

Klase i objekti

Pravimo sopstvene objekte: šablon jednom, primeraka koliko hoćemo

Danas pravimo: klasu Pahuljica i trideset pahuljica u listi



Svaka pahuljica je objekat: zna gde je, koliko je velika i ume da se nacрта

Tri pitanja za zagrevanje

1 • prošla tema: reference

```
l = [3, 1, 2]
m = l
m.sort()
print(l)
```

[1, 2, 3]

2 • PZU1: f-string

```
ime = "Ana"
print(f"{ime}: {len(ime)}")
```

Ana: 3

3 • mešano: rečnik i funkcija

```
def opis(d):
    return d["ime"] + "!"

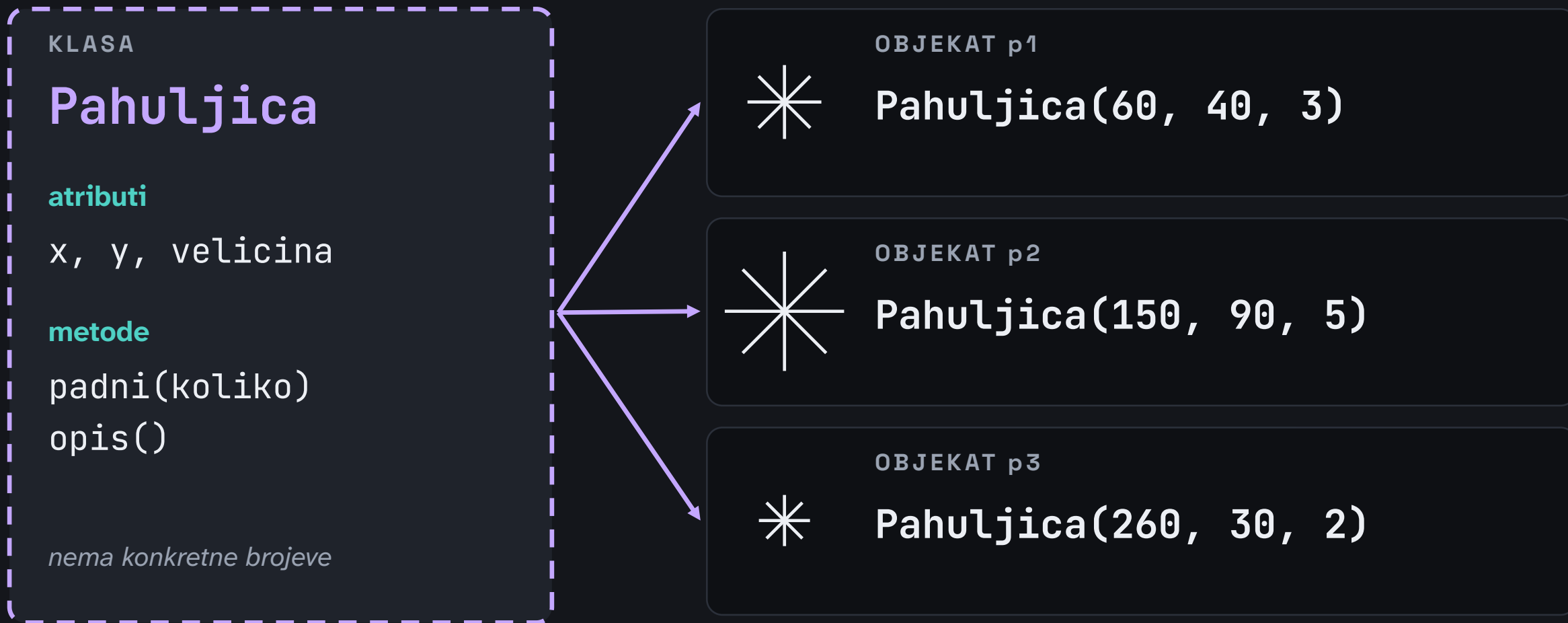
print(opis({"ime": "Led"}))
```

Led!

