FUNKCIONALNO PROGRAMIRANJE

Oznaka predmeta: FPR

Predavanje broj: 01

Nastavna jedinica: PYTHON,

Nastavne teme:

Uvod. Python Interactive Shell. Konvencije. Tipovi. Operatori. Konverzije. Kontrola toka. Naredbe. Petlje. Brojevi. Brisanje reference. Skupovi.

Predavač: prof. dr Perica S. Štrbac, dipl. ing.

Literatura:

Steven Lott: "Functional Python Programming", Packt Publishing, 2015.

Python

- Pošto je Python nastao kombinacijom više programskih jezika, neki elementi kao liste, adresari, ... , predstavljaju dobru sponu sa čistim funkcionalnim jezicima.
- U tom smislu Python će biti obrađen kako bi se kasnije prešlo na čisti funkcionalni programski jezik LISP.
- Python je kreirao Guido van Rossum krajem osamdesetih i početkom devedesetih godina 20. veka u National Research Institute for Mathematics and Computer Science in the Netherlands.
- Python je izveden iz mnogih jezika (ABC, Modula-3, C, C++, Algol-68, SmallTalk, Unix shell, ...
- Python je viši programski, interpreterski, interaktivni, objektno orijentisani jezik.
- Python je dizajniran da bude veoma čitljiv što povlači strog način pisanja njegovog koda (potrebno je malo vremena da se naviknete).
- Python se procesira u vreme izvršavanja od strane interpretera.
  - Ne morate da prevodite program da biste ga posle izvršili.
- Jezik Python je i interaktivan u smislu da u samom Python promptu možete direktno pisati programe.

Python

- Neke Python-ove karakteristike:
- Lak za učenje: relativno malo ključnih reči, jednostavna struktura i čista sintaksa.
- Lak za čitanje : Python-ov kod je predviđen da bude jasno definisan i veoma čitljiv.
- Lak za održavanje.
- Koristi standardne biblioteke tako da je kompatibilan za razne platforme: UNIX, Windows, Macintosh.
- Ima i interaktivni mod rada.
- Prenosivost: Python se može pokrenuti sa istim interfejsom na različitim hardverskim platformama.
- Proširivost različitim modulima niskog nivoa čime se može postići da alat koji se koristi bude efikasniji.
- Baze podataka: Python omogućuje interfejse ka svim značajnim komercijalnim bazama podataka.
- GUI programiranje.
- Python je i objektno orijentisan jezik tako da je omogućeno i lako održavanje velikih kodova.

Python

- Skalabilnost**:** Python obezbeđuje dobru strukturu i podršku za velike programe.
- Python podržava metode funkcionalnog, strukturnog i OOP principa programiranja.
- Može se koristiti kao skript jezik ili za izradu velikih aplikacija.
- Python podržava automatsku dealokaciju (garbage collection).
- Može biti integrisan sa drugim jezicima.
- Omogućuje interaktivan inkrementalni razvoj i testiranje.

- Korišćenje Python-a.
  - shell tools
  - system admin tools, command line programs
  - extension-language work
  - rapid prototyping and development
  - graphical user interfaces
  - database access
  - distributed programming
  - Internet scripting

Python

- Trenutno su prisutne sledeće verzije Python-a.
  - Python ili "CPython" (napisan u C/C++)
  - "Jython" (napisan u programskom jeziku Java za JVM)
  - "IronPython" (napisan u C# za .Net okruženje)
- Okruženja koja se koriste za rad u Python-u:
  - PyCharm
  - PyDev with Eclipse
  - Komodo
  - Emacs
  - Vim
  - TextMate
  - Gedit
  - Idle
  - PIDA (Linux)(VIM Based)
  - NotePad++ (Windows)
  - BlueFish (Linux)

Python

- Python Interactive Shell:

```
C% python
Python 3.5.0 (v3.5.0:374f501f4567, Sep 13 2015, 02:27:37)
[MSC v.1900 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more
information.
>>>
```

- Kada se dobije Python prompt (>>>) može se direktno pisati u Python-u.

```
>>> 2+3*4
14
>>> name = "Pera"
>>> name
'Pera'
>>> print ("Hello ", name)
Hello Pera
>>>
```

