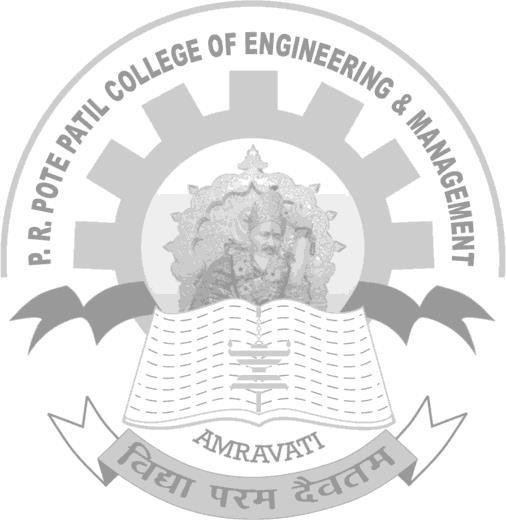
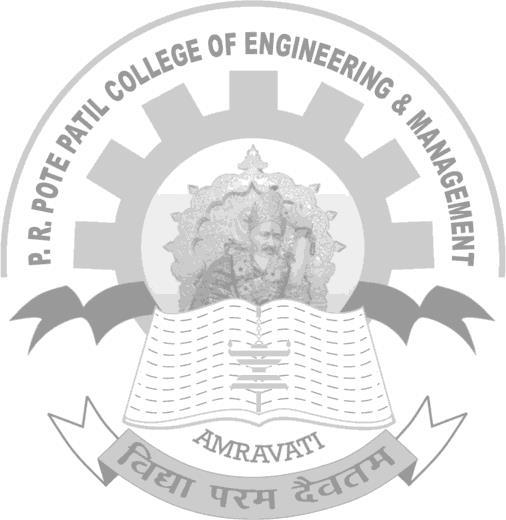
**C-Lab II (Hardware Lab)**

#### PRACTICAL NO. 01

###### AIM OF PRACTICAL :

1. Installation of Anaconda
2. write basic Programs using python in jupyter (control structures & functions.)

Write a program using for loop to display sum of ODD numbers & EVEN numbers upto n (Input – From User).



## Learning objectives:

To understand the installation of anaconda and study about it , and to learn how to work on jupyter.

## Learning Outcomes:

* 1. In this practical we learn Intaallation of anaconda
  2. Working on jupyter ,How to run programs in jupyter .

###### SOFTWARE / HARDWARE REQUIRED :-

|  |  |
| --- | --- |
| **S N** | **Name of Equipment / Items / Software Tool** |
| **1**  **.** | Hardware : Computer System |
| **2**  **.** | Motherboard |

**THEORY:**

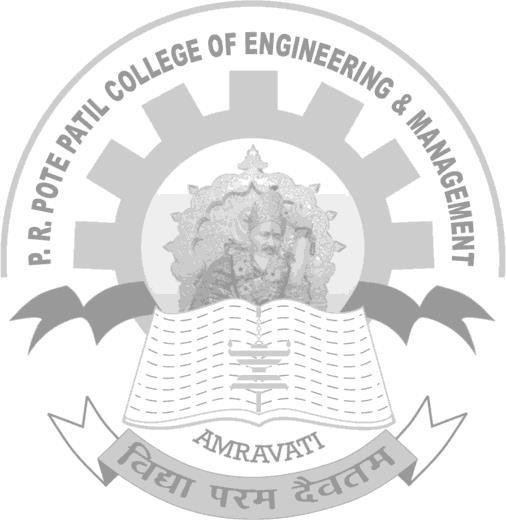
**A)Installation of Anaconda.:**

Step 1: Download Python 3.7 as Python version 2 will have more support by community. Depending on Your computer system , choose either 32bit or 64 bit installer to download the .exe file.

Step 2: After downloading file run the file , the file will open click Next And click I agree to the license.

PRPCEM,

**C-Lab II (Hardware Lab)**



Step 3: Choose just me and choose installation location by clicking browse or leave as it is and continues to click next

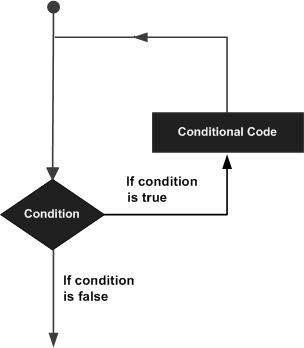
Step 4: and then Click Install , once the installation is done, open anaconda prompt from windows start menu bar.

Step 5: Set path and environment variable , if you are using anaconda prompt

|  |
| --- |
| **B. Write basic Programs using python in jupyter (control structures & functions)**  **Write a program using for loop to display sum of ODD numbers & EVEN numbers upto n (Input – From User).** |
| **Jupyter:-**  The Jupyter Notebook is an open source web application that you can use to create and share documents that contain live code, equations, visualizations, and text. Jupyter Notebook is maintained by the people at Project Jupyter.  **Loop in Python:-**  In general, statements are executed sequentially: The first statement in a function is executed first, followed by the second, and so on. There may be a situation when you need to execute a block of code several number of times.  Programming languages provide various control structures that allow for more complicated execution paths.  A loop statement allows us to execute a statement or group of statements multiple times. The following diagram illustrates a loop statement - |

PRPCEM,

**C-Lab II (Hardware Lab)**



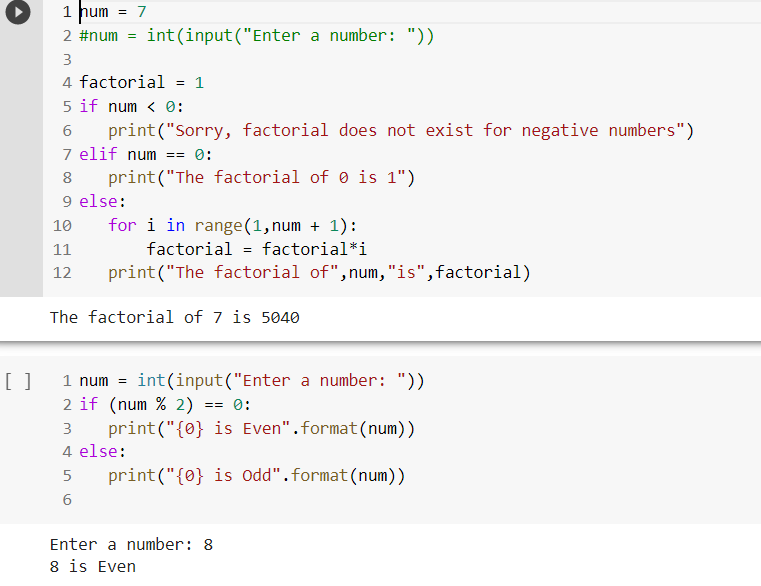
Python programming language provides following types of loops to handle looping requirements.

|  |  |
| --- | --- |
| **Sr.No.** | **Loop Type & Description** |
| 1 | While Loop :Repeats a statement or group of statements while a given condition is TRUE. It tests the condition before executing the loop body. |
| 2 | For loop: Executes a sequence of statements multiple times and abbreviates the code that manages the loop variable. |
| 3 | Nested Loop: You can use one or more loop inside any another while, for or do..while loop. |

# Program:

PRPCEM,

**C-Lab II (Hardware Lab)**

****

PRPCEM,

**C-Lab II (Hardware Lab)**

## 

## Conclusion:

Thus we studied how to install anaconda on our system and how looping statements works in Python.

PRPCEM,

**PRACTICAL NO.2**

**AIM OF PRACTICAL:**

Demonstration of use of built-in data structure . a.List

b.tuple c.set

d.Dictionary

Write a program to create a list and display sum of all items. Write a program to demonstrate the use of indexing & slicing

Write a program to demonstrate the use of list methods ('append','clear', 'copy', 'count', 'extend', 'index', 'insert', 'pop', 'remove', 'reverse', 'sort')

###### LEARNING OBJECTIVES:

To understand the use of built-in data structure such as list, tuple, set, dictionary and to build and demonstrate it.

###### LEARNING OUTCOMES:

1. In this practical we have learn about the use of built-in data structure
2. Learned to build and demonstrate each of the built-in data structure

###### SOFTWARE/HARDWARE REQUIRED:

|  |  |
| --- | --- |
| **SN** | **Name of Equipment/Items/Software Tool** |
| **1.** | Hardware: Computer System |
| **2.** | Software: Anaconda Navigator (Jupyter Notebook) |

**THEORY:**

###### Introduction to built-in data structure:

The basic Python data structures in Python include list, set, tuples, and dictionary. Each of the data structures is unique in its own way. Data structures are “containers” that organize and group data according to type.

The data structures differ based on mutability and order. **Mutability** refers to the ability to change an object after its creation. Mutable objects can be modified, added, or deleted after they’ve been created, while immutable objects cannot be modified after their creation. **Order**, in this context, relates to whether the position of an element can be used to access the element.

#### List :

A list is defined as an ordered collection of items, and it is one of the essential data structures when using Python to create a project. The term “ordered collections” means that each item in a list comes with an order that uniquely identifies them. The order of elements is an inherent characteristic that remains constant throughout the life of the list.

Since everything in Python is considered an object, creating a list is essentially creating a Python object of a specific type. When creating a list, all the items in the list should be put in square brackets and separated by commas to let Python know that a list has been created. A sample list can be written as follows:

**List\_A = [item 1, item 2, item 3….., item n]**

#### Tuple :

A tuple is a built-in data structure in Python that is an ordered collection of objects. Unlike lists, tuples come with limited functionality.The primary differing characteristic between lists and tuples is mutability. Lists are mutable, whereas tuples are immutable. Tuples cannot be modified, added, or deleted once they’ve been created. Lists are defined by using parentheses to enclose the elements, which are separated by commas.

The use of parentheses in creating tuples is optional, but they are recommended to distinguish between the start and end of the tuple. A sample tuple is written as follows:

**tuple\_A = (item 1, item 2, item 3,…, item n)**

#### Set :

A set is defined as a unique collection of unique elements that do not follow a specific order. Sets are used when the existence of an object in a collection of objects is more important than the number of times it appears or the order of the objects. Unlike tuples, sets are mutable – they can be modified, added, replaced, or removed. A sample set can be represented as follows:

**set\_a = {“item 1”, “item 2”, “item 3”,….., “item n”}**

#### Dictionary :

Python includes the following dictionary functions −

Sr.No Function with Description [cmp(dict1, dict2)](https://www.tutorialspoint.com/python/dictionary_cmp.htm)

1

Compares elements of both dict.

[len(dict)](https://www.tutorialspoint.com/python/dictionary_len.htm)

Gives the total length of the dictionary. This would be equal to the number of items in the dictionary.

2

3

[str(dict)](https://www.tutorialspoint.com/python/dictionary_str.htm)

Produces a printable string representation of a dictionary

[type(variable)](https://www.tutorialspoint.com/python/dictionary_type.htm)

Returns the type of the passed variable. If passed variable is dictionary, then it would return a dictionary type.

