O‘ZBEKISTON RESPUBLIKASI MAKTABGACHA VA MAKTAB TA'LIMI VAZIRLIGI

SAMARQAND VILOYATI PEDAGOGLARNI YANGI METODIKALARGA O‘RGATISH MILLIY MARKAZI

Algoritmlash va dasturlash, python dasturlash tili

(umumiy o‘rta ta’lim maktabi informatika fani o‘qituvchilari uchun uslubiy ko‘rsatma)

Samarqand – 2023

O‘ZBEKISTON RESPUBLIKASI MAKTABGACHA VA MAKTAB TA'LIMI VAZIRLIGI

SAMARQAND VILOYATI PEDAGOGLARNI YANGI METODIKALARGA O‘RGATISH MILLIY MARKAZI

Algoritmlash va dasturlash, python dasturlash tili

(umumiy o‘rta ta’lim maktabi informatika fani o‘qituvchilari uchun uslubiy ko‘rsatma)

Samarqand – 2023

Sh. Abduyev. Algoritmlash va dasturlash, python dasturlash tili *(umumiy o‘rta ta’lim maktabi informatika fani o‘qituvchilari uchun uslubiy ko‘rsatma)*. - Samarqand, 2023. 24 bet.

Mas’ul muharrir: **M.Fayziyeva** - ITXHIAT bo‘limi boshlig'i

Taqrizchilar: **I.Aminov -** SamDU Axborotlastirish texnologiyalari kafedrasi dotsenti

D.Qarshiyeva- SamVPYMO‘MM Aniq va tabiiy fanlarni o‘qitish metodikasi kafedrasi katta o‘qituvchisi

SamVPYMO‘MM Ilmiy kengashining 2023-yil 23 iyunda bo‘lib o‘tgan yig’ilishida muhokama etilgan va 3-sonli qarori bilan nashrga tavsiya etilgan

Kirish

Mamlakatimizda mustaqillik yillarida amalga oshirilgan keng ko‘lamli islohotlar milliy davlatchilik va suverenitetni mustahkamlash, xavfsizlik va huquq-tartibotni, jamiyatda qonun ustuvorligini, inson huquq va erkinliklarini, millatlararo totuvlik va diniy bag‘rikenglik muhitini ta'minlash uchun muhim poydevor bo‘ldi, xalqimizning munosib hayot kechirishi, jahon talablari darajasida ta'lim olishi va kasb egallashi, fuqarolarimizning bunyodkorlik salohiyatini ro‘yobga chiqarish uchun zarur shart-sharoitlar yaratdi. Yangi sharoitlardan kelib chiqib, «Ta'lim to‘g‘risida»gi va «Kadrlar tayyorlash milliy dasturi to‘g‘risida»gi O‘zbekiston Respublikasi qonunlariga, 2017-2021 - yillarga mo‘ljallangan “O‘zbekiston Respublikasini yanada rivojlantirish bo‘yicha Harakatlar strategiyasi”, O‘zbekiston Respublikasi Prezidentining 2018 yil 5 sentyabrdagi “Xalq ta'limi tizimiga boshqaruvning yangi tamoyillarini joriy etish chora-tadbirlari to‘g‘risida” PQ-3931, shuningdek Qaroriga muvofiq, ta'lim bosqichlarining uzluksizligi va izchilligini ta'minlash, ta'limning zamonaviy metodologiyasini yaratish, davlat ta'lim standartlarini kompetensiyaviy yondashuv asosida takomillashtirish, o‘quv-metodik majmualarning yangi avlodini ishlab chiqish va amaliyotga joriy etish hamda pedagog xodimlarini qayta tayyorlash va ularning malakasini oshirish tizimini yanada takomillashtirish taqozo etadi. “ Dasturlash asoslari. Vizualizatsiya va Python dasturlash tillarini o’rganish usullari” modulining ishchi o‘quv dasturi Informatika fani o‘qituvchilari malakasini oshirish kursining o‘quv dasturi asosida tuzilgan bo‘lib, ush bu ko‘rsatmada informatika fani o‘qituvchilari Python dasturlash tili tarixi dasturni o‘rnatish va operatorlari bilan ishlash ko‘nikma va malakalarini ega bo‘lishi mumkin.

Algoritmlar.

