Jeymi Chan

Python

0 DAN 100 GACHA

DASTURLASH OLAMIGA QADAM QO'YGAN
HAR BIR DASTURCHI UCHUN

NEW

Jeymi Chan
Python 0 dan 100 gacha

Tarjimon: Sirojiddin To'lanboyev

Python 0 dan 100 gacha

Har bir dasturchi uchun tavsiya qilingan kitob

«Hamroh books» loyihasi
asosida tayyorlandi

Amaliy Loyiha bilan Pythonda darhol kodlashni boshlashingiz kerak bo'lgan yagona kitob

Jeymi Chan

Muqaddima

Ushbu kitob sizga Python dasturlash tilini yaxshi va tez o'rganishda yordam berishi uchun yozildi. Agar siz dasturlash bo'yicha mutlaqo boshlang'ich bo'lsangiz, unda ushbu kitobda murakkab tushunchalar oson usulda tushuntirilgan. Tilni chuqurroq anglashingiz uchun har bir tushunchani namoyish etishda keltirilgan misollar diqqat bilan tanlangan. Agar siz tajribali koder(kodlovchi) bo'lsangiz, ushbu kitob sizga Pythonni o'rganish uchun yaxshi asos bo`lib xizmat qiladi. Kitob so'ngidagi ilovalar sizga Pythonda tez-tez foydalaniladigan ba'zi funksiyalar haqida ma'lumot beradi.
Bundan tashqari, Richard Brenson aytganidek: "Har qanday narsani o'rganishning eng yaxshi usuli - bu bajarish". Kurs oxirida o'rgangan bilimlaringizni amaliyotda qo'llay olishingzga imkon beradigan loyiha bo'yicha ishlaysiz.

Loyiha uchun manba kodini va qo'shimchalarini yuklab olishingiz mumkin bo'lgan havola http://www.learncodingfast.com/python.

1-bob: Python, Python nima?

Hayajonli dasturlash dunyosiga xush kelibsiz. Ushbu kitobniolganingizdanjudaxursandmanvaumidqilamanki, ushbu kitob sizga Python tilini puxta o'rganishingizga va dasturlash quvonchini his qilishingizga yordam beradi. Python dasturlash ummoniga sho'ng'ishdan oldin, avval bir nechta savollarga javob beraylik.

Python nima?

Python - 1980-yillarning oxirida Gvido van Rossum tomonidan yaratilgan keng qo'llaniladigan yuqori darajadagi dasturlash tili. Til kodlarning o'qilishi va soddaligiga katta ahamiyat beradi va dasturchilarga tezkor ravishda dasturlarni ishlab chiqish imkoniyatini yaratadi.

Barcha yuqori darajadagi dasturlash tillari singari, Python kodi ham kompyuterlar anglay olmaydigan ingliz tiliga o'xshaydi.
Biz Pythonda yozadigan kodlarni Python tarjimoni deb nomlanuvchi maxsus dastur tomonidan talqin qilinishi lozim.Biz Python dasturlarini kodlash, sinash va bajarishdan oldin ushbu dasturni o'rnatishimiz kerak bo'ladi. Python tarjimoni dasturini qanday o'rnatishni 2-bobda ko'rib chiqamiz.

Python kodimizni Windows va Mac OS kabi ba'zi eng mashhur operatsion tizimlar uchun yakka o'zi bajariladigan dasturlarga jamlashga imkon beradigan Py2exe yoki Pyinstaller kabi bir qator uchinchi tomon vositalari ham mavjud. Bu bizga foydalanuvchilar Python tarjimonini o'rnatishni talab qilmasdan Python dasturlarini tarqatish imkon beradi.

Nega Pythonni o`rganish kerak?

C, C ++ va Java kabi yuqori darajadagi dasturlash tillari mavjud. Yaxshi yangiligi shundaki, barcha yuqori darajadagi dasturlash tillari bir-biriga juda o'xshashdir. Nimasi bilan farq qiladi, asosan sintaksis, mavjud kutubxonalar va bizning ushbu kutubxonalarga kirish usulimiz. Kutubxona - bu shunchaki o'z dasturlarini yozishda foydalanishimiz mumkin bo'lgan resurslar va oldindan yozilgan kodlar to'plamidir. Agar siz bitta tilni yaxshi o'rgansangiz, birinchi tilni o'rganishingiz kerak bo'lgan vaqt ichida yangi tilni osongina o'rganishingiz mumkin.

Agar siz dasturlash sohasida yangi bo'lsangiz, Pythondan boshlash juda yaxshi qaror. Pythonning asosiy xususiyatlaridan biri bu soddaligi, uni yangi boshlovchilar o'rganishi uchun ideal til ekanligidir. Pythondagi aksariyat dasturlar bir xil vazifani bajarish uchun C kabi boshqa tillarga nisbatan ancha kam kod satrlarini talab qiladi. Bu dasturlashda kamroq xatolarga olib keladi va ishlab chiqish vaqtini qisqartiradi. Bundan tashqari, Python tilning imkoniyatlarini kengaytiradigan uchinchi tomon resurslarining keng to'plamiga ega. Shunday qilib, Python turli xil vazifalar uchun ishlatilishi mumkin, masalan, ish stoli dasturlari, ma'lumotlar bazasi dasturlari, tarmoq dasturlari, o'yinlarni dasturlash va xattoki uyali aloqa uchun.
Va nihoyat, eng muhimi, Python - bu o'zaro faoliyat platforma tili, ya'ni Windows kabi bir operatsion tizim uchun yozilgan kod Python kodiga hech qanday

o'zgartirish kiritmasdan Mac OS yoki Linuxda yaxshi ishlaydi.

Python - o'rganadigan til ekanligiga ishonchingiz komilmi? Qani boshladik...

2-bob: Pythonga tayyorgarlik ko`rish Tarjimon(dasturi)ni o'rnatish

Birinchi Python dasturimizni yozishdan oldin, biz kompyuterlarimiz uchun mos tarjimonni yuklab olishimiz kerak.

Biz ushbu kitobda Python 3 dan foydalanamiz, chunki rasmiy Python saytida aytilganidek *"Python 2.x meros, Python 3.x bu tilning buguni va kelajagi"*. Bundan tashqari, *"Python 3 boshlang'ich dasturchilarni keraksiz ravishda ishdan chiqarishi mumkin bo'lgan ko'plab g'alati narsalarni yo'q qiladi"*.

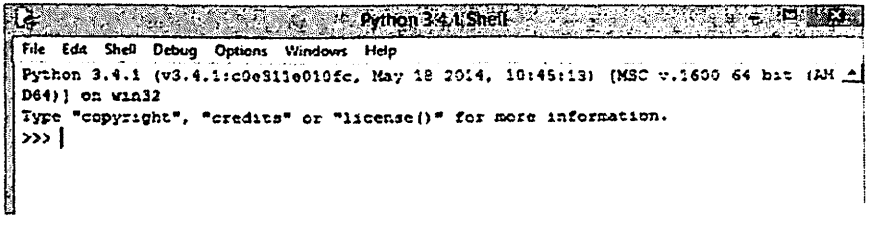
Shunga qaramay, Python 2 hozirgacha juda keng qo'llanilayotganiga e'tibor bering. Python 2 va 3 taxminan 90% o'xshash. Shunday qilib, Python 3 ni o'rgansangiz, Python 2 da yozilgan kodlarni tushunishda hech qanday muammo bo'lmaydi.
Python 3 uchun tarjimonni o'rnatish uchun https://www.python.org/downloads/ saytiga o'ting. To'g'ri versiyasi veb-sahifaning yuqori qismida ko'rsatilishi kerak. Python 3 versiyasini bosing va dasturiy ta'minot yuklab olishni boshlaydi.
Shu bilan bir qatorda, agar siz boshqa versiyani o'rnatmoqchi bo'lsangiz, sahifani pastga aylantiring, shunda siz boshqa versiyalar ro'yxatini ko'rasiz. Siz

ishlatishni xohlagan versiyani ustiga bosing. Ushbu kitobda 3.4.2 versiyasidan foydalanamiz. Siz ushbu versiyani yuklab olish sahifasiga yo'naltirilasiz. Sahifaning oxiriga qarab pastga siljiting va ushbu versiya uchun turli xil o'rnatuvchilar(installers) ro'yxati jamlangan jadvalni ko'rasiz. Kompyuteringiz uchun to'g'ri o'rnatuvchi(installer)ni tanlang. O'rnatish vositasi(installer) ikki omilga bog'liq:

1.The operating system (Windows, Mac OS, or Linux) va
2.The processor (32-bit vs 64-bit) siz foydalanayotgan.

Masalan, agar siz 64-bit Windows kompyuteridan foydalansangiz, ehtimol siz "**Windows x86-64** MSI installer" dan foydalanishingiz mumkin. Uni yuklab olish uchun havolani bosish kifoya. Agar noto'g'ri o'rnatuvchi(unstaller)ni yuklab olib ishga tushirsangiz, tashvishlanmang. Sizga xato xabar keladi va tarjimon uni o'rnatmaydi. Faqat to'g'ri o'rnatuvchi(installer)ni yuklab oling va siz tayyorsiz.

Tarjimonni muvaffaqiyatli o'rnatgandan so'ng, Pythonda kod yozishni boshlashingiz mumkin.



Python Shell, IDLEdan foydalanish va BIRINCHI dasturimizni yozish

Biz kodimizni Python tarjimonimiz bilan birga keltirilgan IDLE dasturi yordamida yozamiz.

Buning uchun avval IDLE dasturini ishga tushiramiz. Siz IDLE dasturini ham boshqa dasturlar kabi ishga tushirasiz. Masalan, Windows 8 da qidiruv maydoniga "IDLE" deb yozib qidirishingiz mumkin. So'ng uni ishga tushirish uchun IDLE (Python GUI) tugmachasini bosing. Sizga quyida ko'rsatilgan Python Shell taqdim etiladi. Python Shell bizga Pythondan interaktiv usulda foydalanish imkonini beradi. Bu bir vaqtning o'zida bitta buyruq kiritishimiz mumkinligni anglatadi. Shell foydalanuvchidan buyruqni kutib, uni bajaradi va bajarish natijasini qaytarib beradi.

Shell-ga quyidagilarni yozib ko'ring. >>> bilan boshlangan qatorlar kiritiladigan buyruqlar bo'lib, buyruqlardan keyingi qatorlar natijalarni ko'rsatadi.

```
>>> 2+3
5
>>> 3>2
True
>>> print ('Hello World')
Hello World
```

`2+3` ni yozganingizda, siz undan 2 + 3 qiymatini

baholashni so'rab Shell-ga buyruq berasiz. Shunga ko'ra, Shell 5-javobni qaytarib beradi. `3>2` ni yozganingizda, Shell-dan 3 ning 2 dan katta ekanligini so'rayapsiz.
Va nihoyat, `print` bu Shell-dan `Hello World` qatorini ko'rsatishni so'raydigan buyruq.

Python Shell - bu Python buyruqlarini sinash uchun juda qulay vosita, ayniqsa birinchi marta til bilan ishlashni boshlaganimizda. Ammo, agar siz Python Shell-dan chiqsangiz va unga qayta kirsangiz, siz kiritgan barcha buyruqlar yo'qoladi. Bundan tashqari, siz Python Shell-dan haqiqiy dastur yaratish uchun foydalana olmaysiz. Haqiqiy dasturni kodlash uchun siz kodingizni matnli faylga yozishingiz va .py kengaytmasi bilan saqlashingiz kerak. Ushbu fayl *Python script* sifatida tanilgan.

Python script-ni yaratish uchun Python Shell-ning yuqori menyusidagi "File > New File" ni bosing. Bu bizning birinchi dasturimiz - "Hello World" dasturini yozishda foydalanadigan matn muharriri haqida ma'lumot beradi. "Hello World" dasturini yozish barcha yangi dasturchilar uchun o'tish jarayonidagi marosim(udum) hisoblanadi. IDLE dasturiy ta'minoti bilan tanishish uchun ushbu dasturdan foydalanamiz.
Text editor-ga quyidagi kodni kiriting (Shell emas).

```
#Prints the Words "Hello World"
print ("Hello World")
```

Siz e'tibor qilishingiz kerakki, `"print"` so'zi siyoh

rangda va "Hello World" so'zi yashil rangda bo'lganda `#Prints the Words "Hello World"` qatori qizil rangda bo'ladi. Bu dasturiy ta'minotning kodimizni o'qishni osonroq qiladigan usuli hisoblanadi. `"print"` va `"Hello World` so'zlari dasturimizda har xil maqsadlarga xizmat qiladi, shu sababli ular turli ranglarda ko'rsatiladi. Keyingi boblarda ko'proq detallarga kirib boramiz.

`#Prints the Words "Hello World"` (qizilda) qatori aslida dasturning bir qismi emas. Bu bizning kodimizni boshqa dasturchilar uchun yanada qulayroq o'qish uchun yozilgan sharhdir. Ushbu qatorga Python tarjimoni e'tibor qilmaydi. Dasturimizga sharhlar qo'shish uchun har bir izoh satri oldida # belgisini quyidagicha yozamiz:

```
#This is a comment
#This is also a comment
#This is yet another comment
```

Alternatively, we can also use three single quotes (or three doublequotes) for multiline comments, like this:

Shu bilan birga, biz ko'p satrli izohlar uchun uchta yakkalik qo'shtirnoq belgisi (yoki uchta iktalik qo'shtirnoq belgisi) dan foydalanishimiz mumkin, masalan:

```
'''
#This is a comment
#This is also a comment
#This is yet another comment
'''
```

Endi kodingizni saqlash uchun File> Save As ... ni bosing. Uni .py kengaytmasi bilan saqlaganingizga ishonch hosil qiling.

Done? Voilà! You have just successfully written your first Python program.
Bajarildimi? Siz birinchi Python dasturini muvaffaqiyatli yozdingiz.

Va nihoyat dasturni ishga tushirish uchun Run> Run Module tugmachasini bosing (yoki F5 tugmachasini bosing). Python Shell-da chop etilgan `Hello World` so'zlarini ko'rishingiz kerak.

Ushbu qadam(bosqich)larni amalda ko'rish uchun siz ushbu ajoyib qo'llanmani mybringback orqali tekshirishingiz mumkin:
https://www.youtube.com/watch?v=pEFr1eYlePw.

Ammo, u videoda Python 2 dan foydalanganligini unutmang, shuning uchun ba'zi buyruqlar sizga xatolik beradi. Agar uning kodlarini sinab ko'rmoqchi bo'lsangiz, print statementlar uchun () belgisini qo'shishingiz kerak.

`print 'Hello World'` deb yozish o'rniga, `print ('Hello World')` shaklida yozishingiz kerak. Bundan tashqari, `raw_input()` ni `input()` ga o'zgartirishingiz kerak. Biz `print()` va `input()` larga 5-bobda to'xtalamiz.

3-bob: Opertaorlar va variable (o'zgaruvchi)lar olami

Hozir biz kirish qismi bilan shug'ullangan bo'lsak, keling, endi haqiqiy(real) jarayonga o'tamiz. Ushbu bobda siz variable(o'zgaruvchi)lar va operatorlar haqida barcha narsani bilib olasiz. Xususan, siz qanday variable(o'zgaruvchi)lar ekanligini va ularni qanday nomlash va e'lon qilishni bilib olasiz. Shuningdek, biz ularni amalga oshiradigan umumiy operatsiyalar haqida bilib olamiz. Tayyormisiz? Qani ketdik.

Variable(o'zgaruvchi)lar nima?

Variable(o'zgaruvchi)lar - bu bizning dasturlarimizda saqlashimiz va boshqarishimiz kerak bo'lgan ma'lumotlarga berilgan nomlardir. Masalan, dasturingiz foydalanuvchi yoshini saqlashi kerak deb taxmin qiling. Buning uchun biz ushbu ma'lumotni `userAge` deb nomlashimiz va `userAge` variable(o'zgaruvchi)ni quyidagi statement yordamida aniqlashimiz mumkin.

```
userAge = 0
```

`userAge` variable(o'zgaruvchisi)ni aniqlaganingizdan so'ng, sizning dasturingiz ushbu ma'lumotlarni saqlash uchun kompyuteringizning ma'lum hajmdagi maydonini ajratadi. Keyin ushbu ma'lumotlarga `userAge` nomi bilan murojaat qilib kirishingiz va o'zgartirishingiz mumkin. Har safar yangi variable(o'zgaruvchi)ni e'lon qilsangiz, unga boshlang'ich qiymatni berishingiz kerak. Ushbu misolda biz unga 0 qiymatini berdik. Keyinroq ushbu qiymatni dasturimizda doim o'zgartirishimiz mumkin.

Shuningdek, biz bir vaqtning o'zida bir nechta variable(o'zgaruvchi)ni aniqlashimiz mumkin. Buning uchun shunchaki yozing

```
userAge, userName = 30, 'Peter'
```

Bu quyidagiga ekvivalent(teng)

```
userAge = 30
userName = 'Peter'
```

Variable(o'zgaruvchi)ga nom berish (nomlash)

Python-da variable(o'zgaruvchi) nom faqat harflar (a - z, A - B), raqamlar yoki pastki chiziqlarni (_) o'z ichiga olishi mumkin. Biroq, birinchi belgi raqam bo'lishi mumkin emas. Shu sababli, siz variable(o'zgaruvchi) ga `userName`, `user_name` yoki `userName2` nomlarini qo'yishingiz mumkin, lekin `2userName` emas.

Bundan tashqari, ba'zi bir saqlangan so'zlar mavjud bo'lib, ulardan variable(o'zgaruvchi) nom sifatida foydalana olmaysiz, chunki ular allaqachon Python-da oldindan tayinlangan ma'nolarga ega. Ushbu saqlangan so'zlarga `print`, `input`, `if`, `while` kabi so'zlar kiradi va ularning har biri haqida keyingi boblarda bilib olamiz.

Vanihoyat, variable(o'zgaruvchi) nomlar katta-kichikligi bilan ajralib turadi. `username` `userName` bilan bir xil emas.

Python-da variable(o'zgaruvchi)ni nomlashda ikkita konvensiya(an'ana) mavjud. Biz camel case yozuvidan yoki pastki chiziqlardan foydalanishimiz mumkin. Camel case - aralash case bilan qo'shma so'zlarni yozish amaliyoti hisoblanadi (masalan, `thisIsAVariableName`). Bu biz kitobning qolgan qismida ishlatadigan konvensiyadir. Shu bilan birga, yana bir keng tarqalgan amaliyot - so'zlarni ajratish uchun pastki chiziqlardan (_) foydalanish. Agar xohlasangiz,

variable(o'zgaruvchi)laringizni shunday nomlashingiz mumkin: `this_is_a_variable_name`.

Topshiriq belgisi

`userAge = 0` statement-dagi = belgisi, biz matematikadan o'rgangan = belgisidan boshqacha ma'noga ega ekanligini unutmang. Dasturlashda = belgisi topshiriq belgisi sifatida tanilgan. Demak, biz = belgisining o'ng tomonidagi qiymatni chapdagi variable(o'zgaruvchi)ga topshiramiz. `userAge = 0` iborasini tushunishning yaxshi usuli bu uni `userAge <- 0` deb o'ylashdir.