4

###### PROGRAM & OUTPUT:

#Python Program to find sum of elements in list total=0

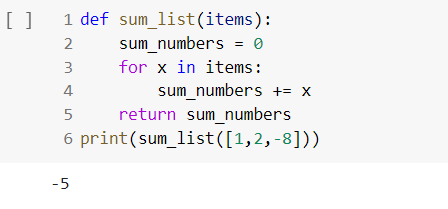
#creating a list list1=[11, 5, 17, 18, 23]

#Iterate each element in list #and add them in variable total for ele in range(0, len(list1)):

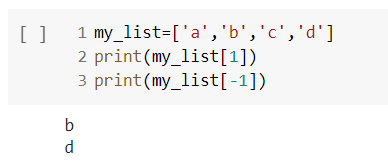
total=total+list1[ele]

#printing total values

print("Sum of all elements in given list:", total)



my\_list=['a', 'b', 'c', 'd'] print(my\_list[1]) print(my\_list[-1])



my\_list = ['a' , 'b','c','d']

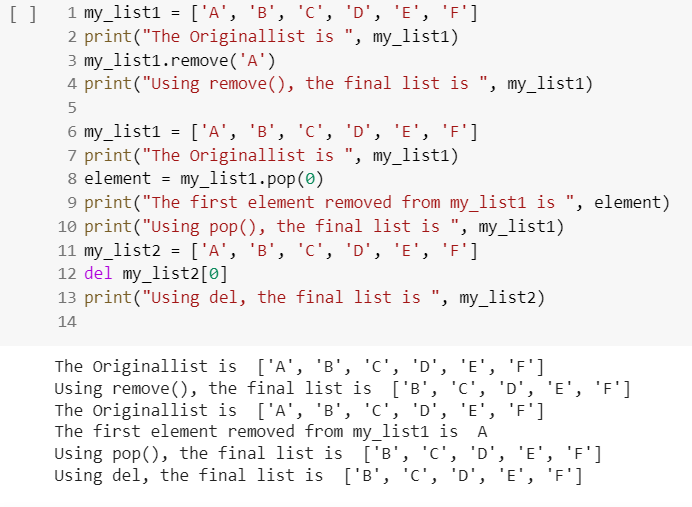
print(my\_list[1:]) #print elements from index 1 to end print(my\_list[:2]) #print elements from start to index 2 print(my\_list[1:3]) #print elements from index 1 to index 3

print(my\_list[::2]) #print elements from start to end using step sizes of 2



# python code to demonstrate the working of # del and pop()

# initializing list



#Python code to demonstrate the working of #insert() and remove()

# initializing List

lis = [2, 1, 3, 5, 3, 8]

# using insert() to insert 4 at 3rd pos lis.insert(3, 4)

#displaying List after inserting

print("List elements after inserting 4 are : ", end=" ") for i in range(0, len(lis)):

print(lis[i], end=" ") print("\r")

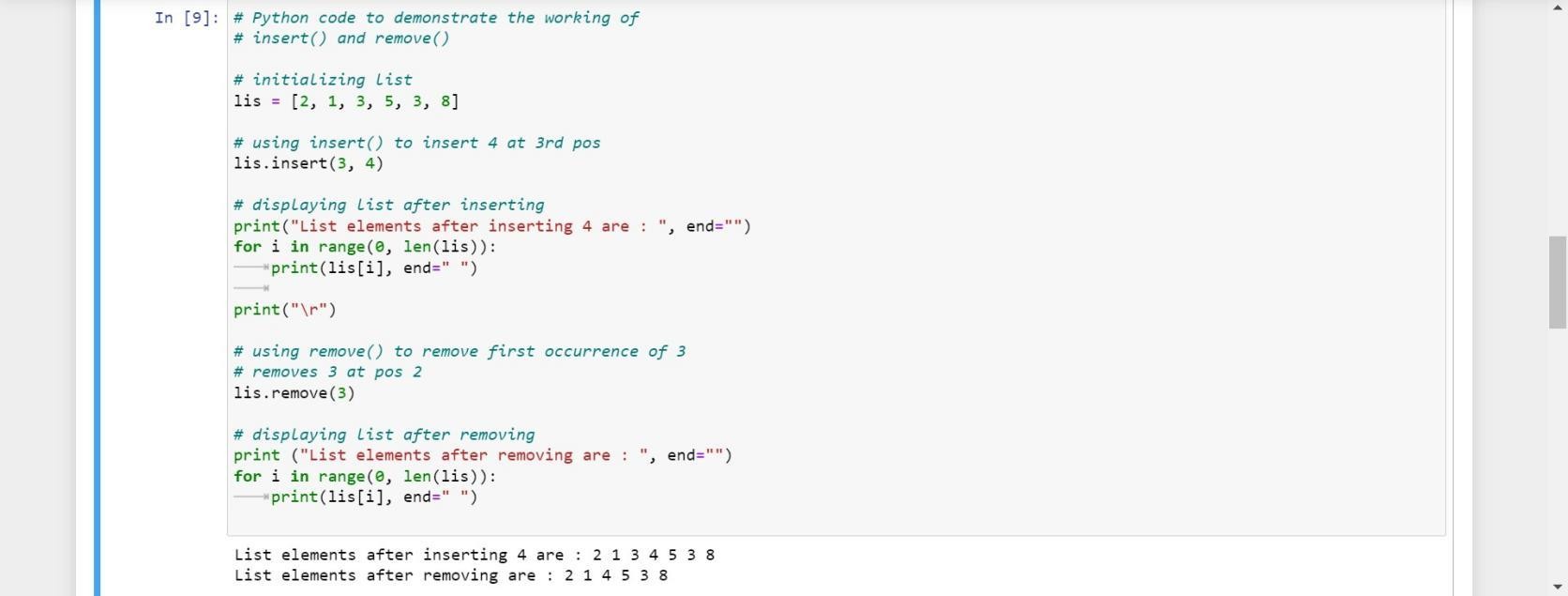
# using remove() to remove first occurrence of 3 # removes 3 at pos 2

lis.remove(3)

#displaying List after removing

print ("List elements after removing are:",end="") for i in range(0, len(lis)):

print(lis[i], end="")



# Python code to demonstrate the working of # sort() and reverse()

# initializing List lis= [2, 1, 3, 5, 3, 8]

# using sort() to sort the list lis.sort()

# displaying List after sorting

print ("List elements after sorting are :", end="") for i in range(0, len(lis)):

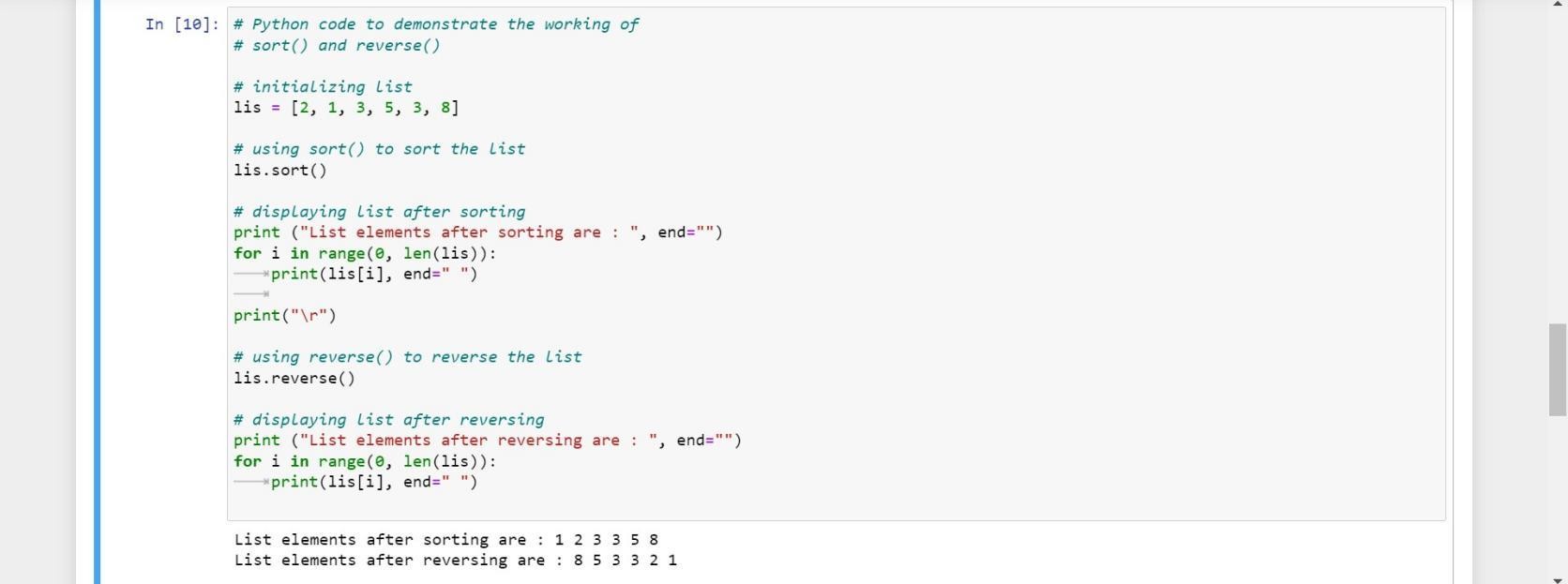
print (lis[i], end=" " print("\n")

# using reverse() to reverse the list lis.reverse()

# displaying List after reversing

print ("List elements after reversing are : ", end="") for i in range(0, len(lis)):

print(lis[i], end=" ")



#python code to demonstrate the working of #extend () and clear()

#initializing list 1

lis1 = [2,1,3,5]

#initializing list 2

lis2 = [6,4,3]

#using extend () to add element of lis2 in lis1 lis1.extend(lis2)

#displaying list after sorting

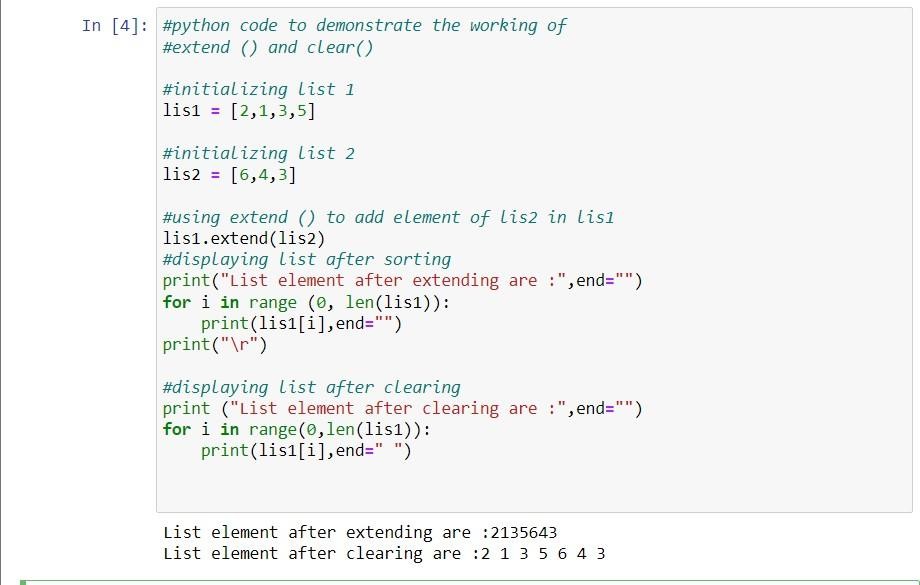
print("List element after extending are :",end="") for i in range (0, len(lis1)):

print(lis1[i],end="") print("\r")

#displaying list after clearing

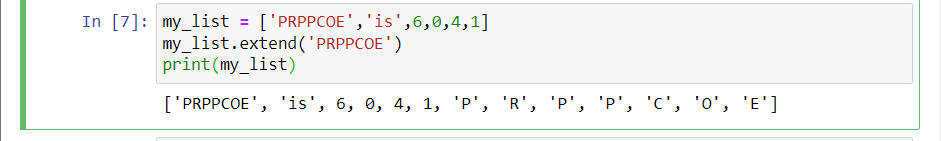
print ("List element after clearing are :",end="") for i in range(0,len(lis1)):

print(lis1[i],end=" ")



my\_list = ['PRPPCOE','is',6,0,4,1]

my\_list.extend('PRPPCOE') print(my\_list)



###### CONCLUSION:

In this practical, we learned a lot of details about the various types of built-in data structure like lists, tuple, sets and dictionary and its uses. Also , we have demonstrated the use of indexing and slicing in python programming.

