PROGRAMIRANJE U INTEGRISANIM TEHNOLOGIJAMA

Oznaka predmeta: PIT

Predavanje broj: 01

Nastavna jedinica: PYTHON,

Nastavne teme:

Uvod. Python Interactive Shell. Konvencije. Tipovi. Operatori. Konverzije. Kontrola toka. Naredbe. Petlje. Brojevi. Brisanje reference. Matematičke funkcije. Biblioteka math. Biblioteka random. Stringovi: operatori, formatiranje, metode.

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

Literatura:

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

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.
>>>
```

na Linuxu python3

- 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“
```

- 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>
option -o requires argument
option is o

$ 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 ("option is ",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 None sa jedne strane i nešto različito od 0 ili None 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)
```

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

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) kardinalitet
  - 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)
```

Python: matematičke funkcije, math

Funkcija	Vraća
`abs(x)`	Apsolutno x
`math.ceil(x)`	Najbliži veći integer od broja x
`cmp(x, y)`	-1 if x < y, 0 if x == y, or 1 if x > y, u Pythonu 3 napisati `def cmp(a,b): return (a > b) - (a < b)`
`math.exp(x)`	e^x
`math.fabs(x)`	Apsolutno x
`math.floor(x)`	Nabliži manji integer od broja x
`math.log(x)`	ln x, za x> 0
`math.log10(x)`	log x, za x> 0 .
`max(x1, x2,...)`	Najveći argument
`min(x1, x2,...)`	Namjanji argument
`math.modf(x)`	Kreira uređeni par realnih brojeva (frakcije, celobrojnog deo), *oba dela imaju isti predznak kao x.*
`pow(x,y),math.pow(x,y)`	x**y gde je rezultat u drugom slučaju obavezno float
`round(x [,n])`	**x** zaokružen na n decimalnih cifara iza decim. tačke.
`math.sqrt(x)`	The square root of x for x > 0

Python: primeri matematičkih funkcija

```
print ("abs(-45) : ", abs(-45))
print ("abs(100.12) : ", abs(100.12))
print ("abs(119) : ", abs(119))
```

```
        abs(-45) :  45
        abs(100.12) :  100.12
        abs(119) :  119
```

```
import math   # This will import math module

print ("math.ceil(-45.17) : ", math.ceil(-45.17))
print ("math.ceil(100.12) : ", math.ceil(100.12))
print ("math.ceil(100.72) : ", math.ceil(100.72))
print ("math.ceil(119) : "   , math.ceil(119))
print ("math.ceil(math.pi) : ", math.ceil(math.pi))
```

```
        math.ceil(-45.17) :  -45.0
        math.ceil(100.12) :  101.0
        math.ceil(100.72) :  101.0
        math.ceil(119) :  119.0
        math.ceil(math.pi) : 4.0
```

Python: primeri matematičkih funkcija

```
def cmp(a, b):
    return (a > b) - (a < b) # za Python 3
print ("cmp(80, 100) : ", cmp(80, 100))
print ("cmp(-80, 100) : ", cmp(-80, 100))
print ("cmp(80, -100) : ", cmp(80, -100))
          cmp(80, 100) :  -1
          cmp(-80, 100) :  -1
          cmp(80, -100) :  1
```

```
import math   # This will import math module

print ("math.exp(-45.17) : ", math.exp(-45.17))
print ("math.exp(100.12) : ", math.exp(100.12))
print ("math.exp(100.72) : ", math.exp(100.72))
print ("math.exp(119) : ", math.exp(119))
print ("math.exp(math.pi) : ", math.exp(math.pi))
          math.exp(-45.17) :  2.41500621326e-20
          math.exp(100.12) :  3.03084361407e+43
          math.exp(100.72) :  5.52255713025e+43
          math.exp(119) :  4.7978133273e+51
          math.exp(math.pi) :  23.1406926328
```

Python: primeri matematičkih funkcija

```
import math   # This will import math module
print ("math.fabs(-45.17) : ", math.fabs(-45.17))
print ("math.fabs(100.12) : ", math.fabs(100.12))
print ("math.fabs(100.72) : ", math.fabs(100.72))
print ("math.fabs(119) : ", math.fabs(119))
print ("math.fabs(math.pi) : ", math.fabs(math.pi))
math.fabs(-45.17) :  45.17
math.fabs(100.12) :  100.12
math.fabs(100.72) :  100.72
math.fabs(119) :  119.0
math.fabs(math.pi) :  3.14159265359
```

```
import math   # This will import math module
print ("math.floor(-45.17) : ", math.floor(-45.17))
print ("math.floor(100.12) : ", math.floor(100.12))
print ("math.floor(100.72) : ", math.floor(100.72))
print ("math.floor(119) : ", math.floor(119))
print ("math.floor(math.pi) : ", math.floor(math.pi))
          math.floor(-45.17) :  -46.0
          math.floor(100.12) :  100.0
          math.floor(100.72) :  100.0
          math.floor(119) :  119.0
          math.floor(math.pi) :  3.0
```

