Introduction to Matplotlib

Matplotlib is a low level graph plotting library in python that serves as a visualization utility.

Matplotlib 1st Steps: install and import

- Matplotlib is an easy package to install. Open up your terminal program (shell or cmd) and install it using either of the following commands:

```
pip install matplotlib
```

- To import matplotlib we usually import it with a shorter name since it's used so much:

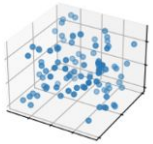
```
import matplotlib
```

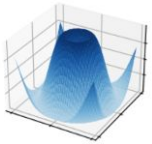
```
print(matplotlib.__version__)
```

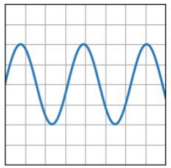
Matplotlib Pyplot

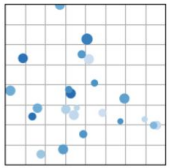
```
import matplotlib.pyplot as plt
```

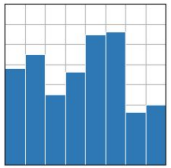
The *pyplot* module is typically used when you want to create basic plots without getting into the more detailed and lower-level aspects of Matplotlib. It provides a simple and intuitive way to create line plots, scatter plots, bar charts, histograms, and more.

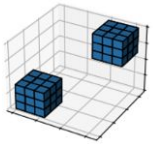

scatter(xs, ys, zs)

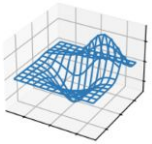

plot_surface(X, Y, Z)

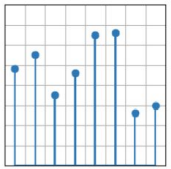

plot(x, y)

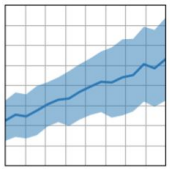

scatter(x, y)

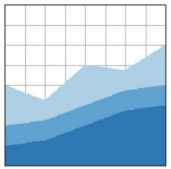

bar(x, height)


voxels([x, y, z], filled)


plot_wireframe(X, Y, Z)


stem(x, y)


fill_between(x, y1, y2)

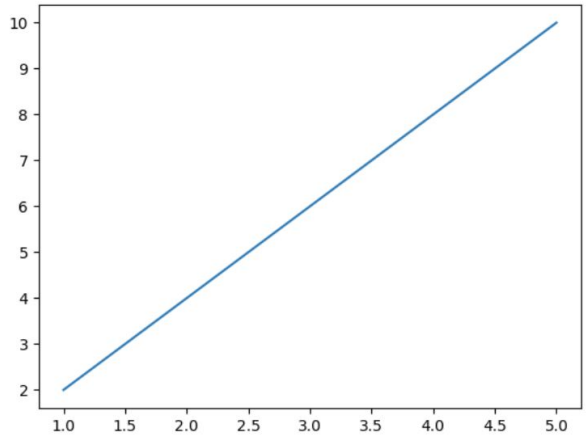

stackplot(x, y)

Matplotlib Plotting

Plotting x and y points: **plot()** function

- The `plot()` function is used to draw points (markers) in a diagram.
- By default, the `plot()` function draws a line from point to point.

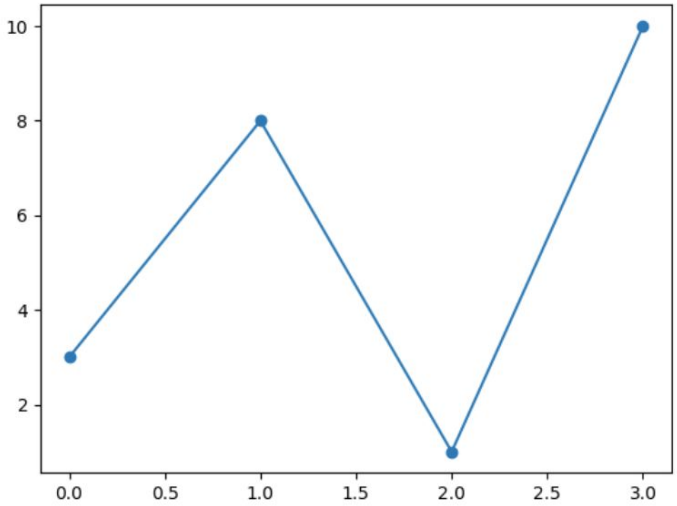
```
import matplotlib.pyplot as plt

# Sample data
x = [1, 2, 3, 4, 5]
y = [2, 4, 6, 8, 10]

# Create a line plot
plt.plot(x, y)
```



Matplotlib Markers

You can use the keyword argument `marker` to emphasize each point with a specified marker