`x = y` va `y = x` iboralari dasturlashda juda boshqacha ma'nolargaega.

Chalkashdingizmi? Misol buni aniqlik kiritib beradi.

IDLE editor-ga quyidagi kodni kiriting va save qiling(saqlang).

```
x = 5
y = 10
x = y
print ("x = ", x)
print ("y = ", y)
```

Endi dasturni ishga tushiring. Siz ushbu output(natija)ni olishingiz kerak:

```
x = 10
y = 10
```

x ning boshlang'ich qiymati 5 ga teng bo'lsa-da (birinchi qatorda e'lon qilingan), uchinchi qator `x = y` y qiymatini x (`x <- y`) ga beradi, shuning uchun x qiymati 10 ga o'zgaradi, y qiymati o'zgarishsiz qoladi .

Keyin FAQAT bitta iborani o'zgartirish orqali dasturni o'zgartiring: Uchinchi qatorni `x = y` dan `y = x` ga o'zgartiring. Matematik jihatdan x = y va y = x bir xil narsani anglatadi. Biroq, dasturlashda bunday emas.

Ikkinchi dasturni ishga tushiring. Endi quyidagicha olasiz

```
x = 5
y = 5
```

Ushbu misolda ko'rib turganingizdek, x qiymati 5 bo'lib qoladi, lekin y qiymati 5 ga o'zgartiriladi, chunki `y = x` iborasi x qiymatini y (`y <- x`) ga beradi. y 5 bo'ladi, x esa 5 ligicha qoladi.

Asosiy Operatorlar

Variable(O'zgaruvchi)ga boshlang'ich qiymatni berishdan tashqari, biz variable-lar bo'yicha odatiy matematik operatsiyalarni ham bajarishimiz mumkin. Python-ning asosiy operatorlariga +, -, *, /, //,% va ** kiradi, ular mos ravishda qo'shish, ayirish, ko'paytirish, bo'lish, qavatli bo'lish, modul va tushuntiruvchini ifodalaydi.

Misol:

Taxmin qilaylik x = 5, y = 2

Qo'shish:
x + y = 7

Ayirish:
x - y = 3

Ko'paytirish:
x*y = 10

Bo'lish:
x/y = 2.5

Qavatli bo'lish(Floor Division):
x//y = 2 (javobni butun songa yaxlitlaydi)

Modul:
x%y = 1 (5 ni 2 ga bo`lganda qoldiqni beradi)
Tushuntiruvchi:
x**y = 25 (5 kvadrat natijasi)

Qo'shimcha topshiriq operatorlari

= belgisidan tashqari, Pythonda (va aksariyat dasturlash tillarida) bir nechta topshiriq operatorlari mavjud. Bularga + =, - = va * = kabi operatorlar kiradi.

Faraz qilaylik, bizda `x` o'zgaruvchisi bor, boshlang'ich qiymati 10 ga teng. Agar x ni 2 ga oshirishni xohlasak, quyidagicha yozishimiz mumkin

```
x = x + 2
```

The program will first evaluate the expression on the right (`x + 2`) and assign the answer to the left. So eventually the statement above becomes `x <- 12`. Dastur avval o'ng tarafdagi ifodani baholaydi (`x + 2`) va javobni chap tarafga belgilaydi. Shunday qilib, oxir-oqibat yuqoridagi ibora
`x <- 12` ga aylanadi.

`x = x + 2` yozish o'rniga, xuddi shu ma'noni ifodalash uchun
`x += 2` yozishimiz mumkin. + = belgisi aslida topshiriq belgisini qo'shish operatori bilan birlashtirgan stenografiyadir(tekstni shartli belgilar bilan tez yozib olish). Demak, `x += 2` shunchaki
`x = x + 2` degan ma'noni anglatadi.

Xuddi shunday, agar biz ayirma qilishni xohlasak, `x = x - 2` yoki `x -= 2` ni yozishimiz mumkin. Yuqoridagi bobda aytib o'tilgan 7 operatorlar uchun ham xuddi shunday ishlaydi.

4-bob: Python-da data turlari

Ushbu bobda biz avval Python-dagi ba'zi bir asosiy ma'lumotlar turlarini, xususan integer(butun son), float va stringni ko'rib chiqamiz. Keyinchalik, biz turdagi casting tushunchasini o'rganamiz. Va nihoyat, biz Python-da yana uchta rivojlangan data turlarini muhokama qilamiz: list, tuple va lug'at(dictionary).

Integer-lar

Integer(butun son)lar - bu o'nlik qismlarsiz raqamlar, masalan -5, -4, -3, 0, 5, 7 va boshqalar.

Python-da butun sonni e'lon qilish uchun `variableName =initial value` ni yozing

Misol:

```
userAge = 20, mobileNumber = 12398724
```

Float

Float 1.234, -0.023, 12.01 kabi o'nli qismlarga ega bo'lgan raqamlarni anglatadi.

Python-da float namoyon qilish uchun, `variableName = initialvalue` shaklida yozamiz

Misol:

```
userHeight = 1.82, userWeight = 67.2
```

String

String matnga ishora qiladi.

Bir string-ni namoyon qilish uchun siz `variableName = 'initial value'` (bir qo'shtirnoq) yoki `variableName ="initial value"` (juft qo'shtirnoq) belgilaridan foydalanishingiz mumkin.

Misol:

```
userName = 'Peter', userSpouseName = "Janet", userAge = '30'
```

Oxirgi misolda, biz `userAge = '30'` yozganimiz uchun, `userAge` integerdir(butun son)

Birlashtiruvchi belgi (+) yordamida bir nechta pastki qatorlarni birlashtira olamiz. Misol uchun, `"Peter" + "Lee"` string-i `"PeterLee"` string-ga ekvivalentdir.

O'rnatilgan string funksiyalari

Python string-larni boshqarish uchun bir qator o'rnatilgan funktsiyalarni o'z ichiga oladi. Funksiya - bu shunchaki ma'lum bir vazifani bajaradigan qayta ishlatiladigan kodlar bloki. Biz 7-bobda funksiyalarni yanada chuqurroq muhokama qilamiz.

Python-da mavjud bo'lgan bir funksiyaning misoli string-

lar uchun `upper()` usulidir. Siz uni string-dagi barcha harflarni katta harflar bilan yozish uchun ishlatasiz. Masalan, `'Peter'.upper()` bizga `"PETER"` string-ini beradi. Python-ning o'rnatilgan string usullaridan qanday foydalanish haqida ko'proq misollar va namunaviy kodlar uchun A Ilovaga murojaat qilishingiz mumkin.

% Operator yordamida string-larni formatlash

String-larni % operatori yordamida ham formatlash mumkin. Bu sizning string-izni qanday ko'rsatilishini va saqlanishini xohlashingiz ustidan nazoratni ta'minlaydi. % Operatoridan foydalanish sintaksisi quyidagicha

```
"string to be formatted" %(values or variables to be inserted into string, separated by commas)
```

Ushbu sintaksisning uchta qismi mavjud. Dastlab qo'shtirnoqlarda string-ni formatlash uchun yozamiz. Keyin % belgisini yozamiz. Va nihoyat, bizda bir qator dumaloq qavslar () mavjud bo'lib, ularning ichida biz string-ga kiritiladigan qiymatlarni yoki o'zgaruvchilarni yozamiz. Ichkarida qiymatlarga ega bo'lgan ushbu dumaloq qavslar, aslida, biz keyingi bo'limlarda ko'rib chiqiladigan ma'lumotlar turi bo'lgan *tuple* deb nomlanadi.

IDLE-ga quyidagi kodni kiriting va ishga tushiring.

```
brand = 'Apple'
exchangeRate = 1.235235245

message = 'The price of this %s laptop is
%d USD
and the exchange rate is %4.2f USD to 1
EUR' % (brand, 1299, exchangeRate)

print (message)
```

Yuqoridagi misolda, string `'The price of this %s laptop is %d USD and the exchange rate is %4.2f USD to 1 EUR'` biz formatlashni istagan string hisoblanadi. Biz stringda `%s`, `%d` va `%4.2f` formatlashtiruvchilaridan joylashtiruvchilar sifatida foydalanamiz.

Ushbu joylashtiruvchilar `brand` o'zgaruvchi bilan bilan almashtiriladi, mos ravishda dumaloq qavsda ko'rsatilgandek 1299 qiymati va `exchangeRate` o'zgaruvchisi ham almashtiriladi. Agar biz kodni ishlatsak, biz quyida natija(output)ni olamiz.

```
The price of this Apple laptop is 1299 USD
and theexchange rate is 1.24 USD to 1 EUR
```

`%s` formatlashtiruvchi stringni ko'rsatish uchun ishlatiladi (bu holda "Apple"),`%d` formatlashtiruvchi esa butun son(integer)ni (1299) ifodalaydi. Agar biz butun son(integer)dan oldin bo'sh joy qo'shishni xohlasak, biz

string-ning kerakli uzunligini ko'rsatish uchun % va d oralig'ida son qo'sha olamiz. Masalan, "%5d" % (123) bizga " 123" ni beradi (oldida 2 bo'sh joy va umumiy uzunligi 5).

%f formatlashtiruvchisi floatlarni formatlashda ishlatiladi (o'nli raqamlar). Here we format it as %4.2f where 4 refers to the total length and 2 refers to 2 decimal places. If we want to add spaces before the number, we can format is as %7.2f, which will give us " 1.24" (with 2 decimal places, 3 spaces in front and a total length of 7).

.format() usuli yordamida string-larni formatlash

Stringlarni formatlash uchun % operatoridan foydalanishdan tashqari, Python bizga string-larni formatlash uchun format() usulini ham taqdim etadi. Sintaksis

```
"string to be formatted".format(values
or variables to be inserted into string,
separated by commas)
```

Formatlash usulidan foydalanganda, biz %s, %f yoki %d ni to'ldiruvchi sifatida ishlatmaymiz. Buning o'rniga biz quyidagicha jingalak qavslardan foydalanamiz:

```
message = 'The price of this {0:s} laptop
is {1:d} USD and the exchange rate is
{2:4.2f} USD to 1 EUR'.format('Apple',
```

1299, 1.235235245)

Jingalak qavs ichida ikki nuqtadan so'ng avval foydalaniladigan parametr pozitsiyanini yozamiz. Ikki nuqtadan keyin formatlashtiruvchini yozamiz. Jingalak qavs ichida bo'sh joy bo'lmasligi kerak.

('Apple', 1299, 1.235235245) format-ni yozganimizda, biz uchta parametr bo'yicha format() usuliga o'tamiz.
Parametrlar - bu metod o'z vazifasini bajarish uchun zarur bo'lgan ma'lumotlar. Parametrlar - 'Apple', 1299 va 1.235235245.

'Apple' parametri 0 pozitsiyasiga,
1299 1 pozitsiyasiga va
1.235235245 2 pozitsiyasiga ega.

Pozitsiyalar doim ZERO dan boshlanadi.

{0:s} ni yozganimizda, biz tarjimondan {0:s} ni 0 holatidagi parametr bilan almashtirishini va uning string ekanligini so'raymiz (chunki formatlashtiruvchi "s" dir).

{1: d} ni yozganimizda, biz 1-pozitsiyadagi integer(formatlashtiruvchi d) bo'lgan parametrga ishora qilamiz.

Biz {2: 4.2f} ni yozganimizda, biz 2-pozitsiyadagi parametrni nazarda tutamiz, bu float va uni 2 ta o'nlik kasrlar bilan va umumiy uzunligi 4 ga teng bo'lishini

xohlaymiz (formatlashtiruvchi 4.2f).

Agar biz `message`(xabar)ni print qilsak, quyidagicha ko'rinish olamiz

```
The price of this Apple laptop is 1299 USD
and theexchange rate is 1.24 USD to 1 EUR
```

Note: If you do not want to format the string, you can simply write

Izoh: Agar siz string-ni formatlashni xohlamasangiz, shunchaki yozishingiz mumkin

```
message = 'The price of this {} laptop is
{} USDand the exchange rate is {} USD to
1 EUR'.format('Apple', 1299, 1.235235245)
```

Bu yerda biz parametrlarning holatini belgilashimiz shart emas. Tarjimon taqdim etilgan parametrlarning tartibiga qarab jingalak qavslarni almashtiradi. Biz olamiz

```
The price of this Apple laptop is 1299 USD
and theexchange rate is 1.235235245 USD
to 1 EUR
```

`format()` usuli yangi boshlanuvchilar uchun chalkash bo'lishi mumkin. Aslida, string-ni formatlash biz ko'rib chiqqan narsalarga qaraganda ancha xayoliy bo'lishi mumkin, ammo biz ko'rib chiqqan narsalar ko'p maqsadlar uchun yetarli. `format()` usulini yaxshiroq tushunish uchun quyidagi dasturni sinab ko'ring.

```
message1 = '{0} is easier than
{1}'.format('Python', 'Java') message2 =
'{1} is easier than
{0}'.format('Python', 'Java')
message3    =    '{:10.2f}    and    {:d}'.
format(1.234234234,        12)
message4 = '{}'.format(1.234234234)

print (message1)
#You'll get 'Python is easier than Java'

print (message2)
#You'll get 'Java is easier than Python'

print (message3)
#You'll get ' 1.23 and 12'
#You do not need to indicate the positions
of theparameters.

print (message4)
#You'll get 1.234234234. No formatting is
done.
```

Python Shell-dan `format()` usuli bilan tajriba o'tkazish uchun foydalanishingiz mumkin usul. Har xil string-larda yozib ko'ring va nima olganingizni ko'ring.

Pythonda Casting turi

Ba'zan bizning dasturimizda bitta ma'lumot turidan ikkinchisiga, masalan, integer-dan string-ga o'tishimiz kerak bo'ladi. Bu casting turi sifatida tanilgan.

Python-da uchta o'rnatilgan funksiyalar mavjud bo'lib, ular biz uchun kasting turini amalga oshirishga imkon beradi. Bular `int()`, `float()` va `str()` funksiyalari.

Python-dagi `int()` funksiyasi float yoki tegishli string-ni oladi va uni integer-ga aylantiradi. Float-ni butun son(integer)ga almashtirish uchun `int(5.712987)` raqamini kiritishimiz mumkin. Natijada 5 ni olamiz (kasr punkti olib tashlanganidan keyin har qanday narsa). String-ni integer-ga o'zgartirish uchun biz `int ("4")` yozsak bo'ladi va biz 4 ni olamiz. Ammo biz `int ("Hello")` yoki `int ("4.22321")` yozolmaymiz. Ikkala holatda ham xatolikka yo'l qo'yamiz.

`float()` funksiyasi butun sonni yoki tegishli qatorni oladi va uni suzuvchi qilib o'zgartiradi. Masalan, biz float (2) yoki float ("2") ni yozsak, biz 2.0 ga egamiz. Agar biz float ("2.09109") yozsak, biz tirnoqlar olib tashlanganligi sababli satr emas, balki float bo'lgan 2.09109 ni olamiz.

Boshqa tomondan `str()` funksiyasi butun son(integer) ni yoki float-ni string-ga aylantiradi. Masalan, `str(2.1)` ni yozsak, biz "2.1" ni olamiz.

Hozir Python-dagi uchta asosiy data turlarini va ularni casting-ini ko'rib chiqdik, keling, yanada rivojlangan data turlariga o'tamiz.

List

List(ro'yxat) odatda bir-biriga bog'liq bo'lgan ma'lumotlar(data) to'plamiga ishora qiladi. Ushbu ma'lumotlarni alohida o'zgaruvchilar sifatida saqlash o'rniga, ularni ro'yxat sifatida saqlashimiz mumkin. Masalan, bizning dasturimiz 5 ta foydalanuvchining yoshini saqlashi kerak deb taxmin qilaylik. Ularni `user1Age`, `user2Age`, `user3Age`, `user4Age` va `user5Age` sifatida saqlash o'rniga ularni ro'yxat sifatida saqlash mantiqan to'g'ri keladi.

Ro'yxatni e'lon qilish uchun siz `listName = [boshlang'ich qiymatlar]`.yozasiz. Ro'yxatni e'lon qilishda biz kvadrat qavslardan [] foydalanamiz. Bir nechta qiymatlar vergul bilan ajratilgan.

Misol:

```
userAge = [21, 22, 23, 24, 25]
```

Shuningdek, biz unga biron bir boshlang'ich qiymat bermasdan ro'yxatni e'lon qilishimiz mumkin. Biz shunchaki `listName = []` yozamiz. Hozir bizda bo'sh ro'yxat, unda hech qanday narsalar yo'q. Ro'yxatga ma'lumotlar qo'shish uchun biz quyida keltirilgan `append()` usulidan foydalanishimiz kerak.

Ro'yxatdagi individual qiymatlarga ularning indekslari bo'yicha kirish mumkin, va indekslar har doim 1 dan emas, ZERO dan boshlanadi. Bu deyarli barcha dasturlash tillarida, masalan C va Java da odatiy

holdir. Shuning uchun birinchi qiymat 0 indeksiga, keyingisi 1 va shunga o'xshash ko'rsatkichlarga ega. Masalan, `userAge[0] = 21, userAge[1] = 22`

Shu bilan bir qatorda, ro'yxat qiymatlariga orqa tomondan kirishingiz mumkin. Ro'yxatdagi oxirgi element -1 indeksiga, oxirgidan ikkinchisi -2 indeksga ega va hokazo. Demak `userAge[-1] = 25, userAge[-2] = 24.`

O'zgaruvchiga ro'yxatni yoki uning bir qismini tayinlashingiz mumkin. Agar`userAge2 = userAge` deb yozsangiz, `userAge2` o'zgaruvchisi `[21,22, 23, 24, 25]` bo'ladi.

Agar siz `userAge3 = userAge[2:4]` yozsangiz, `userAge` ro'yxatidan `userAge3` ro'yxatiga 4-1 indeksiga 2 indeksli elementlarni tayinlaysiz. Boshqacha qilib aytganda, `userAge3 = [23, 24].`

2: 4 yozuvi bo'lak sifatida tanilgan. Python-da har doim bu bo'lak yozuvini ishlatsak, boshlang'ich indeksidagi element har doim qo'shiladi, lekin oxiridagi element har doim chiqarib tashlanadi. Shuning uchun 2: 4 yozuvlari 2-indeksdan 4-1 gacha bo'lgan ko'rsatkichlarga (ya'ni 3-indeks) taalluqlidir, shuning uchun `userAge3 = [23, 24]` dir, `[23, 24, 25]` emas.

Slice(bo'lak) yozuvlari stepper deb nomlanuvchi uchinchi raqamni o'z ichiga oladi. Agar `userAge4 = userAge[1:5:2]` deb yozsak, biz 1-indeksdan 5-1

indeksgacha bo'lgan har bir ikkinchi sondan iborat sub-ro'yxatni olamiz, chunki step 2 ga teng. Demak, `userAge4 = [22, 24]`.

