OSNOVNI rad u Pythonu - kroz primjere

Postoje dvije "inačice" Pythona: Python2 i Python3. Iako su vrlo slične ima i razlika. Python3 ima punu podršku za UTF-8 način kodiranja teksta (pored ostalih stvari) i njegovo korištenje se preporuča (ispred Pythona 2).

Preporučena razvojna okolina je Linux s Pythonom 3. Međutim, može poslužiti i bilo koji operacijski sustav s Pythonom.
Python službene stranice: https://www.python.org/
Anaconda: (python sa skoro svim paketima): https://www.continuum.io/downloads
Na Linuxu se najčešće instalira kao paket (ne skidati s gornjih linkova).

U datotekama (s primjerima) se kao zaglavlje (prve dvije linije) pojavljuju dvije linije:

```
#!/usr/bin/python
# -*- coding: UTF-8 -*-
```

Iako sve što slijedi iza `#` za Python je komentar, on će ipak pogledati te linije te koristiti UTF-8 način kodiranja. Prva linija je za opis kojim programom to pokrenuti (ako to ne radimo ručno iz komandne linije).

Varijable

Varijable u Pyhonu mogu biti različitih tipova.
Osnovni tipovi su:
- brojevi (cijeli, decimalni)
- niz znakova - string
- složeniji tipovi podataka: liste, riječnici, skupovi, ...

Primjeri kroz interaktivni rad s Pythonom (Python je pokrenut u naredbenoj liniji: prefiks (prompt) koji on stavlja je `>>>`. Ako se želi to reproducirati kopirati bez prefiksa.)

```
# brojevi
>>> a = 5
>>> b = 5.5
>>> c = -5.5e10
>>> d = -5.5e-10
>>> e = a + b + c + d
>>> e
-54999999989.5

# stringovi
>>> f = 'Jedan'
>>> len(f)   # duljina stringa => broj znakova (oprez s našim znakovima u
2.)
5
>>> g = 'Dva'
>>> h = f+g
>>> h
```

```
'JedanDva'
>>> h[0]
'J'
>>> h[0:5]
'Jedan'
>>> h[5:]
'Dva'
>>> h[5]='X'
Traceback (most recent call last):
  File '<stdin>', line 1, in <module>
TypeError: 'str' object does not support item assignment
# stringovi se ne mogu ovako mijenjati, ali mogu napraviti novi:
>>> h = h[0:4] + 'X' + h[5:]
>>> h
'JedaXDva'

# liste
>>> i = []                                    # prazna lista
>>> j = [ 0, 1, 2, 3, 4, 5, 6, 7, 8, 9 ]   # lista od 10 elemenata
>>> j[0:5]
[0, 1, 2, 3, 4]
>>> j[5:10]
[5, 6, 7, 8, 9]
>>> j[5:]
[5, 6, 7, 8, 9]
>>> j[2:3] + j[6:8]
[2, 6, 7]
>>> j[5] = 100 # elemente liste moguće je mijenjati
>>> j
[0, 1, 2, 3, 4, 100, 6, 7, 8, 9]

# elementi liste mogu biti razni objekti
>>> k = [ 0, '1', 2, [ 'tri', 4 ], 'pet' ]
>>> k[3][0]
'tri'
>>> k[3][1]
4
```

Ispis na zaslon

Kad se Python koristi interaktivno (Python pa [enter]), onda se sadržaji varijabli mogu vidjeti tako da ih se navede u redak i stisne [enter]. To je već korišteno u prethodnom opisu (prefix `>>>`).

Naredba/funkcija: `print ( varijabla )` pripremiti će varijablu za ispis (pretvoriti ju u niz znakova) te ispisati.

Primjeri:

```
>>> print (1)
1
>>> print ('Jedan')
Jedan
>>> print (f)
Jedan
>>> print(f,g)
('Jedan', 'Dva')   # vrijedi za Python2; za Python3 ispis je: Jedan Dva
>>> print(f+g)
```

```
JedanDva
>>> print(j)        # j od prije
[0, 1, 2, 3, 4, 100, 6, 7, 8, 9]
>>> print (a)
5
>>> print ( "a =", a )
a = 5
>>> print ( "a =", a, "!" )
a = 5 !
>>> print ( "a = " + str(a) )    # operator "str" pretvara argument u
string
a = 5
# u nastavku se koristi zadnji način (+str()) jer je ispis isti u Python2/3
```

Uobičajeno će `print` uključivati i prelazak u novi red. Ako to nije željeno ponašanje, u Pythonu 3 se to može spriječiti dodavanjem argumenta `end=""` na kraj. Npr. u programu može biti:

```
print ( "1", end="")
print ( "2", end="")
print ( "3", end="")
print ( "4" )
```

što će rezultirati ispisom:

```
1234
```

i tek tada prelazak u novi red.

