.environ[’PATH’]

Funksiyalarning katta qismi fayllar va kataloglar bilan ishlashga mo‘ljallangan. Quyida UNIX va Windows OT lar uchun ruxsat etilgan funksiyalar taqdim etilgan:

Access(path, flags)- pathnomli fayl yoki catalog ruxsat etish(доступь) ni tekshiradi. Buyurma qilishga rucsatning tartibi flags raqami bilan belgilanadi. U esa yaratilgan kombinatsiyalar os.F_OK (fayl mavjud), os.R_OK (fayldan o‘qish mumkin), os.W_OK (faylga yozish mumkin) va os.X_OK (fayllarni bajarishni, katalogni ko‘rib chiqish mumkin) bayroqlari bilan belgilash mumkin.

Chdir(path)- path ni joriy ishchi katalog qiladi. Getcwd()- joriy ishchi catalog.

Chmod(path, mode)- mode ga path bo‘lgan ruxsat etish rejimini belgilaydi. Ruxsat etish tartibi bayroqlarni kombinatsiya qilib belgilashi mumkin. Bu ishda chmod() harakatda bo‘lgan tartibni to‘ldirmaydi, uni yangidan belgilamaydi, uni yangidan belgilaydi.

Listdir(dir)- dir katalogidagi fayllar ro‘yxatini qaytaradi. Ro‘yxatga maxsus belgilar “.” va “..” kirmaydi.

Mkdir(path [, mode])- path katalogini tuzadi. Jimlik holatida mode tartibi 0777 ga teng bo‘ladi, bu degani S_IRWXU|S_IRWXG|S_IRWXO agarda stat moduli konstantalari bilan foydalansak.

```
import os
os.mkdir(’C:\Users\Guljakhon\Desktop\Новая папка\katalog\dir2’)
```

```
#ko‘rsatilgan manzilda dir2 nomli yangi katalog yaratadi.
import os os.mkdir(’./dir2’)
#joriy manzilda dir2 nomli yangi catalog yaratadi.
```

Makedirs(path [,mode])- hamma kataloglarni yaratuvchi, agarda ular mavjud bo‘lmasalar mkdir() analogi oxirgi katalog mavjud bo‘lgandan so‘ng mustasnoni ishga tushiradi.

Remove(path), unlink(path)- path katalogini yo‘qotadi. Kataloglarni yo‘qotish uchun rmdir() va removedirs() dan foydalanadi.

Rmdir(path)- path nomli bo‘sh katalogni yo‘qotadi.

Removedirs(path)- birinchi bo‘sh bo‘lgan kataloggacha pathni yo‘q qiladi. Agarda yo‘lda eng oxirgi kiritilgan katalog osti bo‘sh bo‘lmasa OSError mustasnosini ishga tushiradi.

Rename(src, dst)- src fayli yoki katallogini dst deb qayta nomlaydi. Renames(src, dst)- rename() analogi dst yo‘li uchun kerakli kataloglarni yaratadi va src yo‘lining bo‘sh kataloglarini yo‘qotadi.

Stat(path)- path haqidagi malumotni o‘nta elementlik kortej shaklida qaytaradi. Kortej elementlariga kirish uchun stat moduli konstantalaridan foydalanish mumkin. Masalan stat.ST_MTIME (faylning oxirgi modifikatsiyasi vaqti).

Utime(path, times)- oxirgi modifikatsiya (mtime) va faylga kirishga ruxsat(atime) larini belgilaydi. Boshqa holatlarda times ikki elementli kortej (atime, mtime) sifatida ko‘rib chiqiladi. Qaysidir faylni atime va mtime ni olish uchun stat() va stat modulining konstantalarini barobar ishga tushirib olish mumkin.

Os moduli protsesslar bilan ishlash uchun quyidagi funksiyalarni taqdim etadi (ular ham UNIX hamda windowsda ishlaydilar).

System(cmd)- alohida oynada cmd buyruqlar satrini bajaradi. U C tilining system kutubxonasi chqirig’iga analogik bo‘ladi. Qaytarilgan qiymat foydalanadigan platformadan tobe bo‘ladi.

Times()- beshta elementdan iborat bo‘lgan kortejni qaytaradi. U ish jarayoni vaqtini lahzalarda ko‘rsatadi, qo‘shimcha protsesslar vaqtini, qo‘shimcha protsesslarning axborot tizimlari vaqtini, va o‘tgan zamonda qotib qolgan vaqtni ko‘rsatadi (masalan tizim ishga tushgan paytdan).

Getloadavg()- coo, uchta qiymatlik kortejni qaytaradi.

Joriy ishchi katalog bilan ishlash. Joriy ishchi katalogni (CWD) Python ishlaydigan papka sifatida ko‘rib chiqing . Fayllar faqat nomi bilan chaqirilganda, Python u CWD da boshlanadi deb hisoblaydi, ya’ni faqat nomga havola faqat fayl Python CWD da bo‘lsa muvaffaqiyatli bo‘ladi.

Eslatma: Python skripti ishlayotgan papka joriy katalog sifatida tanilgan. Bu Python skripti joylashgan yo‘l emas.

Joriy ishchi katalogni olish. Joriy ishchi katalog manzilini olish uchun os.getcwd() dan foydalaniladi.

Misol:

import os

```
cwd = os.getcwd()
print("Current working directory:", cwd)
```

Natija: Current working directory:
C:\Users\user\Downloads\TicTacToe

Joriy ishchi katalogni o‘zgartirish. Joriy ishchi katalogni (CWD) o‘zgartirish uchun os.chdir() usuli qo‘llaniladi. Ushbu usul CWD ni belgilangan yo‘lga o‘zgartiradi. Yangi katalog yo‘li sifatida faqat bitta argumentni oladi.