Bundan tashqari, slice yozuvlari foydali standartlar(default)ga ega. Birinchi raqam uchun default nolga, ikkinchi raqam uchun default - bu kesilgan listning kattaligi. Masalan, `userAge[ :4]` sizga 0 indeksdan 4-1 indeksgacha, `userAge[1: ]` sizga indeksdan 5gacha indeksgacha qiymatlarni beradi (chunki `userAge` hajmi 5 ga teng, ya'ni `userAge` 5 ta elementga ega).

Ro'yxatdagi elementlarni o'zgartirish uchun `listName[index of item to be modifiedo'zgartirilishi kerak bo'lgan element indeksi] = new value`(yangi qiymat) yozamiz. Masalan, agar siz ikkinchi elementni o'zgartirmoqchi bo'lsangiz, `userAge[1] = 5.` deb yozasiz. Sizning ro'yxatingiz `userAge = [21, 5, 23, 24, 25]` bo'ladi.
Elementlarni qo'shish uchun `append()` funksiyasidan foydalanasiz. Masalan, agar siz

`userAge.append(99)` yozing, siz ro'yxatning oxiriga 99 qiymatini qo'shasiz. Sizning ro'yxatingiz endi `userAge = [21, 5, 23, 24, 25, 99]`

Elementlarni olib tashlash uchun `del listName[index of item tobe deleted]` yozasiz. Masalan, `del userAge[2]` deb yozsangiz, endi sizning ro'yxatingiz

userAge = [21, 5, 24, 25, 99] bo'ladi (uchinchi element o'chirilgan)

Ro'yxat ishini to'liq baholash uchun quyidagi dasturni ishga tushiring.

```
#declaring the list, list elements can be
ofdifferent data types
myList = [1, 2, 3, 4, 5, "Hello"]

#print the entire list.print(myList)
#You'll get [1, 2, 3, 4, 5, "Hello"]

#print the third item (recall: Index starts
fromzero).
print(myList[2])#You'll get 3

#print the last item. print(myList[-1])
#You'll get "Hello"

#assign myList (from index 1 to 4) to myList2
andprint myList2
myList2 = myList[1:5]print (myList2)

#You'll get [2, 3, 4, 5]

#modify the second item in myList and print
theupdated list
myList[1] = 20print(myList)
#You'll get [1, 20, 3, 4, 5, 'Hello']
```

```
#append a new item to myList and print the
updatedlist
myList.append("How are you")print(myList)
#You'll get [1, 20, 3, 4, 5, 'Hello', 'How
areyou']

#remove the sixth item from myList and
print the updated list
del myList[5]print(myList)
#You'll get [1, 20, 3, 4, 5, 'How are
you']
```

Ro'yxat bilan qilishingiz mumkin bo'lgan yana ikkita narsa bor. Namunaviy kodlar va ro'yxat bilan ishlash bo'yicha ko'proq misollar uchun B ilovaga murojaat qiling.

Tuple

Tuple-lar xuddi ro'yxatlarga o'xshaydi, lekin siz ularning qiymatlarini o'zgartira olmaysiz. Dastlabki qiymatlar - bu dasturning qolgan qismida qoladigan qiymatlar. Dasturlar yil oylarining nomlarini saqlashi kerak bo'lgan paytlarda foydali bo'lishi mumkin bo'lgan misol.

Tuple-ni e'lon qilish uchun `tupleName = (initial values(boshlang'ich qiymatlar))` yozasiz. Tuple-ni e'lon qilishda biz dumaloq qavslardan () foydalanamiz.
Bir nechta qiymatlar vergul bilan ajratilgan.

Misol:

```
monthsOfYear = ("Jan", "Feb", "Mar",
"Apr", "May",
"Jun", "Jul", "Aug", "Sep", "Oct", "Nov",
"Dec")
```

Ro'yxatga o'xshash indekslar yordamida tuple-ning individual qiymatlariga kirishingiz mumkin.
Shuning uchun, `monthsOfYear[0] = "Jan"`, `monthsOfYear[-1] = "Dec"`.

Tuple bilan nima qilishingiz mumkinligi haqida ko'proq misollar uchun C Ilovani ko'rib chiqing.

Dictionary

Lug'at(dictionary) - bu tegishli ma'lumotlar to'plamidir PAIRS. Masalan, agar biz 5 foydalanuvchi nomini va yoshini saqlamoqchi bo'lsak, ularni lug'atda saqlashimiz mumkin.

Lug'atni e'lon qilish uchun `dictionaryName = {dictionary key : data}`, lug'at kalitlari noyob bo'lishi sharti bilan (bitta lug'at ichida). Ya'ni, siz bunday lug'atni e'lon qila olmaysiz

```
myDictionary = {"Peter":38, "John":51,
"Peter":13}.
```

Buning sababi, "Piter" lug'at kaliti sifatida ikki marta ishlatilgan. Lug'atni e'lon qilishda biz jingalak qavslardan {} foydalanamiz. Bir nechta juftliklar vergul bilan ajratilgan.

Misol:

```
userNameAndAge = {"Peter":38, "John":51,
"Alex":13, "Alvin":"Not Available"}
```

Siz `dict( )` usuli yordamida lug'at(dictionary)ni e'lon qilishingiz mumkin. Yuqoridagi `userNameAndAge` lug'atini e'lon qilish uchun quyidagicha yozasiz

```
userNameAndAge = dict(Peter = 38, John =
51, Alex = 13, Alvin = "Not Available")
```

Lug'atni e'lon qilish uchun ushbu usuldan foydalanganda, jingalak qavslar o'rniga {} dumaloq qavslar () ishlatasiz va lug'at tugmachalariga qo'shtirnoq(quotation) qo'ymaysiz.

Lug'atdagi alohida elementlarga kirish uchun bizdictionary tugmachasidan foydalanamiz, bu `{dictionary key : data}` juftligining birinchi qiymati. Masalan, Jonning yoshini olish uchun siz `userNameAndAge["John"]` yozasiz. Siz 51 qiymatini olasiz.

Lug'atdagi elementlarni o'zgartirish uchun biz `dictionaryName[dictionary key of item to be modified] = new data.` yozamiz. Masalan, `"John":51` juftligini o'zgartirish uchun `userNameAndAge["John"] = 21` ni yozamiz. Endi bizning lug'at `userNameAndAge = {"Peter":38, "John":21, "Alex":13, "Alvin":"Not Available"}.` ko'rinishda.

Shuningdek, biz lug'atni unga biror bir dastlabki qiymat bermasdan e'lon qilishimiz mumkin. Biz shunchaki `dictionaryName = { }` yozamiz. Hozir bizda bo'sh lug'at bor, unda hech qanday element yo'q.

Lug'atga elementlarni qo'shish uchun quyidagini yozamiz `dictionaryName[dictionary key] = data.` Masalan, agar biz lug'atimizga `"Joe":40` ni qo'shmoqchi bo'lsak, `userNameAndAge["Joe"] = 40` ko'rinishida yozamiz. Bizning lug'at `userNameAndAge = {"Peter":38, "John":21, "Alex":13,`

"Alvin":"Not Available", "Joe":40} ko'rinishda bo'ladi.

Lug'atdan elementlarni olib tashlash uchun del dictionaryName[dictionary key]ni yozamiz. Masalan, "Alex":13 juftini olib tashlash uchun , del userNameAndAge["Alex"]ko'rinishida yozamiz. Endi bizning lug'at userNameAndAge = {"Peter":38, "John":21, "Alvin":"Not Available", "Joe":40}ko'rinishda bo'ladi.

Bularning barchasini amalda ko'rish uchun quyidagi dasturni ishga tushiring.

```
#declaring the dictionary, dictionary keys
anddata can be of different data types
myDict = {"One":1.35, 2.5:"Two Point
Five", 3:"+", 7.9:2}

#print the entire dictionaryprint(myDict)
#You'll get {2.5: 'Two Point Five', 3:
'+', 'One': 1.35, 7.9: 2}
#Note that items in a dictionary are not
stored in the same order as the way you
declare them.

#print the item with key = "One".
print(myDict["One"])
#You'll get 1.35

#print the item with key = 7.9.
```

```
print(myDict[7.9])
#You'll get 2

#modify the item with key = 2.5 and print
theupdated dictionary
myDict[2.5] = "Two and a Half"print(myDict)
#You'll get {2.5: 'Two and a Half', 3:
'+', 'One': 1.35, 7.9: 2}

#add a new item and print the updated
dictionarymyDict["New item"] = "I'm new"
print(myDict)
#You'll get {'New item': 'I'm new', 2.5:
'Two and a Half', 3: '+', 'One': 1.35,
7.9: 2}
#remove the item with key = "One" and
print the updated dictionary
del myDict["One"]print(myDict)
#You'll get {'New item': 'I'm new', 2.5:
'Two anda Half', 3: '+', 7.9: 2}
```

Lug'at bilan ishlashning ko'proq misollari va namunaviy kodlari uchun D ilovaga murojaat qilishingiz mumkin.

5-bob: Dasturingizni interaktiv qilish

O'zgaruvchilar asoslarini ko'rib chiqdik, keling endi, ulardan foydalanadigan dastur yozaylik. Biz 2-bobda yozgan "Hello World" dasturini qayta ko'rib chiqamiz, ammo bu safar biz uni interaktiv qilamiz: Faqat dunyoga salom aytish o'rniga, biz dunyo ham ismlarimiz va yoshimizni bilishini istaymiz. Buning uchun dasturimiz bizdan ma'lumot so'rab, ularni ekranda ko'rsatishi kerak.

Buni biz uchun ikkita built-in(o'rnatilgan) funksiya bajarishi mumkin: `input()` va `print()`.
Hozircha quyidagi dasturni IDLE-da yozamiz. Uni saqlang va ishga tushiring.

```
myName = input("Please enter your name:
")myAge = input("What about your age: ")

print ("Hello World, my name is", myName,
"and Iam", myAge, "years old.")
```

Dastur sizning ismingizni so'rashi kerak.

```
Please enter your name:
```

Aytaylik, siz James ismini kiritdingiz. Endi Enter tugmachasini bosing va bu sizning yoshingizni so'raydi.

```
What about your age:
```

20-raqamni kiritganingizni ayting. Endi yana Enter tugmasini bosing. Siz quyidagi bayonot(statement)ni olishingiz kerak:

```
Hello World, my name is James and I am 20
yearsold.
```

Input()

Yuqoridagi misolda, foydalanuvchimizning ismi va yoshini olish uchun `input(` funksiyasidan foydalandik.

```
myName = input("Please enter your name:
")
```

"Please enter your name:" string(qator)i foydalanuvchiga ko'rsatmalar berish uchun ekranda ko'rsatiladi. Foydalanuvchi tegishli ma'lumotlarni kiritgandan so'ng, bu ma'lumotlar `myName` o'zgaruvchisida **string** sifatida saqlanadi. Keyingi input statement foydalanuvchini yoshini talab qiladi va ma'lumotni `myAge` o'zgaruvchisida **string** sifatida saqlaydi.

`input()` funksiyasi Python 2 va Python 3 da biroz farq qiladi. Python 2 da, agar foydalanuvchi kiritilishini string sifatida qabul qilmoqchi bo'lsangiz, uning o'rniga `raw_input()` funksiyasidan foydalanishingiz kerak.

Print()

`print()` funksiyasi foydalanuvchilarga ma'lumotlarni ko'rsatish uchun ishlatiladi. U nol va undan ortiq ifodalarni vergul bilan ajratilgan parametr sifatida qabul qiladi.

Quyidagi bayonotda biz `print()` funksiyasiga 5 parametrni o'tkazdik. Ularni aniqlay olasizmi?

```
print ("Hello World, my name is", myName,
"and Iam", myAge, "years old.")
```

Birinchisi - `"Hello World, my name is"`
Keyingisi - avvalgi kirish funksiyasi yordamida e'lon qilingan `myName` o'zgaruvchisi.
Keyingi string-da `"and I am"` stringi, so'ngra `myAge` o'zgaruvchisi va nihoyat `"years old."` stringi joylashgan.

`myName` va `myAge` o'zgaruvchilariga murojaat qilishda biz qo'shtirnoq belgisidan foydalanmaymiz. Agar siz qo'shtirnoq belgisidan foydalansangiz, quyidagi output-ni olasiz

```
Hello World, my name is myName and I am
myAge years old.
```

o'rniga, bu biz xohlagan narsa emasligi aniq.

Ko'chirmani o'zgaruvchilar bilan chop etishning yana

bir usuli - 4-bobda o'rgangan % formatlashtiruvchidan foydalanish. Yuqoridagi birinchi print bayonot bilan bir xil natija(output)ga erishish uchun biz yozishimiz mumkin

```
print ("Hello World, my name is %s and I
am %syears old." %(myName, myAge))
```

Va nihoyat, `format()` usuli yordamida bir xil bayonotni chop etish uchun quyidagicha yozamiz

```
print ("Hello World, my name is {} and I
am {}years old".format(myName, myAge))
```

`print()` funksiyasi - bu Python 2 va Python 3 da farq qiladigan yana bir funksiya. Python 2 da uni quyidagi kabi qavslarsiz yozasiz:

```
print "Hello World, my name is " + myName
+ " andI am " + myAge + " years old."
```

Uchtalik qo'shtirnoq

If you need to display a long message, you can use the triple-quotesymbol (''' or """) to span your message over multiple lines. For instance,
Agar sizga uzun xabarni ko'rsatish kerak bo'lsa, o'zingizning xabaringizni bir nechta satrlarga uzatish uchun uch qo'shtirnoqli belgidan (''' yoki """) foydalanishingiz mumkin. Misol uchun,

```
print ('''Hello World. My name is James
and
I am 20 years old.''')
```

quyidagicha natija hosil bo'ladi

```
Hello World.
My name is James andI am 20 years old.
```

Bu sizning xabaringizning o'qilishini oshirishga yordam beradi.

Belgilardan qochish(qochish belgilari)

Masalan, yorliqni chop etish uchun t harfidan oldin teskari chiziq belgisini quyidagicha yozamiz: \t. \ belgisiz t harfi bosilib chiqadi. U bilan yorliq chop etiladi. Shunday qilib, agar siz `print ('Hello\tWorld')` yozsangiz, `Hello World`-ga ega bo'lasiz.

Orqaga burish belgisining boshqa keng tarqalgan ishlatilishi quyida keltirilgan.
>>> buyruqni ko'rsatadi va quyidagi qatorlar natijasini ko'rsatadi.

\n (newline(yangi qator)ni bosib chiqaradi)

```
>>> print ('Hello\nWorld')
Hello
World
```

\\ (Orqaga burish belgisini o'zi bosib chiqaradi)

```
>>> print ('\\')
\
```

\" (Ikkala qo'shtirnoq satr oxiriga ishora qilmasligi uchun, ikkita qo'shtirnoqni bosib chiqaradi)

```
>>> print ("I am 5'9\" tall")I am 5'9"
tall
```

\' (Bitta qo'shtirnoq satr oxiriga ishora qilmasligi uchun bitta qo'shtirnoqni chop eting)

```
>>> print ('I am 5\'9" tall') I am 5'9"
tall
```

Agar oldin \ belgi qo'yilgan belgilar maxsus belgilar sifatida talqin qilinishini istamasangiz, birinchi qo'shtirnoq oldidan r qo'shib chala string-lardan foydalanishingiz mumkin. Masalan, yorliq sifatida talqin qilinishini istamasangiz, `print (r'Hello\tWorld')` yozishingiz kerak. Output sifatida `Hello\tWorld`-ni olasiz.

6-bob: Tanlash va qaror qabul qilish

Tabriklaymiz, siz eng qiziqarli bobga o'tdingiz. Umid qilamanki, siz shu paytgacha kursni yoqtirdingiz. Ushbu bobda biz o'zingizning dasturingizni qanday qilib aqlli, tanlov va qarorlar qabul qilishga qodir qilishni ko'rib chiqamiz. Xususan, biz `if` ifodasini, `for` loop va `while` loop-ni ko'rib chiqamiz. Ular boshqaruv oqimi vositalari sifatida tanilgan; ular dastur oqimini boshqaradilar. Bundan tashqari, biz xatolarni yuzaga keltirganda dastur nima qilishi kerakligini belgilaydigan bayonotdan tashqari, sinab ko'rishni ham ko'rib chiqamiz.

Biroq, ushbu boshqaruv oqimi vositalariga o'tishdan oldin, avval shartlar bayonotlarini ko'rib chiqishimiz kerak.

Condition Statements(Shartbayonotlari)

Barcha nazorat oqimi vositalari shart bayonotini baholashni o'z ichiga oladi. Dastur shart bajarilganligiga qarab turlicha davom etadi.

Shartlarning eng keng tarqalgan bayonoti taqqoslash bayonidir. Agar ikkita o'zgaruvchining bir xilligini taqqoslamoqchi bo'lsak, biz == belgisidan (double =) foydalanamiz. Masalan, agar siz `x == y` yozsangiz, dasturdan `x` ning `y` qiymati bilan uning qiymatiga tengligini tekshirishni so'raysiz. Agar ular teng bo'lsa, shart bajariladi va bayonot `True` ga baho beradi. Boshqa holda, bayonot `False` ga baholanadi.

Boshqa taqqoslash belgilariga quyidagilar kiradi: = (teng emas), <(kichik),> (katta), <= (kichikroq yoki teng) va > = (katta yoki teng). Quyidagi ro'yxat ushbu belgilarni qanday ishlatilishini ko'rsatadi va "`True`" ga baho beradigan bayonotlarga misollar keltiradi.

Teng emas:
5! = 2

Katta:
5> 2
Kichikroq:
2 <5

5> = 2

5> = 5 ga nisbatan katta yoki unga teng
Kichikroq yoki teng:
2 <= 5
2 <= 2

Bizda shuningdek uchta mantiqiy operator `and, or, not(va,yoki,emas,` agar biz bir nechta shartlarni birlashtirmoqchi bo'lsak, bular foydali.

Agar barcha shartlar bajarilsa `and` operatori `True` qiymatini qaytaradi. Aks holda u noto'g'ri bo'ladi. Masalan, `5==5 and 2>1` iborasi `True` ga qaytadi, chunki ikkala shart ham `True`.

Hech bo'lmaganda bitta shart bajarilsa `or` operatori `True` qiymatini qaytaradi. Aks holda u noto'g'ri bo'ladi. `5 > 2 or 7 > 10 or 3 == 2` iborasi `True` bo'ladi, chunki birinchi shart `5>2 True` bo'ladi.

`not` kalit so'zidan keyingi shart noto'g'ri bo'lsa, `not` operatori `True` qiymatini qaytaradi. Aks holda u `False(no'to'g'ri)` bo'ladi. `not 2>5` iborasi `True` qiymatiga qaytadi, chunki 2 5 dan katta emas.

