O’ZBEKISTON RESPUBLIKASI AXBOROT TEXNOLOGIYALARI VA KOMMUNIKATSIYALARINI RIVOJLANTIRISH VAZIRLIGI

MUHAMMAD AL-XOPAZMIY NOMIDAGI TOSHKENT AXBOROT TEXNOLOGIYALARI UNIVERSITETI
NUKUS FILIALI

“Himoyaga ruxsat etildi”
“Dasturiy injiniringi”
kafedrasi mudiri _____
prof. N.Uteuliev
«____» _____2019y

Python dasturlash tilida mantiqiy o’yin dasturini ishlab chiqish

mavzusida

BITIRUV MALAKAVIY ISHI

Bitiruvchi _____ B.P. Jarilkanov
Imzo

Rahbar _____ prof. N.U.Uteuliev
Imzo

Taqrizchi _____ xxx
Imzo

NUKUS – 2019

MUNDARIJA

KIRISH

O’zbekiston Respublikasining mustaqillik odimlarini dadil qo’ygan hozirgi davrda axborotlashgan jamiyat qurish masalasi mamlakatimiz kelajagi uchun naqadar katta ahamiyat kasb etayotgani bizga sir emas. Internet hayotimizning bir bo’lagiga aylandi va biz uning xizmatlaridan har kuni foydalanamib kelamiz. Respublikamizda o’qitish texnologiyalarini zamonaviylashtirishni jadallashtirish dolzarb ahamiyatga ega. Chunkiy hozirgi kunda milliy ta’lim tizimining salohiyati iqtisodiy rivojlanishini yanada yuqori pog’onasiga ko’tarilishga amaliy imkoniyat ta’minlovchi asosiy ijtimoiy manba sifatida gavdalanadi.

Respublikamiz ta’lim tizimidagi asosiy vazifalardan biri, jahon talablariga mos keluvchi axborot texnologiyalarini o’qitish jarayoniga qo’llashdan iboratdir. O’zbekistonda ta’lim tizimining axborotlashtirilishi xalqaro hamjamiyatda ham tan olindi.

Python - mustaqil dasturlarni ishlab chiqishda va dasturlarning keng ko’lamdagi dasturlarida skriptlarni yaratish uchun ishlatiladigan mashhur dasturlash tili. Bu kuchli, portativ, qulay va bepul tildir. Ko’p sohalarda ishlaydigan dasturchilar, Pythonning samaradorlikni va yuqori sifatli dasturiy ta’minotni ishlab chiqishga yo’naltirilganligi kichik va yirik loyihalarda ham strategik afzalliklarga ega bo’lishiga ishonadi.

Python foydalanuvchilari soni butun dunyo bo’ylab 1 million kishiga yaqin bo’lgan raqamdir (2011 yildan boshlab). Ushbu smeta turli statistik ko’rsatkichlarga asoslanadi, masalan, yuklashlar soni va ishlab chiquvchilar tadqiqotlari natijalari. Python ochiq manba kodli dastur bo’lib, uni ishlatish uchun hech qanday litsenziya talab qilinmaydi. Bundan tashqari, Python Linux distributivida sukut bo’yicha kiritilgan bo’lib, u Makintosh kompyuterlari va boshqa dasturiy va apparat mahsulotlari bilan birga keladi, bu esa foydalanuvchilar sonini aniqlashni qiyinlashtiradi. Umuman olganda, Python foydalanuvchilari soni juda katta va u atrofida juda faol ishlab chiquvchi jamoa tashkil etilgan.

Python 23 yildan ortiq vaqtdan beri paydo bo’lganligi va keng tarqalganligi sababli juda barqaror va ishonchli. Python nafaqat shaxsiy foydalanuvchilar

tomonidan qo’llanilgan, balki real daromad keltiradigan mahsulotlarni yaratish uchun kompaniyalar tomonidan ham qo’llaniladi.

TIOBE Index1 ga ko’ra, 2015 yilning sentyabr oyi holatiga ko’ra, Python PHP, JavaScript, Perl, Ruby, Delphi, Paskal, Swift va boshqalar.

Google va Intel, Cisco va Hewlett-Packard kabi taniqli kompaniyalar Python’dan foydalanishadi, uni moslashuvchanlik, foydalanish qulayligi va rivojlanishning yuqori tezligi uchun tanlashadi. Bu sizga boshqa tillarda yozilgan dasturlar va vositalar bilan osongina integratsiya qilingan samarali va ishonchli loyihalarni yaratishga imkon beradi.

Tilning universal tabiati turli sohalarda foydalanish imkoniyatini beradi. Aslida, ma’lum darajada ishonch bilan Python qisqa muddatli taktik vazifalarni hal qilish va uzoq muddatli strategik loyihalarni ishlab chiqish uchun deyarli har bir juda katta dasturiy ta’minotni ishlab chiqish tashkiloti tomonidan ishlatiladigan yoki boshqa yo’l bilan ishlatilishini ta’kidlash mumkin. Ma’lum bo’lishicha, ikkala holatda ham Python o’zini tasdiqladi.

Shunday qilib, Python hozirgi kunda ko’plab yirik kompaniyalar tomonidan har qanday muammolarni hal qilish uchun ishlatiladigan eng mashhur va samarali yuqori darajali dasturlash tillaridan biri hisoblanadi. Shuning uchun, bu ishning maqsadi Python dasturiy tilini umumiy nuqtai nazardan ko’rib chiqish hamda amaliyotda qo’llanilish usullarini batafsil o’rganishdir.

Ushbu bitiruv malakaviy ishi, kirish, 2 ta bob, har bir bobning qisqacha xulosasi, foydalanilgan adabiyotlar ro’yxati va xotimalardan iborat holda bayon qilingan. I bobda Python dasturlash tili, uning sintaksislari haqida ma’lumot keltirilgan, II bobda Pythonda ma’lumotlar tuzilmasi, tilning standart modullari keltirilgan.

I-BOB. PYTHON DASTURLASH TILI VA UNING SINTAKSISI

1.1. Python dasturlash tilining kelib chiqish tarixi

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

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

Bir qancha vaqt o'tib Gvido Gollandiyadan Amerikaga ko'chib o'tdi. Uni CNRI korparatsiyasiga ishlashga taklif etishdi. U o' sha yerda ishladi va korparatsiya shug'ullanayotgan proektlarni Python tilida yozdi va bo'sh ish vaqtlarida tilni interpretatorini rivojlantirib bordi. Bu 1990-yil Python 1.5.2 versiyasi paydo bo'lguncha davom etdi. Gvidoning asosiy vaqti korparatsiyani proektlarini yaratishga ketardi bu esa unga yoqmasdi. Chunki uning Python dasturlash tilini rivojlantirishga vaqti qolmayotgandi. Shunda u o'ziga tilni rivojlantirishga imkoniyat yaratib bera oladigan homiy izladi va uni o'sha paytlarda endi tashkil etilgan BeOpen firmasi qo'llab quvvatladi. U CNRI dan ketdi, lekin shartnomaga

binoan u Python 1.6 versiyasini chiqarib berishga majbur edi. BeOpen da esa u Python 2.0 versiyani chiqardi. 2.0 versiyasi bu oldinga qo'yilgan katta qadamlardan edi. Bu versiyada eng asosiysi til va interpretatorni rivojlanish jarayoni ochiq ravishda bo'ldi.

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

Pythonning kengayishi va C va C ++ tizimlariga qo'shilish qobiliyati boshqa tizimlar va komponentlarning xatti-harakatlarini tavsiflash uchun qulay va moslashuvchan tildir. Misol uchun, C til kutubxonasi bilan integratsiya Pythonga kutubxona qismlarini tekshirish va ishlatish imkonini beradi va Pythonni dasturiy ta'minot mahsulotlariga kiritish bu mahsulotlarni qayta taqsimlamasdan yoki manba kodi bilan ta'minlamasdan dasturiy mahsulotlarni sozlash imkonini beradi. Swing va SIP kabi dastur vositalari avtomatik ravishda dastur kodini ishlab chiqaradi, Python-da olingan komponentlarni skriptlarda ishlatish uchun qadamlarni avtomatlashtiradi va Cython dasturchilarga Python va C dastur kodlarini aralashtirish imkonini beradi. MS Windows-da Jython Java dasturi bo'lib, IronPython. NET asoslangan dasturdir va turli CORBA ilovalari dasturiy komponentlar bilan o'zaro munosabatlarni tashkil qilishning muqobil usullarini ta'minlaydi. Misol uchun, Windows operatsion tizimida Python skriptlari MS Word va Excel kabi boshqaruv platformalaridan foydalanishlari mumkin.

Sybase, Oracle, Informix, ODBC, MySQL, PostgreSQL, SQLite va boshqa ko'plab boshqa muhim ma'lumotlar bazalariga Python interfeyslari kiradi. Python dunyosida, turli ma'lumotlar bazalariga kirishni birlashtiruvchi Python skriptlaridan SQL ma'lumotlar bazalariga kirish uchun mo'ljallangan ko'chma dastur bazasi interfeysi ham mavjud. Misol uchun, ko'chma API foydalanilganda, bepul MySQL ma'lumotlar bazasi bilan ishlashga mo'ljallangan skript juda kam yoki o'zgarishsiz boshqa ma'lumotlar bazalari (Oracle kabi) bilan ishlay oladi. Buning uchun zarur bo'lgan past darajadagi interfeysni almashtirish kerak. Standart tuzlangan modul dasturlarga Python moslamalarni fayllar yoki maxsus moslamalarni saqlash va

tiklash imkonini beruvchi oddiy ob’ektni saqlash tizimini amalga oshiradi. Internetda ZODB deb nomlangan uchinchi taraf ishlab chiquvchilar tomonidan yaratilgan tizimni topishingiz mumkin. Python skriptlarida foydalanish uchun to’liq ob’ektga asoslangan ma’lumotlar bazasi. Bundan tashqari, SQLObject va SQLAlchemy kabi asboblar mavjud, bu relatsion jadvallarni Python sinf modeliga bog’laydi. Python 2.5 dan boshlab, SQLite ma’lumotlar bazasi Pythonning standart qismiga aylandi.

1.2. Xususiyatlari va falsafasi

Python ko’p paradigm dasturlash tilidir. Ob’ektga yo’naltirilgan dasturlash va tuzilgan dasturiy ta’minotlar to’liq qo’llab-quvvatlanadi va uning ko’pgina funktsional dasturlari va aspektga asoslangan dasturlashni qo’llab-quvvatlaydi (metaprogramming va metaobektsiyalar (sehrli usullar)). Ko’plab boshqa paradigmalar kengaytmalar orqali, jumladan, kontrakt bo’yicha loyihalashtiriladi va mantiqiy dasturlash.

Python dinamik yozishni va xotirani boshqarish uchun mos yozuvlar hisobini va tsiklni aniqlaydigan axlat kollektorining kombinatsiyasidan foydalanadi. Bundan tashqari, dasturni bajarish paytida uslub va o’zgarmaydigan nomlarni bog’laydigan dinamik nomning o’lchamlari (kechki majburiy) mavjud.

Pythonning dizayni Lisp an’anasida funktsional dasturlash uchun ba’zi bir yordam beradi. Filtr, xarita va funktsiyalarni kamaytiradi; lug’atlar, guruhlar va generatorlar ifodalari. [49] Standart kutubxonada Haskell va Standard ML dan olingan funktsional vositalarni tatbiq qiluvchi ikkita modul (itertools va functools) mavjud.

Tilning asosiy falsafasi Python The Zen (PEP 20) hujjatida ifodalanadi, unda quyidagi aforizmlar mavjud:

- Chiroyli chirkindan yaxshi
- Ko’rinib turibdiki, ochiq-oydinroqdir
- Oddiy murakkabroqdan yaxshiroqdir

- Kompleks murakkabdan yaxshiroqdir
- O’qish mumkinligini hisoblaydi

Uning barcha funktsiyalarini yadroga joylashtirish o’rniga, Python juda kengaytirilgan bo’lishi uchun mo’ljallangan. Ushbu yilni modulyarlik, mavjud ilovalarga programlanadigan interfeyslarni qo’shish vositasi sifatida ayniqsa mashhur bo’ldi. Van Rossumning katta standart kutubxonaga ega bo’lgan kichik yadro tilidagi tuyulishi va osonlikcha kengaytiriladigan tarjimon uning qarama-qarshiligini qo’llab-quvvatlaydigan ABC bilan hayajonlanishidan kelib chiqdi.

Python oddiy, kamroq murakkab sintaksis va grammatikaga intilishadi, bu esa ishlab chiquvchilarga kodlash metodologiyasini tanlash imkonini beradi. Perlning "buni amalga oshirishning bir nechtasi" shioridan farqli o’laroq, Python "dizayndagi falsafiylikni yagona va eng maqbul tarzda amalga oshirishning yagona yo’li bo’lishi kerak". Python dasturiy ta’minot fondi va Python kitob mualliflaridan Aleksandr Martelli "Python madaniyatida iltifot deb qaralmaydi", deb yozadi.

Python ishlab chiquvchilari tezroq optimallashni oldini olishga harakat qilishadi va CPython mos yozuvi dasturining noaniq qismlariga yamoqlarni rad etishadi, bu esa ravshanlik qiymatida tezlikni marginal oshiradi. [53] Vaqt muhim bo’lsa, Python dasturchi vaqtinchalik funktsiyalarni C kabi tillarda yozilgan kengaytmali modullarga ko’chirishingiz yoki PyPy-dan vaqtinchalik derazadan foydalanishingiz mumkin. Bundan tashqari, Cython ham Python skriptini C-ga aylantiradi va Python tarjimoniga to’g’ridan-to’g’ri C-level API chaqiruvlarini amalga oshiradi.

Python jamoasida keng tarqalgan neologizm pythonik bo’lib, u dasturiy uslub bilan bog’liq keng ma’nolarga ega bo’lishi mumkin. Python kodi Python tilidan yaxshi foydalanganligini aytish mumkin, bu tabiiy yoki u tilni ravonroq qilishini Pythonning minimalist falsafasi va o’qilishi mumkinligiga urg’u beradi degani. Bunga javoban, boshqa dasturiy tilidan qo’pol transkripsiya kabi tushunilishi qiyin bo’lgan kodni "uniktonik" deyiladi.

Python foydalanuvchilari va muxlislari, ayniqsa bilimdon yoki tajribali deb hisoblanganlar ko’pincha Pythonistlar, Pythonistalar va Pythoners deb nomlanadi.

