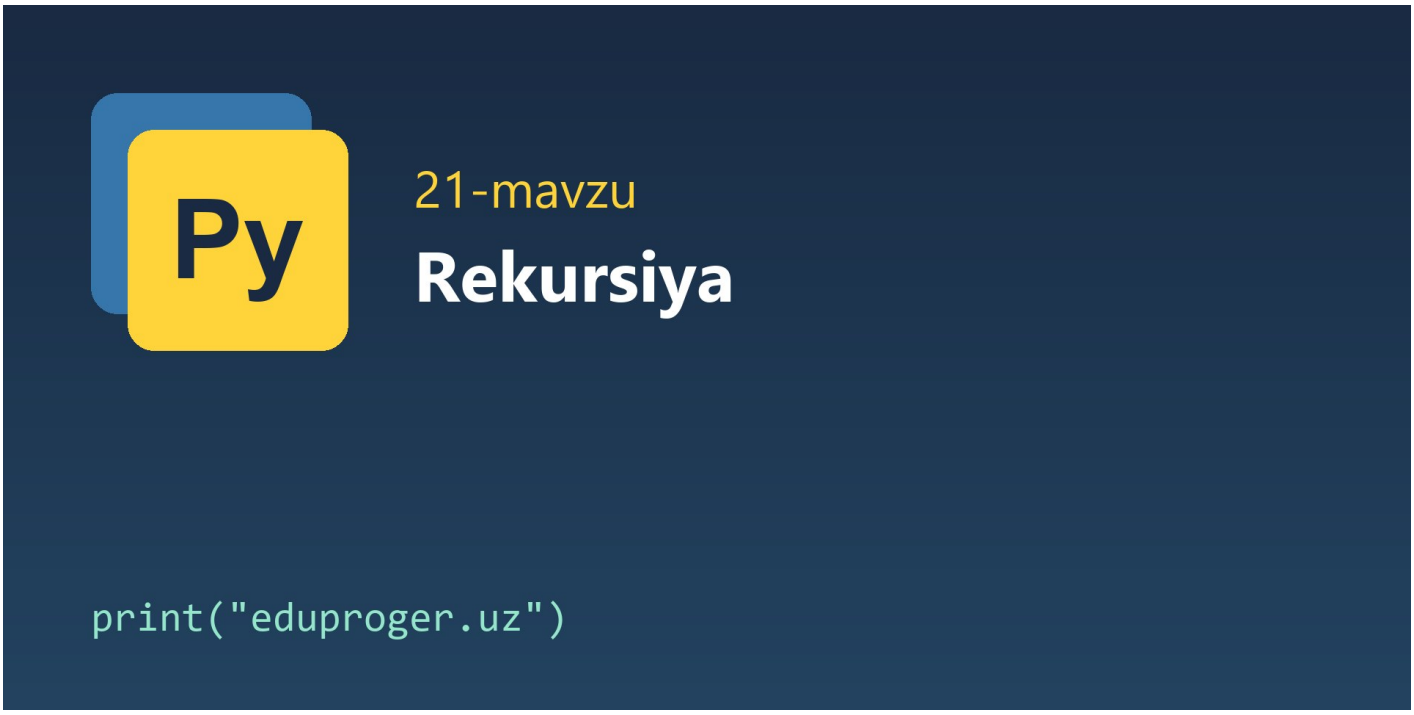
21. Rekursiya



Rekursiya — funksiyaning o'zini o'zi chaqirishi. Masalani o'zining kichikroq nusxasiga keltirib yechish g'oyasi: «n ni hisoblash uchun n−1 ni hisobla». Matematikada bunga o'xshash ta'riflar ko'p: n! = n · (n−1)!.

1. Ikki qism: asos va qadam

- **Asos holat** — rekursiya TO'XTAYDIGAN, javobi to'g'ridan-to'g'ri ma'lum holat (masalan, 0! = 1).
- **Rekursiv qadam** — masalani kichikroq masala orqali ifodalash (n! = n · (n−1)!).

```
def faktorial(n):
    if n == 0:              # asos holat
        return 1
    return n * faktorial(n - 1)   # rekursiv qadam

print(faktorial(5))
print(faktorial(20))
```

Natija:

```
120
2432902008176640000
```

`faktorial(3)` qanday hisoblanadi: `3 · faktorial(2)` → `3 · 2 · faktorial(1)` → `3 · 2 · 1 · faktorial(0)` → `3 · 2 · 1 · 1 = 6`.

