# CSC 3141

IMAGE PROCESSING LABORATORY

## 02 – NumPy and Matplotlib

# NumPy Library

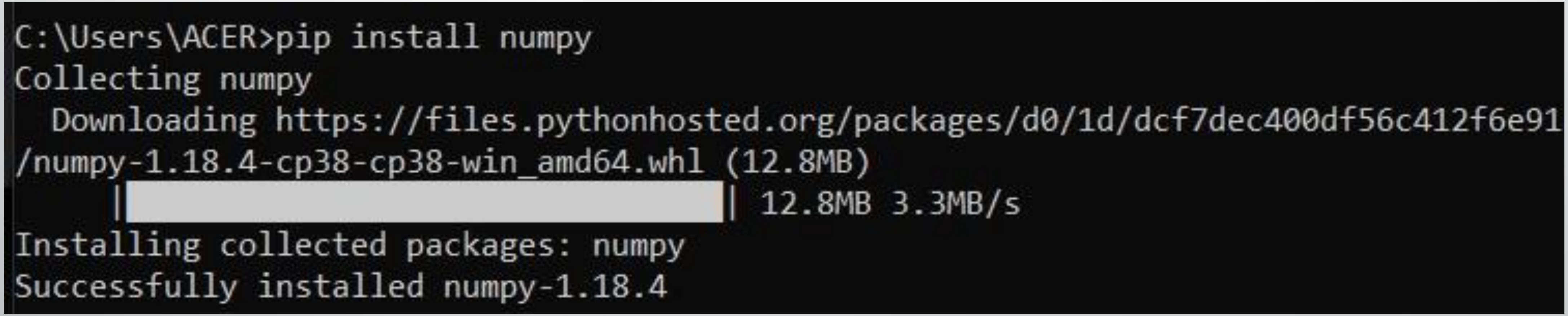
# NumPy

- **NumPy** is a Python package which stands for 'Numerical Python'.
- Fundamental package for scientific computing with Python
- Memory efficiency
- Alternative to Python List: NumPy Array
- Easy and Fast

# NumPy Installation

- **Installation**

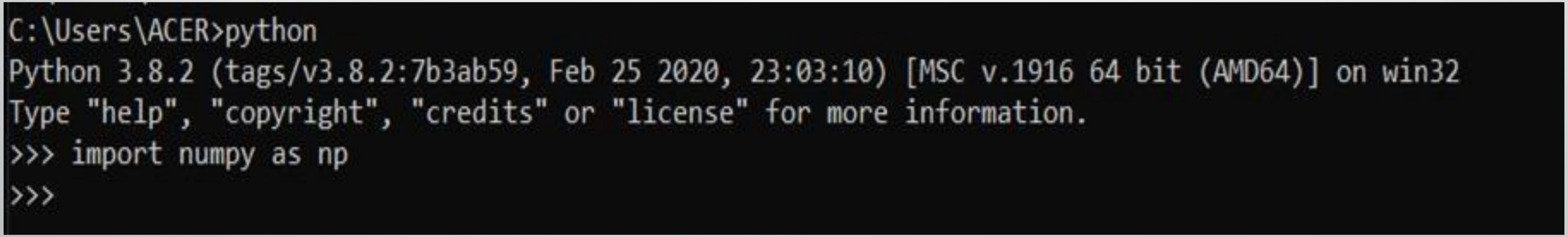
  In the terminal use the  pip command to install NumPy package.



```
C:\Users\ACER>pip install numpy
Collecting numpy
  Downloading https://files.pythonhosted.org/packages/d0/1d/dcf7dec400df56c412f6e91
/numpy-1.18.4-cp38-cp38-win_amd64.whl (12.8MB)
     |████████████████████████████████| 12.8MB 3.3MB/s
Installing collected packages: numpy
Successfully installed numpy-1.18.4
```

# NumPy contd.

- Once the package is installed successfully, type python to get into python prompt.
- Use the import command to include NumPy package and use it. You can also set an alias name (short name) for package.

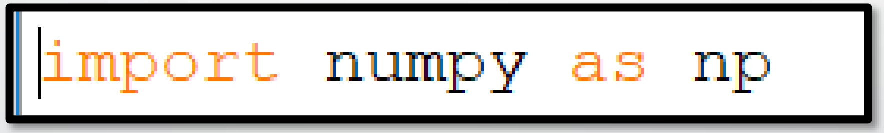


```
C:\Users\ACER>python
Python 3.8.2 (tags/v3.8.2:7b3ab59, Feb 25 2020, 23:03:10) [MSC v.1916 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> import numpy as np
>>>
```

# 1D NumPy Array Creation

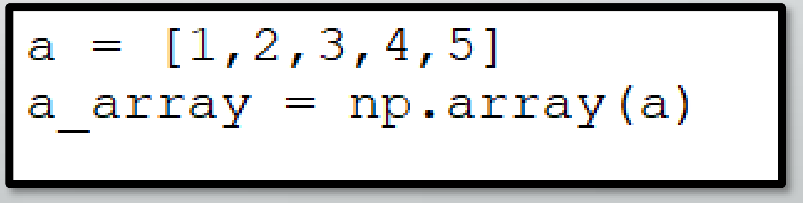
- All elements must be of the same type, preferably **int, float** or **complex**.
- The number of elements must be known when the array is created.

