Matplotlib

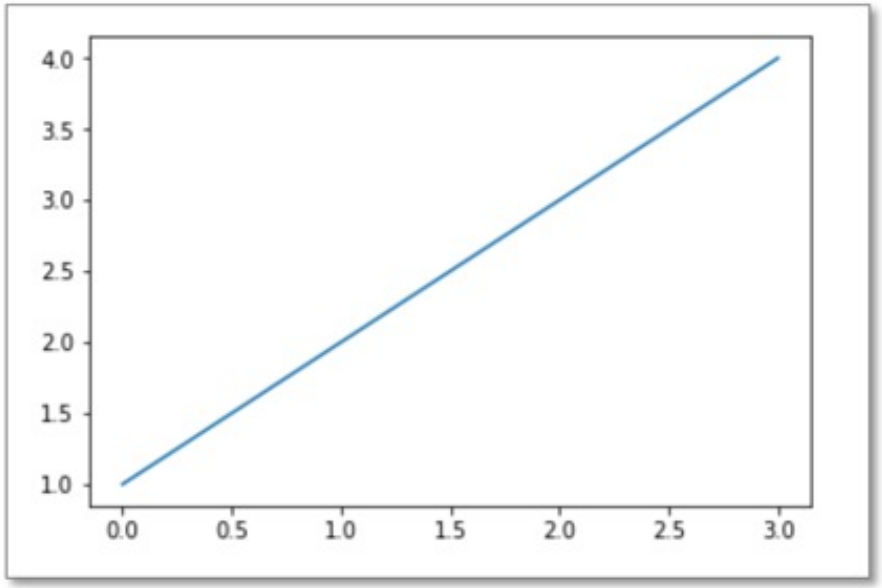
First Plot with Matplotlib

- Let us plot a simple graph on matplotlib

Code

```
import matplotlib.pyplot as plt
plt.plot([1, 2, 3, 4])
plt.show()
```

Plot



First Plot with Matplotlib

- We can specify the values for both axes

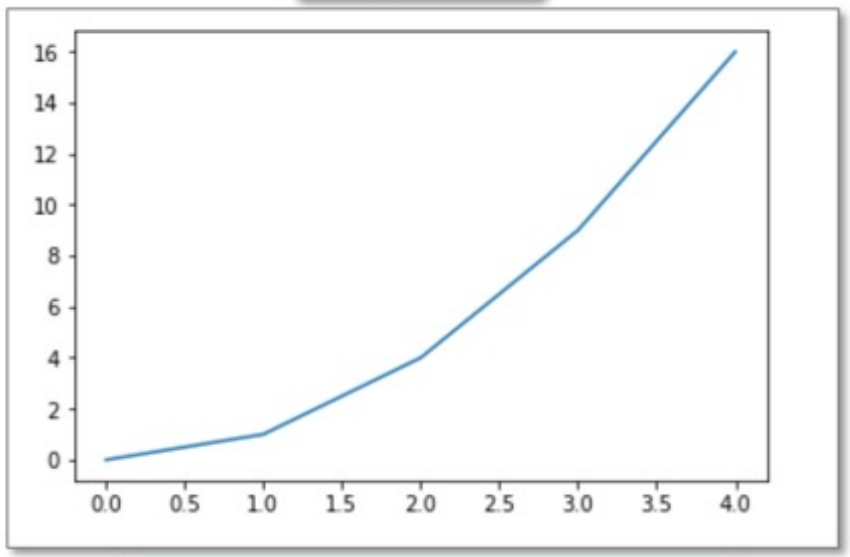
Code

```
import matplotlib.pyplot as plt
x =range(5)
plt.plot(x, [x1**2 for x1 in x])
plt.show()
```

Sequences of values for the x-axis

vertical co-ordinates of the points plotted: y = x^2

Plot



X-axis values specified – [0, 1, 2, 3, 4]

First Plot with Matplotlib

- We can use NumPy to specify the values for both axes with greater precision

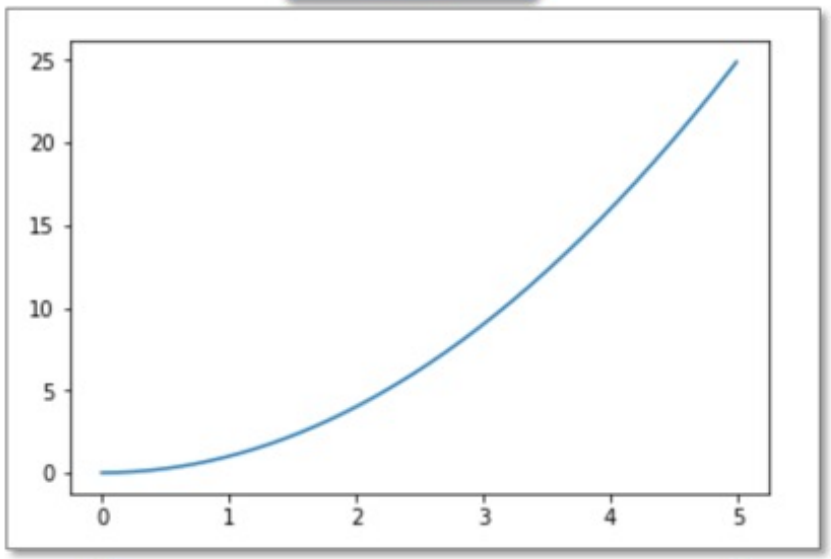
Code

```
import numpy, matplotlib.pyplot as plt
x = numpy.arange(0, 5, 0.01)
plt.plot(x, [x1**2 for x1 in x])
plt.show()
```