Python: primeri matematičkih funkcija

```
import math   # This will import math module

print ("math.log(100.12) : ", math.log(100.12))
print ("math.log(100.72) : ", math.log(100.72))
print ("math.log(119) : ", math.log(119))
print ("math.log(math.pi) : ", math.log(math.pi))
          math.log(100.12) :  4.60636946656
          math.log(100.72) :  4.61234438974
          math.log(119) :  4.77912349311
          math.log(math.pi) :  1.14472988585
```

```
import math   # This will import math module

print ("math.log10(100.12) : ", math.log10(100.12))
print ("math.log10(100.72) : ", math.log10(100.72))
print ("math.log10(119) : ", math.log10(119))
print ("math.log10(math.pi) : ", math.log10(math.pi))
          math.log10(100.12) :  2.00052084094
          math.log10(100.72) :  2.0031157171
          math.log10(119) :  2.07554696139
          math.log10(math.pi) :  0.497149872694
```

Python: primeri matematičkih funkcija

```
print ("max(80, 100, 1000) : ", max(80, 100, 1000))
print ("max(-20, 100, 400) : ", max(-20, 100, 400))
print ("max(-80, -20, -10) : ", max(-80, -20, -10))
print ("max(0, 100, -400) : ", max(0, 100, -400))
          max(80, 100, 1000) :  1000
          max(-20, 100, 400) :  400
          max(-80, -20, -10) :  -10
          max(0, 100, -400) :  100
```

```
print ("min(80, 100, 1000) : ", min(80, 100, 1000))
print ("min(-20, 100, 400) : ", min(-20, 100, 400))
print ("min(-80, -20, -10) : ", min(-80, -20, -10))
print ("min(0, 100, -400) : ", min(0, 100, -400))
          min(80, 100, 1000) :  80
          min(-20, 100, 400) :  -20
          min(-80, -20, -10) :  -80
          min(0, 100, -400) :  -400
```

Python: primeri matematičkih funkcija

```
import math   # This will import math module

print ("math.modf(100.12) : ", math.modf(100.12))
print ("math.modf(-100.72) : ", math.modf(-100.72))
print ("math.modf(119) : ", math.modf(119))
print ("math.modf(math.pi) : ", math.modf(math.pi))
          math.modf(100.12) :  (0.12000000000000455, 100.0)
          math.modf(-100.72) :  (-0.7199999999999886, -100.0)
          math.modf(119) :    (0.0, 119.0)
          math.modf(math.pi) :  (0.14159265358979312, 3.0)
```

```
import math   # This will import math module

print ("math.pow(100, 2) : ", math.pow(100, 2))
print ("math.pow(100, -2) : ", math.pow(100, -2))
print ("math.pow(2, 4) : ", math.pow(2, 4))
print ("math.pow(3, 0) : ", math.pow(3, 0))
          math.pow(100, 2) :  10000.0
          math.pow(100, -2) :  0.0001
          math.pow(2, 4) :  16.0
          math.pow(3, 0) :  1.0
```

Python: primeri matematičkih funkcija

```
print ("round(80.23456, 2) : ", round(80.23456, 2))
print ("round(100.000056, 3) : ", round(100.000056, 3))
print ("round(-100.000056, 3) : ", round(-100.000056, 3))
```

```
    round(80.23456, 2) :  80.23
    round(100.000056, 3) :  100.0
    round(-100.000056, 3) :  -100.0
```

```
import math   # This will import math module

print ("math.sqrt(100) : ", math.sqrt(100))
print ("math.sqrt(7) : ", math.sqrt(7))
print ("math.sqrt(math.pi) : ", math.sqrt(math.pi))
```

```
    math.sqrt(100) :  10.0
    math.sqrt(7) :  2.64575131106
    math.sqrt(math.pi) :  1.77245385091
```

Python: modul random

- Funkcije slučajnih brojeva:

Funkcija modula random	Opis
`choice(seq)`	Slučajni element liste, n-torke ili stringa.
`randrange([start,]stop[,step])`	Slučajno selektovan element opsega (start, stop, step)
`random()`	Slučajan float x takav da je `0 <= x < 1`
`seed([x])`	Postavlja celobrojnu vrednost koja se koristi za generator slučajnih brojeva. Pozovite ovu funkciju pre bilo koje funkcije random modula. Ne vraća vrednost.
`shuffle(lst)`	Promeša redosled elemenata liste. Ne vraća vrednost.
`uniform(x, y)`	Uniformna distribucija slučajnog broja r gde je `x <= r < y`

- Slučajni brojevi se koriste za igre, simulacije, testiranje, zaštitu, privatnost...

