National University of Computer and Emerging Sciences

Lab Manual 06
AL2002-Artificial Intelligence Lab

Course Instructor	Ms. Tehreem Yasir
Lab Instructor (s)	Rida Mahmood Hamza Ayub
Section	F
Semester	Spring 2022

Department of Computer Science
FAST-NU, Lahore, Pakistan

Table of Contents

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:

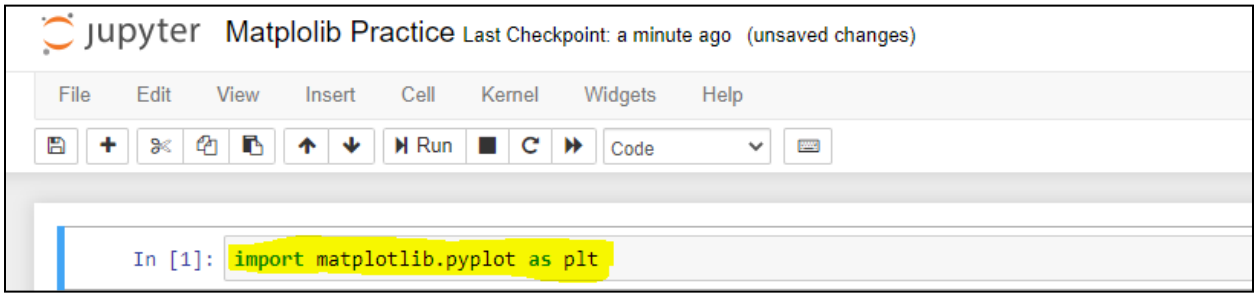
```
C:\Users\Your Name>pip install matplotlib
```

pip install matplotlib

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

3.2 Pyplot

Most of the Matplotlib utilities lies under the pyplot submodule, and are usually imported under the plt alias:

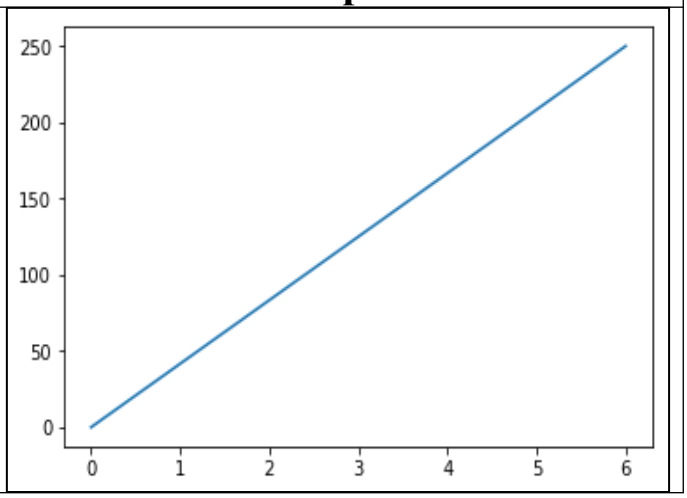



```
import matplotlib.pyplot as plt
```

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
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()	

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**.

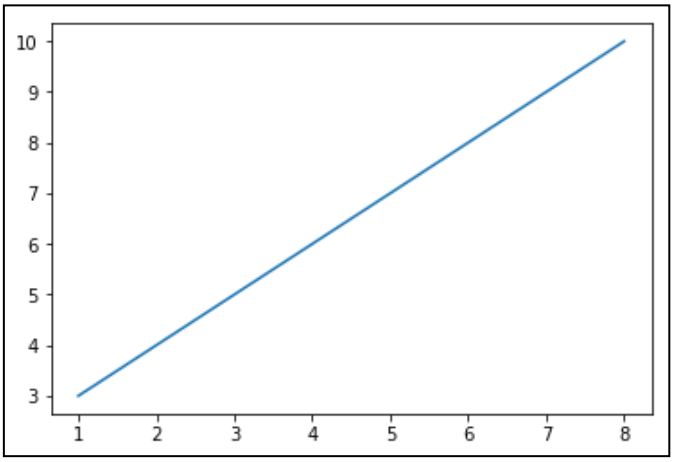
Parameter 2 is an array containing the points on the **y-axis**.