- Importing NumPy



```
import numpy as np
```

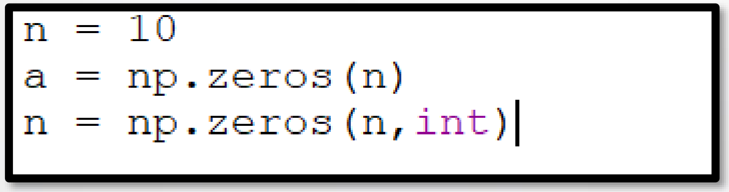
- Create a list and convert it as numpy group



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

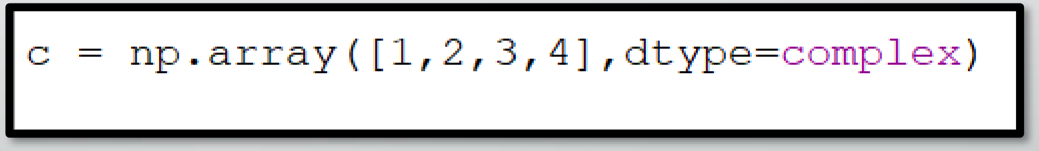
# 1D NumPy Array Creation contd.

- Creating a new array, filled with zeros.



```
n = 10
a = np.zeros(n)
n = np.zeros(n,int)
```

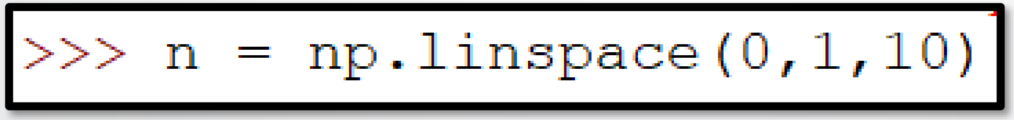
- Type of the array can be explicitly specified at creation time



```
c = np.array([1,2,3,4],dtype=complex)
```

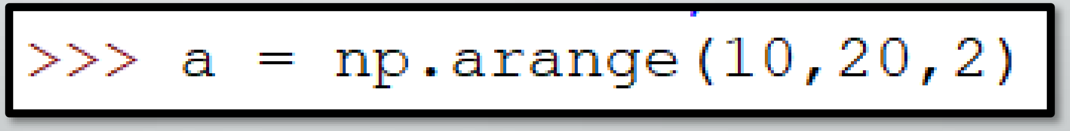
# 1D NumPy Array Creation contd.

- **`linspace(p,q,n)`** – Creating an array of ‘n’ elements with uniformly distributed values in an interval [p,q]



```
>>> n = np.linspace(0,1,10)
```

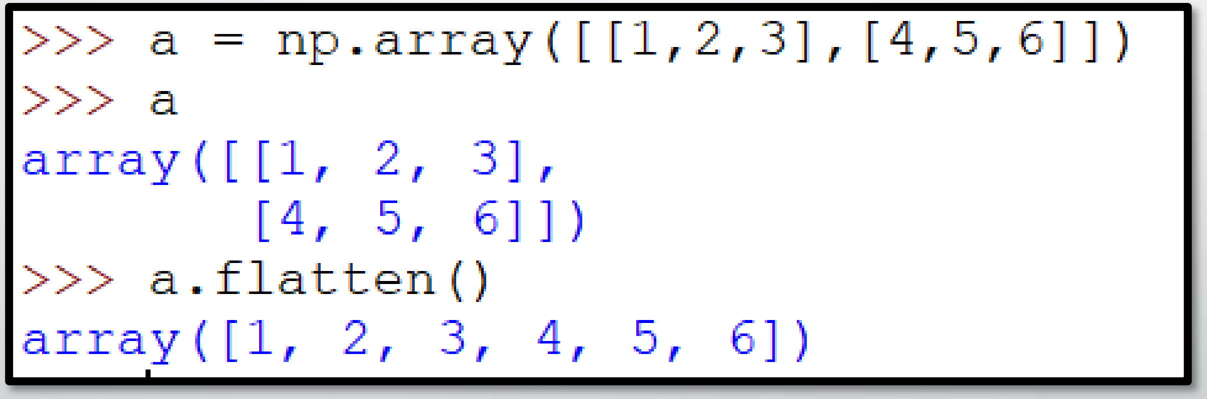
- **`arange(start,stop,step)`** – Creating an array of specified range .
  step : Spacing between values. Default step is 1



```
>>> a = np.arange(10,20,2)
```