Python

- Python fajlovi imaju ekstenziju **.py**.
- Neka je sadržaj fajla `test.py` :

```
print ("Hello, Python!");
```

- Startovanje navedenog programa je kao što sledi (pretpostavka je da ste u PATH dodali navigaciju ka Python interpreteru):

```
$ python test.py
```

izlaz je:

```
Hello, Python!
```

- Ako se skript `test.py` modifikuje tako da je sadržaj:

```
#!/usr/bin/python
print ("Hello, Python!");
```

i ako je Python interpreter smešten u `/usr/bin` folderu. potrebno je učiniti test.py izvršnim fajlom kao što sledi:

```
$ chmod +x test.py
$ ./test.py
```

- Izlaz je: `Hello, Python!`

Python

- Python je case-sensitive, dakle, razlikuje velika i mala slova.
- Ključne reči su:

`and`	`exec`	`not`
`assert`	`finally`	`or`
`break`	`for`	`pass`
`class`	`from`	`print`
`continue`	`global`	`raise`
`def`	`if`	`return`
`del`	`import`	`try`
`elif`	`in`	`while`
`else`	`is`	`with`
`except`	`lambda`	`yield`

Python

- Konvencija za davanje imena je:
  - Python unutar identifikatora ne dozvoljava interpunkcijske karaktere kao što su @, $ i %.
  - Ime klase počinje velikim slovom. Svi ostali identifikatori počinju malim slovom.
  - Za indikaciju privatnog identifikatora navodi se jedna donja crta na početku identifikatora.
  - Za indikaciju sprečavanja redefinisanja navode se dve vodeće donje crte.
  - Ako se identifikator i završava sa dve donje crte onda se radi o jezikom definisanim specijalnim imenom.

```
>>> name = "John"
>>> name.__len__() # ili len(name)
4
>>> number = 10
>>> number.__add__(20) # ili number + 20
30
```

- Linije i uvlačenje
  - Python ne koristi velike zagrade za označavanje blokova koda.
  - Blokovi koda su označeni uvlačenjem linije(a).

Python

- Uvučene linije jednog bloka koda moraju biti jednako uvučene.

```
if True:
        print ("True")
else:
  print ("False")
```

- ili npr. pogrešno:

```
if True:
        print ("Answer")
        print ("True")
else:
        print ("Answer")
     print "False"
```

Python: primer koda

```
import sys
file_name="hm.py"
file_finish="kraj"; file_text="";
try:
  # open file stream
  file = open(file_name, "w")
except IOError:
  print ("There was an error writing to", file_name)
  sys.exit()
print ("Enter:", file_finish, end=' ')
print (" When finished.")
while True:
  file_text = input("Enter text: ")
  if file_text == file_finish:
     break
  file.write(file_text)
  file.write("\n")
file.close()
```

Python

```
file_name = input("Enter filename: ")
if len(file_name) == 0:
  print ("Next time please enter something")
  sys.exit()
try:
  file = open(file_name, "r")
except IOError:
  print ("There was an error reading file")
  sys.exit()
file_text = file.read()
file.close()
print (file_text)
```

- Pisanje linije koda u više redova (karakter za nastavak linije \):

```
total = item_one + \
        item_two + \
        item_three
```

- Pisanje linije koda u više redova za slučaj []. {} ili () se tumači normalno:

```
days = ['Monday', 'Tuesday', 'Wednesday',
        'Thursday', 'Friday']
```

Python

- Python prihvata apostrofe, navodnike i trostruke apostrofe i trostruke navodnike za početak i kraj string literala.
  - Trostruki se koriste za nastavak stringa u više redova.

```
word = 'word'
sentence = "This is a sentence."
paragraph = """This is a paragraph. It is
made up of multiple lines and sentences."""
```

- Komentari u Python-u počinju sa heš znakom (#) koji nije unutar string literala.

```
    #!/usr/bin/python

# First comment
print ("Hello, Python!");  # second comment
# This is a comment.
# This is a comment, too.
```

- U interaktivnoj interpreterskoj sesiji za terminaciju naredbe u više redova mora se ubaciti jedna prazna linija.

