

National University of Computer and Emerging Sciences



Lab Manual
CL461-Artificial Intelligence Lab

Department of Computer Science
FAST-NU, Lahore, Pakistan

Table of Contents

1	Objectives	3
2	Task Distribution	3
3	Matplotlib	3
3.1	Installation of Matplotlib	3
3.2	Pyplot	4
	Example	4
3.3	Plotting x and y points	5
	Example	5
3.4	Plotting Without Line	6
3.5	Multiple Points	6
3.6	Default X-Points	7
3.7	Create Labels and title for a Plot	8
3.8	Add Grid Lines to a Plot	8
3.9	Display Multiple Plots	9
3.10	Creating Scatter Plots	10
3.11	Creating Bars	11
3.12	Histogram	12
3.12.1	Create Histogram	13
3.13	Creating Pie Charts	14
3.14	Box Plot	15

1 Objectives

After performing this lab, students shall be able to understand the following Python concepts and applications:

- ✓ Matplotlib Introduction
- ✓ Application of Matplotlib
- ✓ Min-Max Algorithm (Game theory)

2 Task Distribution

Total Time	170 Minutes
Matplotlib Introduction/Installation Guide	20 Minutes
Application of Matplotlib	25 Minutes
Min-Max Algorithm	25 Minutes
Exercise	90 Minutes
Online Submission	10 Minutes

3 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.

3.1 Installation of Matplotlib

If you have Python and PIP already installed on a system, then installation of Matplotlib is very easy.

Install it using this command:

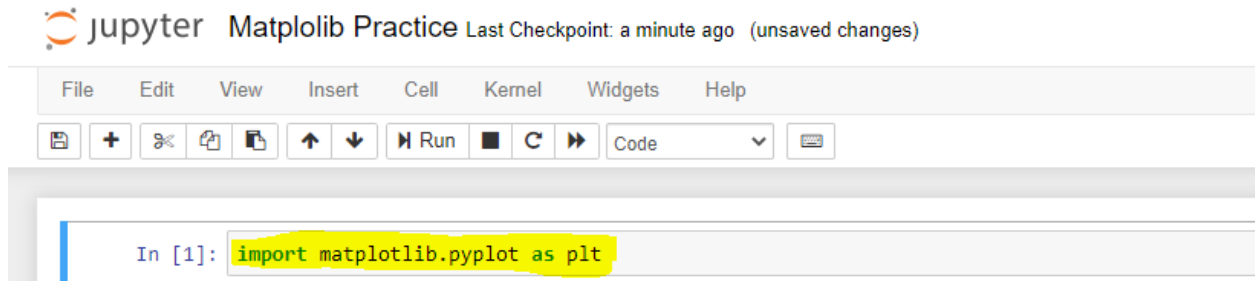
```
C:\Users\Your Name>pip install matplotlib
```

pip install matplotlib

But mostly distribution like Anaconda, Spyder has pre-installed matplotlib.

3.2 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):

Code	Output
<pre>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()</pre>	<p>The output is a line plot with a blue line. The x-axis is labeled from 0 to 6, and the y-axis is labeled from 0 to 250. The line starts at the origin (0,0) and extends to the point (6,250), representing a linear function.</p>

3.3 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.

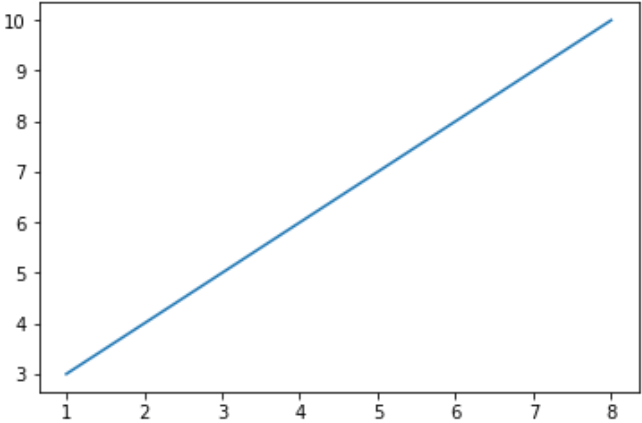
Parameter 1 is an array containing the points on the **x-axis**.

