What is Matplotlib?

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

Matplotlib was created by John D. Hunter.

Matplotlib is open source and we can use it freely.

Matplotlib is mostly written in python, a few segments are written in C, Objective-C and Javascript for Platform compatibility.

## Where is the Matplotlib Codebase?

The source code for Matplotlib is located at this github repository <https://github.com/matplotlib/matplotlib>

# **Matplotlib Getting Started**

Installation of Matplotlib

If you have [Python](https://www.w3schools.com/python/default.asp) and [PIP](https://www.w3schools.com/python/python_pip.asp) already installed on a system, then installation of Matplotlib is very easy.

Install it using this command:

C:\Users\*Your Name*>pip install matplotlib

If this command fails, then use a python distribution that already has Matplotlib installed,  like Anaconda, Spyder etc.

Import Matplotlib

Once Matplotlib is installed, import it in your applications by adding the import *module* statement:

import matplotlib

Now Matplotlib is imported and ready to use:

## Checking Matplotlib Version

The version string is stored under \_\_version\_\_ attribute.

### **Example**

import matplotlib  
  
print(matplotlib.\_\_version\_\_)

--------------------------------------------------------

# **Matplotlib Pyplot**

Pyplot

Most of the Matplotlib utilities lies under the pyplot submodule, and are usually imported under the plt alias:

import matplotlib.pyplot as plt

Now the Pyplot package can be referred to as plt.

### **Example**

Draw a line in a diagram from position (0,0) to position (6,250):

# #Three lines to make our compiler able to draw:

# import sys

# import matplotlib

# matplotlib.use('Agg')

# import matplotlib.pyplot as plt

# import numpy as np

# xpoints = np.array([0, 6])

# ypoints = np.array([0, 250])

# plt.plot(xpoints, ypoints)

# plt.show()

# #Two lines to make our compiler able to draw:

# plt.savefig(sys.stdout.buffer)

# sys.stdout.flush()

# **Matplotlib Plotting**

## Plotting x and y points

The plot() function is used to draw points (markers) in a diagram.

By default, the plot() function draws a line from point to point.

The function takes parameters for specifying points in the diagram.

Parameter 1 is an array containing the points on the **x-axis**.

Parameter 2 is an array containing the points on the **y-axis**.

If we need to plot a line from (1, 3) to (8, 10), we have to pass two arrays [1, 8] and [3, 10] to the plot function.

### **Example**

Draw a line in a diagram from position (1, 3) to position (8, 10):

import matplotlib.pyplot as plt

import numpy as np

import sys

xpoints = np.array([1, 8])

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

plt.plot(xpoints, ypoints)

plt.show()

plt.savefig(sys.stdout.buffer)

sys.stdout.flush()

The **x-axis** is the horizontal axis.

The **y-axis** is the vertical axis.

-------------------------------------------------------------

### **Example**

Draw two points in the diagram, one at position (1, 3) and one in position (8, 10):

#Three lines to make our compiler able to draw:

import sys

import matplotlib

matplotlib.use('Agg')

import matplotlib.pyplot as plt

import numpy as np

xpoints = np.array([1, 8])

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

plt.plot(xpoints, ypoints, 'o')

plt.show()

#Two lines to make our compiler able to draw:

plt.savefig(sys.stdout.buffer)

sys.stdout.flush()

## Multiple Points

You can plot as many points as you like, just make sure you have the same number of points in both axis.

### **Example**

Draw a line in a diagram from position (1, 3) to (2, 8) then to (6, 1) and finally to position (8, 10):

import matplotlib.pyplot as plt  
import numpy as np  
  
xpoints = np.array([1, 2, 6, 8])  
ypoints = np.array([3, 8, 1, 10])  
  
plt.plot(xpoints, ypoints)

plt.show()

#Two lines to make our compiler able to draw:

plt.savefig(sys.stdout.buffer)

sys.stdout.flush()

==========================================

## Default X-Points

If we do not specify the points on the x-axis, they will get the default values 0, 1, 2, 3 (etc., depending on the length of the y-points.