Python

- Čekanje na korisničku akciju:

```
#!/usr/bin/python

input("\n\nPress the enter key to exit.")
```

- Korišćenjem ; moguće je pisati više naredbi u istoj liniji:

```
>>> import sys; x = 'foo'; sys.stdout.write(x + '\n')
foo
4
```

- Višestruke naredbe se grupišu kao garnitura (suite). Složene naredbe kao if, while, def, class zahtevaju liniju zaglavlja i suite.

```
if expression :
     suite
elif expression :
            suite
else :
            suite
```

- Opis prekidača i argumenata iz komandne linije može se dobiti:

```
$ python -h
```

Python:parsiranje argumenata komandne linije

- Parsiranje argumenata i opcija komandne linije omogućuje modul `getopt`.

```
$ python test.py arg1 arg2 arg3
```

- Python-ov modul `sys` omogućuje pristup argumentima komandne linije korišćenjem `sys.argv`.
  - `sys.argv` predstavlja listu argumenata komandne linije.
  - `len(sys.argv)` predstavlja broj argumenata komandne linije.
- Član indeksiran nulom `sys.argv[0]` je program odnosno naziv skripta.

```
#!/usr/bin/python

import sys

print ('Number of arguments: ', len(sys.argv),' arguments.')
print ('Argument List: ', str(sys.argv))
```

Poziv:

```
$ python test.py arg1 arg2 arg3
```

Izlaz:

```
Number of arguments: 4 arguments.
Argument List: ['test.py', 'arg1', 'arg2', 'arg3']
```

Python:parsiranje argumenata komandne linije

- Metoda `getopt.getopt` se koristi kao što sledi:

```
getopt.getopt(args, options[, long_options])
```

  - args: lista argumenata koja će biti parsirana.
  - options: opcije kao prekidači koji se navode, ako opcija ima argument koristi se dvotačka kao delimiter (:).
  - long_options: opcionalni parametar koji je predstavljen listom stringova sa nazivima dugih (long) opcija.
    - Duge opcije, koje zahtevaju argument razdvojene su znakom jednakosti (=).
    - Ako se žele navesti samo duge opcije onda opcije moraju biti prazan string.
- Navedena metoda vraća vrednost koja je sastavljena od dva elementa:
  - lista parova (opcija, vrednost)
    - svaki par opcija-vrednost ima opciju kao prvi element ali sa prefiksom minus (npr. '-x') ili sa dva minusa za duge opcije (npr. '--long-option').
  - lista argumenata preostala nakon skidanja liste opcija.

Python:parsiranje argumenata komandne linije

- Izuzetak getopt.GetoptError se generiše kada se naiđe na nepoznatu opciju u listi argumenata ili kada nije naveden parametar za opciju koja zahteva argument.
- Argument izuzetka je string koji se odnosi na uzrok greške.
- Atributi **msg** i **opt** daju poruku o grešci i opciji.
- Npr. neka je format komandne linije kao što sledi:

```
test.py -i <inputfile> -o <outputfile>
```

i želi se da program reaguje kao što sledi:

```
$ test.py -h
usage: test.py -i <inputfile> -o <outputfile>

$ test.py -i BMP -o
usage: test.py -i <inputfile> -o <outputfile>

$ test.py -i inputfile
Input file is inputfile
Output file is
```

Python:parsiranje argumenata komandne linije

```
#!/usr/bin/python
import sys

print ('Number of arguments:', len(sys.argv), 'arguments.')
print ('Argument List:', str(sys.argv))


#!/usr/bin/python
import sys, getopt

def main(argv):
   inputfile = ''
   outputfile = ''
   try:
      opts, args = getopt.getopt(argv,"hi:o:",["ifile=","ofile="])
   except getopt.GetoptError as err:
      print ('usage: test.py -i <inputfile> -o <outputfile>')
      print (err.msg); print ("opcija je ",err.opt);
      sys.exit(2)
   for opt, arg in opts:
      if opt == '-h':
         print ('usage: test.py -i <inputfile> -o <outputfile>')
         sys.exit()
      elif opt in ("-i", "--ifile"):
         inputfile = arg
      elif opt in ("-o", "--ofile"):
         outputfile = arg
   print ('Input file is ', inputfile)
   print ('Output file is ', outputfile)

if __name__ == "__main__":
   main(sys.argv[1:])
```

Python: dodeljivanje

- Dodeljivanje promenljivih:

```
#!/usr/bin/python

counter = 100          # An integer assignment
miles   = 1000.0       # A floating point
name    = "John"       # A string
print (counter)
print (miles)
print (name)
```

Izlaz:

```
100
1000.0
John
```

- Višestruko dodeljivanje promenljivih:

```
a = b = c = 1

a, b, c = 1, 2, "john"
```

- Standardni tipovi podataka su: brojevi, stringovi, liste, n-torke i rečnici (numbers, string, list, tuple, dictionary).

