17. Funksional uslub: lambda, map, filter, reduce



Pythonda funksiya ham oddiy qiymat: uni o'zgaruvchiga saqlash, boshqa funksiyaga argument qilib berish mumkin. Bu imkoniyat **funksional uslub** deb ataladigan yozish usulini beradi: «har bir element bilan nima qilish kerak» deb aytamiz, siklni esa Python o'zi bajaradi.

1. Funksiya — qiymat

```
def ikki_baravar(x):
    return x * 2

f = ikki_baravar          # qavssiz — funksiyaning o'zi
print(f(21))

def ikki_marta_qol(funksiya, qiymat):
    return funksiya(funksiya(qiymat))

print(ikki_marta_qol(ikki_baravar, 5))
```

Natija:

```
42
20
```

2. lambda — nomsiz qisqa funksiya

`lambda parametrlar: ifoda` — bitta ifodadan iborat funksiya. `return` yozilmaydi — ifoda natijasi o'zi qaytadi:

```
kvadrat = lambda x: x * x
qoshish = lambda a, b: a + b
print(kvadrat(9), qoshish(3, 4))
print((lambda s: s.upper())("salom"))
```

Natija:

```
81 7
SALOM
```

> **Eslatma**
>
> lambda — qisqa, bir martalik funksiyalar uchun (ko'pincha map, filter, sorted ichida). Murakkab mantiq bo'lsa, oddiy `def` bilan nomli funksiya yozing — o'qish osonroq.

3. map() — har bir elementni o'zgartirish

`map(funksiya, ketma-ketlik)` funksiyani har bir elementga qo'llaydi. Natija — «dangasa» ob'ekt, uni `list()` bilan ro'yxatga aylantiramiz:

```
sonlar = [1, 2, 3, 4, 5]
print(list(map(lambda x: x * x, sonlar)))
print(list(map(str.upper, ["olma", "nok"])))
print(list(map(int, "3 8 15".split())))
```

Natija:

```
[1, 4, 9, 16, 25]
['OLMA', 'NOK']
[3, 8, 15]
```

Oxirgi qatorni allaqachon bilasiz — `map(int, input().split())` aynan shu funksiya.

4. filter() — tanlab olish

`filter(funksiya, ketma-ketlik)` funksiya `True` qaytargan elementlarni qoldiradi:

```
sonlar = [3, -1, 8, 0, -5, 12]
print(list(filter(lambda x: x > 0, sonlar)))
sozlar = ["olma", "", "nok", "", "anor"]
print(list(filter(None, sozlar)))   # None — "rost" qiymatlarni qoldiradi
```

Natija:

```
[3, 8, 12]
['olma', 'nok', 'anor']
```

5. reduce() — bitta natijaga yig'ish

`functools.reduce(funksiya, ketma-ketlik, boshlang'ich)` elementlarni ketma-ket «yig'adi»: avval 1- va 2-elementga, so'ng natija va 3-elementga, va hokazo:

```
from functools import reduce
sonlar = [1, 2, 3, 4, 5]
print(reduce(lambda a, b: a + b, sonlar))
print(reduce(lambda a, b: a * b, sonlar, 1))
print(reduce(lambda a, b: a if a > b else b, [4, 9, 2]))
```

Natija:

```
15
120
9
```

Yig'indi uchun `sum()`, eng katta uchun `max()` bor — reduce esa o'zingiz xohlagan «yig'ish» qoidasi uchun.

6. Zanjir: filter + map + reduce

Musbat sonlarning kvadratlari yig'indisi — uch funksiya birga:

```
from functools import reduce
sonlar = [3, -1, 4, -2, 5]
musbat = filter(lambda x: x > 0, sonlar)
kvadrat = map(lambda x: x ** 2, musbat)
print(reduce(lambda a, b: a + b, kvadrat, 0))
```

Natija:

```
50
```

7. zip() va enumerate()

`zip` bir nechta ro'yxatni juftlab beradi, `enumerate` esa element bilan birga uning tartib raqamini:

```
ismlar = ["Ali", "Malika", "Sardor"]
ballar = [78, 95, 88]
for ism, ball in zip(ismlar, ballar):
    print(ism, ball)
print(dict(zip(ismlar, ballar)))
for i, ism in enumerate(ismlar, 1):
    print(i, ism)
```

Natija:

```
Ali 78
Malika 95
```

```
Sardor 88
{'Ali': 78, 'Malika': 95, 'Sardor': 88}
1 Ali
2 Malika
3 Sardor
```

8. key= parametri: sorted, min, max

`sorted`, `min`, `max` ga `key` funksiyasi berilsa, elementlar o'sha funksiya natijasi bo'yicha solishtiriladi:

```
sozlar = ["anor", "olma", "nok", "shaftoli"]
print(sorted(sozlar, key=len))
print(sorted(sozlar, key=lambda s: s[-1]))
print(max(sozlar, key=len))
talabalar = [("Ali", 78), ("Malika", 95), ("Sardor", 88)]
print(sorted(talabalar, key=lambda t: t[1], reverse=True))
```

Natija:

```
['nok', 'anor', 'olma', 'shaftoli']
['olma', 'shaftoli', 'nok', 'anor']
shaftoli
[('Malika', 95), ('Sardor', 88), ('Ali', 78)]
```

9. any() va all()

```
ballar = [78, 95, 54, 88]
print(any(b < 55 for b in ballar))   # birortasi 55 dan pastmi?
print(all(b >= 50 for b in ballar))  # hammasi 50 va undan yuqorimi?
```

Natija:

```
True
True
```

10. Sikl va funksional uslub: taqqoslash

Vazifa	Sikl bilan	Funksional
Kvadratlar	`for` + `append`	`list(map(lambda x: x*x, a))`
Juftlar	`for` + `if` + `append`	`list(filter(lambda x: x%2==0, a))`
Ko'paytma	`p = 1; for x in a: p *= x`	`reduce(lambda p, x: p*x, a, 1)`

Ikkalasi ham to'g'ri. Ko'p Python dasturchilari yana qisqaroq yo'lni afzal ko'radi — **list comprehension** (keyingi mavzu).

11. Xulosa

- lambda — bir ifodali nomsiz funksiya.
- map — har birini o'zgartiradi, filter — tanlaydi, reduce — bittaga yig'adi (functools).
- zip — juftlaydi, enumerate — raqamlaydi; sorted/min/max — key= bilan.
- any — birortasi, all — hammasi.