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):

Code	Output
<pre>import matplotlib.pyplot as plt import numpy as np xpoints = np.array([1, 8]) ypoints = np.array([3, 10]) plt.plot(xpoints, ypoints) plt.show()</pre>	 <p>The output is a line plot with a blue line connecting the points (1, 3) and (8, 10). The x-axis is labeled from 1 to 8, and the y-axis is labeled from 3 to 10.</p>

There are many types of single lines/multiple lines that can be drawn, explore other types at:

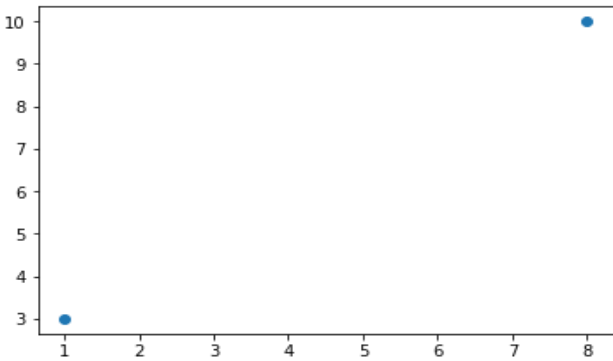
https://www.w3schools.com/python/matplotlib_line.asp

3.4 Plotting Without Line

To plot only the markers, you can use *shortcut string notation* parameter 'o', which means 'rings'.

Example

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

Code	Output
<pre>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()</pre>	

There can be different type of markers, you can explore at:

https://www.w3schools.com/python/matplotlib_markers.asp

3.5 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

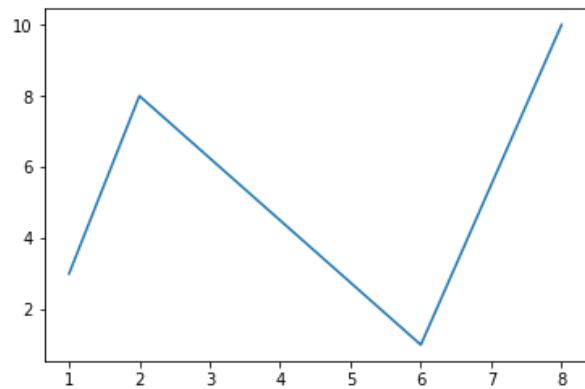
Draw a line in a diagram from position (1, 3) to (2, 8) then to (6, 1) and finally to position (8, 10):

Code	Output
------	--------

```
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()
```



3.6 Default X-Points

If we do not specify the points in 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:

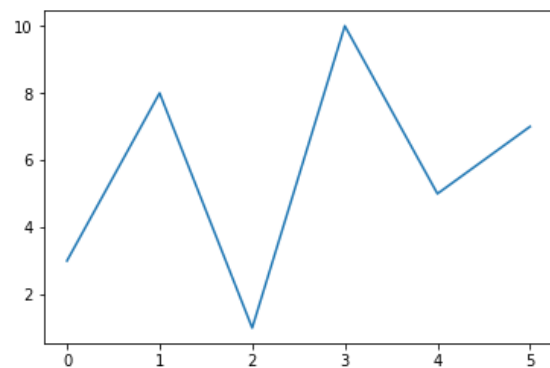
Example

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()
```



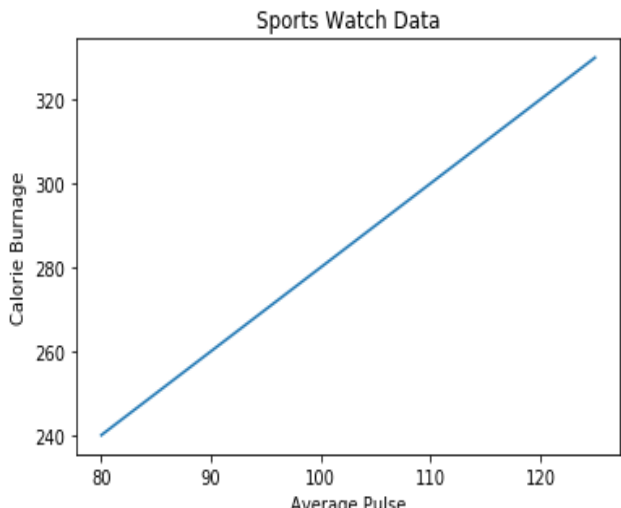
The x-points in the example above are [0, 1, 2, 3, 4, 5] by default.

3.7 Create Labels and title 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:

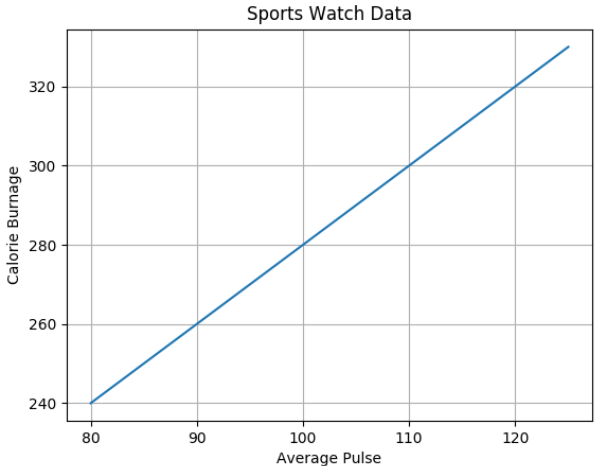
Code	Output																						
<pre>import numpy as np import matplotlib.pyplot as plt x = np.array([80, 85, 90, 95, 100,105, 110, 115, 120, 125]) y = np.array([240, 250, 260, 270, 280, 290, 300, 310, 320, 330]) plt.plot(x, y) plt.title("Sports Watch Data") plt.xlabel("Average Pulse") plt.ylabel("Calorie Burnage") plt.show()</pre>	 <table border="1"><caption>Data points from the Sports Watch Data plot</caption><thead><tr><th>Average Pulse</th><th>Calorie Burnage</th></tr></thead><tbody><tr><td>80</td><td>240</td></tr><tr><td>85</td><td>250</td></tr><tr><td>90</td><td>260</td></tr><tr><td>95</td><td>270</td></tr><tr><td>100</td><td>280</td></tr><tr><td>105</td><td>290</td></tr><tr><td>110</td><td>300</td></tr><tr><td>115</td><td>310</td></tr><tr><td>120</td><td>320</td></tr><tr><td>125</td><td>330</td></tr></tbody></table>	Average Pulse	Calorie Burnage	80	240	85	250	90	260	95	270	100	280	105	290	110	300	115	310	120	320	125	330
Average Pulse	Calorie Burnage																						
80	240																						
85	250																						
90	260																						
95	270																						
100	280																						
105	290																						
110	300																						
115	310																						
120	320																						
125	330																						

3.8 Add Grid Lines to a Plot

With Pyplot, you can use the `grid()` function to add grid lines to the plot.

Example

Add grid lines to the plot:

Code	Output																						
<pre>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.title("Sports Watch Data") plt.xlabel("Average Pulse") plt.ylabel("Calorie Burnage") plt.plot(x, y) plt.grid() plt.show()</pre>	 <p>The plot displays a linear relationship between Average Pulse and Calorie Burnage. The data points are as follows:</p> <table border="1"> <thead> <tr> <th>Average Pulse</th> <th>Calorie Burnage</th> </tr> </thead> <tbody> <tr><td>80</td><td>240</td></tr> <tr><td>85</td><td>250</td></tr> <tr><td>90</td><td>260</td></tr> <tr><td>95</td><td>270</td></tr> <tr><td>100</td><td>280</td></tr> <tr><td>105</td><td>290</td></tr> <tr><td>110</td><td>300</td></tr> <tr><td>115</td><td>310</td></tr> <tr><td>120</td><td>320</td></tr> <tr><td>125</td><td>330</td></tr> </tbody> </table>	Average Pulse	Calorie Burnage	80	240	85	250	90	260	95	270	100	280	105	290	110	300	115	310	120	320	125	330