Sequences of values for the x-axis

vertical co-ordinates of the points plotted: y = x^2

Plot



X-axis values specified – [0, 1, 2, 3, 4]

Multiline Plots

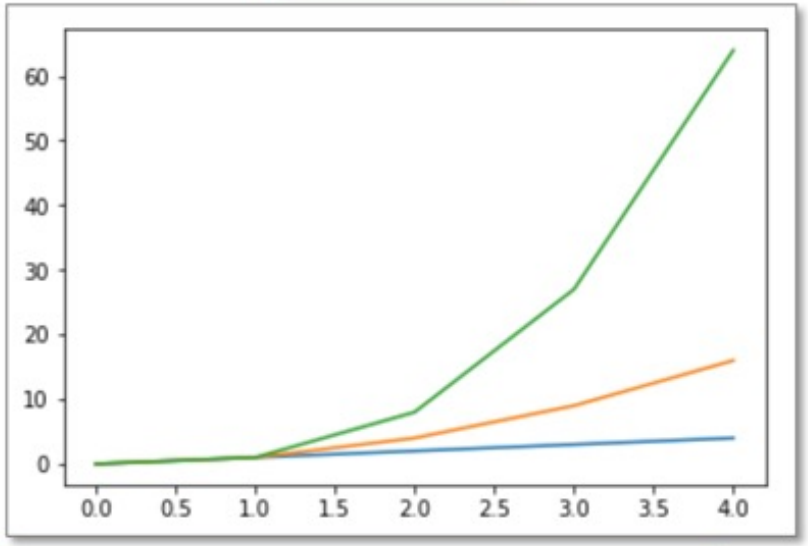
- Multiple functions can be drawn on the same plot

Code

```
import matplotlib.pyplot as plt
x =range(5)
plt.plot(x, [x1 for x1 in x])
plt.plot(x, [x1*x1 for x1 in x])
plt.plot(x, [x1*x1*x1 for x1 in x])
plt.show()
```

Three lines are plotted

Plot



Different colours are used for different lines

Multiline Plots

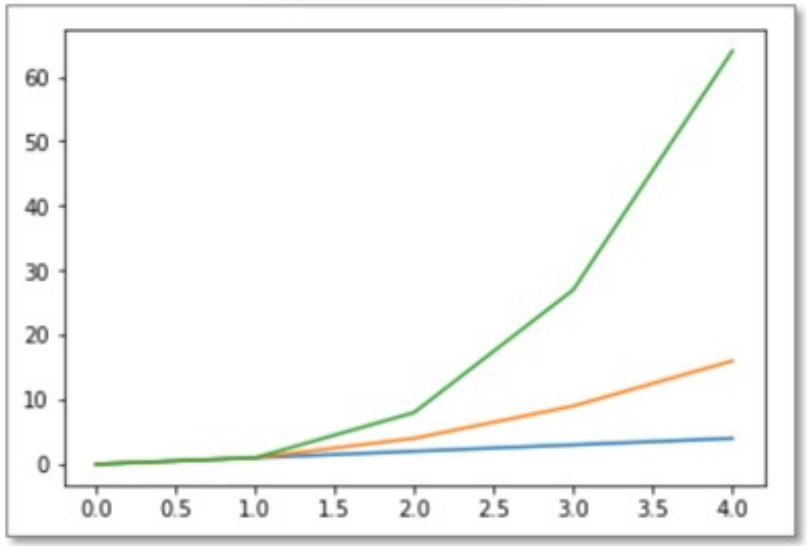
- Multiple functions can be drawn on the same plot

Code

```
import matplotlib.pyplot as plt
x =range(5)
plt.plot(x, [x1 for x1 in x], x,
         [x1*x1 for x1 in x], x,
[x1*x1*x1 for x1 in x])
plt.show()
```

Plot multiple figures using a single plot() function

Plot



Different colours are used for different lines

Adding A Grid

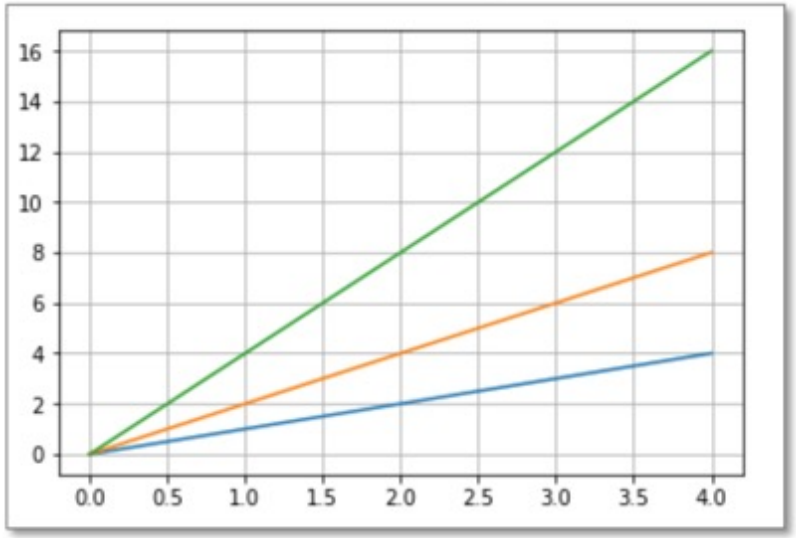
The grid() function adds a grid to the plot