Python: tipovi

- Brojevi u Python-u sadrže numeričke vrednosti: `var1 = 1`
- Python podržava sledeće tipove brojeva:
  - int (celi brojevi)
  - float (floating point realne vrednosti)
  - complex (kompleksni brojevi)
    - kompleksni brojevi se sastoje od para realnih brojeva koji se prikazuju u obliku x + yj (x je realni deo a y imaginarni deo).
      >>> z=5-3j
      >>>print (z) #(5-3j)

```
>>> import random
    >>> random.random()          # slučajni float x, 0.0<=x<1.0,
    0.37444887175646646
    >>> random.uniform(1, 10)  # slučajni float x, 1.0<=x<10.0,
    1.180014607311752 3
    >>> random.randrange(10)             # celi broj od 0 do 9, 7
    >>> random.randrange(0, 101, 2)      # paran broj od 0 do 100, 26
    >>> random.choice('abcdefghij')      # jedan slučajni element, 'c'
    >>> items = [1, 2, 3, 4, 5, 6, 7]
    >>> random.shuffle(items)
    >>> items                                  #[7, 3, 2, 5, 6, 4, 1]
    >>> random.sample([1, 2, 3, 4, 5],  3)   # tri uzorka, [4, 1, 5]
```

Python: tipovi

- Ispis celog broja u binarnom formatu:

```
>>> n = -37
>>> bin(n)
'-0b100101'
>>> n.bit_length() # bice 6
```

- Ispis celog broja po bajtovima:

```
>>> (1024).to_bytes(2, byteorder='big')
b'\x04\x00'
>>> (1024).to_bytes(10, byteorder='big')
b'\x00\x00\x00\x00\x00\x00\x00\x00\x04\x00'
>>> (-1024).to_bytes(10, byteorder='big', signed=True)
b'\xff\xff\xff\xff\xff\xff\xff\xff\xfc\x00'
>>> x = 1000
>>> x.to_bytes((x.bit_length() // 8) + 1, byteorder='little')
b'\xe8\x03'
```

- Stringovi i podstringovi:

```
str = 'Hello World!'
print (str)          # Hello World!
print (str[0])       # H
print (str[2:5])     # llo
print (str[2:])      # llo World!
print (str * 2)      # Hello World!Hello World!
print (str + "TEST") # Hello World!TEST
```

Python: tipovi

- Liste u Python-u predstavljaju najsvestraniji tip podataka.
- Članovi liste su odvojeni zarezom unutar uglatih zagrada ([]).
- Članovi unutar liste mogu biti različitog tipa.
- Vrednosti pohranjene u listi mogu se dohvatiti korišćenjem slajs operatora ([ ] i [:]) sa indeksiranjem počevši od 0.
- Znak plus (+) je operator konkatenacije lista a znak zvezdica (*) predstavlja operator ponavljanja.
- Liste su izmenljive po veličini i sadržaju.

```
#!/usr/bin/python
list = [ 'abcd', 786 , 2.23, 'john', 70.2 ]
tinylist = [123, 'john']
print (list)              # ['abcd', 786, 2.23, 'john', 70.2]
print (list[0])           # abcd
print (list[1:3])         # [786, 2.23]
print (list[2:])          # [2.23, 'john', 70.2]
print (tinylist * 2)      # [123, 'john', 123, 'john']
print (list + tinylist) # ['abcd', 786, 2.23, 'john', 70.2, 123,
'john']
```