Python: funkcije slučajnih brojeva

```
import random
print ("choice([1, 2, 3, 5, 9]) : ", random.choice([1, 2, 3, 5, 9]))
print ("choice('A String') : ", random.choice('A String'))
          choice([1, 2, 3, 5, 9]) :  2
          choice('A String') :  n
```

```
import random
# Select an even number in 100 <= number < 1000
print ("randrange(100, 1000, 2) : ", random.randrange(100, 1000, 2))
# Select another number in 100 <= number < 1000
print ("randrange(100, 1000, 3) : ", random.randrange(100, 1000, 3))
          randrange(100, 1000, 2) :  976
          randrange(100, 1000, 3) :  520
```

```
import random
# First random number
print ("random() : ", random.random())
# Second random number
print ("random() : ", random.random())
          random() :  0.281954791393
          random() :  0.309090465205
```

Python: funkcije slučajnih brojeva

```
import random
random.seed( 10 )
print ("Random number with seed 10 : ", random.random())
random.seed( 10 )
print ("Random number with seed 10 : ", random.random())
```

```
          Random number with seed 10 :  0.57140259469
          Random number with seed 10 :  0.57140259469
```

```
import random
list = [20, 16, 10, 5];
random.shuffle(list)
print ("Reshuffled list : ",  list)
random.shuffle(list)
print ("Reshuffled list : ",  list)
```

```
          Reshuffled list :  [16, 5, 10, 20]
          Reshuffled list :  [20, 16, 5, 10]
```

```
import random
print ("Random Float uniform(5, 10) : ",  random.uniform(5, 10))
print ("Random Float uniform(7, 14) : ",  random.uniform(7, 14))
```

```
          Random Float uniform(5, 10) :  5.52615217015
          Random Float uniform(7, 14) :  12.5326369199
```

Python: trigonometrijske funkcije, math

Funkcija	Opis
`math.acos(x)`	Arkus kosinus od x u radijanima.
`math.asin(x)`	Arkus sinus od x u radijanima.
`math.atan(x)`	Arkus tangens od x u radijanima.
`math.atan2(y, x)`	Arkus tangens kao f(kateta_y,kateta_x) u radijanima.
`math.cos(x)`	Kosinus od x radijana.
`math.hypot(x, y)`	Euklidska norma, sqrt(x*x + y*y), hipotenuza.
`math.sin(x)`	Sinus od x radijana.
`math.tan(x)`	Tangens od x radijana.
`math.degrees(x)`	Konverzija iz radijana u stepene.
`math.radians(x)`	Konverzija iz stepena u radijane.

- Matematičke konstante:
  - `math.pi 3.14159 26535 89793 23846 26433 83279 50288 ...`
  - `math.e  2.71828 18284 59045 23536 02874 71352 66249 ...`

Python: trigonometrijske funkcije

```
import math
print ("acos(0.64) : ",  math.acos(0.64))
print ("acos(0) : ",  math.acos(0))
print ("acos(-1) : ",  math.acos(-1))
print ("acos(1) : ",  math.acos(1))
          acos(0.64) :  0.876298061168
          acos(0) :  1.57079632679
          acos(-1) :  3.14159265359
          acos(1) :  0.0
```

```
import math
print ("asin(0.64) : ",  math.asin(0.64))
print ("asin(0) : ",  math.asin(0))
print ("asin(-1) : ",  math.asin(-1))
print ("asin(1) : ",  math.asin(1))
          asin(0.64) :  0.694498265627
          asin(0) :  0.0
          asin(-1) :  -1.57079632679
          asin(1) :  1.57079632679
```

Python: trigonometrijske funkcije

```
import math
print ("atan(0.64) : ",  math.atan(0.64))
print ("atan(0) : ",  math.atan(0))
print ("atan(10) : ",  math.atan(10))
print ("atan(-1) : ",  math.atan(-1))
print ("atan(1) : ",  math.atan(1))
          atan(0.64) :  0.569313191101
          atan(0) :  0.0
          atan(10) :  1.4711276743
          atan(-1) :  -0.785398163397
          atan(1) :  0.785398163397
```

```
import math
print ("atan2(-0.50,-0.50) : ",  math.atan2(-0.50,-0.50))
print ("atan2(0.50,0.50) : ",  math.atan2(0.50,0.50))
print ("atan2(5,5) : ",  math.atan2(5,5))
print ("atan2(-10,10) : ",  math.atan2(-10,10))
print ("atan2(10,20) : ",  math.atan2(10,20))
          atan2(-0.50,-0.50) :  -2.35619449019
          atan2(0.50,0.50) :  0.785398163397
          atan2(5,5) :  0.785398163397
          atan2(-10,10) :  -0.785398163397
          atan2(10,20) :  0.463647609001
```