If bayonoti

`if` bayonoti eng ko'p ishlatiladigan boshqaruv oqimi bayonotlaridan biridir. Bu dasturga ma'lum bir shart bajarilganligini baholashga va baholash natijasi asosida tegishli harakatlarni bajarishga imkon beradi. `if` ifodasining tuzilishi quyidagicha:

```
if condition 1 is met:
    do A
elif condition 2 is met:
    do B
elif condition 3 is met:
    do C
elif condition 4 is met:
    do D
else:
    do E
```

`elif` "else if" degan ma'noni anglatadi va siz qancha xohlasangiz shuncha `elif` bayonotiga ega bo'lishingiz mumkin

Agar avval C yoki Java kabi boshqa tillarda kod yozgan bo'lsangiz, `if`, `elif` va `else` kalit so'zlaridan keyin Python-da qavslar () kerak emasligini ko'rib hayron bo'lishingiz mumkin. Bundan tashqari, Python `if` ifodasining boshi va oxirini aniqlash uchun jingalak {} qavslardan foydalanmaydi. Aksincha, Python ichkaridan foydalanadi. Chiqib ketgan har qanday narsa, agar

shart to'g'ri bo'lsa, bajariladigan kod bloki sifatida ko'rib chiqiladi.

If iborasi qanday ishlashini to'liq bilish uchun IDLE-ni yoqing va quyidagi kodni kiriting.

```
userInput = input('Enter 1 or 2: ')

if userInput == "1": print ("Hello
World") print ("How are you?")
elif userInput == "2": print ("Python
Rocks!") print ("I love Python")
else:
print ("You did not enter a valid number")
```

Dastur avval foydalanuvchidan input uchun `input` funksiyasidan foydalanib kiritishni so'raydi. Natijada `userInput` o'zgaruvchisida satr(string) sifatida saqlanadi.

`if userInput == "1"` ifodasini keltiring: `userInput` o'zgaruvchisini "1" qatori bilan taqqoslaydi. Agar `userInput` da saqlanadigan qiymat "1" bo'lsa, dastur indentatsiya tugaguniga qadar barcha indentlangan mulohazalarni bajaradi. Ushbu misolda u "`Hello World`" ni, so'ng "`How are you?`" ni chiqaradi.

Shu bilan bir qatorda, agar `userInput`-da saqlanadigan qiymat "2" bo'lsa, dastur "`Python Rocks`" ni, so'ngra "`I love Python`" ni bosib chiqaradi.

Boshqa barcha qiymatlar uchun dasʧur "`You did not enter avalid number`"("Siz raqamni kiritmadingiz") ni chop etadi.

Dasʧurni uch marta ishlating, har bir ish uchun mos ravishda 1, 2 va 3 ni kiriting. Siz quyidagi natijani olasiz:

```
Enter 1 or 2: 1Hello World
How are you?
Enter 1 or 2: 2Python Rocks!
I love Python

Enter 1 or 2: 3
You did not enter a valid number
```

Inline If

Inline `if` iborasi `if` ning sodda shakli va agar siz oddiy vazifani bajarishingiz kerak bo'lsa, qulayroq bo'ladi. Sintaksis:

```
do Task A if condition is true else do Task B
```

Masalan,

```
num1 = 12 if myInt==10 else 13
```

Ushbu ibora, agar `myInt` 10 ga teng bo'lsa, `num1` (Topshiriq A) ga 12 ni beradi, aks holda u `num1` ga (Topshiriq B) 13 beradi.

Yana bir misol

```
print ("This is task A" if myInt == 10 else "This is task B")
```

Agar `myInt` 10 ga teng bo'lsa, bu bayonot `"This is task A"` (Topshiriq A) ni bosib chiqaradi(print qiladi), aks holda u `"This is task B"` (Topshiriq B) ni bosib chiqaradi(print qiladi).

For Loop (Sikli)

Keyin `for` loop-ni ko'rib chiqamiz. `for` sikli `for` bayonidagi shart yaroqsiz bo'lmaguncha kod blokini qayta-qayta bajaradi.

Qaytadan o'tish mumkin (Looping through an iterable)

Python-da iterable string, list yoki tuple kabi ulanishi mumkin bo'lgan hamma narsani anglatadi. Qaytadan o'tish uchun sintaksis quyidagicha:

```
for a in iterable:
    print (a)
```

Misol:

```
pets = ['cats', 'dogs', 'rabbits',
'hamsters']
for myPets in pets:
    print (myPets)
```

Yuqoridagi dasturda biz avval `pets` (uy hayvonlari) ro'yxatini e'lon qilamiz va a'zolarga `'cats'`, `'dogs'`, `'rabbits'` va `'hamsters'` ni beramiz.
Keyinchalik, `for myPets in pets` bayonoti: `pets` ro'yxatini ko'rib chiqing va ro'yxatdagi har bir a'zoni `myPets` o'zgaruvchisiga tayinlang.

Dasfur birinchi marta `for` loop orqali ishlaydi, u `myPets` o'zgaruvchisiga `'cats'` ni tayinlaydi. Bayonot `print (myPets)`

keyin `'cats'` qiymatini bosib chiqaradi. Ikkinchi marta dasturlar `for` ifodasini ko'rib chiqishda myPets-ga `'dogs'` qiymatini beradi va `'dogs'` qiymatini chiqaradi. Dasfur ro'yxat oxirigacha ro'yxatni ko'rib chiqishni davom ettiradi.

Agar dasturni ishga tushirsangiz, quyidagicha ko'rinish bo'ladi

```
cats
dogs
rabbits
hamsters
```

Ro'yxatdagi a'zolarning indeksini ham ko'rsatishimiz mumkin. Buning uchun biz `enumerate()` funksiyasidan foydalanamiz.

```
for index, myPets in enumerate(pets):
    print(index,myPets)
```

Bu bizga quyidagi natijani beradi

```
0  cats
1  dogs
2  rabbits
3  hamster
```

Keyingi misol string-ni qanday aylantirishni ko'rsatadi.

```
message = 'Hello'

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

Natija

```
H
e
l
l
o
```

Raqamlar ketma-ketligini ko'rib chiqish

Raqamlar ketma-ketligini ko'rib chiqish uchun o'rnatilgan `range()` funksiyasi foydalidir. `range()` funksiyasi raqamlar ro'yxatini hosil qiladi va sintaksis `range` ga ega (`start, end, step`).

Agar `start` berilmasa, hosil qilingan raqamlar noldan boshlanadi.

Eslatma: Bu yerda eslash qolish kerak bo'lgan foydali maslahat shundaki, Pythonda (va aksariyat dasturlash tillarida), agar boshqacha ko'rsatilmagan bo'lsa, biz har doim noldan boshlaymiz.
Masalan, list va tuple indeksi noldan boshlanadi. String-lar uchun `format()` usulidan foydalanganda

parametrlarning joylashuvi noldan boshlanadi. `range()` *funksiyasidan foydalanilganda, agar* `start` *berilmasa, hosil qilingan sonlar noldan boshlanadi.*

Agar `step` berilmasa, ketma-ket raqamlar ro'yxati hosil bo'ladi (ya'ni step = 1 (qadam = 1)). `end` (yakuniy) qiymat berilishi kerak. Ammo, `range()` funksiyasi bilan bog'liq bir g'alati narsa shundaki, berilgan `end` qiymat hech qachon hosil qilingan ro'yxat tarkibiga kirmaydi.

Misol uchun,
`range(5)` [0, 1, 2, 3, 4] list-ini hosil qiladi
`range(3, 10)` [3, 4, 5, 6, 7, 8, 9] list-ini hosil qiladi
`range(4, 10, 2)` [4, 6, 8] list-ini hosil qiladi

`range()` funksiyasi `for` operatorida qanday ishlashini ko'rish uchun quyidagi kodni ishlatib ko'ring:

```
for i in range(5):print (i)
```

Quyidagicha ko'rinishda natija olasiz

```
0
1
2
3
4
```

While Loop

Biz ko'rib chiqadigan keyingi nazorat oqimi ifodasi `while` loopidir. Nomidan ham ko'rinib turibdiki, `while` loopi ichidagi ko'rsatmalarni qayta-qayta bajaradi, ma'lum bir shart esa amal qiladi.
`while` ifodasining tuzilishi quyidagicha:

```
while condition is true:
    do A
```

`while` loopdan foydalanganda ko'pincha, biz loop hisoblagichi sifatida ishlash uchun avval o'zgaruvchini e'lon qilishimiz kerak. Keling buni shunchaki o'zgaruvchan `counter` deb nomlaymiz. `while` operatoridagi shart `counter` ning qiymatini uning ma'lum bir qiymatdan kichikligini (yoki kattaroqligini) aniqlash uchun baholaydi. Agar shunday bo'lsa, loop bajariladi. Keling, dasturning namunasini ko'rib chiqaylik.

```
counter = 5

while counter > 0:
    print ("Counter = ", counter)counter
    =counter-1
```

Agar siz dasturni ishga tushirsangiz, quyidagi natijani olasiz

```
Counter = 5
Counter = 4
Counter = 3
```

```
Counter = 2
Counter = 1
```

Bir qarashda, `while` iborasi eng sodda sintaksisga o'xshaydi va ulardan foydalanish eng oson bo'lishi kerak. Biroq, cheksiz ilmoqlar xavfi tufayli `while` looplaridan foydalanishda ehtiyot bo'lish kerak. E'tibor bering, yuqoridagi dasturda bizda `counter = counter - 1` hisoblagichi mavjud? Ushbu yo'nalish hal qiluvchi ahamiyatga ega. U `counter` qiymatini 1 ga kamaytiradi va ushbu yangi qiymatni `counter` ga asl qiymatining ustiga yozib qo'yadi.

`counter` qiymatini 1 ga kamaytirishimiz kerak, shunda `while counter > 0` bo'lsa, loop holati oxir-oqibat `False` (noto'g'ri) bo'ladi. Agar buni qilishni unutib qo'ysak, loop cheksiz ishlaydi va natijada cheksiz loop paydo bo'ladi. Agar siz ushbu tajribani boshdan kechirishni istasangiz, `counter = counter - 1` qatorini o'chirib tashlang va dasturni qayta ishga tushirishga urinib ko'ring. Dasturni qandaydir tarzda yo'q qilmaguningizcha, dastur `counter = 5` ni bosib chiqarishni davom ettiradi. Ayniqsa, agar sizda katta dastur mavjud bo'lsa va siz qaysi kod segmentining noaniq loopga olib kelishini bilmasangiz, bu yoqimli tajriba emas.

Break

Looplar bilan ishlashda, ba'zida ma'lum bir shart bajarilganda butun sikl(loop)dan chiqishni xohlashingiz mumkin. Buning uchun biz `break` kalit so'zidan foydalanamiz. Qanday ishlashini ko'rish uchun quyidagi dasturni ishga tushiring.

```
j = 0
for i in range(5):j = j + 2
print ('i = ', i, ', j = ', j)if j == 6:
break
```

Siz quyidagi natijani olishingiz kerak.

```
i   = 0 , j = 2
i   = 1 , j = 4
i   = 2 , j = 6
```

`break` kalit so'zisiz dastur i = 0 dan i= 4 gacha loop qilishi kerak, chunki biz `range(5)` funksiyasidan foydalanganmiz. Ammo `break` kalit so'zi bilan dastur muddatidan oldin i = 2 da tugaydi, chunki i = 2 bo'lsa, `j` 6 qiymatiga yetadi va `break` kalit so'zi loopning tugashiga olib keladi.
Yuqoridagi misolda e'tibor bering, biz `for` loop ichida `if` iborasini ishlatganmiz. Dasturlashda turli xil boshqaruv vositalarini, masalan, `if` ifodasi ichida `while` loopini ishlatish yoki `while` loopining ichida `for` loopini ishlatish kabi «aralashtirish» juda keng tarqalgan. Bu ichki boshqaruv bayonoti sifatida tanilgan.

Continue

Loop-lar uchun yana bir foydali kalit so'z `continue` so'zidir. `Continue` dan foydalanganimizda, kalit so'zdan keyin qolgan loop shu takrorlash uchun o'tkazib yuboriladi. Bir misol buni yanada aniqroq qiladi.

```
j = 0
for i in range(5):
    j = j + 2
    print ('\ni = ', i, ', j = ', j)
    if j == 6:
        continue
print ('I will be skipped over if j=6')
```

Siz quyidagi natijani olishingiz kerak.

J = 6 bo'lsa, `continue` kalit so'zidan keyingi satr print qilinmaydi. Bundan tashqari, hamma narsa odatdagidek ishlaydi.

Try, Except

Biz ko'rib chiqadigan yakuniy nazorat bayonoti, bu `try, except`. Ushbu bayonot xato yuzaga kelganda dasturning qanday ishlashini nazorat qiladi. Sintaksis quyidagicha:

```
try:
    do something
except:
    do something else when an error occurs
```

Misol uchun, quyidagi dasturni ishga tushirishga harakat qiling

```
try:
    answer =12/0
    print (answer)
except:
    print ("An error occurred")
```

Dasturni ishga tushirganingizda, siz `"An error occurred" ("Xato yuz berdi")` xabarini olasiz. Buning sababi, dastur `try` blokida `answer =12/0` iborasini bajarishga harakat qilganda, xato yuzaga keladi, chunki siz raqamni nolga bo'la olmaysiz. `try` blokining qolgan qismi e'tiborga olinmaydi va uning o'rniga `except` blokidagi bayonot bajariladi.

Agar siz xatolaringizga qarab aniqroq xato xabarlarini

foydalanuvchilaringizga ko'rsatishni istasangiz, xato turini except kalit so'zidan keyin belgilashingiz mumkin.

Quyidagi dasturni ishlatib ko'ring.

```
try:
    userInput1 = int(input("Please enter
a  number:
"))
    userInput2 = int(input("Please enter
another  number:  "))
    answer =userInput1/userInput2
    print ("The answer is ", answer)
    myFile  =  open("missing.txt",  'r')
except ValueError:
    print  ("Error:  You  did  not  enter  a
number")
except ZeroDivisionError:
    print ("Error: Cannot divide by zero")
except  Exception  as  e:
    print ("Unknown error: ", e)
```

Quyidagi ro'yxat turli xil foydalanuvchi kirishlari uchun turli xil natijalarni ko'rsatadi.
>>> foydalanuvchi kiritilishini va => chiqishni bildiradi.

```
>>> Please enter a number: m
=> Error: You did not enter a number
```

Sabab: Foydalanuvchi butun son(integer)ga o'tkazib bo'lmaydigan string-ni kiritdi. Bu ValueError. Demak, ValueError-dan except blokidagi bayonot ko'rsatiladi.

```
>>> Please enter a number: 12
>>> Please enter another number: 0
=> Error: Cannot divide by zero
```

Sabab: `userInput2` = 0. Raqamni nolga bölolmasligimiz sababli, bu `ZeroDivisionError`. `ZeroDivisionError` blokidan `except` bayonot ko'rsatiladi.

```
>>> Please enter a number: 12
>>> Please enter another number: 3
=> The answer is 4.0
=> Unknown error: [Errno 2] No such file or directory: 'missing.txt'
```

Sabab: Foydalanuvchi qabul qilinadigan qiymatlarni kiritadi va chiziqli le values and the line `print ("The answer is ", answer)` to'g'ri bajariladi. Ammo keyingi satrda xatolik yuzaga keladi, chunki yo'qolgan.txt topilmadi. Bu `ValueError` yoki `ZeroDivisionError` emasligi sababli oxirgi `except` blok bajariladi.

`ValueError` va `ZeroDivisionError` - bu Python-da oldindan belgilangan xato turlaridan ikkitasi. `ValueError` o'rnatilgan operatsiya yoki funksiya to'g'ri turga ega, ammo noo'rin qiymatga ega bo'lgan parametrni olganida ko'tariladi. Dastur nolga bo'linishga harakat qilganda `ZeroDivisionError` ko'tariladi. Python-dagi boshqa keng tarqalgan xatolar
IOError:
I/O operatsiyasi (masalan, o'rnatilgan `open ()` funksiyasi)

I / O bilan bog'liq sabablarga ko'ra ishlamay qolganda ko'tariladi, masalan, "fayl topilmadi".

ImportError:
Import bayonoti modul ta'rifini topa olmaganda ko'tariladi

IndexError:
Ketma-ketlik (masalan, string, list, tuple) ko'rsatkichi doiradan tashqarida bo'lganda ko'tariladi.

KeyError:
Lug'at kaliti topilmaganda ko'tariladi.
NameError:
Mahalliy yoki global nom topilmaganda ko'tariladi.

TypeError:
Amaliyot yoki funksiya mos bo'lmagan obyektga nisbatan qo'llanilganda ko'tariladi.
Python-dagi barcha xato turlarining to'liq ro'yxati uchun murojaat qilishingiz mumkin

Python-dagi barcha xato turlarining to'liq ro'yxati uchun murojaat qilishingiz mumkin
https://docs.python.org/3/library/exceptions.html.

Python shuningdek, har xil turdagi har bir xato uchun oldindan belgilangan xato xabarlari bilan birga keladi. Agar siz xabarni ko'rsatishni xohlasangiz, xato turidan keyin kalit so'zidan foydalanasiz. Masalan, standart `ValueError` xabarini ko'rsatish uchun siz quyidagilarni yozasiz:

```
except ValueError as e:
    print (e)
```

e - bu xatoga tayinlangan o'zgaruvchining nomi. Siz unga boshqa ismlarni ham berishingiz mumkin, ammo "e" dan foydalanish odatiy holdir. Dasturimizdagi oxirgi except ifodasi quyidagicha

```
except Exception as e:
    print ("Unknown error: ", e)
```

oldindan belgilangan xato xabaridan foydalanishning misoli. Bu har qanday kutilmagan xatolarga yo'l qo'yishning so'nggi urinishi bo'lib xizmat qiladi.

7-bob: Funksiyalar va Modullar

Avvalgi boblarda biz funksiyalar va modullarni qisqacha eslatib o'tdik. Ushbu bobda, ularni batafsil ko'rib chiqamiz. Shuni takrorlash kerakki, barcha dasturlash tillari dasturchilar sifatida hayotimizni yengillashtirish uchun foydalanishimiz mumkin bo'lgan ichki-o'rnatilgan kodlar bilan ta'minlangan. Ushbu kodlar oldindan yozilgan sinflar, o'zgaruvchilar va ma'lum umumiy vazifalarni bajarish funksiyalaridan iborat bo'lib, modul sifatida tanilgan fayllarda saqlanadi. Avvalo funksiyalarni ko'rib chiqamiz.

Funksiyalar nima?

Funksiyalar - bu shunchaki oldindan ma'lum bir vazifani bajaradigan oldindan yozilgan kodlar. O'xshashlik uchun, MS Excel dasturida mavjud bo'lgan matematik funksiyalar haqida o'ylab ko'ring. Raqamlarni qo'shish uchun A1 + A2 + A3 + A4 + A5 yozish o'rniga sum () funksiyasidan foydalanishimiz va summani (A1: A5) terishimiz mumkin.
Funksiya qanday yozilganiga, u sinfning bir qismi bo'ladimi (sinf - bu obyektga yo'naltirilgan dasturlash konsepsiyas bo'libi, biz uni bu kitobda ko'rib chiqmaymiz) va uni qanday import qilishingizga qarab, funksiya nomi yoki nuqta yozuvidan foydalangan holda biz funksiyani oddiygina yozish orqali chaqirishimiz mumkin. Ba'zi funksiyalar bizdan o'z vazifalarini bajarish uchun ma'lumotlarni uzatishni talab qiladi. Ushbu ma'lumotlar parametrlar sifatida tanilgan va biz ularni funksiyalarga ularning qiymatlarini vergul bilan ajratilgan qavs () ichiga qo'yish orqali yetkazamiz.

Masalan, `print()` funksiyasini ekranda ko'rsatish uchun biz uni `print("Hello World")` yozish orqali chaqiramiz, bu yerda bosib chiqarish funksiyasi nomi va `"Hello World"` esa parametrdir.

Boshqa tomondan, matn satrlarini manipulyatsiya qilish(ishlov berish) uchun `replace()` funksiyasidan foydalanish uchun `"Hello World". replace ("World","Universe")` deb yozishimiz kerak, bu yerda

`replace` funksiya nomi, `"World"` va `"Universe"`-bu parametrlar. Nuqtadan oldingi satr (ya'ni `"Hello World"`) ta'sirlanadigan satr. Demak, `"Hello World"` `"Hello Universe"` ga o'zgartiriladi.

O'zingizning funksiyalaringizni aniqlash

Biz Python-da o'z funksiyalarimizni aniqlay olamiz va ularni dastur davomida qayta foydalanamiz. Funksiyani aniqlash uchun sintaksis quyidagicha:

```
def functionName(parameters):
    code detailing what the function
    should doreturn [expression]
```

Bu yerda 2 ta kalit so'z mavjud, `def` va `return`.

`def` dasturga keyingi qatordan boshlab indentent kodning funktsiya qismi ekanligini aytadi. `return` - bu funksiyadan javob qaytarish uchun foydalanadigan kalit so'z. Funksiyada bir nechta `return` operator-lari bo'lishi mumkin. Ammo, funksiya `return` operatorini bajargandan so'ng, funksiya chiqadi. Agar sizning funksiyangizda biron bir qiymatni qaytarish kerak bo'lmasa, siz `return` operatorini qoldirishingiz mumkin. Shu bilan bir qatorda, `return` va `return None` deb yozishingiz mumkin.

Keling, birinchi funksiyamizni aniqlaymiz. Aytaylik, berilgan son tub son ekanligini aniqlashni xohladik. Bu yerda biz funksiyani 3-bobda o'rgangan modul (%) operatori va `for` loop va `if` 6-bobdan bilib olgan bo'lsak, qanday aniqlay olishimiz bayon etilgan.