Python: tipovi

- Python-ove n-torke (*Tuples*) se sastoje od elemenata koji su odvojeni zarezom i ograđeni malim zagradama.
- N-torke su read-only.

```
#!/usr/bin/python
tuple = ( 'abcd', 786 , 2.23, 'john', 70.2 )
list  = [ 'abcd', 786 , 2.23, 'john', 70.2 ]
tuple[2] = 1000 # Invalid syntax with tuple
list[2] = 1000 # Valid syntax with list
```

- Rečnici (*Dictionaries*) u Python-u su tip hash tabele, dakle, radi se o ascijativnim ključ-vrednost parovima.
  - Ključevi su obično stringovi ili brojevi,
  - Vrednosti su proizvoljni tipovi.

```
dict = {'Ime': 'Elvis', 'Rodjen': 1935, 'Oblast': 'R&R'};
```

- Rečnici su ograđeni veliki zagradama ({ }) a vrednosti se dodeljuju i uzimaju korišćenjem uglatih zagrada ([]).

```
>>> dict['Ime']
>>> 'Elvis'
```

- Rečnici nemaju koncept uređenosti među elemenatima (neuređeni su).

Python: konverzije tipova

Funkcija	Opis konverzije
`int(x [,base])`	Iz x u integer (baza se navodi ako je x string).
`float(x)`	Iz x u float.
`complex(real [,imag])`	Kreira kompleksan broj.
`str(x)`	Iz objekta x u string.
`repr(x)`	Iz objekta x u expression string.
`eval(str)`	Evaluira string i vraća objekat.
`tuple(s)`	Iz s u n-torku.
`list(s)`	Iz s u listu.
`set(s)`	Iz s u skup.
`dict(d)`	Kreiranje rečnika (d je sekvenca (key,value) parova).
`frozenset(s)`	Iz s u (frozen) skup.
`chr(x)`	Integer u karakter.
`ord(x)`	Karakter u njegovu integer vrednost.
`hex(x)`	Integer u heksadecimalni string.
`oct(x)`	Integer u oktalni string.

Python: operatori

- Python podržava sledeće operatore:
  - Aritmetički operatori (Arithmetic Operators)
  - Relacioni operatori (Comparison (Relational) Operators)
  - Operatori dodele (Assignment Operators)
  - Logički operatori (Logical Operators)
  - Operatori na novou bitova (Bitwise Operators)
  - Operatori pripadnosti (Membership Operators)
  - Operatori identiteta (Identity Operators)
- Aritmetički operatori:
  + sabiranje
  - oduzimanje
  * množenje
  / deljenje
  % ostatak pri deljenju
  ** eksponent (npr. 2**3=8)
  // celobrojno deljenje (npr. 9//2 = 4 and 9.0//2.0 = 4.0)

Python: operatori

- Operatori poređenja:

==	jednakost
!=	nejednakost
>	veće
<	manje
>=	veće ili jednako
<=	manje ili jednako

- Operatori dodele:

=	c = a + b		
+=	c += a	kao	c = c + a
-=	c -= a	kao	c = c - a
*=	c *= a	kao	c = c * a
/=	c /= a	kao	c = c / a
%=	c %= a	kao	c = c % a
**=	c **= a	kao	c = c ** a
//=	c //= a	kao	c = c // a

Python: operatori

- Bitwise operatori rade nad bitovima i izvršavaju bit po bit operacije.
- Neka je a = 60 i b = 13 Odgovarajući binarni prikaz i rezultata datih operacija je kao što sledi:

```
a   = 0011 1100
b   = 0000 1101
-----------------
a&b = 0000 1100
a|b = 0011 1101
a^b = 0011 0001
~a  = 1100 0011
```

- Operatori nad bitovima su kao što sledi:

&	bitski AND	(a & b)	12 (0000 1100)
|	bitski OR	(a | b)	61 (0011 1101)
^	bitski XOR	(a ^ b)	49 (0011 0001)
~	prvi komplement	(~a)	-61 (1100 0011)
<<	Left Shift	a <<= 2	240 (1111 0000)
>>	Right Shift	a >>= 2	15 (0000 1111)

Python: operatori

- Logički operatori:

and	logičko AND	true ako su oba true
or	logičko OR	true ako je bar jedan true
not	logičko NOT	negacija

