Programming for Data Science (with Python)

Le Trong Ngoc – http://fit.iuh.edu.vn/giangvien@letrongngoc

Contents

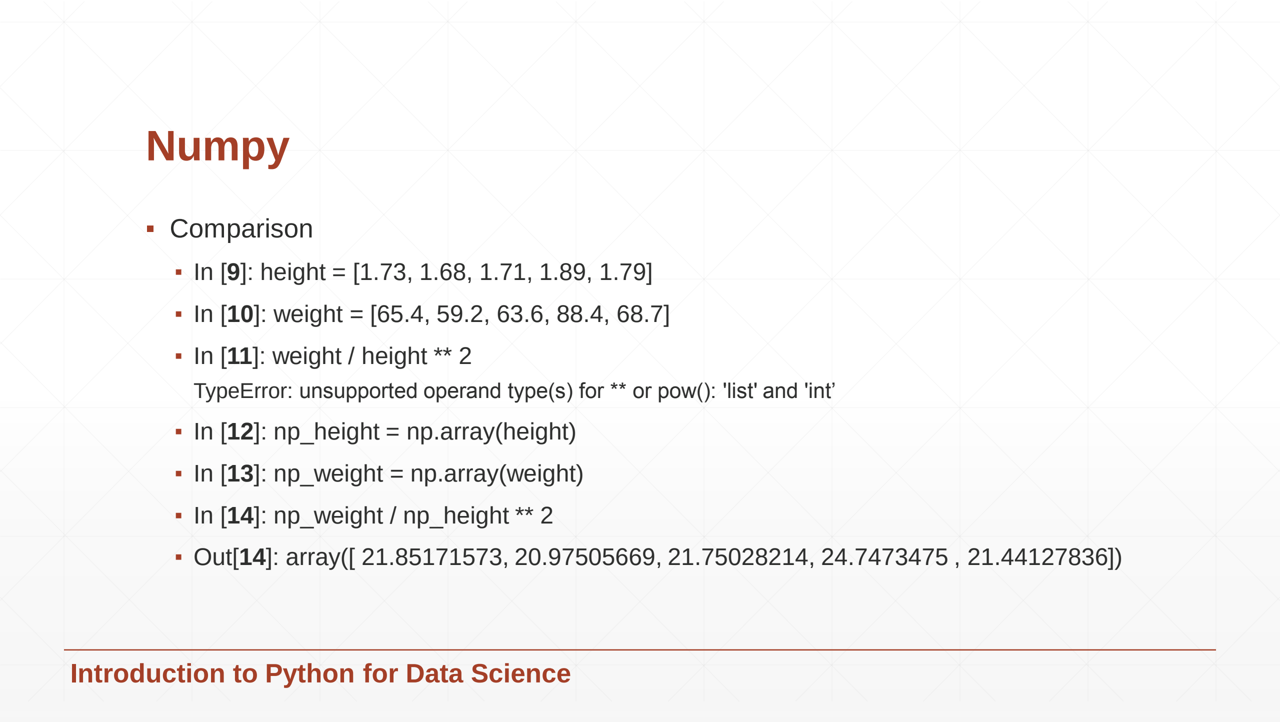
- Introduction to Data Science and Data Analysis
- **Introduction to Python for Data Science**
- Data Visualization with Python
- Statistical Thinking in Python
- Applied Machine Learning in Python

Introduction to Python for Data Science

- Set up the Lab Environment
- Python basics
- List – A Data Structure
- Functions and Packages
- **Numpy**
- **Plotting with Matplotlib**
- Control Flow and Pandas

