* plotting graphs

import matplotlib.pyplot as plt

matplotlib → module

pyplot → a part of it

Arrays representing the coordinates along different axes for the points to be plotted.

```
x = [1, 2, 3]
y = [2, 4, 6]
```

(points are not joined)

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

→ does not show the graph, just puts in stack or something. So that whenever show is performed, all operations compiled come up.

→ shows the graph

random color

(scatters, labels, colors, overlap plots, etc.)

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

→ points are joined

*) both scatter and plot can be utilised to get better visualised data.

customizing plotted graphs.

```
plt.plot(x2, y2, 'bo--')
```

'bo--' → for three arguments

b → color

o → point plot

-- → dashed line

'bo--' → combination

String can be passed as the third argument to get customized graphs.

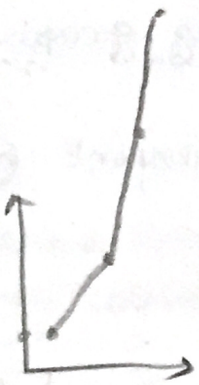
fixed characters:

- `--` : dashed line
- `o` : point plot
- `+` : plus plotted
- `p` : pentagon plotted

(44)

* plotting x^3

```
x = np.array([1,2,3,4])
y = x ** 3
plt.plot(x,y)
plt.show()
```



↳ this would not really represent a x^3 graph, because we are using only 4 points to connect

↓

we need to plot at and value to get better graph

↓

for that

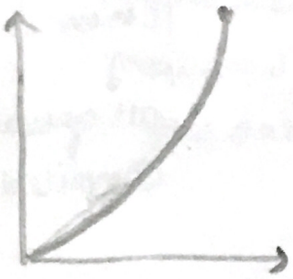
↓ similar to range

(*A) `x = np.arange([0, 5, 0.1])`

↓

0 to 4.9

stepsize = 0.1

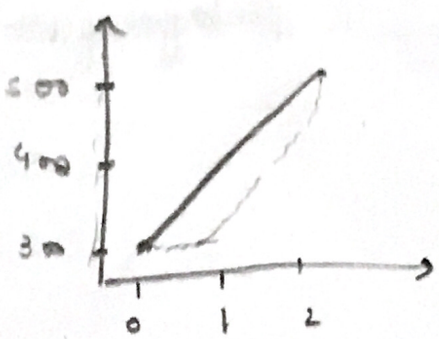


* `a = [3, 4, 5]`

```
plt.plot(a)
plt.show()
```

→ when we plot any array it is provided x values based on index and values are used or plotted on y axis.

(uses range to get corresponding x-axis coordinates)



* alternate method to choose color :

```
plt.plot(x, y, color = "black")
```

↓ color specified

```
plt.plot(x, y, color = "black", marker = "+")
```

↓ P ↓ O

```
plt.plot(x, y, color = "black", linewidth = 5)
```

* placing labels on axis, and title for graph.

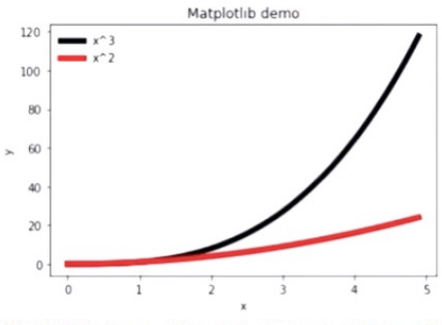
```
plt.ylabel("x^3")
plt.xlabel("x")
plt.title("Matplotlib demo")
plt.plot(x, y, color = "black", linewidth = 5, label = "x^3")
```

↓ need to call `plt.legend()`

before showing graph to print legend along with graph

↓ to recognise a particular line in a graph, when many are given

← label for the plot in graph, to recognise the different plots in a single graph.



* if we want to modify the axis scale, values

↓

plt.axis(0, 10, 0, 100)

↓

set of 4 values

↓

Starting and end points { first 2 for x-axis, other 2 for y-axis } equally distributed

* plt.grid() → to print grid for graph

* plt.text (2, 80, 'text', fontsize = 12)

↓ coordinates to display the text

↓ text to display

pie-graph

```
>>> sizes = [3, 4, 6, 2]
>>> plt.pie(sizes)
>>> plt.show()
```

plt.pie(sizes, colors = colors, labels = labels, autopct:)

↑ this needs to passed with more args, based on color and label list

→ will print an ellipse on screen, that might be undesirable i.e. unequal length of axes

↓

```
plt.axis("equal")
```

* colors can be played with,

```
colors = ["blue", "purple", "red", "pink"]
```

* labels to explain the graph

```
labels = ["A", "B", "C", "D"]
```

* title → `plt.title("split among classes")`

* we can also provide percentages using `(autopct = "%.2f")`

↓

we can send a function too here

↓

if % symbol needs to be added, then

```
autopct = "%.2f%%"
```

* Highlighting a part of the pie chart

↓

explode property

↓

explode = [0.1, 0, 0, 0] → extent of explode for ith part.

↓

plt.pie(sizes, colors = colors, explode = explode,

labels = labels, autopct = '%1.2f%%',

counterclock = False,

startangle = 100)