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

`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()`
	

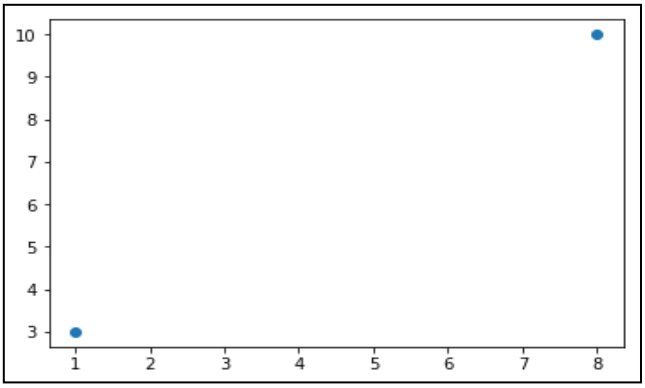
There are many types of single lines/multiple lines that can be drawn, explore other types at: https://www.w3schools.com/python/matplotlib_line.asp

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
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()	

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.

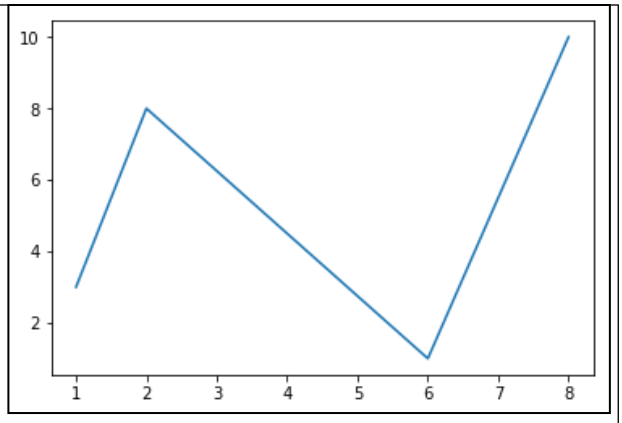
Example

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()
```



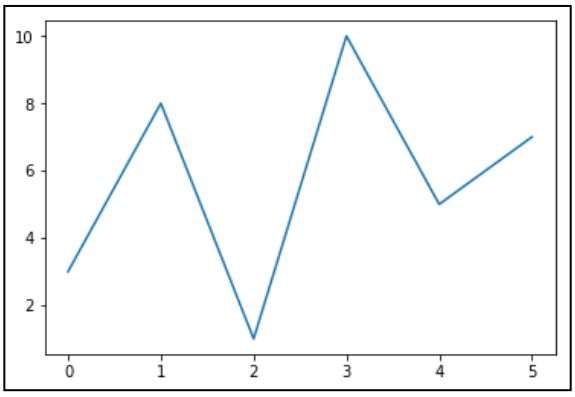
3.6 Default X-Points

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:

Code	Output
import matplotlib.pyplot as plt import numpy as np ypoints = np.array([3, 8, 1, 10, 5, 7]) plt.plot(ypoints) plt.show()	

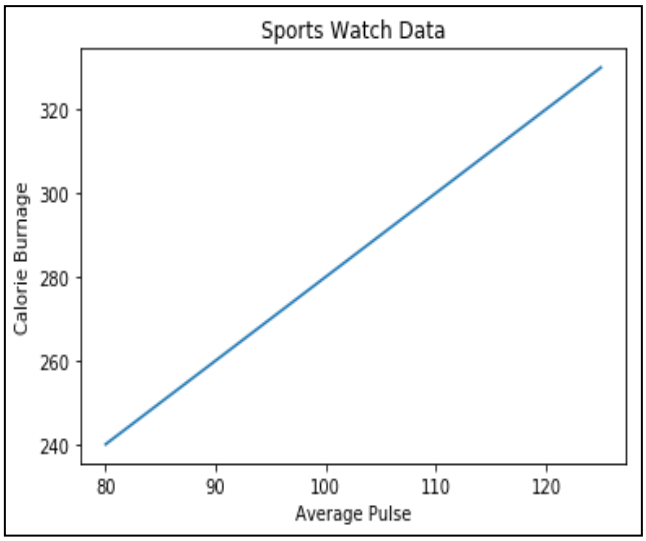
The x-points in the example above are [0, 1, 2, 3, 4, 5] by default.

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:

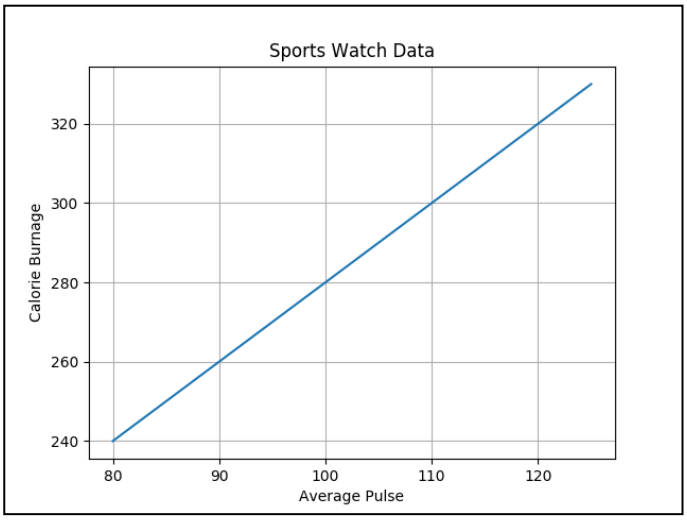
Code	Output
import numpy as np import matplotlib.pyplot as plt x = np.array([80, 85, 90, 95, 100, 105, 110, 115, 120, 125]) y = np.array([240, 250, 260, 270, 280, 290, 300, 310, 320, 330]) plt.plot(x, y) plt.title("Sports Watch Data") plt.xlabel("Average Pulse") plt.ylabel("Calorie Burnage") plt.show()	