Sedam novih reči za danas

klasa	šablon (nacrt) po kome se prave objekti: <code>class Pahuljica</code>
objekat (instanca)	jedan primerak napravljen po šablonu
atribut	podatak koji objekat pamti: <code>p.x</code> , <code>p.velicina</code>
metoda	funkcija napisana u klasi: <code>p.padni(10)</code>
konstruktor	poziv klase koji pravi objekat: <code>Pahuljica(60, 40, 3)</code>
<code>__init__</code>	inicijalizator: postavlja početne attribute novog objekta
<code>self</code>	objekat nad kojim je metoda pozvana

Klasa je šablon, a objekat je jedan primerak



Objekat je imenica, atribut pridev, metoda glagol

imenica

u kodu: objekat

pahuljica, Čikica, jelka

```
p = Pahuljica(60, 40, 3)
```

pridev

u kodu: atribut

mala, bela, na vrhu prozora

```
p.velicina, p.y
```

glagol

u kodu: metoda

pada, crta se, opisuje se

```
p.padni(10)
```

U rečenici „mala pahuljica pada“ reč mala je atribut p.velicina, a pada je metoda p.padni().

Prva klasa

class, `__init__` i `self`: kako nastaje objekat

class pravi novu klasu, a poziv klase nov objekat

```
class Pahuljica:  
    pass
```

```
p = Pahuljica()  
q = Pahuljica()  
print(type(p))  
print(p is q)
```

ISPIS

```
<class '__main__.Pahuljica'>  
False
```

- Ime klase počinje velikim slovom: Pahuljica.
- pass znači: telo je prazno.
- p i q su dva objekta, pa p is q nije tačno.

Atribut možemo dodati tačkom, ali to je zamka

```
class Pahuljica:
    pass

p = Pahuljica()
p.x = 60
p.y = 40
q = Pahuljica()
q.x = 150
print(p.x, p.y, q.x)
print(q.y)
```

ISPIS

60 40 150

PORUKA

```
AttributeError:
'Pahuljica' object has no
attribute 'y'
```

Objekat q nema atribut y, jer ga niko nije postavio. Rešenje: attribute postavljamo u `__init__`, pa ih svaki objekat ima.

`__init__` postavlja attribute svakom novom objektu

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

```
p1 = Pahuljica(60, 40, 3)  
p2 = Pahuljica(150, 90, 5)  
print(p1.x, p2.x)  
print(p1.velicina + p2.velicina)
```

ISPIS

```
60 150  
8
```

`self.x = x`

levo je atribut objekta, desno
parametar metode. Isto ime, dve
različite stvari.

Šta se desi kad napišemo Pahuljica(60, 40, 3)?

- 1 Python napravi nov, prazan objekat klase Pahuljica.
- 2 Pozove `__init__`(taj objekat, 60, 40, 3): `self` je taj novi objekat.
- 3 Redovi `self.x = x ...` upisuju attribute u objekat.
- 4 Konstruktor vrati objekat, a ime `p1` počne da pokazuje na njega.

Konstruktor je poziv klase; `__init__` (inicijalizator) je samo drugi korak.

Šta ispisuje ovaj program?

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

a = Pahuljica(0, 0)
b = Pahuljica(0, 0)
a.y = 50
print(a.y, b.y)
```

A

50 50

B

tačno

50 0

C

0 0

D

AttributeError

self i metode

Šta objekat ume da uradi

Osoba iz starih beleški: metoda jasm

```
class Osoba:
    def __init__(self, ime, prezime, starost):
        self.ime = ime
        self.prezime = prezime
        self.starost = starost
        self.zanimanje = "student"

    def jasm(self):
        print(f"Ja sam {self.ime}, {self.zanimanje}.")

osoba1 = Osoba("Jovan", "Jovanović", 20)
osoba1.jasm()
osoba1.ime = "Aleksandar"
osoba1.jasm()
```

ISPIS

```
Ja sam Jovan, student.
Ja sam Aleksandar, student.
```

- zanimanje ima unapred zadatu vrednost.
- osoba1.ime = ... menja atribut, a metoda jasm to odmah vidi.

self je objekat nad kojim je metoda pozvana

