
# PYTHON DASTURLASH TILI VA SINTAKSISI

## PYTHON DASTURLASH TILI VA UNING IMKONIYATLARI

- **Rasmiy sayt –** www.python.org
- **Python** mashhur dasturlash tili. U Guido van Rossum tomonidan yaratilgan va 1991 yilda chiqarilgan.
- **Python** – bu o'rganishga oson va shu bilan birga imkoniyatlari yuqori bo'lgan oz sonlik zamonaviy dasturlash tillari qatoriga kiradi. Python yuqori darajadagi ma'lumotlar strukturasi va oddiy lekin samarador obyektga yo'naltirilgan dasturlash uslublarini taqdim etadi.

**Stack Overflow** saytining 2019-yildagi dasturchilar o'rtasida dasturlash tillari bo'yicha olib borilgan so'rovnomasida, eng qulay va ko'p foydalaniladigan dasturlash tillari ro'yxatida **Python** JavaScriptdan so'ng ikkinchi o'rinni egallagan. Shu bilan bir qatorda dunyoning **Twitter, Pinterest, HP, Symantec, Instagram va Groupon** kabi yirik korxonalar aynan **Python** dasturlash tilidan foydalanmoqda. **YouTube, DropBox, Google** va **Quora** kabi dunyoning mashhur online platformalarining dasturiyta'minoti ham aynan python dasturlash tilida yozilganligi ushbu dasturlash tiliga bo'lgan talabning yuqori ekanligini anglatadi. Python nafaqat web sohasida balki sun'iy intellekt va robotexnika sohasida ham yuqori talabga ega til hisoblanadi.

## Python DASTURLASH TILIDA YARATILGAN YIRIK LOYIHALAR



# Python dasturlash tilining imkoniyatlari

**Python** – bu o'rganishga oson va shu bilan birga imkoniyatlari yuqori bo'lgan oz sonlik zamonaviy dasturlash tillari qatoriga kiradi. Python yuqori darajadagi ma'lumotlar strukturasi va oddiy lekin samarador obyektga yo'naltirilgan dasturlash uslublarini taqdim etadi.

Python quyidagi sohalarda ishlatiladi:

- Web dasturlash (serverlar bilan)
- Dasturiy ta’minot
- Matematika
- Tizim skriptlari

## Nima uchun aynan Pythonni o’rganish kerak ?

- Oddiy, o'rganishga oson, sodda sintaksisga ega, dasturlashni boshlash uchun qulay, erkin va ochiq kodlik dasturiy ta'minot.
- Dasturni yozish davomida quyi darajadagi detallarni, misol uchun xotirani boshqarishni hisobga olish shart emas.
- Ko'plab platformalarda hech qanday o'zgartirishlarsiz ishlay oladi. (Windows, Mac, Linux, Raspberry Pi va boshqalar)
- Interpretatsiya(Интерпретируемый) qilinadigan til.
- Kengayishga (Расширяемый) moyil til. Agar dasturni biror joyini tezroq ishlashini xoxlasak shu qismni C yoki C++ dasturlash tillarida yozib keyin shu qismni python kodi orqali ishga tushirsa(chaqirsa) bo'ladi.
- Juda ham ko'p xilma-xil kutubxonalarga ega.
- xml/html fayllar bilan ishlash;
- http so`rovlari bilan ishlash;
- GUI(grafik interfeys);
- Web dastur tuzish;
- FTP bilan ishlash;
- Rasmli audio video fayllar bilan ishlash;
- Robot texnikada;
- Matematik va ilmiy hisoblashlarni dasturlashda juda ham qo’l keladi.

Pythonni katta proyektlarda ishlatish mumkin. Bularga misol qilib **Google**, ***Instagram***, **YouTube** ni misol qilib ko’rsatsak bo’ladi. Bu loyiahalr aynan Python dasturlash tilida yozilgan. Chunki, uni chegarasi yo`q, imkoniyati yuqori. Shuningdek, u sodda va universalligi bilan dasturlash tillari orasida eng yaxshisidir.