```
def checkIfPrime (numberToCheck):
```

```
    for x in range(2, numberToCheck):
        if (numberToCheck%x == 0):
            return False
    return True
```

Yuqoridagi funksiyada 2 va 3-qatorlar uchun `for` loop ishlatiladi, natijada berilgan `numberToCheck` parametrini barcha sonlar bo'yicha `2` dan `numberToCheck - 1` gacha bo'lgan raqamlarga ajratib, qoldiq nolga tengligini aniqlaydi. Agar qoldiq nolga teng bo'lsa, `numberToCheck` oddiy son emas. 4-qatorda `False` qaytadi va funksiya tugaydi.

Agar `for` loopining oxirgi takrorlanishi bilan bo'linishlarning hech biri nolga qoldiq bermasa, funksiya 5-qatorga yetib boradi va `True` ga qaytadi. Keyin funksiya tugaydi.

Bufunksiyadanfoydalanishuchun, `checkIfPrime(13)` deb yozamiz va biz uni quyidagi kabi o'zgaruvchiga tayinlaymiz

```
answer = checkIfPrime(13)
```

Bu yerda biz parametr sifatida 13 ga o'tmoqdamiz. Keyin biz `print(answer)` yozish orqali javobni chop etishimiz mumkin. Biz natijani olamiz: `True`.

Variable Scope (O'zgaruvchan Ko'lam)

Funksiyani belgilashda tushunadigan muhim tushuncha - bu o'zgaruvchan ko'lam tushunchasi. Funksiya ichida aniqlangan o'zgaruvchilar tashqarida aniqlangan o'zgaruvchilardan farq qiladi. Ikkita asosiy farqlar mavjud.

Birinchidan, funksiya ichida e'lon qilingan har qanday o'zgaruvchiga faqat funksiya ichida kirish mumkin. Ular mahalliy o'zgaruvchilar sifatida tanilgan. Funksiyadan tashqarida e'lon qilingan har qanday o'zgaruvchi global o'zgaruvchi sifatida tanilgan va unga dasturning istalgan joyida kirish mumkin.

Buni tushunish uchun quyidagi kodni sinab ko'ring:

```
message1 = "Global Variable"
def myFunction():
    print("\nINSIDE THE FUNCTION")
    #Global variables are accessible
inside a function
    print (message1)
    #Declaring a local variable
    message2 = "Local Variable"
    print (message2)

#Calling the function
myFunction()

print("\nOUTSIDE THE FUNCTION")
```

```
#Global variables are accessible outside
functionprint (message1)

#Local variables are NOT accessible
outside function.
print (message2)
```

Agar dasturni ishga tushirsangiz, quyida natijani olasiz.

```
INSIDE THE FUNCTION
Global VariableLocal Variable

OUTSIDE THE FUNCTION
Global Variable
NameError: name 'message2' is not defined
```

Funksiya doirasida mahalliy va global o'zgaruvchilarga kirish mumkin. Funksiya tashqarisida `message2` mahalliy o'zgaruvchisi endi mavjud emas. Biz `NameError` ni funksiyadan tashqarida kirishga urinayotganda olamiz.

O'zgaruvchan ko'lamini tushunishning ikkinchi konsepsiyasi shundaki, agar mahalliy o'zgaruvchi global o'zgaruvchiga o'xshash nomga ega bo'lsa, funksiya ichidagi har qanday kod mahalliy o'zgaruvchiga kirish huquqiga ega. Tashqaridagi har qanday kod global o'zgaruvchiga kirish huquqiga ega. Quyidagi kodni ishlatib ko'ring

```
message1 = "Global Variable (shares same
```

```
name as alocal variable)"

def myFunction():
    message1 = "Local Variable (shares
same name as a global variable)"
    print("\nINSIDE THE FUNCTION").
    print (message1)

# Calling the functionmyFunction()

# Printing message1 OUTSIDE the function
print ("\nOUTSIDE THE FUNCTION")
print (message1)
```

Siz quyidagi natijalarni olasiz:

```
INSIDE THE FUNCTION
Local Variable (shares same name as a
global variable)

OUTSIDE THE FUNCTION
Global Variable (shares same name as a
local variable)
```

Funksiya ichiga `message1` deb yozganimizda, u mahalliy o'zgaruvchini bosib chiqarayotganda "`Local Variable (shares same name as a global variable`"ni bosib chiqaradi.
Biz uni tashqarida chop etganimizda, u global o'zgaruvchiga kirmoqda va shuning uchun "`Global Variable (shares same name as a local variable)`"ni bosib chiqaradi.

Modullarni import qilish

Python ko'plab o'rnatilgan funksiyalar bilan ta'minlangan. Ushbu funksiyalar modul sifatida tanilgan fayllarda saqlanadi. Python modullarida ichki kodlardan foydalanish uchun avval ularni dasturlarimizga import qilishimiz kerak. Buni biz `import` kalit so'zidan foydalanib qilamiz. Buning uchta usuli mavjud.

Birinchi usul - `importmoduleName` yozish orqali butun modulni import qilish.

Masalan, `random (tasodifiy)` modulni import qilish uchun `import random` deb yozamiz. `randrange()` funksiyasini `random` modulda ishlatish uchun biz `random.randrange(1, 10)` ko'rinishdayozamiz.

Agar siz funksiyadan har safar `random` yozishni juda qiyin deb hisoblasangiz, `import random as r` deb yozib modulni import qilishingiz mumkin (bu erda `r` siz tanlagan har qanday ism). Endi `randrange()` funksiyasidan foydalanish uchun shunchaki `r.randrange(1, 10)` yozasiz.

Modullarni import qilishning uchinchi usuli - moduldan ma'lum funksiyalarni yozish orqali import qilish `from moduleName import name1[, name2[,`

```
... nameN]].
```

Masalan, `randrange()` funksiyasini `random` moduldan import qilish uchun biz `from random import randrange` deb yozamiz. Agar biz bir nechta funksiyalarni import qilmoqchi bo'lsak, ularni vergul bilan ajratamiz. `randrange()` va `randint()` funksiyalarini import qilish uchun biz `from random import randrange, randint` deb yozamiz. Funksiyadan hozir foydalanish uchun endi nuqta yozuvidan foydalanishimiz shart emas. Shunchaki `randrange(1, 10)` deb yozing.

O'z modulimizni yaratish

O'rnatilgan modullarni import qilishdan tashqari, biz o'z modullarimizni yaratishimiz mumkin. Agar kelajakda boshqa dasturlash loyihalarida qayta ishlatmoqchi bo'lgan ba'zi funksiyalaringiz bo'lsa, bu juda foydali.

Modulni yaratish juda oddiy. Faylni a .py kengaytmasi bilan saqlang va uni import qilmoqchi bo'lgan Python fayli bilan bir xil papkaga qo'ying.

Siz boshqa Python skriptida ilgari aniqlangan `checkIfPrime ()` funksiyasidan foydalanmoqchisiz. Bu yerda qanday amalga oshirishingiz ko'rsatilgan. Avval yuqoridagi kodni ish stolida `prime.py` sifatida saqlang. `prime.py` quyidagi kodga ega bo'lishi kerak.

```
def checkIfPrime (numberToCheck):
    for x in range(2, numberToCheck):
        if (numberToCheck%x == 0):
            return False
    return True
```

Keyin yana bir Python faylini yarating va unga nom bering `useCheckIfPrime.py`. Uni ish stolingizda ham saqlang. `useCheckIfPrime.py` ko'rinishidagi kodga ega bo'lishi kerak.

```
import prime
answer = prime.checkIfPrime(13) print
(answer)
```

Endi useCheckIfPrime.py-ni ishga tushiring. Siz True natijani olishingiz kerak. Bu oddiy.

Biroq, prime.py va useCheckIfPrime.py-ni turli papkalarda saqlashni xohlayapsiz deylik. Python tarjimoniga modulni qayerdan topishini aytib berish uchun useCheckIfPrime.py uchun ba'zi kodlarni qo'shishingiz kerak bo'ladi.

prime.py-ni saqlash uchun C diskingizda "MyPythonModules" nomli papka yaratganingizni ayting. useCheckIfPrime.py faylining yuqori qismiga quyidagi qatorni qo'shishingiz kerak (satr import prime qilishdan oldin).

```
import sys

if 'C:\\MyPythonModules' not in sys.path:
sys.path.append('C:\\MyPythonModules')
```

sys.path sizning Python tizim yo'lingizga ishora qiladi. Bu Python tomonidan modullar va fayllarni izlash uchun kataloglar ro'yxati. Yuqoridagi kod "C: \ MyPythonModules" papkasini tizim yo'lingizga qo'shib qo'yadi.

Endi prime.py-ni C:\MyPythonModules ga va checkIfPrime.py-ni siz tanlagan boshqa papkaga qo'yishingiz mumkin.

8-bob: Fayllar bilan ishlash

Ajoyib! Loyihadan oldin biz kitobning so'nggi bobiga keldik. Ushbu bobda biz tashqi fayllar bilan ishlashni ko'rib chiqamiz.

Oldin 5-bobda biz `input ()` funksiyasi yordamida foydalanuvchilardan qanday qilib ma'lumot olishni o'rgangan edik. Ammo, ba'zi hollarda, foydalanuvchilarni bizning dasturimizga ma'lumotlarni kiritishlari, amaliy bo'lishi mumkin emas, ayniqsa bizning dasturimiz katta hajmdagi ma'lumotlar bilan ishlashga muhtoj bo'lsa. Bunday hollarda, kerakli ma'lumotni tashqi fayl sifatida tayyorlash va dasturlarimizga ma'lumotni fayldan o'qish uchun qulayroq usul. Ushbu bobda buni bajarishni o'rganamiz. Tayyormisiz?

Matnli fayllarni ochish va o'qish

Biz o'qiydigan birinchi turdagi fayl - bu bir necha qatorli oddiy matnli fayl. Buning uchun avval quyidagi satrlar bilan matnli fayl yarataylik.

Pythonni bir kunda o'rganing va uni yaxshilab o'rganing Python - yangi boshlanuvchilar uchun amaliy loyiha Python-da darhol kodlashni boshlashingiz kerak bo'lgan yagona kitobhttp://www.learncodingfast.com/python

Ushbu matnli faylni `myfile.txt` sifatida ish stoliga saqlang. Keyin, IDLE-ni yoqing va quyidagi kodni kiriting. Ushbu kodni `fileOperation.py` sifatida ish stolingizga saqlang.

```
f = open ('myfile.txt', 'r')

firstline = f.readline()
secondline = f.readline()
print (firstline)
print (secondline)

f.close()
```

Kodning birinchi satri faylni ochadi. Har qanday fayldan o'qishdan oldin biz uni ochishimiz kerak (xuddi o'qish uchun ushbu elektron kitobni yoqish moslamasi yoki ilovasida ochishingiz kerak bo'lganidek). `open ()`

funksiyasi buni amalga oshiradi va ikkita parametrni talab qiladi:

Birinchi parametr - bu faylga yo'l. Agar `fileOperation.py` va `myfile.txt` fayllarini bir xil papkada saqlamagan bo'lsangiz (bu holda ish stoli(desktop), siz `'myfile.txt'` ni matnli faylni saqlagan haqiqiy yo'l bilan almashtirishingiz kerak. Masalan, agar siz uni C diskida "PythonFiles" nomli papkada saqlagan bo'lsangiz, `'C:\\PythonFiles\\myfile.txt'` (ikki tomonlama teskari chiziq bilan \\) yozishingiz kerak.

Ikkinchi parametr - bu rejim. Bu fayl qanday ishlatilishini belgilaydi. Odatda ishlatiladigan rejimlar

'r' rejimi:
Faqat o'qish uchun.

'w' rejimi:
Faqat yozish uchun.
Agar ko'rsatilgan fayl mavjud bo'lmasa, u yaratiladi.
Agar ko'rsatilgan fayl mavjud bo'lsa, fayldagi mavjud ma'lumotlar o'chib ketadi.

'a' rejimi:
Qo'shish uchun.
Agar ko'rsatilgan fayl mavjud bo'lmasa, u yaratiladi.
Agar ko'rsatilgan fayl mavjud bo'lsa, faylga yozilgan har qanday ma'lumotlar avtomatik ravishda oxirigacha qo'shiladi

'r +' rejimi:
Ham o'qish, ham yozish uchun.

Faylni ochgandan so'ng, keyingi qator birinchi `firstline = f.readline()` fayldagi birinchi qatorni o'qiydi va uni birinchi qator(`firstline`) o'zgaruvchisiga beradi.

`readline()` funksiyasi har safar chaqirilganda, u fayldan yangi qatorni o'qiydi. Bizning dasturimizda `readline()` ikki marta chaqirildi. Shuning uchun dastlabki ikkita satr o'qiladi. Dasturni ishga tushirganingizda, quyidagi natijani olasiz:

```
Learn Python in One Day and Learn It Well

Python for Beginners with Hands-on Project
```

Har bir satrdan keyin chiziq oralig'i kiritilganligini sezasiz. Buning sababi, `readline()` funksiyasi har bir satr oxiriga "\ n" belgilarini qo'shib qo'yadi. Agar har bir satr satri orasidagi qo'shimcha chiziqni xohlamasangiz, `print(firstline, end = '')` ni bajarishingiz mumkin. Bu '\ n' belgilarini olib tashlaydi. Dastlabki ikki qatorni o'qigandan va bosib chiqargandan so'ng, `f.close()` oxirgi jumlasi faylni yopadi. Tizimning barcha manbalarini bo'shatish uchun o'qishni tugatgandan so'ng faylni har doim yopishingiz kerak.

Matnli fayllarni o'qish uchun For loopdan foydalanish

Matnli faylni o'qish uchun yuqoridagi `readline()` usulidan tashqari for `for` loopidan ham foydalanishimiz mumkin. Aslida `for` loop matnli fayllarni o'qishning yanada oqlangan va samarali usuli hisoblanadi. Quyidagi dastur bu qanday amalga oshirilishini ko'rsatadi.

```
f = open ('myfile.txt', 'r')

for line in f:
    print (line, end = '')

f.close()
```

`for` loop matn satrlari qatoridan satrga ko'chiriladi. Siz uni ishga tushirganingizda, quyidagicha natija olasiz

```
Learn Python in One Day and Learn It Well
Python for Beginners with Hands-on Project
The only book you need to start coding
in Python immediately http://www.
learncodingfast.com/python
```

Matnli faylga yozish

Endi faylni qanday ochish va o'qishni bilib oldik, keling, unga yozishga harakat qilaylik. Buning uchun biz "a" (append) rejimidan foydalanamiz. Siz "w" rejimidan ham foydalanishingiz mumkin, ammo fayl allaqachon mavjud bo'lsa, fayldagi barcha oldingi tarkibni o'chirib tashlaysiz. Quyidagi dasturni ishlatib ko'ring.

```
f = open ('myfile.txt', 'a')

f.write('\nThis sentence will be
appended.')
f.write('\nPython is Fun!')

f.close()
```

Bu yerda `'This sentence will be appended'` ("Bu jumla qo'shiladi.") va `'Python is Fun!'` Ikkita jumlani faylga qo'shish uchun `write` () funksiyasidan foydalanamiz, ularning har biri `'\n'` qochish belgilaridan foydalanganimizdan beri yangi satrda boshlanadi. Siz quyidagicha natija olasiz

```
Learn Python in One Day and Learn It Well
Python for Beginners with Hands-on Project
The only book you need to start coding in
Python immediately
http://www.learncodingfast.com/python
This sentence will be appended.Python is
Fun!
```

Buffer Size orqali matnli fayllarni ochish va o'qish

Ba'zan, dasturimiz juda ko'p xotira resurslaridan foydalanmasligi uchun faylni bufer hajmi(buffer size) bo'yicha o'qishni xohlashimiz mumkin. Buning uchun biz kerakli bufer hajmini belgilashga imkon beradigan `read ()` funksiyasidan (`readline ()` funksiyasi o'rniga) foydalanishimiz mumkin. Quyidagi dasturni sinab ko'ring:

```
inputFile = open ('myfile.txt', 'r')
outputFile = open ('myoutputfile.txt', 'w')


msg = inputFile.read(10)


while len(msg):
    outputFile.write(msg)
    msg = inputFile.read(10)

inputFile.close()
outputFile.close()
```

Dastlab, ta faylmi ochamiz, `inputFile.txt` va `outputFile.txt` mos ravishda o'qish va yozish uchun fayllar.

So'ngra, `msg = inputFile.read(10)` iborasi va `while` loop yordamida birdaniga 10 baytlik faylni aylanib

o'tamiz. Qavs ichidagi 10 `read()` funksiyasi faqat 10 bayt o'qishini aytadi.
`while` holati `while len(msg):msg` o'zgaruvchining uzunligini tekshiradi. Uzunlik nolga teng emas ekan, loop ishlaydi.

`while` loop ichida `outputFile.write(msg)` iborasi xabarni chiqish fayliga yozadi. Xabarni yozgandan so'ng `msg = inputFile.read(10)` iborasi keyingi 10 baytni o'qiydi va uni butun fayl o'qilguncha davom ettiradi. Bu sodir bo'lganda, dastur ikkala faylni yopadi.

Dasturni ishga tushirishda yangi `myoutputfile.txt` fayli yaratiladi. Faylni ochganingizda, u sizning kiritilgan faylingiz bilan bir xil tarkibga ega ekanligini sezasiz `myfile.txt`. Bir vaqtning o'zida atigi 10 bayt o'qilishini isbotlash uchun dasturdagi `outputFile.write(msg)` qatorini `outputFile.write(msg + '\n')` ga o'zgartirishingiz mumkin. Endi dasturni qayta ishga tushiring. `myoutputfile.txt` endi eng ko'p 10 ta belgidan iborat qatorlarni o'z ichiga oladi. Bu yerda siz oladigan segmentlarning bir qismi.

