Python uvod

Python je interpreterski jezik, čija sintaksa liči na pseudo jezik. Python omogućava da se u nekoliko linija na čitljiv način predstave moćne ideje. Trenutno postoje dve različite verzije Python programskog jezika 2.7 i 3.x. , gde je podrška za 2.7 ukinuta 2020. godine. Na ovom kursu koristiće se Python 3.x.
Python možete preuzeti sa linka.
Provera Python verzije se vrši pomoću komande `python -version` u terminalu.
U ovom tutorijalu biće pokrivene:

- Osnove Python: Osnovni tipovi podataka, Liste, Rečnici, Funkcije, Klase
- Numpy biblioteka: Arrays, Array indexing, Datatypes, Array math
- Matplotlib biblioteka: Plotting, Subplots

7Alternativno, možete koristiti *PyCharm* koji predstavlja integrisano razvojno okruženje za programiranje u Python-u. Dostupan je za Windows, macOS i Linux operativne sisteme. *PyCharm* alat je dostupan na zvaničnom sajtu.

Numpy

Biblioteka za 'scientific computing'. Veoma je slična matlabu, tako da ako ste familijarni sa MATLAB-om, korisno je pogledati tutorijal.

```
#dodavanje numpy biblioteke
import numpy as np
```

Vektori i matrice

Numpy nudi mogućnost za rad sa vekorima i matricama.

```
print (np.array([1,2,3,4,5,6]))
print (np.array([1,2,3,4,5,6],'d'))
print (np.array([1,2,3,4,5,6],'D'))
```

```
[1 2 3 4 5 6]
[1. 2. 3. 4. 5. 6.]
[1.+0.j 2.+0.j 3.+0.j 4.+0.j 5.+0.j 6.+0.j]
```

```
a = np.array([1, 2, 3])  # Create a rank 1 array
print (type(a), a.shape, a[0], a[1], a[2])
a[0] = 5                 # Change an element of the array
print (a)
```

```
<class 'numpy.ndarray'> (3,) 1 2 3
[5 2 3]
```

```
#kreiranje matrice kao dva vektora

b = np.array([[0,1],[1,0]],'d')
print (b.shape)
#kreiranje nula matrice
print ('\n nula matrica \n')
print (np.zeros((3,3),'d'))
```

```
(2, 2)

nula matrica

[[0. 0. 0.]
 [0. 0. 0.]
 [0. 0. 0.]]
```

```
#linspace lak nacin za pravljenje koordinata
print (np.linspace(0,1,11))
```

```
[0.  0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1. ]
```

```
#primer mnozenja matrica
```

```
print (np.identity(2)*np.ones((2,2)))

print (np.dot(np.identity(2),np.ones((2,2))))
```

```
[[1. 0.]
 [0. 1.]]
[[1. 1.]
 [1. 1.]]
```

```
#transponovanje
m = np.array([[1,2],[3,4]])
m.T
```

```
array([[1, 3],
       [2, 4]])
```

```
#dijagonala
np.diag([1,2,3,4,5])
```

```
array([[1, 0, 0, 0, 0],
       [0, 2, 0, 0, 0],
       [0, 0, 3, 0, 0],
       [0, 0, 0, 4, 0],
       [0, 0, 0, 0, 5]])
```

kreiranje nizova:

```
a = np.zeros((2,2))  # Create an array of all zeros
print (a)
```

```
[[0. 0.]
 [0. 0.]]
```

```
b = np.ones((1,2))    # Create an array of all ones
print (b)
```

```
[[1. 1.]]
```

```
c = np.full((2,2), 7) # Create a constant array
print (c)
```

```
[[7 7]
 [7 7]]
```

```
d = np.eye(2)         # Create a 2x2 identity matrix
print (d)
```

```
[[1. 0.]
 [0. 1.]]
```

```
e = np.random.random((2,2)) # Create an array filled with random values
```

```
print (e)
```

```
[[0.67219631 0.59134672]
 [0.44267091 0.75269222]]
```