1.3. Hisob va boshqarish oqimi

Belgilash bayonoti (token ‘=‘, teng belgisi). Bu an’anaviy imperativ dasturlash tillaridan farq qiladi va bu asosiy mexanizm (Pythonning o’zgaruvchan versiyasi tabiatini o’z ichiga olgan holda) tilning boshqa xususiyatlarini yoritadi. Cda tayinlash, masalan, x = 2, "yoziladigan o’zgaruvchining nomi x qiymatining 2 nusxasini oladi". (O’ng tomon) qiymati ajratilgan saqlash joyiga ko’chiriladi, u uchun (chapdan) o’zgarmaydigan nom ramziy manzildir. O’zgaruvchilarga ajratilgan xotira, e’lon qilingan turga etarlicha katta (ehtimol ancha katta). Python tayinlashning oddiy holida, xuddi shu misoldan foydalanib, x = 2 «(umumiy) nomga tarjima qilinadi x x, qiymatning raqamli (int) turidagi alohida, dinamik ravishda ajratilgan ob’ektga havola oladi. Bunga ismni ob’ektga bog’lash deb ataladi. Ismi saqlash joyi ko’rsatilgan qiymatni o’z ichiga olmaganligi sababli, bu o’zgaruvchiga nom berish noto’g’ri. Nomlar keyinchalik har qanday vaqtda turli xil turdagi ob’ektlar, shu jumladan strings, tartib-qoidalar, ma’lumotlar va usullar bilan murakkab ob’ektlar va hokazolarga reboundatsiyalanishi mumkin. Umumiy qiymatning bir nechta nomlarga, masalan, x = 2 ga ketma-ket tayinlanishi; y = 2; z = 2 uchta ism va bitta raqamli ob’ektni saqlash uchun (eng ko’p) uchta ismning bog’langan bo’lishiga sabab bo’ladi. Nomlar umumiy ma’lumotnoma egasi ekanligi sababli, u bilan bog’liq bo’lgan ma’lumotlar turini bog’lash asossizdir. Shu bilan birga, ma’lum bir vaqt ichida nomga bir turga ega bo’lgan ba’zi narsalar bog’lanadi; shuning uchun dinamik yozish bor.

-Agar kod blokini shartli ravishda bajaradigan if va ifodani boshqa va elif (else-if kichrayishi) bilan birga.

-For elementi Har bir elementni biriktirilgan blok tomonidan ishlatilishi uchun mahalliy o’zgaruvchiga tutib olish uchun, uni qaytariladigan ob’ektga aylantiradigan so’z uchun.

Kodning blokini amalga oshiradigan while statement, uning sharti to’g’ri

bo’lsa.

Qo’shib qo’yilgan kod blokida olib tashlangan istisnolar yuqoridagi moddalar tashqari ushlab turilishi va ko’rib chiqilishini ta’minlaydigan harakat kodi; shuningdek, blokning blokirovkada qanday bo’lishidan qat’i nazar, har doim ham ishlaydigan tozalash kodini ta’minlaydi.

Ob’ektga asoslangan dasturlashda foydalanish uchun kod blokini amalga oshiradigan va mahalliy ism maydonini sinfga biriktirgan sinf bayonoti.

Funktsiya yoki usulni belgilaydigan daf bayonoti.

Python 2.5 versiyasidan 2006 yil sentyabrda chiqarilgan, kontekst menejeri ichidagi kod bloklarini qamrab oluvchi (masalan, kod blokidan oldin qulfni sotib olish va undan keyin qulfni ochish yoki faylni ochish, keyin esa Resurslarni sotib olishni boshlashni boshlash (RAII) kabi xatti-harakatlarga yo’l qo’yib, umumiy sinash o’rnini bosadi.

Xatolikni tuzatish paytida ishlatilishi kerak bo’lgan shartlarni tekshirish uchun ishlatilgan tasdiqlov bayoni.

Jenerator funktsiyasidan qiymatni qaytaradigan daromad jadvali. Python 2.5 dan, rentabellik ham operator hisoblanadi. Ushbu ariza korrektsiyalarni qo’llash uchun ishlatiladi.

Joriy dasturda funktsiyalari yoki parametrlari ishlatilishi mumkin bo’lgan modullarni import qilish uchun ishlatiladigan import so’rovi. Importni ishlatishning uchta usuli mavjud: import <modul nomi> [<takrorlash> sifatida] yoki <modul nomi> import * dan yoki <modul nomi> importdan <ta’rif 1> [[[<alias 2> sifatida]

1.4. Python dasturlash tilini o’rnatish

Pythonni o’rnatish ancha sodda. Quyida windows 7 operatsion tizimiga qanday o’rnatilish jarayonini ko’rib chiqamiz

Microsoft Windows 7 uchun Pythonni o’rnatish uchun veb-brauzerdan http://www.python.org/ manziliga o’tib eng so’ngi windows o’rnatuvchisini yuklab oling.

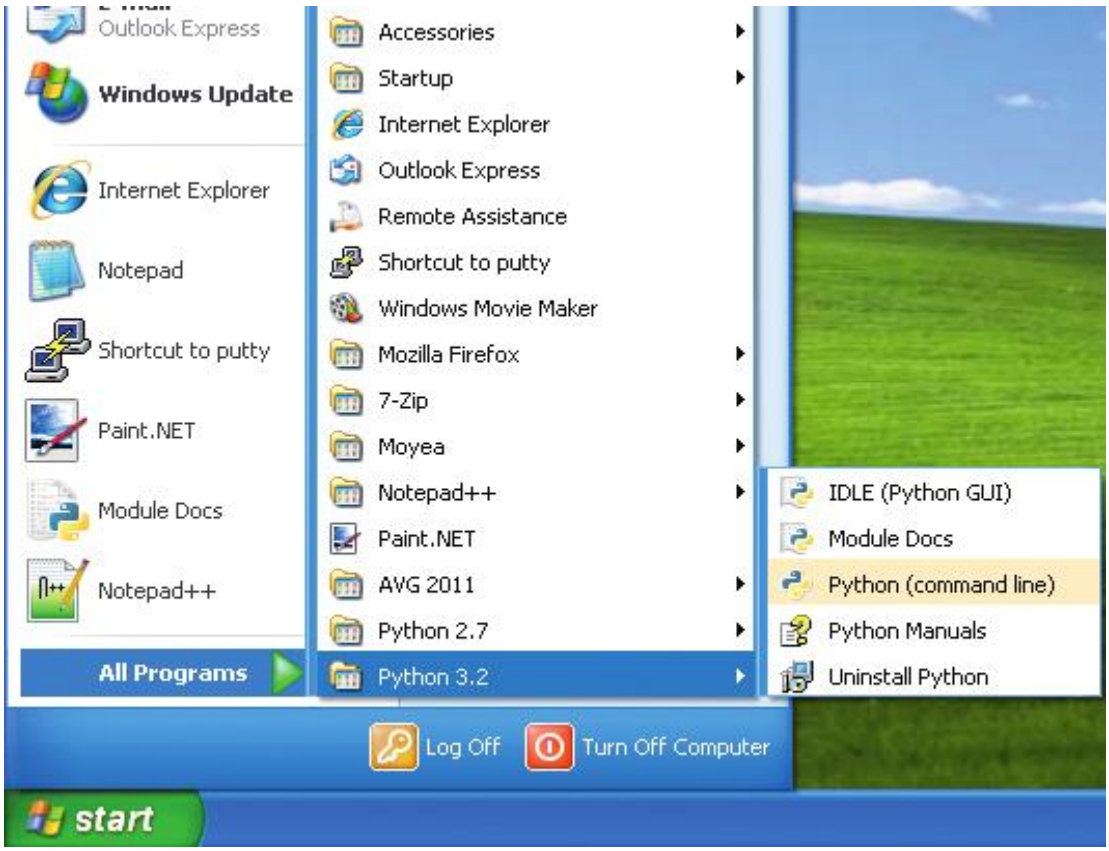
Windows o’rnatuvchini yuklaganingizdan so’ng uni ikki marta bosing va Python dasturini o’rnatish uchun ko’rsatmalarga amal qiling

1. Barcha foydalanuvchilarni o’rnatish-ni tanlang va so’ng oldinga ni bosing;

2. Standart katalogni o’zgartirmang, ammo nomini yozing (masalan: C: \ Python31 yoki C: \Python32) va keyingiga bosing;

3. O’rnatishni Python qismini moslashtiring va keyingiga bosing.

Ushbu jarayonning oxirida siz Python 3-ga kirishingiz kerak



Keyin, Python 3 yorliqni qo’shish uchun quyidagi bosqichlarni bajaring ish stoli:

1. Sizning ish stolingizni o’ng tugmasini bosib, New -> Shortcut -dan tanlang pop-up menyusidan foydalaning.

2. Quyidagilarni kiriting: Joyni kiriting (Siz kiritgan katalog bir xil ekanligiga ishonch hosil qiling avval aytib o’tganingizdek):

c:\Python32\Lib\idlelib\idle.pyw –n

Sizning muloqot oynangiz shunday bo’lishi kerak:

3. Keyingi dialogga o’tish uchun NEXT bosing.

4. IDLE deb nom kiriting va yaratish uchun Finish-ni bosing yorliq.

Endi siz "Python ni o’rnatganingizdan so’ng", Python bilan ishlashni boshlash mumkin

1.5. Python dasturlarini saqlash

Agar siz qayta yozish kerak bo’lsa, Python dasturlari juda foydali bo’lmaydi ularni har doim ishlatishni xohlaganingizda, ularni hech qachon bosmadan chiqaring Siz ularga murojaat qila olasiz. Albatta, adolatli bo’lishi mumkin qisqa dasturlarni qayta yozish, lekin katta dastur, masalan, savol, millionlab qator kodlari bo’lishi mumkin. Hammasi chop eting, va 100 mingdan ortiq sahifa bo’lishi mumkin. Faqat tasavvur qilib ko’ring ulkan qog’ozli qog’ozlar to’plamini olib yurasiz. Yaxshiroq umid qila olmaysiz katta shamol shamoli bilan uchrashish.

Yaxshiyamki, biz dasturlarni kelajakda foydalanish uchun saqlashimiz mumkin. Saqlash uchun yangi dastur, IDLE-ni oching va File->New Window-ni tanlang. Bo’sh menyu ko’rinadi, menyu satrida * Untitled *.

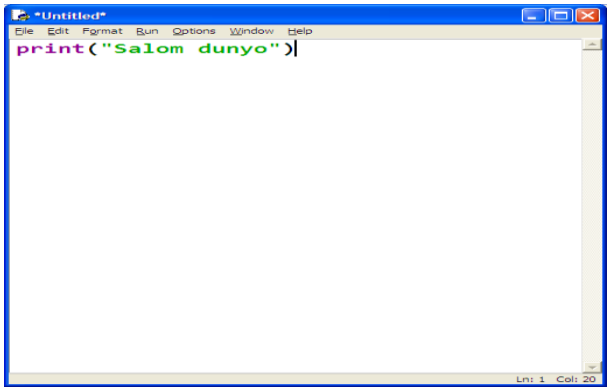
Quyidagi kodni yangi qobiq oynasiga kiriting:

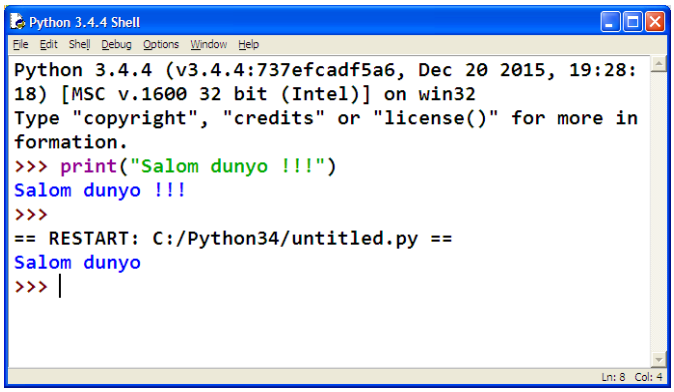
```
print("Salom dunyo")
```

Endi File->Save-ni tanlang. Fayl nomi so’ralganda, kiriting

untitled.py faylini ish stoliga saqlang. Keyin Run->Run Module tanlang.

Har qanday imkoniyat bilan sizning saqlangan dasturingiz quyidagi tarzda bajarilishi kerak:



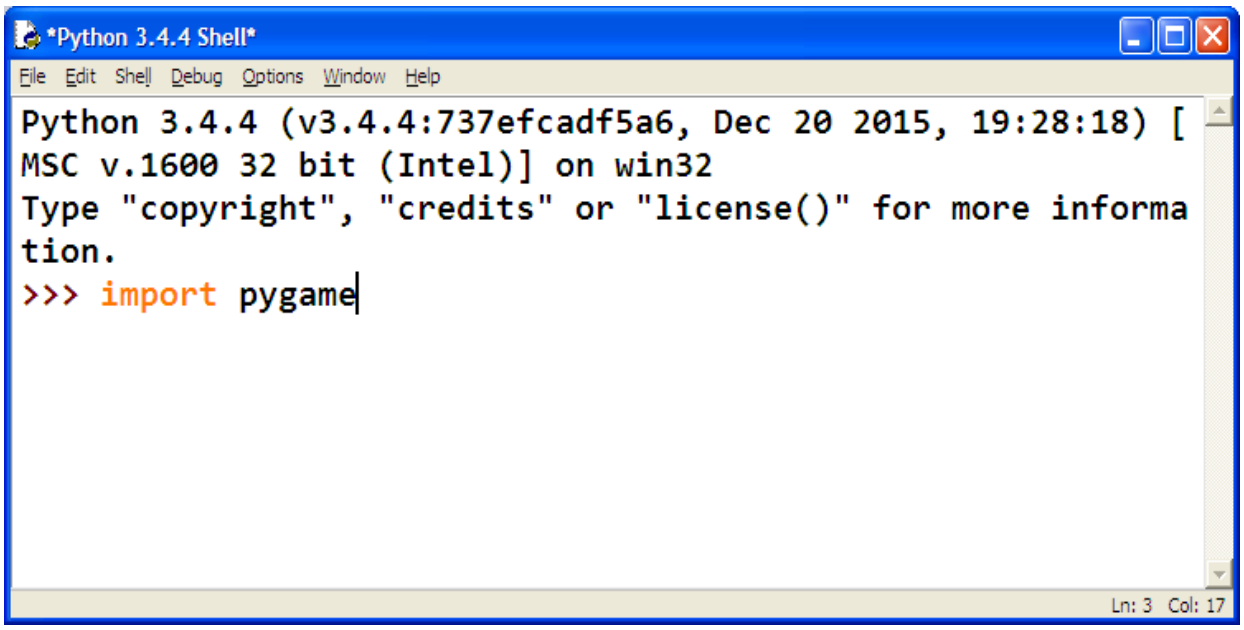


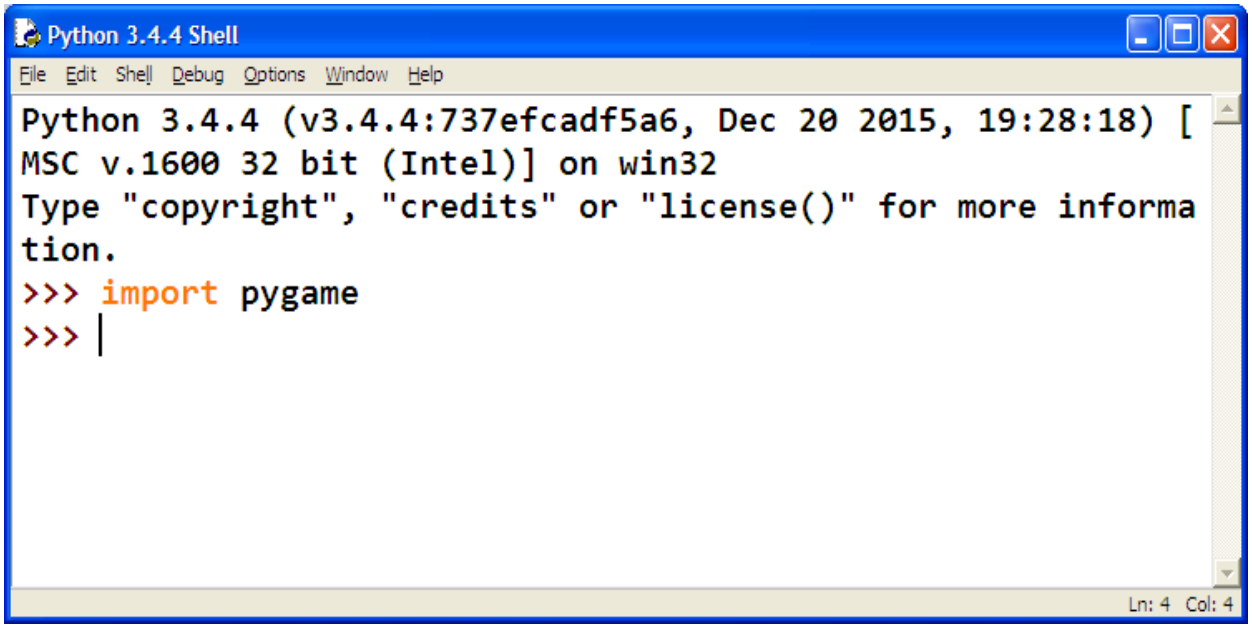
Pygame- ni o’rnatish

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’tamiz va "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 olishni xohlamaysiz operatsion tizim. 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 .msi faylini 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:





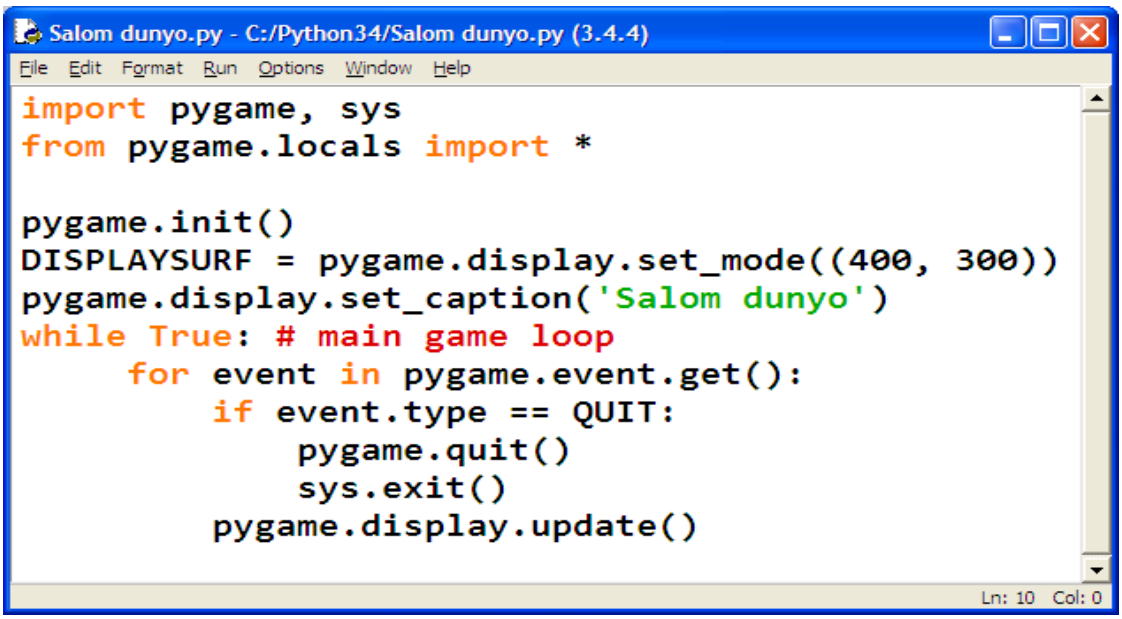
Salom dunyo uchun Pygame bilan Source Code

Pygame bilan tuzilgan birinchi dasturimiz - bu derazani ochadigan kichik dastur

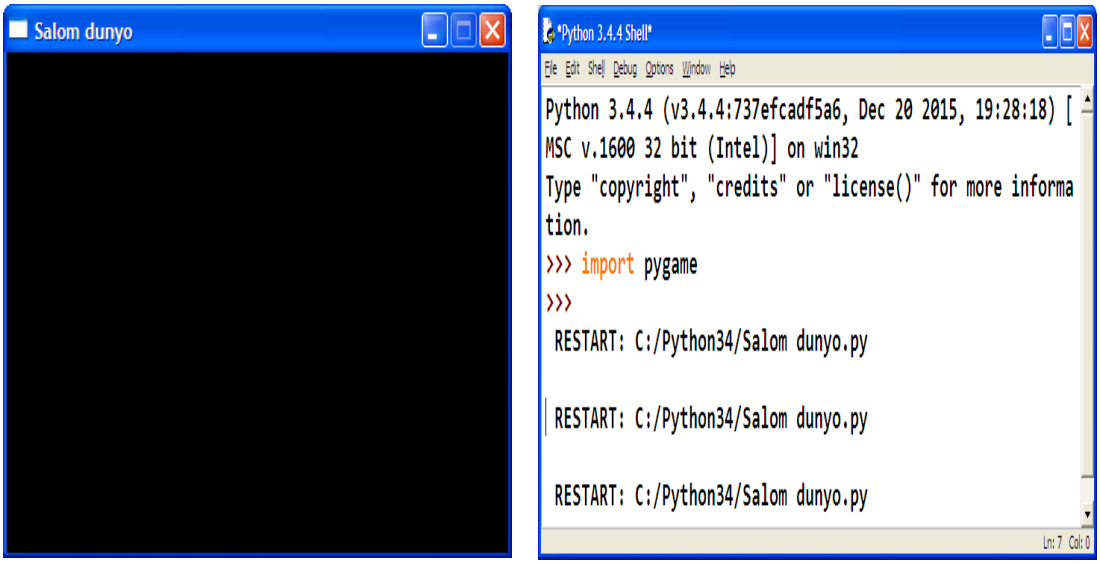
Dunyo!! "Ekranda paydo bo'ladi. IDLE ning File menyusiga bosish orqali yangi fayl muharriri oynasini oching,

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 (faqatgina bu Ushbu kitobda namunalar).



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 boshlanadi Pygame 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 pygame modulida 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 import modullarini 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()
```

4-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))
```

5-qator - bu *pygame.display.set_mode* () funktsiyasiga chaqiruv window uchun pygame.Surface obyekti. (Er yuzidagi ob'ektlar ushbu bobning oxirida tasvirlangan.). Shuni e'tiborga olish kerakki, biz funksiya uchun ikkita tamsayıning bir tayanch qiymatini o'tkazamiz: (400, 300). Bu tuple aytadi *set_mode* () funksiyasi oynani piksellarda qanchalik keng va qanchalik baland qilishini aniqlang. (400, 300) kengligi 400 piksel va balandligi 300 pikselli deraza yaratadi. Set_mode () funktsiyasiga ikkita tamsaytning ikkita sonini emas, balki ikkita tamsayıni 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!')
```

6-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 chaqiruvidan o'tgan:

Game Loops and Game States

```
7. while True: # main game loop
8.     for event in pygame.event.get():
```

7-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

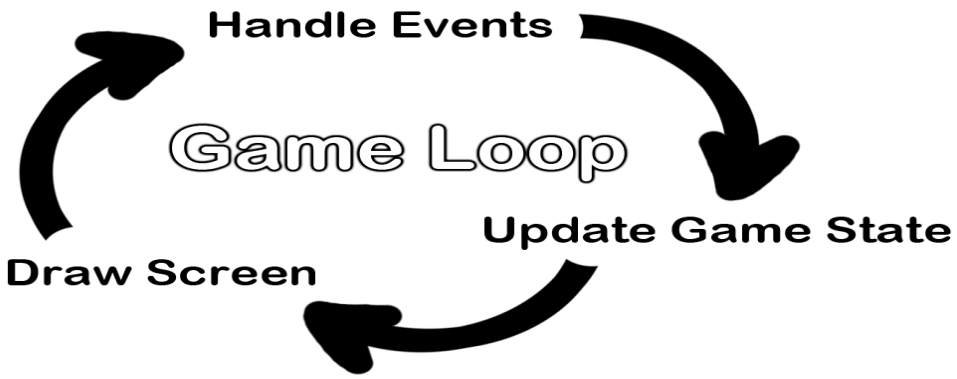
sys.exit () dasturini tugatadi. Agar bunday funktsiyaning bir funktsiyasi ichida bo’lsa, orqaga qaytish iborasi ijroni ko’chadan chiqarib tashlaydi (shuningdek, funktsiya ham). Bu kitobdagi o’yinlarning barchasi bularning barchasini o’z ichiga oladi bu - o’yinning aylanishi ". O’yin loopi (asosiy pastadir deb ham ataladi) kodning bajarilishida pastadir uch narsa:

1.Voqealarni ushlaydi.

2. O’yin holati yangilanadi.

3. 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 tugmachasini bosish yoki dasturning oynasiga sichqonchani suring, apygame.event.Event obyekti "-event" ni yozish uchun Pygame kutubxonasi tomonidan yaratilgan. (Bu Voqealar deb 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 pygame.event.get():
```

8-satr qaytarilgan Voqealar moslamalari ro'yhatining ustida yineluvchi loop uchun *pygame.event.get* (). Har bir iteratsiya uchun for loop orqali voqea deb nomlanuvchi o'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 bir uchun 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 sabab bo’ladi.

Bizda boshqa ob’ektlar uchun kodni ishlatadigan so’zlar yo’q ekan, foydalanuvchi sichqonchani bosganida, klaviatura tugmachalarini bosib yoki boshqa biror narsaga olib kelishi uchun hech qanday voqea kodi mavjud emas. yaratiladigan Voqealar 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, muntazam QUIT 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 Pygamani chizish funktsiyalari.

1.5. Amallar bajarilish ketma-ketligi

Bizga ma’lum ko’paytirish operatori qo’shish operatoriga qaraganda katta prioritega ega. Quyidagi jadvalda Python tilining operatorlari prioriteti ko’rsatilgan. Bunda yuqoridan pastga qarab Python tili operatorlari prioriteti oshib boradi. Bu shuni bildiradiki, ixtiyoriy ifodada Python oldin eng quyidagi operatorlarni hisoblaydi, keyin yuqoridagilarini. Amaliyotda esa amallarni qavslar bilan aniq ajratib yozish tavsiya etiladi. Bu dastur kodini qiyinchiliksiz o’qishga yordam beradi.

#	izoh
lambda	lambda ifoda
or	mantiqiy ‘yoki’
and	mantiqiy ‘va’
not x	mantiqiy ‘emas’
in, not in	tegishlilikga tekshirish
is, is not	bir xillikga tekshirish
	Taqqoslash

&	‘va’ bit operatori
<< >>	Surilishlar
+, -	Qo’shish va ayirish
*, /, //, %	ko’paytirish, bo’lish, qoldiqsiz bo’lish va qoldiqlik bo’lish
+x, -x	Musbat va manfiy
~x	‘emas’ bit operatori
**	Darajaga ko’tarish
x.attribute	Atributga link
x[indeks]	Indeks bo’yicha murojat
x[indeks1:indeks2]	Kesib olish
f(argument)	Funksiyani chaqirish
(ifoda, ...)	Kortej (Связка или кортеж)
[ifoda, ...]	Ro’yxat (Список)
{kalit:qiymat, ...}	Lug’at (Словарь)

1.2.2-chizma. Python operatorlari prioriteti.

Bu jadvalda bir xil prioritetga ega bo’lgan operatorlar bir qatorda joylashgan. Misol uchun ‘+’ va ‘-’.

Break operatori

Break operatori agar siklning bajarilish sharti hali YOLG’ON qiymat olmagan bo’lsa ham yoki ketma-ketlik elementlari hali tugamagan bo’lsada siklni to’xtatish, buyruqlar bajarilishini to’xtatish uchun xizmat qiladi. Shuni aytib o’tish

kerakki, for yoki while sikllari break operatoi bilan to’xtatilsa, ularga tegishli bo’lgan else bloki bajarilmaydi.

Continue opratori Continue opratori joriy blokdagi o’zidan keyingi qolgan barcha buyruqlarni bajarmay siklning keyingi iteratsiyasidan davom etirish uchun ishlatiladi.

Funsiya, fayllar bilan ishlash, sanoq sistemasi va son. Funksiyani aniqlash. Def kalit so’zi funksiyani aniqlashni taqdim etadi. Def so’zidan so ng funksiya nomi va qavs ichida formal parametrlar ro’yxati ko’rsatiladi. Funksiya tanasini hosil qiluvchi instruksiyalar keyingi qatordan boshlab bo’sh joy(oTCTynb) bilan yoziladi.