Average Pulse	Calorie Burnage																						
80	240																						
85	250																						
90	260																						
95	270																						
100	280																						
105	290																						
110	300																						
115	310																						
120	320																						
125	330																						

Different type of grid can be generated, for more details see:

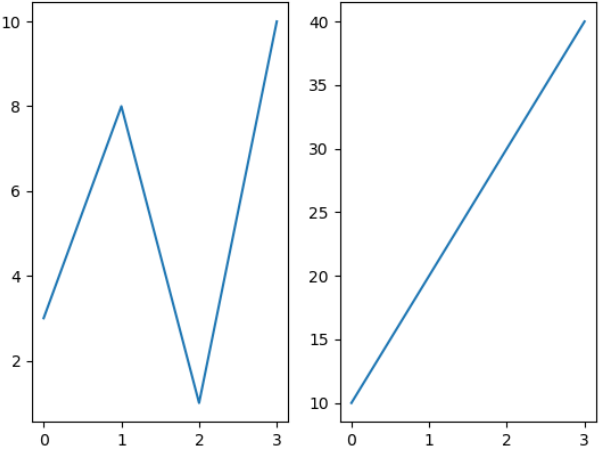
https://www.w3schools.com/python/matplotlib_grid.asp

3.9 Display Multiple Plots

With the subplots() function you can draw multiple plots in one figure:

Example

Draw 2 plots:

Code	Output
<pre>import matplotlib.pyplot as plt import numpy as np #plot 1: x = np.array([0, 1, 2, 3]) y = np.array([3, 8, 1, 10]) plt.subplot(1, 2, 1) plt.plot(x,y) #plot 2: x = np.array([0, 1, 2, 3]) y = np.array([10, 20, 30, 40]) plt.subplot(1, 2, 2) plt.plot(x,y) plt.show()</pre>	 <p>The output displays two side-by-side line plots. The left plot has an x-axis from 0 to 3 and a y-axis from 0 to 10. It shows a blue line connecting the points (0, 3), (1, 8), (2, 1), and (3, 10). The right plot has an x-axis from 0 to 3 and a y-axis from 0 to 40. It shows a blue line connecting the points (0, 10), (1, 20), (2, 30), and (3, 40).</p>

There different ways to plot multiple plots:

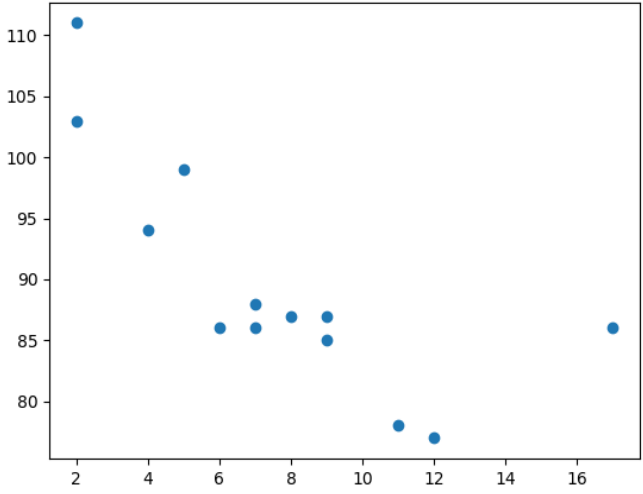
https://www.w3schools.com/python/matplotlib_subplots.asp

3.10 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:

Code	Output																												
<pre>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()</pre>	 <table border="1"> <caption>Data points from the scatter plot</caption> <thead> <tr> <th>Car Age (X)</th> <th>Speed (Y)</th> </tr> </thead> <tbody> <tr><td>2</td><td>111</td></tr> <tr><td>2</td><td>103</td></tr> <tr><td>4</td><td>94</td></tr> <tr><td>5</td><td>99</td></tr> <tr><td>6</td><td>86</td></tr> <tr><td>7</td><td>86</td></tr> <tr><td>7</td><td>87</td></tr> <tr><td>8</td><td>87</td></tr> <tr><td>9</td><td>85</td></tr> <tr><td>9</td><td>87</td></tr> <tr><td>11</td><td>78</td></tr> <tr><td>12</td><td>77</td></tr> <tr><td>17</td><td>86</td></tr> </tbody> </table>	Car Age (X)	Speed (Y)	2	111	2	103	4	94	5	99	6	86	7	86	7	87	8	87	9	85	9	87	11	78	12	77	17	86
Car Age (X)	Speed (Y)																												
