Uvod v Python

Delo z datotekami, razredi in objekti

Delo z datotekami v Pythonu

- Python nam omogoča tudi delo z datotekami. Najbolj osnovne sta branje (read -> argument ”r”) in pisanje (write -> argument ”w”) v datoteko

Primer 1.1:

Branje datoteke

Datoteki:

input1.txt:

"To je moj input text."

input2.txt:

"Prva vrstica
Druga vrstica
Tretja vrstica"

```
#Odpremo datoteko "input.txt", z argumentom "r" povemo, da bomo
#datoteko brali (datoteka se mora nahajati v isti mapi kot
#program!!!). Če želimo pisati v datoteko uporabimo argument "w".
f1 = open("input1.txt", "r")
for line in f1.readlines():   # Branje vrstic
    print(line)

f1.close() # Datoteko zapremo (da sprostimo sistem)
f2 = open("input2.txt", "r")
print(f2.readline()) #Izpis le prve vrstice
print(f2.readline()) #Izpis še druge vrstice
print(f2.readline(8)) #Izpis do 8-ga "characterja" tretje vrstice
f2.close() #Ne pozabimo zapriti datoteke
```

Primer 1.2:

Pisanje/dodajanje teksta v datoteko

```
grske_crke = ['alfa', "beta", 'gama']

f = open("output.txt", "w") #Datoteko odpremo za pisanje. "write"
#izbriše že obstoječo datoteko in ustvari novo
#f = open("output.txt", "a") #Enako kot "write", le da v tem
#primeru če datoteka že obstaja je ne izbriše in se tekst dodaja

for i in grske_crke:
    f.write(i) #Dodamo cel "grske_crke" v datoteko
f.close() #Zapremo datoteko
```

Razredi in objekti v Pythonu

- Objekt je kombinacija spremenljivke in funkcije, ki jih dobi preko razreda
- Objekt si lahko predstavljamo kot podatkovna struktura, ki vsebuje podatke in prav tako funkcije
- Razred je osnovni temelj za definiranje objekta

Primer 2.1:

Definiranje razreda

```
class moj_razred:
    spr = 123
    def fnk(self): #"self" bomo razložili v naslednjih vajah
        print("Pozdrav iz funkcije 'fnk'.")
moj_objekt = moj_razred()
#Spremenljivka "moj_objekt" vsebuje objekt razreda "moj_razred",
#ki vsebuje spremeljivko in "fnk" funkcijo
moj_objekt.fnk()
```

Primer 2.2:

Dostop do spremenljivk v razredu

```
class moj_razred:
    spr1 = 1
    spr2 = 2

    def fnk(self):      #"self" bomo razložili v naslednjih vajah

        print("Pozdrav iz funkcije fnk")

moj_objekt1 = moj_razred()
moj_objekt2 = moj_razred()

moj_objekt1.spr2 = 3      #Spremenimo spremenljivko, shranjeno v "spr2" objekta
                          #"moj_objekt1"

print(moj_objekt1.spr2) #Izpis spremenjenega "spr2" prek objekta "moj_objekt1"
print(moj_objekt2.spr2) #Izpis nespremenjenega "spr2" prek objekta "moj_objekt2"

moj_objekt1.fnk()         #Klicemo funkcijo "fnk()" objekta "moj_objekt1"
print(moj_objekt1.spr1) #Izpis spremeljivke "spr1" v objektu "moj_objekt1"
```

Primer 2.3:

Dostop do spremenljivk v razredu

```
# -*- coding: utf-8 -*-
class avto:
    barva = ""
    def opis(self): #"self" bomo razložili v naslednji vaji
        opis_niz = "To je %s avto." % self.barva
        return opis_niz

avto1 = avto() #Objekt
avto2 = avto() #Objekt

avto1.barva = "moder" #Nastavimo barvo "avta1"
avto2.barva = "rdeč" #Nastavimo barva "avta2"

print(avto1.opis())
print(avto2.opis())
```

“self” parameter

- Je prvi parameter v kateremkoli razredu
- Python uporabi “self” parameter za povezavo na ustvarjen objekt

Primer 2.4:

"Self" parameter

```
# -*- coding: utf-8 -*-
class Kalkulator:
    vrednost1 = 0
    vrednost2 = 55

    def prištej(self, znesek):
        self.vrednost1 += znesek   #Prištejemo
        self.vrednost2 += 444
        #znesek.vrednost2 += 1313 #Error. To lahko naredimo le s prvim parametrom

    def poglej_vrednost1(self): #Ta funkcija nam vrne "vrednost1"
        return self.vrednost1

    def poglej_vrednost2(abc): #"self" je le izraz za prvi parameter razreda. V
programu ga lahko                          #poljubno poimenujemo.
        return abc.vrednost2

obj = Kalkulator() #Objekt
obj.prištej(znesek = 123) #Klicanje funkcije. Čeprav ima funkcija definirana dva
                          #parametra, se "self" pri klicanju ne šteje kot
                          #vhodni parameter
print(obj.poglej_vrednost1())
print(obj.poglej_vrednost2())
```

“__init__” funkcija

- Je konstruktor.
- Se izvede takoj, ko se ustvari objekt (če razred objekta vsebuje konstruktor)
- Vedno vzame vsaj en argument -> “self”
- Navezuje se na vse objekte iz tega razreda

“__del__” funkcija

- Je dekonstruktor
- Se izvede, ko se izbriše objekt razreda
- Navezuje se na vse objekte iz tega razreda

Primer 2.5:

Konstruktor in destruktor

```
class Avto: #Definiramo razred
    vrednost = 111
    def __init__(self, barva): #__init__ funkcija, konstruktor
        self.barva = barva
        print("Start konstruktorja.")
    def __del__(self):          #__del__ funkcija, dekonstruktor
        print("Start dekonstruktorja in izbris objekta.")

avto = Avto("modra")
print(avto.barva)
del avto #Brisanje objekta

#print(avto.barva) #Error, ker objekt "avto" ne obstaja več, je bil izbrisan
```

Primer 2.6:
Konstruktor bolj podrobno

```
class Avto:
    vrednost = 111
    def __init__(self, barva): #__init__ funkcija, konstruktor
        self.barva = barva

    def random1(self, vrednost):
        self.vrednost *= 2 #Funkcija tu pobira spremenljivko "vrednost" iz
                           #razreda, ne kot parameter funkcije!!
        return self.vrednost

    def random2(self):
        self.vrednost *= 3
        return self.vrednost

    def random3(self, vr):
        vr += self.vrednost
        return vr

avto = Avto("modra")    # Opomba: ne podajaj direktno "self" parametra, le "barva" parameter
print(avto.barva)

obj = Avto("zelena")
print(obj.barva, obj.random1(765765)) #Tu dodajanje parametra v funkciji nima efekta
print(obj.barva, obj.random2()) #"vrednost" se je spremenila, ker smo že prej pognali
                                #funkcijo random1

print(obj.barva, obj.random3(765765)) #Tu dodajanje parametra v funkciji ima efekt

#print(avto.vr) #Error, ker spremenljivka "vr" ni definirana v samem razredu (poleg
                #"vrednost")
```

Dedovanje v razredih

- Namesto definiranja razreda od začetka lahko v razred "uvozimo" lastnosti iz že nekega obstoječega razreda
- IZPELJANI RAZRED:
  - Izpeljani razred (child / subclass) deduje lastnosti iz željenega (parent) razreda
  - Dedovane lastnosti lahko uprablja kot bi bile definirane v njem
  - Lahko tudi prepiše (overwrite) spremenljivke in metode iz svojega (parent) razreda

Primer 2.7:

Primer dedovanja iz razreda

```
#A --> Razred (parent)
#B --> Izpeljani razred (child)

class A:        # Definiramo razred
   A_vrednost = 100
   def __init__(self):
      print ("Klicanje (parent) razreda A.")
   def A_metoda(self):
      print ('Klicanje metode (parent) razreda A.')
   def določi_lastnost(self, lastnost):
      A.a_vrednost = lastnost
   def dobi_lastnost(self):
      print ("Lastnost (parent) razreda A :", A.A_vrednost)

class B(A): #Izpeljani razred, vzamemo lastnosti iz (parent) razreda
   def __init__(self):
      print ("Klicanje konstruktorja izpeljanega razreda B.")

   def B_metoda(self):
      print ('Klicanje metode izpeljanega razreda razreda B.')

c = B()                     #Objekt (child) izpeljanega razreda B
c.B_metoda()                #Izpeljani razred kliče svojo metodo
c.A_metoda()                #Izpeljani razred kliče metodo svojega (parent) razreda
c.določi_lastnost(200)      #Izpeljani razred spet kliče metodo svojega (parent) razreda
c.dobi_lastnost()           #Izpeljani razred spet kliče metodo svojega (parent) razreda
```