Python dasturiga kiritilgan funksiyalar. Tiplarni o’zgartiruvchi funksiyalar. bool(x)- rostlikka tekshirishni standart usulidan foydalanuvchi bool tipiga o’zgartirish. Agar x yolg’on bo’lsa yoki tushirib qoldirilgan bo’lsa, False qiymatini qaytaradi, aksincha esa True qaytaradi.

bytearray([manba, [kodlash[xatolar]]])- bytearray ga o’zgartirish. Bytearray- 0<=x<256 diapazondagi butun sonlarni o’zgarmas ketma-ketligi. Konstruktor argumentlari bytearray() ga mos ko’rinishga ega bo’ladi. complex([real],[image])- kompleks songa o’zgartirish. dict(object)- lug’atga o’zg artirish.

float([x])-haqiqiy songa o’zgartirish. Agar argument ko’rsatilmagan bo’lsa, 0.0 qaytaradi.

frozenset([ketma-ketlik])

int([object],[asosiy sanoq sistemasi])- butun sonni berilgan sanoq sistemasidan o’nlik sanoq sistemasiga o’tkazish. list([object])-ro’yxat tuzadi.

memoryview(object)- memoryview obyektini tuzadi. object()-hamma obyektlarga asos bo’lgan bosh obyektni qaytaradi. range([start=0], stop,[step=1])- step qadamli start dan stop gacha bo’lgan arifmetik progressiya.

set(object)-to’plamni yaratadi.

slice([start=0], stop, [step=1])-step qadamga ega startdan stopgachaga bo’lgan kesma obekti.

tuple(obj)- kortejga o’zgartiradi

Qo’shimcha funksiyalar abs(x)- absolyut raqamni

(sonni modulini) qaytaradi.

all(ketmaketlik)- agarda hamma elementlar haqiqiy bo’lsa (yoki ketmaketlik bo’sh bo’lsa) True ni qaytaradi.

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

chr(x)- x ning Unicode ga mos belgini qaytaradi.

classmethod(x)- sinf metodi ko’rsatgan funksiyani taqdim etadi.

compile(source, filename, mode, flags=0, don’t_inherit=False)- ketmaketlik eval yoki exec funksiyalari bilan bajariladigan dastur kodiga komplyatsiya qilinishi. Qator karetkani qaytaruvchi belgilar yoki nolga teng baytlarga ega bo’lmasligi kerak. delattr(object, name)- “name” nomidan atributni qaytaradi.

dir([object])- obyekt nomlarining ro’yxati, agar obyekt ko’rsatilmagan bo’lsa, local maydondagi nomlar ro’yxati. divmod(a,b) - a ni b ga bo’lganda hosil bo’lgan bo’linmaning butun va qoldiq qismi.

enumerate(iterable, start=0)- nomer va unga mos ketmaketlik a’zosidan tarkib topgan kortejni har bir o’tishda taqdim etuvchi iteratorni qaytaradi.

eval(expression, globals=None, locals=None)- dastur kodi qatorini bajaradi.

filter(function, iterable)- function yordamida rost qiymatni elementlarga qaytaruvchi iteratorni qaytaradi.

format(value [,format_spec])- formatlash (qatorni formatlash). getattr(object, name,[default])- obyekt atributini yoki default.globals()-global nomlar lugatini chiqaradi.

hasattr(object, name)- “name” nomidagi atribut obyektga ega ekanligini tekshiradi.

hash(x)- ko’rsatilgan obyektning heshini qaytaradi.

help([object])- dasturni yordam qismiga kiritilgan ma’lumotnoma tizimini chaqirish.

hex(x)- butun sonni o’n oltilik sanoq sistemasiga o’tkazish. id(object)-obyekt manzilini qaytaradi.

input([promt])- foydalanuvchi tomonidan kiritilgan qatorni qaytaradi. Promt-foydalanuvchiga yordam.

isinstance(object, ClassInfo)-agarda obyekt classinfo yoki uning sinfosti ekzemplari

bo’lsa rost qiymat qaytaradi. Agarda ekzemplar berilgan tipdagi obyekt bo’lmasa, funksiya yolg’on qiymat qaytaradi.

issubclass(sinf, ClassInfo)-agarda sinf Classslnfo sinfostisi bo’lsa rost qiymat qaytaradi. Sinf o’z-o’ziga sinfosti bo’ladi.

iter(x)- iterator obyektini qaytaradi.

len(x)-ko’rsatilgan obektni elementlar sonini qaytaradi.

locals()-lokal nomlar lug’ati.

map(function, iterator)-ketmaketlikning har bir elementiga function funksiyasini qo’llash orqali yaratiladigan iterator.

max(iter,[args...]*[, key])-ketma-ketlikning maksimal elementi. min(iter,[args...]*[, key])-ketmaketlikning minimal elementi. next(x)-iteratorning keyingi elementini qaytaradi. oct(x)- butun sonni sakkizlik sanoq sistemasiga o’tkazadi.

open(file, mode=‘r’, buffering=None, encoding=None, errors=None, newline=None, closefd=True)- faylni ochadi va kerakli oqimni qaytaradi. ord(x)-belgi kodi.

pow(x, y[,r])-(x**y)%r. reversed(object)-
yoyilgan obyektning iteratori.

print([object,...],*,sep=” ”, end=‘/n’, file=sys.stdout)- ma’lumotlarni ekranga chop etish.

round(X,[N])- verguldan keyin N- belgilargacha to’g’rilash.

setattr(obekt, nom, qiymat)- obyekt atributini belgilash.

sorted(iterable[, key][, reverse])- tartiblangan ro’yxat.

staticmethod(function)- funksiya uchun statistik metod.

sum(iter, start=0)-ketmaketlik elementlarini yig’indisi.

type(object)- obyekt tipini qaytaradi.

type(name, bases, dict)- name sinfidagi yangi ekzemplarni qaytaradi. vars([object])-obyekt atributlarining ro’yxati. Jimlik holatida- local nomlar lug’ati.

Fayllar bilan ishlash Fayllar bilan ishlash file klassi obyektini hosil qilish hamda uning read, readline va write metodlari yordamida amalga oshiriladi. Faylni o’qish yoki faylga yozish faylni ochish vaqtida ko’rsatilgan rejimga bog’liq. Fayl

bilan ishlab bo’lgandan keyin close metodini chaqirish kerak bo’ladi.

Bu misolda biz birinchi navbatda faylni rejim ko’rsatgan holda open funksiyasi bilan ochyapmiz. Rejim o’qish uchun («r»), yozish uchun («w») yoki fayl oxiriga yozuvni qo’shish uchun («a») bo’lishi mumkin. Faylni yana qanday holda o’qish, yozish yoki matn qo’shish holatini ham ko’rsatish mumkin: («t») tekst ko’rinishida yoki («b») binar ko’rinishida.

Bizning holatda faylni yozish («w») rejimida ochyapmiz va write metodi yordamida matnni faylga yozyapmiz. Shundan so’ng faylni close metodi yordamida yopyapmiz. So’ng xuddi shu faylni o’qish rejimida ochamiz. Bu holda rejimni ko’rsatishga hojat yo’q. Sababi agar rejim ko’rsatilmasa, fayl o’qish rejimida ochiladi. Faylni qatorma-qator readline metodi yordamida, sikl ichida o’qib olamiz. Qachonki bu metod bo’sh qator qaytarsa, u holda bu biz faylning oxiriga yetib borganimizni anglatadi va break yordamida siklni to’xtatamiz.

Shundan so’ng print funksiyasi yordamida o’qib olinayotgan satrlarni ekranga chop qilamiz. Oxirida close metodi yordamida faylni yopamiz. Haqiqatda dastur matnni faylga yozganligini tekshirish uchun poem.txt faylini tekshirib ko’ring.

Sanoq sistemasining ishlatilishi

Maktab kursidagi informatika faninidan bizga ma’lumki, sonlar nafaqat o’nlik sanoq sistemasida balki boshqa sanoq sistemalarida ham bo’lishi mumkin. Masalan: kompyuter ikkilik sanoq sistemasidan foydalanadi ya’ni 19-soni ikkilik sanoq sistemasida (kompyuterda) 10011 ko’rinishida ifodalanadi. Bundan tashqari sonlarni bir sanoq sistemasidan ikkinchi sanoq sistemasiga o’tkazish kerak. Python bu uchun bir qancha funksiyalarni taqdim etadi:

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

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

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

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

»> bir. (19)

1 **ОЫ О О11 ■**

»> oct. (19)

■0o23[1] »> hex (19)

■ 3x13’

»> int (‘10011’ ,2} 19

»> int ([1] 0Ы0011 ‘,2)

19

>» a=ir.t ([1]19 [1] } # 3atrr.i зог.^a o’tkazadi

>>> b=ir_t (19 . 5} # haqiqiy зог.г.1 butun qi3ir.ir.i qaytaradi >» prir.t (a,b)

19 19

Son

Sonlar Python dasturlash tilida 3 turda bo’ladi:

1. butun sonlar,
2. haqiqiy sonlar
3. kompleks sonlar

- Butun songa misol 2,5, ...
- Haqiqiy sonlarga misol 3.23 va 52.3e-4.
- Kompleks sonlarga misol (-5+4i) va (2.3-4.6i)

Butun sonlar

Python interpretatorida yuqorida operator va ifodalar mavzusida ko’rib chiqqan barcha operatorlarni oddiy matemetika kursida ishlatilganidek bajarilishini ko’rdik. Ya’ni ko’paytirish, qo’shish, ayirish, bo’lish, darajaga ko’tarish va hokazo. Endi esa butun sonlar ustida bajarish mumkin bo’lgan qo’shimcha metodlarni ko’ramiz.

int.bit_length()- sonni oldidagi ishora va nollarni hisobga olmasdan uni ikkilik sanoq sistemasida taqdim etish uchun kerakli bo’lgan bitlar soni.

```
>>> n=-37
>>> bin(n)
'-0b100101'
>>> n.bit_length()
6
```

int.to_bytes(length, byteorder, *, signed=False)-shu sonni taqdim etuvchi baytlar qatorini qaytaradi.

»> (1024) . to_bytes (2 , byteoxder^big[1])

Ь'\л04\x00'

»> (1024) . to_bytes (10, byteorder='big[1])

Ь'\л00\x00\я00\x00\я00\x00\я00\x00\я04\x00'

»> (-1024) .to_bytes(10, byteorder='big[1], signed= j.e) b[1] \xf f\xf f\xf f\xf f\xf f\xf f\xf f\xf f\xf cXxOI' [1] >>> x=1000

>>> x. to_bytes ((x.bit_lengtli () // B} 4 1, byteorder=[1] little [1])

fc'\xeS\x03'

classmethod int.from_bytes(bytes, byteorder, *, signed=False)-berilgan baytlar qatoriga mos sonni qaytaradi.

>>> ir.t. f rom_bytes (b [1] \x00\xl0 ' , byteorder= ' big[1])

16

>>> ir.t. f rorr._bytes (b [1] \x00\xl0 [1] , byteorder='little [1])

4096

>>> ir.t. f rom_bytes (b [1] \xf cжOO ' , byteorder= ' big[1] , signed= r ;e)

-1024

»> ir.t. f rorr._bytes (b [1] \xf c\xOO [1] , byteorder='big[1] , signed= alse)

64512

>>> ir.t. froni_bytes ([255, 0, 0], byteorder= 'big[1])

167116B:>

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 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)

1.4.1-chizma. Ekran bilan ishlash ketma-ketliklari. Ko’p qatorli satrlar

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

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.

II BOB. PYTHONDA MA’LUMOTLAR TUZILMASI VA TILNING

2.1.Standart modullar

Pythonda erkin turdagi ob’ektlarning o’zgaruvchan qatorlashgan kolleksiyasi hisoblanadi (massivga o’xshash, lekin tiplar har xil bo’lishi mumkin). Ro’yxatlardan foydalanish uchun ularni tuzish kerak. Ro’yxatni har xil yondashuvlar yordamida yaratish mumkin. Masalan har bir iteratsiya qilinadigan obyektni (masalan satrni) Pythonni o’ziga kiritilgan list funksiyasi yordamida kiritish mumkin. Ro’yxatni yana literallar yordamida tuzish mumkin.

Misoldan ko’rinadiki ro’yxat istalgancha obyektdan yoki hech narsadan (bo’sh) tashkil topishi mumkin.

Ro’yxat yaratishning yana bir usuli- ro’yxatlarning generatorlari. Ro’yxat generatori bu- ketma-ketlikni har bir elementiga arifmetik amalni qo’llab yangi ro’yxat tuzish usuli. Generatorlar for sikliga juda o’xshash bo’ladi.

```
>>> li3t(1 re’yxat1)
[ ■r■, ‘c1, 1 1 , ‘y ’f ‘x1, " a ’T 1 1 1]
```

Ro’yxatlar generatorining juda murakkab konstruksiyalari bor.

```
»> c = [c * 3 for c in 1 list1 if c != 1 i1 ]
»> c
[‘1111, 1333 1 f ‘ttt1]
»> c = [c + d for c in ‘list1 if c != ‘i1 for d in ‘spam1 if d != ‘a1] »> c
[‘Is’, Up’, ‘lm1, ‘ss1, 1p’, ‘sm1, ‘ts’, ‘tp1, 1tm1 ]
```

Ro’yxatning funksiya va metodlari

Ro’yxatni yaratgandan so’ng uning ustida turli amallarni bajarish kerak bo’ladi, albatta, buning uchun esa Pythonni o’ziga kiritilgan bir qancha funksiya va metodlar bor.

Metod	Vazifasi
List.append(x)	Ro’yxat oxiridan element qo’shish
List.extend(L)	Oxiriga hamma elementlarni qo’shib list ro’yxatini kengaytiradi.
List.insert(i,x)	i-elementga x qiymatini kiritadi
List.remove(x)	Ro’yxatdan x qiymatga ega elementni o’chiradi
List.pop([i])	Ro’yxatning i-elementini o’chiradi va qaytaradi. Agarda indeks ko’rsatilmagan bo’lsa oxirgi element o’chiriladi
List. index(x,[start] ,[end])	X qiymatga teng start dan end gacha birinchi elementni qaytaradi
List.count(x)	X qiymatga teng elementlar sonini qaytaradi
List. sort([key=funksiya])	Funksiya asosida ro’yxatni saralaydi
List.reverse()	Ro’yxatni ochadi
List.copy()	Ro’txatning nusxasi
List.clear()	Ro’yxatni tozalaydi

2.1.1-chizma. Ro’yxat metodlari tasnifi

2.2. Kortejlar(tuple)

Kortejlar bir nechta ob’yektlarni birgalikda saqlashga xizmat qiladi. Ularni ro’yxatlarga o’xshatish mumkin. Lekin ular ro’yxatlar kabi boy funksionallikka ega emas. Ularning asosiy jihati qatorlarga o’xshab o’zgarmasliklaridir. Kortej-elementlar orasini vergul bilan ajratish orqali hosil qilinadi. Kortejga ma’no jihatdan o’zgarmas ro’yxat deb ta’rif berdik. Shu o’rinda savol tug’iladi. Ro’yxat bo’lsa kortej nimaga kerak:

1. Turli holatlardan himoyalanish. Bu degani kortej o’zgartirishlardan himoyalangan bo’ladi, rejali (bu yomon) va tasodifiy (bu yaxshi) o’zgarishlardan xalos bo’ladi.

