Konstruktori

Anita Līva
Marina Juzova

Projekts Nr. 8.3.1.1/16/I/002 Kompetenču pieeja mācību saturā

Klase veido sagatavi objektu veidošanai, programmā izmanto konkrētus objektus, konstruktors – inicializēs (dos sākuma vērtības) objektam

Python valodā par konstruktoru sauc metodi: __init__()

Projekts Nr. 8.3.1.1/16/I/002 Kompetenču pieeja mācību saturā

__init__ metode

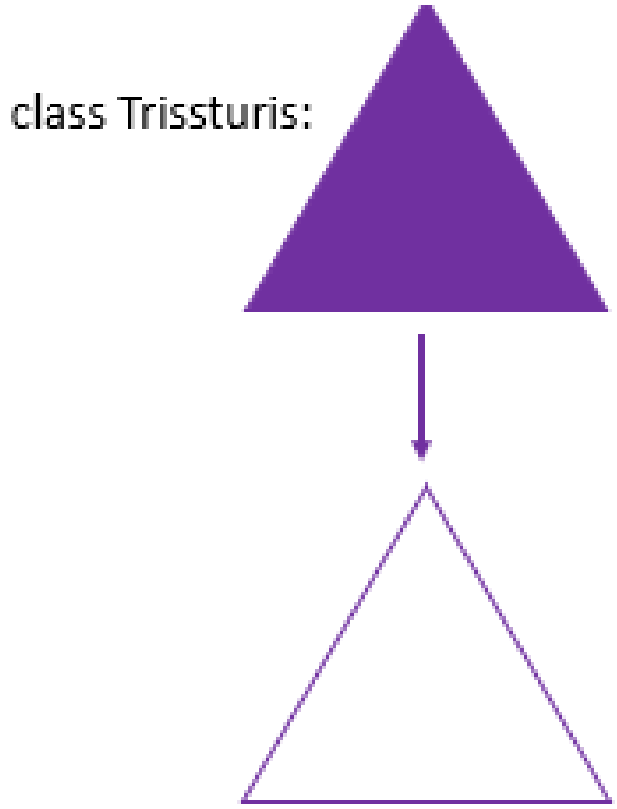
- __init__ methode ir līdzīga konstruktoriem C++ un Java, u.c. Konstruktorus izmanto, lai inicializētu (dotu sākuma vērtības) klases objektam.

- Python var būt tikai viens konstruktors, bet C++ un Java u.c. var būt daudzi.
- Taču, ja norāda, ka konstruktora vērtība ir nekāda, ‘none’ __init__(self, parametrs1 = none), un/vai lietojot cls atslēgvārdu var izveidot vairākus konstruktorus.

Konstruktora veidošanas kods:

```
def __init__(self):
        # konstruktora saturs
```

Šis ir noklusējuma konstruktors, tajā nav argumentu izņemot norādi (self) uz izveidoto klases objektu




```
# Ja neievada nekādu malas garumu, noklusējuma
# konstruktors veidojot objektu veidos vienādmalu
# trīsstūri ar malas garumu 6

def __init__(self):
    self.mala = 6
```

Atslēgvārds this/self

- Python valodā lieto atslēgvārdu self, lai piekļūtu instances (konkrēta objekta) mainīgajiem un klases atribūtiem. **Tas nav obligāti jāsauc self**, bet labā prakse.
- Python lieto atslēgvārdu cls, ko nelieto daudzās citās valodās, cls var tikai piekļūt klases locekļiem, klases metodēm.

- Atslēgvārds **self** Python valodā aptuveni līdzvērtīgs atslēgvārdam **this** C++ valodā.
- **This** ir norāde uz pašreizējo objektu, **self** ir norāde uz pašreizējo klasi.

Noklusējuma konstruktors Python

main.py

```
1  class Trissturis:
2
3      #noklusējuma konstruktors, trīsstura mala ir 6
4      def __init__(self):
5          self.mala = 6
6
7      #metode (jeb funkcija) izdrukāt perimetru
8      def drukat_Perimetru(self):
9          print( "Perimetrs: ", self.mala+self.mala+self.mala )
10
11 # veido klases objektu, ar noklusējuma datiem
12 obj1 = Trissturis()
13
14 # izsauc perimetra drukāšanas metodi objektam obj1
15 obj1.drukat_Perimetru()
```

Izveido noklusējuma konstruktoru

Veidojot objektu, konstruktors tiek automātiski izsaukts, piešķir 6 kā noklusējuma vērtību malai