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
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()	

Different type of grid can be generated, for more details see:
https://www.w3schools.com/python/matplotlib_grid.asp

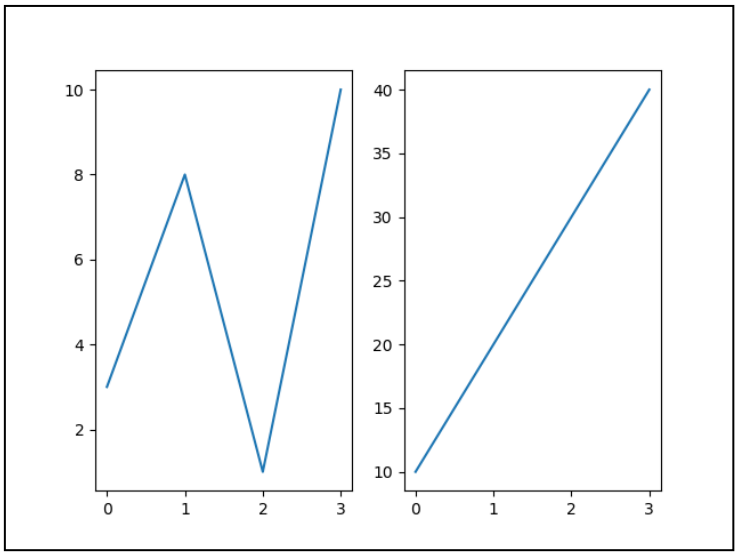
3.9 Display Multiple Plots

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

Example

Draw 2 plots:

Code	Output

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()
	

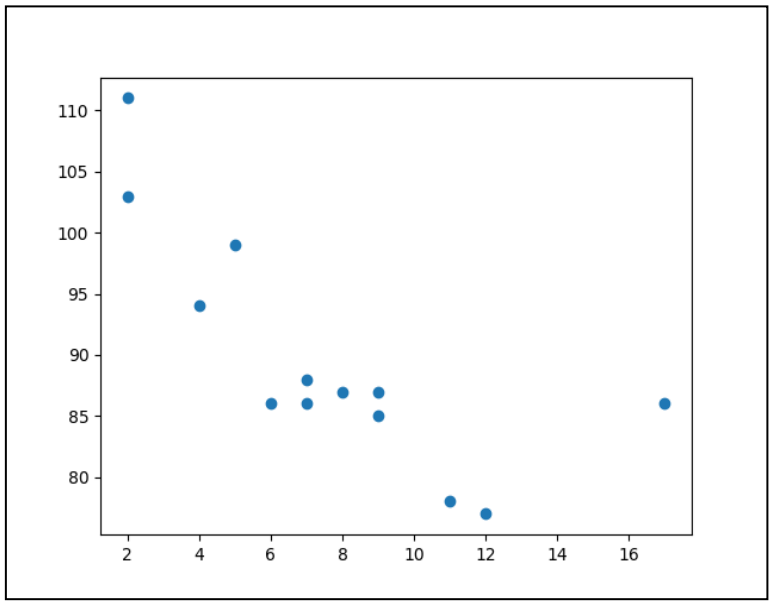
There different ways to plot multiple plots:
https://www.w3schools.com/python/matplotlib_subplots.asp

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
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()	

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.

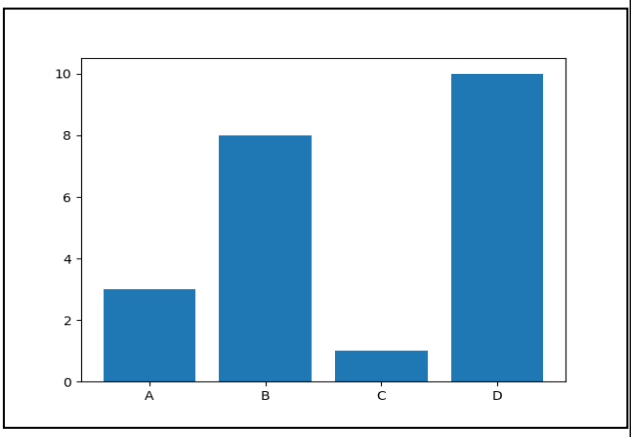
There are different type of scatter graphs that can be created (kindly see the link given, as all examples will make the manual lengthy):
https://www.w3schools.com/python/matplotlib_scatter.asp

3.11 Creating Bars

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

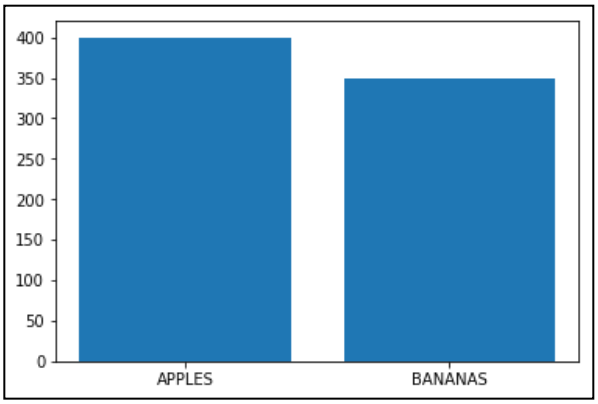
Example

Draw 4 bars:

Code	Output
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()	

The bar() function takes arguments that describes the layout of the bars.

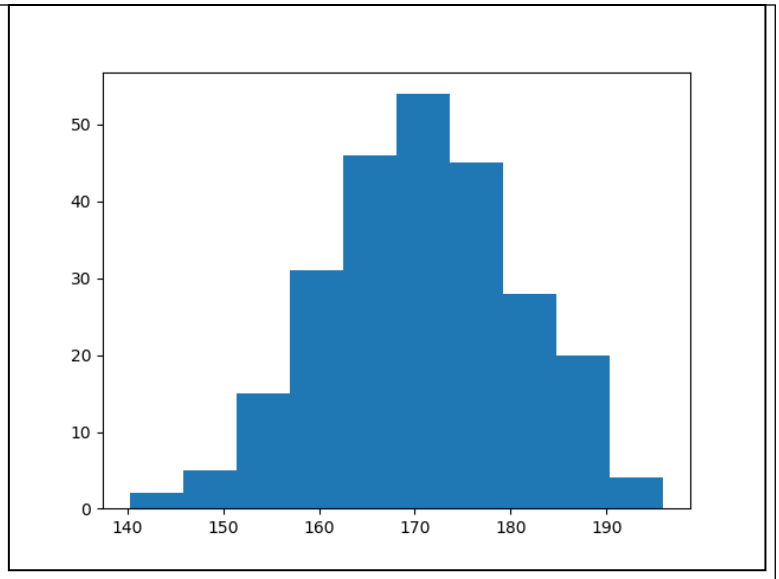