```
Learn Python in One Day and Learn It Wel
```

Ikkilik(Binary) fayllarni ochish, o'qish va yozish

Ikkilik fayllar matnli bo'lmagan har qanday faylni, masalan, rasm yoki videofayllarni anglatadi. Ikkilik fayllar bilan ishlash uchun biz shunchaki "rb" yoki "wb" rejimidan foydalanamiz. Jpeg faylini ish stolingizga nusxalash va uning nomini `myimage.jpg` deb o'zgartiring. Endi dastlabki ikkita satr satrini o'zgartirib, yuqoridagi dasturni tahrirlang

```
inputFile = open ('myfile.txt', 'r')
outputFile = open ('myoutputfile.txt', 'w')

inputFile = open ('myimage.jpg', 'rb')
outputFile = open ('myoutputimage.jpg',
'wb')
```

`outputFile.write(msg + '\n')` iborasini `outputFile.write(msg)` ga qaytarib o'zgartirganingizga ishonch hosil qiling.

Yangi dasturni ishga tushiring. Ish stolingizda `myoutputimage.jpg` nomli qo'shimcha rasm fayli bo'lishi kerak. Rasm faylini ochganingizda, u `myimage.jpg` ga o'xshash bo'lishi kerak.

Fayllarni o'zgartirish va qayta nomlash

Fayllar bilan ishlashda yana ikkita foydali funksiya - `remove()` (olib tashlash) va `rename()` (qayta nomlash) funksiyalari. Ushbu funksiyalar `os` modulida mavjud va ularni ishlatishdan oldin ularni import qilish kerak.

`remove()` funksiyasi faylni o'chiradi. sintaksis(gap tuzilishi) `remove(filename)`. Misol uchun, `myfile.txt` ni o'chirish uchun, `remove('myfile.txt')` deb yozamiz.

`rename()` funksiyasi faylni qayta nomlaydi. Sintaksis `rename (old name, new name)`. `oldfile.txt` ni `newfile.txt`ga qayta nomlash uchun, `rename('oldfile.txt', 'newfile.txt')` ni yozamiz.

Loyiha: Matematik va BODMAS

Tabriklaymiz! Endi biz birinchi to'liq dasturni kodlashni boshlash uchun Python (va umumiy dasturlash) asoslarini ko'rib chiqdik. Ushbu bobda biz BODMAS arifmetik hisoblash qoidasini tushunishimizni tekshiradigan dasturni kodlashtiramiz. Agar BODMAS nima ekanligini bilmasangiz, ushbu sayt http://www.mathsisfun.com/operation-order-bodmas.html bilan tanishishingiz mumkin.

Bizning dasturimiz tasodifiy javob uchun arifmetik savolni belgilaydi. Agar biz javobni noto'g'ri qabul qilsak, dastur to'g'ri javobni ko'rsatadi va yangi savolni sinab ko'rishni xohlaysizmi, deb so'raydi. Agar biz buni to'g'ri deb bilsak, dastur bizni maqtaydi va yangi savolni xohlaysizmi, deb so'raydi. Bundan tashqari, dastur bizning ballarimizni kuzatib boradi va ballarni tashqi matnli faylga saqlaydi. Har bir savoldan so'ng dasturni tugatish uchun "-1" tugmachasini bosishimiz mumkin.

Siz dasturni o'zingiz kodlashni sinab ko'rishingiz uchun dasturni kichik mashqlarga ajratib qo'ydim. Javoblarga murojaat qilishdan oldin mashqlarni sinab ko'ring. Javoblar E ilovasida keltirilgan yoki Python fayllarini yuklab olish uchun http://www.learncodingfast.com/python saytiga kirishingiz mumkin. Sizni manba kodini yuklab olishingizni qat'iyan iltimos qilaman, chunki E ilovasidagi formatlash kodni o'qishni qiyinlashtiradigan ba'zi bir chuqurlikning buzilishiga olib kelishi mumkin.

Python sintaksisini o'rganish oson, ammo zerikarli ekanligini unutmang. Muammolarni hal qilish - bu yerda qiziqarli narsalar yotadi. Agar ushbu mashqlarni bajarishda qiyinchiliklarga duch kelsangiz, ko'proq harakat qiling. Bu yerda mukofot eng katta hisoblanadi.

Tayyormisiz? Qani ketdik!

1-qism: myPythonFunctions.py

Biz dasturlarimiz uchun ikkita fayl yozamiz. Birinchi fayl `myPythonFunctions.py`, ikkinchisi `mathGame.py`. 1-qism `myPythonFunctions.py` uchun kod yozishga qaratilgan.

Boshlash uchun avval `myPythonFunctions.py` faylini yarataylik. Ushbu faylda uchta funksiyani aniqlaymiz.

1-mashq: Modullarni import qilish

`myPythonFunctions.py` uchun ikkita modulni import qilishimiz kerak: `random` modul va `os` modul.

We'll be using the `randint()` function from the `random` module.The `randint()` function generates a random integer within the range provided by us. We'll use that to generate numbers for our questions later.
Biz `random` moduldan `randint` () funksiyasidan foydalanamiz. `Randint` () funksiyasi biz taqdim etgan oraliqda tasodifiy butun sonni hosil qiladi. Keyinchalik savollarimiz uchun raqamlarni yaratish uchun foydalanamiz.

`os` modulidan `remove` () va `rename` () funksiyalaridan foydalanamiz.

Ushbu ikkita modulni import qilib ko'ring.

2-mashq: Foydalanuvchi ballarini olish

Bu erda biz birinchi funksiyamizni aniqlaymiz. Keling, uni `getUserPoint ()` deb ataymiz. Ushbu funktsiya bitta parametrni, `userName`-ni qabul qiladi. Keyin "`userScores.txt`" faylini "r" rejimida ochadi. `userScores.txt` quyidagicha ko'rinadi:

```
Ann, 100
Benny, 102
Carol, 214
Darren, 129
```

Har bir satrda bitta foydalanuvchining ma'lumotlari yozib olinadi. Birinchi qiymat - foydalanuvchining foydalanuvchi nomi, ikkinchisi - foydalanuvchi balidir.

Keyingisi, funksiya `for` loop yordamida fayl satrini satrlar bilan o'qiydi. Keyin har bir satr `split ()` funksiyasi yordamida bo'linadi (`split ()` funksiyadan foydalanishga misol uchun A ilovaga qarang).

`split ()` funksiyasi natijalarini ro'yxat `content` (tarkib) ida saqlaylik.

Keyinchalik, funksiya har qanday satrda parametr sifatida kiritilgan qiymat bilan bir xil foydalanuvchi nomiga ega yoki yo'qligini tekshiradi. Agar mavjud bo'lsa, funksiya faylni yopadi va foydalanuvchi nomi yonidagi natijani qaytaradi. Agar yo'q bo'lsa, funksiya

faylni yopadi va "-1" qatorini qaytaradi.

Hozircha aniqmi? Funksiyani kodlashni harakat qilib ko'ring. Bajarildimi?
Endi bizning kodimizga ba'zi o'zgartirishlar kiritishimiz kerak. Oldinroq faylimizni ochishda biz "r" rejimidan foydalandik. Bu fayldagi tasodifiy o'zgarishlarning oldini olishga yordam beradi. Biroq, "r" rejimida faylni ochishda, agar fayl allaqachon mavjud bo'lmasa, `IOError` paydo bo'ladi. Dasturni birinchi marta ishga tushirganimizda, xato bilan yakun topamiz, chunki `userScores.txt` fayli ilgari mavjud emas edi.
Ushbu xatoning oldini olish uchun biz quyidagilardan birini bajaramiz:

Faylni "r" rejimida ochish o'rniga "w" rejimida ochishimiz mumkin. "W" rejimida ochilayotganda, agar fayl ilgari mavjud bo'lmasa, yangi fayl yaratiladi. Ushbu usul bilan bog'liq xavf, biz tasodifan faylga yozishimiz mumkin, natijada barcha oldingi tarkib o'chiriladi.
Ammo, bizning dasturimiz kichik dastur bo'lgani uchun, biz tasodifiy yozishni oldini olish uchun kodimizni sinchkovlik bilan tekshirib ko'rishimiz mumkin.

Ikkinchi usul - `IOError`-ni boshqarish uchun `try, except` iborasidan foydalanish. Buni amalga oshirish uchun avvalgi barcha kodlarimizni `try` blokiga qo'yishimiz kerak, keyin `except IOError`-dan foydalaning: "Fayl topilmadi" xatosini boshqarish uchun. Qolgan blokda biz foydalanuvchilarga fayl topilmagani to'g'risida xabar beramiz va keyin faylni yaratishga

kirishamiz. Uni yaratish uchun `open ()` funksiyasini "w" rejimi bilan ishlatamiz. Bu yerda farq shundaki, biz "w" rejimidan faqat fayl topilmaganda foydalanamiz. Fayl dastlab mavjud bo'lmaganligi sababli, avvalgi tarkibni o'chirish xavfi yo'q. Faylni yaratgandan so'ng, faylni yoping va "-1" qatorini qaytaring.

Ushbu mashqni bajarish uchun yuqoridagi usullardan birini tanlashingiz mumkin. Taqdim etilgan javob ikkinchi usuldan foydalanadi. Tugatgandan so'ng, 3-mashqqa o'tamiz.

3-mashq: Foydalanuvchi (score)balini yangilash

Ushbu mashqda biz uchta parametrni o'z ichiga olgan `updateUserPoints()` deb nomlangan yana bir funksiyani aniqlaymiz: `newUser`, `userName` va `score`.

`newUser` `True` va `False` bo'lishi mumkin. Agar `newUser(yangi foydalanuvchi)` `True` bo'lsa, funksiya qo'shish rejimida `userScores.txt` faylini ochadi va foydalanuvchining `userName` va `score (ball)` ini o'yindan chiqqanida faylga qo'shadi.

`newUser` `False` bo'lsa, funksiya foydalanuvchi faylidagi balini yangilaydi. Biroq, Python-da (yoki bu uchun dasturlash tillarining ko'pchiligida) bizga matnli faylni yangilashga imkon beradigan hech qanday funksiya mavjud emas. Biz unga faqat yozishimiz yoki qo'shishimiz mumkin, lekin uni yangilamaymiz.

Shuning uchun biz vaqtinchalik faylni yaratishimiz kerak. Bu dasturlashda juda keng tarqalgan amaliyotdir. Keling, ushbu faylni `userScores.tmp` deb nomlaymiz va "w" rejimida ochamiz. Endi biz `userScore.txt` orqali aylanishimiz va ma'lumotlar satrini `userScores.tmp`-ga nusxalashimiz kerak. Biroq, nusxalashdan oldin, ushbu satrda `userName` parametr bilan ta'minlangan bilan bir xilligini tekshiramiz. Agar u xuddi shunday bo'lsa, vaqtinchalik faylga yozishdan oldin balni yangi ballga o'zgartiramiz.

Masalan, agar funksiyaga berilgan parametrlar `False`, `'Benny'` va `'158'` (ya'ni `updateUserPoints(False, 'Benny', '158')`) bo'lsa, quyidagi jadval asl `userScores.txt` va yangi `userScores.tmp`.

userScores.txt

```
Ann, 100
Benny, 102
Carol, 214
Darren, 129
```

userScores.tmp

```
Ann, 100
Benny, 158
Carol, 214
Darren, 129
```

`userScore.tmp`-ga yozishni tugatgandan so'ng, ikkala faylni yopamiz va `userScores.txt`-ni o'chirib tashlaymiz. Va nihoyat, `userScores.tmp` nomini `userScores.txt` deb o'zgartiramiz.

Tsuhunarli bo'ldimi? Endi, Kodlashni harakat qilib ko'ring ...

4-mashq: Savollar tuzish

Hozir biz matematik savollarni tuzish ya'ni dasturning eng muhim qismiga keldik. Tayyormisiz?

Savollarni tuzish uchun avval uchta o'zgaruvchini e'lon qilaylik: ikkita ro'yxat va bitta lug'at.

Biz ikkita ro'yxatni `operandList` va `operatorList` deb nomlaymiz.

`operandList` beshta raqamni saqlashi kerak, ularning dastlabki qiymati 0 ga teng. `operatorList` to'rtta qatorni saqlashi kerak, ' ' belgisi bilan ularning boshlang'ich qiymatlari sifatida .

Lug'at 4 juftlikdan iborat bo'lib, lug'at kalitlari sifatida 1 dan 4 gacha bo'lgan butun sonlar, ma'lumotlar sifatida esa "+", "-", "*", "**" mavjud. Keling, ushbu `operatorDict`-ga qo'ng'iroq qilaylik.

[4.1-mashq: `operandList`-ni tasodifiy(random) raqamlar bilan yangilash]

Avval `operandList`-ning dastlabki qiymatlarini `randint` () funksiyasi tomonidan yaratilgan tasodifiy sonlar bilan almashtirishimiz kerak.

`randint()` ikkita parametrni qabul qiladi, `start` va `end` va tasodifiy N sonini qaytaradi, shunda `start` <= N <= `end`.

Masalan, agar `randint` (1, 9) deb nomlansa, u tasodifiy ravishda 1, 2, 3, 4, 5, 6, 7, 8, 9 raqamlaridan butun sonni qaytaradi.

`operandList`ining o'zgaruvchisini tasodifiy raqamlar bilan yangilash uchun biz buni birma-bir bajarishimiz mumkin, chunki `operandList` faqat beshta a'zodan iborat. Biz quyidagicha yozishimiz mumkin

```
operandList[0] = ranint(1,  9)
operandList[1] = ranint(1,  9)
operandList[2] = ranint(1,  9)
operandList[3] = ranint(1,  9)
operandList[4] = ranint(1,  9)
```

Har safar `randint` (1, 9) nomlanganda, u tasodifiy ravishda 1, 2, 3, 4, 5, 6, 7, 8, 9 raqamlaridan butun sonni qaytaradi.

Biroq, bu `operandList`-ni yangilashning eng oqlangan usuli emas. Agar `operandList` 1000 a'zodan iborat bo'lsa, bu qanchalik og'ir bo'lishini tasavvur qiling. Yaxshi alternativi(muqobili) `for` loopdan foydalanishdir.

Xuddi shu vazifani bajarish uchun `for` loop dan foydalanib ko'ring.
Bajarildimi? Ajoyib!

[4.2-mashq: `operatorList`-ni matematik belgilar bilan yangilash]
Endi bizda ishlash uchun raqamlar mavjud bo'lib, savollarimiz uchun tasodifiy (+, -, *, **) matematik belgilarni yaratishimiz kerak. Buning uchun biz `randint()` funksiyasidan va `operatorDict` lug'atdan foydalanamiz.

`randint ()` lug'at tugmachasini yaratadi, so'ngra `operatorDict` lug'ati yordamida to'g'ri operatorga bog'lanadi. Masalan, `operatorList[0]` ga belgini berish uchun quyidagicha yozamiz

```
operatorList[0] = operatorDict[randint(1,
4)]
```

4.1-mashqga o'xshab, ushbu vazifani bajarish uchun `for` loopidan foydalanishingiz kerak. Biroq, ushbu mashqni 4.1-mashqdan ko'ra qiyinlashtiradigan bitta muammo mavjud.

Eslatib o'tamiz, Python-da ** ko'rsatkich darajasi (ya'ni

2 ** 3 = 2 ^ 3) degan ma'noni anglatadi?

Muammo shundaki, bizda Pythonda ketma-ket ikkita eksponent operatori bo'lganida, masalan, 2**3**2, Python uni (2**3)**2 o'rniga 2**(3**2) deb talqin qiladi. Birinchi holda, 9 ning kuchiga 2 (ya'ni 2^9), ya'ni 512 ga javob beriladi. Ikkinchi holda, 64 ga teng bo'lgan 2 (ya'ni 8^2) kuchiga 8 javob bo'ladi. Shuning uchun biz savol berganimizda 2**3**2 singari, foydalanuvchi (2**3)**2 deb talqin qilsa, javobni noto'g'ri oladi.

Ushbu muammoning oldini olish uchun biz ketma-ket ikkita ** belgini olmaslik uchun kodimizni o'zgartiramiz. Boshqacha qilib aytganda, `operatorList =['+', '-', '**', '**'] ['+', '+', '-', '**']` yaxshi, lekin `operatorList = ['+', '-', '**', '**']` emas.

Ushbu mashq barcha mashqlar orasida eng qiyinidir. Ikkita ketma-ket ** belgining oldini olish uchun yechim topishga harakat qiling. Tugatganingizdan so'ng, biz 4.3-mashqga o'tishingiz mumkin.

Maslahat: Agar siz tiqilib qolsangiz, `for` loop ichida `if` iborasidan foydalanishni o'ylab ko'rishingiz mumkin.

[4.3-mashq: Matematik ifodani yaratish]

Endi bizda operatorlar va operandlar mavjud bo'lib, biz matematik ifodani string sifatida yaratishga harakat qilamiz. Ushbu ibora savol berish uchun `operandList`-

dan beshta raqamni va `operatorList`-dan to'rtta matematik belgini ishlatadi.

`questionString` deb nomlangan yana bir o'zgaruvchini e'lon qilishimiz va `questionString`-ga matematik ifodani tayinlashimiz kerak.
`questionString` misollariga quyidagilar kiradi