Python: trigonometrijske funkcije

```
import math
print ("cos(3) : ",  math.cos(3))
print ("cos(-3) : ",  math.cos(-3))
print ("cos(0) : ",  math.cos(0))
print ("cos(math.pi) : ",  math.cos(math.pi))
print ("cos(2*math.pi) : ",  math.cos(2*math.pi))
```

```
cos(3) :  -0.9899924966
cos(-3) :  -0.9899924966
cos(0) :  1.0
cos(math.pi) :  -1.0
cos(2*math.pi) :  1.0
```

```
import math
print ("hypot(3, 2) : ",  math.hypot(3, 2))
print ("hypot(-3, 3) : ",  math.hypot(-3, 3))
print ("hypot(0, 2) : ",  math.hypot(0, 2))
```

```
hypot(3, 2) :  3.60555127546
hypot(-3, 3) :  4.24264068712
hypot(0, 2) :  2.0
```

Python: trigonometrijske funkcije

```
import math
print ("sin(3) : ",  math.sin(3))
print ("sin(-3) : ",  math.sin(-3))
print ("sin(0) : ",  math.sin(0))
print ("sin(math.pi) : ",  math.sin(math.pi))
print ("sin(math.pi/2) : ",  math.sin(math.pi/2))
          sin(3) :  0.14112000806
          sin(-3) :  -0.14112000806
          sin(0) :  0.0
          sin(math.pi) :  1.22460635382e-16
          sin(math.pi/2) :  1
print ("tan(3) : ",  math.tan(3))
print ("tan(-3) : ",  math.tan(-3))
print ("tan(0) : ",  math.tan(0))
print ("tan(math.pi) : ",  math.tan(math.pi))
print ("tan(math.pi/2) : ",  math.tan(math.pi/2))
print ("tan(math.pi/4) : ",  math.tan(math.pi/4))
          tan(3) :  -0.142546543074
          tan(-3) :  0.142546543074
          tan(0) :  0.0
          tan(math.pi) :  -1.22460635382e-16
          tan(math.pi/2) :  1.63317787284e+16
          tan(math.pi/4) :  1.0
```

Python: trigonometrijske funkcije

```
import math
print ("degrees(3) : ",  math.degrees(3))
print ("degrees(-3) : ",  math.degrees(-3))
print ("degrees(0) : ",  math.degrees(0))
print ("degrees(math.pi) : ",  math.degrees(math.pi))
print ("degrees(math.pi/2) : ",  math.degrees(math.pi/2))
print ("degrees(math.pi/4) : ",  math.degrees(math.pi/4))
```

```
degrees(3) :  171.887338539
degrees(-3) :  -171.887338539
degrees(0) :  0.0
degrees(math.pi) :  180.0
degrees(math.pi/2) :  90.0
degrees(math.pi/4) :  45.0
```

```
print ("radians(3) : ",  math.radians(3))
print ("radians(-3) : ",  math.radians(-3))
print ("radians(0) : ",  math.radians(0))
print ("radians(math.pi) : ",  math.radians(math.pi))
print ("radians(math.pi/2) : ",  math.radians(math.pi/2))
print ("radians(math.pi/4) : ",  math.radians(math.pi/4))
```

```
radians(3) :  0.0523598775598
radians(-3) :  -0.0523598775598
radians(0) :  0.0
radians(math.pi) :  0.0548311355616
radians(math.pi/2) :  0.0274155677808
radians(math.pi/4) :  0.0137077838904
```

Python: stringovi

- Kreiranje stringa se vrši jednostavnom dodelom stringa varijabli:

```
var1 = 'Hello World!'
var2 = "Python Programming"
```

Pristupanje karakteru stringa:

```
print ("var1[0]: ", var1[0])
print ("var2[1:5]: ", var2[1:5])
```

```
var1[0]:  H
var2[1:5]:  ytho
```

- Ažuriranje stringa:

```
var1 = 'Hello World!'
print ("Updated String :- ", var1[:6] + 'Python')
```

```
Updated String :-  Hello Python
```

Escape karakteri:

\a (0x07) Bell	\b (0x08) Backspace	\cx \C-x	Control-x
\e (0x1b) Escape	\f (0x0c) Formfeed	\M-\C-x	Meta-Control-x
\n (0x0a) Newline	\nnn Octal notation	\r (0x0d)	Carriage return
\s (0x20) Space	\t (0x09) Tab	\v (0x0b)	Vertical tab
\x Character x	\xnn Hexadecimal notation		