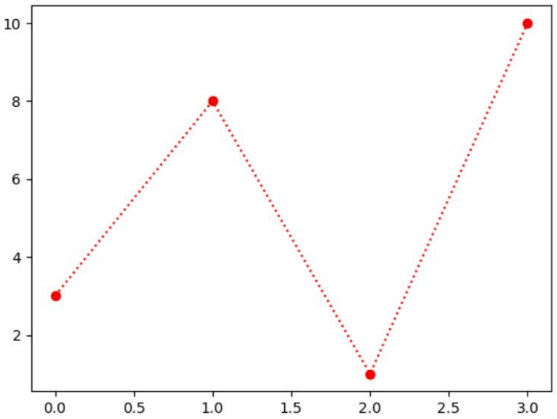
```
ypoints = np.array([3, 7, 5, 10])

plt.plot(ypoints, marker = 'o')
plt.show()
```



Matplotlib Markers

You can also use the *shortcut string notation* parameter to specify the marker. This parameter is also called `fmt`, and is written with this syntax: *`marker|line|color`*

```
plt.plot([3, 8, 1, 10], 'o:r')
```



Matplotlib Markers

Marker	Description
'o'	Circle
'*'	Star
'.'	Point
','	Pixel
'x'	X
'X'	X (filled)
'+'	Plus
'P'	Plus (filled)
's'	Square
'D'	Diamond

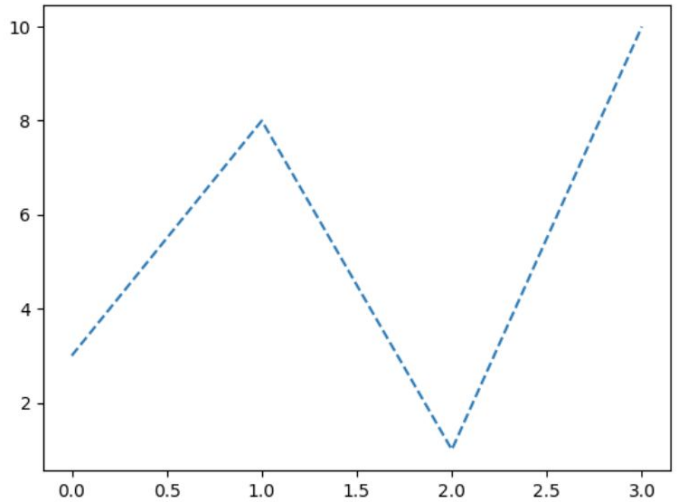
Color Syntax	Description
'r'	Red
'g'	Green
'b'	Blue
'c'	Cyan
'm'	Magenta
'y'	Yellow
'k'	Black
'w'	White

Line Syntax	Description
'-'	Solid line
':'	Dotted line
'--'	Dashed line
'-.'	Dashed/dotted line

Matplotlib Line: Linestyle

You can use the keyword argument `linestyle`, or shorter `ls`, to change the style of the plotted line:

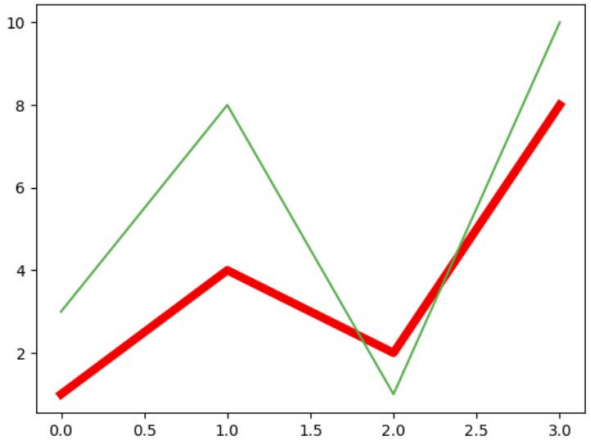
```
plt.plot([3, 8, 1, 10], linestyle = 'dashed')
plt.plot([3, 8, 1, 10], ls = ':')
```



Style	Or
'solid' (default)	'-'
'dotted'	':'
'dashed'	'--'
'dashdot'	'-.'
'None'	'' or ' '

Matplotlib Line: Line Color and Line Width

- You can use the keyword argument `color` or the shorter `c` to set the color of the line
- You can use the keyword argument `linewidth` or the shorter `lw` to change the width of the line.