Eslatma: Joriy ishchi katalog Python skripti ishlaydigan papkadir.
Misol:

```
import os
def current_path():
print("Current working directory before")
print(os.getcwd())
print()
current_path()
os.chdir('../')
current_path()
```

Natija: Current working directory before
C:\Users\user\Downloads\TicTacToe

Current working directory before
C:\Users\user\Downloads

Katalog yaratish. OS modulida katalog yaratishning turli usullari mavjud. Bular:
os.mkdir()
os.makedirs()
os.mkdir() dan foydalanish. Pythonda os.mkdir() usuli belgilangan raqamli rejim bilan yo‘l nomli katalog yaratish uchun ishlatiladi. Agar yaratilishi kerak bo‘lgan katalog allaqachon mavjud bo‘lsa, bu usul FileExistsError ni keltirib chiqaradi.
Misol:

```
import os
directory = "1-papka"
parent_dir = "D:/Pycharm projects/"
path = os.path.join(parent_dir, directory)
os.mkdir(path)
```

```
print("Directory ’% s’ created" % directory)
directory = "2-papka"
parent_dir = "D:/Pycharm projects"
mode = 0o666
path = os.path.join(parent_dir, directory)
os.mkdir(path, mode)
print("Directory ’% s’ created" % directory)
```
Natija:
Directory ’1-papka’ created
Directory ’2-papka’ created
os.makedirs() dan foydalanish. Python’da os.makedirs() usuli rekursiv katalog yaratish uchun ishlatiladi. Ya’ni, barg katalogini yaratishda, agar biron bir o‘rta daRejadagi katalog mavjud bo‘lmasa, os.makedirs() usuli ularning barchasini yaratadi.
```
import os
directory = "Nikhil"
parent_dir = "D:/Pycharm projects/1-papka/Authors"
path = os.path.join(parent_dir, directory)
os.makedirs(path)
print("Directory ’% s’ created" % directory)
directory = "c"
parent_dir = "D:/Pycharm projects/1-papka/a/b"
mode = 0o666
path = os.path.join(parent_dir, directory)
os.makedirs(path, mode)
print("Directory ’% s’ created" % directory
```
Natija:
Directory ’Nikhil’ created
Directory ’c’ created
Python yordamida fayllar va kataloglarni ro‘yxatga olish. Pythonda os.listdir() usuli belgilangan katalogdagi barcha fayllar va kataloglar ro‘yxatini olish uchun ishlatiladi. Agar biz biron bir katalogni ko‘rsatmasak, u holda joriy ishchi katalogdagi fayllar va kataloglar ro‘yxati qaytariladi.
Misol:
```
import os
path = "/"
dir_list = os.listdir(path)
```

```
print("Files and directories in '", path, "' :")
print(dir_list)
```

Natija: Files and directories in ' / ' :
['$RECYCLE.BIN', '$WINDOWS.~BT', '$WinREAgent', '537694377 89084459.pfx', 'AppData', 'Documents and Settings', 'DS4231296531009900 01.pfx', 'DS4231296531009900 01_1.pfx', 'DS4231296531009900 02.pfx', 'DSKEYS', 'DSKEYS99', 'Inprise', 'Intel', 'MSOCache', 'pagefile.sys', 'Program Files', 'Program Files (x86)', 'ProgramData', 'Recovery', 'Skins', 'swapfile.sys', 'System Volume Information', 'Users', 'Windows']

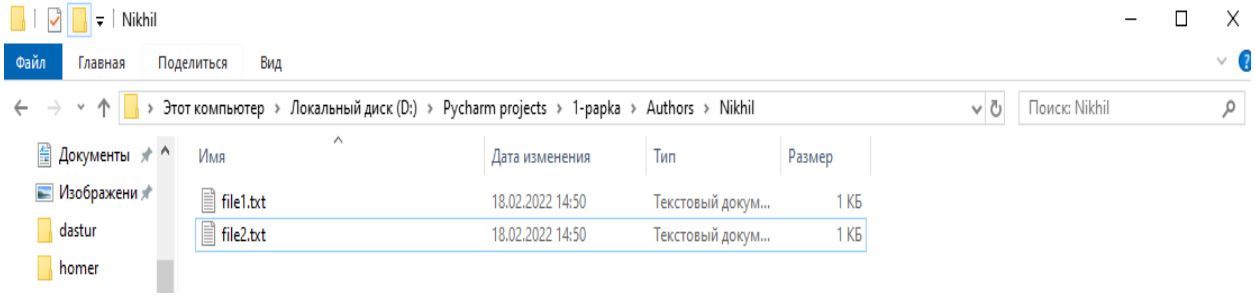
Python yordamida katalog yoki fayllarni o‘chirish. OS moduli Python-da katalog va fayllarni o‘chirishning turli usullarini isbotlaydi. Bular:

os.remove() dan foydalanish

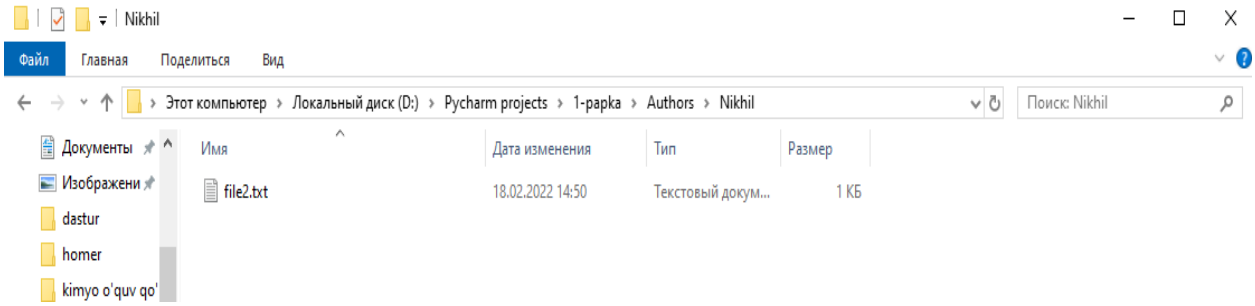
os.rmdir() dan foydalanish

os.remove() dan foydalanish. Python’da os.remove() usuli fayl yo‘lini o‘chirish yoki o‘chirish uchun ishlatiladi. Ushbu usul katalogni olib tashlab yoki o‘chira olmaydi. Belgilangan yo‘l katalog bo‘lsa, OSError usul bilan ko‘tariladi.

Misol: Jilddagi faylni ko‘raylik:




```
import os
file = 'file1.txt'
location = "D:/Pycharm projects/1-papka/Authors/Nikhil/"
path = os.path.join(location, file)
os.remove(path)
```



Tez-tez ishlatiladigan funksiyalar.

os.name**:** Bu funksiya import qilinadigan operatsion tizimga bog‘liq modul nomini beradi. Hozirda quyidagi nomlar ro‘yxatga olingan: ’posix’, ’nt’, ’os2’, ’ce’, ’java’ va ’riscos’.

```
import os
   print(os.name)
>>> nt
```

Eslatma: Bu yerda kodni ishga tushirganingizda "nt" kabi turli tarjimonlarda turli xil natijalar berishi mumkin.

Nazorat savollari.

1. Papkaning kompyuterda mavjud va mavjudmasligini tekshirish?
2. OS moduli konstantalari qaysilar?

27-ma’ruza.Playsound moduli.

Reja:

1. Playsound modulini o‘rnatish.
2. Playsound modulidan foydalanish.

Playsound moduli audio kutubxonalardan foydalangan holda Python-da ovozni qanday o‘ynashni ko‘rib chiqamiz. Ovozni ijro etishning turli usullarini bilib olamiz.

1-usul: Playsound modulidan foydalanish

Paketlarni o‘rnatish uchun quyidagi buyruqni bajaring:

```
pip install playsound
```

Playsound modulida faqat playsound () nomli bitta funksiya mavjud .

Bu bitta dalilni talab qiladi: biz o‘ynashimiz kerak bo‘lgan ovozli faylga yo‘l. Bu mahalliy fayl yoki URL bo‘lishi mumkin .

Ixtiyoriy ikkinchi argument mavjud, blok , sukut bo‘yicha True ga o‘rnatiladi. Funktsiyani asenkron ishlashi uchun biz uni False ga o‘rnatishimiz mumkin .

U WAV va MP3 fayllari bilan ishlaydi.

Misol: WAV formati uchun

```
from playsound import playsound
playsound('/path/note.wav')
print('playing sound using playsound')
```

Misol: MP3 formati uchun

```
# import required module
from playsound import playsound
# for playing note.mp3 file
playsound('/path/note.mp3')
print('playing sound using playsound')
```

Nazorat savollari:

1. Playsound moduli qanday o‘rnatiladi?
2. Playsound modulidan foydalanib .mp3 ni o‘qiting?

28-dars.Googletranslate moduli. Google Translate API o‘rnatilishi.

Reja:

1. Google Translate APIni o‘rnatish.
2. Translate dastur yaratish.

Pythonda Google Translate API bilan ishlashdan oldin uni o‘rnatishingiz kerak bo‘ladi. APIni o‘rnatishning ikki xil usuli mavjud. Birinchi usul to‘g‘ridan-to‘g‘ri oldinga. Shunchaki terminalga o‘ting va pipboshqa Python kutubxonalarida bo‘lgani kabi APIni o‘rnatish uchun o‘rnatuvchidan foydalaning. Buning uchun terminalda quyidagi buyruqni kiriting:

$ pip install googletrans

Bosing Enterva Google Translate API uchun Python moduli tizimingizga o‘rnatiladi.

Agar siz Pythonning Anaconda distributivini o‘rnatgan bo‘lsangiz, API-ni Anaconda Prompt-dan foydalanib o‘rnatishingiz mumkin. Ushbu maxsus usulda siz quyidagi kod parchasida ko‘rsatilganidek, pipyuqoridagi buyruqni bilan almashtirasiz:conda

$ conda install googletrans

Qo‘llab-quvvatlanadigan tillar ro‘yxati

Google Translate API turli tillarni qo‘llab-quvvatlaydi. Barcha qo‘llab-quvvatlanadigan tillarni ro‘yxatga olish uchun quyidagi skriptni ishga tushiring:

```
import googletrans
print(googletrans.LANGUAGES)
```

Yuqoridagi misolda siz modulni importimport qilish uchun kalit so‘zdan foydalanasiz. Keyinchalik, modulning atributini googletranschop etish orqali barcha til nomlarini ro‘yxatlashingiz mumkin .LANGUAGESgoogletrans

Amalga oshirilganda, yuqoridagi kod qismi barcha qo‘llab-quvvatlanadigan tillar nomlari va ularning qisqartmasi bilan birga ro‘yxatlanadi. Chiqish quyidagicha ko‘rinadi:

{’af’: ’afrikaans’, ’sq’: ’albanian’, ’am’: ’amharic’, ’ar’: ’arabic’, ’hy’: ’armenian’, ’az’: ’azerbaijani’, ’eu’: ’basque’, ’be’: ’belarusian’, ’bn’: ’bengali’, ’bs’: ’bosnian’, ’bg‘: ’bulgarian’, ’ca’: ’catalan’, ’ceb’: ’cebuano‘, ’ny’: ’chichewa’, ’zh-cn’: ’chinese (simplified)’, ’zh-tw’: ’chinese (traditional)’, ’co‘: ’corsican’, ’hr’: ’croatian’, ’cs’: ’czech’,

’da’: ’danish’, ’nl’: ’dutch’, ’en’: ’english’, ’eo‘: ’esperanto‘, ’et’: ’estonian’, ’tl’: ’filipino‘, ’fi’: ’finnish’, ’fr’: ’french’, ’fy’: ’frisian’, ’gl’: ’galician’, ’ka’: ’georgian’, ’de’: ’german’, ’el’: ’greek’, ’gu’: ’gujarati’, ’ht’: ’haitian creole’, ’ha’: ’hausa’, ’haw’: ’hawaiian’, ’iw’: ’hebrew’, ’he’: ’hebrew’, ’hi’: ’hindi’, ’hmn’: ’hmong‘, ’hu’: ’hungarian’, ’is’: ’icelandic’, ’ig‘: ’igbo‘, ’id’: ’indonesian’, ’ga’: ’irish’, ’it’: ’italian’, ’ja’: ’japanese’, ’jw’: ’javanese’, ’kn’: ’kannada’, ’kk’: ’kazakh’, ’km’: ’khmer’, ’ko‘: ’korean’, ’ku’: ’kurdish (kurmanji)’, ’ky’: ’kyrgyz’, ’lo‘: ’lao‘, ’la’: ’latin’, ’lv’: ’latvian’, ’lt’: ’lithuanian’, ’lb’: ’luxembourgish’, ’mk’: ’macedonian’, ’mg‘: ’malagasy’, ’ms’: ’malay’, ’ml’: ’malayalam’, ’mt’: ’maltese’, ’mi’: ’maori’, ’mr’: ’marathi’, ’mn’: ’mongolian’, ’my’: ’myanmar (burmese)’, ’ne’: ’nepali’, ’no‘: ’norwegian’, ’or’: ’odia’, ’ps’: ’pashto‘, ’fa’: ’persian’, ’pl’: ’polish’, ’pt’: ’portuguese’, ’pa’: ’punjabi’, ’ro‘: ’romanian’, ’ru’: ’russian’, ’sm’: ’samoan’, ’gd’: ’scots gaelic’, ’sr’: ’serbian’, ’st’: ’sesotho‘, ’sn’: ’shona’, ’sd’: ’sindhi’, ’si’: ’sinhala’, ’sk’: ’slovak’, ’sl’: ’slovenian’, ’so‘: ’somali’, ’es’: ’spanish’, ’su’: ’sundanese’, ’sw’: ’swahili’, ’sv’: ’swedish’, ’tg‘: ’tajik’, ’ta’: ’tamil’, ’te’: ’telugu’, ’th’: ’thai’, ’tr’: ’turkish’, ’uk’: ’ukrainian’, ’ur’: ’urdu’, ’ug‘: ’uyghur’, ’uz’: ’uzbek’, ’vi’: ’vietnamese’, ’cy’: ’welsh’, ’xh’: ’xhosa’, ’yi’: ’yiddish’, ’yo‘: ’yoruba’, ’zu’: ’zulu’}

Asosiy foydalanish

Google Translate API-dan eng asosiy foydalanish, albatta, so‘z yoki jumlalarni bir tildan boshqa tilga tarjima qilishdir. Buning uchun moduldan Translatorsinfni import qilishimiz kerak .googletrans

```
from googletrans import Translator
```

Translator keyinchalik, siz sinfning ob’ektini yaratishingiz kerak .

```
translator = Translator()
```

Translator sinfi ob’ekti yaratilgandan so‘ng, siz quyida ko‘rsatilganidek, boshlang‘ich tildagi matnni sinf ob’ektining translate() usuliga parametr Translator() sifatida o‘tkazasiz:

Usul tomonidan qaytarilgan ob’ekt translate() quyidagi atributlarga ega:

src: Manba tili

dest: Ingliz tiliga o‘rnatilgan maqsad tili (uz)

origin: Asl matn, ya’ni bizning misolimizda "Mitä sinä teet"

text: Tarjima qilingan matn

pronunciation: Tarjima qilingan matnning talaffuzi
Keling, yuqoridagi barcha atributlarni chop qilaylik va qanday natija olishimizni ko‘rib chiqaylik:

```
####
print(result.src)
print(result.dest)
print(result.origin)
print(result.text)
print(result.pronunciation)
```

Natija:

```
fi
en
Mitä sinä teet
What are you doing
What are you doing
```

Natija shuni ko‘rsatadiki, manba tili fin (fi) va maqsad tili ingliz (en). Text tarjima qilingan jumla atribut orqali chop etilishi mumkin.

Yuqoridagi misolda biz manba tilini ko‘rsatmadik. Shuning uchun Google Translate API manba tilining o‘zini aniqlashga harakat qiladi. Xuddi shunday, biz hech qanday maqsad tilini belgilamadik va shuning uchun API manba tilini standart tilga, ya’ni ingliz tiliga tarjima qildi. Ammo, agar siz manba va maqsad tillarini ko‘rsatmoqchi bo‘lsangiz-chi?

