**Python ga kirish**

**Python dasturlash tili**

Python — bu mashhur yuqori darajadagi dasturlash tili bo‘lib, u turli xil dasturlar yaratishga mo‘ljallangan. Unga veb-ilovalar, o‘yinlar, ish stoli dasturlari hamda ma’lumotlar bazalari bilan ishlash kiradi. Python mashinaviy o‘qitish va sun’iy intellekt tadqiqotlari sohasida ham keng qo‘llanilmoqda.

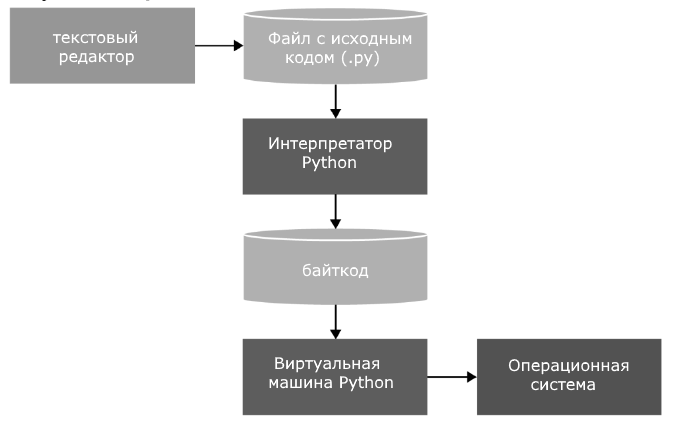
Python dasturlash tili ilk bor 1991-yilda gollandiyalik dasturchi Gvido van Rossum tomonidan e’lon qilingan. Shundan beri til katta rivojlanish yo‘lini bosib o‘tdi. 2000-yilda 2.0-versiya, 2008-yilda esa 3.0-versiya chiqarildi. Versiyalar orasida katta vaqt farqi bo‘lishiga qaramay, doimiy ravishda kichikroq versiyalar ham taqdim etib borildi. Shu bois, ushbu material yozilayotgan paytda eng so‘nggi amaldagi versiya 3.13 bo‘lib, u 2024-yil oktabr oyida chiqarilgan.

Python dasturlash tilining asosiy xususiyatlari:

* **Skript tili.** Dastur kodi skriptlar ko‘rinishida yoziladi.
* **Turli dasturlash paradigmalari qo‘llovi.** Jumladan, obyektga yo‘naltirilgan va funksional dasturlash paradigmalarini ham qo‘llab-quvvatlaydi.
* **Dasturlarni interpretatsiya qilish.** Skriptlarni ishlatish uchun interpretator kerak bo‘ladi, u skriptni ishga tushiradi va bajaradi.

Python dasturining ishlash jarayoni quyidagicha ko‘rinadi:  
Avvalo, biz matn muharririda ushbu dasturlash tilida bir qator ifodalar yozilgan skript yaratamiz. So‘ng bu skriptni bajarish uchun interpretatorga uzatamiz. Interpretator kodni oraliq baytkodga aylantiradi, keyin virtual mashina baytkodni operatsion tizim bajaradigan ko‘rsatmalar to‘plamiga o‘giradi.

Shuni ta’kidlash kerakki, garchi rasmiy jihatdan interpretator tomonidan dastlabki kodni baytkodga tarjima qilish va virtual mashina tomonidan baytkodni mashina buyruqlariga aylantirish ikki alohida jarayon bo‘lsa-da, amalda ular interpretatorning o‘zida birlashtirilgan.



* **Ko‘chirish qulayligi va platformadan mustaqillik.** Qaysi operatsion tizimdan foydalanishimiz — Windows, Mac OS yoki Linux — muhim emas. Interpretator mavjud bo‘lsa, yozilgan skript barcha ushbu tizimlarda ishga tushadi.
* **Xotirani avtomatik boshqarish.**
* **Dinamik tipizatsiya.**

Python — juda sodda dasturlash tili bo‘lib, u ixcham va bir vaqtning o‘zida ancha oddiy hamda tushunarli sintaksisga ega. Shu sababli uni o‘rganish oson, va bu uning aynan ta’lim sohasida eng mashhur dasturlash tillaridan biri bo‘lishiga sabab bo‘lgan. Masalan, 2014-yilda AQShda Python ta’lim uchun eng mashhur dasturlash tili deb e’tirof etilgan.