# PYTHON DASTURLASH TILI SINTAKSISI

Python dasturlash tili sintaksisi o`zi kabi juda sodda:

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

  Pythonda sintaksis juda sodda tuzilishga ega. Quyida "**Salom dunyo**" gapini ekranga chiqaruvchi kod ko'rsatilgan:

```
print ("Hello world")
```

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

  Odatda dasturlash tillarida abzats kodni oson o'qilishi uchun ishlatiladi. Ammo Pythonda abzats kodning blokini ajratib ko'rsatadi. Misol keltiramiz:

```
if 5 > 2:
    print("Besh ikkidan katta")
```

  Agar kodimizni mana bunday tarzda yozsak dasturda xatolik yuz beradi:

```
if 5 > 2:
print("Besh ikkidan katta")
```

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

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

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

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

## Kalit so`zlar

**False** – yolg`on.

**True -** rost.

**None -** “bo`sh” obyekt.

**while** – while tsikli

**with / as** – konteks menejeri.

**break** – sikldan chiqish.

**class** – metod va atributlarda iborat.

**continue** – sikldan keyingi iteratsiyaga o`tish.

**from –** moduldan bir nechta funksiyani import qilish.

**import** – moduldan import.

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

**if -** agar.

**else** – for/else yoki if/elsega qarang.

**elif** – aks holda, agar.

**for** – for sikli.

**def** – funksiyani aniqlash.

**del** – obyektni yo`qotish.

**not** –mantiqiy INKOR amali.

**or** –mantiqiy YOKI amali.

**and** – mantiqiy VA amali.

**lambda** –yashirin funksiyani aniqlash.

## Pythonda izohlar

Izohlar quyidagi holatlarda ishlatiladi:

- **Koddagi bajarilayotgan ishlarni tushuntirish uchun;**
- **Kodning o’qilishini yanada osonlashtirish uchun;**
- **Kodning ba’zi qismlarini vaqtincha hisobga olmay turish uchun;**

## Izohlarni hosil qilish

Izohlar # belgisi bilan hosil qilinadi va python o’sha qismni kod deb qabul qilmaydi:

```
# Bu yerda izoh
print("Salom dunyo")
```

Izohlarni kod yozilgan qator oxiriga yozish ham mumkin:

```
print("Salom dunyo") #Bu yerda izoh
```

Kodning biror qismini izohga kiritsak o’sha qism natija bermaydi. Quyidagi holatda ***Salom dunyo*** jumlasi ekranga chiqmaydi:

```
# print ("Salom dunyo")
print ("Dasturlashni o'rganamiz")
```

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

# Ko’p qatorli izohlar

Python ko’p qatorli izohlar hosil qilish uchun alohida belgiga ega emas . Shuning uchun har bir qator uchun alohida # belgisi ishlatiladi. Ammo 3 talik qo’shtirnoq ichiga yozilgan matnni o’zgaruvchiga biriktirilmasa ko’p qatorli izoh sifatida ishlatish mumkin:

```
"""
Bu izoh
ko'p qatorli
ozihdir
"""

print("Dasturlashni o'rganamiz")
```

# PYTHONDA O`ZGARUVCHILAR

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

Pythonda o'zgaruvchilar ularni qiymatlarini tenglashtirish bilan hosil qilinadi. O'zgaruvchilarning turini alohida e'lon qilish shart emas. Pythonda barchasi avtomatik tarzda ishlaydi:

```
x = 5
y = "Salom dunyo"
```

## O'zgaruvchilarni hosil qilish

O'zgaruvchilar ma'lum bir turdagi qiymatni o'zida saqlovchi konteynerlardir. Boshqa dasturlash tillaridan farqli, Python o'zgaruvchilarni e'lon qilish uchun alohida buyruqqa ega emas. O'zgaruvchilar ularga qiymatni tenglashtirish orqali hosil qilinadi. Quyida biz 2 ta o'zgaruvchini 2xil trudagi qiymatga biriktiramiz va ekranga chiqaramiz:

```
x = 5
y = "Python"
print(x)
print(y)
```

O'zgaruvchilarni qaysi turda ekanligni e'lon qilish shart emas. Buni Python avtomatik tarzda aniqlaydi. O'zgaruvchilarning turlarini kodning istalgan qismida o'zgartirish ham mumkin.

```
x = "Dastur"   # x-satr
x = 5          # x endi butun son
print(x)
```

Satrli o'zgaruvchilar qo'shtirnoq yoki bittalik tirnoqlar ichiga yozilish bilan e'lon qilinishi mumin:

```
x = "Dastur"
# ikkala o'zgaruvchi ham bir xil
y = 'Dastur'
```

## O’zgaruvchi nomlari

O’zgaruvchilar harflar yoki so’zlar bilan ifodalanishi mumkin. Ularni ifodalash uchun ayrim qoidalar mavjud:

- O’zgaruvchi nomi harf yoki tag chiziq bilan boshlanishi kerak;
- O’zgaruvchi nomi raqam bilan boshlanmasligi kerak;
- O’zgaruvchi nomi faqat harflar, raqamlar va tag chiziqdan iborat bo’lishi mumkin;
- O’zgaruvchi nomlari katta-kichik harflarni farqlaydi (ism, iSm, ISM – bular 3 xil o’zgaruvchilar);
- O’zgaruvchi nomi orasida bo’shliq (probel) bo’lmasligi kerak;

```
# To'g'ri nomlangan o'zgaruvchilar:

myvar  =  "Python"
my_var = "Python"
_myvar = "Python"
myVar  =  "Python"
MYVAR  =  "Python"
myvar2 = "Python"

# Noto'g'ri nomlangan o'zgaruvchilar:

2myvar = "Python"
my-var = "Python"
my var = "Python"
```

## Bir nechta o’zgaruvchiga qiymat o’zlashtirish

Pythonda bir nechta o’zgaruvchiga qiymatlarni bir qatorning o’zida o’zlashtirish mumkin:

```
x, y, z = "Olma", "Banan", "Nok"
print(x)
print(y)
print(z)
```

Va aksincha, bir qiymatni bir nechta o’zgaruvchiga o’zlashtirish ham mumkin:

```
x = y = z = "Meva"
print(x)
print(y)
print(z)
```

## O’zgaruvchilarni ekranga chiqarish

Pythonda o’zgaruvchilarni yoki natijalarni ekranga chiqarish uchun **print** funksiyasidan foydalaniladi. Biror matnga satr o’zgaruvchisini biriktirish uchun **“+”** belgisi ishlatiladi:

```
x = "maroqlidir"
print ("Dasturlashni o'rganish "+x)
```

Bundan tashqari **“+”** belgisini o’zgaruvchilarni o’zaro biriktirish uchun ham ishlatsa bo’ladi:

```
x = "Dasturlashni"y
= "o'rganamiz"
print (x+y)
```

Sonli o’zgaruvchilar uchun “+” belgisi matematik amal sifatida ta’sir qiladi:

```
x = 5
y = 10
print (x+y)
```

Satr o’zgaruvchini sonli o’zgaruvchiga qo’shmoqchi bo’lsak **Python** xato yuz berganini ma’lum qiladi:

```
x = 5
y = "besh"
print (x+y)
```

## Global o’zgaruvchilar

Funksiyadan tashqarida hosil qilingan o’zgaruvchilar global o’zgaruvchilar hisoblanadi. Global o’zgaruvchilar kodning istalgan qismida (funksiya ichida ham, tashqarisida ham) ishlatilishi mumkin. Quyidagi kodda funksiya tashqarisida o’zgaruvchi hosil qilamiz va uni funksiya ichida ishlatamiz:

```
x = "qiziq"

def funksiya(): print("Dasturlash
    juda "+ x)

funksiya()
```

Funksiya ichida hosil qilingan o’zgaruvchi lokal o’zgaruvchi deyiladi. Agar lokal va global o’zgaruvchilarni nomlari bir xil bo’lsa , funksiya ichida lokal o’zgaruvchining qiymati funksiya tashqarisida esa global o’zgaruvchining qiymati olinadi:

```
x = "shirin"

def funksiya():
    x = "foydali"
    print("Olma "+ x)

funksiya()

print ("Olma "+ x)
```

# Global kalit so’zi

Oddiy holatda funksiya ichida hosil qillingan o’zgaruvchi lokal o’zgaruvchi hisoblanadi. Ammo funksiya ichida ham global o’zgaruvchi hosil qilish mumkin. Buning uchun ***global*** kalit so’zi ishlatiladi.

```
def funksiya():
    global x
    x = "shirin"
    print("Olma "+x)

funksiya()

print ("Olma "+x)
```

Agar global o’zgaruvchining qiymatini funksiya ichida o’zgartirmoqchi bo’lsangiz ham ***global*** kalit so’zi ishlatiladi:

```
x = "shirin"

def funksiya():
    global x
    x = "foydali"
    print("Olma "+ x)

funksiya()

print ("Olma "+ x)
```

# PYTHON OPERATORLARI

Operatorlar o’zgaruvchi va qiymatlar ustida amallar bajarish uchun ishlatiladi. **Python** operatorlari quyidagilar:

- **Arifmetik operatorlar**
- **O’zlashtirish operatorlar**
- **Taqqoslash operatorlari**
- **Mantiq operatorlari**
- **Aniqlash operatorlari**
- **A’zolik operatorlari**
- **Bitli operatorlar**

## Arifmetik operatorlar

Arifmetik operatorlar odatiy matematik amallarni bajarish uchun ishlatiladi:

| | | |
|---|---|---|
| **+** | **Qo’shish** | **x+y** |
| **-** | **Ayirish** | **x-y** |
| ***** | **Ko’paytirish** | **x*y** |
| **/** | **Bo’lish** | **x/y** |
| **%** | **Qoldiqli bo’lish** | **x%y** |
| **//** | **Butunli bo’lish** | **x//y** |