Python: string, specijalni operatori

- Neka je **a**='Hello' i neka je **b=**'Python'

Operator	Opis	Primer
+	Konkatenacija	a + b HelloPython
*	Ponavaljanje	a*2 HelloHello
[]	Slice - uzima karakter datog indeksa	a[1] daje e
[:]	Range Slice - uzima karaktere datog opsega	a[1:4] daje ell
in	Membership - true ako karakter pripada stringu	H in a True
not in	Membership - suprotno od ptrehodnog	M not in a True
r/R	Raw String - ukida značenje escape karaktera. Ovaj operator (r ili R) ide pre prvog navoda stringa.	print (r'\n') \n print (R'\n') \n
%	Formatiranje stringa.	Sledi u nastavku.

Python: formatiranje stringa

- Primer:

```
print ("My name is %s and weight is %d kg!" % ('Elvis', 90))
My name is Elvis and weight is 90 kg!
```

Simbol za formatiranje	Konverzija
%c	character
%s	string
%i	signed decimal integer
%d	signed decimal integer
%u	unsigned decimal integer
%o	octal integer
%x	hexadecimal integer (lowercase letters)
%X	hexadecimal integer (UPPERcase letters)
%e	exponential notation (with lowercase 'e')
%E	exponential notation (with UPPERcase 'E')
%f	floating point real number
%g	general shorter of %f and %e, "%15.8g"
%G	general shorter of %f and %E

Python: formatiranje stringa

Dodatni simboli	Funkcionalnost
*	Specificira širinu ili preciznost
-	Levo poravnanje.
+	Prikaz predznaka.
<sp>	Ostavljanje praznog mesta pre pozitivnog broja.
#	Dodavanje oktalne vodeće nule ('0') ili heksadecimlanog vodećeg stringa '0x' ili '0X', zavisno od toga da li je korišćen 'x' ili 'X'.
0	Popunjavanje nulama sleva umesto praznih mesta.
%	'%%' ostavlja jedan literal '%'
(var)	Mapiranje varijable (argumenti rečnika)
m.n	Format u kom je m minimalna totalna širina, a n je broj cifara iza decimalne tačke.

Python: formatiranje stringa

Primer načina	Izlaz

`print('%s %s' % ('one', 'two'))`
`print('{} {}'.format('one', 'two'))`
`print('{1} {0}'.format('one', 'two'))`
	`one two`
	`one two`
	`two one`

`print('%d %d' % (1, 2))` `print('{} {}'.format(1, 2))`	`1 2` `1 2`

`class Data(object):`
`  def __str__(self): return 'str'`
`  def __repr__(self):return 'räpr'`
`print('%s %r' % (Data(), Data()))`
`print('{0!s} {0!r}'.format(Data()))`
`print('{0!r} {0!a}'.format(Data()))`
	
	`str räpr`
	`str räpr`
	`räpr r\xe4pr`

`print('%10s' % ('test',))` `print('{:>10}'.format('test'))`	`      test (ukupno 10 znakova)` `      test (ukupno 10 znakova)`
`print('%-10s' % ('test',))` `print('{:10}'.format('test'))`	`test       (ukupno 10 znakova)` `test       (ukupno 10 znakova)`
`print('%*s' % ((- 8), 'test'))` `print('{:<{}}'.format('test', 8))`	`test     (ukupno 8 znakova)` `test     (ukupno 8 znakova)`

Python: formatiranje stringa

Primer načina	Izlaz
`print('{:.<10}'.format('test'))`	`test......`
`print('{:^10}'.format('test'))`	`   test   (10 znakova)`
`print('%.5s' % ('xylophone',))` `print('{:.5}'.format('xylophone'))`	`xylop` `xylop`
`print('%.*s' % (7, 'xylophone'))` `print('{:.{}}'.format('xylophone', 7))`	`xylopho` `xylopho`
`print('%-10.5s' % ('xylophone',))` `print('{:10.5}'.format('xylophone'))`	`xylop     (10 znakova)` `xylop     (10 znakova)`
`print('%d' % (42,))` `print('{:d}'.format(42))`	`42` `42`
`print('%f' % (3.141592653589793,))` `print('{:f}'.format(3.141592653589793))`	`3.141593` `3.141593`
`print('%4d' % (42,))` `print('{:4d}'.format(42))`	`  42 (4 znaka)` `  42 (4 znaka)`
`print('%06.2f' % (3.1415926535,))` `print('{:06.2f}'.format(3.1415926535))`	`003.14` `003.14`

Python: formatiranje stringa

Primer načina	Izlaz
`print('%+d' % (42,))` `print('{:+d}'.format(42))`	`+42` `+42`

