

Za radoznale: Vežbanje i priprema za ispit

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. assert: program sam proverava svoju logiku

Pre ispita ili predaje rada korisno je proveriti delove igre bez prozora, kao što smo radili u pitanjima „Šta ispisuje program?“. Naredba `assert` proverava uslov: ako je tačan, ne dešava se ništa, a ako nije, program staje sa greškom `AssertionError` i porukom koju smo napisali.

```
class Igrac:
    def __init__(self):
        self.zivoti = 3
        self.zastita = 0

    def udaren(self):
        if self.zastita > 0:
            return False
        self.zivoti -= 1
        self.zastita = 1.5
        return True

i = Igrac()
assert i.udaren() == True, "prvi udarac oduzima život"
assert i.udaren() == False, "dok traje zaštita, udarac ne vredi"
assert i.zivoti == 2, "ostala su dva života"
print("sve provere su prošle")
```

Ispis:

```
sve provere su prošle
```

Ovakve provere se pišu jednom i pokreću posle svake izmene. Ako neko kasnije promeni metodu `udaren` tako da zaštita više ne radi, druga provera to odmah javi, pre nego što igra uopšte otvori prozor.

Zašto ovo nismo radili na času: na času smo proveravali pokretanjem i gledanjem; automatske provere su sledeći korak, kad program postane veliki.

Fajl: `primeri/za_radoznale/13_r1_assert.py`

2. @dataclass: klasa bez pisanja init

Mnoge naše klase samo čuvaju podatke: `init` prima vrednosti i upisuje ih u `self`. Modul `dataclasses` iz standardne biblioteke to piše umesto nas. Dobija se i lep ispis i poređenje po vrednostima, pa dva predmeta sa istim vrednostima postaju jednaka.

```

from dataclasses import dataclass

@dataclass
class Predmet:
    ime: str
    tezina: float

a = Predmet("Ledeni mač", 3)
b = Predmet("Ledeni mač", 3)
print(a)
print(a == b)

```

Ispis:

```

Predmet(ime='Ledeni mač', tezina=3)
True

```

Iznad klase stoji @dataclass, a atributi se nabrajaju sa tipom. Uporedite sa turnirskim pitanjem iz teme 02: tamo su dve pahuljice sa istim x bile različite, jer obična klasa poredi da li je to isti objekat. Klasa sa @dataclass poredi vrednosti atributa.

Zašto ovo nismo radili na času: na času smo učili šta **init**, **str** i **eq** zaista rade; dataclass ih piše sam, pa je zgodan tek kad to već razumete.

Fajl: `primeri/za_radoznale/13_r2_dataclass.py`

3. repr: kako objekat izgleda u listi

print(objekat) koristi **str**. Kad isti objekat stoji u listi, print(lista) za svaki element koristi drugu specijalnu metodu, **repr**. Ako je nema, vidi se samo ime klase i adresa u memoriji, što ne pomaže pri traženju greške.

```

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

    def __str__(self):
        return f"pahuljica na x = {self.x}"

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

sneg = [Pahuljica(10), Pahuljica(50)]
print(sneg[0])
print(sneg)

```

Ispis:

```

pahuljica na x = 10
[Pahuljica(10), Pahuljica(50)]

```

str je za čoveka koji igra, a **repr** za programera koji traži grešku: obično izgleda kao poziv kojim bi se napravio isti objekat. Kad ispisujete grupu sprajtova ili listu scena dok tražite grešku, **repr** vam kaže šta je unutra.

Zašto ovo nismo radili na času: za ispis jednog objekta dovoljan je **str**, koji smo radili u temama 02 i 05.

Fajl: `primeri/za_radoznale/13_r3_repr.py`