**PRACTICAL NO –3**

###### AIM OF PRACTICAL :

Introduction and Demostration of data science Numpy Library.

###### LEARNING OBJECTIVES :

* To learn the basic syntax of NumPy arrays.
* To learn important dos and don’ts for how to use them effectively in their own applications.
* Also gain awareness of useful NumPy subpackages and leave with enough know-how to incorporate those packages into their own codes.

###### LEARNING OUTCOMES :

NumPy is the de facto standard for multidimensional arrays in Python data science, and many of the most popular libraries are built on top of it. Learning NumPy is a great way to set down a solid foundation as you expand your knowledge into more specific areas of data science.

###### SOFTWARE/HARDWARE REQUIRED :

* 1. Hardware : Computer System
  2. Software : Anaconda Navigator (Jupyter Notebook)

###### THEORY :

As you explore Data Science using Python language, you would encounter a library called NumPy.

NumPy arrays make doing operations and functions on matrices more efficient and convenient for users. NumPy can do what list can do and much more. Data scientists mostly use NumPy to extract information from the raw data. They do this by doing some operations to the data array and storing its result. The result sometimes is stored in a

datatype from another library called Pandas DataFrame which is also built on the NumPy library.

NumPy stands for numeric python which is a python package for the computation and processing of the multidimensional and single dimensional array elements.

Travis oliphat created NumPy package in 2005 by injecting the features of the ancestor module Numeric into another module Numarray.

It is an extension module of Python which is mostly written in C. It provides various functions which are capable of performing the numeric computations with a high speed.NumPy provides various powerful data structures, implementing multi- dimensional arrays and matrices. These data structures are used for the optimal computations regarding arrays and matrices

ADVANTAGES OF USING NUMPY LIBRARY

1. NumPy performs array-oriented computing.
2. It efficiently implements the multidimensional arrays.
3. It performs scientific computations.
4. NumPy provides the in-built functions for linear algebra and random number generation.
5. It is capable of performing Fourier Transform and reshaping the data stored in multidimensional arrays.

# Importing NumPy import numpy as np

###### PROGRAM & OUTPUT :

# Creating 1-D array using Functions import numpy as np

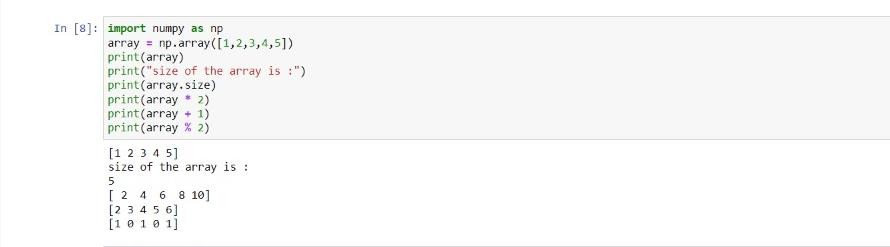
array = np.array([1,2,3,4,5])

print(array)

print(“size of the array is : “) print(array.size) print(“array using function”) print(array \* 2)

print(array + 1)

print(array % 2)



# Creating a two-dimensional array

import numpy as np

array = np.array([[1,2,3],[4,5,6]]) print(array)



###### CONCLUSION :

In this practical,We learnesd that NumPy is a powerful and faster Python library for mathematical and scientific computing. And provides platform for implementing multi- dimensional arrays and matrices. provides various functions which are capable of performing the numeric computations with a high speed.

#### PRACTICAL NO. 4

###### AIM OF PRACTICAL :

Introduction and Demonstration of data science Pandas Library.

###### LEARNING OBJECTIVE:

To understand the use of Pandas Library.

###### LEARNING OUTCOMES:

**SOFTWARE/HARDWARE REQUIRED:**

|  |  |
| --- | --- |
| **SN** | **Name of Equipment/Items/Software Tool** |
| **1.** | Hardware: Computer System |
| **2.** | Software: Anaconda Navigator (Jupyter Notebook) |

###### THEORY:

Pandas is a Python library used for working with data sets. It has functions for analysing

, cleaning, exploring, and manipulating data.

The name "Pandas" has a reference to both "Panel Data", and "Python Data Analysis" and was created by Wes McKinney in 2008.Pandas allows us to analyse big data and make conclusions based on statistical theories. Pandas can clean messy data sets, and make them readable and relevant. Relevant data is very important in data science.

###### Pandas Series

A Pandas Series is like a column in a table.

It is a one-dimensional array holding data of any type.

###### Pandas DataFrame

A Pandas DataFrame is a 2 dimensional data structure, like a 2 dimensional array, or a table with rows and columns.

###### Read CSV Files

A simple way to store big data sets is to use CSV files (comma separated files).

CSV files contains plain text and is a well know format that can be read by everyone including Pandas.

###### Read JSON

Big data sets are often stored, or extracted as JSON.

JSON is plain text, but has the format of an object, and is well known in the world of programming, including Pandas.

###### Viewing the Data

One of the most used method for getting a quick overview of the DataFrame, is the head() method.

The head() method returns the headers and a specified number of rows, starting from the top.

###### PROGRAM :

1. import pandas as pd a = [1, 7, 2, 8, 9] myvar = pd.Series(a) print(myvar)
2. print (myvar [0]) data = {

“calories” : [420, 380, 390]

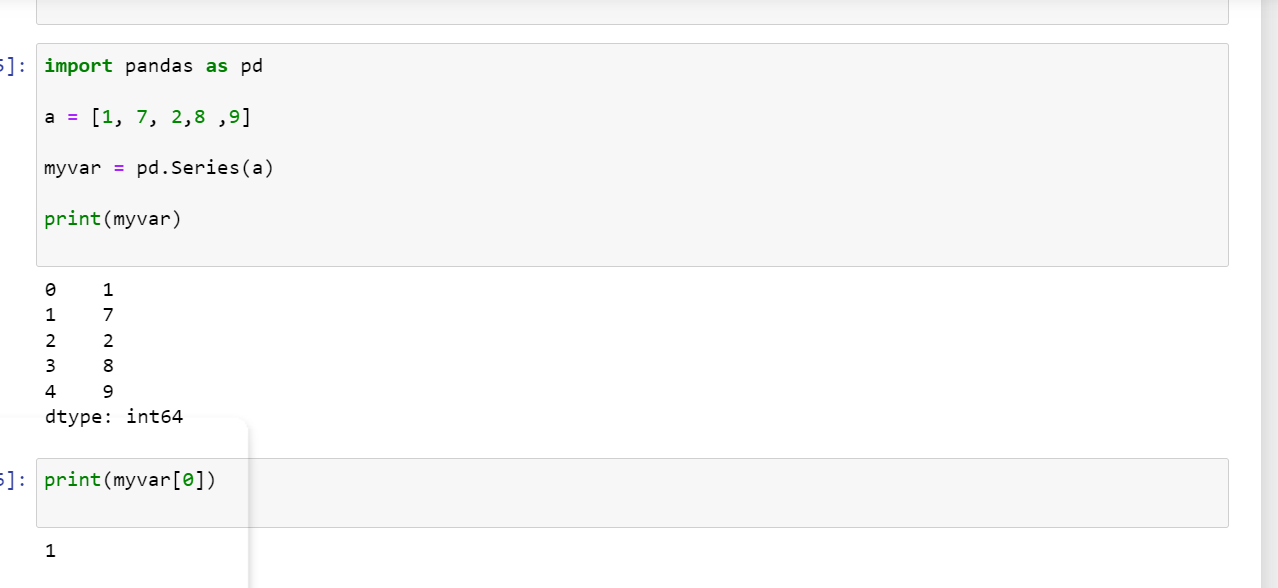
“duration” : [50, 40, 45]

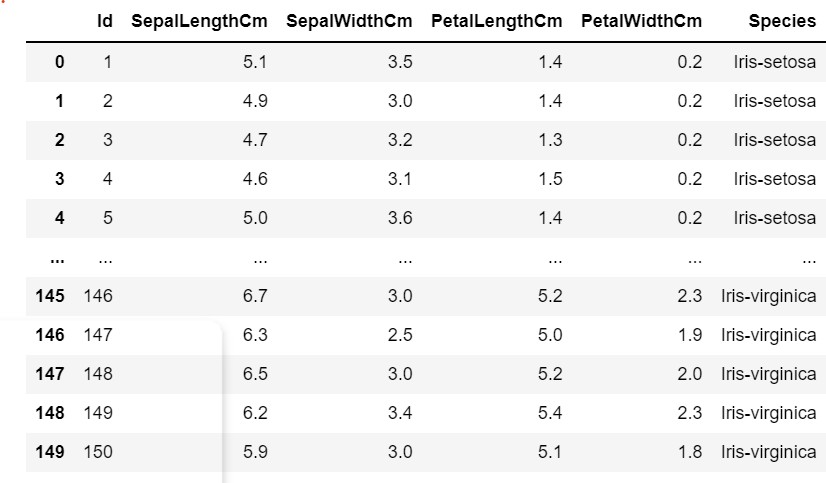
}

myvar = pd.DataFrames(data) print (myvar)

1. df = pd.read\_csv(‘C:\\Users\\user\Downloads\\archive\\Iris.csv’) df
2. df.head()
3. df.tail()

###### AND OUTPUT:







**CONCLUSION:**

In this practical, we learned a lot of details about Pandas library and its uses. Also , we have demonstrated the use of series and Dataframe in python programming.

#### PRACTICAL N0 - 5

###### AIM OF PRACTICAL:

Introduction & Demonstration of data science Matplotlib library

###### LEARNING OBJECTIVES:

* To Learn about the fundamentals of a matplotlib plot such as title, axis, grid, etc.
* To Learn how to plot various two-dimensional plots such as line, bar, scatter, pie, etc.
* To Learn how to plot various three-dimensional plots such as space, line, scatter, etc.
* To Learn how to read and manipulate images.

###### LEARNING OUTCOMES:

One of the greatest benefits of visualization is that it allows us visual access to huge amounts of data in easily digestible visuals. Matplotlib consists of several plots like line, bar, scatter, histogram etc.

###### SOFTWARE/HARDWARE REQUIRED:

* 1. Hardware: Computer System
  2. Software: Anaconda Navigator (Jupyter Notebook)

#### Theory:

There are thousands of [**libraries**](https://www.quora.com/What-is-a-Python-library-and-what-can-I-use-it-for) in Python, and Matplotlib is one of the most powerful tools for data visualization in Python.

[**Matplotlib**](https://matplotlib.org/) tries to make easy things easy and hard things possible. You can generate plots, histograms, power spectra, bar charts, error chart, scatterplots, etc., with just a few lines of code.

Matplotlib is a cross-platform, data visualization and graphical plotting library for Python and its numerical extension NumPy. As such, it offers a viable open source alternative to MATLAB. Developers can also use matplotlib’s APIs (Application Programming Interfaces) to embed plots in GUI applications.