Numpy

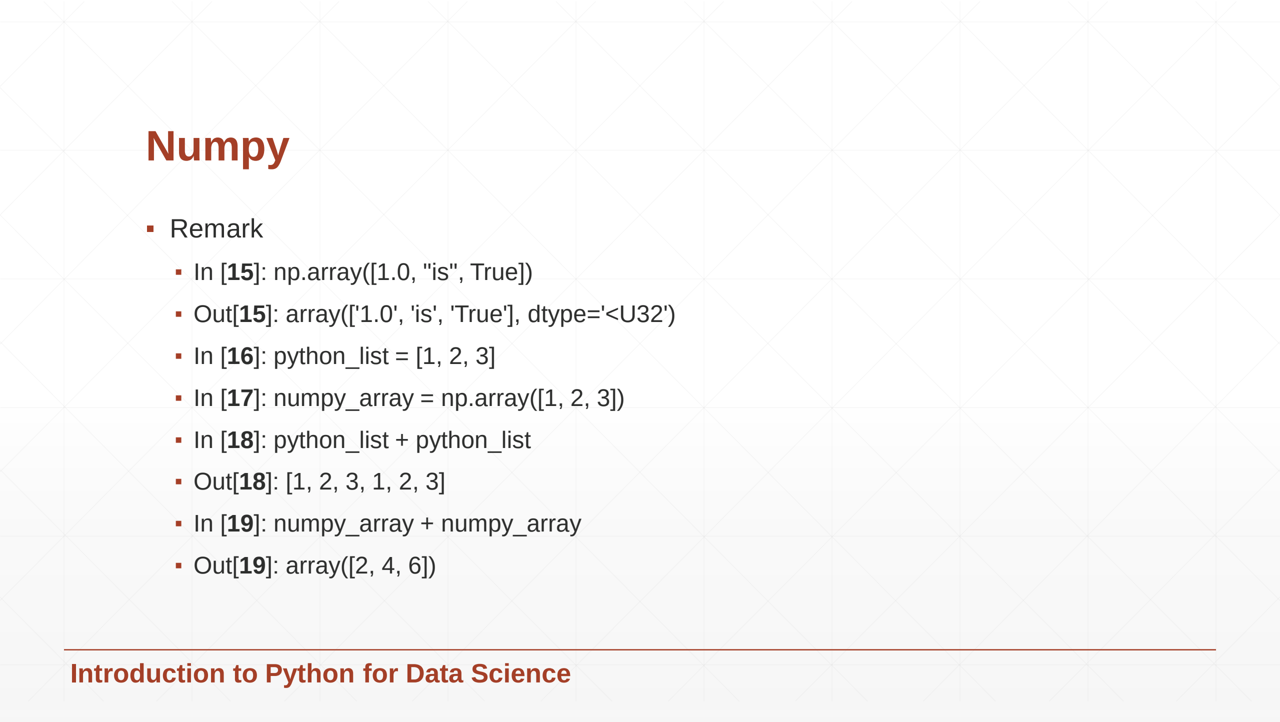
- Numeric Python
- Alternative to Python List: Numpy Array
- Calculations over entire arrays
- Easy and Fast
- Installation
  - In the terminal: pip3 install numpy



Numpy

- Comparison
  - In [**9**]: height = [1.73, 1.68, 1.71, 1.89, 1.79]
  - In [**10**]: weight = [65.4, 59.2, 63.6, 88.4, 68.7]
  - In [**11**]: weight / height ** 2
    TypeError: unsupported operand type(s) for ** or pow(): 'list' and 'int’
  - In [**12**]: np_height = np.array(height)
  - In [**13**]: np_weight = np.array(weight)
  - In [**14**]: np_weight / np_height ** 2
  - Out[**14**]: array([21.85171573, 20.97505669, 21.75028214, 24.7473475 , 21.44127836])

