## Functions

[matplotlib.pyplot](https://matplotlib.org/api/_as_gen/matplotlib.pyplot.html#module-matplotlib.pyplot) is a state-based interface to matplotlib

P:

**Logic**:

1.  If you provide a single list or array to the [plot()](https://matplotlib.org/api/_as_gen/matplotlib.pyplot.plot.html#matplotlib.pyplot.plot) command, matplotlib assumes it is a sequence of y values, and automatically generates the x values for you. Since python ranges start with 0, the default x vector has the same length as y but starts with 0. Hence the x data are [0,1,2,3].

**Code**:

import matplotlib.pyplot as plt;

plt.plot([1,2,3,4,5]);

plt.ylabel("Some numbers in y");

plt.xlabel("Some numbers in x");

plt.show();

**Output**:

Chart, line chart

Description automatically generated

P:

**Logic**:

**Code**:

import matplotlib.pyplot as plt

plt.plot([1, 2, 3, 4], [1, 4, 9, 16])

plt.ylabel("Some numbers in y");

plt.xlabel("Some numbers in x");

plt.show();

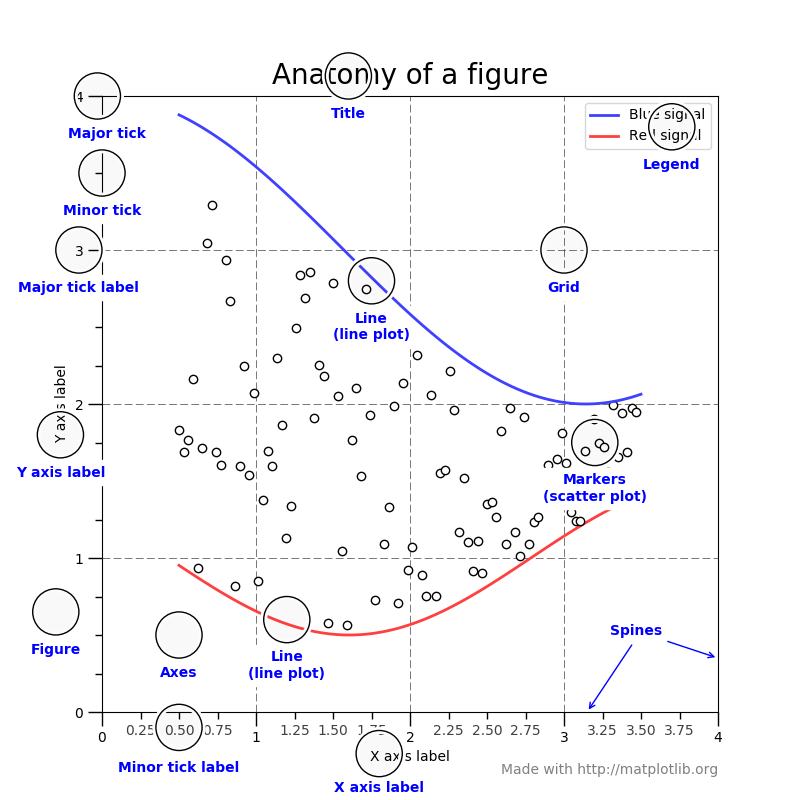
**Output**:

Chart, line chart

Description automatically generated

*HEADING*

**Parts of figure**:



[**Figure**](https://matplotlib.org/api/_as_gen/matplotlib.figure.Figure.html#matplotlib.figure.Figure)**:**

The **whole** figure. The figure keeps track of all the child [Axes](https://matplotlib.org/api/axes_api.html#matplotlib.axes.Axes), a smattering of 'special' artists (titles, figure legends, etc), and the **canvas**. A figure can have any number of [Axes](https://matplotlib.org/api/axes_api.html#matplotlib.axes.Axes), but to be useful should have at least one.

**Axes**:

This is what you think of as 'a plot', it is the region of the image with the data space. A given figure can contain many Axes, but a given [Axes](https://matplotlib.org/api/axes_api.html#matplotlib.axes.Axes) object can only be in one [Figure](https://matplotlib.org/api/_as_gen/matplotlib.figure.Figure.html#matplotlib.figure.Figure).

### [Axis](https://matplotlib.org/api/axis_api.html#matplotlib.axis.Axis):

### These are the number-line-like objects.

### They take care of setting the graph limits and generating the ticks (the marks on the axis) and ticklabels (strings labeling the ticks).

### The location of the ticks is determined by a [Locator](https://matplotlib.org/api/ticker_api.html#matplotlib.ticker.Locator) object and the ticklabel strings are formatted by a [Formatter](https://matplotlib.org/api/ticker_api.html#matplotlib.ticker.Formatter). The combination of the correct Locator and Formatter gives very fine control over the tick locations and labels.

### [Artist](https://matplotlib.org/api/artist_api.html#matplotlib.artist.Artist):

### Basically everything you can see on the figure is an artist (even the Figure, Axes, and Axis objects). This includes Text objects, Line2D objects,collection objects, Patch objects .