2. Kichik hajm. So’zlar bilan ifodalamasdan.

3. Kortejdan lug’at kaliti sifatida foydalanish mumkin:

>>> d = { {l, л., Л.) : л.>

»> d {(1, 1, 1) : 1 »> ei = t [1, *1*, 1] : 1>

Trdceback [mo3t recer.t call last) :

File "<py3hell#7>"r llr.e 1, in <module> cl

= { [1, 1, 1] : i:

TypeError: unhdshable type: ‘ll3t[1]

Kortej afzalliklari haqida bilib oldik. Endi kortej bilan qanday ishlashni ko’ramiz.

Bu xuddi ro’yxatlar bilan ishlashga o’xshaydi. Bo’sh kortejni yaratamiz:

>>> a=tuple (} #PYthonning 3tandart Tлрlеf.л:кзlуаз1 yordamida >>> a 0

>» a=(}# kortej literal! yordamida

>>> a 0

Kortejning funksiya va metodlari

Count(x)-kortejdagi x elementi sonini qaytaradi.

Index(x)-kortejdagi x elementining indeksini qaytaradi.

Any()-agar kortej elementi mavjud bo’lsa True qiymat qaytaradi, aks holda (kortej bo’sh bo’lsa) False qiymat qaytaradi.

Max()-kortejning maksimal elementini qaytaradi.

Min()- kortejning minimal elementini qaytaradi.

Len()-kortejning uzunligini qaytaradi.

Sorted()-kortej elementlaridan iborat yangi tartiblangan ro’yxatni qaytaradi.

Sum()-kortej elementlari yig’indisini qaytaradi.

k=(‘2’j’12’j’9’j’7S’j’ 15 ‘ *[1]12[1]j) j=tuple() print(k)

print (‘ kortej dagi x element soni [1] л k. count (‘ 12 ‘))

print(‘kortejdagi x element indeksi [1] r k. indexf [1] 78 [1]))

print(‘kortejni tekshirish[1]^any(k))

printf’kortejni tekshirish’.,any(j))

print(‘max elementjmax(k))

print(‘min elementлmin(k))

print(‘kortej uzunligi:[1]jlen(k))

(¹ 2 ¹ , ‘ 12 ‘ , ‘ 9 ‘ , TB", ‘ 15 ‘ , *‘12’)*
kortejdagi x element soni 2
kortejdagi x element indeksi 3
kortejni tekshirish True kortejni
tekshirish False max element: 9 min
element: 12 kortej uzunligi: *6*
[¹12 ¹ , ¹12 ¹ , ‘IS’л ¹ *2* ‘7S\ ¹ 9 ¹]

Elementni kortejga tegishli ekanligini tekshirish uchun *i n* kalit so’zidan foydalaniladi: loortej = ([T]s ‘l’j’o’j’n’j,)

print^’o[1] in kortej)

Natija:

True

2.2. Modul tushunchasi, standart kutubxonalar, sys va copy moduli.

Standart kutubxona modullarini o’rganishdan oldin Pythonda modul tushunchasiga aniqlik kiritib olish lozim.

Python tilida bir xil vazifani bajaruvchi modullar yig’indisini bitta paketga joylashtirish mumkin. Shunday paketlardan biri sifatida XML paketini misol qilib keltirish mumkin. Ushbu paket XML ning har xil aspektlarini qayta ishlashga mo’ljallangan modullardan tashkil topgan.

Python tilda dastur tuzishda modul atributlari modulda aniqlangan nomlar bo’lgan obyekt modul sifatida taqdim etiladi.

>>> dl=datetime.date(2017

Bu misolda datetime moduli import qilinayapti. Import operatorining ishi natijasida mazkur nomlar kengligida datetime nomi bilan obyekt paydo bo’lyapti.

Python tilida modullar oddiy (Pythonda yozilgan) va kengaytiriladigan ya’ni boshqa tilda yoziladigan masalan Python interpretatori yozilgan C dasturlash tilida yozilgan modullarga bo’linadi. Foydalanuvchi nuqtai nazarida ular ishlash tezligi

bilan farq qiladi. Satandart kutubxonada modul 2 xil variantda bo'ladi: Pytonda yozigan yoki C da. Bunga misol sifatida *pickle* va *cpickle* modullarini keltirish mumkin. Odatda Pythonda oddiy modullar kengaytirilgan modullarga nisbatan ishlatishda qulay hisoblanadi.

Modul funksiyalaridan foydalanish uchun uni boshqa dasturdan yuklash (импортировать) mumkin. Dastlab standart kutubxonalar modullarini qanday ishlatishni ko'raylik.

Natija:

1. $ Python3 using_sys.py biz argumentlamniz
2. Buyruqlar qatori argiumentlari :
3. using_sys.py
4. biz
5. argumentlarmiz
6. D'zgaruvchi PYTHONPATH qiymati ["/home/user/python darslari/7- dans'j '/usr/lib/pyrhonB.4[1]^ [1]/usr/lib/python3.4/plat-x86_64- linux-gnu[1]j '/usr/lib/python3.4/llb-dynload', '/Lisr/local/lib/pythor3 .4/dist-packages " , ' /usr/lib/pyt:hon3/dist pac kages']

Bu misolda dastlab sys moduli import buyrug'i yordamida yuklanyapti. sys moduli Python interpretatoriga va uning muxitiga ya'ni tizimiga (system) tegishli funksiyalardan tashkil topgan. Python import sys buyrug'ini bajarayotganda sys modulini qidiradi. Bu holatda sys standart modullardan biri bo'lganligi uchun, Python uni qayerdan izlash kerakligini biladi. Agar bu oddiy modul, ya'ni Pythonda yozilgan modul bo'lganida edi, u holda Python uni sys.path ko'rsatilgan kataloglardan izlagan bo'lar edi. Agar modul topilsa, undagi buyruqlar bajariladi va bu modul foydalanishga (доступным) shay holatga keladi. sys modulidagi argv o'zgaruvchisiga murojat qilish nuqta orqali amalga oshiriladi ya'ni sys.argv. Bunday ifodalashning afzalligi dasturda ishlatilishi mumkin bo'lgan argv

o’zgaruvchisi bilan xatoliklar yuz bermaydi. sys.argv qatorlar ro’yxati hisoblanadi. U buyruqlar qatori argumentlaridan ya’ni buyruqlar qatoridan dasturga uzatilgan argumentlardan tashkil topgan.

Python dasturida modullarni ulash *import* operatori orqali amalga oshirilishi yuqoridagi misolda ko’rdik. Modullarni ulashini ham 2 xil shakli mavjud: birinchisi *import* operatori orqali bo’lsa, ikkinchisi *from-import* operatori orqalidir. From... import ... operatori- argv o’zgaruvchisini dasturga to’g’ridan-to’g’ri yuklash uchun hamda har doim sys.argv deb yozmaslik uchun, from sys import argv ifodasidan foydalanish mumkin. sys modulida ishlatiladigan hamma nomlarni yuklash uchun “from sys import *” buyrug’ini bajarish mumkin.

```
1. from math import
2.
2. n = int(input(‘Biror son kiriting: ‘))
4.
3. print{sqrt(n)}
```

2.3.Standart kutubxonalar

Python tili standart kutubxonasining modullarini shartli ravishda mavzular bo’yicha quyidagi guruhlarga ajratish mumkin:

1. Bajarish davri servislari. Modular: sys, atexit, copy, traceback, math, cmath, random, time, calendar, datetime, sets, array, struct, intertools, locale, gettext.
2. Siklni qayta ishlashni qo’llab-quvvatlovchi. Modullar: pdb, hotshot, profile, unittest, pydoc. Paketlar: docutils, distutils.
3. OS (fayllar, protseslar) bilan ishlash. Modullar: os, os.path, getopt, glob, popen2, shutil, select, signal, stat, tempfile.
4. Matnlarni qayta ishlovchi. Modullar: string, re, StringlO, codecs, difflib, mmap, sgmllib, htmllib, htmlentitydefs. Paket: xml.
5. Ko’p oqimli hisoblashlar. Modullar: threading, thread, Queque.
6. Ma’lumotlarni saqlash. Arxivlash. Modullar: pickle, shelve, anydbm, gdbm, gzip, zlib, zipfile, bz2, csv, tarfile.
7. Platformaga tobe modullar. UNIX uchun: commands, pwd, grp, fcntl, resource,

termios, readline, rlcompleter. Windows uchun: msvcrt, _winreg, winsound.

8. Tarmoqni qo’llab-quvvatlash. Internet protokollari. Modullar: cgi, Cookie, urllib, urlparse, htttplib, smtplib, poplib, telnetlib, socket, asyncore. Serverlarga misollar: SocketServer, BaseHTTPServer, xmlrpclib, asynchat.

9. Internetni qo’llab-quvvatlash. Ma’lumotlar formatlari. Modullar: quopri, uu, base64, binhex, binascii, rfc822, mimetools, MimeWriter, multifile, mailbox. Paket: email.

10. Python uchun. Modullar: parser, symbol, token, keyword, incpect, tokenize, pyclbr, py_compile, compileall, dis, compiler.

11. Grafik interfeys. Modul: Tkinter.

Ko’pincha modullar o’zida bir yoki bir nechta sinflarni saqlaydilar. Bu sinflar yordamida kerakli tipdagi obyekt yaratiladi, lekin gap moduldagi nomlar haqida emas, aksincha shu obtekt atributi haqida boradi. Bir nechta modullar faqat erkin obyetlar ustida ishlash uchun umumiy bo’lgan funksiyalardan iborat bo’ladilar.

Sys moduli

Sys moduli Python interpretatorida dastumi bajaruvchi muhitdir. Quyida bu modulni eng ko’p qo’llaniladigan obyektlari keltilgan:

Exit([c])- dasturdan chiqish. Tugatishning raqamli kodini yuborish mumkin: agarda dastumi tugatish muvafaqqiyatli amalga oshsa 0 ni yuboradi, aksincha bo’lsa ya’ni xatolik yuz bersa boshqa raqamlarni yuboradi.

Argv- buyruqlar qatori argumentlari ro’yxati. Oddiy holatda sys.argv[0] buyruqlar qatoriga ishga tushirilgan dastur nomini va boshqa parametrlar yuboriladi.

Platform- interpretator ishlaydigan platforma.

Stdin, stdout, stderr- standart kiritish, chiqarish, xalolarni chiqarish. Ochiq faylli obyektlar.

Version- interpretator versiyasi.

Serecursionlimit(limit)- rekursiv chaqirishlarni maksimal kiritish darajasini o’rnatadi.

Exc_info()-kiritish-chiqarish istisnosi haqida ma’lumot.

Copy moduli

Bu modul obyektlarni nusxalashga mo’ljallangan funksiyalarga ega. Boshida Pyhtonda sal sarosimaga solish uchun “paradoks” ni ko’rib chiqish tavsiya etiladi.

```
lst1 = [0] * 3
lst = [lst1] * 3
lst[0][1] = 1
print(lst)
```

Va biz kutmagan natija paydo bo’ladi:

[[0, 0, 0], [0, 0, 0], [0, 0, 0]]

[[0, 1, 0], [0, 1, 0], [0, 1, 0]]

Gap shundaki bu yerda lst ro’yxati shu ro’yxatnining izohiga ega. Agarda rostdan ham ro’yxatni ko’paytirmoqchi bo’lsak, copy modulidagi copy() funksiyasini qo’llash kerak.

```
from copy import copy lst1 = [0, 0, 0]
lst = [copy(lst1) for i in range(3)]

print (lst) lst[0][1] = 1 print (lst)
```

Endi kutilgan natija paydo bo’ladi:

[[0, 0, 0], [0, 0, 0], [0, 0, 0]]

[[0, 1, 0], [0, 0, 0], [0, 0, 0]]

Copy modulida yuqori aniqalikda nusxalash uchun deepcopy() funksiyasi bor bu funksiya yordamida obektlar butun imkoniyati bilan rekursiv nusxalanadi.

2.4.Math, cmath, random va os moduli.

Math va cmath modullarida haqiqiy va compleksli argumentlar uchun matematik funksiyalar to’plangan. Bu C tilida foydalaniladigan funksiyalar. Quyida math modulining funksiyalari keltirilgan. Qayerda z harfi bilan argumentga belgilash kiritilgan bo’lsa, u cmath modulidagi analogik funksiya ham shunday belgilanishini bildiradi.

Acos(z)-arkkosinus z.

Asin(z)- arksinus z.

Atan(z)- arktangens z.

Atan2(y, x)- atan(y/x).

Ceil(x)- x ga teng yoki katta eng kichik butun son.

Cos(z)- kosinus z.

Cosh(x)- giperbolik x kosinusi. e- e konstantasi.

Exp(z)- eksponenta (bu degani e**z)

Fabs(x)-x absolute raqami.

Floor(x)- xga teng yoki kichik eng katta butun

son Fmod(x,y)- x ni y ga bo'lgandagi qoldiq

qism.

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) qaytarsa boshqa paytda 0.5<=abs(m)<1.0 bo'ladi.

Factorial(x)- x ning faktoriali. N!=1 *2*3*.. ,*n Hypot(x,y)- sqrt(x*x+y*y)

Ldexp(m,i)- m*(2**i).

Log(z)- natural logarifm z.

Log10(z)- o' nlik logarifm z.

Log2(z)-logarifm ikki asosga ko'ra z.

Modf(x)- (y,q) juftlikda x ning butun va kasr qismini

qaytaradi. p-pi konstantasi.

Pow(x,y)- x**y.

Sin(z)- z ning sinusi.

Sinh(z)- z ning giperbolik sinusi.

Sqrt(z)- z ning kvadrat ildizi.

Tan(z)- z ning tangensi.

Tanh(z)- z ning giperbolik tangensi.

Trunc(x)- x haqiqiy sonning butun qismini qaytaradi.

degrees(x)-x ni radiandan gradusga o’tkazish. radians(x)- x ni gradusdan radianga o’tkazish.

```
>>> impor t ra t
>>> ra a. t . factorial (1*3)
ЗЙ28800
>>> ra a. t . nvpot (3 ₹ 4}
5 .
>>> ra a. t . ceil ( Ю. a>
11
>>> ra a. t . ceil (Ю . 2 i
11
>>> ra a. t . loglO (1OO)
2 .
>>> ra a. t . log2(64)
6 .
>>> ra a. t . trunc(lO .6)
1 O
```

Random moduli

Bu modul har xil taqsimotlar uchun tasodifiy raqamlarni generatsiya qiladi. Eng ko’p qo’llaniladigan funksiyalari:

Random()-[0.0, 1.0) yarim ochiq diapozondagi tasodifiy sonlarni generatsiya qiladi. Choice(s)- s ketma- ketlikdan tasodifiy elementni tanlab oladi.