So, if we take the same example as above, and leave out the x-points, the diagram will look like this:

### **Example**

Plotting without x-points:

import matplotlib.pyplot as plt  
import numpy as np  
  
ypoints = np.array([3, 8, 1, 10, 5, 7])  
  
plt.plot(ypoints)  
plt.show()

#Two lines to make our compiler able to draw:

plt.savefig(sys.stdout.buffer)

sys.stdout.flush()

-----------------------------------------------

# **Matplotlib Markers**

Markers

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

### **Example**

Mark each point with a circle:

import matplotlib.pyplot as plt  
import numpy as np  
  
ypoints = np.array([3, 8, 1, 10])  
  
plt.plot(ypoints, marker = 'o')  
plt.show()

#Two lines to make our compiler able to draw:

plt.savefig(sys.stdout.buffer)

sys.stdout.flush()

# **Matplotlib Line**

## Linestyle

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

### **Example**

Use a dotted line:

import matplotlib.pyplot as plt  
import numpy as np  
  
ypoints = np.array([3, 8, 1, 10])  
  
plt.plot(ypoints, linestyle = 'dotted')  
plt.show()

#Two lines to make our compiler able to draw:

plt.savefig(sys.stdout.buffer)

sys.stdout.flush()

### **Example**

Use a dashed line:

plt.plot(ypoints, linestyle = 'dashed')

# **Matplotlib Labels and Title**

## Create Labels for a Plot

With Pyplot, you can use the xlabel() and ylabel() functions to set a label for the x- and y-axis.

### **Example**

Add labels to the x- and y-axis:

import numpy as np  
import matplotlib.pyplot as plt  
  
x = np.array([80, 85, 90, 95, 100, 105, 110, 115, 120, 125])  
y = np.array([240, 250, 260, 270, 280, 290, 300, 310, 320, 330])  
  
plt.plot(x, y)  
  
plt.xlabel("Average Pulse")  
plt.ylabel("Calorie Burnage")  
  
plt.show()

#Two lines to make our compiler able to draw:

plt.savefig(sys.stdout.buffer)

sys.stdout.flush()

============================================================

## Create a Title for a Plot

With Pyplot, you can use the title() function to set a title for the plot.

### **Example**

Add a plot title and labels for the x- and y-axis:

import numpy as np  
import matplotlib.pyplot as plt  
  
x = np.array([80, 85, 90, 95, 100, 105, 110, 115, 120, 125])  
y = np.array([240, 250, 260, 270, 280, 290, 300, 310, 320, 330])  
  
plt.plot(x, y)  
  
plt.title("Sports Watch Data")  
plt.xlabel("Average Pulse")  
plt.ylabel("Calorie Burnage")  
  
plt.show()

#Two lines to make our compiler able to draw:

plt.savefig(sys.stdout.buffer)

sys.stdout.flush()

-------------------------------------------------------

# **Matplotlib Scatter**

## Creating Scatter Plots

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:

### **Example**

A simple scatter plot:

import matplotlib.pyplot as plt  
import numpy as np  
  
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()

#Two lines to make our compiler able to draw:

plt.savefig(sys.stdout.buffer)

sys.stdout.flush()

The observation in the example above is the result of 13 cars passing by.

The X-axis shows how old the car is.

The Y-axis shows the speed of the car when it passes.

Are there any relationships between the observations?

It seems that the newer the car, the faster it drives, but that could be a coincidence, after all we only registered 13 cars.