Qo‘yilgan masalani u yoki bu turini yechishning algoritmlarini shakllantirish va ishlab chiqish eng ma‘suliyatli hamda muhim bosqichlardan hisoblanadi, chunki bu bosqichda keyinchalik shaxsiy kompyuterda bajarilishi kerak bo‘lgan amallarning ketma-ketligi oldindan belgilab olinadi. Algoritmda yo‘lga qo‘yilgan xatoliklar hisoblash jarayonini noto‘g‘ri bajarilishiga olib keladi, ya‘ni noto‘g‘ri natijalarni beradi.

Algoritm tushunchasi. A l g o r i t m – bu masalani yechish usullarini izohlashdir, yoki boshqacha qilib aytganda, kutilayotgan natijalarni shaxsiy kompyuter tomonidan olish uchun bajarilayotgan hisoblash jarayolarining ketma-ketliklaridir. Algoritm - bu biror masalani yechish uchun bajarilishi zarur bo‘lgan buyruqlarning tartiblargan ketma-ketligidir. Har bir algoritm aniq va tugallangan qadamlarga bo‘lingan bo‘lishi kerak. «Axborot-kommunikasiya texnologiyalari» izohli lo‘g‘atida quyidagi ta’riflar keltirilgan:

1. Vazifani bajarishga qaratilgan aniq belgilangan qoidalarning tartiblangan chekli to‘plami.

2. Dastlabki ma‘lumotlarni oxirgi natijaga o‘tkazuvchi xisoblash jarayoni orqali masala yechimini aniq ko‘rsatuvchi amallar mazmuni va ketma-ketligi. Algoritm deb, masalani echish uchun bajarilishi lozim bo‘lgan amallar ketma-ketligini aniq tavsiflaydigan qoidalar tizimiga aytiladi.

Boshqacha aytganda, algoritm –boshlang‘ich va oraliq malumotlarni masalani echish natijasiga aylantiradigan jarayonni bir qiymatli qilib, aniqlab beradigan qoidalarning biror bir chekli ketma-ketligidir. Buning mohiyati shundan iboratki, agar algoritm ishlab chiqilgan bo‘lsa, uni echilayotgan masala bilan tanish bo‘lmagan biron bir ijrochiga, shu jumladan kompyuterga xam bajarish uchun topshirsa bo‘ladi va u algoritmning qoidalariga aniq rioya qilib masalani echadi. Algoritm atamasi o‘rta asrlarda yashab ijod etgan buyuk o‘zbek matematigi Al-Xorazmiy nomidan kelib chiqqan. Algoritm so‘zi al-Xorazmiyning arifmetikaga bag‘ishlangan asarining dastlabki betidagi “Dixit Algoritmi” (“dediki al-

Xorazmiy” ning lotincha ifodasi) degan jumlalardan kelib chiqqan. U o‘zi kashf etgan o‘nli sanoq tizimida IX asrning 825 yilidayoq to‘rt arifmetika amallarini bajarish qoidalarini bergan. Arifmetika amallarini bajarish jarayoni esa alxorazm deb atalgan. Bu atama 1747 yildan boshlab algorismus, 1950 yilga kelib algoritm deb ham ataldi. Bu yerda al-Xorazmiyning sanoq sistemasini takomillashtirishga qo‘shgan hissasi, uning asarlari algoritm tushunchasining kiritilishiga sabab bo‘lganligi o‘quvchilarga ta‘kidlab o‘tiladi. Kompyuterlar paydo bo‘lishi bilan algoritm atamasining hozirgi ma‘nosi bilan axborot-kommunikasiyalar texnologiyalari sohasida eng asosiy atamalardan biri bo‘lib qoldi. Dars jarayonida o‘quvchilarga algoritm nima degan savolga, u asosiy tushuncha sifatida qabul qilinganligidan, uning faqat tavsifi beriladi, ya‘ni biror maqsadga erishishga yoki qandaydir masalani yechishga qaratilgan ko‘rsatmalarning (buyruqlarning) aniq, tushunarli, chekli hamda to‘liq tizimi ekanligi tushuntiriladi. Hisoblash mashinasining ishi algoritmlarni bajarishdan iborat bo‘ladi. SHuning uchun hisoblash mashinalarining umumiy imkoniyatlari kaysi muammomasalalarni algoritm sifatida tasvirlash mumkinu, qaysilarini mumkin emasligiga bog‘lik bo‘ladi.Matematikaning eng asosiy tushunchalarnidan biri bo‘lgan algoritm tushunchasi hisoblash masalalari paydo bo‘lganidan ancha oldin vujudga kela boshlagan edi. Asrlar davomida kishilar algoritm tushunchalaridan foydalanib kelgandlar. Bu tushunchani shunday ta‘riflash mumkin: Algoritm – bu qoidalarning qat‘iy va chekli sistemasi bo‘lib, ba‘zi ob‘ektlar ustida bajariladigan amallarni aniqlaydi va chekli kadamdan keyin kuyilgan maksadga olib kelishni ta‘minlaydi. Xususiy holda bunday qoidalar sistemasi algoritm hisoblanadi, qachonki, ishning mazmuni bilan tanish bo‘lmagan kishilarga uni ko‘rsatma sifatida berilganda , ularning barchasi bir xil harakat qilsa. Qadimgi Gresiyalik matematik Evklid 2 ta natural A va B sonlarning eng katta umumiy bo‘luvchisini topish algoritmini taklif etdi. Uning ma‘nosi quyidagicha: Katta sondan kichigini ayirish, natijani katta son o‘rniga qo‘yish va ikkala son tenglashguncha bu amalni takrorlash. Ushbu teng sonlar izlangan natijadir. Evklid algoritmida A va B sonlarning eng katta umumiy bo‘luvchisi ushbu sonlar ayirmasining eng katta