# 1D NumPy Array Creation contd.

- To change the shape of a NumPy array from a multi-dimensional array, to a 1-dimensional array,



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

# 1D NumPy Array Creation contd.

- To generate a sequence of Random Numbers

```
np.random.normal(mean,sd,size)
```

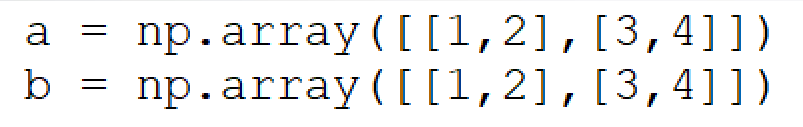
***mean*** – The center of distribution
***sd*** – standard deviation
***size*** – number of returns

# NumPy Arrays – Statistical Functions

- Available measures of dispersion in NumPy arrays

| | | |
|---|---|---|
| ***Min*** | - | `np.min()` |
| ***Max*** | - | `np.max()` |
| ***Mean*** | - | `np.mead()` |
| ***Median*** | - | `np.median()` |
| ***Standard Deviation*** | - | `np.stdt()` |

# Product of two NumPy arrays



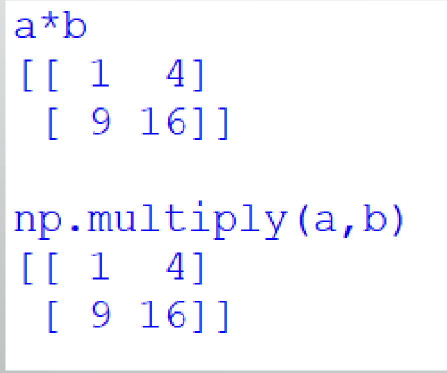
```
a = np.array([[1,2],[3,4]])
b = np.array([[1,2],[3,4]])
```

- **Element-wise multiplication**

  a*b

  ***or***

  np.multiply(a,b)



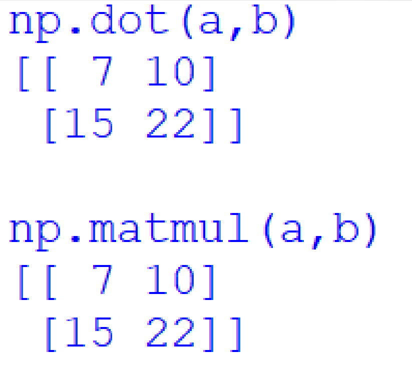
```
a*b
[[ 1  4]
 [ 9 16]]

np.multiply(a,b)
[[ 1  4]
 [ 9 16]]
```

- **Dot product**

  np.dot(a,b)

- **Matrix multiplication**

  np.matmul(a,b)



```
np.dot(a,b)
[[ 7 10]
 [15 22]]

np.matmul(a,b)
[[ 7 10]
 [15 22]]
```

# 2D NumPy Arrays

- **Syntax :**

  `<array_name> = np.array([[e11,e12, ….e1n],[e21,e22,…., e2n]])`

- **Indexing:**

  Multidimensional arrays can have one index per axis.

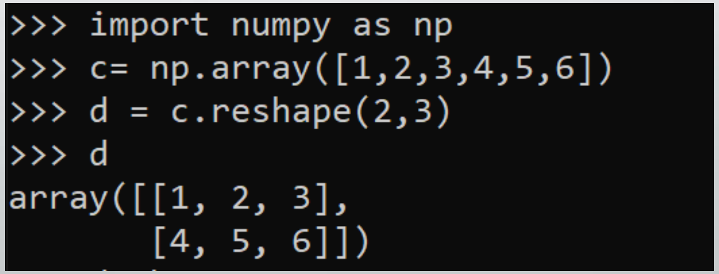
  The ‘ : ’ symbol indicates a complete slice

- **Shape Manipulation:**

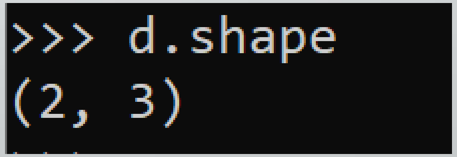
  The shape of an array is given by the number of elements along each axis. NumPy array have a “ shape” attribute that holds the shape of the array

  ***shape()***

  ***reshape()***



```
>>> import numpy as np
>>> c= np.array([1,2,3,4,5,6])
>>> d = c.reshape(2,3)
>>> d
array([[1, 2, 3],
       [4, 5, 6]])
```