Python nafaqat ta’lim sohasida, balki amaliy dasturlar, jumladan tijorat maqsadidagi dasturlar yaratishda ham mashhur. Shuning uchun bu til uchun juda ko‘p kutubxonalar yozilgan va biz ulardan foydalanishimiz mumkin.

Bundan tashqari, ushbu dasturlash tilining juda katta dasturchilar hamjamiyati mavjud. Internetda Python bo‘yicha juda ko‘plab foydali materiallar, misollar topish va mutaxassislar tomonidan malakali yordam olish mumkin.

**Paketlar va kutubxonalar**

Python interpretatori dasturlar yaratish uchun yetarli funksionallik bilan birga keladi. Shunga qaramay, ba’zi vazifalar uchun bu imkoniyatlar yetarli bo‘lmasligi mumkin. Lekin tilning butun dunyo bo‘ylab keng dasturchilar hamjamiyati mavjudligi sababli Python’da turli maqsadlar uchun ishlatilishi mumkin bo‘lgan katta ekotizimga ega paketlar va kutubxonalar to‘plangan.

METANIT.COM saytining Python bo‘limida ushbu kutubxonalarning ayrimlari ko‘rib chiqiladi. Quyida ularning asosiylari keltiriladi.

Grafik dasturlar yaratish uchun:

* **Tkinter**
* **PyQt / PySide**
* **wxPython**
* **DearPyGui**
* **EasyGUI**

**Mobil ilovalar yaratish uchun:**

* **Kivy**
* **Toga**

**Veb-ilovalar yaratish uchun:**

* **Django**
* **Flask**
* **FastAPI**
* **Pylons**
* **Bottle**
* **CherryPy**
* **TurboGears**
* **Nagare**

**Jarayonlarni avtomatlashtirish uchun:**

* **Selenium** (veb-ilovalarni test qilish uchun)
* **Flask**
* **FastAPI**
* **Pylons**
* **Bottle**
* **CherryPy**
* **TurboGears**
* **Nagare**
* **robotframework**
* **pywinauto**
* **Lettuce**
* **Behave**
* **Requests**

**Turli fayl turlari bilan ishlash uchun:**

* **OpenPyXL** (Excel)
* **lxml** (XML)
* **ReportLab / borb** (PDF)
* **pdfrw / PyPDF2** (PDF)
* **Pandas** (CSV va Excel)

**Mashina o‘qitish, sun’iy intellekt va Data Science uchun:**

* **Pandas**
* **SciPy**
* **PyTorch**
* **Matplotlib**
* **Theano**
* **TensorFlow**
* **OpenCV**
* **Scikit-Learn**
* **Keras**
* **NumPy**

**Vizualizatsiya uchun:**

* **Matplotlib**
* **Seaborn**
* **Plotly**
* **Bokeh**
* **Altair**
* **HoloViews**

**Python versiyasini tekshirish**

O‘rnatilgandan so‘ng biz Python versiyasini tekshirishimiz mumkin. Buning uchun terminalda quyidagi buyruqni ishga tushiramiz:

python --version

O‘rnatilgan Python versiyasini bilish juda muhim. Hozirda Python’ning ikkita asosiy versiyasi mavjud: **Python 2** va **Python 3**. Biroq Python 2’ning oxirgi versiyasi eskirgan va endi qo‘llab-quvvatlanmaydi. Bu shuni anglatadiki, unda xatolar tuzatilmaydi va yangilanishlar chiqmaydi. Amaldagi versiya faqat **Python 3.0 va undan yuqori** hisoblanadi. Ushbu qo‘llanmada barcha darslar aynan shu versiya uchun tayyorlangan.

**Python’da ekranga ma’lumot chiqarish**

print funksiyasi kerakli ma’lumotni konsolga chiqarishga yordam beradi. Keling, qandaydir satrni chiqarib ko‘ramiz:

**print**('hello')

Endi esa sonni chiqaramiz:

**print**(1)

**Python’da izohlar (kommentariyalar)**

Python tilida izohlar qoldirish mumkin. Ular **bir qatorli** va **ko‘p qatorli** bo‘lishi mumkin.

🔹 **Bir qatorli izoh** # belgisi orqali yoziladi:

*# test*

🔹 **Ko‘p qatorli izoh** esa matnni uchta qo‘shtirnoq ichiga olish orqali yoziladi:

"""

    test1

    test2

    test3

"""

**Python’dagi o‘zgaruvchilar**

Har qanday dasturlash tilining asosiy tushunchasi bu o‘zgaruvchi hisoblanadi. O‘zgaruvchi — bu satrlar yoki sonlar kabi ma’lumotlarni saqlashimiz mumkin bo‘lgan konteynerdir.