Manba va maqsad tillarini belgilash

Aslida, Google Translate API-da maqsad va manba tillarini belgilash juda oson. Bu erda siz faqat manba tilini o‘tkazish uchun foydalanadigan kod:

```
result = translator.translate('Mikä on nimesi', src='fi')
```

Faqat maqsadli tilni qo‘shish uchun destatributni, so‘ngra til kodini qo‘shishingiz kerak:

```
result = translator.translate('Mikä on nimesi', dest='fr')
```

Siz bir vaqtning o‘zida manba va maqsad tillarini ham o‘tkazishingiz mumkin:

```
result = translator.translate('Mikä on nimesi', src='fi', dest='fr')
```

Endi, fin tilidagi jumlani frantsuz tiliga tarjima qilamiz va keyin manba va maqsad tillarini, shuningdek tarjima qilingan matnni chop etamiz. Bu safar biz manba va maqsad tillarini aniqlaymiz.

```
from googletrans import Translator
translator = Translator()
result = translator.translate('Mikä on nimesi', src='fi', dest='fr')
print(result.src)
print(result.dest)
print(result.text)
```

Yuqoridagi kod qismi quyidagi natijani beradi.

```
fi
fr
Quel est votre nom
```

Iboralar ro'yxatini tarjima qilish

Google Translate API yordamida matnli iboralar ro'yxatini ham tarjima qilish mumkin. Asosiy jarayon yuqorida muhokama qilinganidek. Usulga parametr sifatida iboralarni o'z ichiga olgan ro'yxatni o'tkazish kifoya translate() . Bu alohida-alohida tarjima qilingan iboralar to'plamiga ega bo'lish uchun foydalidir, lekin barchasi bitta API chaqiruvida.

Keling, frantsuz tilidan ba'zi iboralarni o'z ichiga olgan qatorlar ro'yxatini tuzamiz.

```
sentences = ['Bienvenu', 'Comment allez-vous', 'je vais bien']
```

Endi usulni chaqirish translate() va ro'yxatni, manba tilini va maqsad tilini parametr sifatida o'tkazish vaqti keldi.

```
result = translator.translate(sentences, src='fr', dest='sw')
```

Yuqoridagi skriptda manba tili frantsuz, maqsad tili esa suahili.

Agar translate() siz unga iboralar ro'yxatini uzatsangiz, usul ob'ektlar ro'yxatini qaytaradi. Usul bo'yicha qaytarilgan ro'yxatdagi har bir ob'ekt translate() tarjima qilinishi kerak bo'lgan kirish ro'yxatidagi har bir iboraga mos keladi. Ro'yxatdagi har bir kirish iborasining tarjimasini topishning eng yaxshi usuli chiqish ob'ektlari ro'yxatini takrorlashdir. Keyin kirish ro'yxatidagi alohida iboralarning tarjimasini ko'rish uchun alohida ob'ektlarning text, origin, , va boshqa atributlaridan foydalanishingiz mumkin.src

Quyidagi skriptda biz translate() usul bilan qaytarilgan ob'ektlar ro'yxatini takrorlaymiz va keyin asl va tarjima qilingan matnni chop etamiz:

```
for trans in result:
   print(f'{trans.origin} -> {trans.text}')
```

Quyidagi natija ekranda ko'rsatiladi.

Bienvenu -> karibu
Comment allez-vous -> Vipi wewe
je vais bien -> Niko sawa
Matnli hujjatlarni tarjima qilish

Google Translate API orqali matnli hujjatlarni ham tarjima qilishingiz mumkin. Siz qilishingiz kerak bo‘lgan narsa Python-dagi matn faylini openusul yordamida o‘qish, matnni o‘qish va uni translate() usulga o‘tkazishdir.

Birinchi qadam faylni "o‘qish" rejimida ochishdir:

```
f = open('C:\\Users\\Khurram\\Desktop\\test.txt', 'r')
```

Bundan tashqari, xususiyat yordamida faylning "o‘qish" rejimida yoki yo‘qligini tekshirishingiz mumkin (mode):

```
if f.mode == 'r':
```

Keyinchalik, f.read() fayl mazmunini o‘qish uchun usuldan foydalanishingiz mumkin. Fayl mazmuni har qanday o‘zgaruvchida saqlanishi mumkin. Bizning holatda, o‘zgaruvchining nomi bo‘ladi (contents).

Contents Python matn faylini to‘g‘ri o‘qiyaptimi yoki yo‘qligini tekshirish uchun biz o‘zgaruvchini ham chop etamiz :

```
contents = f.read()
print(contents)
```

Fayl mazmunining chiqishi:

```
We are going to translate this text file using Python.
Subsequently, we will also translate it into French.
```

Eslatma.Bu misolimizga amal qilmoqchi bo‘lsangiz, matn faylingizda yuqoridagi tarkib mavjudligiga ishonch hosil qiling.

Biz Python matn fayliga kirish va o‘qishni aniqladik. TranslateEndi biz oldingi sinfni import qilish orqali natijani tarjima qilamiz .

```
from googletrans import Translator
file_translate = Translator()
```

Keyingi qadam, contents kiritilgan matnni o‘z ichiga olgan o‘zgaruvchini translate() funktsiyaga o‘tkazishdir. Nihoyat, usul textbilan qaytarilgan ob’ektning atributini chop translate() eting va siz tarjima qilingan qatorni olasiz.

```
result = translator.translate(contents, dest='fr')
print(result.text)
```

Chiqish quyidagicha ko‘rinishi kerak:

Nous allons traduire ce fichier texte en Python.