bo‘luvchisi hamda ikkala A,B sonlarning ham umumiy eng katta bo‘luvchisi bo‘lishligidan foydalanilgan. Evklid algoritmining bu ifodasiga aniqlik etishmaydi, shuning uchun uning konkretlashtirish zarur bo‘ladi.

Haqiqiy Evklid algoritmi quyidagicha:

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

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

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

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

5. Birinchi sonni natija sifatida qabul qilinsin.

Tamom. Bu qoidalar ketma-ketligi algoritmning tashkil etadi, chunki ularni bajargan ixtiyoriy ayirishni biladigan kishi ixtiyoriy sonlar jufti uchun eng katta umumiy bo‘luvchini topa oladi. Matematiklar uzoq vaqtlar davomida algoritmlarning bunday ifodalaridan keng foydalanib turli hisoblash algoritmlarini ishlab chiqdilar. Masalan, kvadrat va kubik tenglamalar ildizlarini topish algoritmlari topildi. Asta-sekin olimlar qiyinroq masalalar ustida bosh qotirib, masalan, ixtiyoriy darajali algebraik tenglamalar ildizlarini topish algoritmlarini qidiradilar. Hatto, XVII –asrda Leybnis ixtiyoriy matematik masalani echishning umumiy algoritmini topishga urinib ko‘rgan. Ammo bunga o‘xshash algoritmlarni kuo‘rishning iloji bo‘lmagan va asta-sekin buning butunlay imkoni yo‘q degan xulosaga kelingan. Shunday bo‘lishiga qaramay, algoritm tushunchasining aniq tavsifi berilmagunga qadar, masalaning algoritmik echimsizligini isbotlash mumkin emas edi. Algoritm tushunchasi juda qadim zamonlardan shakllanib kelgan. Shunga qaramay, asrimizning yarmiga qadar matematiklar bu ob‘ekt hakida ma‘lum bir qarashlarga qanoatlanib kelganlar. Algoritm atamasi matematiklar tomonidan faqat konkret masalalarni echish bilan boglik xolda olinar edi. XX asr boshida matematika asoslarida vujudga kelgan qarama-qarshiliklar va