Indeksiranje nizova

Postoji nekoliko načina za indeksiranje nizova. Takođe, numpy podržava i slicing, uz jedan dodatak da nizovi mogu biti višedimenzionalani pa se mora zadati slice za svaku dimenziju.

```
# Create the following rank 2 array with shape (3, 4)
# [[ 1  2  3  4]
#  [ 5  6  7  8]
#  [ 9 10 11 12]]
a = np.array([[1,2,3,4], [5,6,7,8], [9,10,11,12]])

# Use slicing to pull out the subarray consisting of the first 2 rows
# and columns 1 and 2; b is the following array of shape (2, 2):
# [[2 3]
#  [6 7]]
b = a[:2, 1:3]
print (b)
```

```
[[2 3]
 [6 7]]
```

Slice nekog niza predstavlja pogled na iste podatke, tako da će se svaka modifikacija odraziti na originalni niz.

```
print (a[0, 1])
b[0, 0] = 77    # b[0, 0] je isti podatak kao a[0, 1]
print (a[0, 1])
```

```
2
77
```

Mix integer indeksiranja i slice indeksiranja.

Napomena Nije isto kao u MATLAB-u

```
# Create the following rank 2 array with shape (3, 4)
a = np.array([[1,2,3,4], [5,6,7,8], [9,10,11,12]])
print (a)
```

```
[[ 1  2  3  4]
 [ 5  6  7  8]
 [ 9 10 11 12]]
```

Selekcija podataka u srednjem redu. Obratiti pažnju na rang niza koji je predstavljen kao rezultat.

```
row_r1 = a[1, :]    # Rank 1 view of the second row of a
```

```
row_r2 = a[1:2, :]  # Rank 2 view of the second row of a
row_r3 = a[[1], :]  # Rank 2 view of the second row of a
print (row_r1, row_r1.shape)
print (row_r2, row_r2.shape)
print (row_r3, row_r3.shape)
```

```
[5 6 7 8] (4,)
[[5 6 7 8]] (1, 4)
[[5 6 7 8]] (1, 4)
```

Isto važi i za kolone

```
col_r1 = a[:, 1]
col_r2 = a[:, 1:2]
print (col_r1, col_r1.shape)
print ()
print (col_r2, col_r2.shape)
```

```
[ 2  6 10] (3,)


[[ 2]
 [ 6]
 [10]] (3, 1)
```

Integer indeksiranje

```
a = np.array([[1,2], [3, 4], [5, 6]])
print (a)

# An example of integer array indexing.
# The returned array will have shape (3,) and
print (a[[0, 1, 2], [0, 1, 0]])

# The above example of integer array indexing is equivalent to this:
print (np.array([a[0, 0], a[1, 1], a[2, 0]]))
```

```
[[1 2]
 [3 4]
 [5 6]]
[1 4 5]
[1 4 5]
```

```
# When using integer array indexing, you can reuse the same
# element from the source array:
print (a[[0, 1], [1, 1]])

# Equivalent to the previous integer array indexing example
print (np.array([a[0, 1], a[1, 1]]))
```

```
[2 4]
[2 4]
```

Trik za selekciju ili izmenu jednog elementa iz svakog reda

```
# Create a new array from which we will select elements
a = np.array([[1,2,3], [4,5,6], [7,8,9], [10, 11, 12]])
print (a)
```

```
[[ 1  2  3]
 [ 4  5  6]
 [ 7  8  9]
 [10 11 12]]
```

```
# Create an array of indices
b = np.array([0, 2, 0, 1])

# Select one element from each row of a using the indices in b
print (a[np.arange(4), b])  # Prints "[ 1  6  7 11]"
```

```
[ 1 6 7 11]
```

```
# Mutate one element from each row of a using the indices in b
a[np.arange(4), b] += 10
print (a)
```

```
[[11  2  3]
 [ 4  5 16]
 [17  8  9]
 [10 21 12]]
```