A Python matplotlib script is structured so that a few lines of code are all that is required in most instances to generate a visual data plot. The matplotlib scripting layer overlays two APIs:

* The pyplot API is a hierarchy of Python code objects topped by *matplotlib.pyplot*
* An OO (Object-Oriented) API collection of objects that can be assembled with greater flexibility than pyplot. This API provides direct access to Matplotlib’s backend layers.

#### Importing the library:

To get matplotlib up and running in our environment, we need to import it.

import **matplotlib.pyplot** as plt

It is common practice to import matplotlib under the alias **plt *—*** that way, we have to type less code to reference it further down the line.

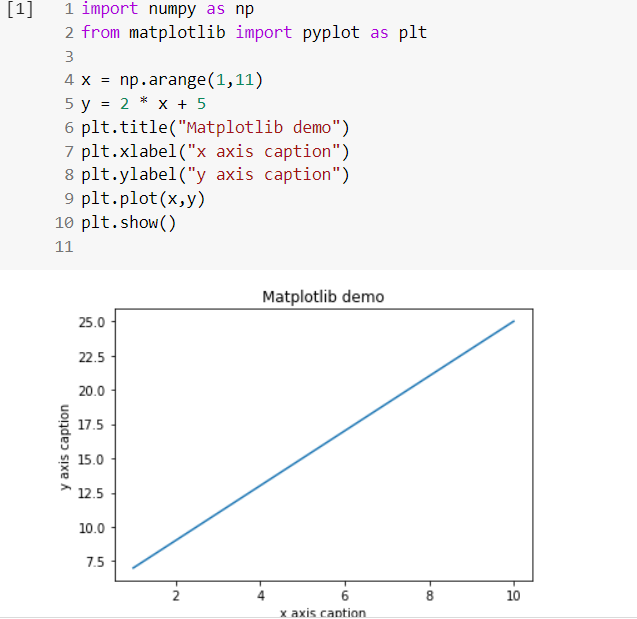
Whenever you plot with matplotlib, the two main code lines should be,

* + 1. Type of graph — this is where you define a bar chart, line chart, etc.
    2. Show the graph — this is to display the graph

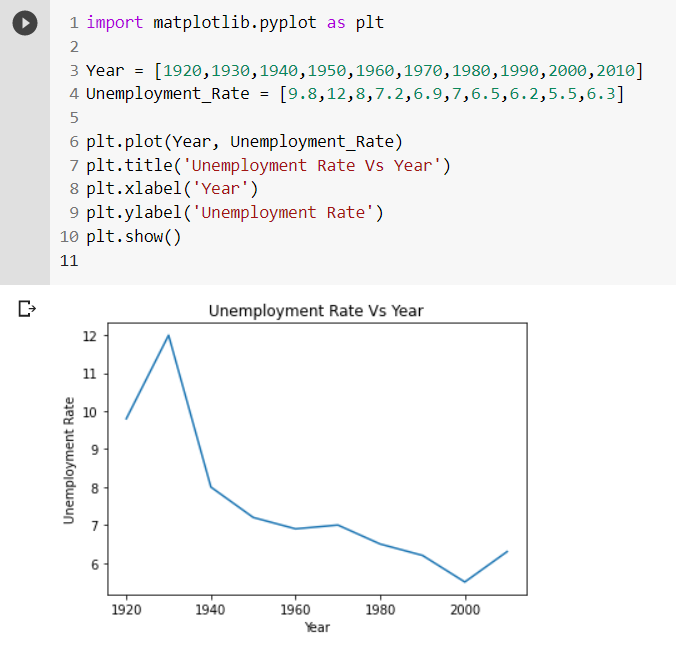
#### Functions of Matplotlib:

|  |  |
| --- | --- |
| **Function** | **Description** |
| [acorr](https://matplotlib.org/3.1.1/api/_as_gen/matplotlib.pyplot.acorr.html#matplotlib.pyplot.acorr) | Plot the autocorrelation of *x*. |
| [angle\_spectrum](https://matplotlib.org/3.1.1/api/_as_gen/matplotlib.pyplot.angle_spectrum.html#matplotlib.pyplot.angle_spectrum) | Plot the angle spectrum. |
| [annotate](https://matplotlib.org/3.1.1/api/_as_gen/matplotlib.pyplot.annotate.html#matplotlib.pyplot.annotate) | Annotate the point *xy* with text *text*. |
| [arrow](https://matplotlib.org/3.1.1/api/_as_gen/matplotlib.pyplot.arrow.html#matplotlib.pyplot.arrow) | Add an arrow to the axes. |
| [autoscale](https://matplotlib.org/3.1.1/api/_as_gen/matplotlib.pyplot.autoscale.html#matplotlib.pyplot.autoscale) | Autoscale the axis view to the data (toggle). |
| [axes](https://matplotlib.org/3.1.1/api/_as_gen/matplotlib.pyplot.axes.html#matplotlib.pyplot.axes) | Add an axes to the current figure and make it the current axes. |
| [axhline](https://matplotlib.org/3.1.1/api/_as_gen/matplotlib.pyplot.axhline.html#matplotlib.pyplot.axhline) | Add a horizontal line across the axis. |
| [axhspan](https://matplotlib.org/3.1.1/api/_as_gen/matplotlib.pyplot.axhspan.html#matplotlib.pyplot.axhspan) | Add a horizontal span (rectangle) across the axis. |
| [axis](https://matplotlib.org/3.1.1/api/_as_gen/matplotlib.pyplot.axis.html#matplotlib.pyplot.axis) | Convenience method to get or set some axis properties. |
| [axvline](https://matplotlib.org/3.1.1/api/_as_gen/matplotlib.pyplot.axvline.html#matplotlib.pyplot.axvline) | Add a vertical line across the axes. |
| [axvspan](https://matplotlib.org/3.1.1/api/_as_gen/matplotlib.pyplot.axvspan.html#matplotlib.pyplot.axvspan) | Add a vertical span (rectangle) across the axes. |
| [bar](https://matplotlib.org/3.1.1/api/_as_gen/matplotlib.pyplot.bar.html#matplotlib.pyplot.bar) | Make a bar plot. |
| [barbs](https://matplotlib.org/3.1.1/api/_as_gen/matplotlib.pyplot.barbs.html#matplotlib.pyplot.barbs) | Plot a 2D field of barbs. |
| [barh](https://matplotlib.org/3.1.1/api/_as_gen/matplotlib.pyplot.barh.html#matplotlib.pyplot.barh) | Make a horizontal bar plot. |
| [box](https://matplotlib.org/3.1.1/api/_as_gen/matplotlib.pyplot.box.html#matplotlib.pyplot.box) | Turn the axes box on or off on the current axes. |
| [boxplot](https://matplotlib.org/3.1.1/api/_as_gen/matplotlib.pyplot.boxplot.html#matplotlib.pyplot.boxplot) | Make a box and whisker plot. |
| [broken\_barh](https://matplotlib.org/3.1.1/api/_as_gen/matplotlib.pyplot.broken_barh.html#matplotlib.pyplot.broken_barh) | Plot a horizontal sequence of rectangles. |
| [cla](https://matplotlib.org/3.1.1/api/_as_gen/matplotlib.pyplot.cla.html#matplotlib.pyplot.cla) | Clear the current axes. |
| [clabel](https://matplotlib.org/3.1.1/api/_as_gen/matplotlib.pyplot.clabel.html#matplotlib.pyplot.clabel) | Label a contour plot. |
| Title | Set a title for axes. |
| [clim](https://matplotlib.org/3.1.1/api/_as_gen/matplotlib.pyplot.clim.html#matplotlib.pyplot.clim) | Set the color limits of the current image. |
| [close](https://matplotlib.org/3.1.1/api/_as_gen/matplotlib.pyplot.close.html#matplotlib.pyplot.close) | Close a figure window. |
| [cohere](https://matplotlib.org/3.1.1/api/_as_gen/matplotlib.pyplot.cohere.html#matplotlib.pyplot.cohere) | Plot the coherence between *x* and *y*. |

**Matplotlib: Plot a Numpy Array:**



#### Line Graphs:



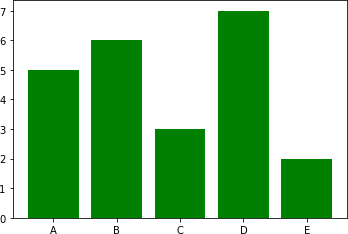
**Bar graphs:**

#create data for plotting x\_values = [5,6,3,7,2]

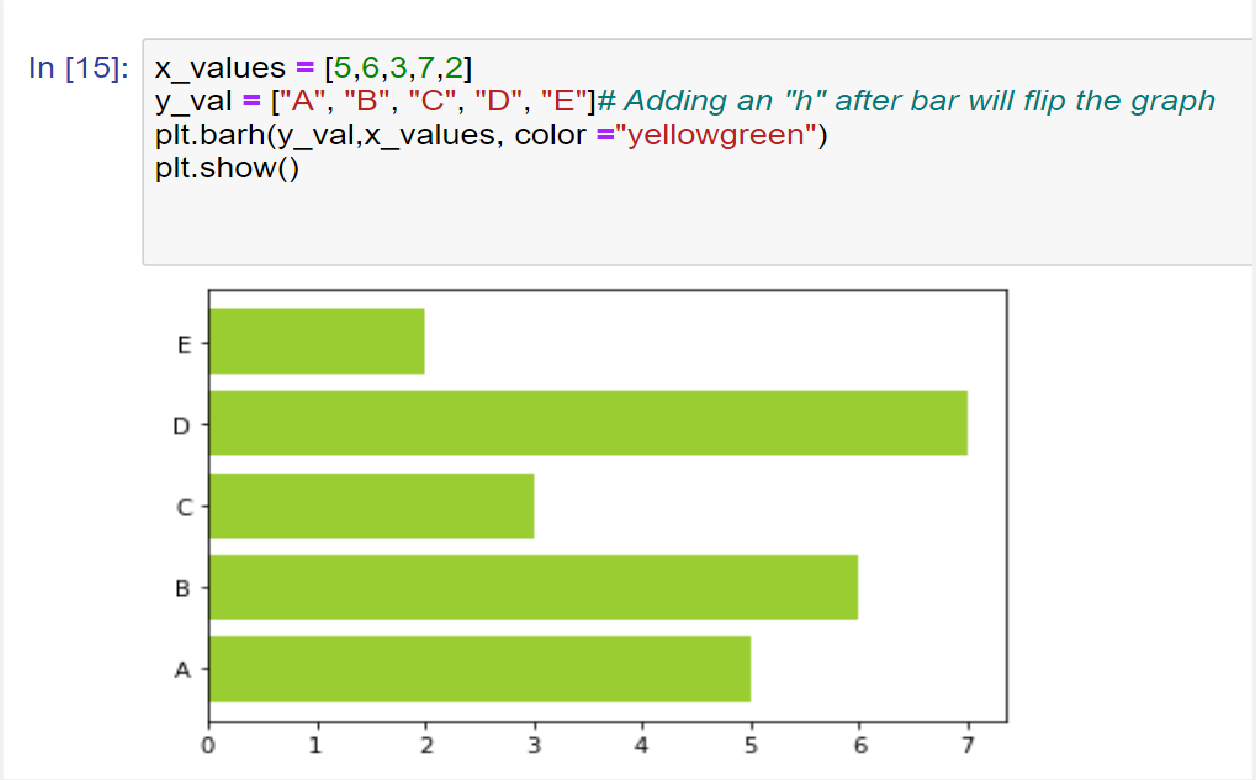
y\_values = ["A", "B", "C", "D", "E"]plt.**bar**(y\_val,x\_values, **color = "green"**) plt.show()

When using a bar graph, the change in code will be from *plt.plot()* to *plot.bar()* changes it into a bar chart. If you look inside the body of

the code, I also added an argument [**color**](https://matplotlib.org/3.1.0/gallery/color/named_colors.html) — this helps us quickly customize the color of the graph.