The categories and their values represented by the *first* and *second* argument as arrays.

import matplotlib.pyplot as plt x = ["APPLES", "BANANAS"] y = [400, 350] plt.bar(x, y)	

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:

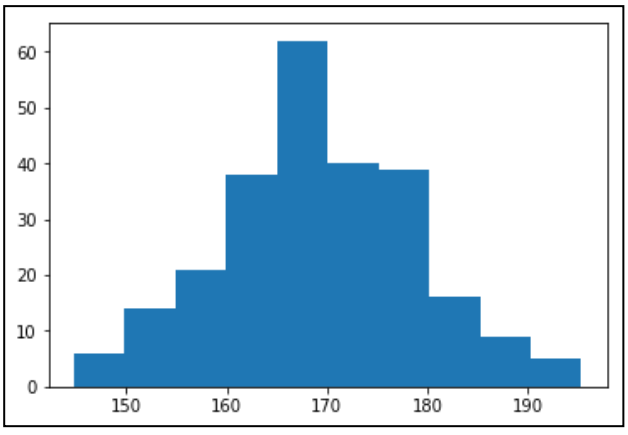
You can read from the histogram that there are approximately:	

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

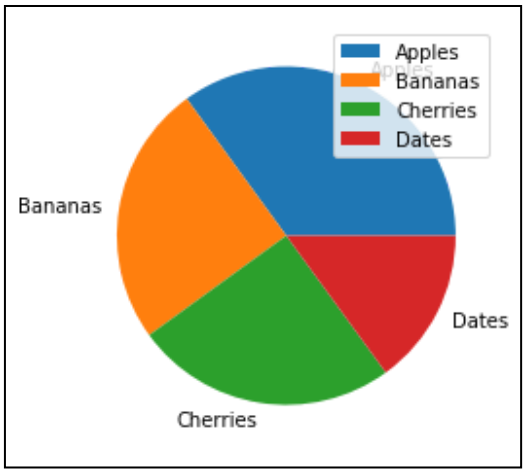
In Matplotlib, we use the hist() function to create histograms.

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
import matplotlib.pyplot as plt import numpy as np x = np.random.normal(170, 10, 250) plt.hist(x) plt.show()	

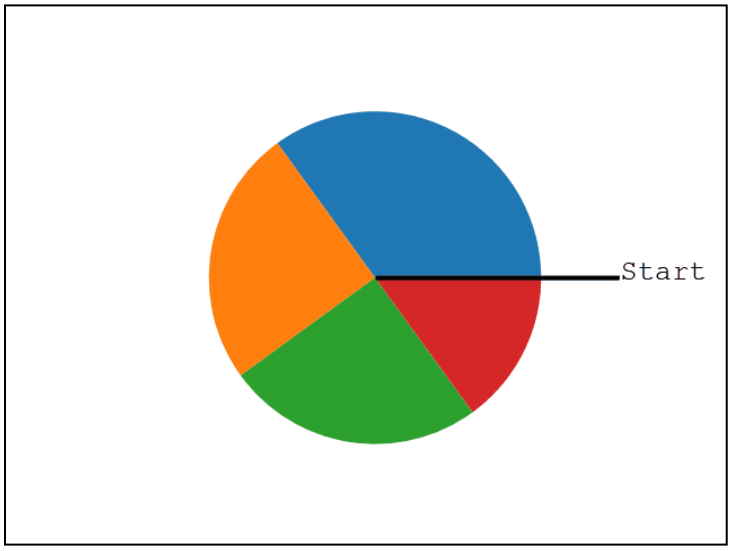
3.13 Creating Pie Charts

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

Code	Output
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()	

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*:



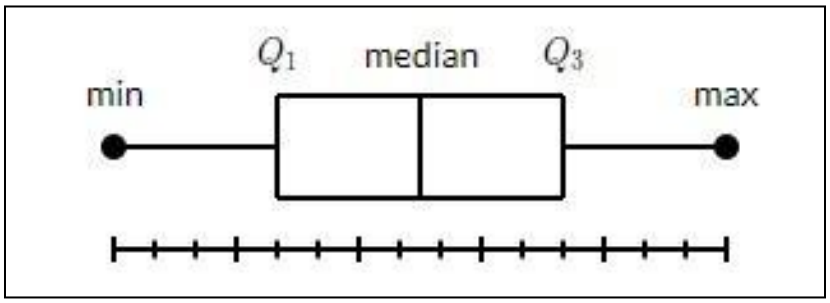
Note: The size of each wedge is determined by comparing the value with all the other values, by using this formula:

The value divided by the sum of all values: x/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

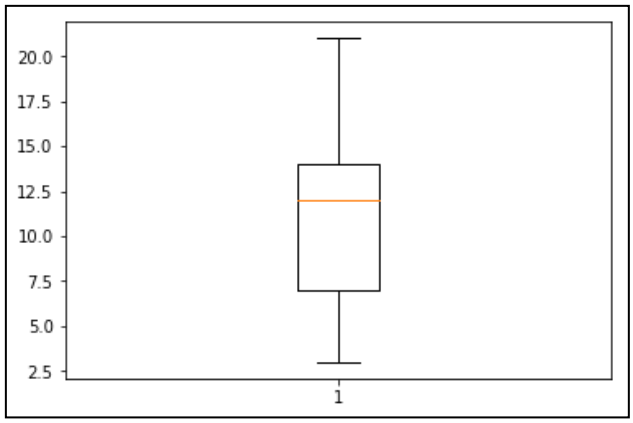
minimum, first quartile, median, third quartile, and maximum. In a box plot, we draw a box from the first quartile to the third quartile. A vertical line goes through the box at the median. The whiskers go from each quartile to the minimum or maximum.