2	111																												
2	103																												
4	94																												
5	99																												
6	86																												
7	86																												
7	87																												
8	87																												
9	85																												
9	87																												
11	78																												
12	77																												
17	86																												

Explanation of above plot:

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.

There are different type of scatter graphs that can be created (kindly see the link given, as all examples will make the manual lengthy):

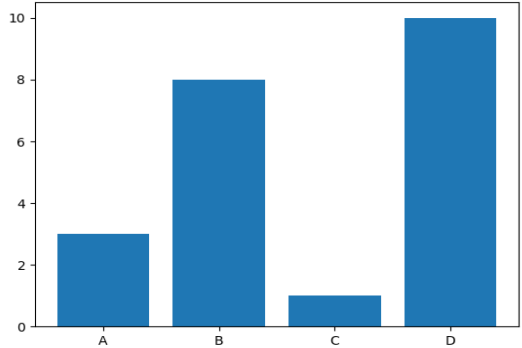
https://www.w3schools.com/python/matplotlib_scatter.asp

3.11 Creating Bars

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

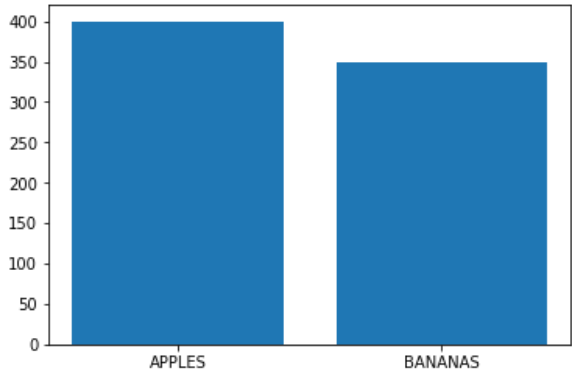
Example

Draw 4 bars:

Code	Output										
<pre>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()</pre>	 <table border="1"> <thead> <tr> <th>Category</th> <th>Value</th> </tr> </thead> <tbody> <tr> <td>A</td> <td>3</td> </tr> <tr> <td>B</td> <td>8</td> </tr> <tr> <td>C</td> <td>1</td> </tr> <tr> <td>D</td> <td>10</td> </tr> </tbody> </table>	Category	Value	A	3	B	8	C	1	D	10
Category	Value										
A	3										
B	8										
C	1										
D	10										

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.

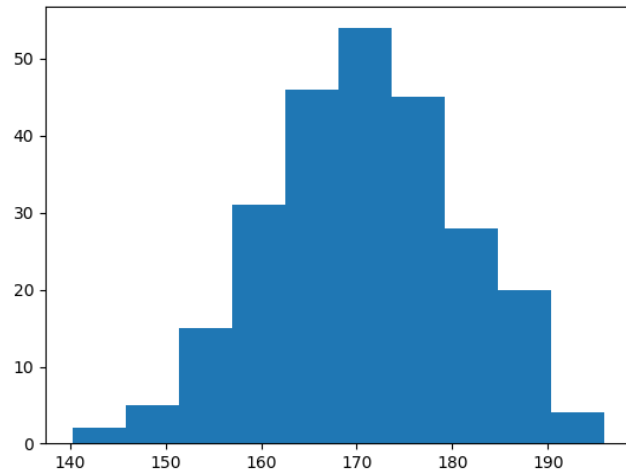
<pre>import matplotlib.pyplot as plt x = ["APPLES", "BANANAS"] y = [400, 350] plt.bar(x, y)</pre>	 <table border="1"> <thead> <tr> <th>Category</th> <th>Value</th> </tr> </thead> <tbody> <tr> <td>APPLES</td> <td>400</td> </tr> <tr> <td>BANANAS</td> <td>350</td> </tr> </tbody> </table>	Category	Value	APPLES	400	BANANAS	350
Category	Value						
APPLES	400						
BANANAS	350						

3.12 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:

You can read from the histogram that there are approximately:

2 people from 140 to 145cm
 5 people from 145 to 150cm
 15 people from 151 to 156cm
 31 people from 157 to 162cm
 46 people from 163 to 168cm
 53 people from 168 to 173cm
 45 people from 173 to 178cm
 28 people from 179 to 184cm
 21 people from 185 to 190cm
 4 people from 190 to 195cm



3.12.1 Create Histogram

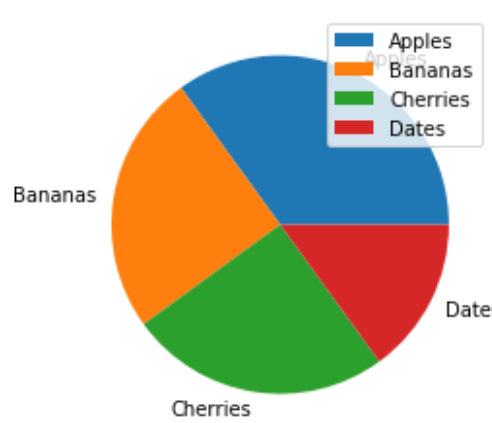
In Matplotlib, we use the `hist()` function to create histograms.

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.

Code	Output																						