Shuffle(s)- s o’zgaruvchan ketma-ketlik elementlarini joyiga joylashtiradi. Randrange([start], stop, [step])- renge(start, stop, step) diapozondagi tasodifiy butun raqamni chiqaradi. Choice(range(start, stop, step)) ga analogik holatda.

Normalvariate(mu, sigma)- normal holatda taqsimlangan ketma-ketlikdan raqamni chiqaradi. Bu yerda mu- o’rtacha, sigma-o’rta kvadratli (sigma>0) sonlar.

Boshqa funksiyalar va uning parametrlarini hujjatlashdan aniqlab olish mumkin. Modulda qandaydir holatga tasodifiy raqamlar generatorini joylashtirishga imkon beruvchi seed(n) funksiyasi ham mavjud. Masalan: agarda bitta tasodifiy raqamlar ketma-ketligidan ko’p marta foydalanishga ehtiyoj sezilsa.

Time va Sets moduli Time moduli joriy vaqtni olish uchun va vaqt formatlarini o’zgartirish uchun fuksiyalarni taqdim etadi.

Sets moduli to’plamlar uchun ko’rsaatgichlar tipini amalga oshiradi. Quyidagi misol bu moduldan qanday foydalanish mumkinligini ko’rsatadi. Su o’rinda bilishimiz kerakki, Python 2.4 va undan yuqori versiyalarda set tipi sets

o’rniga kiritilgan.

```
import sets
A = sets.Set([1, 2, 3])
B = sets.Set([2, 3, 4])
print (A | B, A & B, A - B, A ^ B)
for i in A:
    if I in B: print (i,)
```

Natijada:

Set([1, 2, 3, 4]) Set([2, 3]) Set([1]) Set([1, 4]) 2 3

2.5. Array va struct modullari

Bu modullar past darajali massiv va korsatgichlar tuzilmasini amalgam oshiradi. Ularning asosiy vazifasi- ko’rsatgichlarning ikkilamchi formatlarini ko’rib chiqish.

Os moduli

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	Boshqa yo’lning elementlarini taqsimlovchi
Os.pathsep	Yo’llar ro’yxatidagi yo’llarni taqsimlovchi
Os.defpath	Yashirin yo’llar ro’yxati
Os.linesep	Satrni yakunlovchi belgi

2.4.1-chizma. Kataloglarni bo’luvchi os moduli ifodalari konstanta ko’rinishida

Pythondagi dastur operatsion tizimda alohida jarayon ko’rinishida ishlaydi. Os modulining funksiyalari protsesda, muhitda bajariladigan turli xildagi ahamiyatga ega bo’lgan kirishlarga ruxsat etadilar. Os modulining eng muhim ruxsat 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)- path nomli fayl yoki catalog ruxsat etish^ocTynb) 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.

III-BOP. Python dasturlash tilining C++ dasturlash tilidan afzalligi

3.1. Python va C++ ni solishtirish

Subaru Software Development jamoasi (SSD) C / C ++ dan Pythonga instrumentdan boshqarish dasturlarini ko’chiradi Python dasturlash tiliga ko’chiradi Ishlashning etishmasligi ikki til o’rtasidagi bevosita taqqoslash.

Migratsiya kodidan ishlash ko’rsatkichlari haqida ma’lumot olish kerak.

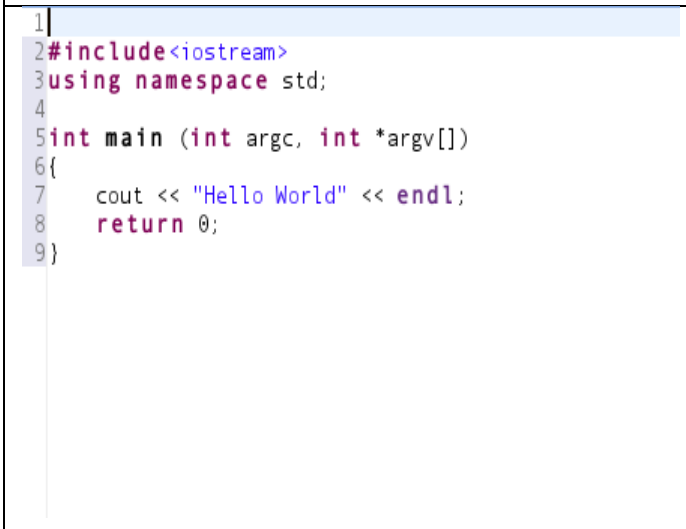
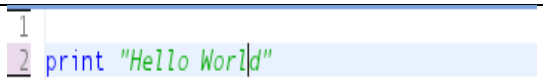
Mening loyiham da ular quyidagicha ifodalanadi

C / C ++ va Python da ishlash ko’rsatkichlarini sinash dasturini yozing.

Subaru Real Time tizimida dasturiy ta’minotni bajarish (AO188RTS)

Subaru-ning Wiki-dagi Wiki-sahifani natija bo’yicha tahlil qilish uchun qurish.

Nima uchun Python va uninig afzalligi ?

C++	Python
	
Murakkab sintaksisi O’qish qiyin	Minimal sintaksis O’qish va disk raskadrovka qilish osonroq Tezroq rivojlanish vaqti Dastur ishlachini oshiradi

3.2. Kompliyatsiya va interpretatsiya jaroyoni

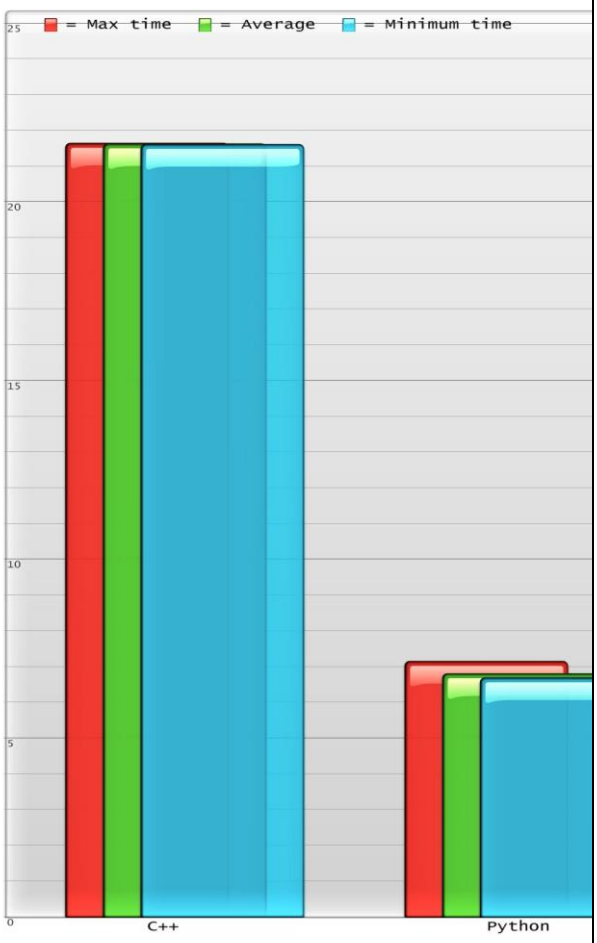
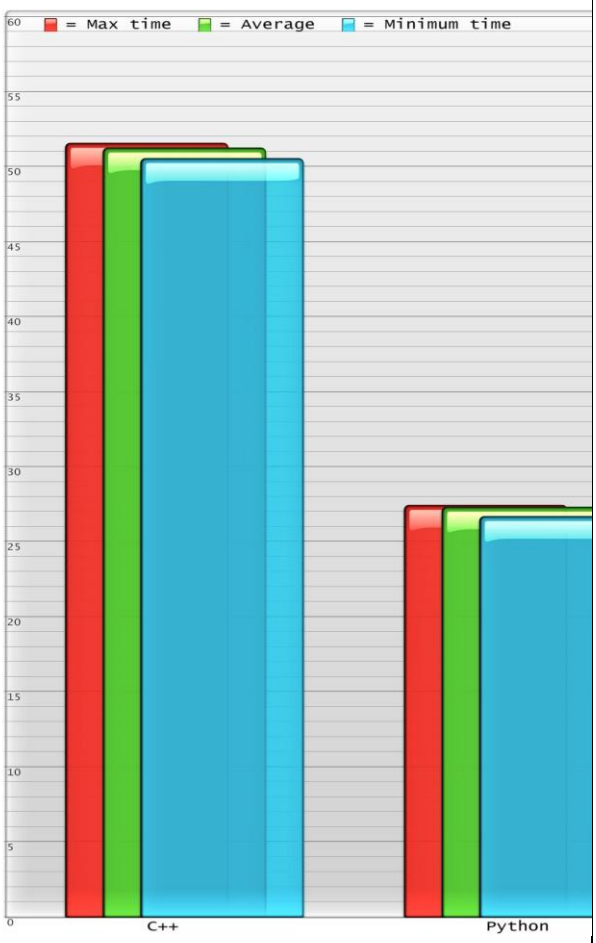
C++	Python
C ++ - bu kompilyatsiya qilingan til.	Python tarjima qilingan tildir.
Kod inson o’qiydiradigan matn shaklidan mashinaning o’qiy oladigan bajariladigan formatiga tarjima qilinadi.	Kod tarjimon dasturi tomonidan ish vaqtida kompyuter o’qilishi mumkin formatga tarjima qilinadi.
Qilingan kod apparatga xosdir.	Interpretatsiya kodi o’rnatilgan tarjimon bilan har qanday platformada ishlaydi.

Sinov muhiti

AO188RTS - adaptiv optik asboblarni boshqarish uchun Subaru Real Time tizimi 4x Intel Xeon 2GHz protsessorlari RedHawk 4 Linux Python v2.3, GCC v3.4.6 (eskirgan) Haqiqiy vaqtli tizim - boshqa jarayonlardan minimal aralashuv bilan darhol javob beradigan dasturiy ta’minot uchun mo’ljallangan maxsus kompyuter.	

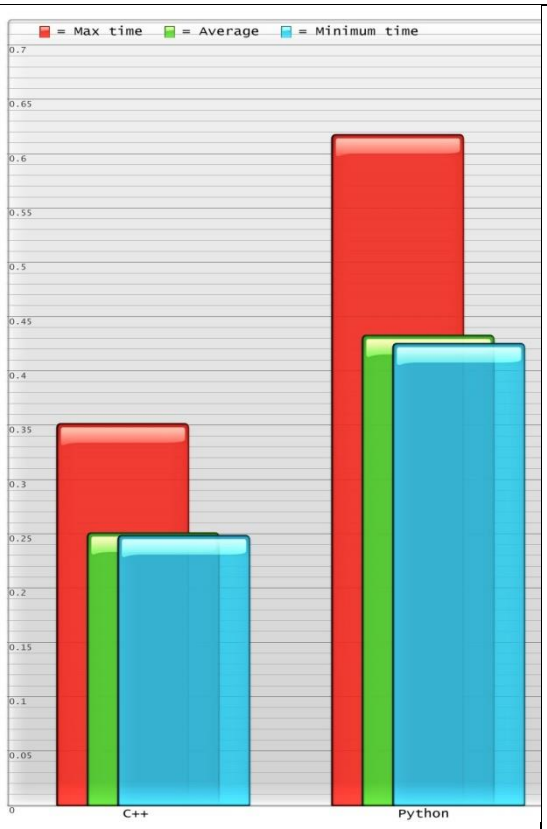
Solishtirich natijalar – Yaxshi

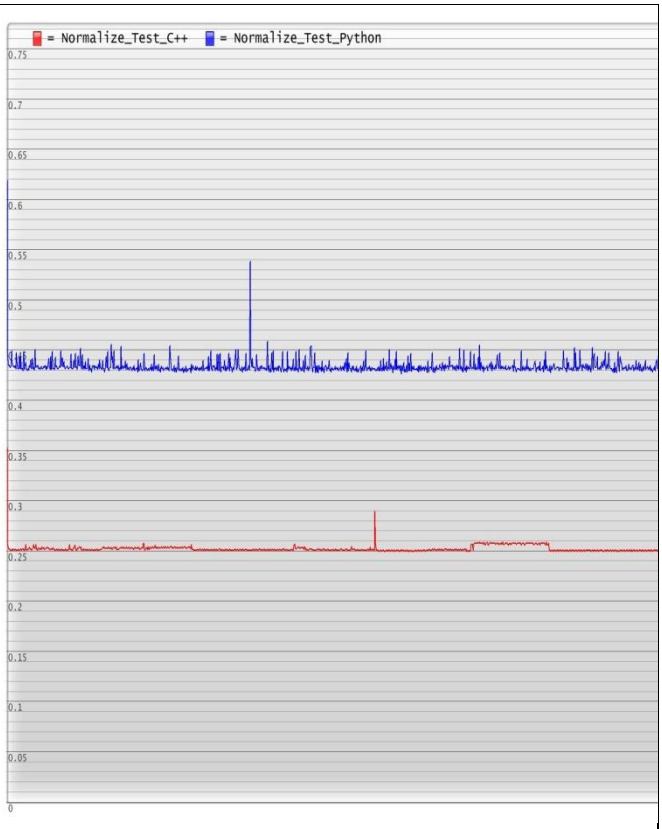
(Qisqacha yaxshi)

Qidirish(Searching)	Tartiblash(Sorting)
	
C++ Python	C++ Python
Runtime (ms) (Ishlach tezligi)	