Par la suite, nous le traduirons également en français.
Tarjima qilingan matnni bir xil faylga yoki boshqa matnli faylga yozish uchun faylni yozish rejimida (“w”) ochish kifoya. Keyinchalik, siz write()usulni chaqirishingiz va quyida ko‘rsatilganidek, tarjima qilingan matnni topshirishingiz kerak:

```
with open('C:\\Users\\Khurram\\Desktop\\test_2.txt', 'w') as f:
    f.write(result.text)
```

Yuqoridagi misolda biz write oqimini avtomatik ravishda ochish va yopish uchun kontekst menejeridan foydalanganmiz. withIkkinchidan, biz faylni yozish rejimida ochdik. Va nihoyat, biz write() tarjima qilingan satrni yangi faylga yozish usulidan foydalandik.
API bilan tanishish. Sodda ko‘rinishda ishlatish:

```
from googletrans import Translator
translator = Translator()
print(translator.translate('안녕하세요.'))
```

Ekranga quyidagilar chiqadi:
<Translated src=ko dest=en text=Good evening. pronunciation=Good evening.>
src=ko —
avtomatik tarzda kiritilgan so‘zni koreys tilida deb topdi
dest=en — avtomatik tarzda inglizchaga tarjima qildi
text=Good evening — tarjima qilingan matn
pronunciation=Good evening — ingliz tilidagi talafuzi
Kodni qulaylashtiramiz:

```
from googletrans import Translator
translator = Translator()
s = input("Satrni kiriting: ")
des = input("Qaysi tilga tarjima qilsin: ")
res = translator.translate(s, dest=des)
print(res.text)
```

Dasturni ishga tushurganimizda. Sizdan satrni va tilni so‘raydi. Siz kiritgan satrni belgilangan tilga tarjima qilib beradi.
Satrni kiriting: Men kuchli dasturchi bo‘laman
Qaysi tilga tarjima qilsin: ko
나는 강한 개발자 수 있습니다

Nazorat savollari

1. Google Translate APIni o‘rnatish ketma-ketligi qanday?
2. Ingliz-o‘zbek translate dasturini yarating?

29-30-ma’ruza. Pythonda telegram bot yaratish

Reja:

1. Telegram bot yaratish modullarini o‘rnatish.
2. Telegram bot yaratish.

Hech bo‘lmaganda bir marta Telegram messenjeriga duch kelgan har bir kishi uning faoliyati bilan hayratda qoldi. Bu erda siz nafaqat shaxsiy xabarlarda do‘stlaringiz bilan suhbat qilishingiz, yangiliklar o‘qishingiz va kanallar qilishingiz mumkin. Xizmatning afzalligi – bu botlarni yaratish va ishlatish qobiliyatidir. Siz ularni istalgan tilda dasturlashingiz mumkin, ammo Python bugungi kunda eng keng tarqalgan holga aylandi. Maqolada python telegram botini qanday qilish haqida batafsil ma’lumot beriladi.

Ilovalarni dasturlash interfeysi – ishlab chiquvchi dasturlarni shakllantirishi mumkin bo‘lgan interfeys.

Unga rahmat, dasturning turli qismlarini bir-biri bilan uyg‘un va to‘g‘ri o‘zaro bog‘lash uchun sozlash paydo bo‘ldi.

Dastlab, api turli xil dasturlar o‘rtasida ma’lumot va buyruqlarni uzatish uchun ishlatilgan. Bugungi kunda bu boshqa serverdagi manbalarga kirish imkoniyatini beradi.

Uning qo‘llanilishi quyidagi afzalliklarga ega:

Hamkorlik dasturining mavjudligi.

Oldindan formatlangan havolalarni ID bilan bir vaqtda yuklash bilan ishlash.

Istalgan vaqtda eng aniq va dolzarb ma’lumotlarni taqdim etish qobiliyati.

Javob ma’lumotlarini JSON yoki XML formatlarida olish.

API bo‘lishi mumkin:

Umumiy Oson kirish.

Xususiy. U faqat bitta kompaniya ichida ishlatilishi mumkin. Agar u ko‘plab mahsulotlarni ishlab chiqqan bo‘lsa, unda interfeys turli xil dasturlarni bir-biri bilan o‘zaro ishlashga imkon beradi.

Dasturlash interfeysining asosiy vazifalari quyidagilardan iborat:

kodlarni yozishda yordam berish;

murakkab vazifalarni oddiylariga aylantirish.

Pythonda Telegram botini yaratish bo‘yicha ko‘rsatmalar

O‘z robotingizni olishning bir nechta variantlari mavjud:

O‘zingiz yozing. Buning uchun turli xil dasturlash tillaridan

foydalanishingiz mumkin. Hozirgi kunda eng tushunarli va ommaboplardan biri bu Python. Ushbu usul koʻp vaqt talab qiladigan boʻlsa-da, lekin ayni paytda siz universal echimni topishga imkon beradi.

Dizaynerning xizmatlaridan foydalaning. Ammo bu erda siz cheklangan funktsiyaga duch kelishingiz mumkin, bu har doim ham tarjima qilish uchun etarli emas.

Xarid qilish. Biroq, siz asosan telebot yozish uchun python ishlatilishini tushunishingiz kerak. Va bu juda oson deb hisoblanadi, hatto yangi boshlanuvchilar ham undan foydalanishlari mumkin.

Oʻz-oʻzini oʻrganish va muayyan qoidalarga muvofiq ishlashi mumkin boʻlgan 2 turdagi botlar mavjud:

Birinchi tur kamroq tarqalgan. Robot ma'lum qoidalar asosida oʻqitiladi, ularning asosida u berilgan savollarga javob beradi. Bot oddiy soʻrovlarni bajaradi va murakkab boʻlganlar qiyinchiliklarga olib kelishi mumkin.