muammolar ularni hal etishga qaratilgan turli konsepsiyalar va oqimlarning vujudga kelishiga olib keldi. 20- yillarga kelib, effektiv hisoblash masalalari koʻndalang boʻldi. Algoritm tushunchasining oʻzi matematik tadqiqotlar obʻekti boʻlib qolganligi uchun aniq va qatʻiy taʻrifga muhtoj edi. Bundan tashqari komputer asrini yaqinlashtiruvchi fizika va texnikaning rivojlanishi ham shuni taqozo etar edi. XX asr boshlarida matematiklar baʻzi ommaviy masalalar algoritmik echimga ega emas degan xulosaga kela boshladilar. Biror bir obʻektning mavjud emasligi ni qatʻiy isbotlash uchun esa, ushbu obʻektning aniq taʻrifiga ega boʻlish kerak edi. Uzluksizlik, egri chiziq, sirt, uzunlik, yuza, hajm va boshqa shu kabi tushunchalarni aniqlashtirish zarurati tugʻilganda xuddi ana shunday holat vujudga kelgan edi. Algoritm tushunchasini aniqlashtirish va oʻrganish, yaʻni algoritmlar nazariyasi boʻyicha eng birinchi tadqiqotlar 1936-37 yillarda A.Tyuring, E.Post, E.Erbran, K. Gedel, A.Markov, A.Cherchlar tomonidan bajarildi. Algoritm tushunchasi boʻyicha bir qancha taʻriflar ishlab chiqildi. Ammo keyinchalik ularning teng kuchliligi aniqlandi. Hayotimizda algoritmlarni turli sohalarda baʻzan bilgan holda baʻzan esa bilmagan holda ishlatamiz. Algoritmlar nafaqat matematik xarakterga ega boʻlmasdan ularni oddiy hayotiy turmushimizda ham koʻp qoʻllaymiz. Masalan, ovqat tayyorlash, choy damlash, biror berilgan ishni bajarish va boshqa. Bu ishlarni bajarishda maʻlum boʻlgan aniq koʻrsatmalarni ketma ket bajaramiz. Agar bu koʻrsatmalar aniq bir ketma ketlik tartibida bajarilmasa kerakli natijani olaolmaymiz.. Algoritmni mukammalloq tushunib olish uchun oʻquvchilarga turli hayotdan, fandan algoritmlarga misollar keltirish va bunga oʻzlari tuzishga harakat qilishlarini taklif etish mumkin. Masalan, taom tayyorlash, turli qurilmalarni ishlatish, sport musobaqasi yoki yoʻl harakati qoidalari algoritmlarini keltirish mumkin, yoki matematik formula boʻyicha qiymat hisoblash algoritmi yoki kompyuterni ishlatish boʻyicha algoritm kabi misollar keltirilishi mumkin

Python dasturlash tili yaratilishi tarixi

Python dasturlash tili 1980-yillar oxirida 1990-yillar boshlarida yaratilgangan boʻlib . Ushbu dasturlash tilini Gollandiyaning CWI instituti xodimi

Gvido van Rossum ABC tilini yaratilish loyihasida ishtirok etgan va ABC tili Basic dasturlash tili o‘quvchilarga asosiy dasturlash konsepsiyalarini o‘rgatish uchun mo‘ljallangan til edi. Gvido bu ishlardan charchadi va 2 hafta ichida o‘zining Macintoshida boshqa soddaroq tilning interpretatorini yaratdi, bunda u ABC tilining ba’zi 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. Ushbuu dasturlash tiliga qiziqqan va tushunadigan foydalanuvchilar soni ko‘paydi. Bu juda sodda til edi. Shunchaki kichik interpretator bir nechta funksiyalarga ega edi. 1991-yil birinchi ob’yektga yo‘naltirilgan dasturlash vositalari paydo bo‘la boshladi. Bir qancha vaqt o‘tmay Gvido Gollandiyadan Amerikaga ko‘chib o‘tdi. Uni CNRI korparatsiyasiga ishlashga taklif etishdi. U o‘sha yerda tashkilotda shug‘ullanayotgan loyihalarni 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 tashkilotning loyihalarini 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 1994-

yilda 1.0 versiyani, 2000-yil 2.0 versiyasini, 2008-yilda esa 3.0 versiyasini ishlab chiqarildi. Hozirgi vaqtda esa uchinchi versiyasi keng qo‘llanilib kelmoqda.

Python dasturlash tili imkoniyatlari

Python – ushbu dasturni o‘rganish oson va shu bilan birga imkoniyatlari yuqori bo‘lgan va zamonaviy dasturlash tillarida biri hisoblanadi Python yuqori darajadagi ma'lumotlar strukturasi va oddiy lekin samarador ob’yektga yo‘naltirilgan dasturlash tili hisoblanadi.

Pythonning avzalliklari

• O‘rganish oson, sodda sintaksis, havaskor dasturchilar uchun dasturlashni boshlash uchun qulay, erkin va ochiq kodli dasturiy ta'minot.

• Dasturni yozish davomida quyi darajadagi detallarni, misol uchun xotirani boshqarishni hisobga olish shart emas.

• Ko‘plab platformalarda hech qanday o‘zgartirishlarsiz ishlay oladi.

• Interpretatsiya(Интерпретируемый) qilinadigan til.

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

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

• xml/html fayllar bilan ishlash

• http so‘rovlari bilan ishlash

• GUI(grafik interfeys)

• Web ssenariy tuzish

• FTP bilan ishlash

• Rasmli audio video fayllar bilan ishlash

• Robot texnikada

• Matematik va ilmiy hisoblashlarni programmalash

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

Python tili sintaksisi, asosiy operatorlari.