Har bir o‘zgaruvchi nomga ega bo‘lishi kerak, u lotin harflari, raqamlar va pastki chiziq belgisidan iborat bo‘lishi mumkin. Python’da o‘zgaruvchi e’lon qilish uchun uning nomini yozib, = operatori orqali qiymatini ko‘rsatamiz.

test = 1

**Python’da o‘zgaruvchini konsolga chiqarish**

O‘zgaruvchining qiymatini chiqarish uchun print funksiyasidan foydalaniladi:

test = 1

**print**(test)  *# konsolda 1 ni chiqaradi*

**Python’da sonlar bilan matematik amallar**

Python’da barcha asosiy matematik amallarni bajarish mumkin. Quyida ularning har birini ko‘rib chiqamiz.

➕ **Qo‘shish:**

test = 1 + 2

**print**(test)  *# 3 chiqadi*

➖ **Ayirish:**

test = 2 - 1

**print**(test)  *# 1 chiqadi*

✖️ **Ko‘paytirish:**

test = 3 \* 2

**print**(test)  *# 6 chiqadi*

➗ **Bo‘lish:**

test = 4 / 2

**print**(test)  *# 2 chiqadi*

➗ **Butun bo‘lish:**

test = 10 // 3

**print**(test)  *# 3 chiqadi*

**Python’da o‘zgaruvchilar bilan matematik amallar**

Matematik amallarni nafaqat sonlar ustida, balki o‘zgaruvchilar ustida ham bajarish mumkin. Keling, ikkita o‘zgaruvchining qiymatlari yig‘indisini topamiz:

num1 = 1

num2 = 3

**print**(num1 + num2)  *# 4 chiqadi*

Natijani darhol chiqarish shart emas, uni avval biror o‘zgaruvchiga yozib olib, keyin chiqarish ham mumkin:

num1 = 1

num2 = 3

res = num1 + num2  *# yig‘indini res o‘zgaruvchisiga yozamiz*

**print**(res)  *# 4 chiqadi*

**Python’da qisqartirilgan amallar**

Ba’zida matematik amallar kodda ko‘p joy egallashi mumkin. Masalan, o‘zgaruvchiga 1 qo‘shamiz:

num = 1

num = num + 1  *# num o‘zgaruvchisiga 1 qo‘shyapmiz*

**print**(num)  *# 2 chiqadi*

Bu kodni qisqa yozish mumkin, ya’ni += operatoridan foydalanib:

num = 1

num += 1  *# num o‘zgaruvchisiga 1 qo‘shyapmiz*

**print**(num)  *# 2 chiqadi*

Xuddi shu usulda sonni ayirish ham mumkin:

num = 1

num -= 1  *# num o‘zgaruvchisidan 1 ayiryapmiz*

**print**(num)  *# 0 chiqadi*

**Python’da matematik amallarning ustuvorligi**

Python’dagi matematik amallar odatdagi matematikadagi kabi ustuvorlikka ega. Ya’ni avval **ko‘paytirish** va **bo‘lish**, keyin esa **qo‘shish** va **ayirish** bajariladi.

Quyidagi misolda avval 2 \* 2 hisoblanadi, so‘ng natijaga 3 qo‘shiladi:

res = 2 \* 2 + 3  *# natija 7 (4 + 3)*

**print**(res)

**Misol**

Quyidagi kod berilgan:

res = 10 - 6 \* 3 - 5

**print**(res)

Konsolda nima chiqadi?

**Python’da teng ustuvorlik**

Ko‘paytirish va bo‘lish bir xil ustuvorlikka ega va ular chapdan o‘ngga qarab ketma-ket bajariladi. Keling, misollar orqali ko‘rib chiqamiz.

Quyidagi kodda avval bo‘lish, keyin ko‘paytirish bajariladi:

res = 8 / 2 \* 4

**print**(res)  *# 16 chiqadi (4 \* 4)*

Agar belgilar o‘rin almashsa, avval ko‘paytirish, keyin bo‘lish bajariladi:

res = 8 \* 2 / 4

**print**(res)  *# 4 chiqadi (16 / 4)*

Keyingi misolda har bir yangi bo‘lish amali oldingisiga qo‘llanadi:

res = 16 / 2 / 2 / 2

**print**(res)  *# 2 chiqadi*

**Python’da sonning uzunligi**

Python 2’da sonli o‘zgaruvchining hajmi ishlatilayotgan protsessorga qarab 32 yoki 64 bit bilan cheklangan edi. Ammo Python 3’da bu cheklov olib tashlangan va son istalgancha katta bo‘lishi mumkin.