Code

```
import matplotlib.pyplot as plt
x =range(5)
plt.plot(x, [x1 for x1 in x], x,
          [x1*2 for x1 in x],
         x, [x1*4 for x1 in x])
plt.grid(True)
plt.show()
```

grid() takes a single Boolean parameter

Plot



Grid appears in the background of the plot

Limiting The Axes

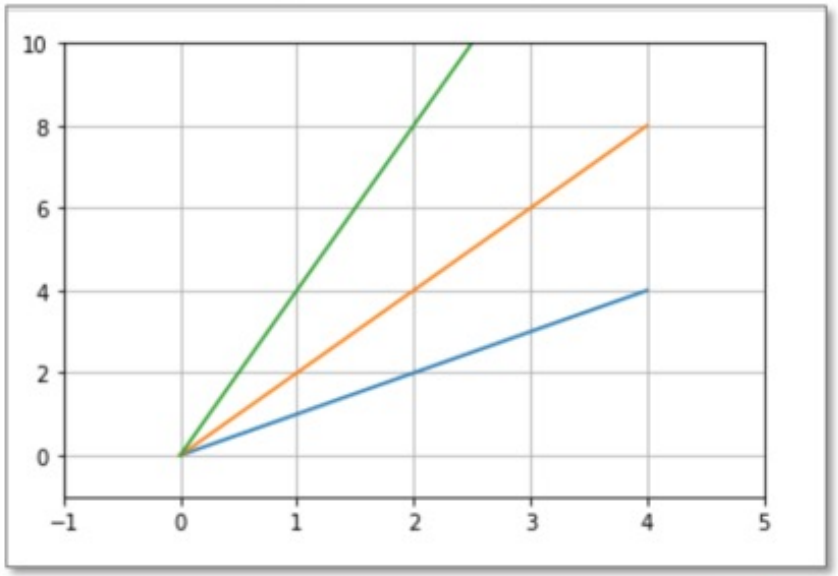
The scale of the plot can be set using axis()

Code

```
import matplotlib.pyplot as plt
x =range(5)
plt.plot(x, [x1 for x1 in x], x, [x1*2
for x1 in x], x, [x1*4 for x1 in x])
plt.grid(True)
plt.axis([-1, 5, -1, 10])
plt.show()
```

Sets new axes limits

Plot



Plot with the new boundaries of the axes

Limiting The Axes

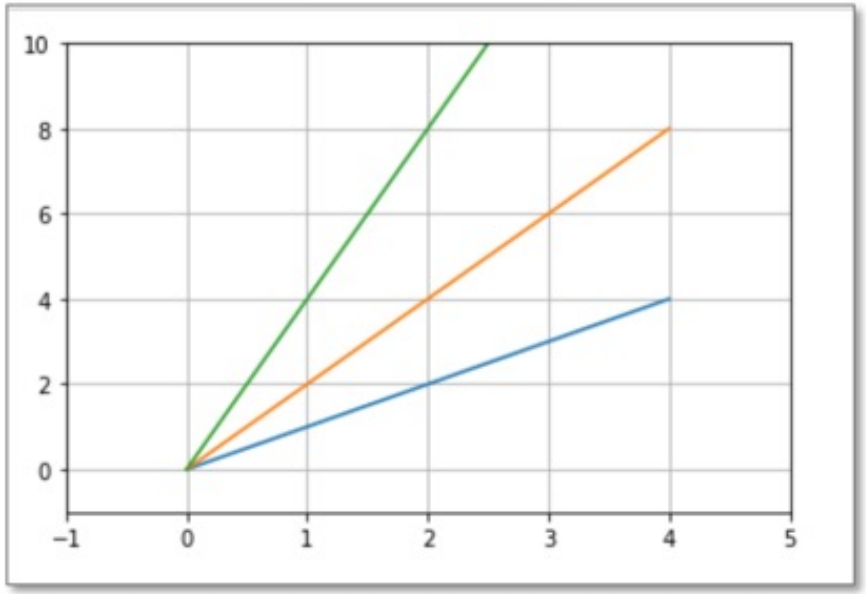
The scale of the plot can also be set using xlim() and ylim()

Code

```
import matplotlib.pyplot as plt
x =range(5)
plt.plot(x, [x1 for x1 in x], x, [x1*2
for x1 in x], x, [x1*4 for x1 in x])
plt.grid(True)
plt.xlim(-1, 5)
plt.ylim(-1, 10)
plt.show()
```