```
6 – 2*3 – 2**1
4 + 5 – 2*6 + 1
8 – 0*2 + 5 – 8
```

Ushbu iborani o'zingiz yaratishga harakat qiling.

Maslahat: Siz matematik ifodani olish uchun `operandList` va `operatorList`-dan alohida satrlarni birlashtirish uchun `for` loopidan foydalanishingiz mumkin.

[4.4-mashq: Natijani baholash]

Endi biz matematik ifodani string sifatida o'zgartiramiz, o'zgaruvchiga `questionString` berilgan. Ushbu iboraning natijasini baholash uchun biz Python, `eval()` bilan birga keltirilgan ajoyib ichki funksiyadan foydalanamiz.

`eval()` string-ni kod sifatida izohlaydi va kodni bajaradi. Masalan, agar `eval("1+2+4")` yozsak, biz 7 raqamini olamiz.

Shuning uchun matematik ifodamiz natijasini baholash uchun `questionString`-ni `eval()` funksiyasiga o'tkazamiz va natijani `result` deb nomlangan yangi o'zgaruvchiga beramiz.

Ushbu mashq oldinga siljiydi va uni bir qadamda bajarish mumkin.

[4.5-mashq: Foydalanuvchi bilan o'zaro aloqalar]

Nihoyat, biz foydalanuvchimiz bilan o'zaro aloqada bo'lamiz. Ushbu mashqda biz bir nechta amallarni bajaramiz:

1-qadam: savolni foydalanuvchiga ko'rsatish
2-qadam: foydalanuvchini javob uchun rag'batlantirish
3-qadam: Javobni baholash, tegishli xabarni ko'rsatish va foydalanuvchi balini qaytarish.

1-qadam uchun biz (string)satrlarni boshqarish uchun o'rnatilgan funksiyadan foydalanishimiz kerak. Yuqorida aytib o'tganimizdek, Python-da ** belgisi yuqori darajani bildiradi. Ya'ni, 2**3 = 8. Biroq, ko'pchilik foydalanuvchilar uchun ** ma'nosi yo'q. Shuning uchun savolni 2**3 + 8 -5 deb ko'rsatadigan bo'lsak, foydalanuvchi chalkashib ketishi mumkin. Buning oldini olish uchun `questionString`-dagi har qanday ** belgini ^ belgisiga almashtiramiz.

Buning uchun `replace ()` funksiyasidan foydalanamiz. Undan foydalanish juda oddiy,

shunchaki `questionString = questionString.replace("**", "^")`ni yozing

Endi siz paydo bo'lgan ifodani foydalanuvchiga chop etishingiz mumkin.

2-qadam uchun foydalanuvchi kiritilishini qabul qilish uchun `input()` funksiyasidan foydalanishingiz mumkin.

3-qadam uchun javobni baholash va to'g'ri xabarni ko'rsatish uchun `if` iborasidan foydalanishingiz kerak. Agar foydalanuvchi buni to'g'ri qabul qilsa, biz foydalanuvchini maqtaymiz va 1 qiymatini qaytaramiz. Agar foydalanuvchi noto'g'ri tushunsa, biz to'g'ri javobni ko'rsatamiz va 0 qiymatini qaytaramiz.

`input()` funksiyasi foydalanuvchi kirishini string sifatida qaytarishini esladingizmi? Demak, foydalanuvchi kiritgan ma'lumotni to'g'ri javob bilan solishtirganda (4.4-mashqda olingan), foydalanuvchi kirishini integer-ga o'zgartirish uchun ba'zi turdagi kastingni bajarishingiz kerak. Foydalanuvchi kirishini butun songa o'zgartirganda, foydalanuvchi raqamni yozganligini tekshirish uchun `try, except` iborasidan foydalanib ko'rishingiz kerak. Agar foydalanuvchi o'rniga qatorni yozgan bo'lsa, dastur foydalanuvchini xato haqida xabardor qilishi va foydalanuvchidan raqam yozishni taklif qilishi kerak.

Agar foydalanuvchi uni bajarmasa, raqamni so'rashni

davom ettirish uchun `while True` loopdan foydalanishingiz mumkin. `while True` yozish `while 1==1` kabi yozishga tengdir. 1 har doim 1 ga teng bo'lgani uchun (shuning uchun har doim `True`), loop cheksiz ishlaydi.

Bu yerda qanday qilib ushbu mashq uchun `while True` loop-dan foydalanishingiz mumkinligi haqida taklif mavjud.

```
while True:
try:
    cast user's answer to an integer and
    evaluate the answer
    return user score based on the answer
except:
    print error message if casting fails
    prompt user to key in the answer again
```

`while` sharti har doim `True` (`to'g'ri`) bo'lgani uchun `while True` loop halqalanishni davom ettiradi. Loop faqat `try` bloki to'g'ri bajarilganda va `return` operatoriga yetganda chiqadi.

Ushbu mashqni bajarib ko'ring. Tugatgandan so'ng, biz haqiqiy dasturni yozadigan 2-qismga o'tishimiz mumkin.

2-qism: mathGame.py

1-qismni bajarganingiz uchun tabriklayman va 2-qismga xush kelibsiz. 2-qism "yengil shabada" bo'lib qoladi, shunchaki biz asosan avval belgilagan funksiyalarni chaqiramiz.

5-mashq: Asosiy dasturni yozish

Birinchidan, keling, bayonotdan tashqari asosiy dasturimizni sinab ko'raylik. Biz asosiy dasturni ishga tushirishda kutilmagan xatolarga yo'l qo'yishni xohlaymiz.

`try` (sinov) bloki uchun kod yozishdan boshlaymiz.

Birinchidan, biz `myPythonFunctions` modulini import qilishimiz kerak. Keyin, foydalanuvchidan foydalanuvchi nomini so'raylik va o'zgaruvchi `userName`-iga qiymat belgilaymiz. Ushbu o'zgaruvchini `getUserScore()` funksiyasiga parametr sifatida o'tkazing.

`getUserScore()` foydalanuvchining natijasini qaytaradi yoki qaytadi
"-1" (agar foydalanuvchi topilmasa). Keling, ushbu natijani butun songa tashlaymiz va `userScore` o'zgaruvchisiga tayinlaymiz.

Endi yana bir o'zgaruvchining qiymatini o'rnatishimiz kerak `newUser`. Agar foydalanuvchi topilmasa,

`newUser = True`, aks holda `newUser = False`. Agar `newUser = True` bo'lsa, biz `userScore` -ni -1 dan 0 ga almashtirishimiz kerak.

Dasturimizning keyingi qismi `while` loopni o'z ichiga oladi. Xususan, bizning dasturimiz foydalanuvchidan dasturni tugatishi yoki boshqa biron bir ishni bajarishi kerakligini aniqlash uchun so'rov yuboradi.

1-qadam:
Siz yana bir o'zgaruvchi `userChoice`ni e'lon qilishingiz va unga boshlang'ich qiymatini 0 berishingiz kerak.

2-qadam:
Keyin, `while` loopidan foydalanib, `userChoice`ni o'zingiz tanlagan qator bilan taqqoslang, "-1" deb ayting. Agar `userChoice` "-1" bilan bir xil bo'lmasa, yangi savol yaratish uchun `generateQuestion()` funksiyasini chaqiring.

3-qadam:
`generateQuestion()` foydalanuvchi ushbu savol uchun olgan balini qaytaradi. `userScore` o'zgaruvchisini yangilash uchun ushbu natijadan foydalaning.

4-qadam:
Va nihoyat, cheksiz loopning oldini olish uchun biz `input()` funksiyasini while loop ichida foydalanuvchi kirishini qabul qilish va `userChoice` qiymatini yangilash uchun ishlatishimiz kerak.

Tushundingizmi? Kodlashni harakat qilib ko'ring. Haqiqiy kodlashni bajarish hamma narsani aniqroq qiladi.

Nihoyat, while loop tugagandan so'ng, keyingi bosqich `userScores.txt` faylini yangilashdir. Buning uchun shunchaki `updateUserPoints()` funksiyasiga murojaat qilamiz.

Bularning barchasi `try` bloki uchun. Endi bundan `except` blok uchun biz foydalanuvchiga xato yuz berganligi va dastur tugashi haqida xabar beramiz.

Bo'ldi shu! Ushbu qadamni tugatgandan so'ng sizda to'liq dastur, Pythondagi birinchi dastur bo'ladi. Dasturni ishlatib ko'ring

`mathGame.py`. Kutilganidek ishladimi? Xursandmisiz? Ishonamanki, bundan men kabi hayajonlandingiz.:)

O'zingizni sinang

Biz ushbu bobning oxiriga keldik va umid qilamizki siz birinchi dasturni muvaffaqiyatli kodladingiz. Agar biron bir mashqni bajarishda muammolarga duch kelsangiz, javoblarni E ilovada o'rganishingiz mumkin. Boshqa odamlarning kodlarini o'rganib ko'p narsalarni bilib olasiz.
Ushbu bo'limda o'zingizni sinashingiz uchun uchta qo'shimcha mashqlar mavjud.

1-sinov mashqi
Hozirgacha kodlagan dasturda men bo'linish operatoridan foydalanishdan qochdim. Dasturni bo'linish belgisi bilan ham savollar tug'diradigan qilib o'zgartira olasizmi? Qanday qilib foydalanuvchining javobini to'g'ri javob bilan tekshirasiz?

Maslahat: `round()` funksiyasini tekshiring.

2-sinov mashqi

Ba'zida hosil bo'lgan savol juda katta yoki juda kichik javobga olib kelishi mumkin. Masalan, 6*(8^9/1)^3 savoliga 145071098537555009647411 2 javob beriladi.

Foydalanuvchilar uchun juda ko'p sonni hisoblash va kiritish juda noqulay. Shuning uchun biz juda katta yoki kichik javoblardan qochishni istaymiz. Javoblari 50 000 dan katta yoki -50000 dan kichik bo'lgan savollarga yo'l qo'ymaslik uchun dasturni o'zgartira olasizmi?

3-sinov mashqi

Oxirgi mashqlar eng qiyin.

Hozircha berilgan savollarda qavslar yetishmayapti. Savollar ham qavslardan foydalanishi uchun dasturni o'zgartira olasizmi? Savolga misol 2 + (3 * 7 -1) + 5 bo'ladi.

Ushbu mashqlar bilan zavqlaning. Tavsiya etilgan yechim E ilovasida keltirilgan.

Rahmat sizga

Biz kitobning oxiriga keldik. Ushbu kitobni o'qiganingiz uchun tashakkur va kitob sizga yoqdi degan umiddamiz. Eng muhimi, ushbu kitob sizga Python dasturlash asoslarini o'zlashtirishda yordam berganiga chin dildan umid qilamiz.

Bilaman, siz Python dasturlash bo'yicha o'nlab kitoblardan birini tanlashingiz mumkin edi, ammo siz ushbu kitob bilan imkoniyat topdingiz. Ushbu kitobni yuklab olib, oxirigacha o'qiganingiz uchun yana bir bor rahmat.
Iltimos, mashq va topshiriqlarni sinab ko'ring. Siz ko'p narsalarni bilib olasiz.

Endi "kichik" yaxshilikni so'ramoqchimiz. Iltimos, Amazonda ushbu kitob uchun sharh qoldirish uchun bir necha daqiqa yoki ikki daqiqa vaqt ajrata olasizmi?

Ushbu mulohaza bizga juda katta yordam beradi va dasturlash bo'yicha qo'shimcha qo'llanmalar yozishni davom ettiradi. Agar sizga kitob yoqsa yoki uni yaxshilash bo'yicha takliflaringiz bo'lsa, iltimos, bizga xabar bering. Biz juda minnatdormiz. :)

Va nihoyat, esda tutingki, loyiha uchun manba kodini va qo'shimchalarni http://www.learncodingfast.com/python saytidan yuklab olishingiz mumkin.

Shuningdek, men bilan jamie@learncodingfast.com elektron manzili orqali bog'lanishingiz mumkin.

Ilova A: Stringlar bilan ishlash

Izoh: yozuv [start, [end]] *start* va *end* degani ixtiyoriy parametrlar. Agar parametr sifatida faqat bitta raqam berilgan bo'lsa, u *start* uchun qabul qilinadi.

sharhning boshlanishini belgilaydi
''' Ko'p satrli sharhning boshi va oxirini belgilaydi. Haqiqiy kod `monotype` (monotip) shriftda.
=> chiqish boshlanishini belgilaydi

`count (sub, [start, [end]])`

Substring *sub* satrda necha marta paydo bo'lganligini qaytaring. Bu funksiya (case sensitive)harflar katta-kichiklikka sezgirdir.

[Misol]

Quyidagi misollarda "s" 3, 6 va 10 indekslarida uchraydi

butun string-ni hisoblang
```
'This is a string'.count('s')
```
=> 3

4-indeksdan stringning oxirigacha hisoblang
```
'This is a string'.count('s', 4)
```
=> 2

4-indeksdan 10-1 gacha hisoblang

```
'This is a string'.count('s', 4, 10 )
```
=> 1
count 'T'. There's only 1 'T' as the function is case sensitive.
"T" ni hisoblash. Bu yerda faqat 1"T" mavjud, chunki funksiya (case sensitive)katta-kichiklikka sezgir.
```
'This is a string'.count('T')
```
=> 1

endswith (suffix, [start, [end]])

Agar string belgilangan *suffix(*qo'shimcha) bilan tugagan bo'lsa, True-ga qayting, aks holda False-ni qaytaring. *suffix*, shuningdek, izlash uchun qo`shimchalarning katakchasi bo`lishi mumkin. Ushbu funksiya (case sensitive)katta-kichiklikka sezgir.

[Misol]

'man' 4 dan 6 gacha bo'lgan ko'rsatkichlarda uchraydi

butun stringni tekshiring
```
'Postman'.endswith('man')
```
=> True

3-indeksdan stringning oxirigacha tekshiring
```
'Postman'.endswith('man', 3)
```
=> True

2-indeksdan 6-1gacha teskhiring
```
'Postman'.endswith('man', 2, 6)
```

=> False

2-indeksdan 7-1gacha tekshiring
```
'Postman'.endswith('man', 2, 7)
```
=> True

#Qo'shimchalarning katakchasidan foydalanish (2-indeksdan 6-1 gacha tekshirish)
```
'Postman'.endswith(('man', 'ma'), 2, 6)
```
=> True

find/index (sub, [start, [end]])

Substring *sub* birinchi hodisa topilgan satrda indeksni qaytaring.
find() returns -1 if *sub* is not found.
index() returns ValueError is *sub* is not found. This function is case-sensitive.
find () *sub* topilmasa -1 qaytaradi.
index () return ValueError is *sub* topilmadi. Ushbu funksiya katta-kichiklikka sezgir.

[Misol]

butun stringni tekshiring
```
'This is a string'.find('s')
```
=> 3

4-indeksdan stringning oxirigacha tekshiring
```
'This is a string'.find('s', 4)
```

=> 6

7-indeksdan 11-1gacha tekshiring
```
'This is a string'.find('s', 7,11 )
```
=> 10

Sub topilmadi
```
'This is a string'.find('p')
```
=> -1

```
'This is a string'.index('p')
```
=> ValueError

isalnum()

Agar satrdagi barcha belgilar alfanumerik bo'lsa va kamida bitta belgi bo'lsa, aks holda false bo'lsa, haqiqiy qiymatni qaytaring.
Alfanumerik bo'sh joylarni o'z ichiga olmaydi.

[Misol]
```
'abcd1234'.isalnum()
```
=> True

```
'a b c d 1 2 3 4'.isalnum()
```
=> False

```
'abcd'.isalnum()
```
=> True

```
'1234'.isalnum()
```

=> True

isalpha()

Agar satrdagi barcha belgilar alfavitga ega bo'lsa va kamida bitta belgi bo'lsa, aks holda false.

[Misol]

```
'abcd'.isalpha()
```
=> True
```
'abcd1234'.isalpha()
```
=> False

```
'1234'.isalpha()
```
=> False

```
'a b c'.isalpha()
```
=> False

isdigit()

Agar satrdagi barcha belgilar raqamli bo'lsa va kamida bitta belgi bo'lsa, aks holda false.

[Misol]

```
'1234'.isdigit()
```
=> True

```
'abcd1234'.isdigit()
```
=> False

```
'abcd'.isdigit()
```
=> False

```
'1 2 3 4'.isdigit()
```
=> False

islower()

Agar satrdagi barcha harfli belgilar kichik bo'lsa va kamida bitta harfli belgi bo'lsa, aks holda false.

[Misol]

```
'abcd'.islower()
```
=> True

```
'Abcd'.islower()
```
=> False

```
'ABCD'.islower()
```
=> False

isspace()

Agar satrda faqat bo'sh joy belgilar bo'lsa va kamida bitta belgi bo'lsa, aks holda false bo'lsa, haqiqiy qiymatni

qaytaring.

[Misol]

```
' '.isspace()
```
=> True

```
'a b'.isspace()
```
=> False

istitle()

Agar string saralangan string bo'lsa va kamida bitta belgi bo'lsa, haqiqiy qiymatni qaytaring

[Misol]

```
'This Is A String'.istitle()
```
=> True

```
'This is a string'.istitle()
```
=> False

isupper()

Agar satrdagi barcha harflar katta harf bilan yozilgan bo'lsa va kamida bitta bitta harfli belgi bo'lsa, aks holda false.

[Misol]

```
'ABCD'.isupper()
```
=> True

```
'Abcd'.isupper()
```
=> False

```
'abcd'.isupper()
```
=> False

join()

Taqdim etilgan parametr ajratuvchi bilan birlashtirilgan qatorni qaytaring.

[Misol]

```
sep = '-'
myTuple = ('a', 'b', 'c')
myList = ['d', 'e', 'f']myString = "Hello
World"

sep.join(myTuple)
```
=> 'a-b-c'

```
sep.join(myTuple)
```
=> 'd-e-f'

```
sep.join(myString)
```
=> 'H-e-l-l-o- -W-o-r-l-d"

lower()

Kichik harfga o'girilgan string nusxasini qaytaring.
[Misol]

```
'Hello Python'.lower()
```

=> 'hello python'

replace(old, new[, count])

Stringning nusxasini eskirgan substringning barcha holatlari yangisiga almashtirilgan holda qaytaring. *count (hisoblash)* ixtiyoriy. Agar berilgan bo'lsa, faqat birinchi *count* (hisoblash) hodisalari almashtiriladi. Ushbu funksiya katta-kichiklikka sezgir.

[Misol]

Barcha hodisalarni almashtiring

```
'This is a string'.replace('s', 'p')
```

=> 'Thip ip a ptring'

Dastlabki 2 ta hodisani almashtiring

```
'This is a string'.replace('s', 'p', 2)
```

=> 'Thip ip a string'

split([sep_[,maxsplit]])

Ajratuvchi qator sifatida *sep* yordamida satrdagi so'zlar ro'yxatini qaytaring.
sep va *maxsplit* ixtiyoriy.
Agar *sep* berilmasa, ajratuvchi sifatida bo'sh joy ishlatiladi. Agar *maxsplit* berilgan bo'lsa, ko'pi bilan *maxsplit* bo'linish amalga oshiriladi.
Ushbu funksiya katta-kichiklikka sezgir.

[Misol]
'''
Ajratuvchi sifatida vergul yordamida bo'ling
E'tibor bering, output-da "is", "a" va "string" so'zlaridan oldin bo'sh joy mavjud.
'''

```
'This, is, a, string'.split(',')
```
=> ['This', ' is', ' a', ' string']

Bo'sh joyni ajratuvchi sifatida bo'linish(split)
```
'This is a string'.split()
```
=> ['This', 'is', 'a', 'string']

Faqat 2 ta bo'linish(split)ni bajaring
```
'This, is, a, string'.split(',' 2)
```
=> ['This', ' is', ' a, string']

splitlines ([keepends])

Satr chegaralarini buzgan holda string-dagi qatorlar ro'yxatini qaytaring. Qatlamlar berilmasa va to'g'ri bo'lsa, qatorlarning tanaffuslari natijalar ro'yxatiga kiritilmaydi.

[Misol]

```
# \n bilan ajratilgan chiziqlar
'This is the first line.\nThis is the second
line'.splitlines()
```
=> ['This is the first line.', 'This is the second line.']