We can also flip the bar graph horizontally with the following,

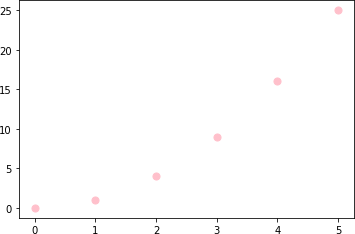


Notice the change in the color argument

#### Scatter Plots:

#create data for plotting x\_values = [0,1,2,3,4,5]

squares = [0,1,4,9,16,25]plt.**scatter**(x\_values,squares, **s=10,** color = "pink") plt.show()

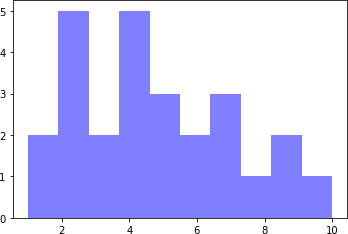
Can you see the pattern? Now the code changed from *plt.bar()* to *plt.scatter().* I also added the **s** argument. The s stands for size, and it allows us to control how big we want the points on the graph.

#### Histograms:

#generate fake data

x = [2,1,6,4,2,4,8,9,4,2,4,10,6,4,5,7,7,3,2,7,5,3,5,9,2,1]#plot for a histogram

plt.**hist**(x, **bins = 10**, color='blue', **alpha=0.5**) plt.show()



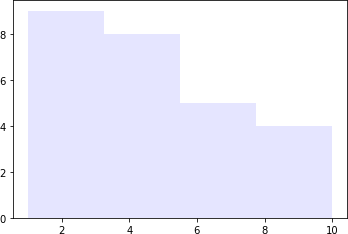
Looking at the code snippet, I added two new arguments.

1. Bins — is an argument specific to a histogram and allows the user to customize how many bins they want.
2. Alpha — is an argument that displays the level of transparency of the data points.

If I were to adjust both the bins and alpha, I would get something like this,

x = [2,1,6,4,2,4,8,9,4,2,4,10,6,4,5,7,7,3,2,7,5,3,5,9,2,1]

num\_bins = 10plt.hist(x, **bins= 4**, color='blue', **alpha=0.1**) plt.show()



#### Overview:

We just touched the surface of the power of matplotlib. Once you dive deeper into this subject, you can see how much customizability you can have creating colorful, detailed, and vibrant graphs.

There are a lot more graphs available in the matplotlib library as well as other popular libraries available in python, including [**seaborn**](https://seaborn.pydata.org/), [**pandas plot**](https://pandas.pydata.org/pandas-docs/stable/user_guide/visualization.html)**,** and [**plotly**](https://plot.ly/python/). It is worth exploring all the different options and finding which library suits your style of coding and analysis.

#### Conclusion:

Matplotlib is a multi-platform **data visualization library built on NumPy arrays** and designed to work with the broader SciPy stack visualization is that it allows us visual access to huge amounts of data in easily digestible visuals.

In this practical we implemented numpy array , line graph , Bar graph , Scatter Plot and Histograms

#### PRACTICAL NO. 6

###### AIM OF PRACTICAL:

Write a program to read a csv file using python's inbuilt module.

###### LEARNING OBJECTIVES:

To understand the Comma Separated Value (CSV), read csv file using python.

###### LEARNING OUTCOMES:

1. This article explains how to load and parse a CSV file in Python.
2. Learned to store tabular data, such as a spreadsheet or database.

###### SOFTWARE/HARDWARE REQUIRED:

|  |  |
| --- | --- |
| **SN** | **Name of Equipment/Items/Software Tools** |
| **1.** | Hardware: Computer System |
| **2.** | Software: Anaconda Navigator (Jupyter Notebook) |

**THEORY:**

**CSV** (Comma Separated Values) is a simple **file format** used to store tabular

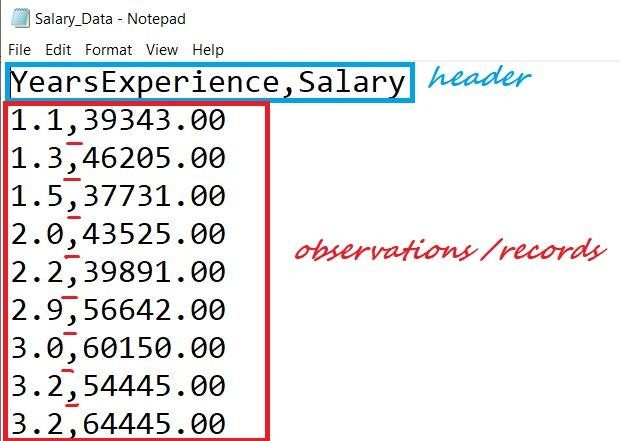
data, such as a spreadsheet or database. A CSV file stores tabular data (numbers and text) in plain text. Each line of the file is a data record. Each record consists of one or more fields, separated by commas. The use of the comma as a field separator is the source of the name for this file format.

For working CSV files in python, there is an inbuilt module called CSV.

###### Why CSV files used?

These files serve a number of different business purposes. They help companies export a high volume of data to a more concentrated database, for instance. CSV files are plain-text files, making them easier for the website developer to create Since they're plain text, they're easier to import into a spreadsheet or another storage database, regardless of the specific software you're using To better organize large amounts of data.

###### Structure of CSV:



We have a file named “Salary\_Data.csv.” The first line of a CSV file is the header and contains the names of the fields/features.

After the header, each line of the file is an observation/a record. The values of a record are separated by “comma.”

###### Reading a CSV Using csv.reader

Reading a CSV using Python’s inbuilt module called csv using csv.reader object.

###### Steps to read a CSV file:

1. Import the csv library.
2. Open the CSV file.

The .open() method in python is used to open files and return a file object.

1. Use the csv.reader object to read the CSV file.
2. Extract the field names, Create an empty list called header. Use the next() method to obtain the header.The .next() method returns the current row and moves to the next row.The first time you run next() it returns the header and the next time you run it returns the first record and so on.
3. Extract the rows/records ,Create an empty list called rows and iterate through the csvreader object and append each row to the rows list.
4. Close the file ,close() method is used to close the opened file. Once it is closed, we cannot perform any operations on it.

###### Program:







**OUTPUT:**



###### CONCLUSION:

In this tutorial at Comma Separated Value, we have learned to open a file to be read in python using inbuilt module csv.

#### PRACTICAL N0. 7

###### AIM OF PRACTICAL:

W.A.P in python using pymongo library to perform basic CRUDE operation using mongo DB

###### LEARNING OBJECTIVES:

To understand pymongo library and we are able to perform CRUDE operation on mongoDB

###### LEARNING OUTCOMES:

1. In this practical we have learn to implement CRUDE operation on mongodb
2. Learned to handle the database of mongodb
3. Learned how to use CRUDE operation on mongoDB

###### SOFTWARE/HARDWARE REQUIRED:

|  |  |
| --- | --- |
| **SN** | **Name of Equipment/Items/Software Tool** |
| **1.** | Hardware: Computer System |
| **2.** | Software: mongoDB database |

**THEORY:**

#### Introduction of CRUDE operation: -

CRUD operations describe the conventions of a user-interface that let users view, search, and modify parts of the database.

MongoDB documents are modified by connecting to a server, querying the proper documents, and then changing the setting properties before sending the data back to the database to be updated. CRUD is data-oriented, and it’s standardized according to HTTP action verbs.

#### CRUD operations:

* The Create operation is used to insert new documents in the MongoDB database.
* The Read operation is used to query a document in the database.
* The Update operation is used to modify existing documents in the database.
* The Delete operation is used to remove documents in the database.

# Create Operations:-

Mongo store the data in the form of JSON objects. So every record for a collection in mongo is called a document. If the collection does not currently exist, insert operations will create the collection. We can insert the documents into collection in 3 ways

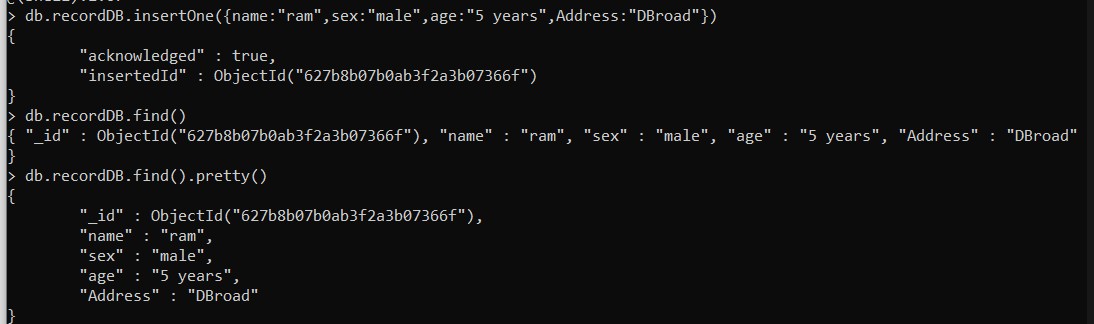
###### inseítone()

* **inseítmany()**

**insertOne()**

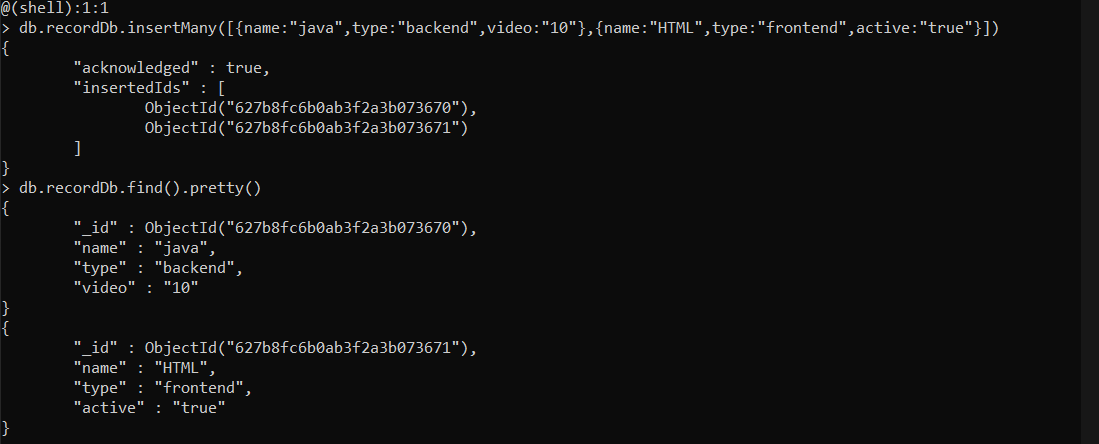
As the namesake, insertOne() allows you to insert one document into the collection. For this example, we’re going to work with a collection called RecordsDB. We can insert a single entry into our collection by calling the insertOne() method on

RecordsDB. We then provide the information we want to insert in the form of key-value pairs, establishing the schema.



#### insertMany()

It’s possible to insert multiple items at one time by calling the *insertMany()* method on the desired collection. In this case, we pass multiple items into our chosen collection (*RecordsDB*) and separate them by commas. Within the parentheses, we use brackets to indicate that we are passing in a list of multiple entries. This is commonly referred to as a nested method.