Sets new axes limits

Plot



Plot with the new boundaries of the axes

Adding Labels

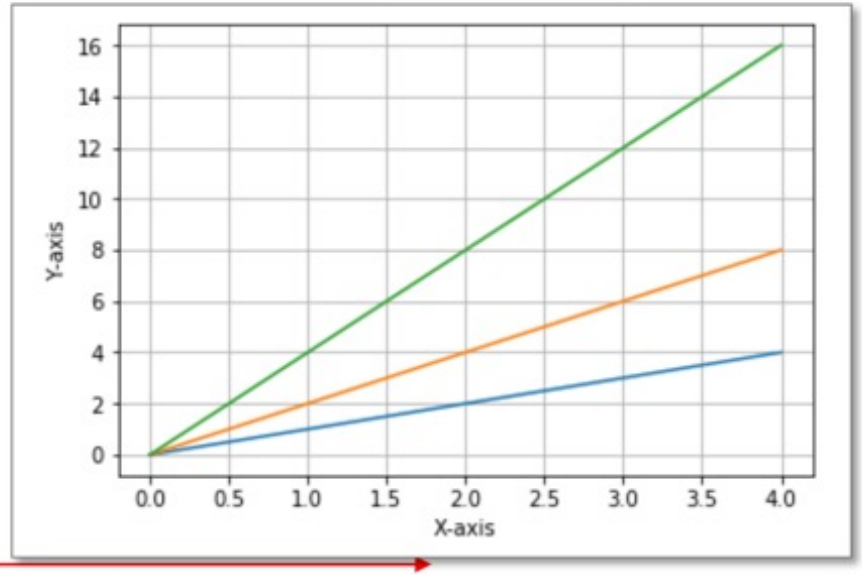
Labels can be added to the axes of the plot

Code

```
import matplotlib.pyplot as plt
x =range(5)
plt.plot(x, [x1 for x1 in x], x, [x1*2
for x1 in x], x, [x1*4 for x1 in x])
plt.grid(True)
plt.xlabel('X-axis')
plt.ylabel('Y-axis')
plt.show()
```

Sets axes labels

Plot



Plot with labels for the axes

Adding The Title

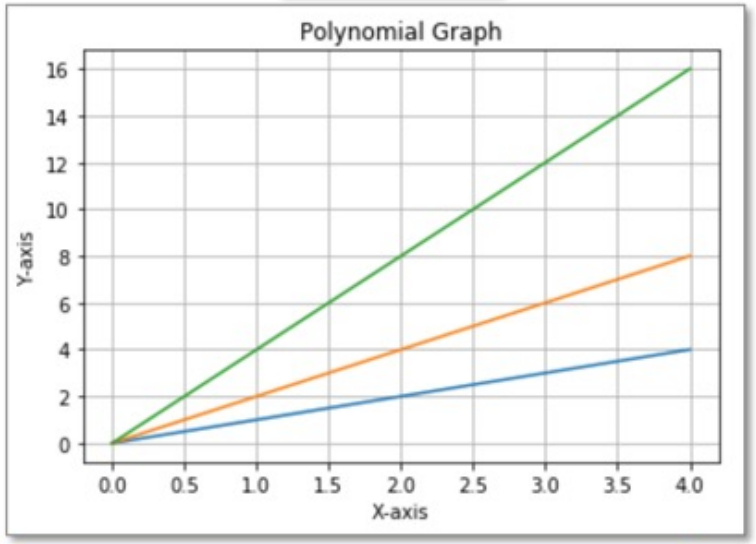
The title defines the data plotted on the graph

Code

```
import matplotlib.pyplot as plt
x =range(5)
plt.plot(x, [x1 for x1 in x], x, [x1*2
for x1 in x], x, [x1*4 for x1 in x])
plt.grid(True)
plt.xlabel('X-axis')
plt.ylabel('Y-axis')
plt.title('Polynomial Graph')
plt.show()
```

Pass the title as a parameter to title()

Plot



Plot with a title

Adding A Legend

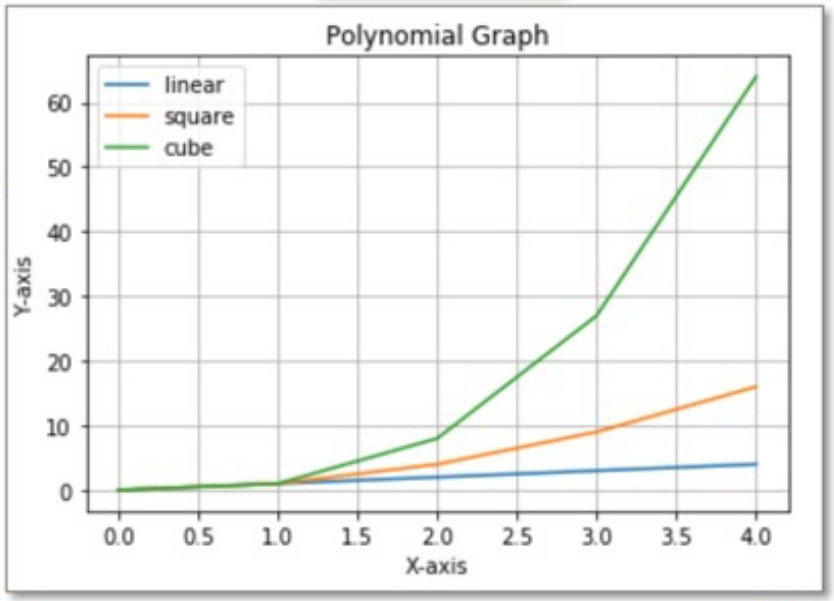
Legends explain the meaning of each line in the graph