```
# Ko'p qatorli string-ni ajratish (masalan, ''' belgisini
ishlatadigan string)
'''This is the first line.
This is the second line.'''.splitlines()
```
=> ['This is the first line.', 'This is the second line.']

```
# Qator uzilishlarini ajratish va ushlab turish
'This is the first line.\nThis is the second
line.'.splitlines(True)
```
=> ['This is the first line.\n', 'This is the second line.']

```
'''This is the first line.
Thisisthesecondline.'''.splitlines(True)
```
=> ['This is the first line.\n', 'This is the second line.']

startswith (prefix[, start[, end]])

Agar satr prefiks bilan boshlangan bo'lsa, True-ga qaytaring, aks holda False-ga qayting.
prefix , shuningdek, qidirish uchun prefikslarning katakchasi bo'lishi mumkin.
Ushbu funksiya katta-kichiklikka sezgir(case-sensitive).

[Misol]

'Post' 0 dan 3 gacha bo'lgan indexlarda uchraydi

butun stringni tekshiring

```
'Postman'.startswith('Post')
```

=> True

3 indeksidan string oxirigacha tekshiring

```
'Postman'.startswith('Post',3)
```

=> False

indeks 2 dan 6-1 gacha tekshiring

```
'Postman'.startswith('Post', 2, 6)
```

=> False

2 indeksidan 6-1 gacha tekshiring

```
'Postman'.startswith('stm', 2, 6)
```

=> True

Prefikslar to'plamidan foydalaning (3 indeksdan satr oxirigacha tekshiring)

```
'Postman'.startswith(('Post', 'tma'), 3)
```

=> True

strip_([chars])

Stringning nusxasini yetakchi va oxirgi belgilar bilan qaytaring
char olib tashlandi.
Agar *char* taqdim etilmasa, bo'sh joylar o'chiriladi. Ushbu funksiya katta-kichiklikka sezgir.

[Misol]

Strip whitespaces(Bo'shliqlarni kesib oling)

```
' This is a string '.strip()
```

=> 'This is a string'

Strip 's'. "S" satrning boshida yoki oxirida bo'lmaganligi sababli hech narsa olib tashlanmaydi

```
'This is a string'.strip('s')
```

=> 'This is a string'

Strip 'g'.

```
'This is a string'.strip('g')
```

=> 'This is a strin'

upper()

Katta harfga o'girilgan string nusxasini qaytaring.
[Misol]

```
'Hello Python'.upper()
```

=> 'HELLO PYTHON'

=> output boshlanishini belgilaydi

ap_pend(.)

Ro’yxatning oxiriga element qo’shing

[Misol]

```
myList = ['a', 'b',    'c',  'd']
myList.append('e')
print (myList)
```

=> [‘a’, ‘b’, ‘c’, ‘d’, ‘e’]

del

Ro’yxatdan elementlarni
olib tashlang

[Misol]

```
myList = ['a', 'b',    'c',  'd',    'e',
'f', 'g', 'h','i',    'j',  'k',    'l']
```

uchinchi elementni o’chiring (indeks = 2)

```
del myList[2] print (myList)
```
=> ['a', 'b', 'd', 'e', 'f', 'g', 'h', 'i', 'j', 'k', 'l']

indeks1 dan 5-1 gacha bo'lgan elementlarni o'chiring

```
del myList[1:5]print (myList)
```
=> ['a', 'g', 'h', 'i', 'j', 'k', 'l']

0 dan 3-1 gacha bo'lgan indekslarni o'chiring
```
del myList [ :3]print (myList)
```
=> ['i', 'j', 'k', 'l']

2-indeksdan oxirigacha elementlarni o'chiring
```
del myList [2:]print (myList)
```
=> ['i', 'j']

extend( )

Ikki ro'yxatni birlashtiring

[Misol]
```
myList = ['a', 'b', 'c', 'd', 'e']myList2 =
[1, 2, 3, 4]
myList.extend(myList2)print (myList)
```
=> ['a', 'b', 'c', 'd', 'e', 1, 2, 3, 4]

In

Ob'ekt ro'yxatda mavjudligini tekshirish

[Misol]

```
myList = ['a', 'b', 'c', 'd']'c' in myList
```
=> True

```
'e' in myList
```
=> False

insert( )

Muayyan pozitsiyada ro'yxatga element qo'shing
[Misol]

```
myList = ['a', 'b', 'c', 'd', 'e']myList.
insert(1, 'Hi')
print (myList)
```
=> ['a', 'Hi', 'b', 'c', 'd', 'e']

len( )

Ro'yxatdagi elementlar sonini toping
[Misol]

```
myList = ['a', 'b', 'c', 'd'] print
(len(myList))
```
=> 4

pop_( )

Ob'ektning qiymatini oling va uni ro'yxatdan olib tashlang
Parametr sifatida element indeksini talab qiladi
[Misol]

```
myList = ['a', 'b', 'c', 'd', 'e']
```

3-elementni olib tashlang
```
member = myList.pop(2)
print (member)
```
=> c

```
print (myList)
```
=> ['a', 'b', 'd', 'e']

#oxirgi elementni olib tashlang
```
member = myList.pop( )print (member)
```
=> e

```
print (myList)
```
=> ['a', 'b', 'd']

remove( )

Ob'ektni ro'yxatdan olib tashlang. Parametr sifatida elementning qiymatini talab qiladi.

[Misol]

```
myList = ['a', 'b', 'c', 'd', 'e']
```

‘c’elementini olib tashlang `myList.remove('c')`
`print (myList)`
=> [‘a’, ‘b’, ‘d’, ‘e’]

`reverse()`

Ro’yxatdagi elementlarni teskari yo’naltiring(reverse)
[Misol]
`myList = [1, 2, 3, 4]myList.reverse() print (myList)`
=> [4, 3, 2, 1]

`sort()`

Ro’yxatni alfavit yoki raqam bo’yicha saralang
[Misol]
`myList = [3, 0, -1, 4, 6]`
`myList.sort()print(myList)`
=> [-1, 0, 3, 4, 6]

`sorted()`

Original ro’yxatni saralashsiz yangi tartiblangan ro’yxatni qaytaring. Parametr sifatida ro’yxatni talab qiladi
[Misol]

```
myList = [3, 0, -1, 4, 6]
myList2 = sorted(myList)
```

#Original ro'yxat saralanmagan

```
print (myList)
```

=> [3, 0, -1, 4, 6]

#Yangi ro'yxat saralangan

```
print (myList2)
```

=> [-1, 0, 3, 4, 6]

Addition Operator: +Ro'yxatni birlashtiring [Misol]

```
myList = ['a', 'b', 'c', 'd']print (myList
+ ['e', 'f'])
```

=> ['a', 'b', 'c', 'd', 'e', 'f']

```
print (myList)
```

=> ['a', 'b', 'c', 'd']

Multiplication Operator: *

Ro'yxatni takrorlang va ro'yxatning oxirigacha birlashtiring

[Misol]

```
myList = ['a', 'b', 'c', 'd']print (myList*3)
```
=> ['a', 'b', 'c', 'd', 'a', 'b', 'c', 'd', 'a', 'b', 'c', 'd']

```
print (myList)
```
=> ['a', 'b', 'c', 'd']

Eslatma:
+ va * belgilar ro'yxatni o'zgartirmaydi. Ro'yxat ikkala holatda ham ['a','b', 'c', 'd'] shaklida qoladi.

Ilova C: Tuplelar bilan ishlash

=> chiqish(output) boshlanishini belgilaydi

del

To'liq oynani o'chirib tashlang
[Misol]

```
myTuple = ('a', 'b', 'c', 'd')del myTuple
print (myTuple)
```

=> NameError: "myTuple" nomi aniqlanmagan

in

Ob'ektning katakchada ekanligini tekshiring
[Misol]

```
myTuple = ('a', 'b', 'c', 'd') 'c' in
myTuple
```

=> True

```
'e' in myTuple
```

=> False

len( )

Tupledagi elementlar sonini toping

[Misol]

```
myTuple = ('a', 'b', 'c', 'd') print
(len(myTuple))
```
=> 4

Addition Operator: + Tuplelarni birlashtiring [Misol]

```
myTuple = ('a', 'b', 'c', 'd') print
(myTuple + ('e', 'f'))
```
=> ('a', 'b', 'c', 'd', 'e', 'f')

```
print (myTuple)
```
=> ('a', 'b', 'c', 'd')

Multiplication Operator: *

Tuplening nusxasini oling va tuplening oxirigacha birlashtiring

[Misol]

```
myTuple = ('a', 'b', 'c', 'd')
print(myTuple*3)
```
=> ('a', 'b', 'c', 'd', 'a', 'b', 'c', 'd', 'a', 'b', 'c', 'd')

```
print (myTuple)
```
=> ('a', 'b', 'c', 'd')

Izoh: + va * belgilari gorizontalni o'zgartirmaydi. Tuple ikkala holatda ham 'a', 'b', 'c', 'd'] shaklida qoladi.

Ilova D: (Lug'at)Dictionary-lar bilan ishlash

=> output boshlanishini belgilaydi

clear( )

Lug'atning barcha elementlarini olib tashlaydi, bo'sh lug'atni qaytaradi

[Misol]

```
dic1 = {1: 'one', 2: 'two'}print (dic1)
```
=> {1: 'one', 2: 'two'}

```
dic1.clear()print (dic1)
```
=> { }

del
Lug'atni to'liq o'chirib tashlang
[Misol]
```
dic1 = {1: 'one', 2: 'two'}del dic1
print (dic1)
```
=> NameError: "dic1" nomi aniqlanmagan

get( )

Berilgan kalit uchun qiymatni qaytaradi.
Agar kalit topilmasa, u None kalit so'zini qaytaradi.
Shu bilan bir qatorda, agar kalit topilmasa, qaytariladigan qiymatni belgilashingiz mumkin.

[Misol]

```
dic1 = {1: 'one', 2: 'two'}dic1.get(1)
```

=> 'one'

```
dic1.get(5)
```

=> None

```
dic1.get(5, "Not Found")
```

=> 'Not Found'

In

Elementning lug'atda mavjudligini tekshiring

[Misol]

```
dic1 = {1: 'one', 2: 'two'}

# kalit asosida
1 in dic1
```

=> True

```
3 in dic1
```

=> False

```
# qiymatga asoslanib
'one' in dic1.values()
```

=> True

```
'three' in dic1.values()
```

=> False

items( )

Lug'at juftliklari ro'yxatini tuplega qaytaradi
[Misol]

```
dic1 = {1: 'one', 2: 'two'}dic1.items()
```

=> dict_items([(1, 'one'), (2, 'two')])

keys( )
Lug'at kalitlari ro'yxatini qaytaradi
[Misol]

```
dic1 = {1: 'one', 2: 'two'}dic1.keys()
```

=> dict_keys([1, 2])

len( )
Lug'atdagi elementlar sonini toping
[Misol]

```
dic1 = {1: 'one', 2: 'two'}print (len(dic1))
```

=> 2

update( )

Bir lug'atning kalit qiymatlari juftligini boshqasiga qo'shadi. Dublikatlar olib tashlandi.

[Misol]

```
dic1 = {1: 'one', 2: 'two'}
dic2 = {1: 'one', 3: 'three'}

dic1.update(dic2)print (dic1)
```

=> {1: 'one', 2: 'two', 3: 'three'}

```
print (dic2) #no change
```

=> {1: 'one', 3: 'three'}

values( )

(Dictionary values)Lug'at qiymatlari ro'yxatini qaytaradi
[Misol]

```
dic1 = {1: 'one', 2: 'two'}dic1.values()
```

=> dict_values(['one', 'two'])

Ilova E: Loyiha javoblari

Mashq 1

```
from random import randint from os import
remove, rename
```

Mashq 2

```
def getUserScore(userName):
    try:
input = open('userScores.txt', 'r') for
line in input:
    content = line.split(',')
        if content[0] == userName:
            input.close()
            return content[1]
input.close()
return "-1"
except IOError:
print ("\nFile userScores.txt not found.
Anew file will be created.")
input = open('userScores.txt', 'w')input.
close()
return "-1"
```

Mashq 3

```
def updateUserPoints(newUser, userName,
score): if newUser:
input = open('userScores.txt', 'a') input.
write('\n' + userName + ', ' +
score)
```

```
input.close()else:
input = open('userScores.txt', 'r') output
= open('userScores.tmp', 'w')
for line in input:
content = line.split(',') if content[0]
== userName:
content[1] = score
line = content[0] + ', ' + content[1]
+ '\n'

output.write(line) input.close() output.
close()

remove('userScores.txt')
rename('userScores.tmp', 'userScores.
txt')
```

Mashq 4

```
def generateQuestion(): operandList = [0,
0, 0, 0, 0] operatorList = ['', '', '', '']
operatorDict = {1:' + ', 2:' - ', 3:'*',
4:'**'}

for index in range(0, 5): operandList[index]
= randint(1, 9)

for index in range(0, 4):
if index > 0 and operatorList[index-1] !=
'**':4)]

3)]
else:
```

```
operator = operatorDict[randint(1,
operator = operatorDict[randint(1,
operatorList[index] = operatorquestionString
=str(operandList[0])

for index in range(1, 5): questionString =
questionString +
operatorList[index-1]                    +
str(operandList[index])      result      =
eval(questionString)
questionString      =      questionString.
replace("**",      "^")

print ('\n' + questionString) userResult =
input('Answer: ')
while True:
try:
if int(userResult) == result:print ("So
Smart") return 1
else:
print ("Sorry, wrong answer.
The correct answer is", result)
return 0 except Exception as e:
print ("You did not enter anumber. Please
try again.")
userResult = input('Answer: ')
```

[4.2-mashq uchun tushuntirish]

operatorList-dagi ikkinchi elementdan (ya'ni indeks = 1) boshlab, if index > 0 and

`operatorList[index-1] != '**'` bo'lsa, chiziq `operatorList`dagi avvalgi element `'**'` belgisi ekanligini tekshiradi.
Agar u bo'lmasa, `operator = operatorDict[randint(1, 4)]` bajaradi. `randint` funksiyasiga berilgan diapazon 1 dan 4 gacha bo'lganligi sababli 1, 2, 3 yoki 4 raqamlari hosil bo'ladi. Demak, '+', '-', '*' yoki '**' belgilar o'zgaruvchan `operator`ga beriladi.

Ammo, avvalgi belgi '**' bo'lsa, else operatori (`operator = operatorDict[randint(1, 3)]`) bajariladi. Bunday holda, `randint` funksiyasiga berilgan diapazon 1 dan 3 gacha.
Demak `operatorDict` -da 4-tugmachaga ega bo'lgan '**' belgisi `operator` o'zgaruvchisiga BERILMAYDI.

Mashq 5

```
try:

import myPythonFunctions as m

userName = input('''Please enter your
user name or
create a new one if this is the first time
you are running the program: ''')


userScore = int(m.getUserScore(userName))


if userScore == -1:
```

```
newUser = True
userScore = 0
else:
newUser = False


userChoice = 0

while userChoice != '-1':
userScore += m.generateQuestion() print
("Current Score = ", userScore)userChoice
= input("Press Enter To
Continue or -1 to Exit: ")

m.updateUserPoints(newUser,     userName,
str(userScore))

except Exception as e:
print ("An unexpected error occurred.
Program will be exited.")
```

O'zingizni sinang

Barcha muammolar uchun faqat `generateQuestion()` **funksiyasini o'zgartirishingiz kerak. Mana taklif qilingan yechim.**

```
def  generateQuestion():
operandList = [0, 0, 0, 0, 0]
operatorList = ['', '', '', '']
operatorDict = {1:' + ', 2:' - ', 3:'*',
```

```
4:'/', 5:'**'}

result = 500001

while result > 50000 or result < -50000:
for index in range(0, 5):
operandList[index] = randint(1, 9)
for index in range(0, 4):

        if index > 0 and
operatorList[index-1] != '**':
        operator =
operatorDict[randint(1, 4)]
        else:
            operator =
operatorDict[randint(1, 5)]
        operatorList[index] = operator

'''
Randomly generate the positions of ( and
)
E.g. If openBracket = 2, the ( symbol will
be placed in front of the third number
If closeBracket = 3, the ) symbol will be
placed behind the fourth number
Since the closing bracket cannot be before
the opening bracket, we have to generate
the position for the closing bracket from
openBracket + 1 onwards
'''
```

```
openBracket = randint(0, 3)
closeBracket = randint(openBracket+1, 4)

if openBracket == 0: questionString = '('
+
str(operandList[0])
else:
questionString =str(operandList[0])

for index in range(1, 5):
if index == openBracket:

questionString = questionString
+ operatorList[index-1] + '(' +
str(operandList[index])
elif index == closeBracket:questionString
=
questionString + operatorList[index-1] +
str(operandList[index]) + ')'
else:
questionString = questionString
+ operatorList[index-1] +
str(operandList[index])

result = round(eval(questionString), 2)
#End of While Loop
questionString = questionString.
replace("**", "^")

print ('\n' + questionString)
```

```
userResult = input('Answer (correct to 2
d.p.if not an integer): ')

while True:
try:
if float(userResult) == result:print ("So
Smart")
return 1
else:
print ("Sorry, wrong answer.
The correct answer is", result)
return 0
except Exception as e:
print ("You did not enter anumber. Please
try again.")
userResult = input('Answer(correct to 2
d.p. if not an integer): ')
```

So'nggi so'z ...

Sahifani ochganingizda, Amazon ushbu kitobga baho berishingizni va Facebook va Twitter-da o'z fikrlaringiz bilan o'rtoqlashishingizni so'raydi.

Agar ushbu qo'llanma sizga yordam bergan bo'lsa, do'stlaringizga bu haqida xabar berish uchun bir necha soniya vaqt sarflasangiz, juda minnatdorman.

Men uchun dasturlash - bu san'at va fan. Bu juda o'ziga qaram va yoqimli. Ushbu ehtirosni iloji boricha ko'proq odamlarga baham ko'rilishiga umid qilaman.

Bundan tashqari, umid qilamanki, siz bu yerda o'rganishni to'xtatmaysiz. Agar siz ko'proq dasturiy muammolarga, jumboqlarga qiziqsangiz, https://projecteuler.net/ saytiga tashrif buyurishingiz mumkin. Rohatlaning!

Mundarija

Jeymi Chan

Python 0 dan 100 gacha

«Hamroh books» loyihasi asosida tayyorlandi

Loyiha rahbari: Subxon Ali
Tarjimon: Sirojiddin To'lanboyev
Muharrir: Shahzod Bahodirov
Sahifalovchi: Zarifjon G'ulomov