Python tili sintaksisi o‘zi kabi sodda

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

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

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

Bir nechta maxsus holatlar

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

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

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

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

```
if (a == 1 and b == 2 and
c == 3 and d == 4):
print('spam'*3)
```

Kalit so‘zlar

False – yolg‘on.

True - rost.

None - “bo‘sh” obyekt.

and – mantiqiy VA amali.

with / as – konteks menejeri.

break –tsikldan chiqish.

class – metod va atributlarda iborat.

continue – tsikldan keyingi iteratsiyaga o‘tish.

def – funksiyani aniqlash.

del – obyektni yo‘qotish.

elif – aks holda, agar.

else – for/else yoki if/elsega qarang.

for – for tsikli.

from – moduldan bir nechta funksiyani import qilish.

if - agar.

import – moduldan import.

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

lambda –yashirin funksiyani aniqlash.

not –mantiqiy inkor amali.

or –mantiqiy Yoki amali.

while – while tsikli.

Komentariy

Kommentariy. Kommentariy # simvolidan keyin yoziladi va dastur kodini o‘qiyotgan dasturchi uchun eslatma bo‘lib xizmat qiladi. Misol uchun:

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

yoki:

 # print — bu funksiya

print('salom dunyo! ')

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

O‘zgaruvchilar

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

olamiz. Konstantaga esa oldindan ma'lum bir qiymat beriladi va bu qiymatni o‘zgartirib bo‘lmaydi.

O‘zgaruvchilarni nomlashda quyidagi qoidalarga amal qilish kerak:

• O‘zgaruvchining birinchi belgisi alifbo harfi (ASCII simvollari katta va kichik registrda) yoki “_” (ostki chiziq) simvoli bo‘lishi mumkin.

• O‘zgaruvchilarning qolgan qismi harflardan (ASCII simvollari katta va kichik registrda), “_" (ostki chiziq) simvoli va raqamlardan(0-9) tashkil topishi mumkin.

• O‘zgaruvchilar nomlashda katta va kichik registrlar farqlanadi. Masalan, myname va myName – bular boshqa-boshqa o‘zgaruvchi hisoblanadi.

• O‘zgaruvchilarni to‘g'ri nomlashga misollar: i, _my_name, name_23, a1b2_c3

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

O‘zgaruvchi va konstantalarni qo‘llanishiga misol:

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

Natija:

```
5
6
Bu ko‘p qatorlik satr.
Bu uning ikkinchi qatori.
```

Yuqoridagi misolda dastlab biz 5 konstanta qiymatini '=' operatori yordamida i o‘zgaruvchiga o‘zlashtirib olamiz.

```
i = 5
```

so‘ng i o‘zgaruvchi qiymatini print funksiyasi orqali ekranga chop etamiz.

print(i)

i o‘zgaruvchining qiymatiga 1 qo‘shamiz va o‘zgaruvchining o‘ziga saqlaymiz. So‘ng i o‘zgaruvchining qiymatini chop etamiz.

i = i + 1

print(i)

Yuqoridagi kabi satr konstanta qiymatini s o‘zgaruvchiga biriktiramiz va shundan so‘ng uni chop etamiz.

s = "'Bu ko‘p qatorlik satr.

Bu uning ikkinchi qatori.'"

print(s)

Eslatma: O‘zgaruvchilar oddiy qiymat biriktirish bilan ishlatiladi. Hech qanday oldindan e'lon qilib qo‘yish talab etilmaydi.

Operatorlar va ifodalar

Dasturdagi ko‘p amallar (mantiqiy qatorlar) ifodalardan tashkil topgan. Bunga oddiy misol: 2 + 3. Ifodani operatorlar va operandlarga ajratish mumkin. **Operator** – bu biror amalni bajaruvchi va simvol yordamida yoki zaxiraga olingan so‘zlar yordamida ifodalanadigan funksional operatorlar qiymatlar ustida biror amalni bajaradi va bu qiymatlar operatorlar deyiladi. Bizning xolatda 2 va 3 – bu operatorlar.