- Operatori pripadnosti u Python-u testiraju pripadnost u sekvenci (string, lista, n-torka, rečnik, skup). Operatora pripadnosti su: `in` i `not in`.

```
a = 10
b = 20
list = [1, 2, 3, 4, 5 ];

if ( a in list ):
   print ("Line 1 - a is available in the given list")
else:
   print ("Line 1 - a is not available in the given list")

if ( b not in list ):
   print ("Line 2 - b is not available in the given list")
else:
   print ("Line 2 - b is available in the given list")
a = 2
if ( a in list ):
   print ("Line 3 - a is available in the given list")
else:
   print ("Line 3 - a is not available in the given list")
```

Python: operatori

- Operatori identiteta proveravaju memorijske lokacije dva objekta:
  - is — proverava da li obe promenljive gledaju na isti objekt
    x is y je 1 ako je id(x) jednak id(y).
  - is not — suprotno od is.

```
a = 20
b = 20
if ( a is b ): print ("Line 1 - same identity")
else: print ("Line 1 - do not have same identity")
if ( id(a) == id(b) ): print ("Line 2 - same identity")
else: print ("Line 2 - do not have same identity")
b = 30
if ( a is b ): print ("Line 3 - have same identity")
else: print ("Line 3 - do not have same identity")
if ( a is not b ): print ("Line 4 - do not have same identity")
else: print ("Line 4 - a and b have same identity")
```

```
>>> l1=[1]
>>> l2=[1]
>>> print(id(l1)==id(l2))


False
```

```
>>> l1=[1]
>>> l2=[1]
>>> l2=l1
>>> print(id(l1)==id(l2))

True
```

Python, prioritet opratora

- Prioritet operatora je kao što sledi ali je bolje koristiti zagrade kako bi se izbegli potencijalni bagovi:

**
~ + -
* / % //
+ -
>> <<
&
^ |
<= < > >=
== !=
= %= /= //= -= += *= **=
is is not
in not in
not or and

Python: kontrola toka

- Za kontrolu toka Python razlikuje 0 i null sa jedne strane i nešto različito od 0 ili null sa druge strane.
- Primer naredbe if:

```
var1 = 100
if var1:
   print ("1 - Got a true expression value")
   print (var1)
var2 = 0
if var2:
   print ("2 - Got a true expression value")
   print (var2)
print ("Good bye!")
```

```
1 - Got a true expression value
100
Good bye!
```

- Primer naredbe if else

```
var1 = 0
if var1:
   print ("1 - true expression value")
   print (var1)
else:
   print ("2 - false expression value")
   print (var1)
```

```
2 - false expression value
0
```

Python: kontrola toka

- Primer naredbe elif (trivijalno neoptimizovano):

```
var = 100
if var == 200:
   print ("1 - Got a true expression value")
   print (var)
elif var == 150:
   print ("2 - Got a true expression value")
   print (var)
elif var == 100:
   print ("3 - Got a true expression value")
   print (var)
else:
   print ("4 - Got a false expression value")
   print (var)
```

- Gnežđenje uslova (razmotrite slučajeve):

```
var = 100
if var < 200:
   print ("Expression value is less than 200")
   if var == 150:
      print ("Which is 150")
   elif var == 100:
      print ("Which is 100")
   elif var < 50:
     print ("Expression value is less than 50")
else:
   print ("Could not find true expression")
```

Python: petlje

- Jedna naredba u suite-u:

```
var = 100
if ( var  == 100 ) : print ("Value of expression is 100")
print ("Good bye!")
```

- Primer petlje while:

```
count = 0
while (count < 9):
  print ('The count is:', count)
  count = count + 1
```

- Primer beskonačne while petlje:

```
var = 1
while var == 1 :  # infinite loop
  num = int(input("Enter a int number  :"))
  print ("You entered: ", num)
```

- Primer kombinacije while i else:

```
count = 0
while count < 5:
  print (count, " is less than 5")
  count = count + 1
else: print (count, " is not less than 5")
```

Python: petlje

- Primer for petlje:

```
for letter in 'Python':     # slovo po slovo
        print ('Current Letter :', letter)
fruits = ['banana', 'apple',  'mango']
for fruit in fruits:        # clan po clan
        print ('Current fruit :', fruit)
```