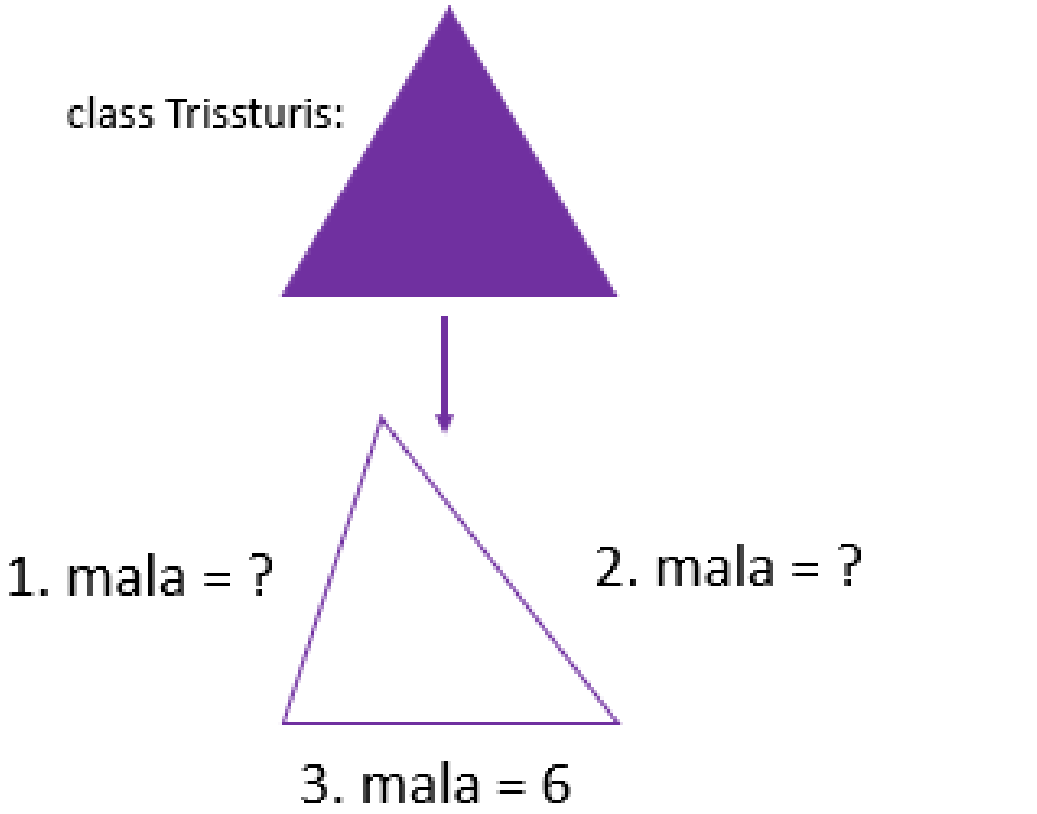
```
Perimetrs:  18


...Program finished with exit code 0
Press ENTER to exit console.
```

Konstruktors ar parametriem:

def __init__(self, parametrs1, …):
        # konstruktora saturs

Šis ir konstruktors, kurā var padot parametrus, vienu vai daudzus. Neskaitot norādi uz klases objektu (self), jo to vajag vienmēr.




```
#konstruktors ar parametriem
#lietotājs varēs ievadīt pats cik gara pirmā un otrā mala

def __init__(self, p_mala, o_mala):
        self.pirma_mala = p_mala
        self.otra_mala = o_mala
        self.tresa_mala = 6
```

```
class Trissturis:
-----------------------
pirma_mala
otra_mala
tresa_mala
perimetrs
-----------------------
drukat_Par_Trissturi()
perimetrs()
__init__(self, p_mala,
o_mala, t_mala)
```

Klase (vispārējs objekta apraksts)

```
obj1.Trissturis(5,4,6)
obj1.drukat_Par_Trissturi()
obj1.perimetrs()
-----------------------------------
pirma_mala = 5
otra_mala = 4
tresa_mala = 6
Perimetrs = 15
```

Objekts (konkrēts objekts, konkrēti lielumi, var pielietot metodes)

Konstruktors ar parametriem Python

main.py

```
1  class Trissturis:
2      pirma_mala = 0
3      otra_mala = 0
4      tresa_mala = 0
5      perimetrs = 0
6
7      #konstruktors ar parametriem,
8      #uzskatei atstāts tā, ka pirmo un otro malu veido kā parametrus,
9      #trešo atstāj pēc noklusējuma vienmēr 6
10     def __init__(self, p_mala, o_mala):
11         self.pirma_mala = p_mala
12         self.otra_mala = o_mala
13         self.tresa_mala = 6
14
15     #metode (jeb funkcija) izdrukāt malas un perimetru
16     def drukat_Par_Trissturi(self):
17         #str() pārvērš skaitli par teksta rindu
18         print("Pirmā mala  = " + str(self.pirma_mala))
19         print("Otrā mala   = " + str(self.otra_mala))
20         print("Trešā mala  = " + str(self.tresa_mala))
21         print("Perimetrs   = " + str(self.perimetrs))
22
23     def perimetrs(self):
24         self.perimetrs = self.pirma_mala + self.otra_mala + self.tresa_mala
25
26 # veido klases objektu ar argumentiem,
27 # veidojot izsauks konstruktoru ar parametriem
28 obj1 = Trissturis(5, 4)
29 # izrēķina perimetru objektam obj1
30 obj1.perimetrs()
31 # izsauc perimetra drukāšanas metodi objektam obj1
32 obj1.drukat_Par_Trissturi()
```

PAŠA VEIDOTIE PARAMETRI

ARGUMENTI

```
Pirmā mala  = 5
Otrā mala   = 4
Trešā mala  = 6
Perimetrs   = 15


...Program finished with exit code 0
Press ENTER to exit console.
```

Argumenti pret parametriem

Metodes (jeb funkcijas) parametrus piemin funkcijas definēšanas laikā.

Metodes (jeb funkcijas) argumenti ir īstās vērtības, ko padod funkcijai, kad šo metodi (jeb funkciju) izsauc.

Jautājumi refleksijai

1. Vai noklusējuma konstruktoram ir parametri?
2. Kāda ir atšķirība starp this un self?
3. Kurā valodā vieglāk veidot vairākus konstruktorus Python vai C++?

4. Kādu vienu jaunu lietu iemācījies, ko līdz šim nezināji?
5. Kā to varēs pielietot reāli dzīvē?

Click to add text

Paldies!

www.skola2030.lv
facebook.com/Skola2030