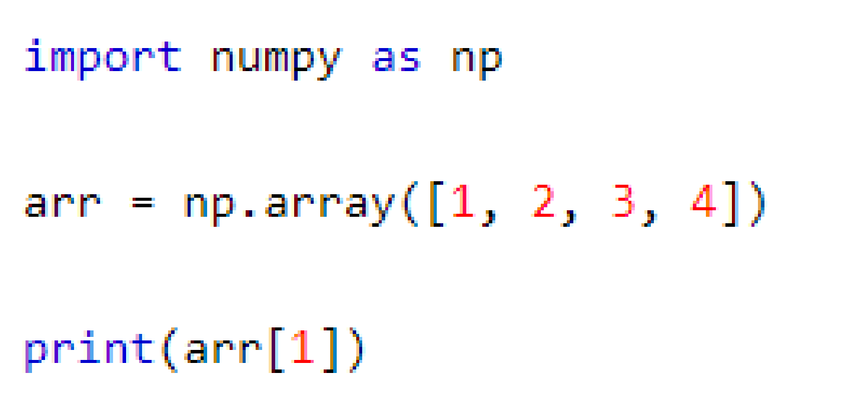


```
>>> d.shape
(2, 3)
```

# NumPy Array Indexing

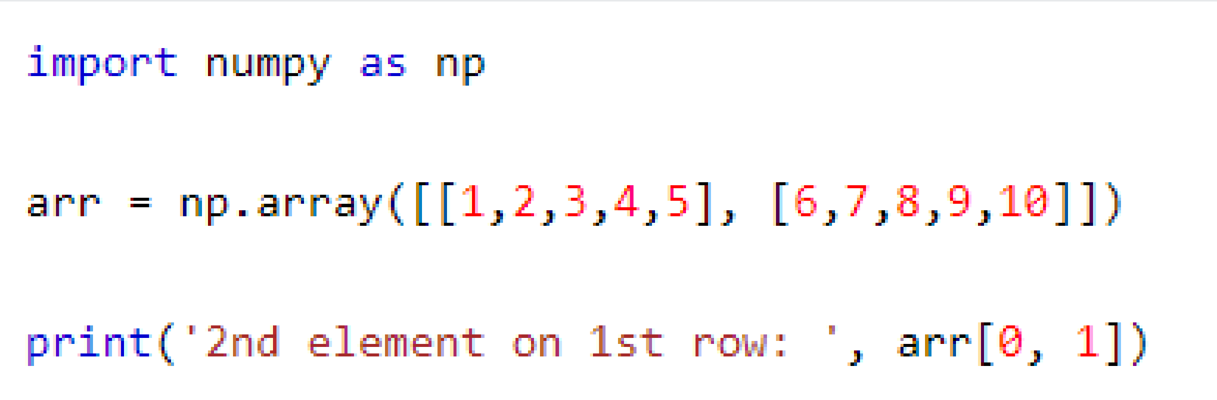
- NumPy arrays can be indexed and sliced the same way as Python lists, using square brackets

✓ To access the 2nd element



```
import numpy as np

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

print(arr[1])
```

✓ To access the element on the first row, second colum



```
import numpy as np

arr = np.array([[1,2,3,4,5], [6,7,8,9,10]])

print('2nd element on 1st row: ', arr[0, 1])
```

# Matplotlib Library

# Matplotlib

- **Matplotlib** is a comprehensive library/module for creating static, animated, and interactive visualizations in Python.
- **Installation**
  Within the terminal :
  ```
  pip install matplotlib
  ```
- **Importing matplotlib**
  ```
  import matplotlib.pyplot as plt
  ```

# matplotlib.pyplot

- **matplotlib.pyplot** is a collection of command style functions.

- pyplot functions can be used to make some change to a figure.

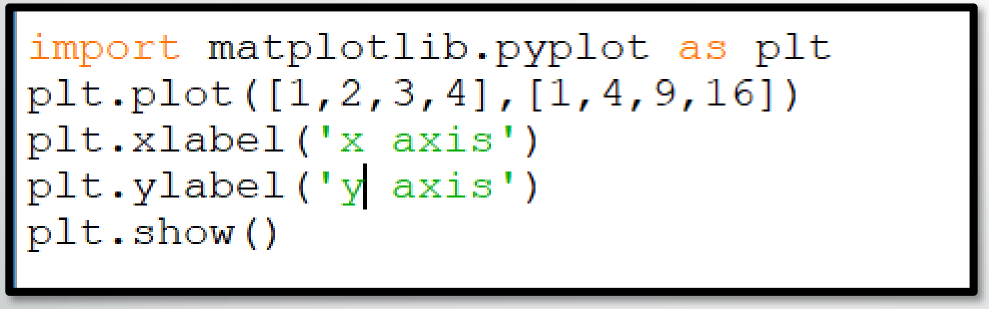
  **Ex:**

  - ✓ creates a figure
  - ✓ creates a plotting area in a figure
  - ✓ plots some lines in a plotting area
  - ✓ decorate the plot with labels

