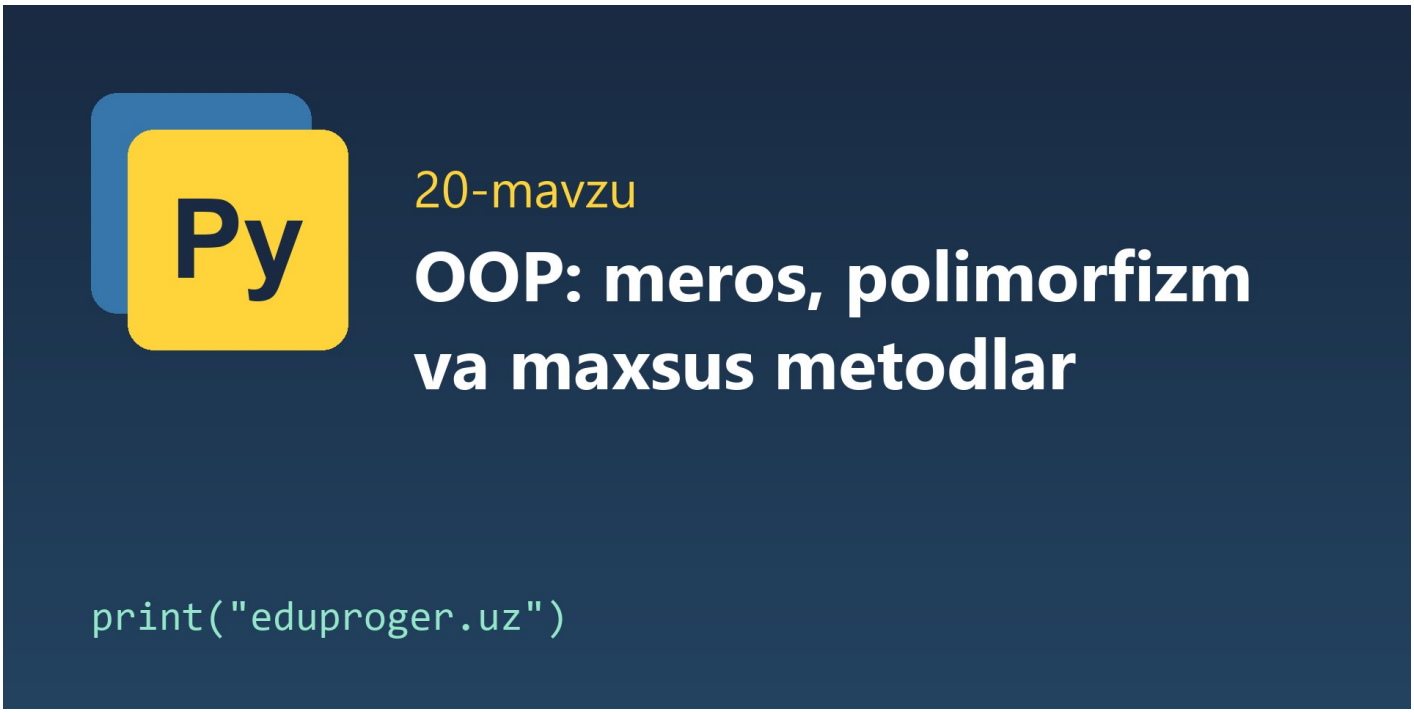
20. OOP: meros, polimorfizm va maxsus metodlar



O'tgan darsda klass va obyekt yaratishni o'rgandik. Endi OOP ning eng kuchli imkoniyatlari: bir klassdan boshqasini yasash (**meros**), bir xil buyruqqa turli obyektlarning turlicha javobi (**polimorfizm**) va obyektlarni +, ==, < kabi operatorlar bilan ishlatish (**maxsus metodlar**).

1. Meros

`class Bola(Ota):` — Bola klassi Ota klassining barcha atribut va metodlarini oladi va o'zinikini qo'shishi mumkin:

```
class Hayvon:
    def __init__(self, nom):
        self.nom = nom

    def tanishtir(self):
        return f"Men {self.nom}man"

class It(Hayvon):
    def qoriqla(self):
        return f"{self.nom} uyni qo'riqlayapti"

b = It("Bobik")
print(b.tanishtir())
print(b.qoriqla())
```

