Uvod u Python[1]

Programski jezik *Python* je veoma popularan jezik opšte namjene. *Python* pripada skript paradigmi, ali ima i imperativne, objektno-orijentisane i funkcionalne karakteristike. Skriptovi pisani u *Python*-u imaju ekstenziju *.py*. U pitanju je interpretirani programski jezik čiji interaktivni interpreter možemo dobiti komandom *python3*[2] u terminalu. Dinamički je tipiziran pa se tipovi određuju tek u fazi izvršavanja.
Prednosti:

- jednostavna sintaksa
- čitljivost
- veliki broj biblioteka
- besplatan
- …

Mane:

- veliko zauzeće memorije
- loša efikasnost
- …

Pokretanje programa:

1. način: *python3* naziv.py
2. način: *./naziv.py*[3]

help - za dobijanje dokumentacije u interaktivnom interpreteru
print - funkcija za ispis
dir - funkcija koja izlistava sadržaj tekućeg prozora ili željenog modula
len - funkcija koja vraća dužinu kolekcije
range - funkcija koja vraća niz elemenata u zadatom intervalu
sorted - sortira elemente neke kolekcije
logičke konstante: *True* i *False*
modul *math* - za rad sa matematičkim funkcijama i konstantama

Komentari

```
# jednolinijski komentari u Python-u

"""
    Ovo bi bio
    višelinijski
    komentar
"""
```

[1] Korisna literatura: https://petlja.org/biblioteka/r/lekcije/TxtProgInPythonSrLat/02_console-toctree.
[2] *python3* je poziv za interpreter na većini operativnih sistema, na *Windows* operativnim sistemima pozivi su *python* ili *py*.
[3] U prvoj liniji skripta pišemo kao komentar #! putanju do *Python* interpretera i dodajemo dozvolu za izvršavanje skripta.

Rad sa brojevima

```
# Brojevni tipovi u Python-u: celobrojni (int), realni (float) i kompleksni (complex) brojevi.
# Dokumentacija u interpreteru se može dobiti sa: help(x) ili help(int).
# U nastavku su navedene neke osnovne brojevne operacije. Za više informacija pogledati dokumentaciju za odgovarajući tip podatka.
x = 10
y = 3

# sabiranje
print(x + y)

# oduzimanje
print(x - y)

# množenje
print(x * y)

# stepenovanje x^y
print(x ** y)

# cio dio količnika
print(x // y)

# realni količnik
print(x / y)

# ostatak pri dijeljenju
print(x % y)

# zaokruživanje na 2 decimale, formatirani ispis
y = 10.5678
print(f'Broj y zaokružen na 2 decimale je {y:.2f}')

# kastovanje - pozivanje odgovarajućeg konstruktora
print(float(x))

# jedan način definisanja kompleksnog broja
z = 5 + 10j

# moguće je izvršiti više dodjela u jednoj naredbi
x, y, z = 1, 2, 3

# samim tim moguće je swap izvršiti u jednoj naredbi
x, y = y, x
```

Kontrola toka

```
# Blokovi u Python-u su određeni identacijom odnosno vodećim belinama - sve
# susedne naredbe koje imaju isti broj vodećih belina nalaze se u istom bloku.

# grananje
x = 10
if x < 0:
    print('x je negativan broj')
elif x == 0:
    print('x je jednak nuli')
else:
    print('x je pozitivan broj')

# petlje
i = 0
while i < 10:
    print(i)
    i += 1

# for je kolekcijska petlja u Python-u
for i in range(10):
    print(i)
```

Rad sa niskama

```
# Niske u Python-u su imutabilni objekti tj. konstantne su i ne mogu se menjati.
# Mogu se zapisivati sa jednostrukim ili dvostrukim navodnicima.

print('Na ovaj \
nacin mozemo \
tekst pisati u vise redova \
a da se on tretira \
kao jednolinijski')

s = 'Ovo je neka niska'

# čitanje sa standardnog ulaza
t = input('Unesite nisku t: ')
```

```
# nadovezivanje
print(s + t)
# nadovezivanje 5 puta niske s
print(s * 5)

# višelinijski (doslovni) stringovi
# moguće je koristiti i jednostruke navodnike
t = """She's got a smile that it seems to me
Reminds me of childhood memories
Where everything was as fresh as the bright blue sky
Now and then when I see her face
She takes me away to that special place"""

# indeksiranje niske ‘Zdravo’
#  Z   d   r   a   v   o
#  0   1   2   3   4   5
# -6  -5  -4  -3  -2  -1
# pristupanje 0. elementu
print(s[0])

# slice operator omogućava da jednostavno dobijemo neku podnisku date niske
print(s[3:5])
# ako se izostavi leva granica kreće se od početka niske, a ako se izostavi desna
# ide se do kraja niske. Ako se izostave obe granice onda se čita kompletna niska s.

# operatori in i not in proveravaju da li je neka niska podniska date niske
if 'zdravo' in 'zdravo svima':
    print('jeste podniska')
else:
    print('nije podniska')

if 'zdravo' not in 'zdravo svima':
    print('nije podniska')
else:
    print('jeste podniska')
```