Introduction to Python for Data Science



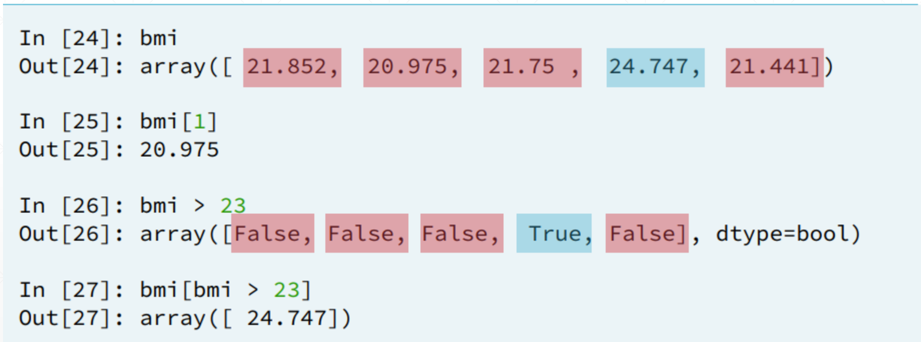
Numpy

- Remark
  - In [**15**]: np.array([1.0, "is", True])
  - Out[**15**]: array(['1.0', 'is', 'True'], dtype='<U32')
  - In [**16**]: python_list = [1, 2, 3]
  - In [**17**]: numpy_array = np.array([1, 2, 3])
  - In [**18**]: python_list + python_list
  - Out[**18**]: [1, 2, 3, 1, 2, 3]
  - In [**19**]: numpy_array + numpy_array
  - Out[**19**]: array([2, 4, 6])

Introduction to Python for Data Science

Numpy

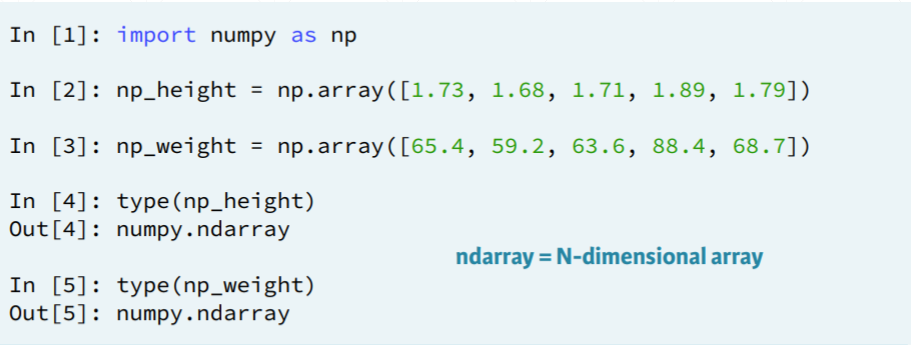
- Subsetting



```
In [24]: bmi
Out[24]: array([ 21.852,  20.975,  21.75 ,  24.747,  21.441])

In [25]: bmi[1]
Out[25]: 20.975

In [26]: bmi > 23
Out[26]: array([False, False, False,  True, False], dtype=bool)

In [27]: bmi[bmi > 23]
Out[27]: array([ 24.747])
```

Numpy

- Type of Numpy Arrays

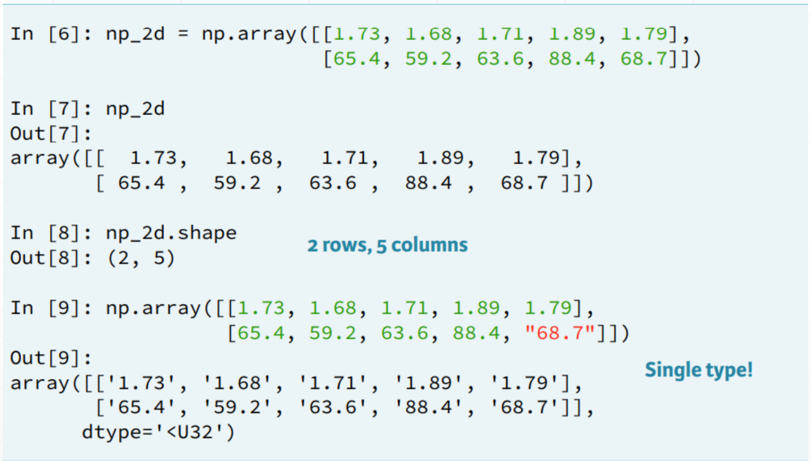


```
In [1]: import numpy as np

In [2]: np_height = np.array([1.73, 1.68, 1.71, 1.89, 1.79])

In [3]: np_weight = np.array([65.4, 59.2, 63.6, 88.4, 68.7])

In [4]: type(np_height)
Out[4]: numpy.ndarray

In [5]: type(np_weight)
Out[5]: numpy.ndarray
```