Ularni amalda sinab ko’rsak yaxshiroq tushunamiz:

```
x = 10
y = 3

print(x + y)
print(x - y)
print(x * y)
print(x / y)
print(x % y)
print(x ** y)
print(x // y)
```

```
13
7
30
3.3333333333333335
1
1000
3
```

## O’zlashtirish operatorlari

| | | |
|---|---|---|
| = | x = 5 | x=5 |
| += | x += 3 | x = x + 3 |
| - = | x -= 3 | x= x - 3 |
| *= | x *= 3 | x= x * 3 |
| /= | x /= 5 | x = x / 5 |
| %= | x %= 5 | x = x % 5 |
| //= | x //= 5 | x = x // 5 |
| **= | x **= 5 | x = x ** 3 |
| &= | x &= 3 | x = x & 3 |
| \|= | x \|= 3 | x = x \|3 |
| ^= | x ^= 3 | x = x ^ 3 |
| >>= | x >>= 3 | x = x >> 3 |
| <<= | x <<= 3 | x = x << 3 |

```
x = 5

x +=3
print(x)

x -=3
print(x)

x *=3
print(x)
```

## Taqqoslash operatorlari

Taqqoslash operatorlari qiymatlarni o’zaro taqqoslash uchuyn ishlatiladi:

| | | |
|---|---|---|
| = = | Teng | x == y |
| != | Teng emas | x != y |
| > | Katta | x > y |
| < | Kichik | x < y |
| >= | Katta yoki teng | x >= y |
| <= | Kichik yoki teng | x <= y |

## Mantiq operatorlari

Mantiq operatorlar shartlarni birlashtirib ishlatish uchun kerak:

- **and** - Agar ikkala shart ham rost bo'lsa, rost qiymat qaytaradi.
- **or** - Kamida bitta shart rost bo'lsa ham rost qiymat qaytaradi.
- **not** - Shart qiymatini teskarisiga o'zgartiradi, ya'ni rost bo'lsa yolg'on, yolg'on bo'lsa rost bo'ladi.

```
a = 5

print (a>3 and a<10)
print (a>3 or a<4)
print (not(a>3 and a<10))
```

```
True
True
False
```

## Aniqlash operatorlari

Aniqlash operatorlari o'zaro 2 ta obyektlarni solishtiradi. Bunda ularning o'zaro qiymatlarini tengligi bo'yicha emas, haqiqatdan ham ular bir xil obyekt ekanligi va bir xil xotira yo'nalishiga egaekanligi bo'yicha taqqoslanadi. Bu operatorlar 2 ta:

- **is** - Ikkala o'zgaruvchi ham bir xil obyekt bo'lsa rost, aks holda yolg'on qiymat qaytaradi.
- **is not** - Obyektlar bir xil bo'lmasa rost, aks holda yolg'on qiymat qaytaradi.

```
x = ["olma", "banan"]
y = ["olma", "banan"]z
= x


print(x is z)
print(x is y)
print(x == z)

#----------------------------------------------------------

print(x is not z)
print(x is not y)
print(x != z)
```

```
True
False
True
False
True
False
```

## A’zolik operatorlari

A’zolik operatorlari biror ketma-ketlik obyektga tegishli ekanligini tekshiradi:

- **in** - Belgilangan qiymat obyektda mavjud bo’lsa, rost qiymat qaytaradi.
- **not in** - Belgilangan qiymat obyektda mavjud bo’lmasa, rost qiymat qaytaradi.

```
x = ["audi", "mustang"]

print("audi" in x)
print("audi" not in x)
```

```
True
False
```

## Bitli operatorlar

Bitli operatorlar ikkilik sanoq sistemasi bilan ishlashda kerak bo’ladi:

- **&** **(AND)** - Ikkala bit ham 1 ga teng bo’lsa, 1 ga o’rnatiladi.
- **|** **(OR)** - Kamida bitta bt 1 ga teng bo’lsa, ikkala bitni ham 1 ga o’rnatadi.
- **^** **(XOR)** - Faqat bitta bit 1 ga teng bo’lsa, ikkala bitni ham birga o’rnatadi.
- **~** **(NOT)** - Barcha bitlarni invertlaydi (teskarisiga o’zgartiradi)
- **<<** - O’ngdan chapga nollarni siljitib, chapdagi chetki bo’laklarni tushirib yuboradi.
- **>>** - Chapdan o’ngga bitlarning nusxalari kiritilib siljitib boriladi. O’ngdagi chetki bitlar tushib qoladi.