Operator	Nomlanishi	Ta'rifi	Misol
+	Qo‘shish	Ikkita ob'yektning yig'indisini hisoblaydi	3 + 5 ifoda 8 ni beradi; 'a' + 'b' ifoda 'ab' ni beradi.
–	Ayirish	Ikkata sonning farqini beradi. Agar birinchi operand mavjud bo‘lmasa, uning qiymati 0 ga teng deb olib ketiladi.	-5.2 manfiy qiymat beradi, 50 – 24 ning qiymati esa 26 ga teng.
*	Ko‘paytirish	Ikkita son ko‘paytmasini	2 * 3 ifoda 6 beradi. 'xa' * 3 ifoda 'xaxaxa'

		beradi yoki satrni ko‘rsatilgan miqdorda takrorlangan yangi satrni qaytaradi.	ni qaytaradi.
**	Darajaga ko‘tarish	x sonini y darajaga ko‘tarilganda hosil bo‘lgan qiymatni qaytaradi.	3**4 ifoda 81 ni qaytaradi (ya'ni 3*3*3*3) hosil bo‘lgan qiymatni qaytaradi.
/	Bo‘lish	'x' ni 'y' ga bo‘lganda hosil bo‘lgan bo‘linmani qaytaradi.	4 / 3 ifoda 1.3333333333333333 ni beradi.
//	Qoldiqsiz bo‘lish	Bo‘lishdan hosil bo‘lgan bo‘linmaning qoldiqsiz butun qismini qaytaradi.	4 // 3 ifoda 1 ni qaytaradi.
%	Qoldiqlik bo‘lish	Bo‘lishdan hosil bo‘lgan qoldiqni qaytaradi.	8 % 3 ifoda 2 ni beradi. -25.5 % 2.25 ifoda 1.5 ni beradi.
<<	Chapga surish	Bit sonni chapga ko‘rsatilgan miqdorda suradi.	2 << 2 ifoda 8 ni beradi. Ikkilik sanoq tizimida 2 soni 10 ko‘rinishiga ega bo‘ladi. Chapga 2 bit miqdorida surish 1000 beradi, bu o‘nlik sanoq tizimida 8 ni beradi.
>>	O‘ngga surish	Bit sonni o‘ngga ko‘rsatilgan miqdorda suradi.	11 >> 1 ifoda 5 ni beradi. Ikkilik sanoq sistemasida 11 soni 1011 ko‘rinishida bo‘ladi uni 1 bit o‘ngga sursak 101 hosil bo‘ladi va bu onlik sanoq tizimida

			ni beradi.
&	'Va' bit operatori	(Побитовое И) Sonlar ustida 'va' bit operatsiyasini bajaradi.	5 & 3 ifoda 1 ni beradi
|	'Yoki' bit operatori	(Побитовое ИЛИ) Sonlar ustida 'yoki' bit operatsiyasini bajaradi.	5 | 3 ifoda 7 ni beradi
^	'shartlik yoki' bit operatori (Побитовое ИСКЛЮЧИТЕЛЬНО ИЛИ)	Sonlar ustida 'shartlik yoki' bit operatsiyasini bajaradi.	5 ^ 3 ifoda 6 ni beradi
~	'Emas' bit operatori Побитовое НЕ	'Emas' bit operatsiyasi x soni uchun -(x+1) ga to‘g'ri keladi.	~5 ifoda 6 ni beradi.
<	Kichik	X qiymat y qiymatdan kichikligini aniqlaydi. Hamma qiyoslash operatorlari True yoki False qaytaradi. Bu so‘zlardagi katta xarflarga e'tibor bering.	5 < 3 False qaytaradi 3 < 5 ifoda esa True qaytaradi. Ixtiyoriy bir – biri bilan bog'langan ifodalar tuzish ham mumkin: 3 < 5 < 7 ifoda True ni qaytaradi
>	Katta	X qiymat y qiymatdan katta ekanligini aniqlaydi.	5 > 3 ifoda True ni qaytaradi.
<=	Kichik yoki teng	x qiymat y qiymatdan kichik yoki teng ekanligini aniqlaydi.	x = 3; y = 6; x <= y ifoda True qaytaradi.

