

Matplotlib

What is matplotlib?

- created by John D Hunter.
- open source.
- low level graph plotting library in python that helps in data visualization.

Import

```
import matplotlib as
```

check version

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

Pyplot

most of its utilities lies under the pyplot imported as `matplotlib.pyplot` as `plt`.

Eg Draw a line from (0,0) to (6,250)

```
x = np.array([0, 6])
```

```
y = np.array([0, 650])
```

```
xpoints = np.linspace
```

```
plt.plot(xpoints, ypoints)
```

```
plt.show()
```

Matplotlib plotting

`plot()` is used to draw points in a diagram.
by default it draws a line point to point.

Takes two parameters

$x \rightarrow$ array containing points on x axis

$y \rightarrow$ array containing points on y axis.

To plot a line

\Rightarrow `plt.plot(xpoint, ypoint, 'r')`

To remove line.

\rightarrow `plt.plot(x, y, 'o')`

Note we can use multiple numbers to plot but the count in both axis should match.

Default points

if we leave the x points as default.

\therefore if we do not specify x points then they are assumed as 0, 1, 2, ... etc.

```
ypoint = np.array([3, 8, 10, 5, 7])
```

```
plt.plot(ypoint)
```

```
plt.show()
```

Here x points are assumed $\rightarrow 0, 1, 2, 3, 4$



Matplotlib Markers

we can use argument 'marker' to emphasize each point.

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

Note There are a lot of markers like star, diamond, etc.

fmt

we can use shortcut string notation.

This parameter is called 'fmt' and syntax is.

marker | line | color

eg) `plt.plot(x, y, o:r)`

This tells `o` → marker

`:` → dotted line

`r` → red.

line reference

`'-'` → dashed solid

`'.'` → dotted

`'--'` → dashed.

`'-.'` → dashed dotted

Note If we do not specify any line then no line is plotted

marker size (ms)

we can also define size of marker.

```
plt.plot(ypoints, marker='o', ms=20)
```

marker color

we can set color of marker by mec \rightarrow markeredgecolor.

```
plt.plot(ypoints, marker='x', ms=20, mec='r')
```

It is used for edge.

for whole marker

we use mfc \rightarrow markerfacecolor.

```
plt.plot(ypoints, marker='o', ms=20, mfc='r')
```

Note use both 'mec' and 'mfc' to color entire marker.

Matplotlib line

we can plot different types, color, size of line.

To plot different types of line.

\rightarrow linestyle or ls \rightarrow ls = 'dashed' or '--'

\rightarrow color \rightarrow 'r'

\rightarrow linewidth or lw \rightarrow lw = 15

```
plt.plot(ypoint, ls='dashed', color='r', lw=15)
```


Multiple lines

we can plot multiple lines.

```
x = np.array([1, 2, 3, 4])  
y = np.array([5, 7, 9, 3])
```

```
plt.plot(x)
```

```
plt.plot(y)
```

```
plt.show()
```

Note We can also add multiple lines in same plot.

```
x1 = np.array([2, 4, 6, 8])
```

```
y1 = np.array([3, 5, 7, 9])
```

```
x2 = np.array([10, 9, 6, 4])
```

```
x2 = np.array([3, 9, 2, 5])
```

```
plt.plot(x1, y1, x2, y2)
```

```
plt.show()
```

Matplotlib labels

we can create labels for the plot.

To do this we use

xlabel → for x axis

ylabel → for y axis

```
plt.xlabel('duration')
```

```
plt.ylabel('calories')
```

```
plt.show()
```

Title

we can give a title to the plot.

```
plt.title('sports match')
```

Note we can set font properties of these functions also.

we use 'fontdict' to do this.

```
plt.title(" ", fontdict=font1)
plt.xlabel(" ", fontdict=font2)
```

font1 → for title

font2 → for labels.

Title Position

we can set position of title on left, center, right.

To achieve this we 'loc' parameter.

```
plt.title(" " loc='left', fontdict=font1)
```

Add Grid Lines

we can use grid lines to plot data.

```
→ plt.grid()
```

we can also specify which grid lines to display.
legal values are x, y and both.