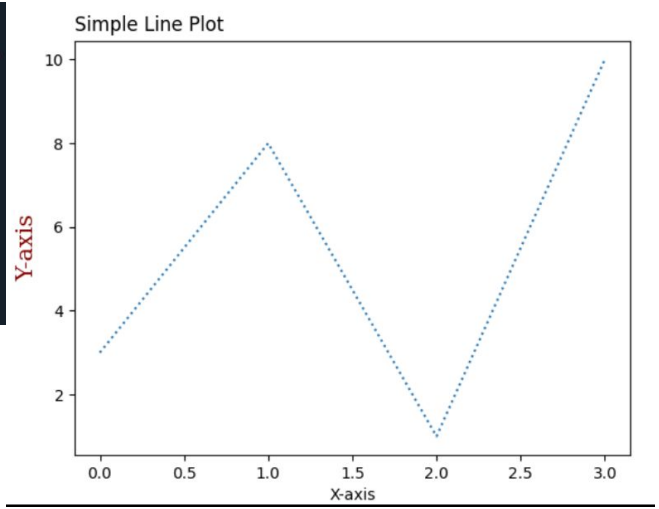



```
plt.plot([1, 4, 2, 8], color = 'r', lw = '5.5')
plt.plot([3, 8, 1, 10], c = '#4CAF50')
```

Matplotlib Labels and Title

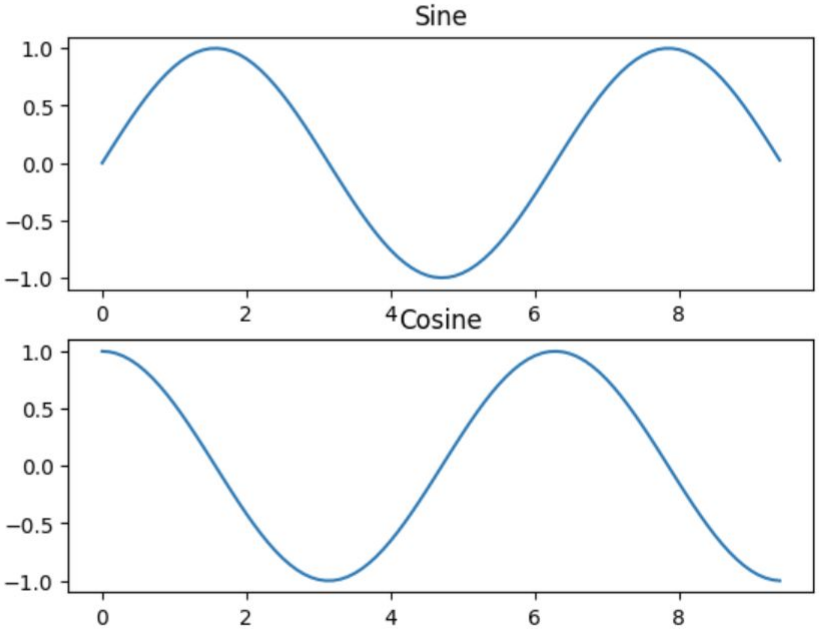
With Pyplot, you can use the `xlabel()` and `ylabel()` functions to set a label for the x- and y-axis. With Pyplot, you can use the `title()` function to set a title for the plot. You can use the `fontdict` parameter in `xlabel()`, `ylabel()`, and `title()` to set font properties for the title and labels.

```
font1 = {'family':'serif','color':'darkred','size':15}
plt.plot([3, 8, 1, 10], ls = ':')
plt.xlabel('X-axis')
plt.ylabel('Y-axis', fontdict = font1)
plt.title('Simple Line Plot', loc = 'left')
plt.show()
```



Matplotlib Subplot

With the `subplot()` function you can draw multiple plots in one figure:

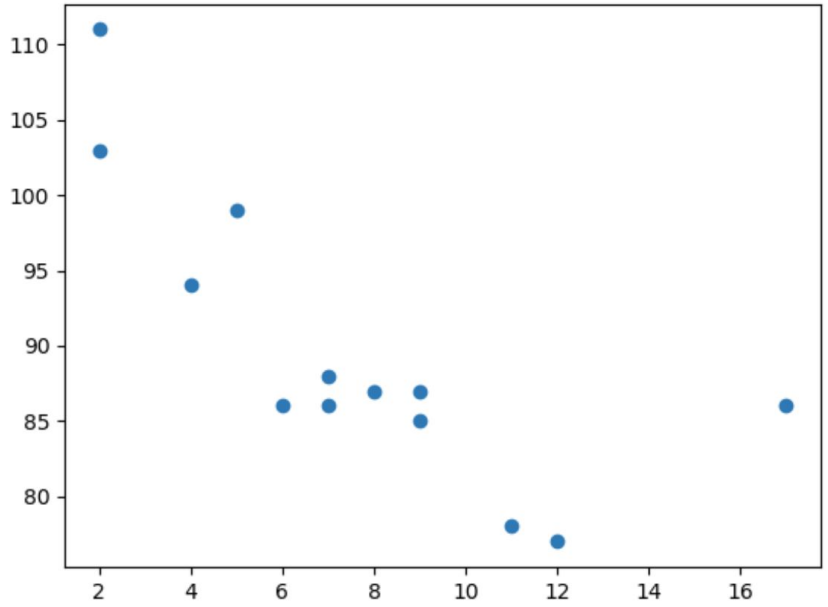



```
# Compute the x and y coordinates for points on sine and cosine curves
x = np.arange(0, 3 * np.pi, 0.1)
y_sin = np.sin(x)
y_cos = np.cos(x)

# 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()
```