Keling, katta sonni olib, uni 5-darajaga ko‘taramiz:

tst = 1342453547656758678993553453 \*\* 5

**print**(tst)

**Misol**

Quyidagi sonlar berilgan:

num1 = 200 \*\* 500

num2 = 3000

Ularning ko‘paytmasini oling.

**Python’da o‘nli kasr sonlar**

Python’da o‘nli kasr (floating point) sonlar ham mavjud. Bunday sonlarda butun va kasr qismlari nuqta (.) orqali ajratiladi:

num = 0.5

**print**(num)  *# 0.5 chiqadi*

Bunday sonlar bilan barcha matematik amallarni bajarish mumkin. Masalan, ikkita kasr sonni qo‘shamiz:

res = 0.5 + 0.5  *# 1.0 chiqadi*

**print**(res)

**Python’da manfiy sonlar**

Sonlar manfiy bo‘lishi mumkin. Buning uchun sonning oldiga minus (-) belgisi qo‘yiladi:

num = -1

**print**(num)  *# -1 chiqadi*

Minus belgisini sonlarga ham, o‘zgaruvchilarga ham qo‘llash mumkin:

num1 = 1

num2 = -num1  *# num1 qiymatini teskari ishora bilan num2 ga yozdik*

**print**(num2)  *# -1 chiqadi*

Yoki quyidagicha ham yozish mumkin:

num = 1

**print**(-num)  *# -1 chiqadi*

**Python’da bo‘lishdan qolgan qoldiq**

Maxsus % operatori mavjud bo‘lib, uning yordamida bir sonning ikkinchisiga bo‘linishidan qolgan qoldiqni aniqlash mumkin:

**print**(10 % 3)  *# 1 chiqadi*

Agar bir son ikkinchisiga butun bo‘linadigan bo‘lsa, qoldiq nolga teng bo‘ladi:

**print**(10 % 2)  *# 0 chiqadi*

% operatorini nafaqat sonlarga, balki o‘zgaruvchilarga ham qo‘llash mumkin:

num1 = 10

num2 = 3

**print**(num1 % num2)  *# 1 chiqadi*

**Python’da darajaga oshirish**

Sonni darajaga oshirish uchun maxsus \*\* operatori mavjud. Keling, uning yordamida sonni uchinchi darajaga oshiramiz:

**print**(10 \*\* 3)  *# 1000 chiqadi*

Darajaga oshirishni o‘zgaruvchi qiymati bilan ham bajarish mumkin:

num = 10

**print**(num \*\* 3)  *# 1000 chiqadi*

Ham son, ham daraja alohida o‘zgaruvchilarda saqlanishi ham mumkin:

num1 = 10

num2 = 3

**print**(num1 \*\* num2)  *# 1000 chiqadi*

**Python’dagi satrlar (strings)**

Avval aytilganidek, ma’lumotlar turli xil turlarga ega bo‘lishi mumkin. Ulardan biri — **sonlar**, biz ular bilan allaqachon tanishdik. Endi esa **satrlar**ni ko‘rib chiqamiz.

Satrlar **qo‘shtirnoq** yoki **yakkatirnoq** ichida yoziladi:

txt = 'abc'

**print**(txt)  *# 'abc' chiqadi*

Satrni **ikkita qo‘shtirnoq** ichida ham yozish mumkin:

txt = "abc"

**print**(txt)  *# 'abc' chiqadi*

**Python’da satrlarni qo‘shish (konkatenatsiya)**

Satrlarni qo‘shish uchun + operatoridan foydalaniladi:

test = 'abc' + 'def'  *# ikki satrni qo‘shyapmiz*

**print**(test)  *# 'abcdef' chiqadi*

Satrlar o‘zgaruvchilarda ham saqlanishi mumkin:

test1 = 'abc'

test2 = 'def'

test3 = test1 + test2  *# ikki satrni qo‘shyapmiz*

**print**(test3)  *# 'abcdef' chiqadi*

O‘zgaruvchi va oddiy satrlarni ham qo‘shish mumkin:

test1 = 'abc'

test2 = 'def'

test3 = test1 + '!!!' + test2

**print**(test3)  *# 'abc!!!def' chiqadi*

**Python’da satrlarni ko‘paytirish**

Python’da satrni son bilan ko‘paytirish mumkin, ya’ni satrni bir necha marta takrorlash:

**print**(3 \* 'abc')  *# 'abcabcabc' chiqadi*

Bu yerda 'abc' satri 3 marta ketma-ket yoziladi.