Boolean indeksiranje koristi se za selekciju elementata koji zadovoljavaju neki uslov

```
a = np.array([[1,2], [3, 4], [5, 6]])
print (a)
print ()
bool_idx = (a > 2)  # Find the elements of a that are bigger than 2;
                    # this returns a numpy array of Booleans of the same
                    # shape as a, where each slot of bool_idx tells
                    # whether that element of a is > 2.

print (bool_idx)
```

```
[[1 2]
 [3 4]
 [5 6]]
[[False False]
 [ True  True]
 [ True  True]]
```

```
# We use boolean array indexing to construct a rank 1 array
# consisting of the elements of a corresponding to the True values
# of bool_idx
print (a[bool_idx])
```

```
# We can do all of the above in a single concise statement:
print (a[a > 2])
```

```
[3 4 5 6]
[3 4 5 6]
```

Tipovi podataka

Svaki numpy niz je mreža elemenata istog tipa. Numpy nudi veliki skup numeričkih tipova podataka koji se mogu koristiti za pravljenje nizova. Ako se ne navede tip podataka, numpy pokušava da pogodi o kom tipu se radi. Obično se prilikom pravljenja niza navede i koji je tip podataka.

```
x = np.array([1, 2])  # Let numpy choose the datatype
y = np.array([1.0, 2.0])  # Let numpy choose the datatype
z = np.array([1, 2], dtype=np.int64)  # Force a particular datatype

print (x.dtype, y.dtype, z.dtype)
```

```
int64 float64 int64
```

Više o tipovima podataka u Numpy biblioteci se može pronaći na linku.

Matematičke operacije

Osnovne matematičke operacije se primenjuju na svaki element u nizu. Operacije su dostupne kao preklopljeni operator i kao funkcija.

```
x = np.array([[1,2],[3,4]], dtype=np.float64)
y = np.array([[5,6],[7,8]], dtype=np.float64)

# Elementwise sum; both produce the array
print (x + y)
print (np.add(x, y))
```

```
[[ 6.  8.]
 [10. 12.]]
 [[ 6. 8.]
 [10. 12.]]
```

```
# Elementwise difference; both produce the array
print (x - y)
print (np.subtract(x, y))
```

```
[[-4. -4.]
 [-4. -4.]]
[[-4. -4.]
 [-4. -4.]]
```

```
# Elementwise product; both produce the array
print (x * y)
print (np.multiply(x, y))
```

```
[[ 5. 12.]
 [21. 32.]]
[[ 5. 12.]
 [21. 32.]]
```

```
# Elementwise division; both produce the array
# [[ 0.2         0.33333333]
#  [ 0.42857143  0.5       ]]
print (x / y)
print (np.divide(x, y))
```

```
[[0.2 0.33333333]
 [0.42857143 0.5 ]]
[[0.2 0.33333333]
 [0.42857143 0.5 ]]
```

```
# Elementwise square root; produces the array
# [[ 1.          1.41421356]
#  [ 1.73205081  2.        ]]
print (np.sqrt(x))
```

```
[[1. 1.41421356]
 [1.73205081 2. ]]
```

Za razliku od MATLAB-a * je operator koji množi svaki element, i ne predstavlja matrično množenje. Za matrično množenje, Numpy ima funkciju `dot`, koja je dostupna kao zasebna funkcija i kao funkcija samog niza.

```
x = np.array([[1,2],[3,4]])
y = np.array([[5,6],[7,8]])

v = np.array([9,10])
w = np.array([11, 12])

# Inner product of vectors; both produce 219
print (v.dot(w))
print (np.dot(v, w))
```

```
219
219
```

```
# Matrix / vector product; both produce the rank 1 array [29 67]
```

```
print (x.dot(v))
print (np.dot(x, v))
```

```
[29 67]
[29 67]
```

```
# Matrix / matrix product; both produce the rank 2 array
# [[19 22]
#  [43 50]]
print (x.dot(y))
print (np.dot(x, y))
```

```
[[19 22]
 [43 50]]
[[19 22]
 [43 50]]
```

Pored standardnih matematičkih operacija, Numpy nudi veoma korisne funkcije za izračunavanje nad nizovima.

```
x = np.array([[1,2],[3,4]])

print (np.sum(x))  # Compute sum of all elements; prints "10"
print (np.sum(x, axis=0))  # Compute sum of each column; prints "[4 6]"
print (np.sum(x, axis=1))  # Compute sum of each row; prints "[3 7]"
```

```
10
[4 6]
[3 7]
```

Više o matematičkim operacijama na linku.