ndarray = N-dimensional array

Numpy

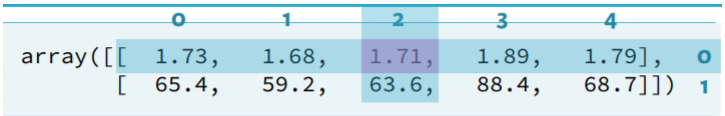
- 2D Numpy Array



```
In [6]: np_2d = np.array([[1.73, 1.68, 1.71, 1.89, 1.79],
                          [65.4, 59.2, 63.6, 88.4, 68.7]])

In [7]: np_2d
Out[7]:
array([[  1.73,   1.68,   1.71,   1.89,   1.79],
       [ 65.4 ,  59.2 ,  63.6 ,  88.4 ,  68.7 ]])

In [8]: np_2d.shape
Out[8]: (2, 5)

In [9]: np.array([[1.73, 1.68, 1.71, 1.89, 1.79],
                  [65.4, 59.2, 63.6, 88.4, "68.7"]])
Out[9]:
array([['1.73', '1.68', '1.71', '1.89', '1.79'],
       ['65.4', '59.2', '63.6', '88.4', '68.7']],
      dtype='<U32')
```

2 rows, 5 columns

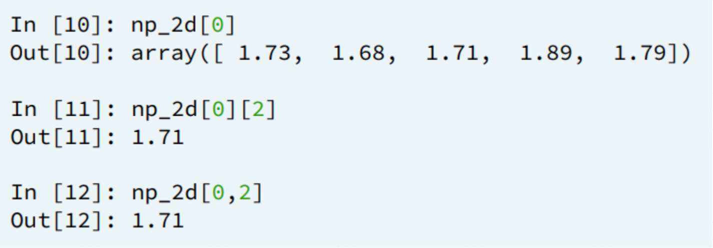
Single type!



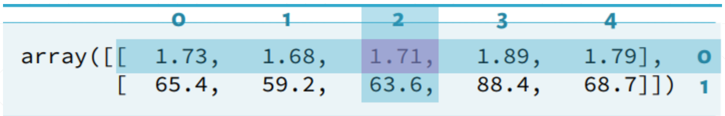
```
array([[ 1.73,  1.68,  1.71,  1.89,  1.79],
       [ 65.4,  59.2,  63.6,  88.4,  68.7]])
```

Numpy

- Subsetting



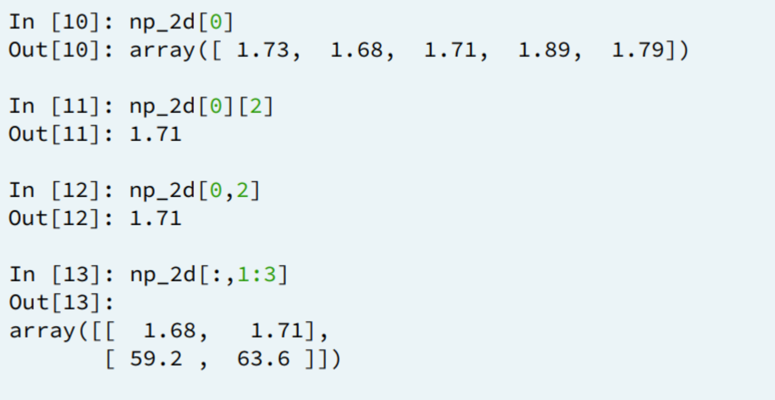
```
In [10]: np_2d[0]
Out[10]: array([ 1.73,  1.68,  1.71,  1.89,  1.79])

In [11]: np_2d[0][2]
Out[11]: 1.71

In [12]: np_2d[0,2]
Out[12]: 1.71
```



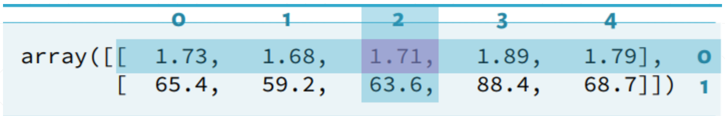
```
array([[ 1.73,  1.68,  1.71,  1.89,  1.79],
       [ 65.4,  59.2,  63.6,  88.4,  68.7]])
```