#### Read Operations

The Read operations allow you to supply special query filters and criteria that let you specify which documents you want. The MongoDB documentation contains more information on the available query Filter. Query modifiers may also be used to change how many results are returned.

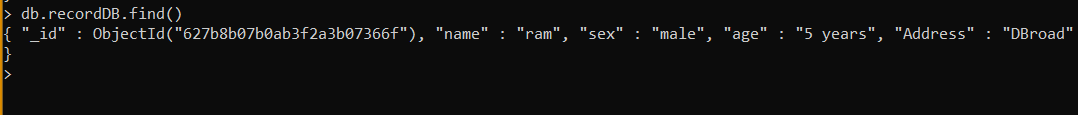
MongoDB has two methods of reading documents from a collection:

###### find()

* **findOne()**

### find()

In order to get all the documents from a collection, we can simply use the *find()* method on our chosen collection. Executing just the *find()* method with no arguments will return all records currently in the collection.



###### findOne()

In order to get one document that satisfies the search criteria, we can simply use the *findOne()* method on our chosen collection. If multiple documents satisfy the query,this method returns the first document according to the natural order which reflects the order of documents on the disk. If no documents satisfy the search criteria, the function returns null. The function takes the following form of syntax.

# Update Operations

Like create operations, update operations operate on a single collection, and they are atomic at a single document level. An update operation takes filters and criteria to select the documents you want to update.

You should be careful when updating documents, as updates are permanent and can’t be rolled back. This applies to delete operations as well.

For MongoDB CRUDE, there are three different methods of updating documents:

###### updateone()

* **updatemany()**
* **íeplaceone()**

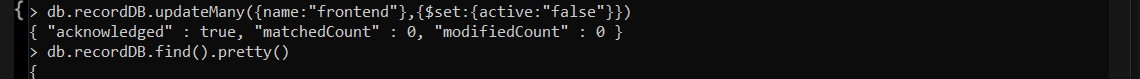
**updateone()**

We can update a currently existing record and change a single document with an update operation. To do this, we use the *updateOne()* method on a chosen collection, which here is “RecordsDB.” To update a document, we provide the method with two arguments: an update filter and an update action.

The update filter defines which items we want to update, and the update action defines how to update those items. We first pass in the update filter. Then, we use the “$set” key and provide the fields we want to update as a value. This method will update the first record that matches the provided filter.

### updateMany()

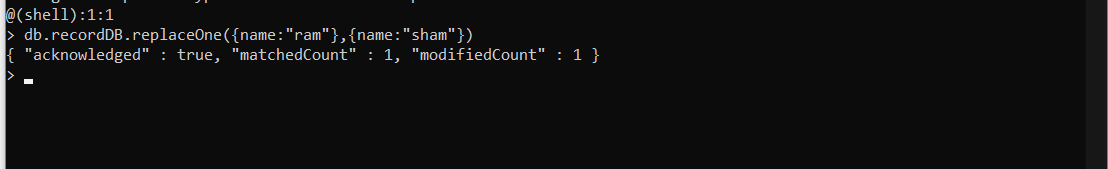
*updateMany()* allows us to update multiple items by passing in a list of items, just as we did when inserting multiple items. This update operation uses the same syntax for updating a single document.



### ReplaceOne()

The *replaceOne()* method is used to replace a single document in the specified

collection. *replaceOne()* replaces the entire document, meaning fields in the old document not contained in the new will be lost.



# Delete Operations

Delete operations operate on a single collection, like update and create operations. Delete operations are also atomic for a single document. You can provide delete operations with filters and criteria in order to specify which documents you would like to delete from a collection. The filter options rely on the same syntax that read operations utilize.

MongoDB has two different methods of deleting records from a collection:

* deleteone()
* deletemany()

## deleteOne()

*deleteOne()* is used to remove a document from a specified collection on the MongoDB server. A filter criteria is used to specify the item to delete. It deletes the first record that matches the provided filter

***deleteMany()***

*deleteMany()* is a method used to delete multiple documents from a desired collection with a single delete operation. A list is passed into the method and the individual items are defined with filter criteria as in *deleteOne()*.

**Conclusion**:-In this practicle,We have to learned PyMongo, CRUDE Operation, to connect Python code to MongoDB, as well as creating, retrieving, updating, and deleting document.

PRACTICAL NO.8

# AIM OF PRACTICAL:

W.A.P. in python to learn and implement decision tree classification, attribute selection measures, and how to build and optimize decision tree classifier using python scikit-learn package.

# LEARNING OBJECTIVES:

To understand decision tree classification, it's attribute selection measures and to build and optimize decision tree classifier using python scikit - learn package

# LEARNING OUTCOMES:

1. In this practical we have learn to implement decision tree classification
2. Learned to optimize decision tree classifier
3. Learned how to use Scikit-learn package

# SOFTWARE/HARDWARE REQUIRED:

|  |  |
| --- | --- |
| **SN** | **Name of Equipment/Items/Software Tool** |
| **1.** | Hardware: Computer System |
| **2.** | Software: Anaconda Navigator (Jupyter Notebook) |

**THEORY:**

# Introduction to Decision Tree :-

Formally a decision tree is a **graphical representation of all possible solutions to a decision**. These days, tree-based algorithms are the most commonly used algorithms in the case of supervised learning scenarios. They are easier to interpret and visualize with great adaptability. We can use tree-based algorithms for both regression and classification problems, However, most of the time they are used for classification problem.

# Attribute Selection Measures :-

Attribute selection measure is a heuristic for selecting the splitting criterion that partition data into the best possible manner. It is also known as splitting rules because it helps us to determine breakpoints for tuples on a given node. ASM provides a rank to each feature(or attribute) by explaining the given dataset. Best score attribute will be selected as a splitting attribute (Source). In the case of a continuous-valued attribute, split points for branches also need to define. Most popular selection measures are Information Gain, Gain Ratio, and Gini Index.

1. Entropy: Entropy is the measure of uncertainty or randomness in a data set. Entropy handles how a decision tree splits the data.

It is calculated using the following formula:



1. Information Gain: The information gain measures the decrease in entropy after the data set is split.

It is calculated as follows:

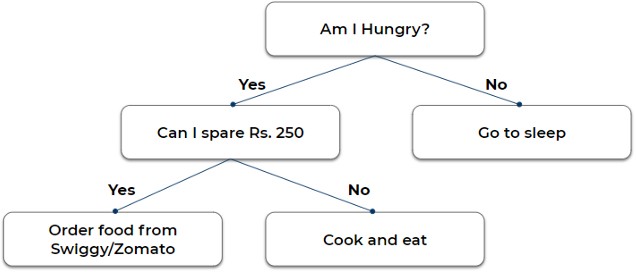
IG( Y, X) = Entropy (Y) - Entropy ( Y | X)

1. Gini Index: The Gini Index is used to determine the correct variable for splitting nodes. It measures how often a randomly chosen variable would be incorrectly identified.
2. Root Node: The root node is always the top node of a decision tree. It represents the entire population or data sample, and it can be further divided into different sets.
3. Decision Node: Decision nodes are subnodes that can be split into different subnodes; they contain at least two branches.
4. Leaf Node: A leaf node in a decision tree carries the final results. These nodes, which are also known as terminal nodes, cannot be split any further.

# Advantages of Using Decision Trees

* + Decision trees are simple to understand, interpret, and visualize
  + They can effectively handle both numerical and categorical data
  + They can determine the worst, best, and expected values for several scenarios

**Decision Tree Example** :- Yesterday evening, I *skipped dinner* at my usual time because I was busy taking care of some stuff. Later in the night, I felt butterflies in my stomach. I thought only if *I wasn’t hungry*, I could have gone to sleep as it is but as that was not the case, *I decided to eat* something. I had two options, to *order something from outside* or cook myself. I figured if I order, I will have to spare at least *INR 250* on it. I finally decided to order it anyway as it was pretty late and I was in no mood of cooking. This complete incident can be graphically represented as shown in the following figure.



This representation is nothing but a decision tree.

# Building a Decision Tree in Python

We’ll now predict if a consumer is likely to repay a loan using the decision tree algorithm in Python. The data set contains a wide range of information for making this prediction, including the initial payment amount, last payment amount, credit score, house number, and whether the individual was able to repay the loan.

In Scikit-learn, optimization of decision tree classifier performed by only pre-pruning. Maximum depth of the tree can be used as a control variable for pre-pruning.

# PROGRAM & OUTPUT:

In [1]:

**import** pandas **as** pd

In [2]:

**from** sklearn.tree **import** DecisionTreeClassifier *# Import Decision Tree Classifier*

In [3]:

**from** sklearn.model\_selection **import** train\_test\_split *# Import train\_test\_split function*

In [4]:

**from** sklearn **import** metrics *#Import scikit-learn metrics module for accuracy calculation*

In [5]:

col\_names **=** ['pregnant', 'glucose', 'bp', 'skin', 'insulin', 'bmi', 'pedigree', 'age', 'label']

*# load dataset*

In [8]:

pima **=** pd.read\_csv("pima-indians-diabetes.csv", header**=None**, names**=**col\_names)

In [9]:

pima.head()

Out[9]:

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **pregnant** | **glucose** | **bp** | **skin** | **insulin** | **bmi** | **Pedigree** | **age** | **label** |
| **0** 6 | 148 | 72 | 35 | 0 | 33.6 | 0.627 | 50 | 1 |
| **1** 1 | 85 | 66 | 29 | 0 | 26.6 | 0.351 | 31 | 0 |
| **2** 8 | 183 | 64 | 0 | 0 | 23.3 | 0.672 | 32 | 1 |
| **3** 1 | 89 | 66 | 23 | 94 | 28.1 | 0.167 | 21 | 0 |
| **4** 0 | 137 | 40 | 35 | 168 | 43.1 | 2.288 | 33 | 1 |

In [10]:

*#split dataset in features and target variable*

feature\_cols **=** ['pregnant', 'insulin', 'bmi', 'age','glucose','bp','pedigree']

X **=** pima[feature\_cols] *# Features*

y **=** pima.label *# Target variable*

In [11]:

*# Split dataset into training set and test set*

X\_train, X\_test, y\_train, y\_test **=** train\_test\_split(X, y, test\_size**=**0.3, random\_state**=**1)

*#70% training and 30% test*

In [12]:

*# Create Decision Tree classifer object*

clf **=** DecisionTreeClassifier()

*# Train Decision Tree Classifer*

clf **=** clf.fit(X\_train,y\_train)

*#Predict the response for test dataset*

y\_pred **=** clf.predict(X\_test)

In [13]:

*# Model Accuracy, how often is the classifier correct?*

print("Accuracy:",metrics.accuracy\_score(y\_test, y\_pred))

Accuracy: 0.670995670995671

In [14]:

**import** six

**import** sys sys.modules['sklearn.externals.six'] **=** six **from** sklearn.tree **import** export\_graphviz **from** sklearn.externals.six **import** StringIO **from** IPython.display **import** Image