Unos s tipkovnice

Unos vrijednosti s tipkovnice i pridjela te vrijednosti varijabli postiže se funkcijom `input`. Povratna vrijednost je string - niz znakova učitanih s tipkovnice do pritiska znaka ENTER.

```
>>> a = input("Unesi nešto: ")
Unesi nešto: 55 je broj s dvije petice
>>> a
'55 je broj s dvije petice'
>>>
```

Ukoliko je potrebno znakove pretvoriti u broj koristiti funkciju `int` ili `float`.

```
>>> a = int(input("Unesi broj: "))
Unesi broj: 61
>>> a
61
```

Više brojeva u istom retku je najprije potrebno odvojiti a potom pretvoriti u brojeve.

```
>>> a = input("Unesi brojeve: ")
Unesi brojeve: 4 9 88 1
>>> a
'4 9 88 1'
>>> a = a.split(" ") # podijeli na podnizove omeđene razmakom
>>> a
['4', '9', '88', '1']
>>> a = [int(x) for x in a]
>>> a
[4, 9, 88, 1]
```

Kompaktno:

```
>>> a = [int(x) for x in input("Unesi brojeve: ").split(" ")]
Unesi brojeve: 4 9 88 1
>>> a
[4, 9, 88, 1]
```

Grananja (`if/else/elif`)

Kod "složenijeg" koda treba paziti na strukturu koda, tj. na uvlačenja jer kod Pythona to označava oznaku pripadnosti ili petlji ili if-u i slično. Uobičajena praksa je da se za poravnanje koriste dva znaka razmaka ili jedan tab. U nastavku se koristi jedan tab.

Primjeri (if.py):

```
# inicijalizacija varijabli (od prije)
a = 5
b = 5.5
c = -5.5e10
d = -5.5e-10

if a > b:
    print ( "Od " + str(a) + " i " + str(b) + " veći je " + str(a) )
else:
    print ( "Od " + str(a) + " i " + str(b) + " veći je " + str(b) )
print('------------------------------------------------------------')

if a > b:
    print ( "Od " + str(a) + " i " + str(b) + " veći je " + str(a) )
elif b > a:
    print ( "Od " + str(a) + " i " + str(b) + " veći je " + str(b) )
else:
    print ( "Varijable a i b su jednake i iznose " + str(a) )
print('------------------------------------------------------------')

najveci = a # Python3 dozvoljava i naše grafeme u imenima varijabli!
            # pa bi u njemu mogli pisati i najveći
            # ali radi prenosivosti se to ne preporuča
if b > najveci:
    najveci = b
if c > najveci:
    najveci = c
if d > najveci:
    najveci = d

print ( 'Najveći broj je: ' + str(najveci) )
print('------------------------------------------------------------')

# logički operatori: or, and, not, not in
if a > b and a > c:
    # a je najveći napravi nešto
    pass # ne radi ništa - to se može staviti i privremeno
if ( a > b and a > c ) or ( a == 42 ):
    # koristi a
    pass
print('------------------------------------------------------------')

prazno = [] # prazna lista
if prazno:
    print ( "prazno nije prazno!")
if not prazno:
    print ( "prazno je prazno!")
print('------------------------------------------------------------')
```

```
lista = [ 'a', 'b', 'c', 1, 2, 3 ]
if 'a' in lista:
    print ( "imam 'a'")
if 'x' not in lista:
    print ( "nemam 'x'")
a = 2
if a in lista:
    print ( "imam " + str(a) )
```

Petlje (`for`, `while`)