Numpy

- Subsetting



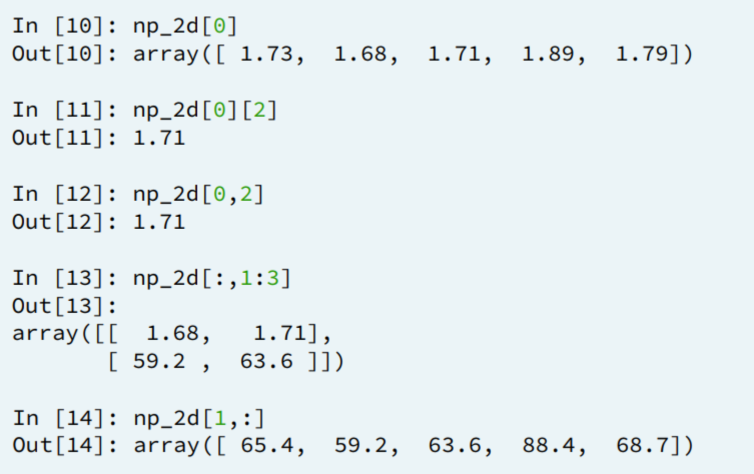
```
In [10]: np_2d[0]
Out[10]: array([ 1.73,  1.68,  1.71,  1.89,  1.79])

In [11]: np_2d[0][2]
Out[11]: 1.71

In [12]: np_2d[0,2]
Out[12]: 1.71

In [13]: np_2d[:,1:3]
Out[13]:
array([[  1.68,   1.71],
       [ 59.2 ,  63.6 ]])
```



```
array([[ 1.73,  1.68,  1.71,  1.89,  1.79],
       [ 65.4,  59.2,  63.6,  88.4,  68.7]])
```

Numpy

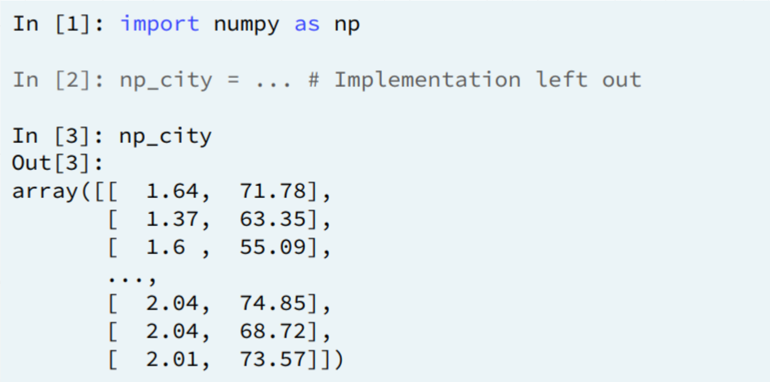
- Subsetting



```
In [10]: np_2d[0]
Out[10]: array([ 1.73,  1.68,  1.71,  1.89,  1.79])

In [11]: np_2d[0][2]
Out[11]: 1.71

In [12]: np_2d[0,2]
Out[12]: 1.71

In [13]: np_2d[:,1:3]
Out[13]:
array([[  1.68,   1.71],
       [ 59.2 ,  63.6 ]])

In [14]: np_2d[1,:]
Out[14]: array([ 65.4,  59.2,  63.6,  88.4,  68.7])
```