**Misol:**

Quyidagi natijani qanday olish mumkinligini yozing:

'a1b2a1b2a1b2a1b2'

**Python’da qator belgilariga murojaat qilish**

Qatorning istalgan belgisiga murojaat qilish uchun uning indeksidan (tartib raqamidan) foydalanamiz va uni kvadrat qavs ichiga yozamiz:

txt = 'abcde'

**print**(txt[1]) *# natija: 'b'*

E’tibor bering, belgilar raqamlash **0 dan** boshlanadi.

Shuningdek, belgini qatordan oxiridan ham olish mumkin. Buning uchun indeksni **manfiy** ishora bilan yozamiz. Bu holda oxirgi belgi -1 indeksdan boshlanadi:

txt = 'abcde'

**print**(txt[-1]) *# natija: 'e'*

**Python’da qator belgilarini ekranlash**

Agar qator ichida maxsus belgilar, masalan, bitta qo‘shtirnoq (') chiqarmoqchi bo‘lsak, ularni **ekranlash** kerak bo‘ladi. Buning uchun **teskari qiya chiziq (\)** ishlatiladi:

tst = 'sfgs\'sdf'

**print**(tst) *# natija: 'sfgs'sdf'*

**Misol:**

Quyidagi satr berilgan:

tst = "abcd"fr'23"

Uni xatosiz qilib konsolga chiqaring.

**Python’da satr uzunligi**

Satrdagi belgilar sonini (ya’ni uning uzunligini) len funksiyasi yordamida topish mumkin:

**print**(**len**('abc')) *# ekranga 3 chiqaradi*

Satr o‘zgaruvchida ham saqlanishi mumkin:

txt = 'abc'

**print**(**len**(txt))

Bo‘sh joy (probеl) ham belgi hisoblanadi:

txt = 'ab c'

**print**(**len**(txt)) *# ekranga 4 chiqaradi*

**Ko‘p qatorli satrlar Python’da**

Python’da satrlarni bir necha qatordan iborat qilib yozish mumkin. Buning uchun satrni boshida va oxirida uchta bir xil qo‘shtirnoq (''' yoki """) qo‘yiladi:

**print**('''

    1

    2

    3

''')

Shuningdek, qo‘shtirnoqlardan ham foydalanish mumkin:

**print**("""

    1

    2

    3

""")

**Python’da mantiqiy qiymatlar**

Sonlar va satrlardan tashqari yana bir ma’lumot turi mavjud — **mantiqiy (boolean)**. Bu tur faqat ikki qiymatdan iborat: **True** yoki **False**, ya’ni **rost** yoki **yolg‘on**.

Mantiqiy tur shunday holatlar uchun ishlatiladi, u yerda javob faqat ikki xil bo‘lishi mumkin: ha yoki yo‘q. Masalan, “Sizga allaqachon 18 yosh bo‘ldimi?” degan savolga javob **ha** bo‘lsa True, **yo‘q** bo‘lsa False bo‘ladi.

Misol:

isAdult = True  *# allaqachon voyaga yetgan*

Qiymatni boshqa qiymatga o‘zgartirish mumkin:

isAdult = False  *# hali voyaga yetmagan*

Endi o‘zgaruvchi qiymatini ekranga chiqaramiz:

isAdult = False

**print**(isAdult)  *# False chiqadi*

**Python’da *None* qiymati**

Python’da yana bir maxsus qiymat mavjud — **None**, u “hech narsa yo‘q” degan ma’noni bildiradi. Masalan, o‘zgaruvchiga bu qiymatni berish orqali unda hech narsa saqlanmaganini bildirishingiz mumkin.

Misol:

test = None

**print**(test)  *# None chiqadi*

**Python’da qat’iy tiplar**

Python’ning o‘ziga xosligi shundaki, u **qat’iy tipga ega**. Bu shuni anglatadiki, turli tipdagi obyektlar bilan to‘g‘ridan-to‘g‘ri amal bajarib bo‘lmaydi. Masalan, son va satrni qo‘shib bo‘lmaydi:

num = 12

txt = 'ab'

**print**(num + txt)  *# xato beradi*

Bu muammoni hal qilish uchun ikkala obyektni **bir xil tipga o‘tkazish** kerak bo‘ladi. Buni qanday qilishni keyingi darslarda o‘rganasiz.