`print('% d' % ((- 23),))`
`print('{: d}'.format((- 23)))`
`print('% d' % (42,))`
`print('{: d}'.format(42))`
	`-23`
	`-23`
	` 42  ( predznak)`
	` 42  ( predznak)`

`print('{:=5d}'.format((- 23)))`	`-  23  (5 znakova)`

`data = {'first':'Pr','last':'Dr'}`
`print('%(first)s %(last)s' % data)`
`print('{first} {last}'.format(**data))`
`print('{first} {last}'.format(first='Pr', last='Dr'))`
	
	`Pr Dr`
	`Pr Dr`
	`Pr Dr`

`person = {'first': 'Pera', 'last': 'Detlic'}` `print('{p[first]} {p[last]}'.format(p=person))`	 `Pera Detlic`
`data = [4, 8, 15, 16, 23, 42]` `print('{d[4]} {d[5]}'.format(d=data))`	 `23 42`

`class Plant(object):`
`    type = 'tree'`
`print('{p.type}'.format(p=Plant()))`
	
	`tree`

Python: formatiranje stringa

Primer načina	Izlaz

`class Plant(object):`
`  type = 'tree'`
`  kinds = [{'name': 'oak'}, {'name': 'maple'}]`
`print('{p.type}:{p.kinds[0][name]}'.format(p=Plant()))`
	`tree:oak`

`from datetime import datetime`
`print(`
`'{:%Y-%m-%d %H:%M}'.format(datetime(2001, 2, 3, 4, 5)))`
	`2001-02-03 04:05`

`class HAL9000(object):`
`  def __format__(self, format):`
`  if (format == 'open-the-pod-bay-doors'):`
`  return "I'm afraid I can't do that."`
`  return 'HAL 9000'`
`print('{:open-the-pod-bay-doors}'.format(HAL9000()))`
	`I'm afraid I can't do that.`

Python: formatiranje stringa

- Trostruki navodnici (ili apostrofi) omogućuju protezanje stringa na više redova:

```
para_str = """this is a long string that is made up of
several lines and non-printable characters such as
TAB ( \t ) and they will show up that way when displayed.
NEWLINEs within the string, whether explicitly given like
this within the brackets [ \n ], or just a NEWLINE within
the variable assignment will also show up.
"""
print (para_str);
```

```
this is a long string that is made up of
several lines and non-printable characters such as
TAB (            ) and they will show up that way when displayed.
NEWLINEs within the string, whether explicitly given like
this within the brackets [
 ], or just a NEWLINE within
 the variable assignment will also show up.
```

```
print ('C:\\nowhere')
   print (r'C:\\nowhere')
```

```
C:\nowhere
C:\\nowhere
```

Python: formatiranje stringa, metode stringa

- Korišćenje unicode stringa:

```
print (u'Hello, world!')
```

```
Hello, world
```

Metode stringa:

- capitalize()

```
str = "this is string example....wow!!!";
print ("str.capitalize() : ", str.capitalize())
```

```
str.capitalize() : This is string example....wow!!!
```

- center(width, fillchar)

```
str = "this is string example....wow!!!";
print ("str.center(40, 'a') : ", str.center(40, 'a'))
```

```
str.center(40, 'a') : aaaathis is string example....wow!!!aaaa
```

- count(substr, beg=0,end=len(string))

```
str = "this is string example....wow!!!";
sub = "i";
print ("str.count(sub, 4, 40) : ", str.count(sub, 4, 40))
sub = "wow"; print ("str.count(sub) : ", str.count(sub))
```

```
str.count(sub, 4, 40) : 2
str.count(sub, 4, 40) : 1
```

Python, metode stringa

```
# u Python-u 3.5, probajte u interaktivnom promptu
from base64 import *
str = 'Python'
sen = b64encode(str.encode()) # b'UHl0aG9u'
print(bytes(sen).decode('utf-8'))
sde = b64decode(sen)          # b'Python'
print(bytes(sde).decode('utf-8'))
```

```
UHl0aG9u
Python
```

- `endswith(suffix, beg=0, end=len(string))`
  –proverava da li string završava datim sufiksom u datom podstringu (podrazumevano ceo string)

Python, metode stringa

```
str = "this is string example....wow!!!";
suffix = "wow!!!";
print (str.endswith(suffix));
print (str.endswith(suffix,20));
suffix = ("is"," ");
print (str.endswith(suffix, 2, 4));
print (str.endswith(suffix, 2, 7));
```

```
True
True
True
True
```

- expandtabs(tabsize=8)

```
str = "this is\tstring example....wow!!!";
print ("Original string: " + str);
print ("Defualt exapanded tab: " + str.expandtabs(0));
print ("12 exapanded tab: " + str.expandtabs(12));
```

```
Original string: this is       string example....wow!!!
Defualt exapanded tab: this isstring example....wow!!!
12 exapanded tab: this is           string example....wow!!!
```