Matplotlib Scatter

With Pyplot, you can use the `scatter()` function to draw a scatter plot. The `scatter()` function plots one dot for each observation. It needs two arrays of the same length, one for the values of the x-axis, and one for values on the y-axis:

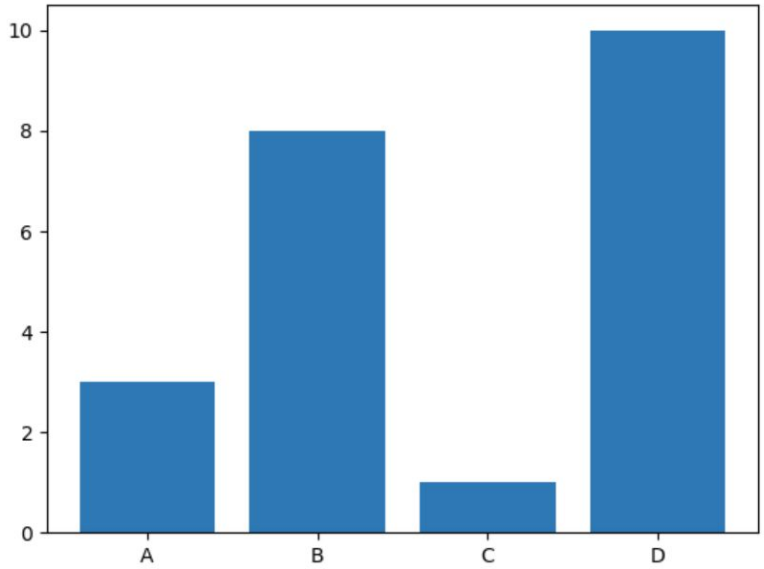
```
x = np.array([5,7,8,7,2,17,2,9,4,11,12,9,6])
y = np.array([99,86,87,88,111,86,103,87,94,78,77,85,86])

plt.scatter(x, y)
plt.show()
```



Matplotlib Bars

With Pyplot, you can use the `bar()` function to draw bar graphs:

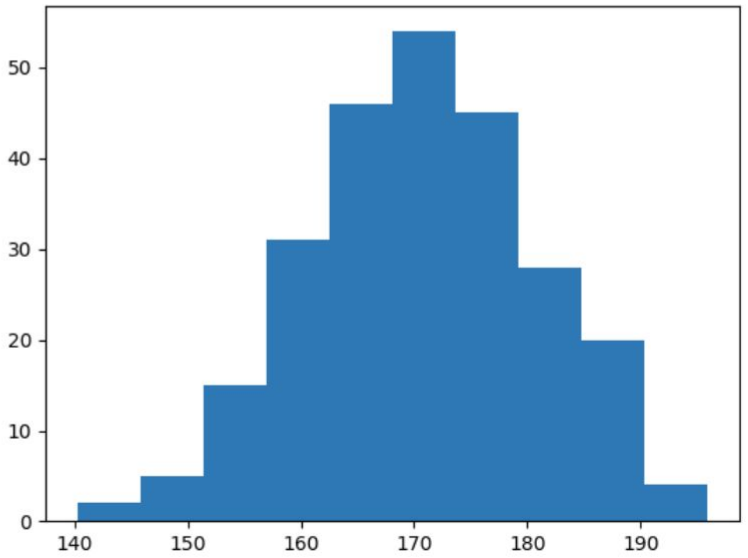
```
x = np.array(["A", "B", "C", "D"])
y = np.array([3, 8, 1, 10])

plt.bar(x,y)
plt.show()
```



Matplotlib Histograms

In Matplotlib, we use the `hist()` function to create histograms. The `hist()` function will use an array of numbers to create a histogram, the array is sent into the function as an argument.

```
x = np.random.normal(170, 10, 250)

plt.hist(x)
plt.show()
```



Matplotlib Pie Charts

With Pyplot, you can use the `pie()` function to draw pie charts:

```
y = np.array([35, 25, 25, 15])
mylabels = ["Apples", "Bananas", "Cherries", "Dates"]
myexplode = [0.2, 0, 0, 0]

plt.pie(y, labels = mylabels, explode = myexplode)
plt.show()
```