```
class Osoba:
    def __init__(self, ime):
        self.ime = ime

    def jasm(self):
        print("Ja sam", self.ime)

ana = Osoba("Ana")
ana.jasm()
Osoba.jasm(ana)
```

ISPIS

```
Ja sam Ana
Ja sam Ana
```

ana.jasm()

Python izvršava

**Osoba.jasm(ana)**

Objekat ispred tačke postaje prvi argument, self. Zato self pišemo kao prvi parametar svake metode.

Čudovište: zaboravljen self

```
class Pahuljica:  
    def __init__(self, x):  
        self.x = x  
  
    def opisi():  
        print("pahuljica")
```

```
p = Pahuljica(10)  
p.opisi()
```

TypeError:
Pahuljica.opisi() takes 0
positional arguments but 1 was
given

Kako izgleda

TypeError: ... takes 0 positional arguments but 1 was given, pri pozivu metode.

Zašto se pojavljuje

Poziv p.opisi() uvek šalje objekat p kao prvi argument, a metoda nema mesto za njega.

Kako se pobeđuje

Svaka metoda u klasi počinje parametrom self: def opisi(self):

Jedna metoda menja objekat, druga vraća opis

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

```
    def padni(self, koliko):  
        self.y += koliko
```

```
    def opis(self):  
        return f"{self.velicina} px u ({self.x}, {self.y})"
```

```
p = Pahuljica(60, 40, 3)  
p.padni(10)  
p.padni(5)  
print(p.opis())
```

ISPIS

3 px u (60, 55)

- padni menja self.y.
- opis ne menja ništa: vraća string sa return.
- Posle dva poziva padni y je 55.

Šta ispisuje ovaj program?

```
class Lik:  
    def __init__(self, ime):  
        self.ime = ime  
  
    def pozdrav(self):  
        print("Zdravo,", self.ime)  
  
c = Lik("Ana")  
print(c.pozdrav())
```

A

Zdravo, Ana

B

tačno

Zdravo, Ana
None

C

None

D

TypeError

Podrazumevane vrednosti važe i u `__init__`

```
class Pravougaonik:
    def __init__(self, sirina=10, visina=10):
        self.sirina = sirina
        self.visina = visina

    def obim(self):
        return 2 * (self.sirina + self.visina)

    def povrsina(self):
        return self.sirina * self.visina

p1 = Pravougaonik()
p2 = Pravougaonik(20, 30)
p3 = Pravougaonik(visina=5)
print(p1.povrsina(), p2.povrsina(), p3.povrsina())
print(p2.obim())
```

ISPIS

```
100 600 50
100
```

- `Pravougaonik()` je 10×10 .
- `visina=5` je imenovani argument; širina ostaje 10.
- Zadatak iz starih beleški, sa ispravljenim ispisom.

`__str__` i lista objekata

Kako se objekat ispisuje i kako ih pravimo mnogo

__str__ kaže kako print ispisuje objekat

Bez __str__

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

```
p = Pahuljica(60, 40)
print(p)
```

ISPIS

```
<__main__.Pahuljica object at 0x7fb1d2b36900>
```

Sa __str__

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

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

```
p = Pahuljica(60, 40)
print(p)
print("Nova: " + str(p))
```

ISPIS

```
Pahuljica u (60, 40)
Nova: Pahuljica u (60, 40)
```

Deset pahuljica u listi, jedna metoda za sve

```
pahuljice = []  
for i in range(10):  
    p = Pahuljica(i * 60, 0, 2 + i % 3)  
    pahuljice.append(p)  
  
for p in pahuljice:  
    p.padni(p.velicina * 10)  
  
print(len(pahuljice))  
print(pahuljice[3])  
print(pahuljice[4].y, pahuljice[-1].x)
```

ISPIS

```
10  
Pahuljica 2 px u (180, 20)  
30 540
```

- Klasa Pahuljica je ista kao na prošlim slajdovima, sa `__str__`.
- `i * 60` ih raspoređuje po širini prozora.
- Petlja poziva `padni` nad svakim objektom.

Šta ispisuje ovaj program?