Python, metode stringa

- find(substr, beg=0 end=len(string))

```
str1 = "this is string example....wow!!!";
str2 = "exam";
print (str1.find(str2));
print (str1.find(str2, 10));
print (str1.find(str2, 40));
```

```
15
15
-1
```

- index(substr, beg=0, end=len(string))
  - za iste vrednosti str1 i str2 kao u prethodnom slučaju

```
print (str1.index(str2));
print (str1.index(str2, 10));
print (str1.index(str2, 40));
```

```
15
15
Traceback (most recent call last):
File "test.py", line 8, in print
str1.index(str2, 40); ValueError:
substring not found
shell returned 1
```

- isalnum()

```
str = "this2009"; # bez razmaka
print (str.isalnum());
str = "this is string example....wow!!!"; print (str.isalnum());
```

```
True
False
```

- isalpha()

```
str = "this"; # bez razmaka i samo slova
print (str.isalnum());
str = "this is string example....wow!!!"; print (str.isalpha());
```

```
True
False
```

Python, metode stringa

- isdigit()

```
str = "123456"; # Only digit in this string
print (str.isdigit()); # True
str = "this is string !!!"; print (str.isdigit()); #False
```

- islower()

```
str = "THIS is string example....wow!!!";
print (str.islower()); # False
str = "this is string example....wow!!!";
print (str.islower()); # True
```

- isnumeric()

```
str = u"this2009";
print (str.isnumeric()); # False
str = u"23443434";
print (str.isnumeric()); # True
```

- isspace()

```
str = "     ";
print (str.isspace()); # True
str = "This is string example....wow!!!";
print (str.isspace()); # False
```

Python, metode stringa

- istitle()

```
str = "This Is String Example...Wow!!!";
print (str.istitle()); # True
str = "This is string example....wow!!!";
print (str.istitle()); # False
```

- isupper()

```
str = "THIS IS STRING EXAMPLE....WOW!!!";
print (str.isupper()); # True
str = "THIS is string example....wow!!!";
print (str.isupper()); # False
```

- join(seq)

```
str = "--";            # spona
seq = ("a", "b", "c"); # sekvenca stringova
print (str.join( seq )); # a--b--c
```

- funkcija: len(string)

```
str = "this is string example....wow!!!";
print ("Length of the string: ", len(str));
```

```
Length of the string:  32
```

Python, metode stringa

- `ljust(width[, fillchar])`

```
str = "this is string example....wow!!!";
print (str.ljust(49, '0'));
```

```
this is string example....wow!!!0000000000000000
```

- `lower()`

```
str = "THIS IS STRING EXAMPLE....WOW!!!";
print (str.lower());
```

```
this is string example....wow!!!
```

- `lstrip(chars=' ')`

```
str = "     this is string example....wow!!! ";
print (str.lstrip());
str = "88888888this is string example....wow!!!8888888";
print (str.lstrip('8'));
```

```
this is string example....wow!!!
this is string example....wow!!!8888888
```

Python, metode stringa

- funkcija: max(str)

```
str = "this is really a string example....wow!!!";
print ("Max character: " + max(str)); # Max character: y
str = "this is a string example....wow!!!";
print ("Max character: " + max(str)); # Max character: x
```

- funkcija: min(str)

```
str = "this-is-real-string-example....wow!!!";
print ("Min character: " + min(str)); # Min character: !
```

- replace(oldsub, newsub [, maxrepl])

```
str = "this is string example....wow!!! this is really string";
print (str.replace("is", "was"));
print (str.replace("is", "was", 3));
      thwas was string example....wow!!! thwas was really string
      thwas was string example....wow!!! thwas is really string
```

- rfind(str, beg=0,end=len(string))

```
str = "this is really a string example....wow!!!"; sss = "is";
print (str.rfind(sss));           # 5
print (str.rfind(sss, 0, 10));    # 5
print (str.rfind(sss, 10, 0));    # -1
print (str.find(sss));            # 2
print (str.find(sss, 0, 10));     # 2
print (str.find(sss, 10, 0));     # -1
```

Python, metode stringa

- rindex(substr, beg=0, end=len(string))

```
str1 = "this is string example....wow!!!";
str2 = "is";
print (str1.rindex(str2)); # 5
print (str1.index(str2));  # 2
```

- rjust(width,[, fillchar])

```
str = "this is string example....wow!!!";
print (str.rjust(50, '0'));
       000000000000000000this is string example....wow!!!
```

- rstrip(chars=' ')

```
str = " this is string example....wow!!!      ";
print (str.rstrip());
str = "88888888this is string example....wow!!!8888888";
print (str.rstrip('8'));
                  this is string example....wow!!!
                  88888888this is string example....wow!!!
```

