

Za radoznale: Klase i objekti

Ovo nije obavezno i neće biti na ispitu. Za one koji žele da vide kako se ista stvar radi naprednije. Svi ispisi su dobijeni pokretanjem u pygame-ce.

1. repr: lepa lista objekata

Na času smo videli da `print(pahuljica)` koristi **str**, ali `print(lista_pahuljica)` i dalje ispisuje adrese. Lista za svoje elemente koristi drugu posebnu metodu, **repr**. Ona treba da vrati string koji liči na kod kojim se objekat pravi.

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

    def __str__(self):
        return f"pahuljica u ({self.x}, {self.y})"

    def __repr__(self):
        return f"Pahuljica({self.x}, {self.y})"

pahuljice = [Pahuljica(60, 40), Pahuljica(150, 90)]
print(pahuljice[0])
print(pahuljice)
```

Ispis:

```
pahuljica u (60, 40)
[Pahuljica(60, 40), Pahuljica(150, 90)]
```

Prvi print koristi **str**, a drugi ispisuje listu, pa za svaki element koristi **repr**. Ako klasa ima samo **repr**, Python ga koristi i umesto **str**.

Zašto ovo nismo radili na času: za ispis jednog objekta dovoljan je **str**; **repr** je koristan kad često gledamo cele liste objekata dok tražimo grešku.

Fajl: `primeri/za_radoznale/02_r1_repr.py`

2. Zamka: lista kao podrazumevana vrednost

Podrazumevana vrednost parametra napravi se samo jednom, kada Python čita definiciju metode, a ne pri svakom pozivu. Za brojeve i stringove to nije važno, jer se ne menjaju. Za listu jeste: svi objekti napravljeni bez argumenta dobijaju istu listu.

```

class Lik:
    def __init__(self, ime, predmeti=[]):    # zamka
        self.ime = ime
        self.predmeti = predmeti

a = Lik("Ana")
b = Lik("Mia")
a.predmeti.append("šal")
print(b.predmeti, a.predmeti is b.predmeti)

class DobarLik:
    def __init__(self, ime, predmeti=None):
        self.ime = ime
        self.predmeti = predmeti if predmeti is not None else []

a = DobarLik("Ana")
b = DobarLik("Mia")
a.predmeti.append("šal")
print(b.predmeti)

```

Ispis:

```

['šal'] True
[]

```

U prvoj klasi Mia „dobija“ Anin šal, jer obe imaju istu listu (is daje True). U drugoj klasi podrazumevana vrednost je None, a nova prazna lista pravi se u **init**, posebno za svaki objekat.

Zašto ovo nismo radili na času: ovo je ista ideja kao reference iz teme 01, ali skrivena u definiciji metode; slična zamka sa listom kao atributom klase je čudovište teme 03.

Fajl: `primeri/za_radoznale/02_r2_lista_kao_podrazumevana_vrednost.py`

3. dataclass: Python sam piše init

Kada klasa uglavnom samo čuva podatke, Python može sam da napiše **init** i **repr**. Iznad klase stavimo `@dataclass` (to je dekorator, oznaka koja menja klasu) i navedemo attribute sa tipovima.

```

from dataclasses import dataclass

@dataclass
class Slika:
    naslov: str
    autor: str
    godina: int = 2026

    def starost(self, ove_godine):
        return ove_godine - self.godina

s = Slika("Zimsko jutro", "Ana Ilić", 2019)
print(s)
print(s.starost(2026))
print(Slika("Pahulje", "Mia Petrović"))

```

Ispis:

```
Slika(naslov='Zimsko jutro', autor='Ana Ilić', godina=2019)
7
Slika(naslov='Pahulje', autor='Mia Petrović', godina=2026)
```

Nismo pisali **init**, a objekat ipak dobija naslov, autora i godinu; godina ima podrazumevanu vrednost. print koristi **repr** koji je Python napisao sam. Metode pišemo kao i do sada.

Zašto ovo nismo radili na času: da bismo razumeli šta se dešava u pozadini, prvo je trebalo da **init** napišemo sami.

Fajl: `primeri/za_radoznale/02_r3_dataclass.py`