Default is both.

`plt.grid (axis = x, linestyle = '--', linewidth = 0.5)`

Note we cannot use short forms here.

Subplot

This helps to plot multiple plots in same plot.
It takes three parameters.

- i no of rows
- ii no of columns
- iii Which plot number.

for eg.

`plt.subplot (1 2 1)`

This state that one row and 2 columns.
Here the two plots are side by side.

`plt.subplot (2, 1 2)`

This shows 2 rows and 1 column.
Here both figure on top of one another.

Note we can add as much plots as we want.

we can add titles to each plot.

Note we can add a supertitle for the plot.

`plt.suptitle (" ")`

Matplotlib Scatter

we can use `scatter()` to draw a scatter plot.
it needs two array of same length.

Multiple plots

we can plot two scatter plots in one figure. They have different colors.

Color

we can change color of different plots by color argument.

```
plt.scatter(x, y, color='hotpink')
```

Color each dot

we can create an array of same length containing different colors.

The syntax is:

```
color = np.array([ ])
```

```
plt.scatter(x, y, c=colors)
```

```
plt.show()
```

colormap

It is a list of colors.

Each color has a value range of 0 to 100.

we can specify colormap as.



```
color = np.array([0, 10, 20, 30, 40, 50, 60, 70, 80, 90, 100])
```

```
plt.scatter(x, y, c=colors, cmap='viridis')
```

Note There are a lot of cmaps available.

we can also show colormap in the plot by:-

→ `plt.colorbar()`

Size

we can change size of dots by 's' argument.

```
sizes = np.array([1, 5, 6, 10, 20, 40, 9, 11, 5])
```

```
plt.scatter(x, y, s=sizes)
```

Make sure array sizes is of same length as x and y.

Alpha

This is used to adjust transparency of dots.

```
plt.scatter(x, y, alpha=0.5)
```

Matplotlib bars

we can use `bar()` to create bar charts.

```
x = np.array(['A', 'B', 'C'])  
y = np.array([3, 8, 9, 1])
```

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

This function takes several arguments.

category \rightarrow x
values \rightarrow y.

if we want horizontal bars then.

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

width

set the width of bars.

```
plt.bar(x, y, width=0.1)
```

color

used to set color of bars.

```
plt.bar(x, y, width=0.1, color='red')
```

height

we use height instead of width in horizontal bars.



```
plt.bar(x, y, height=1, color='red')
```

Note the default height and width is 0.8

Histograms

graph showing frequency distribution

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

We use `hist()` to achieve this.

This takes array of numbers as an argument.

```
import numpy as np
```

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

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

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

```
plt.show()
```

It takes parameters

- `bins` - number of bins
- `color` - bin color
- `edgecolor` - edge color of bins
- `data` - array of numbers

Pie charts

To create pie chart we use `pie()`.

`plt.pie(y)`

It starts drawing counterclockwise.

Size of each wedge = $\frac{x}{\text{sum}(x)}$

labels

- we can also provide labels to it.
- The labels should be a on array with one label for each wedge

```
x = np.array([55, 10, 15, 20])
mylabel = ["apple", "bannana", "kiwi", "orange"]
```

```
plt.pie(x, labels = mylabels)
plt.show()
```

autopct

This is also used to show percent.

```
(autopct = '%.1f%%')
```

startangle

It is used to change the start angle.
By default it is 0

`plt.pie(x, labels=mylabels, autopct='%1.1f%%', startangle=90)`

Explode

- This is used to highlight one of the edge.
- There must be an array or list having values for each wedge.
- This value tells how far it is from the center.

`myexplode = [0.2, 0, 0, 0]`

`plt.pie(x, label=mylabels, explode=myexplode)`

Shadow

used to add shadows to all the wedges.

⇒ `Shadow = True`.

Colors

must be a list or array containing colors for all value.

`colors = mycolors`

colors here refers to an array with color values.

Legend

used to add a small list to explain each wedge.

`plt.legend()`



we can also add title to our legend.

```
plt.legend(title = 'Four Fruits')
plt.show()
```