Python_N2_20160923_AaronYu

September 23, 2016

0.1 Summary

- 1.Control Flow
  - for
  - if, elif, else
  - range
- 2.List Comprehension
- 3.Numpy Basics - 1
  - Creating ndarrays
  - shape & data type of ndarrays

0.2 Control Flow

- for loop

```
In [63]: countries = ['US', 'CN', 'UK','JP']
         for country in countries:
             print(country)
```

```
US
CN
UK
JP
```

- if, elif, else

```
In [81]: score = 75
         if score > 90:
             print('A')
         elif score > 80:
             print('B')
         elif score > 60:
             print('C')
         else:
             print('D')
```

C

- for and if combined

```
In [3]: num = [2,3,7,8,1]
        for i in num:
            if i % 2 == 0:
                print('Even')
            else:
                print('Odd')
```

```
Even
Odd
Odd
Even
Odd
```

- range

```
In [4]: ## Return an object that produces a sequence of integers from start to stop
```

```
In [58]: list(range(10))
```

```
Out[58]: [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
```

```
In [57]: list(range(2,10))
```

```
Out[57]: [2, 3, 4, 5, 6, 7, 8, 9]
```

```
In [7]: list(range(2,10,2))
```

```
Out[7]: [2, 4, 6, 8]
```

```
In [8]: ## Combine range and for loop allows you to display both index and item
```

```
In [9]: countries
```

```
Out[9]: ['US', 'CN', 'UK', 'JP']
```

```
In [10]: for i in range(len(countries)):
             print(countries[i])
```

```
US
CN
UK
JP
```

```
In [11]: for i in range(len(countries)):
             print('The %dth item is: ' % i, countries[i])
```

```
The 0th item is:  US
The 1th item is:  CN
The 2th item is:  UK
The 3th item is:  JP
```

0.3 List Comprehension

- List Comprehension (LC) is concise, fast and elegant

```
In [61]: from IPython.display import Image
         Image(filename='/Users/Aaron/Desktop/Academic/BIA_Python_Sharing/ListCompr
```

```
Out[61]:
```

In a general sense, a FOR loop works as:

```
for (set of values to iterate):
  if (conditional filtering):
    output_expression()
```

```
The same gets implemented in a simple LC construct in a single line as:
 [ output_expression() for(set of values to iterate) if(conditional filtering) ]
```

- Example 1

```
In [66]: coun1 = []
         for c in countries:
             c = c.lower()
             coun1.append(c)
         coun1
```

```
Out[66]: ['us', 'cn', 'uk', 'jp']
```

```
In [68]: coun2 = [c.lower() for c in countries]
         coun2
```

```
Out[68]: ['us', 'cn', 'uk', 'jp']
```

```
In [76]: import pandas as pd
```

```
In [79]: countries_ser = pd.Series(countries)
```

```
In [80]: countries_ser.map(str.lower)
```

```
Out[80]: 0    us
         1    cn
         2    uk
         3    jp
         dtype: object
```

- Example 2

```
In [82]: score_lc = [79,92,65,80,99]
         A = [score for score in score_lc if score >= 90]
         A

Out[82]: [92, 99]

In [83]: for score in score_lc:
             if score >= 90:
                 print(score)
             else:
                 pass

92
99
```

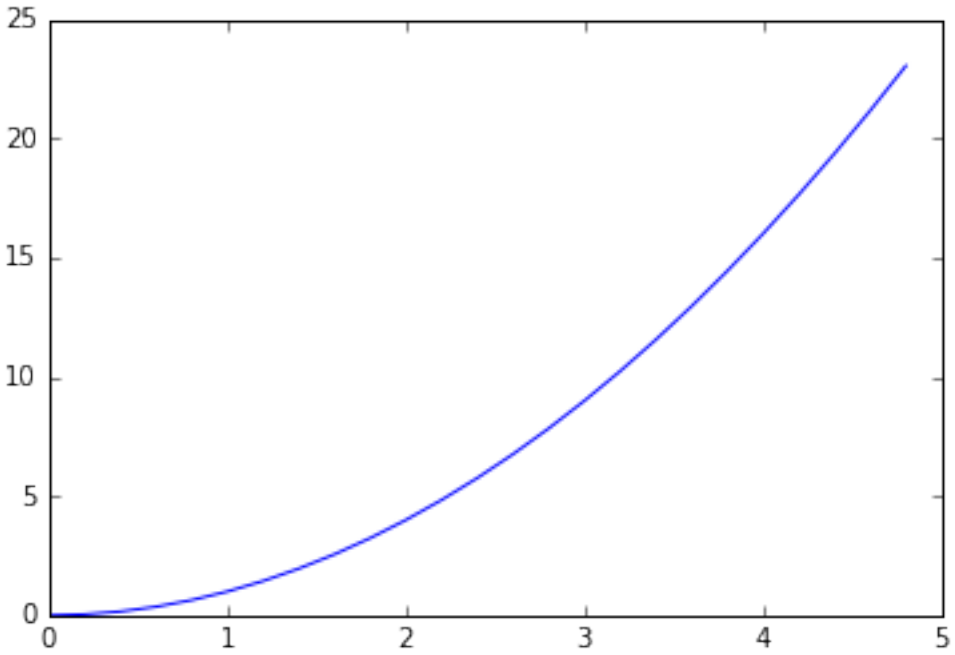
0.4 Numpy Basics - 1

```
In [1]: import numpy as np
```