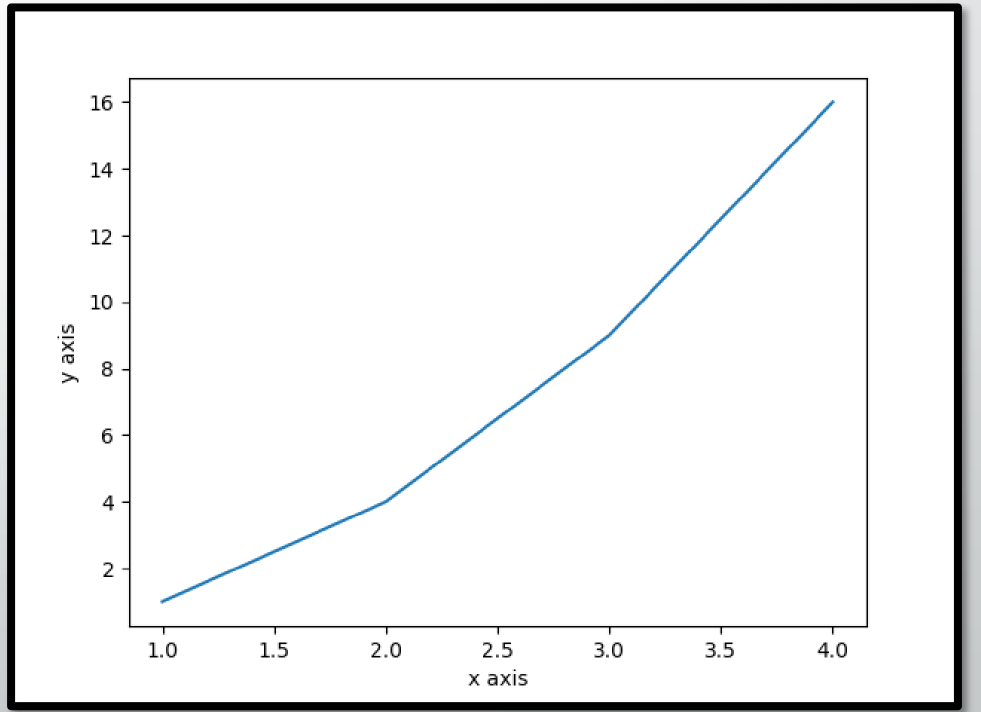
# Line Plot

- **Line plot syntax:**

  plt.plot([<x_values>] ,[<y_values>])



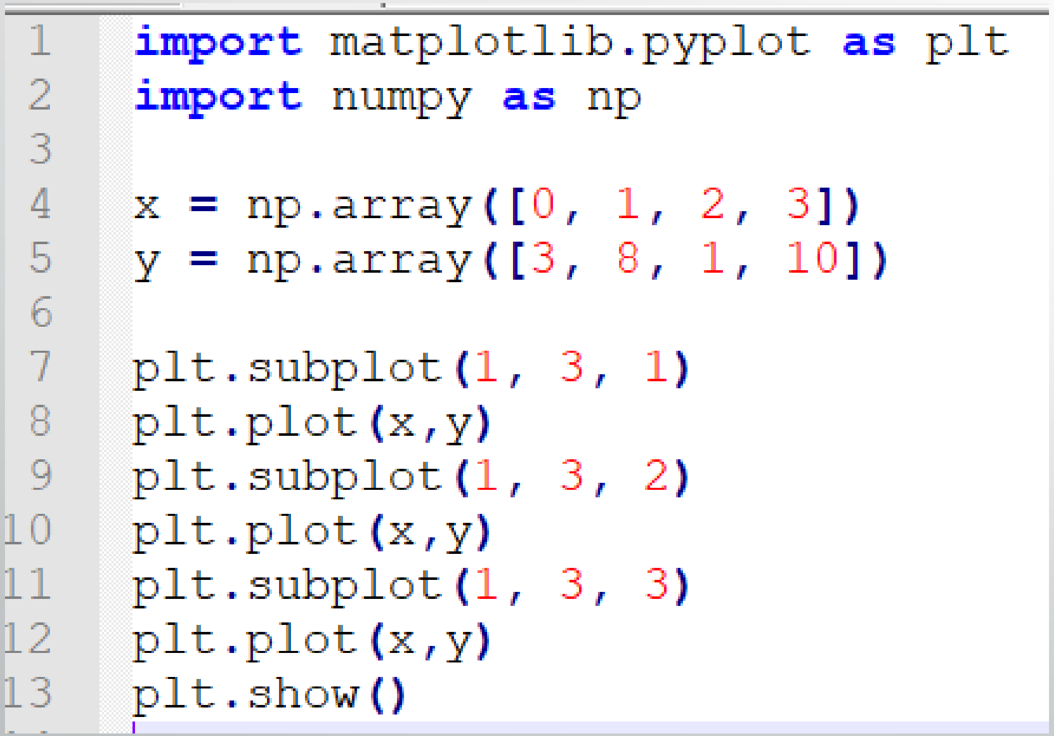
```
import matplotlib.pyplot as plt
plt.plot([1,2,3,4],[1,4,9,16])
plt.xlabel('x axis')
plt.ylabel('y axis')
plt.show()
```