Code

```
import numpy, matplotlib.pyplot as plt
x = numpy.arange(5)
plt.plot(x, x, label='linear')
plt.plot(x, x*x, label='square')
plt.plot(x, x*x*x, label='cube')
plt.grid(True)
plt.xlabel('X-axis')
plt.ylabel('Y-axis')
plt.title('Polynomial Graph')
plt.legend()
plt.show()
```

legend() displays the legend on the plot

Plot



Plot with a title and legend

Saving Plots

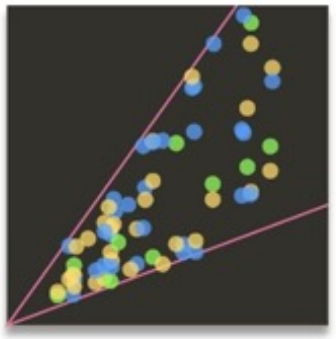
Plots can be saved using savefig()

```
import numpy, matplotlib.pyplot as plt

x = numpy.arange(5)
plt.plot(x, x, label='linear')
plt.plot(x, x*x, label='square')
plt.plot(x, x*x*x, label='cube')

plt.grid(True)
plt.xlabel('X-axis')
plt.ylabel('Y-axis')
plt.title('Polynomial Graph')
plt.legend()
plt.savefig('plot.png')
plt.show()
```

Saves an image named 'plot.png' in the current directory

Types Of Plots

Matplotlib provides many types of plot formats for visualising information



Scatter Plot

Histogram

Plot Formats

Bar Graph

Pie Chart

Histogram

Histograms display the distribution of a variable over a range of frequencies or values

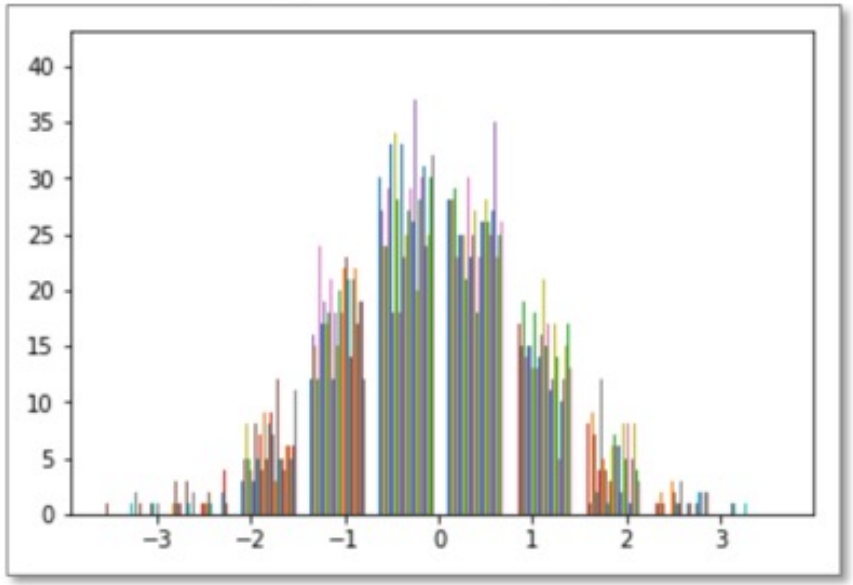
Code

```
import matplotlib.pyplot as plt, numpy

y = numpy.random.randn(100, 100)
plt.hist(y)
plt.show()
```

Function to plot the histogram takes the dataset as the parameter

100x100 array of a Gaussian distribution

Plot



Histogram

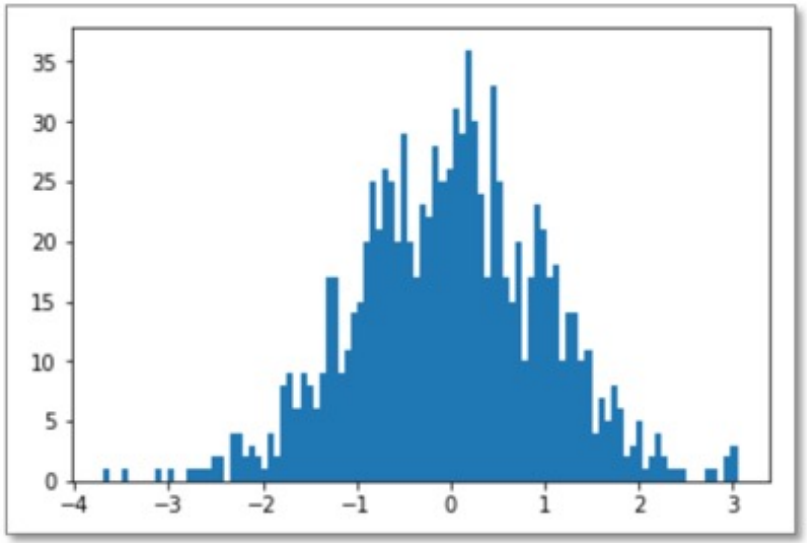
Histogram groups values into non-overlapping categories called bins

Code

```
import matplotlib.pyplot as plt, numpy

y = numpy.random.randn(1000)
plt.hist(y, 100)
plt.show()
```