>=	Katta yoki teng	x qiymat y qiymatdan katta yoki teng ekanligini anqlaydi.	x = 4; y = 3; x >= 3 ifoda True qaytaradi.
==	Teng	Ob'yektlarning tengligini tekshiradi	x = 2; y = 2; x == y ifoda True qaytaradi. x = 'str'; y = 'stR'; x == y ifoda False qaytaradi. x = 'str'; y = 'str'; x == y ifoda True qaytaradi.
!=	Teng emas	Ob'yektlar teng emasligi to‘g'riligini tekshiradi.	x = 2; y = 3; x != y ifoda True qaytaradi.
Not	Mantiqiy 'emas'	(Логическое НЕ) Agar x True bo‘lsa, operator False qaytaradi. Agar x False bo‘lsa operator True qaytaradi.	x = True; not x ifoda False qaytaradi.
And	Mantiqiy 'va' (Логическое И)	x and y ifoda False qaytaradi agar x False bo‘lsa. Aks holda y ning qiymatini qaytaradi.	x = False; y = True; x and y ifoda False qaytaradi, sababi x равно False. Bu holda Python y ning qiymatini tekshirib o‘tirmaydi sababi 'and' operatoridan chapdagi ifoda qismi False ga teng va butun ifoda qiymati boshqa oparatorlar qiymatlariga bog'liqsiz ham False

			bo‘ladi.
Or	Mantiqiy 'yoki'	x or y agar x True ga teng bo‘lsa True qaytaradi aks xolda y ning qiymatini qaytaradi.	x = True; y = False; x or y ifoda True qaytaradi.

1.2.1-chizma. Operatorlar va ularning qo‘llanilishi

Operatorlar va ularning qo‘llanilishini qisqacha ko‘rib chiqamiz. Misol uchun, arifmetik ifodalarini tekshirib ko‘rish uchun interaktiv interpretatordan foydalanishimiz mumkin. Python interpretatori xuddi kalkulator kabi ishlaydi.

```
>>> 2+3
5
>>> 10.5-2.0
8.5
>>> 15-3
12
>>> 8*9
72
>>> 25/3
8.333333333333334
>>> 8**150
290735489718242756219729523155201813741456544274927224112596079672255715245359169330476420285505426224305008642506471173413840651445862 4
>>> 36%5
1
>>> 36//5
7
```

Matematik amallar va o‘zlashtirishlarni qisqacha yozish

Matematik amallar va o'zlashtirishlarni qisqacha yozish

Ko'pincha bir o'zgaruvchi ustida biror matematik amal bajarib, natijani o'sha o'zgaruvchining o'ziga o'zlashtirish zaruriyati tug'iladi. Bu holda amallarni qisqacha yozish mumkin. Siz

a = 2; a = a * 3 ni quyidagicha yozishingiz mumkin:

a = 2; a *= 3

Amallar bajarilish ketma-ketligi

2 + 3 * 4 ifodada qaysi amal birinchi bajariladi: qo'shishmi yoki ko'paytirish?Matematika fanida ko'paytirish birinchi bajarilishi ko'rsatilgan. Demak, ko'paytirish operatori qo'shish operatoriga qaraganda katta

prioritetga(muhimlik darajasiga) ega. Quyidagi jadvalda Python operatorlari prioriteti ko'rsatilgan. Bunda yuqoridan pastga qarab Python operatorlari prioriteti oshib boradi. Bu shuni anglatadiki, ixtiyoriy ifodada Python oldin eng quyidagi operatorlarni hisoblaydi va keyin esa yuqoridagilarini. Amaliyotda esa amallarni qavslar bilan aniq ajratish tavsiya etiladi. Bu dastur kodini oson o'qishga yordam beradi.

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

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

Hisoblash tartibini o'zgartirish

Ifodalarni o'qishni osonlashtirish uchun qavslarni ishlatish mumkin. Misol uchun, 2 + (3 * 4) ni tushunish oson operatorlar prioriteni bilish lozim

bo'lgan 2 + 3 * 4 ifodadan ko'ra. Qavslarni o'ylab ishlatish kerak. Ortiqcha qavslarni ishlatishdan saqlaning. Misol uchun: (2 + (3 * 4)).

Qavslarni ishlatishni ya'na bir afzalligi hisoblash tartibini o'zgartirish imkonini beradi. Misol uchun, qo'shish amalini ko'paytirish amalidan birinchi bajarish kerak bo'lsa, quyidagicha yozish mumkin:

(2 + 3) * 4.

If- shart operatori.

If operatori shartni tekshirish uchun ishlatiladi. Pythonda shart operatorini bir necha xil koʻrinishi mavjud:

1. **if (mantiqiy ifoda):-** shart operatorining bu koʻrinishi mantiqiy ifoda rost boʻlgan holda qandaydir kod bajarilishi uchun ishlatiladi.

