

# Napredno programiranje i programski jezici

## 12 Python

Fakultet tehničkih nauka, Novi Sad  
23-24/Z  
Dunja Vrbaški

- objektno orijentisano programiranje
- klase, objekti
  
- dosadašnji pojmovi OOP
- sintaksa, semantika, izvršavanje

Sve su objekti

- sve ima tip:
  - podaci
  - načini za izmenu tih podataka

42 je instanca tipa int

“dobar dan” je instanca tipa string

možemo definisati svoje klase, svoje tipove i instancirati objekte tog tipa

```
class Pravougaonik:
```

```
    # polja i metode
```

```
class Pravougaonik:
```

```
    def info(self):  
        print("Ja sam pravougaonik")
```

*pravougaonik.py*

```
import pravougaonik as pr
```

```
p = pr.Pravougaonik()  
p.info()
```

*test.py*

Ja sam pravougaonik

```
class Pravougaonik:
```

```
    def init(self, a, b):  
        self.a = a  
        self.b = b
```

```
    def info(self):  
        print("Ja sam pravougaonik")
```

- konstruktor
- dve donje crte
- definišemo podatke (polja, atributi)

```
class Pravougaonik:
```

```
    def init (self, a, b):
```

```
        self.a = a
```

```
        self.b = b
```

```
    def info(self):
```

```
        print("Ja sam pravougaonik")
```

Ja sam pravougaonik

```
p1 = pr.Pravougaonik()
```

```
p1.info()
```

```
p2 = pr.Pravougaonik(3, 5)
```

```
p2.info()
```

```
class Pravougaonik:
```

```
    def __init__(self, a, b):
```

```
        self.a = a
```

```
        self.b = b
```

```
    def info(self):
```

```
        print(f"Ja sam pravougaonik: {a} {b}")
```

```
        print(f"Ja sam pravougaonik: {self.a} {self.b}")
```



```
class Pravougaonik:
```

```
    def __init__(self, a, b):  
        self.a = a  
        self.b = b
```

```
    def info(self):  
        print(f"Ja sam pravougaonik: {self.a} {self.b}")
```

```
p = pr.Pravougaonik(3, 5)  
p.info()
```

```
Ja sam pravougaonik: 3 5
```

self  
eksplicitno u definiciji  
implicitno pri pozivu (p → self)

```
class Pravougaonik:
    def __init__(self, a, b):
        self.a = a
        self.b = b

    def info(self):
        print(f"Ja sam pravougaonik: {self.a} {self.b}")
```

```
Ja sam pravougaonik: 3 5
Ja sam pravougaonik: 3 5
```

```
p = pr.Pravougaonik(3, 5)

p.info()                # instanca.metod

pr.Pravougaonik.info(p) # modul.klasa.metod(instancja)
```

```
class Pravougaonik:

    def __init__(self, a, b):
        self.a = a
        self.b = b

    def info(self):
        print(f"Ja sam pravougaonik: {self.a} {self.b}")

    def info2(x):
        print(f"Ja sam pravougaonik: {x.a} {x.b}")
```

```
p = pr.Pravougaonik(3, 5)
p.info()
p.info2()
```

```
Ja sam pravougaonik: 3 5
Ja sam pravougaonik: 3 5
```

self - konvencija (pridrřavajte se)

```
class Pravougaonik:
    def __init__(self, a, b):
        self.a = a
        self.b = b

    def info(self):
        print(f"Ja sam pravougaonik: {self.a} {self.b}")
```

```
p = pr.Pravougaonik(3, 5)
p.info()

print(p)
```

```
Ja sam pravougaonik: 3 5
<pravougaonik.Pravougaonik object at
0x0000028C4A52FB10>
```

```
class Pravougaonik:

    def __init__(self, a, b):
        self.a = a
        self.b = b

    def info(self):
        print(f"Ja sam pravougaonik: {self.a} {self.b}")

    def str(self):
        return f"Pravougaonik: a = {self.a}, b = {self.b}"
```

```
p = p.Pravougaonik(3, 5)
p.info()

print(p)
```

```
Ja sam pravougaonik: 3 5
Pravougaonik: a = 3, b = 5
```

```
class Pravougaonik:
```

```
    def __init__(self, a = 1, b = 2):  
        self.a = a  
        self.b = b
```

```
    def __str__(self):  
        return f"Pravougaonik: a = {self.a}, b = {self.b}"
```

```
p = pr.Pravougaonik(3, 5)  
print(p)
```

```
p = pr.Pravougaonik()  
print(p)
```

```
Pravougaonik: a = 3, b = 5
```

```
Pravougaonik: a = 1, b = 2
```

```
class Pravougaonik:
```

```
    def __init__(self):
```

```
        self.a = 1
```

```
        self.b = 2
```

```
    def __str__(self):
```

```
        return f"Pravougaonik: a = {self.a}, b = {self.b}"
```

```
Pravougaonik: a = 1, b = 2
```

```
p = pr.Pravougaonik(3, 5)
```

```
print(p)
```

```
p = pr.Pravougaonik()
```

```
print(p)
```

```
class Pravougaonik:

    def __init__(self, a, b):
        self.a = a
        self.b = b

    def __str__(self):
        return f"Pravougaonik: a = {self.a}, b = {self.b}"
```

```
p = p.Pravougaonik(3, 5)
print(p)

p.a += 1
print(p)
```

```
Ja sam pravougaonik: 3 5
Ja sam pravougaonik: 4 5
```