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

4 Mini-Max Algorithm in Artificial Intelligence

1- Mini-max algorithm is a recursive or backtracking algorithm which is used in decision-making and game theory. It provides an optimal move for the player assuming that opponent is also playing optimally.

2- Mini-Max algorithm uses recursion to search through the game-tree.

3- Min-Max algorithm is mostly used for game playing in AI. Such as Chess, Checkers, tic-tac-toe, go, and various tow-players game. This Algorithm computes the minimax decision for the current state.

4- In this algorithm two players play the game; one is called MAX and other is called MIN.

5- Both the players fight it as the opponent player gets the minimum benefit while they get the maximum benefit.

6- Both Players of the game are opponent of each other, where MAX will select the maximized value and MIN will select the minimized value.

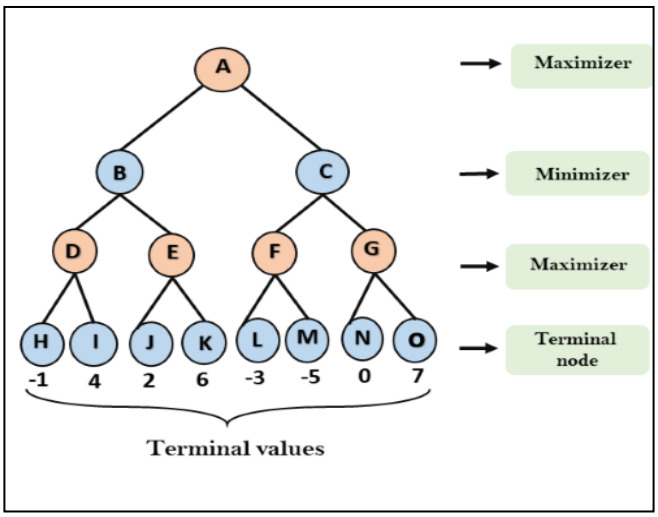
7- The minimax algorithm performs a depth-first search algorithm for the exploration of the complete game tree.

8- The minimax algorithm proceeds all the way down to the terminal node of the tree, then backtrack the tree as the recursion.

4.1 Working of Min-Max Algorithm

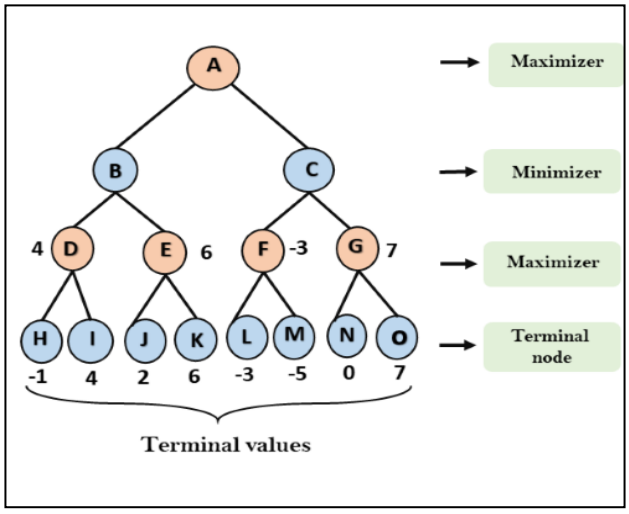