2. **if (mantiqiy ifoda):…else**-shart operatorining bu koʻrinishida mantiqiy ifoda rost boʻlsa, birinchi ifodalar bloki bajariladi(bu blok **“if-blok”** deb nomlanadi), aks holda keyingi ifodalar bloki bajariladi(bu blok **“else-blok”** deb nomlanadi).

3. **if (mantiqiy ifoda):…elif(mantiqiy ifoda):…else-** shart operatorining bu koʻrinishida oldingi shart yolgʻon boʻlganda keyingi shart tekshiriladi. Bu ifoda oʻzida ikkita bir-biriga bogʻliq boʻlgan **if else-if else** ifodani bir ifodada **if elif else** saqlaydi. Bu dasturni oʻqishni osonlashtiradi.

Pythonda rostlikka tekshirish

- Har qanday nolga teng boʻlmagan son yoki boʻsh boʻlmagan obyekt-rost
- Nol yoki boʻsh obyekt-yolgʻon
- Taqqoslash amallari True yoki False qiymat qaytaradi
- Mantiqiy operatorlar and va or rost yoki yolgʻon obyekt-operandni qaytadi

Mantiqiy operatorlar:

X and Y

Rost, agar x va y ham rost boʻlsa

X or Y

Rost, agar x yoki y dan bittasi rost bo’lsa

Not X

Rost, agar x yolg‘on bo‘lsa

While sikl operatori

While operatori quyidagi umumiy ko‘rinishga ega:

While sikl operatorining ishlash tartibi

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

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

For operatori

Python dasturlash tilida for operatori C va Paskal dasturlash tillarida qo‘llanishidan farq qiladi. Python da for operatori biroz murakkabroq, lekin while sikliga qaraganda ancha tezroq bajariladi. For…in operatori obyektlar ketmaketligida iteratsiyani amalga oshiradi, ya’ni bu sikl har qanday iteratsiya qilinadigan obyekt bo‘ylab o‘tadi(satr yoki ro‘yxat bo‘ylab) va har bir o‘tish vaqtida sikl tanasini bajaradi.

Range() va xrange funksiyasi

Agar dasturda sonlarni ketma-ket chiqarish kerak bo‘lsa range() funksiyasidan foydalaniladi. U arifmetik progressiyaga asoslangan ro‘yxat tuzadi.

```
>>> range(10)
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
```

Range(10) funksiyasi 10 ta elementdan iborat bo‘lgan ro‘yxat tuzadi. Bunda yuqori chegara sifatida 10 beriladi, lekin u yaratilgan ro‘yxat ketma-ketligiga kirmaydi. Shuningdek funksiyaga quyi chegara va qadamni ham berish mumkin.

```
>>> range(5,10)
[5, 6, 7, 8, 9]
>>> range(0,10,3)
[0, 3, 6, 9]
>>> range(-10,-100,-30)
[-10, -40, -70]
```

Ketma-ketlik indekslarini tanlash uchun range() va len() funksiyalarini birgalikda ishlating.

Foydalanish uchun adabiyotlar

1. Дмитрий Мусин. Самоучитель Python. 2015 г
2. К.Ю. Поляков, В.М. Гуровиц. Язык Python в школьном курсе информатики – М.: Издательский дом МЭИ, 2011. – 424.
3. Г.Россум, Ф.Л.Дж.Дрейк, Д.С.Откидач. Язык программирования Python
4. К.Ю. Поляков, Е.А. Еремин. Информатика, 10 класс.
5. Марк Лутц. Программирование на Python. 1995г.
6. Девид Бизли. Python -Санкт-Петербург: МЭИ, 2008. – Часть III.
7. Сергей Лебедев. Модули и пакеты
8. Прохоренок Н.А. Python.Самое необходимое. – Санкт-Петербург: БХВ-Петербург, 2011, –416 с.

Internet manbalari manzili

9. www.python.org
10. www.uhlib.ru
11. ww.dasturchi.uz

ABDUYEV SHEROZ BOBIROVICH

Algoritmlash va dasturlash, python dasturlash tili

Terishga berildi:_____

Bosishga ruxsat etildi:_____

Ofset bosma qog‘ozi. Qog‘oz bichimi 60x80 1/16

«Times» garniturasi. Ofset bosma usuli.

1 bosma taboq.

Adadi: 50 nusxa.

Buyurma №_____

Samarqand viloyati pedagoglarni yangi metodikalarga o‘rgatish milliy markazi bosmaxonasida chop etildi.

Samarqand shahar, Boysunqur ko‘chasi 3-uy.