```
a = Pahuljica(0, 0, 2)
pahuljice = [a, a]
for p in pahuljice:
    p.padni(10)
print(a.y)
```

Klasa Pahuljica je ista kao na prošlom slajdu; y počinje od 0.

A
10

B tačno
20

C
0

D
TypeError

Student iz starih beleški: indeks GGGGXXXX

```
class Student:
    def __init__(self, ime, prezime, indeks, prosek=0):
        self.ime = ime
        self.prezime = prezime
        self.indeks = indeks      # GGGGXXXX, npr. 20230107
        self.prosek = prosek

    def formatirani_indeks(self):
        godina = self.indeks // 10000
        broj = self.indeks % 10000
        return f"{broj:04d}/{godina}"

    def __str__(self):
        indeks = self.formatirani_indeks()
        return f"{self.ime} {self.prezime} ({indeks}), {self.prosek}"
```

- // celobrojno deljenje: godina.
- % ostatak: broj studenta.
- :04d dopunjuje nulama: 0107.

Atribut dodat posle pravljenja ima samo taj objekat

```
s1 = Student("Jovana", "Petrović", 20230107, 9.5)
s2 = Student("Marko", "Nikolić", 20240256)
print(s1)
print(s2)
s2.mesto_rodjenja = "Novi Sad"
print(hasattr(s1, "mesto_rodjenja"))
print(hasattr(s2, "mesto_rodjenja"))
print(s2.mesto_rodjenja)
```

ISPIS

```
Jovana Petrović (0107/2023), 9.5
Marko Nikolić (0256/2024), 0
False
True
Novi Sad
```

- s2 nema prosek u pozivu, pa dobija 0.
- `hasattr(objekat, "ime")` pita: ima li objekat taj atribut?
- Bolje je sve attribute postaviti u `__init__`.

Pahuljice kao objekti u prozoru

Cilj: za 20 minuta klasa Pahuljica ume da se nacрта, a trideset objekata crta zimsko nebo.

- 1 Klasa: x, y, velicina i boja
- 2 Metoda `__str__` za proveru u konzoli
- 3 Metoda `crtaj(povrsina)`: krst i dijagonale
- 4 Lista od 30 objekata na slućajnim mestima
- 5 Petlja: `p.crtaj(ekran)` za svaku pahuljicu

KAKO IZGLEDA NA KRAJU



U temi 07 iste pahuljice poćinju da padaju.

Metoda crtaj dobija površinu na kojoj crta

```
def crtaj(self, površina):  
    v = self.velicina  
    d = round(v * 0.7) # krak ukoso je kraći  
    for (dx, dy) in [(v, 0), (0, v), (d, d), (d, -d)]:  
        od = (self.x - dx, self.y - dy)  
        do = (self.x + dx, self.y + dy)  
        pg.draw.line(površina, self.boja, od, do, 2)  
    pg.draw.circle(površina, self.boja, (self.x, self.y), 2)
```



Svaka torka (dx, dy) daje jednu liniju od (x - dx, y - dy) do (x + dx, y + dy).

(v, 0)

(0, v)

(d, d)

(d, -d)

Lista objekata, pa crtanje u jednoj petlji

```
pahuljice = []
for i in range(30):
    x = random.randint(10, SIRINA - 10)
    y = random.randint(10, 280)
    v = random.randint(3, 9)
    pahuljice.append(Pahuljica(x, y, v, paleta["sneg"]))
print(len(pahuljice), "pahuljica; prva:", pahuljice[0])

ekran.fill(paleta["nebo"])
pg.draw.ellipse(ekran, paleta["sneg"], (-100, 290, 840, 160))
for p in pahuljice:
    p.crtaj(ekran)
pg.display.flip()
```



- `p.crtaj(ekran)`: `self` je `p`, a površina je `ekran`.
- `print(pahuljice[0])` koristi `__str__`.

Pronađi grešku: pahuljica bez y

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