- split(str=" ", numberofsplits=string.count(str))

```
str = "Line1-abcdef \nLine2-abc \nLine4-abcd";
print (str.split( ));
print (str.split(' ', 1 ));
        ['Line1-abcdef', 'Line2-abc', 'Line4-abcd']
        ['Line1-abcdef', '\nLine2-abc \nLine4-abcd']
```

Python, metode stringa

- splitlines(numlinebreakincluded=string.count('\n'))

```
str = "Line1-a b c d e f\nLine2- a b c\n\nLine4- a b c d";
print (str.splitlines( ));
print (str.splitlines( 0 ));
print (str.splitlines( 3 ));
print (str.splitlines( 4 ));
print (str.splitlines( 5 ));
```

```
['Line1-a b c d e f', 'Line2- a b c', '', 'Line4- a b c d']
['Line1-a b c d e f', 'Line2- a b c', '', 'Line4- a b c d']
['Line1-a b c d e f\n', 'Line2- a b c\n', '\n', 'Line4- a b c d']
['Line1-a b c d e f\n', 'Line2- a b c\n', '\n', 'Line4- a b c d']
['Line1-a b c d e f\n', 'Line2- a b c\n', '\n', 'Line4- a b c d']
```

- startswith(substr, beg=0, end=len(string))

```
str = "this is string example....wow!!!";
print (str.startswith( 'this' ));          # True
print (str.startswith( 'is', 2, 4 ));      # True
print (str.startswith( 'this', 2, 4 ));    # False
```

- strip(chars=' ')

```
str = "0000000this is string example....wow!!!0000000";
print (str.strip( '0' ));
```

```
this is string example....wow!!!
```

Python, metode stringa

- swapcase()

```
str = "This is string example....wow!!!";
print (str.swapcase());
str = "THIS IS STRING EXAMPLE....WOW!!!";
print (str.swapcase());
```

```
tHIS IS STRING EXAMPLE....WOW!!!
this is string example....wow!!!
```

- title()

```
str = "this is string example....wow!!!";
print (str.title());
```

```
This Is String Example....Wow!!!
```

Python, metode stringa

- upper()

```
str = "this is string example....wow!!!";
print (str.upper())
```

```
THIS IS STRING EXAMPLE....WOW!!!
```

- zfill (width)

```
str = "this is string example....wow!!!";
print (str.zfill(40));
print (str.zfill(50));
```

```
00000000this is string example....wow!!!
0000000000000000this is string example....wow!!!
```

- isdecimal()

```
str = u"this2009";
print (str.isdecimal()); # False
str = u"23443434";
print (str.isdecimal()); # True
str = u"2.3443434";
print (str.isdecimal()); # False
```

- Razmislite o monoalfabetskoj substituciji (u vreme Julija Cezara) npr. recipročnog alfabeta.

```
A B C D E F G H I J K L M N O P Q R S T U V W X Y Z
L K J I H G F E D C B A Z Y X W V U T S R Q P O N M
```

Mali primer

- Napisati program za šifrovanje teksta substitucijom.

```
monoalpha = {    'a': 'm', 'b': 'n', 'c': 'b', 'd': 'v', 'e': 'c',
    'f': 'x’,     'g': 'z',
    'h': 'a',    'i': 's',
    'j': 'd',    'k': 'f',
    'l': 'g',    'm': 'h',
    'n': 'j',    'o': 'k',
    'p': 'l',    'q': 'p',
    'r': 'o',    's': 'i',
    't': 'u',    'u': 'y',
    'v': 't',    'w': 'r',
    'x': 'e', 'y': 'w', 'z': 'q', ' ': ' ', '.': '.’
}
```

- Razmislite o upotrebi funkcije zip:

```
>>> list_a = [1, 2, 3, 4]
>>> list_b = [5, 6, 7, 8]
>>> list(zip(list_a, list_b))
[(1, 5), (2, 6), (3, 7), (4, 8)]
```

Mali primer

```
inverse_monoalpha = {}
for key, value in monoalpha.items():
    inverse_monoalpha[value] = key

message = 'Python je interesantan.'
encrypted_message = []
for letter in message:
    encrypted_message.append( monoalpha[letter.lower()] )
strencrypted=''.join(encrypted_message)
print (strencrypted)

# desifrovanje
decrypted_message = []
for letter in strencrypted:
    try:
        decrypted_message.append( inverse_monoalpha[letter] )
    except KeyError:
        decrypted_message.append( letter )
stredecrypted=''.join(decrypted_message)
print (stredecrypted)
```

```
IZLAZ
lwuakj dc sjucocimjumj.
python je interesantan.
```