- The working of the minimax algorithm can be easily described using an example. Below we have taken an example of game-tree which is representing the two-player game.
- In this example, there are two players one is called Maximizer and other is called Minimizer.
- Maximizer will try to get the Maximum possible score, and Minimizer will try to get the minimum possible score.
- This algorithm applies DFS, so in this game-tree, we have to go all the way through the leaves to reach the terminal nodes.
- At the terminal node, the terminal values are given so we will compare those value and backtrack the tree until the initial state occurs. Following are the main steps involved in solving the two-player game tree:

Step-1: In the first step, the algorithm generates the entire game-tree and apply the utility function to get the utility values for the terminal states. In the below tree diagram, let's take A is the initial state of the tree. Suppose maximizer takes first turn which has worst-case initial value =- infinity, and minimizer will take next turn which has worst-case initial value = +infinity.



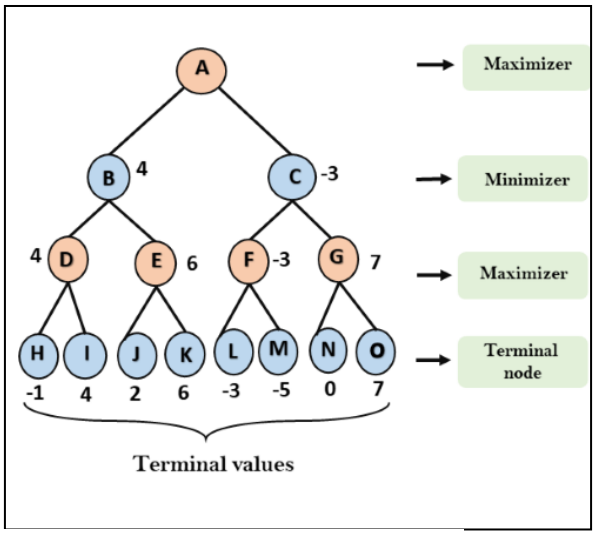
Step 2: Now, first we find the utilities value for the Maximizer, its initial value is $-\infty$, so we will compare each value in terminal state with initial value of Maximizer and determines the higher nodes values. It will find the maximum among the all.