Numpy

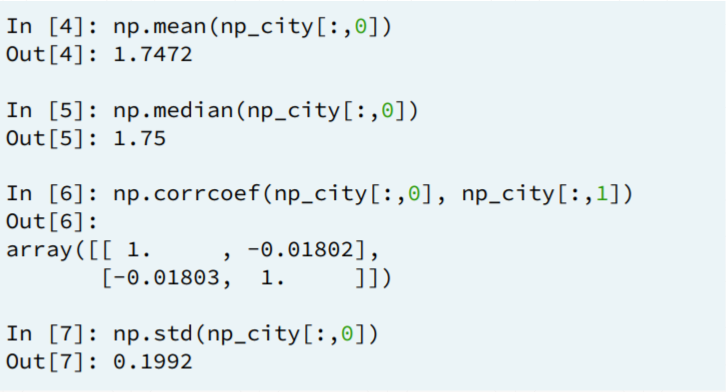
- Basic Statistics



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

In [2]: np_city = ... # Implementation left out

In [3]: np_city
Out[3]:
array([[  1.64,   71.78],
       [  1.37,   63.35],
       [  1.6 ,   55.09],
       ...,
       [  2.04,   74.85],
       [  2.04,   68.72],
       [  2.01,   73.57]])
```

Numpy

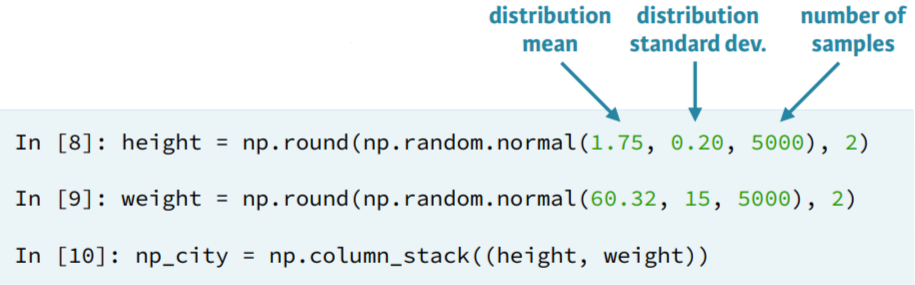
- Basic Statistics



```
In [4]: np.mean(np_city[:,0])
Out[4]: 1.7472

In [5]: np.median(np_city[:,0])
Out[5]: 1.75

In [6]: np.corrcoef(np_city[:,0], np_city[:,1])
Out[6]:
array([[ 1.      , -0.01802],
       [-0.01803,  1.      ]])

In [7]: np.std(np_city[:,0])
Out[7]: 0.1992
```

Numpy

- Basic Statistics



distribution mean

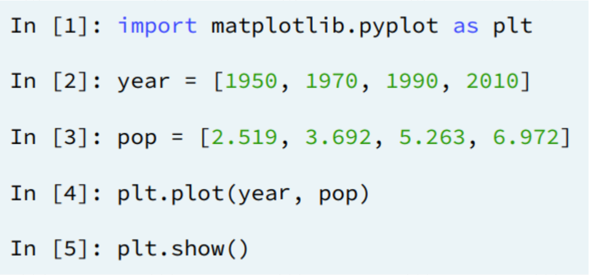
distribution standard dev.

number of samples

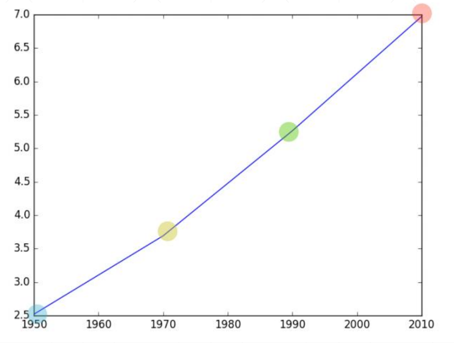
```
In [8]: height = np.round(np.random.normal(1.75, 0.20, 5000), 2)

In [9]: weight = np.round(np.random.normal(60.32, 15, 5000), 2)

In [10]: np_city = np.column_stack((height, weight))
```

Plotting with Matplotlib

- Matplotlib

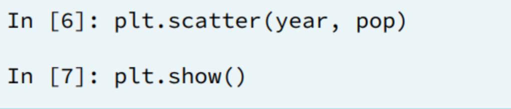


```
In [1]: import matplotlib.pyplot as plt

In [2]: year = [1950, 1970, 1990, 2010]

In [3]: pop = [2.519, 3.692, 5.263, 6.972]

In [4]: plt.plot(year, pop)

In [5]: plt.show()
```

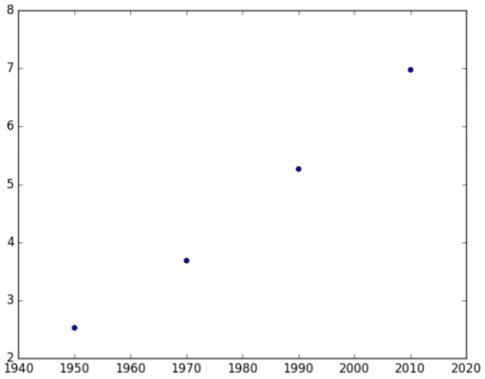


Plotting with Matplotlib

- Scatter plot