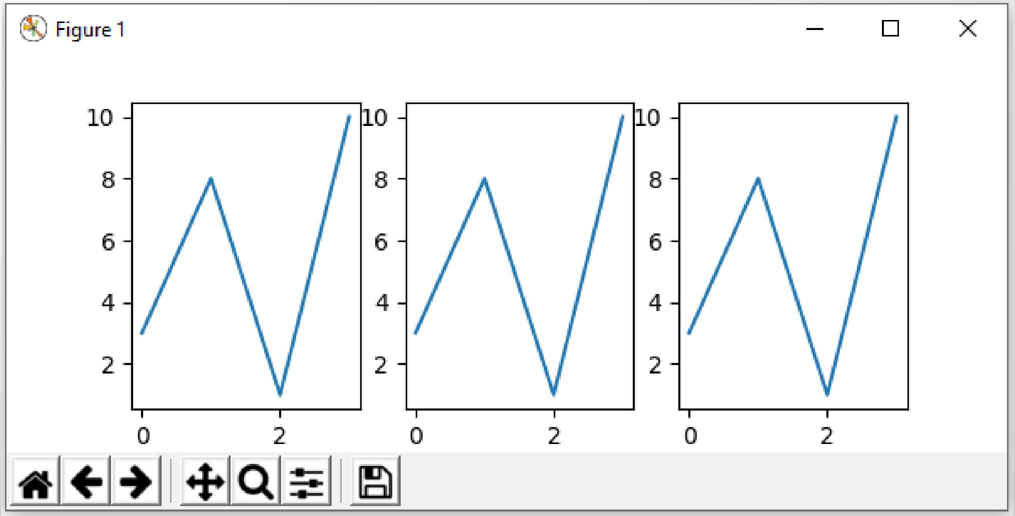
# Sub-Plots

- **Line plot syntax:**

  `plt.subplot(#row #column position)`



```
import matplotlib.pyplot as plt
import numpy as np

x = np.array([0, 1, 2, 3])
y = np.array([3, 8, 1, 10])

plt.subplot(1, 3, 1)
plt.plot(x,y)
plt.subplot(1, 3, 2)
plt.plot(x,y)
plt.subplot(1, 3, 3)
plt.plot(x,y)
plt.show()
```



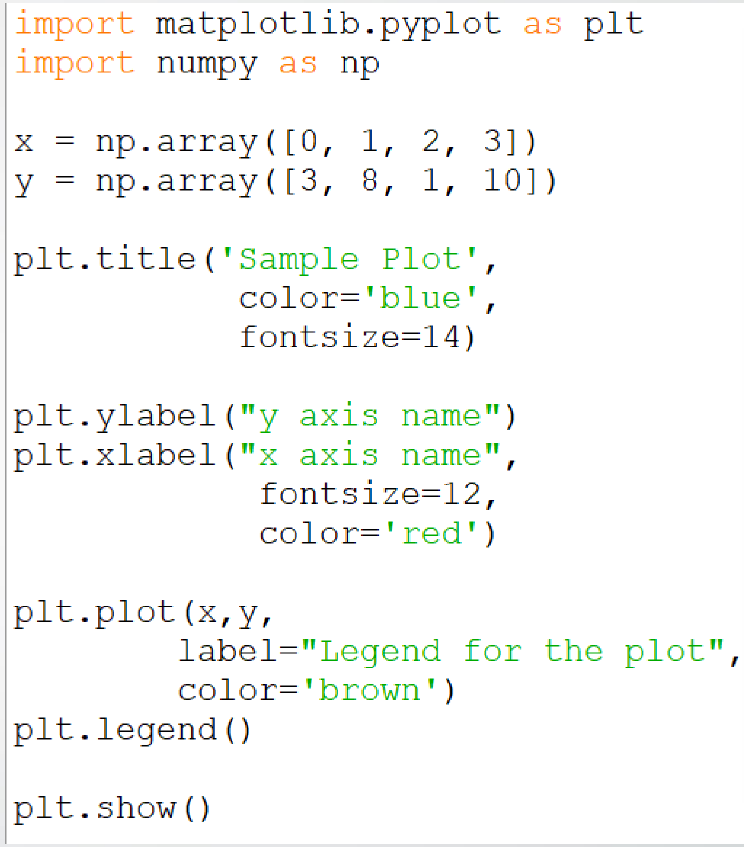
# Plots Features

- **Title and labels:**

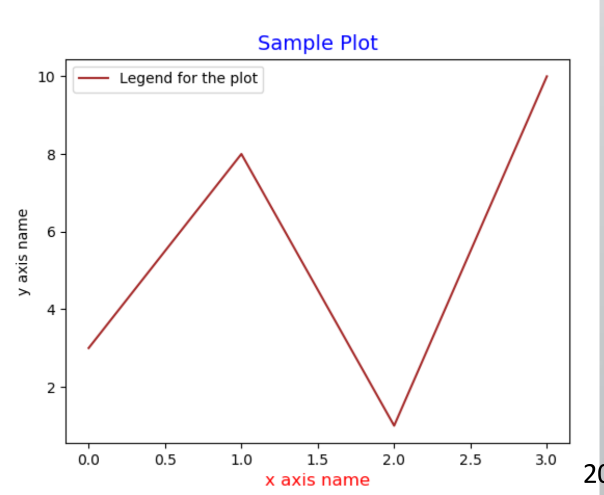
```
plt.title(‘<plot name>’)
plt.ylabel(‘<y axis name>’)
plt.xlabel(‘<x axis name>’,
             fontsize=<size>,
             color=<matplotlib color>)
```