**Python’da satrga o‘tkazish**

Sonni satrga o‘tkazish uchun str funksiyasidan foydalaniladi. Keling, uning yordamida sonni satrga aylantiramiz:

test1 = 'abc'

test2 = 1

test3 = test1 + **str**(test2)

**print**(test3)  *# 'abc1' chiqadi*

**Python’da butun songa o‘tkazish**

Satrni songa o‘tkazish uchun int funksiyasidan foydalaniladi. Masalan, ikkita satr o‘zgaruvchisini songa aylantirib, ularning yig‘indisini olamiz:

txt1 = '1'

txt2 = '2'

res = **int**(txt1) + **int**(txt2)

**print**(res)  *# 3 chiqadi*

Shuningdek, int funksiyasini to‘g‘ridan-to‘g‘ri chaqirish orqali ham ishlatish mumkin:

**print**(**int**('123'))  *# 123 chiqadi*

**Python’da raqamli satrlar**

Agar satr faqat raqamlardan iborat bo‘lsa:

txt = '123'  *# raqamli satr*

Masalan, uning birinchi va ikkinchi belgilarining yig‘indisini topamiz:

txt = '123'

**print**(txt[0] + txt[1])  *# natija: '12' – satr sifatida qo‘shiladi*

Ko‘rib turganingizdek, satrimiz belgilar ham satr bo‘lib, ular **satr sifatida** qo‘shiladi. Agar ularni **son sifatida** qo‘shmoqchi bo‘lsak, har bir belgiga int funksiyasini qo‘llaymiz:

txt = '123'

**print**(**int**(txt[0]) + **int**(txt[1]))  *# natija: 3*

**Python’da sondan belgilarni olish**

Agar bir sonning barcha raqamlarini qo‘shmoqchi bo‘lsak, unga indeks orqali murojaat qilish xato beradi:

num = 123

**print**(num[0] + num[1])  *# xato beradi*

Xatoni oldini olish uchun avval sonni satrga aylantirish kerak, so‘ng kerakli belgini yana songa o‘tkazamiz:

txt = '123'

**print**(txt[0] + txt[1])  *# '12' – satr sifatida qo‘shiladi*

Agar raqamlarni **son sifatida** qo‘shmoqchi bo‘lsak:

num = 123

txt1 = **str**(num)[0]

txt2 = **str**(num)[1]

res = **int**(txt1) + **int**(txt2)

**print**(res)  *# 3 chiqadi*

**Python’da o‘nli songa o‘tkazish**

O‘nli songa (floating point) o‘tkazish uchun float funksiyasidan foydalaniladi.

Satrga qo‘llash misoli:

txt = '12.1'

**print**(**float**(txt))  *# 12.1 chiqadi*

Butun songa qo‘llash misoli:

num = 12

**print**(**float**(num))  *# 12.0 chiqadi*

O‘tkazishdan so‘ng o‘zgaruvchilar bilan matematik amallarni bajarish mumkin:

test1 = '1.1'

test2 = '2.1'

test3 = **float**(test1) + **float**(test2)

**print**(test3)  *# 3.2 chiqadi*

**Python’da *input* funksiyasi**

input funksiyasi konsolda ma’lumot kiritishga imkon beradi va ularni keyinchalik ishlatish uchun saqlaydi. Funksiya parametriga foydalanuvchiga ko‘rsatish uchun matn berish mumkin. Kod ishga tushganda, foydalanuvchi o‘ng tomondagi oynaga ma’lumot kirita oladi. Ma’lumotni saqlash uchun **Enter** tugmasini bosish kerak. input funksiyasi har doim natija sifatida **satr** qaytaradi.

**Misol 1. Ma’lumotni chiqarish**

tst = **input**('enter number: ')

Kodni ishga tushirganimizda konsolda:

enter number:

ko‘rinadi. Foydalanuvchi o‘ng tomondagi oynaga qiymat kiritadi, masalan 12. Keyin bu qiymat tst o‘zgaruvchisiga yoziladi. Uni konsolga chiqarish uchun:

**print**(tst)

Natija:

enter number: 12

12

**Misol 2. Kiritilgan qiymat bilan matematik amallar**

tst = **input**('enter number: ')

**print**(tst + 1)  *# xato beradi*

Bu xato, chunki input har doim satr qaytaradi. Matematik amal bajarish uchun kiritilgan qiymatni songa aylantirish kerak:

**print**(**int**(tst) + 1)  *# 13 chiqadi*