Matplotlib

Matplotlib je biblioteka za plotovanje. U ovom delu će biti predstavljen deo ove biblioteke `matplotlib.pyplot`, koji nudi slične mogućnosti kao MATLAB.

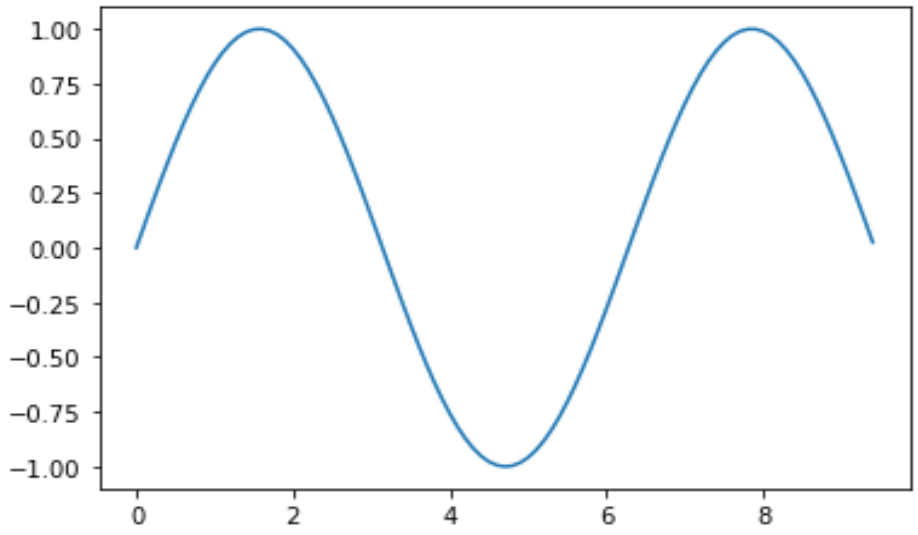
```
import matplotlib.pyplot as plt
```

Plot 2D podataka

```
# Compute the x and y coordinates for points on a sine curve
x = np.arange(0, 3 * np.pi, 0.1)
y_sin = np.sin(x)

# Plot the points using matplotlib
plt.plot(x, y_sin)
```

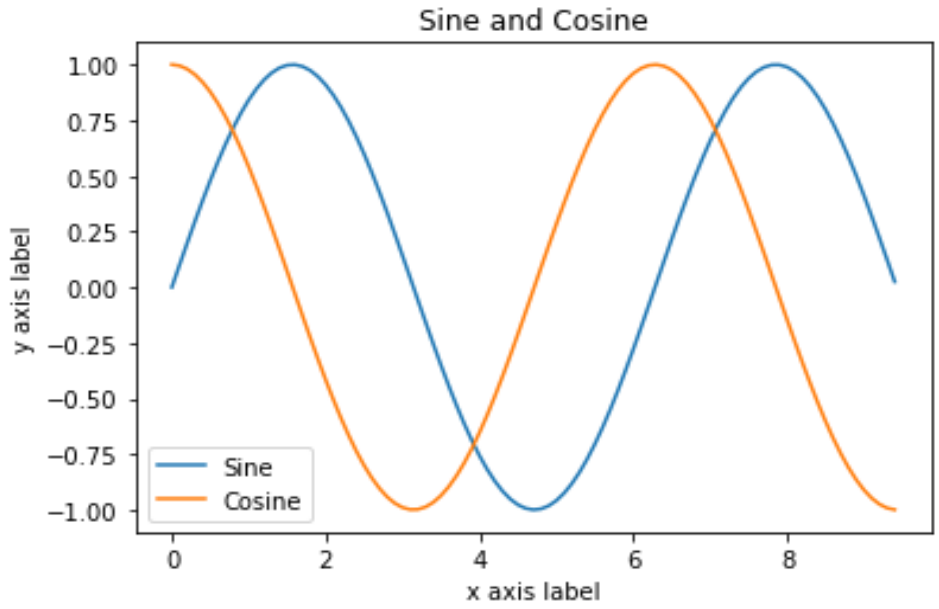
```
[<matplotlib.lines.Line2D at 0x7fa4a5b517b8>]
```




```
y_cos = np.cos(x)

# Plot the points using matplotlib
plt.plot(x, y_sin)
plt.plot(x, y_cos)
plt.xlabel('x axis label')
plt.ylabel('y axis label')
plt.title('Sine and Cosine')
plt.legend(['Sine', 'Cosine'])
```

```
<matplotlib.legend.Legend at 0x7fa4a548f438>
```