- **Legend:**

```
plt.plot(x,y,label=‘name’)
plt.legend()
```



```
import matplotlib.pyplot as plt
import numpy as np

x = np.array([0, 1, 2, 3])
y = np.array([3, 8, 1, 10])

plt.title('Sample Plot',
           color='blue',
           fontsize=14)

plt.ylabel("y axis name")
plt.xlabel("x axis name",
            fontsize=12,
            color='red')

plt.plot(x,y,
         label="Legend for the plot",
         color='brown')
plt.legend()

plt.show()
```



# Formatting the style of the plot

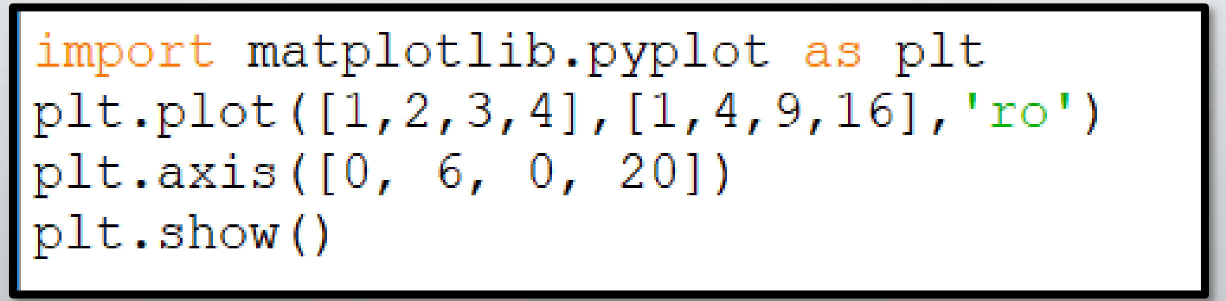
```
plt.plot([<x_values>], [<y_values>], <formatting style>)
```

- **formatting style:**

  ‘ro’ – red circles

  ‘b-’ – solid blue line(default)

  ‘gs’ – green squares

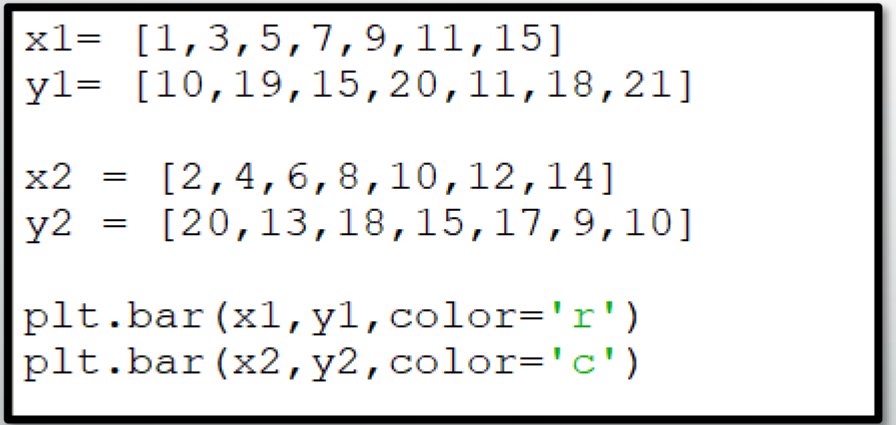


```
import matplotlib.pyplot as plt
plt.plot([1,2,3,4],[1,4,9,16],'ro')
plt.axis([0, 6, 0, 20])
plt.show()
```