Oʻz-oʻzini oʻrganish roboti samaraliroq. Bu sodir boʻladi:
qidirish – javob uchun kutubxona ma'lumotlar bazasida roʻyxatga olingan nusxalarni ishlatadi. Suhbatning mazmuniga qarab, u roʻyxatdan matnni tanlaydi;
generativ – soʻrovda oʻrganilgan soʻzlar asosida xabarlar yarata oladi.

Botni roʻyxatdan oʻtkazish

Jarayonning bu qismi oson. @BotFather-da roʻyxatdan oʻtish va unga xabarga "Ishga tushirish" buyrugʻini yuborish.

Ixtiyoriy ism. Bunga javoban u hujjatga (hujjatlar) va tokenga havolani yuboradi. Ularni darhol saqlash tavsiya etiladi, chunki ular yordamchi bilan oʻzaro aloqada avtorizatsiya qilishning yagona kaliti boʻladi.

Kodni yozish

Python telegram bot api moduli robotning yaratilishi va ishlashi uchun javobgardir. Buning uchun pip oʻrnatish pyTelegramBotAPI buyrugʻini yuboring.

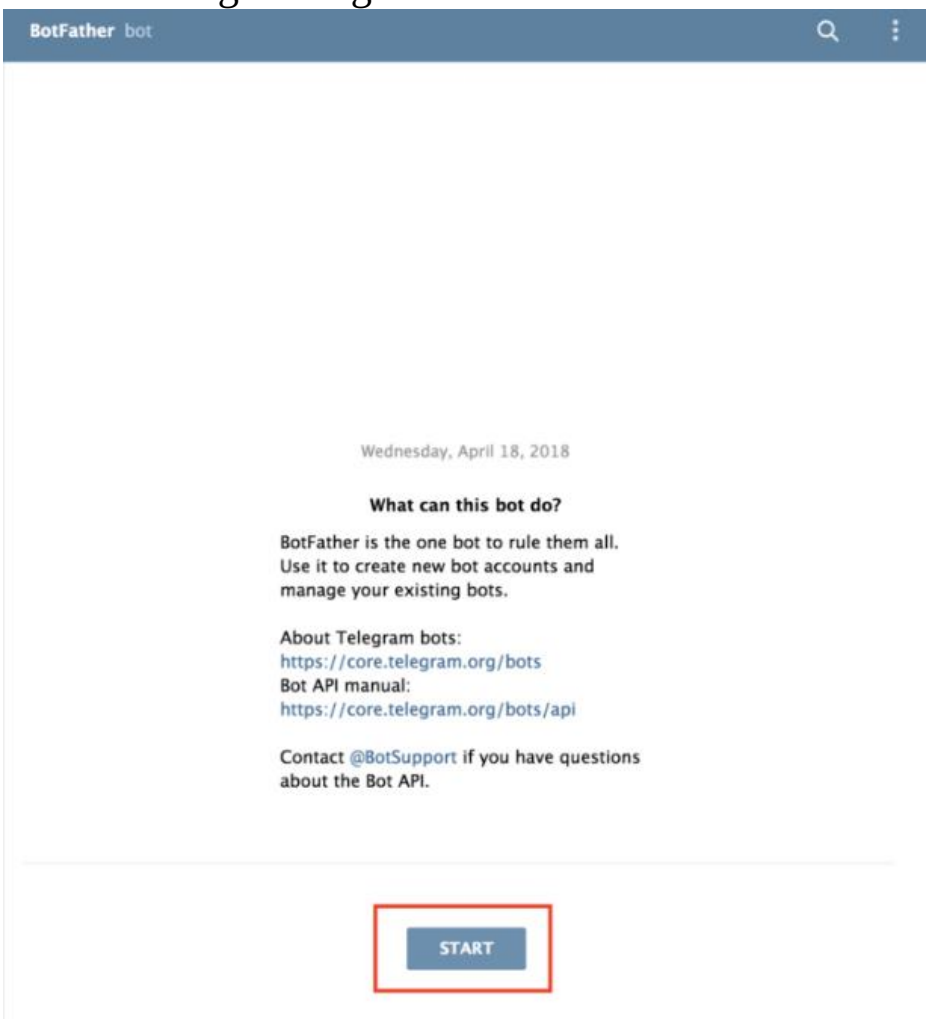
Kodni istalgan joyga yozing. Bu Word fayli yoki bloknot boʻlishi mumkin. Ammo buni aqlli muhitda bajarish ancha samaralidir. Mumkin boʻlgan xatolar bu erda avtomatik ravishda ta'kidlanadi.

Kodni yaratish telebotni ulash bilan boshlanadi. Bu erda sizga avval saqlangan token kerak boʻladi. Birinchi qator quyidagicha koʻrinadi: xabar: TOKEN = bot yuborgan kalit.

Ikkinchi satr biz bot deb nomlagan ob’ektni yaratadi. Autentifikatsiya kodi argumentlarda yozilgan.

Endi yordamchiga nima qilish kerakligi haqida o‘ylash kerak. PyTelegramBotAPI katalogida dekorativlar mavjud, ulardan foydalanishda robot standart savollarga javob berishni o‘rganadi.

Botni sozlang va ishga tushirish:



Yordamchi ishlashi uchun sizga quyidagilar kerak:
Xabarchiga kiring.
Robot qayd hisobini oching.
Yangi dialog oynasida yuqori qismida uchta nuqta bo‘lgan rasmni bosing.
Sozlamalarga o‘ting.
Robotni ishga tushirish uchun sizga kerak:
Qo‘llaringiz bilan klaviaturadagi qidirish qatoriga yordamchining ismini kiriting va u bilan suhbatni boshlang.

Uni “Yangi a’zo qo‘shish” tugmachasi yordamida kanalga ulang. Ro‘yxatdagi o‘zingiz xohlagan variantni tanlang va “taklif qilish” ni bosing.