The second parameter sets the bin value

Plot



Default bin value of the histogram plot is 10

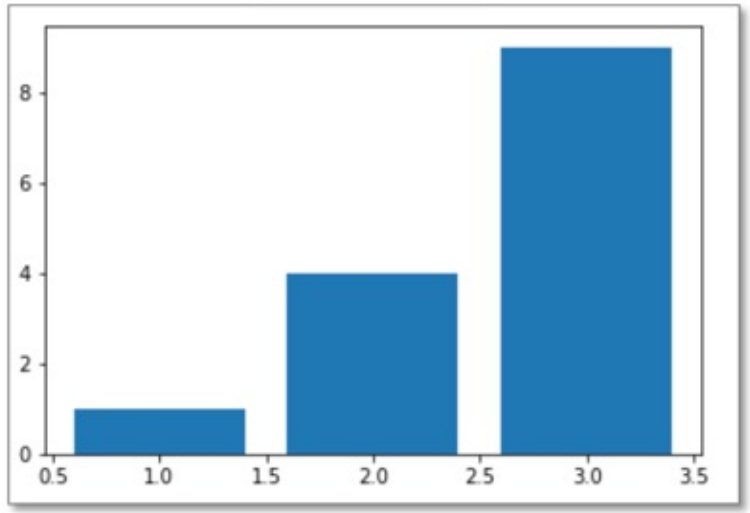
Bar Chart

Bar charts are used to visually compare two or more values using rectangular bars

Code

```
import matplotlib.pyplot as plt

plt.bar([1, 2, 3], [1, 4, 9])
plt.show()
```

Plot



Mid-point of the lower face of every bar

Heights of the successive bars in the plot

Default width of each bar is 0.8 units

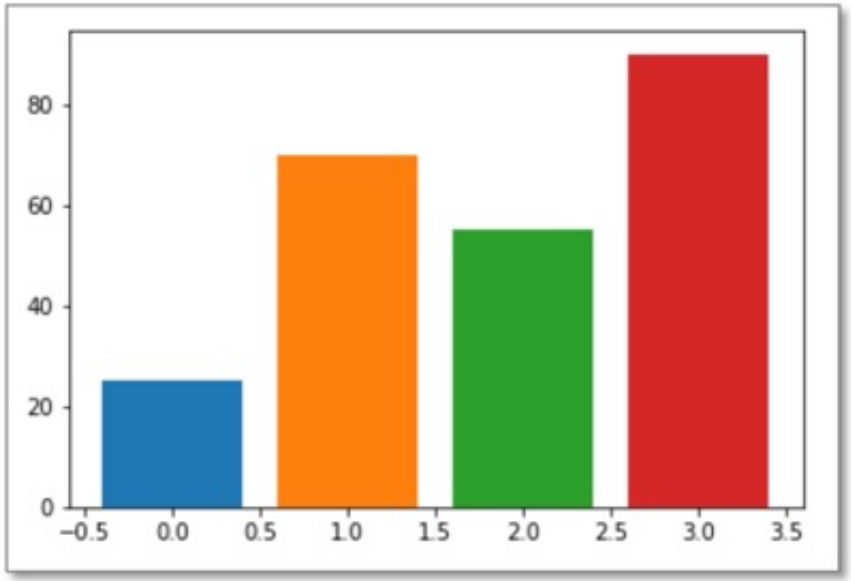
Plotting A Dictionary Using Bar Chart

Code

```
import matplotlib.pyplot as plt

dictionary = {'A':25, 'B':70, 'C':55,
'D': 90}
for i, key in enumerate(dictionary):
    plt.bar(i, dictionary[key])

plt.show()
```

Each key-value pair is plotted individually as dictionaries are not iterable

Plot



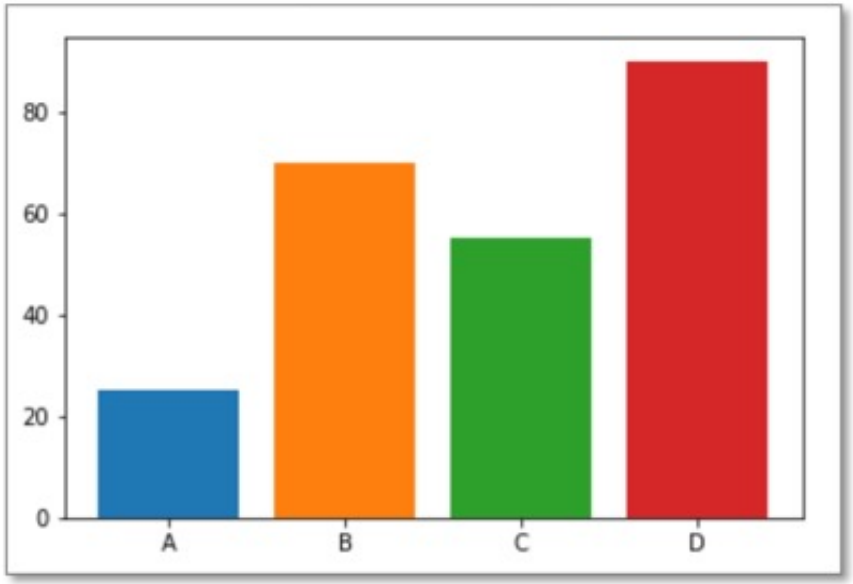
Plotting A Dictionary Using Bar Chart

Code

```
import matplotlib.pyplot as plt, numpy

dictionary = {'A':25, 'B':70, 'C':55, 'D':
90}
for i, key in enumerate(dictionary):
    plt.bar(i, dictionary[key])
plt.xticks(numpy.arange(len(dictionary)),
dictionary.keys())
plt.show()
```