- Primer for petlje po indeksu:

```
fruits = ['banana', 'apple', 'mango']
for index in range(len(fruits)):
        print ('Current fruit :', fruits[index])
```

- Primer kombinacije for petlje i else:

```
for num in range(10,20):                # od 10 do 19
   for i in range(2,num):               # trazi se delitelj
      if num%i == 0:                    # prvi faktor
         j=num/i                        # drugi factor
         print ('%d equals %d * %d' % (num,i,j))
         break # probaj sledeci broj
   else:                        # else deo druge for petlje
        print (num, 'is a prime number')
```

Python: petlje

- Gnežđenje petlji:

```
i = 2
while(i < 100):
   j = 2
   while(j <= (i/j)):
      if not(i%j): break
      j = j + 1
   else : print (i, " is prime")
   i = i + 1
```

- Prekid petlje pomoću break:

```
for letter in 'Python':     # break u for petlji
   if letter == 'h':
      break
   print ('Current Letter :', letter)
var = 10                    # break u while petlji
while var > 0:
   print ('Current variable value :', var)
   var -= 1

   if var == 5: break
```

Python: prekid petlje

- Korišćenje continue za prelazak na sledeću iteraciju:

```
for letter in 'Python':     # continue u for petlji
   if letter == 'h':
      continue
   print ('Current Letter :', letter)
var = 10                    # continue u while petlji
while var > 0:
   if var == 5:
      continue
   print ('Current variable value :', var)
   var = var -1
```

Korišćenje pass:

```
for letter in 'Python':     # pass u for petlji
   if letter == 'h':
      pass; print ('This is pass block')
   print ('Current Letter :', letter)
 if True:pass # nista ako je true
 else: print('samo za else')
```

```
Current Letter : P
Current Letter : y
Current Letter : t
This is pass block
Current Letter : h
Current Letter : o
Current Letter : n
```

- Uslovna dodela: a `if` b `else` c

```
odg = 'podne je' if sat==12 else 'nije podne'
```

Python: brisanje referenci

- Brojevi u Python-u su neizmenljivi (immutable) što znači da promena vrednosti tipa broj rezultira novim alociranim objektom.
- Objekti brojeva se kreiraju kada im se dodeli vrednost.

```
var1 = 1
var2 = 10
```

- Brisanje reference na brojeve radi naredba del:

```
del var1[,var2[,var3[....,varN]]]
```

primer

```
del var
del var_a, var_b
```

- Brisanje reference na listu, skup, rečnik, n-torku se izvodi kao i za brojeve.
- Razmotrite sledeći kod:

```
lista = [1,2,3]
print(lista)
tuple(lista)
print(lista)
del lista
lista
```

Python: skup

- Skup predstavlja kolekciju nepromenljivih vrednosti bez duplikata.
- Karakteristično je kao što sledi:
  - len(s) kardinatlitet
  - x in s pripadnost
  - x not in s nepripadnost
  - s.issubset(t) s<=t, podskup
  - s.issuperset(t) s>=t, nadskup
  - s.union(t) s | t, unija
  - s.intersection(t) s & t, presek
  - s.difference(t) s - t, razlika
  - s.symmetric_difference(t) s ^ t, skup ekskluzivnih elemenata gledano do na skupove
  - s.copy() novi kopirani skup

```
>>> A={1,2,3}
    >>> B={3,4,5}
    >>> C=A^B
    >>> D=A-B
```

Python: skup

- Set podržava
  dok
  frozenset NE podržava sledeće:
  - s.update(t) — s |= t, dodela unije
  - s.intersection_update(t) — s &= t, dodela preseka
  - s.difference_update(t) — s -= t, dodela razlike
  - s.symmetric_difference_update(t) — s ^= t, dodela ekskluzivnih elemenata do na skupove
  - s.add(x) — dodavanje elementa x
  - s.remove(x) — uklanjanje elementa; ako istog nema nastaje KeyError
  - s.discard(x) — uklanja element x ako postoji
  - s.pop() — uklanja i vraća prvi element iz s; ako je skup prazan nastaje KeyError
  - s.clear() — uklanjanje svih elemenata skupa

```
>>> A={1,2,3}
>>> B={3,4,5}
>>> A.update(B)
```