public



```
class Pravougaonik:

    def __init__(self, a, b):
        self.a = a
        self.b = b
        ...

    def get_a(self):
        return self.a

    def get_b(self):
        return self.b

    def set_a(self, a):
        self.a = a

    def set_b(self, b):
        self.b = b
```

```
setattr, getattr,  
property, getter, setter,  
@property
```

```
class Pravougaonik:
    def __init__(self, a, b):
        ...

    def get_a(self):
        return self.a

    def get_b(self):
        return self.b

    def set_a(self, a):
        self.a = a

    def set_b(self, b):
        self.b = b
```

```
Pravougaonik: a = 3, b = 5
Pravougaonik: a = 4, b = 5
4 4
```

```
p = p.Pravougaonik(3, 5)

p.a += 1
p.info()

print(p.get_a(), p.a)
```

Zašto get i set?

```

class Pravougaonik:
    def __init__(self, a, b):
        self.a = a
        self.b = b

    ...

    def get_a(self):
        return self.a

    def get_b(self):
        return self.b

    def set_a(self, a):
        self.a = a

    def set_b(self, b):
        self.b = b

```

```

p = Pravougaonik(3, 5)

p.a += 1
p.info()

print(p.get_a(), p.a)

```

```

class Pravougaonik:
    def __init__(self, a = 1, b = 2):
        self.duzina = a
        self.sirina = b

    ...

    def get_a(self):
        return self.a

    def get_b(self):
        return self.b

    def set_a(self, a):
        self.a = a

    def set_b(self, b):
        self.b = b

```

```

p = Pravougaonik(3, 5)

p.a += 1
p.info()

print(p.get_a(), p.a)

```

Možemo sakriti.

Priča o donjim crticama → u nastavku

```
self._a  
self.__a
```

```
class Pravougaonik:
```

```
    br_instanci = 0
```

```
    def __init__(self, a, b):
```

```
        self.a = a
```

```
        self.b = b
```

```
        Pravougaonik.br_instanci += 1
```

```
print(pr.Pravougaonik.br_instanci)
```

```
p = pr.Pravougaonik(3, 5)
```

```
print(pr.Pravougaonik.br_instanci)
```

```
0
```

```
1
```

```
class Pravougaonik:
```

```
    br_instanci = 0
```

```
    def __init__(self, a, b):
```

```
        self.a = a
```

```
        self.b = b
```

```
        Pravougaonik.br_instanci += 1
```

0

1

```
x = pr.Pravougaonik
```

```
print(x.br_instanci)
```

```
p = x(3, 5)
```

```
print(x.br_instanci)
```

```
class Pravougaonik:

    br_instanci = 0

    def __init__(self, a, b):
        self.a = a
        self.b = b
        Pravougaonik.br_instanci += 1
```

```
x = pr.Pravougaonik
```

```
print(x.br_instanci)
p = x(3, 5)
print(x.br_instanci)
```

0

1

sve je objekat pa i sama klasa

x je tipa ili klase “tip”

p je tipa ili klase “Pravougaonik”

```
class Pravougaonik:
```

```
    br_instanci = 0
```

```
    def __init__(self, a, b):
```

```
        self.a = a
```

```
        self.b = b
```

```
        Pravougaonik.br_instanci += 1
```

```
<class 'type'> <class 'pravougaonik.Pravougaonik'>
```

```
x = pr.Pravougaonik
```

```
p = x(3, 5)
```

```
print(type(x), type(p))
```



