

FUNKCIONALNO PROGRAMIRANJE

Oznaka predmeta: FPR

Predavanje broj: 02

Nastavna jedinica: PYTHON,

Nastavne teme:

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: matematičke funkcije, math

Funkcija	Vraća
<code>abs(x)</code>	Apsolutno x
<code>math.ceil(x)</code>	Najbliži veći integer od broja x
<code>cmp(x, y)</code>	-1 if $x < y$, 0 if $x == y$, or 1 if $x > y$, u Pythonu 3 napisati <code>def cmp(a,b): return (a > b) - (a < b)</code>
<code>math.exp(x)</code>	e^x
<code>math.fabs(x)</code>	Apsolutno x
<code>math.floor(x)</code>	Nabliži manji integer od broja x
<code>math.log(x)</code>	$\ln x$, za $x > 0$
<code>math.log10(x)</code>	$\log x$, za $x > 0$.
<code>max(x1, x2, ...)</code>	Najveći argument
<code>min(x1, x2, ...)</code>	Namjanji argument
<code>math.modf(x)</code>	Kreira uređeni par realnih brojeva (frakcije, celobrojnog deo), <i>oba dela imaju isti predznak kao x.</i>
<code>pow(x, y), math.pow(x, y)</code>	$x^{**}y$ gde je rezultat u drugom slučaju obavezno float
<code>round(x [,n])</code>	x zaokružen na n decimalnih cifara iza decim. tačke.
<code>math.sqrt(x)</code>	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) :  (0.120000000000000455, 100.0)
        math.modf(-100.72) :  (-0.719999999999999886, -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
<code>choice(seq)</code>	Slučajni element liste, n-torke ili stringa.
<code>randrange([start,]stop[, step])</code>	Slučajno selektovan element opsega (start, stop, step)
<code>random()</code>	Slučajan float x takav da je $0 \leq x < 1$
<code>seed([x])</code>	Postavlja celobrojnu vrednost koja se koristi za generator slučajnih brojeva. Pozovite ovu funkciju pre bilo koje funkcije random modula. Ne vraća vrednost.
<code>shuffle(lst)</code>	Promeša redosled elemenata liste. Ne vraća vrednost.
<code>uniform(x, y)</code>	Uniformna distribucija slučajnog broja r gde je $x \leq r < y$

- 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
<code>math.acos(x)</code>	Arkus kosinus od x u radijanima.
<code>math.asin(x)</code>	Arkus sinus od x u radijanima.
<code>math.atan(x)</code>	Arkus tangens od x u radijanima.
<code>math.atan2(y, x)</code>	Arkus tangens kao f(kateta_y,kateta_x) u radijanima.
<code>math.cos(x)</code>	Kosinus od x radijana.
<code>math.hypot(x, y)</code>	Euklidska norma, $\sqrt{x^2 + y^2}$, hipotenuza.
<code>math.sin(x)</code>	Sinus od x radijana.
<code>math.tan(x)</code>	Tangens od x radijana.
<code>math.degrees(x)</code>	Konverzija iz radijana u stepene.
<code>math.radians(x)</code>	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
*	Ponavljanje	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 prethodnog	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
<pre>print('%s %s' % ('one', 'two')) print('{} {}'.format('one', 'two')) print('{1} {0}'.format('one', 'two'))</pre>	<pre>one two one two two one</pre>
<pre>print('%d %d' % (1, 2)) print('{} {}'.format(1, 2))</pre>	<pre>1 2 1 2</pre>
<pre>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()))</pre>	<pre>str räpr str räpr räpr r\xe4pr</pre>
<pre>print('%10s' % ('test',)) print('{:>10}'.format('test'))</pre>	<pre>test (ukupno 10 znakova) test (ukupno 10 znakova)</pre>
<pre>print('%-10s' % ('test',)) print('{:10}'.format('test'))</pre>	<pre>test (ukupno 10 znakova) test (ukupno 10 znakova)</pre>
<pre>print('%*s' % ((- 8), 'test')) print('{:<{s}'.format('test', 8))</pre>	<pre>test (ukupno 8 znakova) Test (ukupno 8 znakova)</pre>

Python: formatiranje stringa

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

Python: formatiranje stringa

Primer načina	Izlaz
<pre>print('%+d' % (42,)) print('{:+d}'.format(42))</pre>	+42 +42
<pre>print('% d' % ((- 23),)) print('{: d}'.format((- 23))) print('% d' % (42,)) print('{: d}'.format(42))</pre>	-23 -23 42 (predznak) 42 (predznak)
<pre>print('{:=5d}'.format((- 23)))</pre>	- 23 (5 znakova)
<pre>data = {'first': 'Pr', 'last': 'Dr'} print('%(first)s %(last)s' % data) print('{first} {last}'.format(**data)) print('{first} {last}'.format(first='Pr', last='Dr'))</pre>	Pr Dr Pr Dr Pr Dr
<pre>person = {'first': 'Pera', 'last': 'Detlic'} print('{p[first]} {p[last]}'.format(p=person))</pre>	Pera Detlic
<pre>data = [4, 8, 15, 16, 23, 42] print('{d[4]} {d[5]}'.format(d=data))</pre>	23 42
<pre>class Plant(object): type = 'tree' print('{p.type}'.format(p=Plant()))</pre>	tree

Python: formatiranje stringa

Primer načina	Izlaz
<pre>class Plant(object): type = 'tree' kinds = [{'name': 'oak'}, {'name': 'maple'}] print('{p.type}:{p.kinds[0][name]}'.format(p=Plant()))</pre>	tree:oak
<pre>from datetime import datetime print('{:%Y-%m-%d %H:%M}'.format(datetime(2001, 2, 3, 4, 5)))</pre>	2001-02-03 04:05
<pre>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()))</pre>	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

- `base64.b64encode(bajtstring)`
`base64.b64decode(bajtstring)`

```
stari način (ver 2.7) str = "Python";  
stari način (ver 2.7) sen = str.encode('base64','strict'); # UHl0aG9u\n  
stari način (ver 2.7) sde = sen.decode('base64','strict') # Python
```

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 ("Default expanded tab: " + str.expandtabs(0));
print ("Double expanded tab: " + str.expandtabs(12));
```

Original string: this is string example....wow!!!

Default expanded tab: this isstring example....wow!!!

Double expanded 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());
```

True

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

False

- `isalpha()`

```
str = "this"; # bez razmaka i samo slova
```

```
print (str.isalnum());
```

True

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

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
```
- `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!!!000000000000000000
```

- `lower()`

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

- `lstrip()`

```
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

- `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
```
- `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'));`
`00000000000000000000this is string example....wow!!!`
- `rstrip([char])`
`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=" ", numberofplits=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([char])`
 `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!!!  
00000000000000000000this 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', ' ': ' ', '.': '.'  
}
```

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.
```