Solishtirich natijalar – Yomon

Vektorni normallashtirish

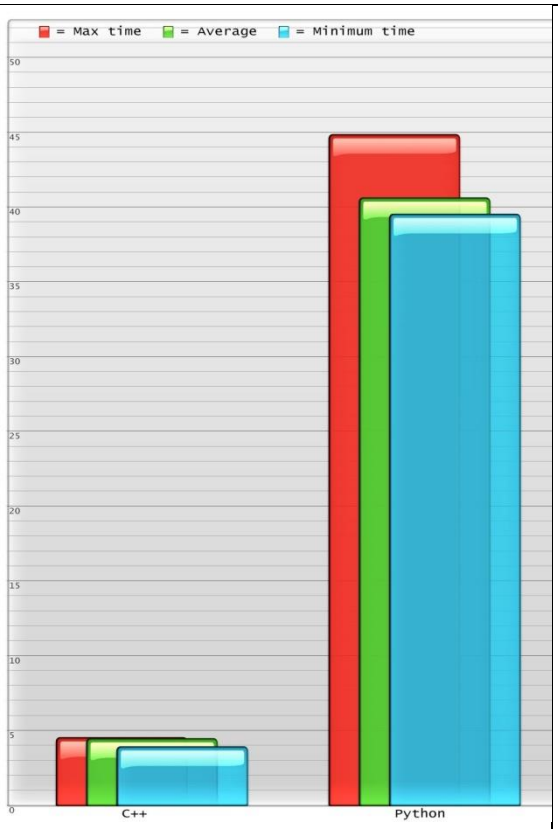


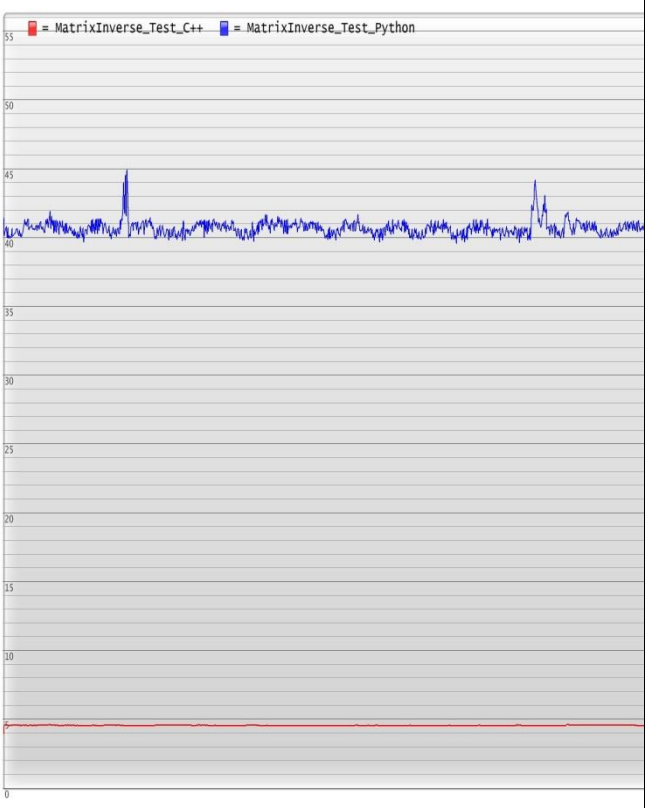


C++ Python	Jitter

Solishtirich natijalar – Yomon

Matris inversiyasi





C++	Python	Jitter

Natijalar

Python sinov natijalarini o’rtacha hisobda C ++ ga nisbatan o’rtacha 4x sekin harakat qildi.

Runtime jitter o’rtacha ijro muddatlariga nisbatan real vaqtda dasturlarga nisbatan muhimroqdir.

Matematik, xotira jadal yoki murakkab algoritmlar Pythonda eng katta ishlash ta’siriga ega.

Uslublar yoki tashqi modullarda yaratilgan Pythonlardan foydalanish C ++ ishlashidan ko’ra yaqinroq yoki yaxshiroq ishlab chiqarishi mumkin.

XULOSA

Shunday qilib, Python Guido van Rossum tomonidan yaratilgan dasturiy til bo’lib, birinchi navbatda dinamik terish bilan ob’ektiv yo’naltirilgan. Pythonning juda ko’p sonli ilovalari mavjud va til ko’pchilikka mashhur platformalarda joylashtirilgan. Tildagi urg’u soddalik, ravshanlik va rivojlanishni jadallashtirish, kodning o’qilishi mumkinligi haqida va bu juda ko’p afzalliklardan bir nechtasi. Asosiy kamchilik, ba’zi ilovalar uchun yomon ishlashi bo’lishi mumkin, ammo ayrim vazifalarni e’tiborsiz qoldirish mumkin.

O’qish qobiliyatiga ega bo’lgan ko’plab belgilar tildan olinib tashlanganligi tufayli, kod bloklari yoki nuqta-vergulni belgilash uchun parcha-parcha bo’lib, buyruqlar bajarilishini bildiradi. Buning o’rniga, bo’shliqlar bo’shliqlar yoki yorliqlar shaklida qo’llaniladi. Bundan tashqari Python dasturchilarining ko’pchiligiga rioya qilish qoidalari mavjud, bu qoidalar Python Zen va PEP da tasvirlangan.

Ushbu bitiruv malakaviy ishida python dasturlash tilini afzalliklari, qulayliklari haqida to’qtalib o’tdik. Shuningdek python dasturlash tilini C++ dasturlash tili bilan solishdirdik.

FOYDALANILGAN ADABIYOTLAR RO’YXATI

1. Karimov I. A. Barkamol avlod orzusi //Nashr uchun mas’ul T. Risqiyev.-T.: “Sharq” nashriyoti—matbaa konserni, 1999.-184 b.
2. “Kadrlar tayyorlash Milliy dasturi” // Barkamol avlod- O’zbekiston taraqqiyotining poydevori.-T.: “Sharq” nashriyoti—matbaa konserni, 1997. -
3. Дмитрий Мусин. Самоучитель Python. 2015 г
4. К.Ю. Поляков, В.М. Гуровиц. Язык Python в школьном курсе информатики - М.: Издательский дом МЭИ, 2011. - 424.
5. Г.Россум, Ф.Л.Дж.Дрейк, Д.С.Откидач. Язык программирования Python
6. К.Ю. Поляков, Е.А. Еремин. Информатика, 10 класс.
7. Марк Лутц. Программирование на Python. 1995г.
8. Девид Бизли. Python -Санкт-Петербург: МЭИ, 2008. - Часть III.
9. Сергей Лебедев. Модули и пакеты
10. Прохоренок Н.А. Python.Самое необходимое. - Санкт-Петербург: БХВ-Петербург, 2011, -416 с.