```
In [6]: plt.scatter(year, pop)

In [7]: plt.show()
```

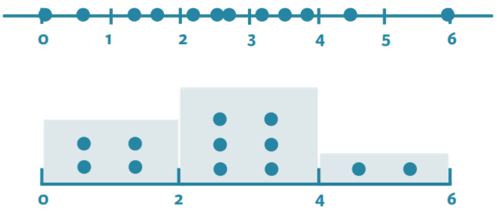


Plotting with Matplotlib

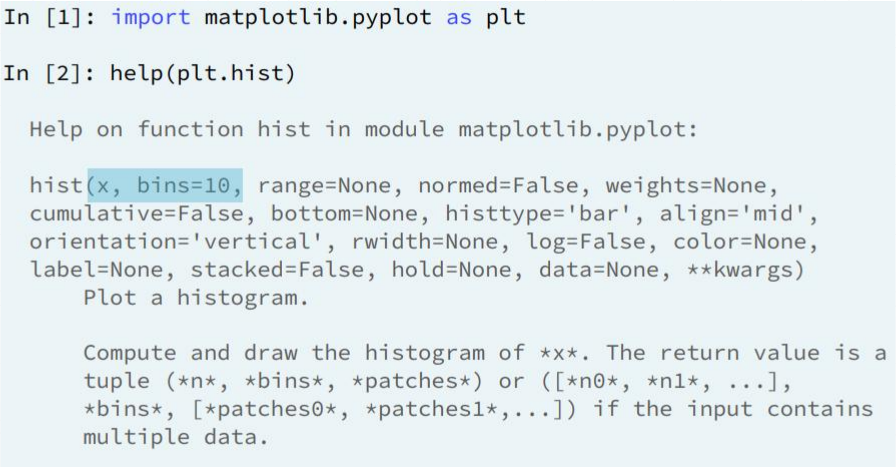
- Histogram

Explore dataset

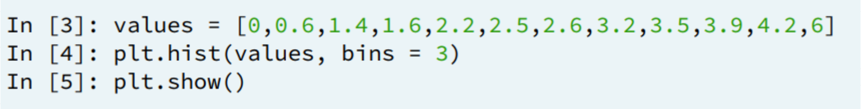
Get idea about distribution



Plotting with Matplotlib

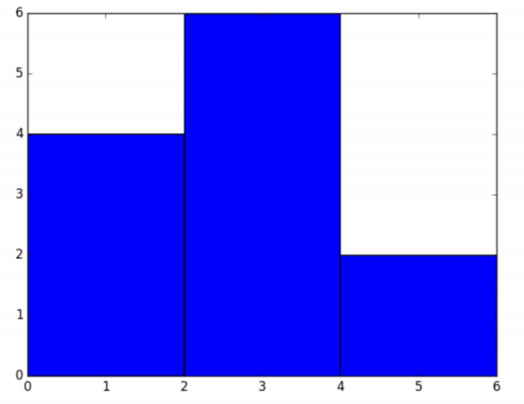


```
In [1]: import matplotlib.pyplot as plt

In [2]: help(plt.hist)

  Help on function hist in module matplotlib.pyplot:

  hist(x, bins=10, range=None, normed=False, weights=None,
  cumulative=False, bottom=None, histtype='bar', align='mid',
  orientation='vertical', rwidth=None, log=False, color=None,
  label=None, stacked=False, hold=None, data=None, **kwargs)
      Plot a histogram.

      Compute and draw the histogram of *x*. The return value is a
      tuple (*n*, *bins*, *patches*) or ([*n0*, *n1*, ...],
      *bins*, [*patches0*, *patches1*,...]) if the input contains
      multiple data.
```



```
In [3]: values = [0,0.6,1.4,1.6,2.2,2.5,2.6,3.2,3.5,3.9,4.2,6]
In [4]: plt.hist(values, bins = 3)
In [5]: plt.show()
```

Plotting with Matplotlib


6
5
4
3
2
1
0
0
1
2
3
4
5
6