Kratki podsjetnik za Python 3 V1.0

Ukoliko imate u budućnosti bilo kakva pitanja vezana za Python slobodno mi pošaljite mail na **sergej.jakovljev@gmail.com**.
Dio gradiva je je namjerno izostavljen radi jednostavnosti. Krajem ljeta će na **http://www.srolija.com** biti nova verzija teksta sa gradivom predavanja.

Učitavanje biblioteka

```
import {naziv biblioteke}

import math
print(math.pi) # potrebno naglasit od kuda je pi


from {naziv biblioteke} import {ime funkcije}

from math import pi
from turtle import fd, lt, rt, up, down
print(pi) # nije potrebno naglasit od kuda je pi
```

Osnovni tipovi podataka

- integer (cijeli brojevi)

```
int(5.7) # zaokružuje float na manji cijeli (5)
int("12") # pretvara string u integer 12
```

- float (decimalni brojevi)

```
float(5) # pretvara integer u float 5.0
int("12") # pretvara string u float 12.0
```

- string (niz znakova / tekst)

```
string(5) # "5"
string(12.7) # "12.7"
string(True) # "True"
```

- boolean (logička istinitost)

```
# za sve što nije ili prazno ili jednako 0 je True
bool(1.5) # True
string(0) # False
string("") # False
```

Grananje

```
if ( {uvjet} ):
        # izvršava se ukoliko je uvjet istinit
elif ( {drugi_uvjet} ):
        # izvršava se ukoliko je drugi istinit
elif ( {treci_uvjet} ):
        # izvršava se ukoliko je treći istinit
else:
        # ukoliko niti jedan nije bio istint

a = 5
if (a % 2 == 0):
        print ("Dijeliv je sa 2, paran je!")
else:
        print("Neparan je!")
```

opcionalno

Ponavljanje

- poznati broj ponavljanja

```
for i in range({pocetak}, {kraj}, {korak}):
        # ovo se svaki put izvodi u rasponu
        # [pocetak, kraj> sa povečanjima za korak

for i in range(3): # ekvivalentno range(0, 3, 1)
        print(i) # 0, 1, 2

for i in range(1, 9, 2)
        print(i) # 1, 3, 5, 7 (završni nije uljučen)
```

- dok je nešto / dok se nešto ne dogodi

```
while( {uvjet ponavljanja} ):
        # ovo se svaki put izvodi dok god je uvjet
        # ponavljanja istinit

broj = 0
while(broj < 10):
        print(broj)
        broj += 2 # isto kao broj = broj + 2
# ispisat će: 0, 2, 4, 8
```

- prekidanje petlje / preskakanje koraka

```
for i in range(10):
        if (i % 2 == 0):
        # prekače ostatak uvučenog koda za parne
                continue
        if (i == 7):
        # prekida prvu vanjsku petlju kad je i = 7
                break
        print(i)
# ispisat će: 1, 3, 5
```

Važnije biblioteke

```
math:
- sqrt(broj) - kvadratni korijen
- pi - konstanta broja PI (3.141592653589793)

random:
- randint(pocetak, kraj) - cijeli broj iz intervala [pocetak, kraj]
- random() - nasumičan broj iz intervala [0, 1>

turtle:
- up(), down() - dizanje i spuštanje kornjace na papir (pali/gasi boju)
- fd(koraci), bk(koraci) - pomicanje naprijed/nazad
- rt(stupnjevi), lt(stupnjevi) - rotiranje desno/lijevo
- goto(x, y) - odlazi na zadanu poziciju
- begin_fill(), end_fill() - kada crtamo oblik i želimo da bude ispunjen prije i
                             nakon crtanja oblika pozovemo kako bi ga ispunili
- color(boja) - kojom bojom crtamo (npr. "red", "blue" ...)
- pensize(debljina) - kojom debljinom crtamo linije
```

Važnije funkcije

```
min({prva}, {druga}, {treca} ...) # vraća namjanji od predanih
min(5, 1) # 1

max({prva}, {druga}, {treca} ...) # vraća najveći od predanih
max(3, 1, 7, -300) # 7

input()
input("Poruka kod unosa: ") # učitava od korisnika i vraća unos kao string
ime = input("Unesi ime: ")
broj = int(input("Unesi broj: "))

round({decimalni broj} # zaokružuje brojeve
round(3.14) # 3
round(5.7) # 6

print({vrijednost za ispisat})
```

Operatori prema prioritetu

```
(, ) zagrade
** potenciranje
*, /, %, // množenje, dijeljenje, cijelobrojni ostatak i cijelobrojno dijeljenje
+, - zbrajanje i oduzimanje
<=, <, >, >= usporedba veličina
==, != provjera jednakosti
=, %=, /=, //=, *=, **=, +=, -= operatori dodjele (a -= 1 je isto kao a = a - 1)
in, not in operatori pripadnosti (kod lista i rječnika)
not, or, and logički operatori

Prioritet govori kojim se redosljedom izvode operacije. Kada želimo operacije
izvesti drugačijim redostljedom od standardnog prioriteta trebamo pisati zagrade:

(2 + 3) * 4 => 20, bez zagrada bi se zbog prioriteta izvelo kao 2 + (3 * 4) => 14
```

Funkcije

```
- mogu ali ne moraju primat jednu ili više vrijednosti (argumenti) i mogu ali
  ne moraju vraćat neku vrijednost (return)

def ime_funkcije({argument}, {drugi} ...):
        # tijelo funkcije
        return # opcionalno povratna vrijednost

def kvadriraj(broj):
        return broj * broj

print(kvadriraj(56)) # 3136
```

Liste (poredane)

```
prazna_lista = []
lista = [1, "t"]

lista.append(3) # dodajemo 3 na kraj
print(lista) # [1, "t", 3]

lista.insert({gdje}, {što})
lista.insert(1, 4)
print(lista) # [1, 4, "t", 3]

# indeksi liste kreću od 0!
# ispisuje drugi član (sa indeksom 1)
print(lista[1]) # 4

lista[0] = 5 # postavlja prvi na 5
del lista[2] # briše treći član

sadrzi_pet = 5 in lista
print(sadrzi_pet) # True
```

Rječnici (ključ:vrijednost)

```
prazan_rjecnik = {}
rjecnik = { "ime": "Tanja", "zgodna": True }

rjecnik["dob"] = 20 # postavimo novi ključ
print(rjecnik)
# {'ime': 'Tanja', 'dob': 20, 'zgodna': True}

# promijena vrijednosti za postojeći ključ
rjecnik["dob"] = 21
print(rjecnik)
# {'ime': 'Tanja', 'dob': 21, 'zgodna': True}

# brisanje vrijednosti za dani ključ
del rjecnik["dob"]
# postaje {'ime': 'Tanja', 'zgodna': True}

# primjer provjere je li ne postoji
nema_ime = "ime" not in rjecnik
print(nema_ime) # False -> znaci ima ga!
```