**import** pydotplus

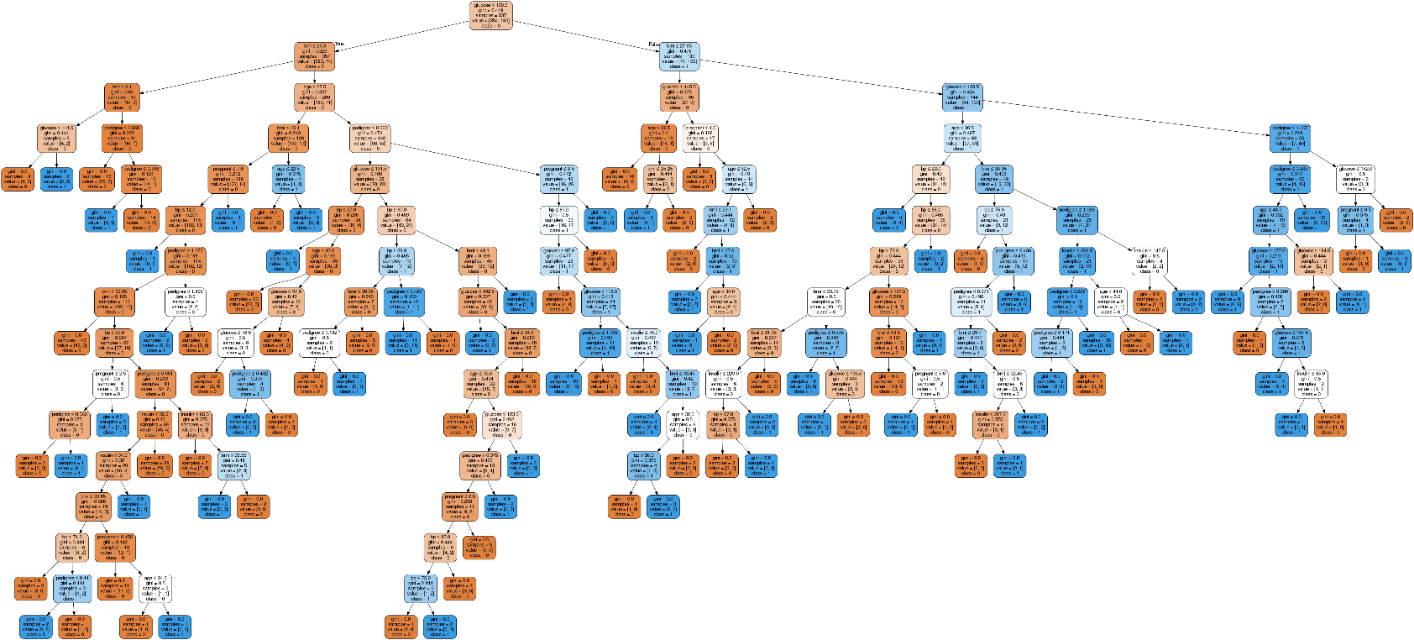
dot\_data **=** StringIO() export\_graphviz(clf, out\_file**=**dot\_data,

filled**=True**, rounded**=True**,

special\_characters**=True**,feature\_names **=** feature\_cols,class\_names**=**['0','1']) graph **=** pydotplus.graph\_from\_dot\_data(dot\_data.getvalue())

graph.write\_png('diabetes.png') Image(graph.create\_png())

Out[14]:



In [15]:

*# Create Decision Tree classifer object*

clf **=** DecisionTreeClassifier(criterion**=**"entropy", max\_depth**=**3)

*# Train Decision Tree Classifer*

clf **=** clf.fit(X\_train,y\_train)

*#Predict the response for test dataset*

y\_pred **=** clf.predict(X\_test)

*# Model Accuracy, how often is the classifier correct?*

print("Accuracy:",metrics.accuracy\_score(y\_test, y\_pred))

Accuracy: 0.7705627705627706

In [16]:

**from** sklearn.externals.six **import** StringIO

**from** IPython.display **import** Image

**from** sklearn.tree **import** export\_graphviz

**import** pydotplus dot\_data **=** StringIO()

export\_graphviz(clf, out\_file**=**dot\_data,

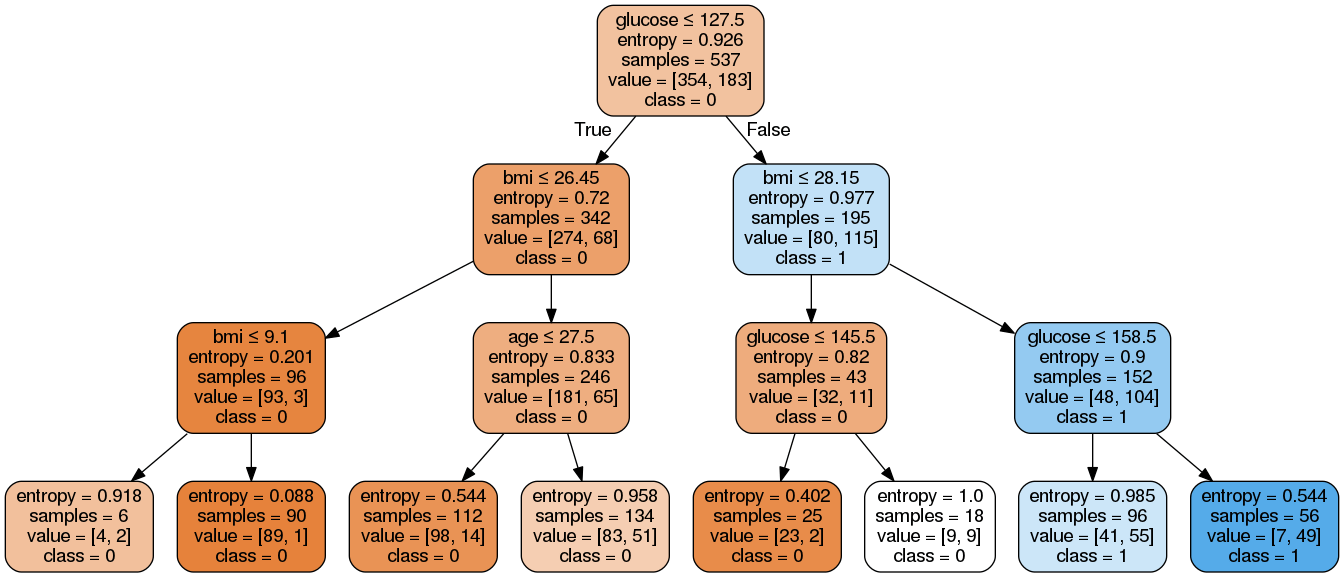
filled**=True**, rounded**=True**,

special\_characters**=True**, feature\_names **=** feature\_cols,class\_names**=**['0','1']) graph **=** pydotplus.graph\_from\_dot\_data(dot\_data.getvalue())

graph.write\_png('diabetes.png')

Image(graph.create\_png())

Out[16]:



# CONCLUSION:

In this practical, we learned a lot of details about Decision Tree; It's working, attribute selection measures such as Information Gain, Gain Ratio, and Gini Index, decision tree model building, visualization and evaluation on diabetes dataset using Python Scikit-learn package. Also, discussed optimizing Decision Tree performance using parameter tuning.

#### PRACTICAL N0. 9

###### AIM OF PRACTICAL:

W.A.P in python to implement Support vector Machine.

###### LEARNING OBJECTIVES:

To study Support vector Machine

To find a hyperplane in an N-dimensional space that distinctly classifies the data points. Predict if cancer is benign or malignant.

###### LEARNING OUTCOMES:

Understand concept of Support vector Machine.

The data taken and pre-processing methods to make optimal hyperplanes using matplotlib function.

###### SOFTWARE/HARDWARE REQUIRED:

|  |  |
| --- | --- |
| **SN** | **Name of Equipment/Items/Software Tool** |
| **1.** | Hardware: Computer System |
| **2.** | Software: Anaconda Navigator (Jupyter Notebook) |

**THEORY:**

Support Vector Machine(SVM) is a supervised machine learning algorithm used for both classification and regression. Though we say regression problems as well its best suited for classification. The objective of SVM algorithm is to find a hyperplane in an N-dimensional space that distinctly classifies the data points. The dimension of the hyperplane depends upon the number of features. If the number of input features is two, then the hyperplane is just a line. If the number of input features is three, then the hyperplane becomes a 2-D plane. It becomes difficult to imagine when the number of features exceeds three.

Support Vector Machine or SVM is one of the most popular Supervised Learning algorithms, which is used for Classification as well as Regression problems. However, primarily, it is used for Classification problems in Machine Learning.The goal of the SVM algorithm is to create the best line or decision boundary that can segregate n- dimensional space into classes so that we can easily put the new data point in the correct category in the future. This best decision boundary is called a hyperplane.SVM chooses the extreme points/vectors that help in creating the hyperplane. These extreme cases are called as support vectors, and hence algorithm is termed as Support Vector Machine. Consider the below diagram in which there are two different categories that are classified using a decision boundary or hyperplane:

**Advantages of SVM:**

* Effective in high dimensional cases
* Its memory efficient as it uses a subset of training points in the decision function called support vectors
* Different kernel functions can be specified for the decision functions and its possible to specify custom kernels

##### Program:-

# importing required libraries

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

# reading csv file and extracting class column to y. x = pd.read\_csv("C:\...\cancer.csv")

a = np.array(x)

y = a[:,30] # classes having 0 and 1

# extracting two features

x = np.column\_stack((x.malignant,x.benign))

# 569 samples and 2 features x.shape

print (x),(y)

**Output:-**

|  |  |  |
| --- | --- | --- |
| [[ | 122.8 1001. | ] |
| [ | 132.9 1326. | ] |
| [ | 130. 1203. | ] |

...,

[ 108.3 858.1 ]

[ 140.1 1265. ]

[ 47.92 181. ]]

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| array([ 0.,  0., | 0., | 0., | 0., | 0., | 0., | 0., | 0., | 0., | 0., | 0., | 0., |
| 0., | 0., | 0., | 0., | 0., | 0., | 1., | 1., | 1., | 0., | 0., | 0., |
| 0., |  |  |  |  |  |  |  |  |  |  |  |
| 0., | 0., | 0., | 0., | 0., | 0., | 0., | 0., | 0., | 0., | 0., | 1., |
| 0., |  |  |  |  |  |  |  |  |  |  |  |
| 0., | 0., | 0., | 0., | 0., | 0., | 0., | 1., | 0., | 1., | 1., | 1., |
| 1., |  |  |  |  |  |  |  |  |  |  |  |
| 1.,  ...., | 0., | 0., | 1., | 0., | 0., | 1., | 1., | 1., | 1., | 0., | 1., |

1.])

###### CONCLUSION:

* + Many software tools are available for SVM implementation.
  + SVMs are really good for text classification.
  + SVMs are good at finding the best linear separator. The kernel trick makes SVMs non- linear learning algorithms.
  + Choosing an appropriate kernel is the key for good SVM and choosing the right kernel function is not easy.
  + We need to be patient while building SVMs on large datasets. They take a lot of time for training

#### PRACTICAL N0. 10

###### AIM OF PRACTICAL:

To implement the naïve bayes theorem by using python.

###### LEARNING OBJECTIVES:

1. To understand Naive Bayes Theorem.
2. To understand the posteriori probability given to training data.

###### LEARNING OUTCOMES:

To predict the probability of different classes based on various attributes.

###### SOFTWARE/HARDWARE REQUIRED:

|  |  |
| --- | --- |
| **SN** | **Name of Equipment/Items/Software Tool** |
| **1.** | Hardware: Computer System |
| **2.** | Software: Anaconda Navigator (Jupyter Notebook) |

**THEORY:**

###### Introduction to Naive Bayes

Naive Bayes is among one of the very simple and powerful algorithms for classification based on **Bayes Theorem** with an assumption of independence among the predictors. The Naive Bayes classifier assumes that the presence of a feature in a class is not related to any other feature. Naive Bayes is a classification algorithm for binary and multi-class classification problems.