Adds the keys as labels on the x-axis

Plot



Pie Chart

Pie charts are used to compare multiple parts against the whole

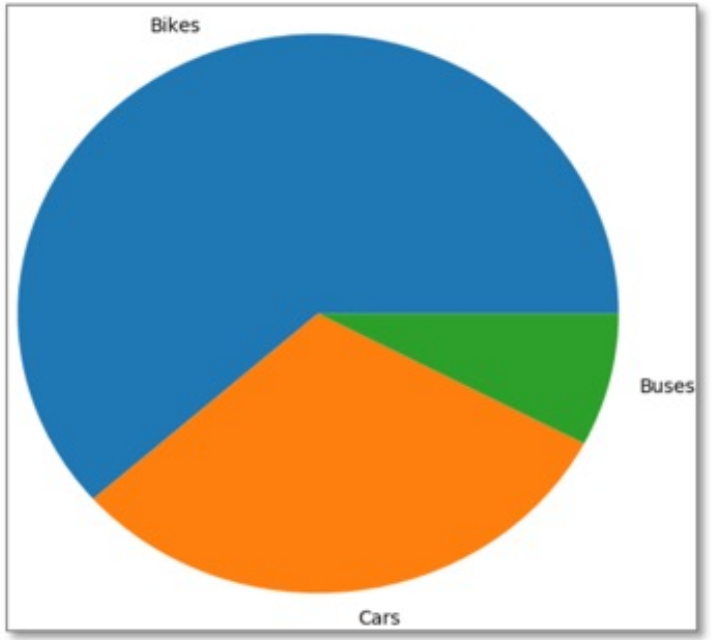
Code

```
import matplotlib.pyplot as plt
plt.figure(figsize=(3,3))
x = [40, 20, 5]
labels = ['Bikes', 'Cars', 'Buses']
plt.pie(x, labels=labels)
plt.show()
```

Proportions of the sectors

Size of the plot in inches

Plot



Scatter Plot

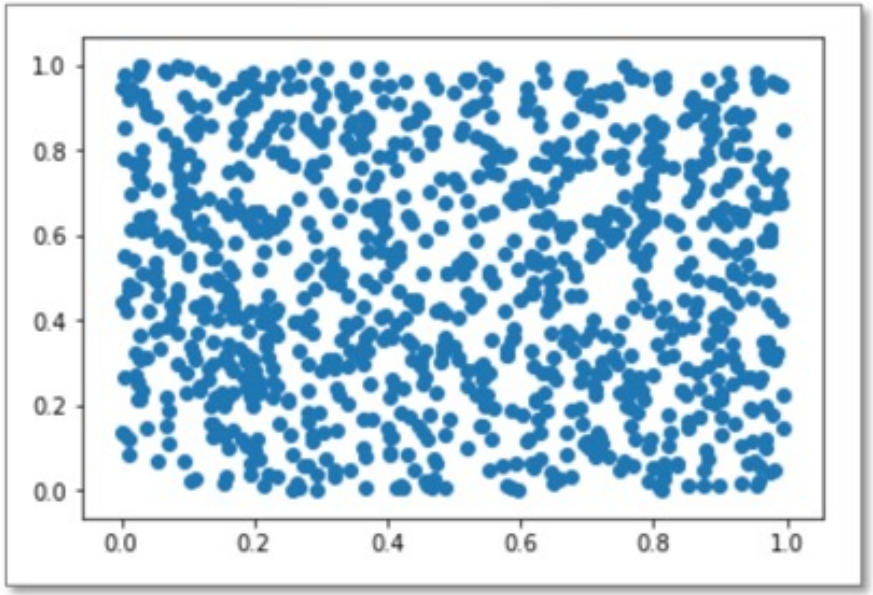
Scatter plots display values for two sets of data, visualised as a collection of points

Code

```
import matplotlib.pyplot as plt
import numpy as np
x = np.random.rand(1000)
y = np.random.rand(1000)
plt.scatter(x, y)
plt.show()
```

Two Gaussian distributions plotted

Plot



Control Colors

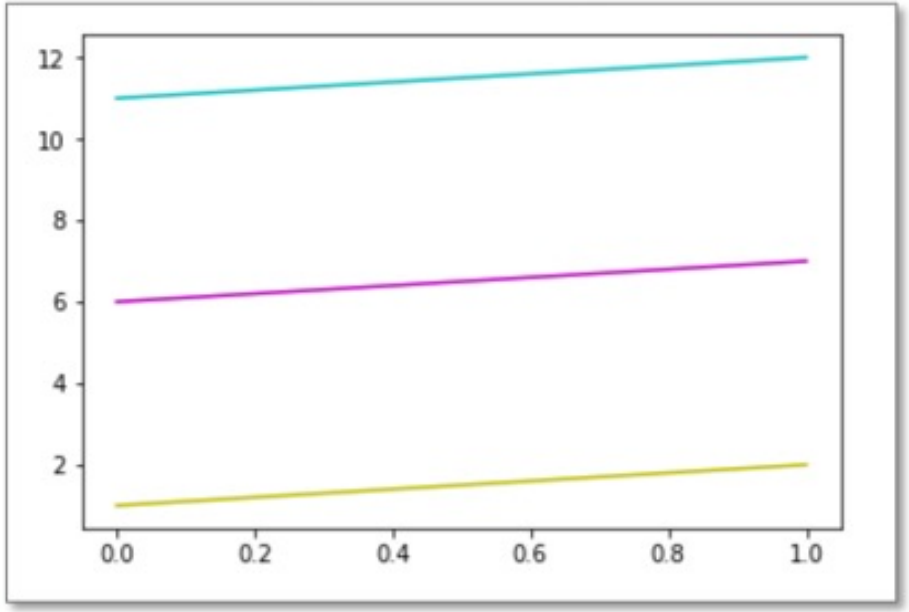
Matplotlib allows to choose custom colours for plots