Natija:

```
Men Bobikman
Bobik uyni qo'riqlayapti
```

2. Metodni qayta yozish va super()

Bola klass ota metodini o'zgartira oladi (qayta yozish). Ota versiyasini ham chaqirish kerak bo'lsa — `super()`:

```
class Xodim:
    def __init__(self, ism, oylik):
        self.ism = ism
        self.oylik = oylik

    def maosh(self):
        return self.oylik

class Rahbar(Xodim):
    def __init__(self, ism, oylik, bonus):
        super().__init__(ism, oylik)
        self.bonus = bonus

    def maosh(self):
        return super().maosh() + self.bonus

for x in [Xodim("Ali", 5000000), Rahbar("Vali", 8000000, 2000000)]:
    print(x.ism, x.maosh())
```

Natija:

```
Ali 5000000
Vali 10000000
```

3. Polimorfizm

Turli klasslarda bir xil nomli metod bo'lsa, ularni bir xil usulda chaqirish mumkin — har biri o'z ishini qiladi:

```
import math

class Shakl:
    def yuza(self):
        return 0

class Kvadrat(Shakl):
    def __init__(self, a):
        self.a = a
    def yuza(self):
        return self.a ** 2

class Doira(Shakl):
    def __init__(self, r):
        self.r = r
    def yuza(self):
```

```
        return round(math.pi * self.r ** 2, 2)

for s in [Kvadrat(3), Doira(1), Kvadrat(5)]:
    print(type(s).__name__, s.yuza())
```

Natija:

```
Kvadrat 9
Doira 3.14
Kvadrat 25
```

4. isinstance va issubclass

```
class Hayvon: pass
class Mushuk(Hayvon): pass

m = Mushuk()
print(isinstance(m, Mushuk), isinstance(m, Hayvon), isinstance(5, Hayvon))
print(issubclass(Mushuk, Hayvon))
```

Natija:

```
True True False
True
```

5. Maxsus (dunder) metodlar

Ikki pastki chiziq bilan o'ralgan metodlar (`__init__`, `__str__` ...) Python operatorlari va funksiyalari bilan bog'langan. Ularni yozsak, o'z obyektlarimiz ham `+`, `==`, `<`, `len()` bilan ishlaydi:

Metod	Qachon chaqiriladi
`__str__`	`print(obj)`, `str(obj)`
`__eq__`	`a == b`
`__lt__`	`a < b` (va `sorted`)
`__add__`	`a + b`
`__len__`	`len(obj)`
`__contains__`	`x in obj`

```
class Vektor:
    def __init__(self, x, y):
        self.x, self.y = x, y

    def __add__(self, b):
        return Vektor(self.x + b.x, self.y + b.y)

    def __eq__(self, b):
        return self.x == b.x and self.y == b.y

    def __str__(self):
        return f"Vektor({self.x}, {self.y})"
```

```
v = Vektor(1, 2) + Vektor(3, 4)
print(v)
print(v == Vektor(4, 6))
```

Natija:

```
Vektor(4, 6)
True
```

```
class Talaba:
    def __init__(self, ism, ball):
        self.ism, self.ball = ism, ball
    def __lt__(self, b):
        return self.ball < b.ball
    def __str__(self):
        return f"{self.ism}({self.ball})"

guruh = [Talaba("Ali", 78), Talaba("Malika", 95), Talaba("Sardor", 88)]
print(*sorted(guruh))
print(max(guruh))
```

Natija:

```
Ali(78) Sardor(88) Malika(95)
Malika(95)
```

6. @property — tekshiriladigan atribut

```
class Harorat:
    def __init__(self, c):
        self.c = c

    @property
    def f(self):
        return self.c * 9 / 5 + 32

h = Harorat(25)
print(h.f)
h.c = 100
print(h.f)
```

Natija:

```
77.0
212.0
```

7. dataclass — qisqa klass

Faqat ma'lumot saqlaydigan klasslarni `dataclasses` moduli bilan juda qisqa yozish mumkin — `__init__`, `__eq__` va chiroyli ko'rinish avtomatik yaratiladi:

```
from dataclasses import dataclass

@dataclass
class Mahsulot:
```

```
    nom: str
    narx: int
    soni: int = 1

m = Mahsulot("daftar", 5000, 3)
print(m)
print(m.narx * m.soni, m == Mahsulot("daftar", 5000, 3))
```

Natija:

```
Mahsulot(nom='daftar', narx=5000, soni=3)
15000 True
```

8. Xulosa

- Meros: `class Bola(Ota)`; ota metodini chaqirish — `super()`.
- Polimorfizm: bir xil nomli metod, turli klasslarda turli ish.
- Maxsus metodlar: `__add__`, `__eq__`, `__lt__`, `__len__`, `__str__` — operatorlar bilan ishlash.
- `@property` — hisoblanadigan atribut; `@dataclass` — qisqa ma'lumot klassi.