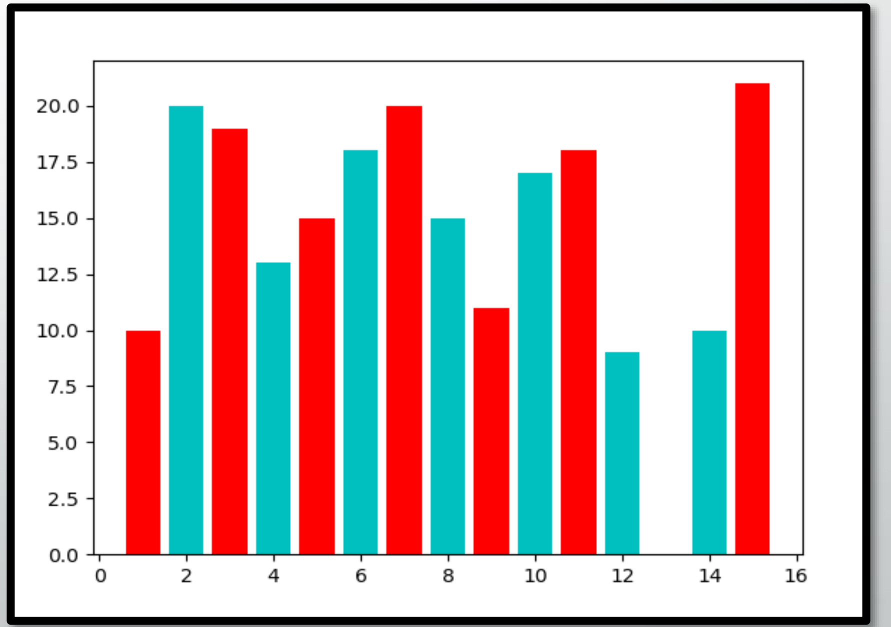
# Bar Plot

- **Bar plot syntax:**

```
plt.bar(<x_axis>,<y_axis>)
plt.bar(<x_axis>,<y_axis >, label = ‘<bar_name>’, color=‘<mat_color>’)
plt.plot([<x_values>] ,[<y_values>])
```



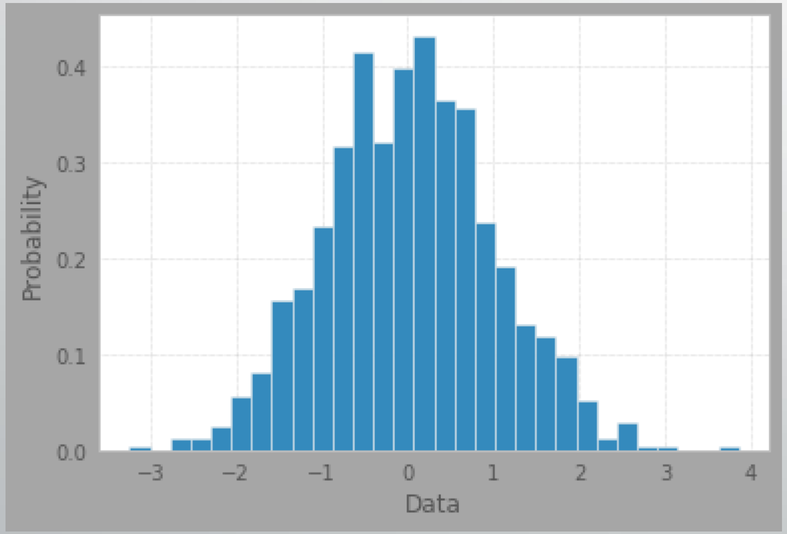
```
x1= [1,3,5,7,9,11,15]
y1= [10,19,15,20,11,18,21]

x2 = [2,4,6,8,10,12,14]
y2 = [20,13,18,15,17,9,10]

plt.bar(x1,y1,color='r')
plt.bar(x2,y2,color='c')
```



# Histogram & Scatterplot

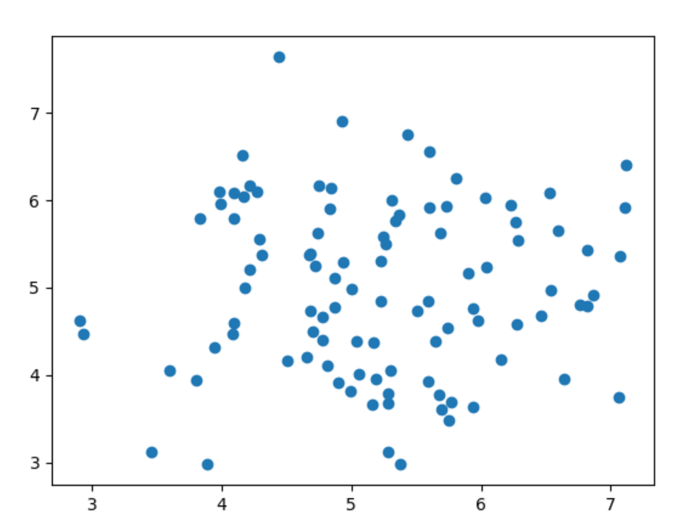
- **Histogram**

```
plt.hist(<data>,<bin>,
         histtype=<type>,
         rwidth= <size>)
```



- **Scatterplot**

```
plt.scatter(x, y,
            c = ‘<mat_color>’,
            s = <size>)
```



# Visualizing Images

- **Basic Syntax**

```
import numpy as np
import matplotlib.pyplot as plt

numpy_img_array = <read image using OpenCV>

plt.imshow(numpy_img_array)

plt.show()
```

# — END —