2. Chaqiruvlarni kuzatish

```
def sanoq(n, chuqurlik=0):
    print("  " * chuqurlik + f"sanoq({n})")
    if n == 0:
        return
    sanoq(n - 1, chuqurlik + 1)

sanoq(3)
```

Natija:

```
sanoq(3)
  sanoq(2)
    sanoq(1)
      sanoq(0)
```

3. Klassik misollar

Raqamlar yig'indisi

```
def raqam_yig(n):
    if n < 10:
        return n
    return n % 10 + raqam_yig(n // 10)

print(raqam_yig(2026))
```

Natija:

```
10
```

Daraja

```
def daraja(a, n):
    if n == 0:
        return 1
    if n % 2 == 0:
        yarim = daraja(a, n // 2)
        return yarim * yarim       # tez daraja: qadamlar soni ~log n
    return a * daraja(a, n - 1)

print(daraja(2, 10), daraja(3, 13))
```

Natija:

```
1024 1594323
```

Satrni teskari aylantirish va palindrom

```
def teskari(s):
    if len(s) <= 1:
        return s
    return teskari(s[1:]) + s[0]
```

```
def palindrom(s):
    if len(s) <= 1:
        return True
    return s[0] == s[-1] and palindrom(s[1:-1])

print(teskari("python"), palindrom("kiyik"), palindrom("olma"))
```

Natija:

```
nohtyp True False
```

EKUB (Evklid algoritmi)

```
def ekub(a, b):
    return a if b == 0 else ekub(b, a % b)

print(ekub(48, 36), ekub(17, 5))
```

Natija:

```
12 1
```

4. Hanoy minoralari

n ta disk A ustundan C ga ko'chirilishi kerak, B — yordamchi; katta disk kichigining ustiga qo'yilmaydi. Rekursiv g'oya: n−1 diskni B ga o'tkaz, eng kattasini C ga, so'ng n−1 diskni B dan C ga:

```
def hanoy(n, a="A", b="B", c="C"):
    if n == 0:
        return 0
    k = hanoy(n - 1, a, c, b)
    print(f"disk {n}: {a} -> {c}")
    return k + 1 + hanoy(n - 1, b, a, c)

print("Jami yurishlar:", hanoy(3))
```

Natija:

```
disk 1: A -> C
disk 2: A -> B
disk 1: C -> B
disk 3: A -> C
disk 1: B -> A
disk 2: B -> C
disk 1: A -> C
Jami yurishlar: 7
```

n disk uchun $2^n - 1$ ta yurish kerak: 3 disk — 7, 64 disk — 18 kvintilliondan ortiq.

5. Fibonachchi va memoizatsiya

Oddiy rekursiv Fibonachchi bir xil qiymatlarni qayta-qayta hisoblaydi va n = 35 da ham juda sekinlashadi. Natijalarni eslab qolish (**memoizatsiya**) uni tezlashtiradi — `functools.lru_cache` buni bitta qator bilan qiladi:

```
from functools import lru_cache

chaqiruv = 0
def fib_oddiy(n):
    global chaqiruv
    chaqiruv += 1
    return n if n < 2 else fib_oddiy(n - 1) + fib_oddiy(n - 2)

print(fib_oddiy(20), "—", chaqiruv, "ta chaqiruv")

@lru_cache(maxsize=None)
def fib(n):
    return n if n < 2 else fib(n - 1) + fib(n - 2)

print(fib(100))
```

Natija:

```
6765 — 21891 ta chaqiruv
354224848179261915075
```

6. Chuqurlik chegarasi

Har chaqiruv xotirada joy egallaydi. Python standart holda ~1000 qavat chuqurlikka ruxsat beradi; asos holat unutilsa — **RecursionError**:

```
def cheksiz(n):
    return cheksiz(n + 1)

cheksiz(0)
```

Natija:

```
RecursionError: maximum recursion depth exceeded
```

> **Eslatma**
>
> Rekursiya chiroyli, lekin har doim ham eng yaxshi tanlov emas. Oddiy takrorlash (yig'indi, faktorial) uchun sikl tezroq va xavfsizroq. Rekursiya daraxtsimon masalalarda (papkalar, Hanoy, qidiruv) o'zini oqlaydi.

7. Ichma-ich ro'yxatlar

```
def yoy(royxat):
    natija = []
    for x in royxat:
        if isinstance(x, list):
```

```
            natija.extend(yoy(x))
        else:
            natija.append(x)
    return natija

print(yoy([1, [2, [3, 4]], 5, [[6]]]))
```

Natija:

```
[1, 2, 3, 4, 5, 6]
```

8. Xulosa

- Rekursiv funksiya o'zini chaqiradi; asos holat SHART — aks holda RecursionError.
- Har chaqiruvda masala kichrayib, asos holatga yaqinlashishi kerak.
- Takroriy hisoblar — `@lru_cache` bilan eslab qolinadi.
- Daraxtsimon masalalar (Hanoy, ichma-ich tuzilmalar) — rekursiyaning kuchli tomoni.