-------------------------------------------------------------------------

## Colors

You can set your own color for each scatter plot with the color or the c argument:

### **Example**

Set your own color of the markers:

import matplotlib.pyplot as plt  
import numpy as np  
  
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, color = 'hotpink')  
  
x = np.array([2,2,8,1,15,8,12,9,7,3,11,4,7,14,12])  
y = np.array([100,105,84,105,90,99,90,95,94,100,79,112,91,80,85])  
plt.scatter(x, y, color = '#88c999')  
  
plt.show()

#Two lines to make our compiler able to draw:

plt.savefig(sys.stdout.buffer)

sys.stdout.flush()

------------------------------------------------------------  
Matplotlib Bars

## Creating Bars

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

### **Example**

Draw 4 bars:

import matplotlib.pyplot as plt  
import numpy as np  
  
x = np.array(["A", "B", "C", "D"])  
y = np.array([3, 8, 1, 10])  
  
plt.bar(x,y)  
plt.show()

#Two lines to make our compiler able to draw:

plt.savefig(sys.stdout.buffer)

sys.stdout.flush()

The bar() function takes arguments that describes the layout of the bars.

The categories and their values represented by the first and second argument as arrays.

### **Example**

x = ["APPLES", "BANANAS"]  
y = [400, 350]  
plt.bar(x, y)

## Horizontal Bars

If you want the bars to be displayed horizontally instead of vertically, use the barh() function:

### **Example**

Draw 4 horizontal bars:

import matplotlib.pyplot as plt  
import numpy as np  
  
x = np.array(["A", "B", "C", "D"])  
y = np.array([3, 8, 1, 10])  
  
plt.barh(x, y)  
plt.show()

#Two lines to make our compiler able to draw:

plt.savefig(sys.stdout.buffer)

sys.stdout.flush()

## Bar Color

The bar() and barh() take the keyword argument color to set the color of the bars:

### **Example**

Draw 4 red bars:

import matplotlib.pyplot as plt  
import numpy as np  
  
x = np.array(["A", "B", "C", "D"])  
y = np.array([3, 8, 1, 10])  
  
plt.bar(x, y, color = "red")  
plt.show()

#Two lines to make our compiler able to draw:

plt.savefig(sys.stdout.buffer)

sys.stdout.flush()

### **Color Names**

You can use any of the 140 supported color names.

### **Example**

Draw 4 "hot pink" bars:

import matplotlib.pyplot as plt  
import numpy as np  
  
x = np.array(["A", "B", "C", "D"])  
y = np.array([3, 8, 1, 10])  
  
plt.bar(x, y, color = "hotpink")  
plt.show()

#Two lines to make our compiler able to draw:

plt.savefig(sys.stdout.buffer)

sys.stdout.flush()

=======================================================

# **Matplotlib Histograms**

## Histogram

A histogram is a graph showing frequency distributions.

It is a graph showing the number of observations within each given interval.

Example: Say you ask for the height of 250 people, you might end up with a histogram like this:

## Create Histogram

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.

For simplicity we use NumPy to randomly generate an array with 250 values, where the values will concentrate around 170, and the standard deviation is 10

### **Example**

A Normal Data Distribution by NumPy:

import numpy as np  
  
x = np.random.normal(170, 10, 250)  
  
print(x)

The hist() function will read the array and produce a histogram:

### **Example**

A simple histogram:

import matplotlib.pyplot as plt  
import numpy as np  
  
x = np.random.normal(170, 10, 250)  
  
plt.hist(x)  
plt.show()

#Two lines to make our compiler able to draw:

plt.savefig(sys.stdout.buffer)

sys.stdout.flush()

=================================================================

# **Matplotlib Pie Charts**

## Creating Pie Charts

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

### **Example**

A simple pie chart:

import matplotlib.pyplot as plt  
import numpy as np  
  
y = np.array([35, 25, 25, 15])  
  
plt.pie(y)  
plt.show()

#Two lines to make our compiler able to draw:

plt.savefig(sys.stdout.buffer)

sys.stdout.flush()

## Labels

Add labels to the pie chart with the label parameter.

The label parameter must be an array with one label for each wedge:

### **Example**

A simple pie chart:

import matplotlib.pyplot as plt  
import numpy as np  
  
y = np.array([35, 25, 25, 15])  
mylabels = ["Apples", "Bananas", "Cherries", "Dates"]  
  
plt.pie(y, labels = mylabels)  
plt.show()

#Two lines to make our compiler able to draw:

plt.savefig(sys.stdout.buffer)

sys.stdout.flush()