Code

```
import matplotlib.pyplot as plt
import numpy as np
y = np.arange(1, 3)
plt.plot(y, 'y')
plt.plot(y+5, 'm')
plt.plot(y+10, 'c')
plt.show()
```

Specifying line colours

Plot



Control Colors - Codes

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

Control Line Styling

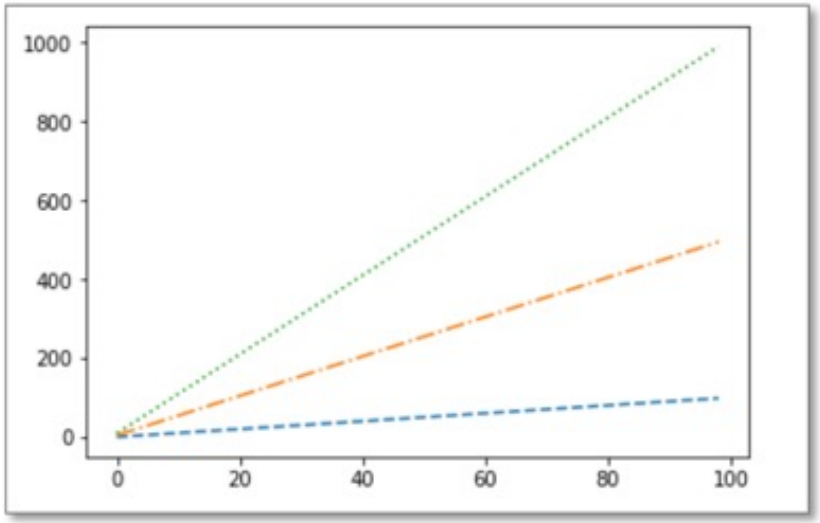
Matplotlib allows different line styles for plots

Code

```
import matplotlib.pyplot as plt
import numpy as np

y = np.arange(1, 100)
plt.plot(y, '--', y*5, '-.', y*10, ':')
plt.show()
```

Specifying line styling

Plot

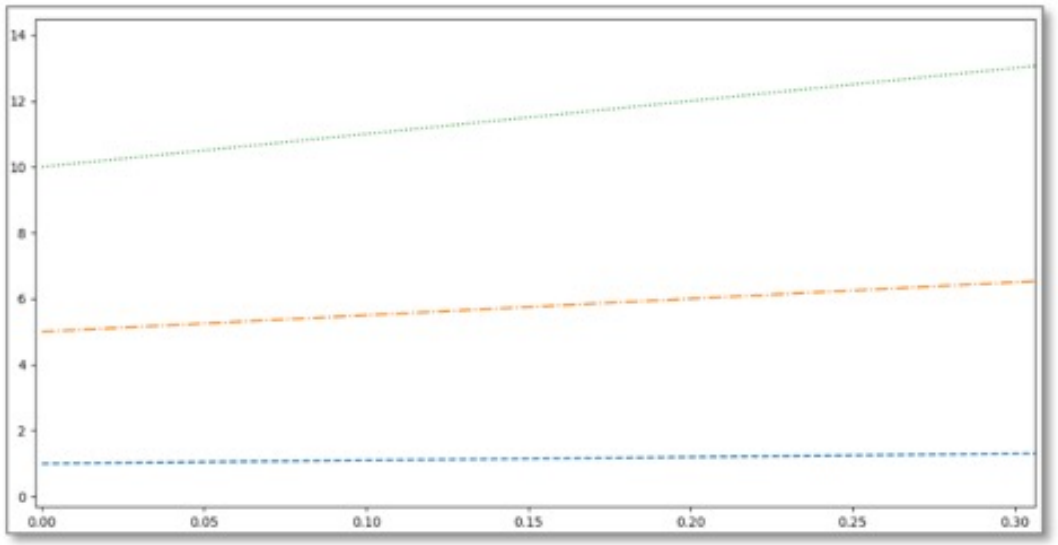


Control Line Styling

Matplotlib allows different line styles for plots

Plot

Style	Style Name
-	Solid line
--	Dashed line
-.	Dash-Dot line
:	Dotted Line



Control Marker Styling

Matplotlib provides customization options for markers

Code

```
import matplotlib.pyplot as plt
import numpy as np

y = np.arange(1, 3, 0.2)
plt.plot(y, '*', y+0.5, 'o', y+1, 'D',
y+2, '^', y+3, 's')
plt.show()
```

Specifying line styling

Plot