<pre>import matplotlib.pyplot as plt import numpy as np x = np.random.normal(170, 10, 250) plt.hist(x) plt.show()</pre>	<table border="1"> <caption>Histogram Data (Approximate)</caption> <thead> <tr> <th>Height Range (cm)</th> <th>Frequency</th> </tr> </thead> <tbody> <tr><td>140-145</td><td>2</td></tr> <tr><td>145-150</td><td>5</td></tr> <tr><td>151-156</td><td>15</td></tr> <tr><td>157-162</td><td>31</td></tr> <tr><td>163-168</td><td>46</td></tr> <tr><td>168-173</td><td>53</td></tr> <tr><td>173-178</td><td>45</td></tr> <tr><td>179-184</td><td>28</td></tr> <tr><td>185-190</td><td>21</td></tr> <tr><td>190-195</td><td>4</td></tr> </tbody> </table>	Height Range (cm)	Frequency	140-145	2	145-150	5	151-156	15	157-162	31	163-168	46	168-173	53	173-178	45	179-184	28	185-190	21	190-195	4
Height Range (cm)	Frequency																						
140-145	2																						
145-150	5																						
151-156	15																						
157-162	31																						
163-168	46																						
168-173	53																						
173-178	45																						
179-184	28																						
185-190	21																						
190-195	4																						

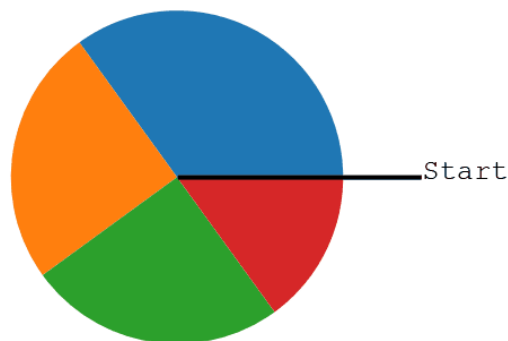
3.13 Creating Pie Charts

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

Code	Output
<pre>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.legend() plt.show()</pre>	

As you can see the pie chart draws one piece (called a wedge) for each value in the array (in this case [35, 25, 25, 15]).

By default, the plotting of the first wedge starts from the x-axis and move *counterclockwise*:

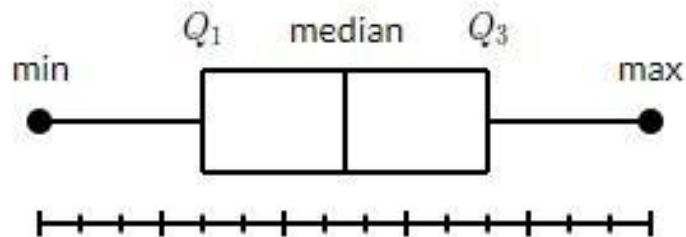


Note: The size of each wedge is determined by comparing the value with all the other values, by using this formula:

The value divided by the sum of all values: $x/\text{sum}(x)$

3.14 Box Plot

A box plot which is also known as a whisker plot displays a summary of a set of data containing the minimum, first quartile, median, third quartile, and maximum. In a box plot, we draw a box from the first quartile to the third quartile. A vertical line goes through the box at the median. The whiskers go from each quartile to the minimum or maximum.



The image is taken from: https://www.tutorialspoint.com/matplotlib/matplotlib_box_plot.htm

Example 1: Draw a box-and-whisker plot for the data set {3, 7, 8, 5, 12, 14, 21, 13, 18}.

Minimum: 3, Q_1 : 6, Median: 12, Q_3 : 16, and Maximum: 21.

Code	Output
<pre>import matplotlib.pyplot as plt data = [3, 7, 8, 5, 12, 14, 21, 13, 18] plt.boxplot(data) plt.show()</pre>	