### When the figure is rendered, all of the artists are drawn to the **canvas**.

P:

**Logic**:

**Code**:

from matplotlib import pyplot as plt

fig = plt.figure() # an empty figure with no axes

fig.suptitle('No axes on this figure') # Add a title so we know which it is

fig, ax\_lst = plt.subplots(2, 2) # a figure with a 2x2 grid of Axes

**Output**:

Chart, table

Description automatically generated with medium confidence

fig, ax\_lst = plt.subplots(3, 2)

Chart

Description automatically generated

P:

**Logic**:

**Code**:

import matplotlib;

import matplotlib;

#Importing pyplot

from matplotlib import pyplot as plt

#Plotting to our canvas

plt.plot([1,2,3],[4,5,1])

#Showing what we plotted

plt.show()

**Output**:

Chart, line chart

Description automatically generated

P:Labels

**Logic**:

**Code**:

from matplotlib import pyplot as plt

x = [5,8,10]

y = [12,16,6]

plt.plot(x,y)

plt.title('Epic Info')

plt.ylabel('Y axis')

plt.xlabel('X axis')

plt.show()

**Output**:

Chart, line chart

Description automatically generated

*HEADING*

Using numpy

**Logic**:

**Code**:

# Import the necessary packages and modules

import matplotlib as mpl

import matplotlib.pyplot as plt

import numpy as np

# Uncomment following line to see the effect

#mpl.rcParams['lines.linewidth'] = 5

# Prepare the data

x = np.linspace(0, 10, 100)

# Plot the data

plt.plot(x, x, label='linear')

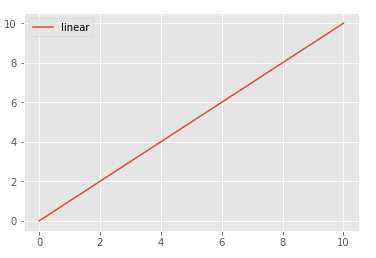
# Add a legend

plt.legend()

# Show the plot

plt.show()

**Output**:



P:Pie

**Logic**:

**Code**:

import matplotlib.pyplot as plt

# Pie chart, where the slices will be ordered and plotted counter-clockwise:

labels = 'Frogs', 'Hogs', 'Dogs', 'Logs'

sizes = [15, 30, 45, 10]

explode = (0, 0.1, 0, 0) # only "explode" the 2nd slice (i.e. 'Hogs')

fig1, ax1 = plt.subplots()

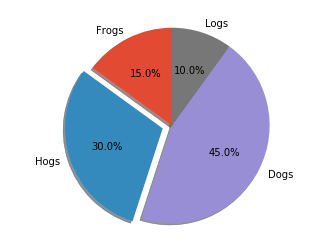
ax1.pie(sizes, explode=explode, labels=labels, autopct='%1.1f%%',

shadow=True, startangle=90)

ax1.axis('equal') # Equal aspect ratio ensures that pie is drawn as a circle.

plt.show()

**Output**:



P:

**Logic**:

**Code**:

import matplotlib.pyplot as plt

labels = ['Cookies', 'Jellybean', 'Milkshake', 'Cheesecake']

sizes = [38.4, 40.6, 20.7, 10.3]

colors = ['yellowgreen', 'gold', 'lightskyblue', 'lightcoral']

patches, texts = plt.pie(sizes, colors=colors, shadow=True, startangle=90)

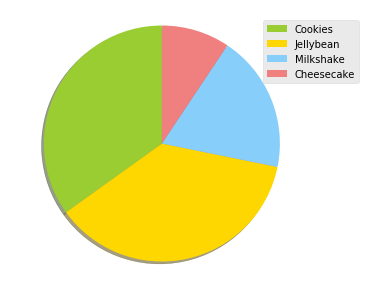
plt.legend(patches, labels, loc="best")

plt.axis('equal')

plt.tight\_layout()

plt.show()

**Output**:



P:Quiver Plot

**Logic**:

**Code**:

import numpy as np

import matplotlib.pyplot as plt

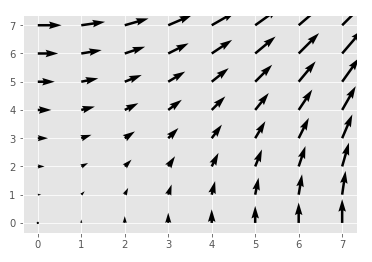
n = 8

X,Y = np.mgrid[0:n,0:n]

plt.quiver(X,Y)

plt.show()

**Output**:



P:

**Logic**:

**Code**:

import numpy as np

import matplotlib.pyplot as plt

from mpl\_toolkits.mplot3d import Axes3D

fig = plt.figure()

ax = Axes3D(fig)

X = np.arange(-4, 4, 0.25)

Y = np.arange(-4, 4, 0.25)

X, Y = np.meshgrid(X, Y)

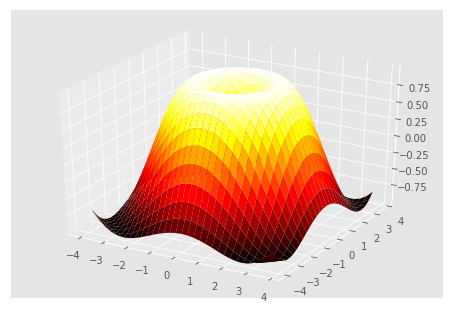
R = np.sqrt(X\*\*2 + Y\*\*2)

Z = np.sin(R)

ax.plot\_surface(X, Y, Z, rstride=1, cstride=1, cmap='hot')

plt.show()

**Output**:



P:

**Logic**:

**Code**:

from matplotlib import pyplot as plt

from matplotlib import style

style.use('ggplot')

x = [1,2,6,10]

y = [12,16,14,6]

plt.bar(x, y,color='g', align='center')

plt.title('Epic Info')

plt.ylabel('Y axis')

plt.xlabel('X axis')

plt.show()

**Output**:

Chart, bar chart

Description automatically generated

P:Scatter Plot

**Logic**:

scatter plots

**Code**:

from matplotlib import pyplot as plt

from matplotlib import style

style.use('ggplot')

x = [5,8,10]

y = [12,16,6]

plt.scatter(x, y)#, align='center')

plt.title('Epic Info')

plt.ylabel('Y axis')

plt.xlabel('X axis')

plt.show()

**Output**:

Chart, scatter chart

Description automatically generated