```
p = Pahuljica(60, 40)  
print(p.x, p.y)
```

Konstruktor dobija i x i y. Zašto Python kaže da y ne postoji?

PORUKA

```
AttributeError:  
'Pahuljica' object has no  
attribute 'y'
```

y = y

dodeljuje parametru njegovu vrednost;
objekat ne dobija ništa. Ispravka:

self.y = y

Izaberite nivo i počnite

LEVEL 1

Klasa Lik

Lik ima ime i živote (podrazumevano 3), metodu izgubi_zivot i __str__.

Pomoć: životi ne smeju ispod nule.

LEVEL 2

Galerija

Klasa Slika (naslov, autor, godina) i lista od tri slike. Ispišite sve i najstariju.

Pomoć: zapamtite najstariju u petlji.

LEVEL 3

Zvezdano nebo

Klasa Zvezda sa metodom crtaj; 80 zvezda u prozoru i Mesec.

Pomoć: kao Pahuljica, sa dve linije i krugom.

Sledeći put: atribut koji dele svi objekti

```
class Pahuljica:
    ukupno = 0          # atribut klase

    def __init__(self, x, y):
        self.x = x
        self.y = y
        Pahuljica.ukupno += 1

a = Pahuljica(1, 2)
b = Pahuljica(3, 4)
print(Pahuljica.ukupno)
```

U temi 03 videćemo attribute koji pripadaju klasi, a ne jednom objektu, kao brojač svih napravljenih pahuljica. Ovaj program ispisuje 2.

Rezime

klasa i objekat

```
class Pahuljica:  
    p = Pahuljica(60, 40, 3)
```

Šablon jednom, primeraka mnogo.

__init__ i self

```
def __init__(self, x):  
    self.x = x
```

self je objekat koji se pravi.

metode

```
def padni(self, k):  
    p.padni(10)
```

Prvi parametar je uvek self.

__str__

```
def __str__(self):  
    return "..."
```

print(p) koristi __str__.

Pitanje pre odlaska

```
class Brojac:  
    def __init__(self):  
        self.n = 0  
  
    def dodaj(self):  
        self.n += 1  
  
a = Brojac()  
b = Brojac()  
a.dodaj()  
a.dodaj()  
b.dodaj()  
print(a.n, b.n)
```

Šta ispisuje poslednji red?

Pošaljite dva broja pre nego što izađete.



Literatura

- Bunn, T. Learn Python Visually: Creative Coding with Processing.py, No Starch Press, 2021 (iz nje je ideja crteža „klasa kao šablon“)
- Lutz, M. Learning Python, 5. izdanje, O'Reilly Media, 2013 (deo „Classes and OOP“)
- Wentworth, P., Elkner, J., Downey, A. B., Meyers, C. How to Think Like a Computer Scientist: Learning with Python 3 (besplatna knjiga na internetu)
- Zvanični Python tutorijal, poglavlje 9 „Classes“: docs.python.org/3/tutorial/classes.html

Dodatna literatura i zanimljivi sajtovi

Python: klase

docs.python.org/3/tutorial/classes.html

Zvanični tutorijal: `__init__`, `self`, atributi i metode.

Python Tutor

pythontutor.com

Crta objekte i njihove attribute korak po korak.

Program Arcade Games: klase

programarcadegames.com · [klase](#)

Poglavlje besplatne knjige o igrama: klase za lika i loptu.

pygame-ce: draw

pyga.me/docs

Dokumentacija za pygame-ce: line, circle i ostale funkcije za crtanje.

Hvala!

Pitanja su uvek dobrodošla: na času, u četvrtu ili međim.

Slajdovi, primeri i kartica za podsećanje su na sajtu predmeta.

PREDAVANJA

dr Staša Vujičić Stanković

stasa.vujicic.stankovic@math.rs

www.matf.bg.ac.rs/~stasa.vujicic.stankovic

VEŽBE

Milan Kocić

milan.kocic@matf.bg.ac.rs

poincare.matf.bg.ac.rs/~milan.kocic