Internet manbalari manzili

I. www.python.org
   3. www.uhlib.ru
   4. ww.dasturchi.uz

Ilova

```
import random, sys, copy, os, pygame
from pygame.locals import *

FPS = 30 # frames per second to update the screen
WINWIDTH = 800 # width of the program's window, in pixels
WINHEIGHT = 600 # height in pixels
HALF_WINWIDTH = int(WINWIDTH / 2)
HALF_WINHEIGHT = int(WINHEIGHT / 2)

# The total width and height of each tile in pixels.
TILEWIDTH = 50
TILEHEIGHT = 85
TILEFLOORHEIGHT = 40

CAM_MOVE_SPEED = 5 # how many pixels per frame the camera moves

# The percentage of outdoor tiles that have additional
# decoration on them, such as a tree or rock.
OUTSIDE_DECORATION_PCT = 20

BRIGHTBLUE = (  0, 170, 255)
WHITE      = (255, 255, 255)
BGCOLOR = BRIGHTBLUE
TEXTCOLOR = WHITE

UP = 'up'
DOWN = 'down'
LEFT = 'left'
RIGHT = 'right'


def main():
    global FPSCLOCK, DISPLAYSURF, IMAGESDICT, TILEMAPPING,
OUTSIDEDECOMAPPING, BASICFONT, PLAYERIMAGES, currentImage

    # Pygame initialization and basic set up of the global variables.
    pygame.init()
    FPSCLOCK = pygame.time.Clock()

    # Because the Surface object stored in DISPLAYSURF was returned
    # from the pygame.display.set_mode() function, this is the
    # Surface object that is drawn to the actual computer screen
```

```
# when pygame.display.update() is called.
DISPLAYSURF = pygame.display.set_mode((WINWIDTH, WINHEIGHT))

pygame.display.set_caption('Juldizsha')
BASICFONT = pygame.font.Font('freesansbold.ttf', 18)

# A global dict value that will contain all the Pygame
# Surface objects returned by pygame.image.load().
IMAGESDICT = {'uncovered goal': pygame.image.load('RedSelector.png'),
        'covered goal': pygame.image.load('Selector.png'),
        'juldiz': pygame.image.load('juldiz.png'),
        'corner': pygame.image.load('Wall_Block_Tall.png'),
        'wall': pygame.image.load('Wood_Block_Tall.png'),
        'inside floor': pygame.image.load('Plain_Block.png'),
        'outside floor': pygame.image.load('Grass_Block.png'),
        'title': pygame.image.load('star_title.png'),
        'solved': pygame.image.load('star_solved.png'),
        'bala': pygame.image.load('bala.png'),
        'malika': pygame.image.load('malika.png'),
        'robot': pygame.image.load('robot.png'),
        'horngirl': pygame.image.load('horngirl.png'),
        'pinkgirl': pygame.image.load('pinkgirl.png'),
        'rock': pygame.image.load('Rock.png'),
        'short tree': pygame.image.load('Tree_Short.png'),
        'tall tree': pygame.image.load('Tree_Tall.png'),
        'ugly tree': pygame.image.load('Tree_Ugly.png')}

# These dict values are global, and map the character that appears
# in the level file to the Surface object it represents.
TILEMAPPING = {'x': IMAGESDICT['corner'],
         '#': IMAGESDICT['wall'],
         'o': IMAGESDICT['inside floor'],
         ' ': IMAGESDICT['outside floor']}
OUTSIDEDECOMAPPING = {'1': IMAGESDICT['rock'],
              '2': IMAGESDICT['short tree'],
              '3': IMAGESDICT['tall tree'],
              '4': IMAGESDICT['ugly tree']}

# PLAYERIMAGES is a list of all possible characters the player can be.
# currentImage is the index of the player's current player image.
currentImage = 0
PLAYERIMAGES = [IMAGESDICT['bala'],
         IMAGESDICT['malika'],
         IMAGESDICT['robot'],
         IMAGESDICT['horngirl'],
```

```
                IMAGESDICT['pinkgirl']]

    startScreen() # show the title screen until the user presses a key

    # Read in the levels from the text file. See the readLevelsFile() for
    # details on the format of this file and how to make your own levels.
    levels = readLevelsFile('juldizshabasqishi.txt')
    currentLevelIndex = 0

    # The main game loop. This loop runs a single level, when the user
    # finishes that level, the next/previous level is loaded.
    while True: # main game loop
        # Run the level to actually start playing the game:
        result = runLevel(levels, currentLevelIndex)

        if result in ('solved', 'next'):
            # Go to the next level.
            currentLevelIndex += 1
            if currentLevelIndex >= len(levels):
                # If there are no more levels, go back to the first one.
                currentLevelIndex = 0
        elif result == 'back':
            # Go to the previous level.
            currentLevelIndex -= 1
            if currentLevelIndex < 0:
                # If there are no previous levels, go to the last one.
                currentLevelIndex = len(levels)-1
        elif result == 'reset':
            pass # Do nothing. Loop re-calls runLevel() to reset the level


def runLevel(levels, levelNum):
    global currentImage
    levelObj = levels[levelNum]
    mapObj = decorateMap(levelObj['mapObj'], levelObj['startState']['player'])
    gameStateObj = copy.deepcopy(levelObj['startState'])
    mapNeedsRedraw = True # set to True to call drawMap()
    levelSurf = BASICFONT.render('Bosqish %s soni %s' % (levelNum + 1,
len(levels)), 1, TEXTCOLOR)
    levelRect = levelSurf.get_rect()
    levelRect.bottomleft = (20, WINHEIGHT - 35)
    mapWidth = len(mapObj) * TILEWIDTH
    mapHeight = (len(mapObj[0]) - 1) * TILEFLOORHEIGHT + TILEHEIGHT
    MAX_CAM_X_PAN = abs(HALF_WINHEIGHT - int(mapHeight / 2)) +
TILEWIDTH
```

```
    MAX_CAM_Y_PAN = abs(HALF_WINWIDTH - int(mapWidth / 2)) +
TILEHEIGHT

    levelIsComplete = False
    # Track how much the camera has moved:
    cameraOffsetX = 0
    cameraOffsetY = 0
    # Track if the keys to move the camera are being held down:
    cameraUp = False
    cameraDown = False
    cameraLeft = False
    cameraRight = False

    while True: # main game loop
        # Reset these variables:
        playerMoveTo = None
        keyPressed = False

        for event in pygame.event.get(): # event handling loop
            if event.type == QUIT:
                # Player clicked the "X" at the corner of the window.
                terminate()

            elif event.type == KEYDOWN:
                # Handle key presses
                keyPressed = True
                if event.key == K_LEFT:
                    playerMoveTo = LEFT
                elif event.key == K_RIGHT:
                    playerMoveTo = RIGHT
                elif event.key == K_UP:
                    playerMoveTo = UP
                elif event.key == K_DOWN:
                    playerMoveTo = DOWN

                # Set the camera move mode.
                elif event.key == K_a:
                    cameraLeft = True
                elif event.key == K_d:
                    cameraRight = True
                elif event.key == K_w:
                    cameraUp = True
                elif event.key == K_s:
                    cameraDown = True
```

```
            elif event.key == K_n:
                return 'next'
            elif event.key == K_b:
                return 'back'

            elif event.key == K_ESCAPE:
                terminate() # Esc key quits.
            elif event.key == K_BACKSPACE:
                return 'reset' # Reset the level.
            elif event.key == K_p:
                # Change the player image to the next one.
                currentImage += 1
                if currentImage >= len(PLAYERIMAGES):
                    # After the last player image, use the first one.
                    currentImage = 0
                mapNeedsRedraw = True

        elif event.type == KEYUP:
            # Unset the camera move mode.
            if event.key == K_a:
                cameraLeft = False
            elif event.key == K_d:
                cameraRight = False
            elif event.key == K_w:
                cameraUp = False
            elif event.key == K_s:
                cameraDown = False

    if playerMoveTo != None and not levelIsComplete:
        # If the player pushed a key to move, make the move
        # (if possible) and push any stars that are pushable.
        moved = makeMove(mapObj, gameStateObj, playerMoveTo)

        if moved:
            # increment the step counter.
            gameStateObj['stepCounter'] += 1
            mapNeedsRedraw = True

        if isLevelFinished(levelObj, gameStateObj):
            # level is solved, we should show the "Solved!" image.
            levelIsComplete = True
            keyPressed = False

    DISPLAYSURF.fill(BGCOLOR)
```

```
        if mapNeedsRedraw:
            mapSurf = drawMap(mapObj, gameStateObj, levelObj['goals'])
            mapNeedsRedraw = False

        if cameraUp and cameraOffsetY < MAX_CAM_X_PAN:
            cameraOffsetY += CAM_MOVE_SPEED
        elif cameraDown and cameraOffsetY > -MAX_CAM_X_PAN:
            cameraOffsetY -= CAM_MOVE_SPEED
        if cameraLeft and cameraOffsetX < MAX_CAM_Y_PAN:
            cameraOffsetX += CAM_MOVE_SPEED
        elif cameraRight and cameraOffsetX > -MAX_CAM_Y_PAN:
            cameraOffsetX -= CAM_MOVE_SPEED

        # Adjust mapSurf's Rect object based on the camera offset.
        mapSurfRect = mapSurf.get_rect()
        mapSurfRect.center = (HALF_WINWIDTH + cameraOffsetX,
HALF_WINHEIGHT + cameraOffsetY)

        # Draw mapSurf to the DISPLAYSURF Surface object.
        DISPLAYSURF.blit(mapSurf, mapSurfRect)

        DISPLAYSURF.blit(levelSurf, levelRect)
        stepSurf = BASICFONT.render('Yurishlar soni: %s' %
(gameStateObj['stepCounter']), 1, TEXTCOLOR)
        stepRect = stepSurf.get_rect()
        stepRect.bottomleft = (20, WINHEIGHT - 10)
        DISPLAYSURF.blit(stepSurf, stepRect)

        if levelIsComplete:
            # is solved, show the "Solved!" image until the player
            # has pressed a key.
            solvedRect = IMAGESDICT['solved'].get_rect()
            solvedRect.center = (HALF_WINWIDTH, HALF_WINHEIGHT)
            DISPLAYSURF.blit(IMAGESDICT['solved'], solvedRect)

            if keyPressed:
                return 'solved'

        pygame.display.update() # draw DISPLAYSURF to the screen.
        FPSCLOCK.tick()


def isWall(mapObj, x, y):
    """Returns True if the (x, y) position on
    the map is a wall, otherwise return False."""
```

```
    if x < 0 or x >= len(mapObj) or y < 0 or y >= len(mapObj[x]):
        return False # x and y aren't actually on the map.
    elif mapObj[x][y] in ('#', 'x'):
        return True # wall is blocking
    return False


def decorateMap(mapObj, startxy):
    """Makes a copy of the given map object and modifies it.
    Here is what is done to it:
        * Walls that are corners are turned into corner pieces.
        * The outside/inside floor tile distinction is made.
        * Tree/rock decorations are randomly added to the outside tiles.

    Returns the decorated map object."""

    startx, starty = startxy # Syntactic sugar

    # Copy the map object so we don't modify the original passed
    mapObjCopy = copy.deepcopy(mapObj)

    # Remove the non-wall characters from the map data
    for x in range(len(mapObjCopy)):
        for y in range(len(mapObjCopy[0])):
            if mapObjCopy[x][y] in ('$', '.', '@', '+', '*'):
                mapObjCopy[x][y] = ' '

    # Flood fill to determine inside/outside floor tiles.
    floodFill(mapObjCopy, startx, starty, ' ', 'o')

    # Convert the adjoined walls into corner tiles.
    for x in range(len(mapObjCopy)):
        for y in range(len(mapObjCopy[0])):

            if mapObjCopy[x][y] == '#':
                if (isWall(mapObjCopy, x, y-1) and isWall(mapObjCopy, x+1, y)) or \
                   (isWall(mapObjCopy, x+1, y) and isWall(mapObjCopy, x, y+1)) or \
                   (isWall(mapObjCopy, x, y+1) and isWall(mapObjCopy, x-1, y)) or \
                   (isWall(mapObjCopy, x-1, y) and isWall(mapObjCopy, x, y-1)):
                    mapObjCopy[x][y] = 'x'

            elif mapObjCopy[x][y] == ' ' and random.randint(0, 99) <
OUTSIDE_DECORATION_PCT:
                mapObjCopy[x][y] =
random.choice(list(OUTSIDEDECOMAPPING.keys()))
```

```
    return mapObjCopy


def isBlocked(mapObj, gameStateObj, x, y):
    """Returns True if the (x, y) position on the map is
    blocked by a wall or star, otherwise return False."""

    if isWall(mapObj, x, y):
        return True

    elif x < 0 or x >= len(mapObj) or y < 0 or y >= len(mapObj[x]):
        return True # x and y aren't actually on the map.

    elif (x, y) in gameStateObj['stars']:
        return True # a star is blocking

    return False


def makeMove(mapObj, gameStateObj, playerMoveTo):
    """Given a map and game state object, see if it is possible for the
    player to make the given move. If it is, then change the player's
    position (and the position of any pushed star). If not, do nothing.

    Returns True if the player moved, otherwise False."""

    # Make sure the player can move in the direction they want.
    playerx, playery = gameStateObj['player']

    # This variable is "syntactic sugar". Typing "stars" is more
    # readable than typing "gameStateObj['stars']" in our code.
    stars = gameStateObj['stars']

    # The code for handling each of the directions is so similar aside
    # from adding or subtracting 1 to the x/y coordinates. We can
    # simplify it by using the xOffset and yOffset variables.
    if playerMoveTo == UP:
        xOffset = 0
        yOffset = -1
    elif playerMoveTo == RIGHT:
        xOffset = 1
        yOffset = 0
    elif playerMoveTo == DOWN:
        xOffset = 0
```

```
        yOffset = 1
    elif playerMoveTo == LEFT:
        xOffset = -1
        yOffset = 0

    # See if the player can move in that direction.
    if isWall(mapObj, playerx + xOffset, playery + yOffset):
        return False
    else:
        if (playerx + xOffset, playery + yOffset) in stars:
            # There is a star in the way, see if the player can push it.
            if not isBlocked(mapObj, gameStateObj, playerx + (xOffset*2), playery +
(yOffset*2)):
                # Move the star.
                ind = stars.index((playerx + xOffset, playery + yOffset))
                stars[ind] = (stars[ind][0] + xOffset, stars[ind][1] + yOffset)
            else:
                return False
        # Move the player upwards.
        gameStateObj['player'] = (playerx + xOffset, playery + yOffset)
        return True


def startScreen():
    """Display the start screen (which has the title and instructions)
    until the player presses a key. Returns None."""

    # Position the title image.
    titleRect = IMAGESDICT['title'].get_rect()
    topCoord = 50 # topCoord tracks where to position the top of the text
    titleRect.top = topCoord
    titleRect.centerx = HALF_WINWIDTH
    topCoord += titleRect.height

    # Unfortunately, Pygame's font & text system only shows one line at
    # a time, so we can't use strings with \n newline characters in them.
    # So we will use a list with each line in it.
    instructionText = ['Bosqarish belgilari bilan boshlang .',
                       'Bosqarish belgilari , WASD kamera kuzatuvi , P odamni
o`zgartirish.',
                       'Backspace qayta boslash ushun, Esc oyinnan chiqish .',
                       'N keyingi bosqishni boslash, B ortga qaytish .']

    # Start with drawing a blank color to the entire window:
    DISPLAYSURF.fill(BGCOLOR)
```

```
    # Draw the title image to the window:
    DISPLAYSURF.blit(IMAGESDICT['title'], titleRect)

    # Position and draw the text.
    for i in range(len(instructionText)):
        instSurf = BASICFONT.render(instructionText[i], 1, TEXTCOLOR)
        instRect = instSurf.get_rect()
        topCoord += 10 # 10 pixels will go in between each line of text.
        instRect.top = topCoord
        instRect.centerx = HALF_WINWIDTH
        topCoord += instRect.height # Adjust for the height of the line.
        DISPLAYSURF.blit(instSurf, instRect)

    while True: # Main loop for the start screen.
        for event in pygame.event.get():
            if event.type == QUIT:
                terminate()
            elif event.type == KEYDOWN:
                if event.key == K_ESCAPE:
                    terminate()
                return # user has pressed a key, so return.

        # Display the DISPLAYSURF contents to the actual screen.
        pygame.display.update()
        FPSCLOCK.tick()


def readLevelsFile(filename):
    assert os.path.exists(filename), 'Cannot find the level file: %s' % (filename)
    mapFile = open(filename, 'r')
    # Each level must end with a blank line
    content = mapFile.readlines() + ['\r\n']
    mapFile.close()

    levels = [] # Will contain a list of level objects.
    levelNum = 0
    mapTextLines = [] # contains the lines for a single level's map.
    mapObj = [] # the map object made from the data in mapTextLines
    for lineNum in range(len(content)):
        # Process each line that was in the level file.
        line = content[lineNum].rstrip('\r\n')

        if ';' in line:
            # Ignore the ; lines, they're comments in the level file.
```

```
        line = line[:line.find(';')]

    if line != '':
        # This line is part of the map.
        mapTextLines.append(line)
    elif line == '' and len(mapTextLines) > 0:
        # A blank line indicates the end of a level's map in the file.
        # Convert the text in mapTextLines into a level object.

        # Find the longest row in the map.
        maxWidth = -1
        for i in range(len(mapTextLines)):
            if len(mapTextLines[i]) > maxWidth:
                maxWidth = len(mapTextLines[i])
        # Add spaces to the ends of the shorter rows. This
        # ensures the map will be rectangular.
        for i in range(len(mapTextLines)):
            mapTextLines[i] += ' ' * (maxWidth - len(mapTextLines[i]))

        # Convert mapTextLines to a map object.
        for x in range(len(mapTextLines[0])):
            mapObj.append([])
        for y in range(len(mapTextLines)):
            for x in range(maxWidth):
                mapObj[x].append(mapTextLines[y][x])

        # Loop through the spaces in the map and find the @, ., and $
        # characters for the starting game state.
        startx = None # The x and y for the player's starting position
        starty = None
        goals = [] # list of (x, y) tuples for each goal.
        stars = [] # list of (x, y) for each star's starting position.
        for x in range(maxWidth):
            for y in range(len(mapObj[x])):
                if mapObj[x][y] in ('@', '+'):
                    # '@' is player, '+' is player & goal
                    startx = x
                    starty = y
                if mapObj[x][y] in ('.', '+', '*'):
                    # '.' is goal, '*' is star & goal
                    goals.append((x, y))
                if mapObj[x][y] in ('$', '*'):
                    # '$' is star
                    stars.append((x, y))
```

```
            # Basic level design sanity checks:
            assert startx != None and starty != None, 'Level %s (around line %s) in %s
is missing a "@" or "+" to mark the start point.' % (levelNum+1, lineNum,
filename)
            assert len(goals) > 0, 'Level %s (around line %s) in %s must have at least
one goal.' % (levelNum+1, lineNum, filename)
            assert len(stars) >= len(goals), 'Level %s (around line %s) in %s is
impossible to solve. It has %s goals but only %s stars.' % (levelNum+1, lineNum,
filename, len(goals), len(stars))

            # Create level object and starting game state object.
            gameStateObj = {'player': (startx, starty),
                        'stepCounter': 0,
                        'stars': stars}
            levelObj = {'width': maxWidth,
                    'height': len(mapObj),
                    'mapObj': mapObj,
                    'goals': goals,
                    'startState': gameStateObj}

            levels.append(levelObj)

            # Reset the variables for reading the next map.
            mapTextLines = []
            mapObj = []
            gameStateObj = {}
            levelNum += 1
    return levels


def floodFill(mapObj, x, y, oldCharacter, newCharacter):
    """Changes any values matching oldCharacter on the map object to
    newCharacter at the (x, y) position, and does the same for the
    positions to the left, right, down, and up of (x, y), recursively."""

    # In this game, the flood fill algorithm creates the inside/outside
    # floor distinction. This is a "recursive" function.
    # For more info on the Flood Fill algorithm, see:
    #   http://en.wikipedia.org/wiki/Flood_fill
    if mapObj[x][y] == oldCharacter:
        mapObj[x][y] = newCharacter

    if x < len(mapObj) - 1 and mapObj[x+1][y] == oldCharacter:
        floodFill(mapObj, x+1, y, oldCharacter, newCharacter) # call right
    if x > 0 and mapObj[x-1][y] == oldCharacter:
```

```
        floodFill(mapObj, x-1, y, oldCharacter, newCharacter) # call left
    if y < len(mapObj[x]) - 1 and mapObj[x][y+1] == oldCharacter:
        floodFill(mapObj, x, y+1, oldCharacter, newCharacter) # call down
    if y > 0 and mapObj[x][y-1] == oldCharacter:
        floodFill(mapObj, x, y-1, oldCharacter, newCharacter) # call up


def drawMap(mapObj, gameStateObj, goals):
    """Draws the map to a Surface object, including the player and
    stars. This function does not call pygame.display.update(), nor
    does it draw the "Level" and "Steps" text in the corner."""

    # mapSurf will be the single Surface object that the tiles are drawn
    # on, so that it is easy to position the entire map on the DISPLAYSURF
    # Surface object. First, the width and height must be calculated.
    mapSurfWidth = len(mapObj) * TILEWIDTH
    mapSurfHeight = (len(mapObj[0]) - 1) * TILEFLOORHEIGHT +
TILEHEIGHT
    mapSurf = pygame.Surface((mapSurfWidth, mapSurfHeight))
    mapSurf.fill(BGCOLOR) # start with a blank color on the surface.

    # Draw the tile sprites onto this surface.
    for x in range(len(mapObj)):
        for y in range(len(mapObj[x])):
            spaceRect = pygame.Rect((x * TILEWIDTH, y * TILEFLOORHEIGHT,
TILEWIDTH, TILEHEIGHT))
            if mapObj[x][y] in TILEMAPPING:
                baseTile = TILEMAPPING[mapObj[x][y]]
            elif mapObj[x][y] in OUTSIDEDECOMAPPING:
                baseTile = TILEMAPPING[‘ ‘]

            # First draw the base ground/wall tile.
            mapSurf.blit(baseTile, spaceRect)

            if mapObj[x][y] in OUTSIDEDECOMAPPING:
                # Draw any tree/rock decorations that are on this tile.
                mapSurf.blit(OUTSIDEDECOMAPPING[mapObj[x][y]], spaceRect)
            elif (x, y) in gameStateObj[‘stars’]:
                if (x, y) in goals:
                    # A goal AND star are on this space, draw goal first.
                    mapSurf.blit(IMAGESDICT[‘covered goal’], spaceRect)
                # Then draw the star sprite.
                mapSurf.blit(IMAGESDICT[‘juldiz’], spaceRect)
            elif (x, y) in goals:
                # Draw a goal without a star on it.
```

```
                mapSurf.blit(IMAGESDICT[‘uncovered goal’], spaceRect)

            # Last draw the player on the board.
            if (x, y) == gameStateObj[‘player’]:
                # Note: The value "currentImage" refers
                # to a key in "PLAYERIMAGES" which has the
                # specific player image we want to show.
                mapSurf.blit(PLAYERIMAGES[currentImage], spaceRect)

    return mapSurf


def isLevelFinished(levelObj, gameStateObj):
    """Returns True if all the goals have stars in them."""
    for goal in levelObj[‘goals’]:
        if goal not in gameStateObj[‘stars’]:
            # Found a space with a goal but no star on it.
            return False
    return True


def terminate():
    pygame.quit()
    sys.exit()


if __name__ == ‘__main__’:
    main()
```