Torke i liste

```
# Torke i liste su uređeni konačni nizovi podataka. Podaci u okviru torke i liste mogu biti
# različitog tipa. Torke i liste se razlikuju po tome što su torke imutabilne tj. konstantne i
# ne mogu se menjati.
# Indeksiranje, sabiranje, množenje i slice operator imaju istu sintaksu i semantiku kao
# kod stringova.
# Operatori in (not in) proveravaju da li se neki element nalazi (ne nalazi) u kolekciji.

# definicija torke
```

```
t = (1, 2, 3, 'zdravo')

# definicija liste
l = [ 'neki string',  'dan',  'godina',  'zdravo']

# dodavanje elementa na kraj liste
l.append('niska')

# kopiranje liste l u listu l1
l1 = l[:]

# U dokumentaciji se može naći veliki broj korisnih funkcija. Ovdje navodimo korišćenje
# funkcije sort kako bismo prikazali sintaksu lambda (anonimne) funkcije.
# sortiranje elemenata liste stringova u zavisnosti od poslednjeg karaktera
l.sort(key = lambda x : x[-1])
print(l)
```

Skupovi

```
# Skup je struktura podataka bez ponavljanja elemenata.

s1 = set([1, 1, 2, 3])
s2 = set([2, 3, 4, 4])

# skupovne operacije

# presjek
print(s1 & s2)

# unija
print(s1 | s2)

# razlika
print(s1 - s2)

# simetrična razlika
print(s1 ^ s2)
```

Mape

```
# Mape su strukture podataka koje nam omogućavaju da čuvamo parove:
# ključ -> vrednost. Ključevi moraju biti jedinstveni.

studenti = {'Ana' : 10, 'Marko' : 7, 'Nikola' : 8}

# pristupanje vrednosti određenog ključa
vrednost_kljuca_ana = studenti['Ana']

# ključevi u mapi
print(studenti.keys())

# vrednosti u mapi
print(studenti.values())

# jedan način iteriranja kroz mapu
for (kljuc, vrednost) in studenti.items():
    print(kljuc, vrednost)

# operator in (not in) proverava da li se ključ nalazi (ne nalazi) u mapi
if 'Marko' in studenti:
    print('Marko jeste kljuc')

# sortiranje ključeva mape
sortirani_kljucevi = sorted(studenti.keys())
print(sortirani_kljucevi)
```

Zadaci

1. Pročitati sadržaj datoteke čiji se naziv navodi kao argument komandne linije.

```
#!/usr/bin/python3

# Numeracija argumenata komandne linije:
#    ./naziv.py arg1 arg2 ag3 …
#          0         1      2      3   …
# Uključivanje sys modula u kome se nalaze argumenti komandne linije. U nastavku
# svim funkcijama i promenljivim iz modula sys pristupamo sa sys.naziv_funkcije ili
# sys.naziv_promenljive.
# Moguće je korišćenje funkcija i promenljivih iz nekog modula bez prefiksa naziva tog
# modula: from naziv_modula import *
import sys

if len(sys.argv) != 2:
   sys.exit('Pogresan broj argumenata komandne linije')
```

```
# otvaranje fajla može dovesti do izbacivanja izuzetka, zbog toga je neophodno
# koristiti try/except naredbu
try:
    f = open(sys.argv[1], 'r')
except:
    sys.exit('Greska pri otvaranju fajla')

# Za čitanje iz datoteke: f.read(), f.readline(), f.readlines()...
# Za pisanje u datoteku: f.write(...)
sadrzaj = f.read()

f.close()

print(sadrzaj)
```

2. Napisati funkcije za računanje zbira, razlike i ispis brojeva koji se unose sa standardnog ulaza.

```
#!/usr/bin/python3

def zbir(x, y):
    return x + y

def razlika(x, y):
    return x - y

def ispis(x):
    print(x)

x = int(input('Unesite 1. broj: '))
y = int(input('Unesite 2. broj: '))

print('Zbir je:')
ispis(zbir(x, y))

print('Razlika je:')
ispis(razlika(x, y))
```