Primjeri (for.py):

```
### for
for i in range(5):
    print(i)
# 0 1 2 3 4 (svaki u svom redu)
print('----------------------------------------------------------------')

for i in range(1,5):
    print(i)
# 1 2 3 4 (svaki u svom redu)
print('----------------------------------------------------------------')

lista = [ 0, 1, 2, 3, 4, 5 ]
for i in lista:
    print(i)
# 0 1 2 3 4 5 (svaki u svom redu)
print('----------------------------------------------------------------')

# isto, ali na drugačiji način
for i in range(len(lista)):  # len - broj elemenata u listi
    print(lista[i])
print('----------------------------------------------------------------')

for i in lista[2:5]:
    print(i)
# 2 3 4 (svaki u svom redu)
print('----------------------------------------------------------------')

for i in lista[0:2] + ['a', 'b', 'c']:
    print(i)
# 0 1 2 a b c (svaki u svom redu)
print('----------------------------------------------------------------')

### while
a = 1
b = 10
while a < b:
    a = a + a
    print (a)
# 2 4 8 16
print('----------------------------------------------------------------')

a = 1
b = 10
while a < b:
    a = a + a
    if a < b:
        print (a)
```

```
# 2 4 8
print('------------------------------------------------------------------')

for i in range(5):
    for j in range(5):
        if i < j:
            continue # nastavlja s prvom bližom petljom
        if i == j:
            print ( i )
            break # prekida prvu bližu petlju
        print ( j, end="" )
```

Argumenti komandne linije

Ako se program pokrene sa: `python program.py Pero 1234 53-43` kako u programu doći do argumenata iz komandne linije? Preko `sys.argv` !

Primjeri (arg.py):

```
import sys  # učitaj modul sys

for arg in sys.argv:
    print (arg)

print('------------------------------------------------------------------')

# uz pretpostavku pokretanja: python3 arg.py Pero 1234 53-43
if len(sys.argv) > 2:
    a = sys.argv[2]
    b = int(a) # b je sada broj
    print ( "a:" + str(a) + " - tip: " + str(type(a)) )
    print ( "b:" + str(b) + " - tip: " + str(type(b)) )

print('------------------------------------------------------------------')

# učitavanje samo dijela modula
from sys import argv  # iz modula sys učitaj samo argv

for arg in argv:
    print (arg)
```

Lista `sys.argv` sadrži argumente, počevši s imenom datoteke s izvornim kodom `sys.argv[0]` (arg.py), preko prvog argumenta `sys.argv[1]` (Pero) itd. Argumenti su stringovi. Ako je potrebno koristiti funkcije za pretvorbu u druge tipove podataka (npr. `int()`) kao u primjeru.

Funkcije (`def`)

Primjeri (funkcije.py):

```
def funkcija(a,b,c):
    print ("Argumenti: " + str(a) + ", " + str(b) + ", " + str(c))

# poziv
funkcija ( 1, 2, 'tri' )

print('------------------------------------------------------------------')
```

```
# vraćanje vrijednosti
def povecaj (x):
    x = x + 1
    return x

a = 5
print ( a ) # 5
b = povecaj(a)
print ( b ) # 6

print('-----------------------------------------------------------------')

# promjena globalne varijable: krivo
a = 5
b = 0
def povecaj ():
    b = a + 1   # stvoriti će se nova "lokalna varijabla" b

print ( a ) # 5
povecaj()
print ( b ) # 0 !!!

print('-----------------------------------------------------------------')

# promjena globalne varijable: ispravno
a = 5
b = 0
def povecaj ():
    global b
    b = a + 1

print ( a ) # 5
povecaj()
print ( b ) # 6

print('-----------------------------------------------------------------')

# složenije globalne varijable koje se samo dijelom mijenjaju su OK
a = [ 0, 1, 2, 3, 4 ]
def povecaj ():
    for i in range(len(a)):
        a[i] += 1

print ( a )  # [0, 1, 2, 3, 4]
povecaj()
print ( a )  # [1, 2, 3, 4, 5]
```

Zadaci za provjeru znanja (bez rješenja!)

1. Ispisati višekratnike broja 2 manjih od 1000 (računajući ih!).

   Npr. `python3 vis2.py`
   ```
   2 4 8 16 32 64 128 256 512
   ```
   (ispis može biti u retcima, svaki broj u svom retku)

2. Ispisati najmanji i najveći od upisanih brojeva.

   Npr. `python3 brojevi.py 3 7 2 4 -8 1`
   ```
   najmanji: -8
   najveci: 7
   ```

3. Iz niza upisanih brojeva, parne staviti u listu `a`, neparne u `b` te ih ispisati (`print(a)`, `print(b)`).

   Napomene:
   - dijeljenje: realno: `7 / 3 = 2,5` cjelobrojno: `7 // 3 = 2`
   - modul: `7 % 3 = 1, 4 % 2 = 0`
   - binarni &: `7 & 1 = 1, 4 & 1 = 0`

   Npr. `python3 parni.py 3 7 2 4 -8 1`
   ```
   neparni:
   [3, 7, 1]
   parni:
   [2, 4, -8]
   ```