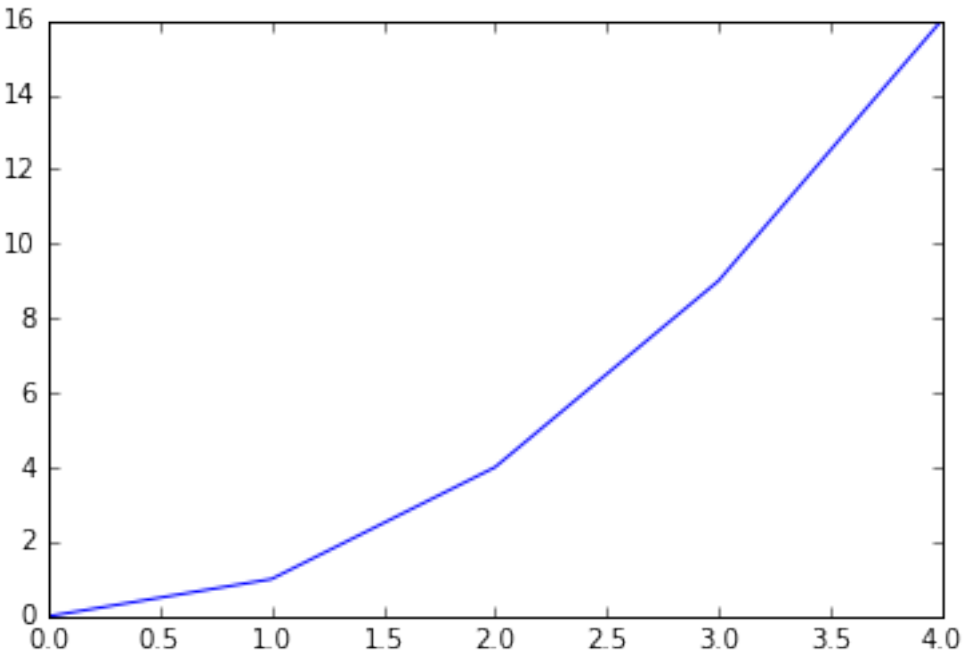
- Creating ndarrays

```
1.np.array()

In [23]: arr1 = np.array([1,2,3])
         arr1

Out[23]: array([1, 2, 3])

In [24]: arr2 = np.array([[1,2,3],[4,5,6]])
         arr2

Out[24]: array([[1, 2, 3],
                [4, 5, 6]])

In [25]: arr3 = np.array(range(10))
         arr3

Out[25]: array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9])

2.np.arange()

In [26]: ## np.arange() returns evenly spaced values within a given interval.
         ## For integer arguments the function is equivalent to the range function
         ## But returns an ndarray rather than a list

In [27]: arr4 = np.arange(10)
         arr4
```

```
Out[27]: array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9])

In [28]: arr5 = np.arange(1,3,0.1)
         arr5

Out[28]: array([ 1. ,  1.1,  1.2,  1.3,  1.4,  1.5,  1.6,  1.7,  1.8,  1.9,  2. ,
                 2.1,  2.2,  2.3,  2.4,  2.5,  2.6,  2.7,  2.8,  2.9])

In [29]: import matplotlib.pyplot as plt
         %matplotlib inline

In [30]: x = np.arange(0,5,0.2)
         y = [ele**2 for ele in x]
         plt.plot(x,y)

Out[30]: [<matplotlib.lines.Line2D at 0x10fc8ac18>]
```




```
In [31]: x_2 = range(5)
         y_2 = [ele**2 for ele in x_2]
         plt.plot(x_2, y_2)

Out[31]: [<matplotlib.lines.Line2D at 0x10fd7ef60>]
```



3.np.ones(), np.zeros()

```
In [8]: np.ones(3)

Out[8]: array([ 1.,  1.,  1.])

In [10]: np.zeros((3,2))

Out[10]: array([[ 0.,  0.],
                [ 0.,  0.],
                [ 0.,  0.]])
```

4.np.random

```
In [19]: np.random.rand(3,2)

Out[19]: array([[ 0.10923271,  0.06571421],
                [ 0.37396238,  0.51140212],
                [ 0.54983074,  0.02170531]])

In [14]: np.random.randn(3,2)

Out[14]: array([[-0.03914872,  0.04524103],
                [-0.70360568, -1.53674235],
                [ 2.67906455,  0.94277808]])

In [17]: np.random.randint(5,size = (2,2))
```

```
Out[17]: array([[4, 0],
                [4, 3]])
```

- shape & data type of ndarrays

```
In [32]: arr2

Out[32]: array([[1, 2, 3],
                [4, 5, 6]])

In [33]: arr2.shape

Out[33]: (2, 3)

In [34]: arr2.shape[0]

Out[34]: 2

In [38]: arr1

Out[38]: array([1, 2, 3])

In [39]: arr1.shape

Out[39]: (3,)

In [40]: arr1.dtype

Out[40]: dtype('int64')

In [43]: arr_f = np.arange(3,0.5)
         arr_f.dtype

Out[43]: dtype('float64')

In [ ]: ## Use astype() to convert

In [44]: arr1_f = arr1.astype('float64')
         arr1_f

Out[44]: array([ 1.,  2.,  3.])

In [55]: arr_str = np.array([1.2,2.0,3.9], dtype = np.int64)
         arr_str

Out[55]: array([1, 2, 3])
```