Moguće je plotovati raličite stvari unutar iste slike

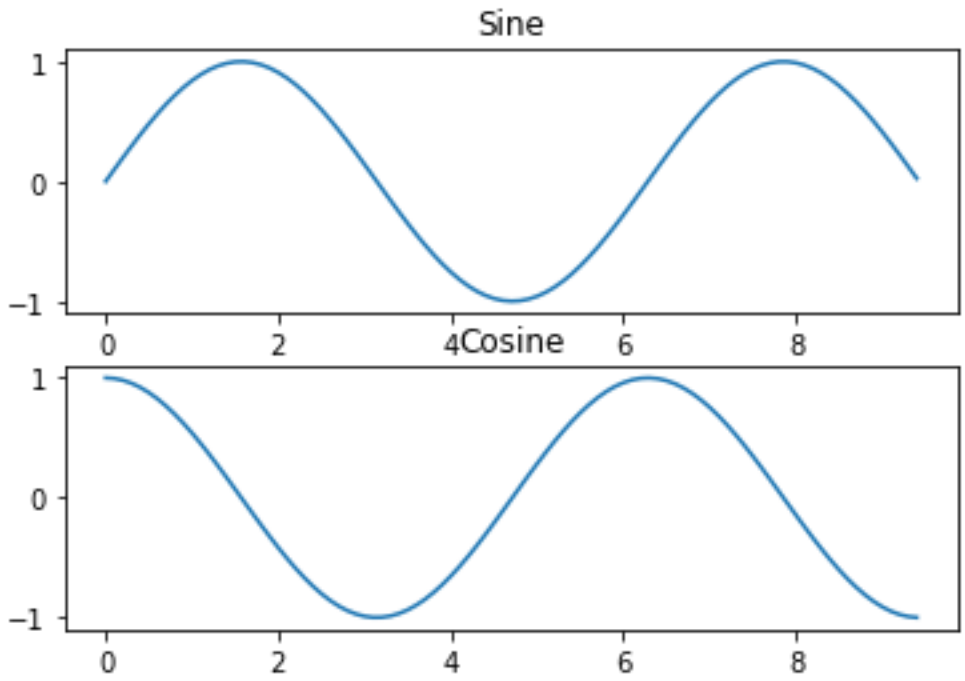
```
# Compute the x and y coordinates for points on sine and cosine curves
x = np.arange(0, 3 * np.pi, 0.1)
y_sin = np.sin(x)
y_cos = np.cos(x)

# Set up a subplot grid that has height 2 and width 1,
# and set the first such subplot as active.
plt.subplot(2, 1, 1)

# Make the first plot
plt.plot(x, y_sin)
plt.title('Sine')

# Set the second subplot as active, and make the second plot.
plt.subplot(2, 1, 2)
plt.plot(x, y_cos)
plt.title('Cosine')

# Show the figure.
plt.show()
```



Više o samoj biblioteci se može pronaći na linku.