*Buyruq ishlovchilari.*Buning uchun maxsus dastur Handler ishlatiladi. U media-fayllar va matnli materiallar bilan ishlaydi. Ammo bizning holatlarimizda yordamchi dastur buyruqlarni botga yuborish uchun kerak. Siz shuningdek ishlov beruvchini o‘zingiz yozishingiz mumkin. Bunday holda, birinchi qatorda “ishlov beruvchida” haydash kerak. Ushbu parametr har doim ishlaydi, “help” yoki “start” kiritishlardan tashqari.

*Tugma ishlovchilari.*Telegram api python ham klaviatura manbasiga asoslangan bo‘ladi. Klaviatura tugmachasini ishga tayyorlashda, kerakli parametr bu tugmachani bosgandan so‘ng foydalanuvchi oldinga yuborishi mumkin bo‘lgan matndir.

Kod yozishda turli xil usullar qo‘llaniladi:

Qo‘shish – istalgan sonli tugmalar. Shu bilan birga, ular bir qatorda turishadi. Agar dastlab o‘rnatilgan kenglik allaqachon erishilgan bo‘lsa, ular o‘tkaziladi.

Qator – tugmalar soni ham cheklanmagan, ammo ularning barchasi bir xil chiziqda joylashgan.

Qo‘shish – birinchi usulni eslatadi, ammo oxirgi qatorga piktogrammalar qo‘shiladi.

*O‘rnatilgan rejim.*Bu botlardan foydalanish uchun bitta variant. Buning yordamida ular yanada ko‘proq imkoniyatlarga ega bo‘lishdi. Bunday robotlar har qanday vazifani bajarishga qodir. Misollardan misol: saytdan suhbatga matn yuborish, gif yoki rasmni joylashtirish. Funktsiyani amalda ko‘rish uchun bot uchun uning nomi va kalit so‘zidan (@gif, @bold, @pic) foydalanib buyruq berish kerak. Bunday holda, robot bir nechta javoblarni taklif qiladi. Foydalanuvchi ulardan birini tanlaydi va suhbatga yuboradi. O‘rnatilgan rejim tarixga kira olmaydi, faqat foydalanuvchi unga yozganlariga javob beradi.

Serverda botlarni joylashtiring. Bu Telegram-da yordamchi yaratishda yakuniy qadam. Buning uchun siz qimmatbaho uskunalar sotib olishingiz shart emas. Siz mtproto protokoli bilan bulutli proksi-serverlardan foydalanishingiz mumkin, bu erda ular biron bir dasturni bepul joylashtirishni taklif qilishadi.

Avval GitHub-da ro‘yxatdan o‘tishingiz kerak. Ushbu hisob yordamida siz botni Heroku proksi-serveriga joylashtirishingiz

mumkin. Agar dastur ishlamasa, jurnallarni tekshirish tavsiya etiladi.

Nazorat savollari

1. Telegram bot yaratish uchun token qanday olinadi?
2. Ingliz-Rus telegram botini yataing?

Foydalanilgan adabiyotlar

1. Eric Matthes. Python Crash Course Paperback.England 2015.205p.
2. Krishna Rungta. Learn Python in 1 Day: Complete Python Guide with Examples. India 2016. -182 p.
3. Narasimha Karumanchi. Data Structure and Algorithmic Thinking with Python Paperback. India 2015. 170p.
4. Сысоева М.В., Сысоев И. В. Программирование для «нормальныХ» с нуля на языке Python Москва. 2018. -180с.
5. Федоров Д. Ю.Основы программирования на примере языка Python.Санкт-Петербург 2018. -167 с.
6. Eshtemirov S. Nazarov F. Algoritmlash va dasturlash asoslari. O‘quv qo‘llanma. Samarqand 2019. -208 b.
7. Nazarov F. C++ tilida dasturlash asoslari. Uslubiy qo‘llanma. Samarqand 2017. -208 b.
8. Вирт Н. Алгоритмы + структуры данных = программа.- М.:Мир,1985.-405с.
9. Информатика. Базовой курс. Учебник для Вузов., СанкПетербург,2001.под редакцией С.В.Симоновича.
10. Непейвода Н.Н, Скоплин И.Н. Основания программирования. –Москва Ижевск: Институт компьютерныХ исследований, 2003 г. 864 с.
11. Лойко В.Л. Структуры и алгоритмы обработки данных. Учебное пособие для вузов.- Краснодар: КубГАУ. 2000. — 261 с.
12. Алфред В. Хоп Крофт, Джефри Д. Ульман. Структуры данныи алгоритмов. Издательский дом «Вильяме» Москва — Санкт-Петербург - Киев, 2003-384 с.

Qaydlar uchun

Z.M. Adizova

DASTURLASH TEXNOLOGIYALARI

O‘QUV QO‘LLANMA

Muharrir: *E.Eshov*
Tex.muharrir: *D.Abduraxmonova*
Musahhih: *M.Shodiyeva*
Badiiy rahbar: *M.Sattorov*

Nashriyot litsenziyasi № 022853. 08.03.2022.
Original maketdan bosishga ruxsat etildi: 10.05.2023. Bichimi 60x84. Kegli 16 shponli. “Times New Roman” garnitura 1/16. Ofset bosma usulida. Ofset bosma qog‘ozi.
Bosma tabog‘i 12,75 Adadi 20. Buyurtma № 71

KAMOLOT

“BUXORO DETERMINANTI” MCHJ
bosmaxonasida chop etildi.
Buxoro shahar Namozgoh ko‘chasi 24 uy
Tel:. + 998 91 310 27 22