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

    def __str__(self):
        return f"Osoba: {self.ime} {self.prezime}"

osoba = Osoba("Petar", "Petrovic")
print(osoba)
```

Osoba: Petar Petrovic

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

    def __str__(self):
        return f"Osoba: {self.ime} {self.prezime}"

class Student(Osoba):
    def __init__(self, ime, prezime, bri):
        super().__init__(ime, prezime)
        self.bri = bri

    def __str__(self):
        return f"Student: {self.ime} {self.prezime} {self.bri}"

osoba = Osoba("Petar", "Petrovic")
print(osoba)

student = Student("Petar", "Petrovic", 123)
print(student)
```

```
Osoba: Petar Petrovic
Student: Petar Petrovic 123
```

```
class Figura
    __init__

    povrsina

class Pravougaonik

    __init__

    povrsina
```

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

    def povrsina(self):
        pass

class Pravougaonik(Figura):

    def __init__(self, ime, a, b):
        super().__init__(ime)
        self.a = a
        self.b = b

    def povrsina(self):
        return self.a * self.b

    def __str__(self):
        return f"Ja sam pravougaonik ime = {self.ime}, P = {self.povrsina()}"
```

```
import figura as fig

p = fig.Pravougaonik("P1", 1, 2)
print(p)
```

```

class Figura:
    def __init__(self, ime):
        self.ime = ime

    def povrsina(self):
        pass

class Pravougaonik(Figura):

    def __init__(self, ime, a, b):
        super().__init__(ime)
        self.a = a
        self.b = b

    def povrsina(self):
        return self.a * self.b

    def __str__(self):
        return f"Ja sam pravougaonik ime = {self.ime}, P = {self.povrsina()}"

```

Ja sam pravougaonik ime = Pravougaonik 1, P = 2

```

import figura as fig

p = fig.Pravougaonik("P1", 1, 2)
print(p)

```

```

class Figura:
    def __init__(self, ime):
        self.ime = ime

    def povrsina(self):
        pass

class Pravougaonik(Figura):

    def __init__(self, ime, a, b):
        super().__init__(ime)
        self.a = a
        self.b = b

    def povrsina(self):
        return self.a * self.b

    def __str__(self):
        return f"Ja sam pravougaonik ime = {self.ime}, P = {self.povrsina()}"

```

```

import figura as fig

p = fig.Pravougaonik("P1", 1, 2)
print(p)

```

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

```
    def povrsina(self):
        pass
```

```
class Pravougaonik(Figura):
```

```
    def __init__(self, ime, a, b):
        super().__init__(ime)
        self.a = a
        self.b = b
```

```
    def povrsina(self):
        return self.a * self.b
```

```
    def __str__(self):
        return f"Ja sam pravougaonik ime = {self.ime}, P = {self.povrsina()}"
```

Ja sam pravougaonik ime = Pravougaonik 1, P = None

```
import figura as fig
```

```
p = fig.Pravougaonik("P1", 1, 2)
print(p)
```

```

class Figura:
    def __init__(self, ime):
        self.ime = ime

    def površina(self):
        # komentar umesto pass

class Pravougaonik(Figura):

    def __init__(self, ime, a, b):
        super().__init__(ime)
        self.a = a
        self.b = b

    def površina(self):
        return self.a * self.b

    def __str__(self):
        return f"Ja sam pravougaonik ime = {self.ime}, P = {self.površina()}"

```

```

import figura as fig

p = fig.Pravougaonik("P1", 1, 2)
print(p)

```



```

class Figura:
    def __init__(self, ime):
        self.ime = ime

    def površina(self):
        # komentar umesto pass

class Pravougaonik(Figura):

    def __init__(self, ime, a, b):
        super().__init__(ime)
        self.a = a
        self.b = b

    def površina(self):
        return self.a * self.b

    def __str__(self):
        return f"Ja sam pravougaonik ime = {self.ime}, P = {self.površina()}"

```

```

import figura as fig

p = fig.Pravougaonik("P1", 1, 2)
print(p)

```



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

    def povrsina(self):
        pass

class Pravougaonik(Figura):

    def __init__(self, ime, a, b):
        super().__init__(ime)
        self.a = a
        self.b = b

    def povrsina(self):
        return self.a * self.b

    def __str__(self):
        return f"Ja sam pravougaonik ime = {self.ime}, P = {self.povrsina()}"

import figura as fig

p = fig.Pravougaonik("P1", 1, 2)
f = fig.Figura("F1")
```

```
from abc import ABC, abstractmethod
```

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

```
    @abstractmethod  
    def povrsina(self):  
        pass
```

```
...
```

```
class Pravougaonik(Figura):  
    ...
```

```
import figura as fig
```

```
p = fig.Pravougaonik("P1", 1, 2)
```

```
f = fig.Figura("F1")
```

```
TypeError: Can't instantiate abstract class Figura  
with abstract method povrsina
```