- For node D $\max(-1,-\infty) \Rightarrow \max(-1,4) = 4$
- For Node E $\max(2,-\infty) \Rightarrow \max(2,6) = 6$
- For Node F $\max(-3,-\infty) \Rightarrow \max(-3,-5) = -3$
- For node G $\max(0,-\infty) = \max(0,7) = 7$



Step 3: In the next step, it's a turn for minimizer, so it will compare all nodes value with $+\infty$, and will find the 3rd layer node values.

- For node B= min(4,6) = 4
- For node C= min (-3, 7) = -3



That was the complete workflow of the minimax two player game.

4.2 Properties of Mini-Max algorithm:

- **Complete-** Min-Max algorithm is Complete. It will definitely find a solution (if exist), in the finite search tree.
- **Optimal-** Min-Max algorithm is optimal if both opponents are playing optimally.
- **Time complexity-** As it performs DFS for the game-tree, so the time complexity of Min-Max algorithm is **$O(b^m)$**, where b is branching factor of the game-tree, and m is the maximum depth of the tree.
- **Space Complexity-** Space complexity of Mini-max algorithm is also similar to DFS which is **O(bm)**.

4.3 Limitation of the Mini-Max Algorithm:

The main drawback of the minimax algorithm is that it gets really slow for complex games such as Chess, go, etc. These type of games has a huge branching factor, and the player has lots of choices to decide.