###### Bayes Theorem

* Based on prior knowledge of conditions that may be related to an event, Bayes theorem describes the probability of the event
* conditional probability can be found this way
* Assume we have a Hypothesis(*H*) and evidence(*E*),

According to Bayes theorem, the relationship between the probability of

Hypothesis before getting the evidence represented as *P(H)* and the probability of the hypothesis after getting the evidence represented as *P(H|E)* is:

*P(H|E) = P(E|H)\*P(H)/P(E)*

* **Prior probability** = *P(H)* is the probability before getting the evidence

**Posterior probability** = *P(H|E)* is the probability after getting evidence

* In general,

*P(class|data) = (P(data|class) \* P(class)) / P(data)*

###### Bayes Theorem Example

Assume we have to find the probability of the randomly picked card to be king given that it is a face card.

There are *4* Kings in a Deck of Cards which implies that *P(King) = 4/52*

as all the Kings are face Cards so *P(Face|King) = 1*

there are *3* Face Cards in a Suit of *13 cards* and there are *4 Suits* in total so *P(Face) = 12/52*

Therefore,

*P(King|face) = P(face|king) \* P(king)/P(face) = 1/3*

###### PROGRAM:

# Importing library

**import** math **import** random **import** csv

# the categorical class names are changed to numberic data # eg: yes and no encoded to 1 and 0

**def** encode\_class(mydata): classes **=** []

**for** i **in** range(len(mydata)):

**if** mydata[i][**-**1] **not in** classes: classes.append(mydata[i][**-**1])

**for** i **in** range(len(classes)):

**for** j **in** range(len(mydata)):

**if** mydata[j][**-**1] **==** classes[i]: mydata[j][**-**1] **=** i

**return** mydata

# Splitting the data

**def** splitting(mydata, ratio):

train\_num **=** int(len(mydata) **\*** ratio) train **=** []

# initially testset will have all the dataset test **=** list(mydata)

**while** len(train) < train\_num:

# index generated randomly from range 0 # to length of testset

index **=** random.randrange(len(test))

# from testset, pop data rows and put it in train train.append(test.pop(index))

**return** train, test

# Group the data rows under each class yes or # no in dictionary eg: dict[yes] and dict[no] **def** groupUnderClass(mydata):

dict **=** {}

**for** i **in** range(len(mydata)):

**if** (mydata[i][**-**1] **not in** dict): dict[mydata[i][**-**1]] **=** []

dict[mydata[i][**-**1]].append(mydata[i])

**return** dict # Calculating Mean **def** mean(numbers):

**return** sum(numbers) **/** float(len(numbers))

# Calculating Standard Deviation

**def** std\_dev(numbers): avg **=** mean(numbers)

variance **=** sum([pow(x **-** avg, 2) **for** x **in** numbers]) **/**

float(len(numbers) **-** 1)

**return** math.sqrt(variance)

**def** MeanAndStdDev(mydata):

info **=** [(mean(attribute), std\_dev(attribute)) **for** attribute **in**

zip(**\***mydata)]

# eg: list = [ [a, b, c], [m, n, o], [x, y, z]]

# here mean of 1st attribute =(a + m+x), mean of 2nd attribute = (b + n+y)/3

# delete summaries of last class

**del** info[**-**1]

**return** info

# find Mean and Standard Deviation under each class

**def** MeanAndStdDevForClass(mydata): info **=** {}

dict **=** groupUnderClass(mydata)

**for** classValue, instances **in** dict.items(): info[classValue] **=** MeanAndStdDev(instances)

**return** info

# Calculate Gaussian Probability Density Function

**def** calculateGaussianProbability(x, mean, stdev):

expo **=** math.exp(**-**(math.pow(x **-** mean, 2) **/** (2 **\*** math.pow(stdev, 2))))

**return** (1 **/** (math.sqrt(2 **\*** math.pi) **\*** stdev)) **\*** expo

# Calculate Class Probabilities

**def** calculateClassProbabilities(info, test): probabilities **=** {}

**for** classValue, classSummaries **in** info.items(): probabilities[classValue] **=** 1

**for** i **in** range(len(classSummaries)): mean, std\_dev **=** classSummaries[i] x **=** test[i]

probabilities[classValue] **\*=** calculateGaussianProbability(x, mean, std\_dev)

**return** probabilities

# Make prediction - highest probability is the prediction

**def** predict(info, test):

probabilities **=** calculateClassProbabilities(info, test) bestLabel, bestProb **=** None, **-**1

**for** classValue, probability **in** probabilities.items():

**if** bestLabel **is** None **or** probability > bestProb: bestProb **=** probability

bestLabel **=** classValue

**return** bestLabel

# returns predictions for a set of examples

**def** getPredictions(info, test): predictions **=** []

**for** i **in** range(len(test)):

result **=** predict(info, test[i]) predictions.append(result)

**return** predictions

# Accuracy score

**def** accuracy\_rate(test, predictions): correct **=** 0

**for** i **in** range(len(test)):

**if** test[i][**-**1] **==** predictions[i]: correct **+=** 1

**return** (correct **/** float(len(test))) **\*** 100.0 # driver code

# add the data path in your system

filename **=** r'E:\user\MACHINE LEARNING\machine learning algos\Naive bayes\filedata.csv'

# load the file and store it in mydata list mydata **=** csv.reader(open(filename, "rt")) mydata **=** list(mydata)

mydata **=** encode\_class(mydata)

**for** i **in** range(len(mydata)):

mydata[i] **=** [float(x) **for** x **in** mydata[i]]

# split ratio = 0.7

# 70% of data is training data and 30% is test data used for testing ratio **=** 0.7

train\_data, test\_data **=** splitting(mydata, ratio)

**print**('Total number of examples are: ', len(mydata))

print('Out of these, training examples are: ', len(train\_data))

**print**("Test examples are: ", len(test\_data))

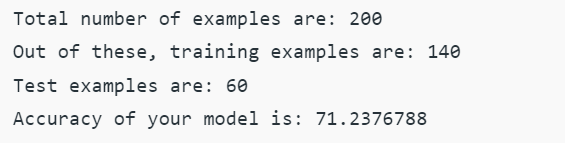
# prepare model

info **=** MeanAndStdDevForClass(train\_data)

# test model

predictions **=** getPredictions(info, test\_data) accuracy **=** accuracy\_rate(test\_data, predictions) **print**("Accuracy of your model is: ", accuracy)

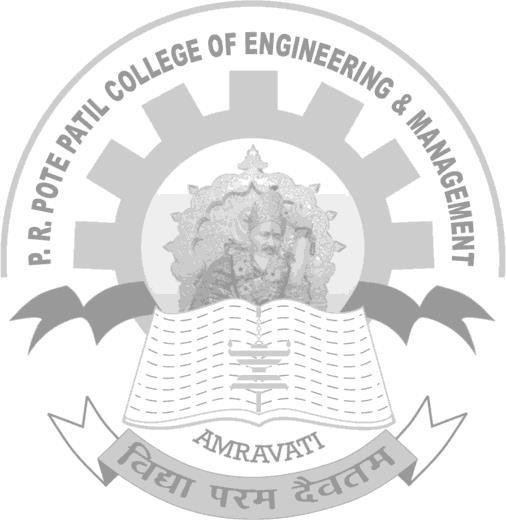
###### OUTPUT:



###### CONCLUSION:

Naive Bayes algorithms are mostly used in sentiment analysis, spam filtering, recommendation systems etc. They are fast and easy to implement but their biggest disadvantage is that the requirement of predictors to be independent.

**C-Lab II (Hardware Lab)**



**SOFTWARE / HARDWARE REQUIRED :-**

## PRACTICAL NO. 11

|  |
| --- |
| **AIM OF PRACTICAL :** |
| Write a Program in python to implement K-means algorithm. |
| **LEARNING OBJECTIVES: -** |
| 1. Understanding the K-Means algorithm. 2. Minimize the sum of distances between the data point and their corresponding clusters. |
| **LEARNING OUTCOMES: -** |
| 1. Understanding the K-Means algorithm. 2. Minimize the sum of distances between the data point and their corresponding clusters. |

|  |  |
| --- | --- |
| **SN** | **Name of Equipment / Items / Software Tool** |
| **1.** | Hardware : Computer System |
| **2.** | Operating System |

PRPCEM, Amravati

**C-Lab II (Hardware Lab)**

**THEORY: -**

K-means clustering algorithm computes the centroids and iterates until we it finds optimal centroid. It assumes that the number of clusters are already known. It is also called **flat clustering** algorithm. The number of clusters identified from data by algorithm is represented by ‘K’ in K-means.

In this algorithm, the data points are assigned to a cluster in such a manner that the sum of the squared distance between the data points and centroid would be minimum. It is to be understood that less variation within the clusters will lead to more similar data points within same cluster.

**WORKING OF K-MEANS ALGORITHM**

We can understand the working of K-Means clustering algorithm with the help of following steps −

**Step 1** − First, we need to specify the number of clusters, K, need to be generated by this algorithm.

**Step 2** − Next, randomly select K data points and assign each data point to a cluster. In simple words, classify the data based on the number of data points.

**Step 3** − Now it will compute the cluster centroids.

**Step 4** − Next, keep iterating the following until we find optimal centroid which is the assignment of data points to the clusters that are not changing any more

* + **4.1** − First, the sum of squared distance between data points and centroids would be computed.
  + **4.2** − Now, we have to assign each data point to the cluster that is closer than other cluster (centroid).
  + **4.3** − At last compute the centroids for the clusters by taking the average of all data points of that cluster.

PRPCEM, Amravati

**C-Lab II (Hardware Lab)**

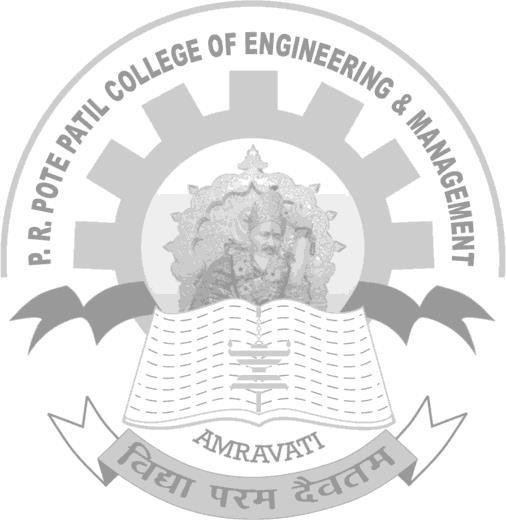
**PROGRAM: -**

import math

import numpy as np import pandas as pd np.random.seed(123)

def circulo(num\_datos = 100,R = 1, minimo = 0,maximo= 1, center\_x = 0 , center\_y

= 0):



pi = math.pi

r = R \* np.sqrt(np.random.uniform(minimo, maximo, size = num\_datos)) theta = np.random.uniform(minimo, maximo, size= num\_datos) \* 2 \* pi

x = center\_x + np.cos(theta) \* r y = center\_y + np.sin(theta) \* r

x = np.round(x,3) y = np.round(y,3)

we have studied

df = np.column\_stack([x,y]) df = pd.DataFrame(df) df.columns = ['x','y'] return(df)

the introduction

# Create data

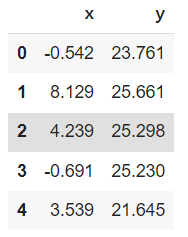
datos\_1 = circulo(num\_datos = 20,R = 10, center\_x = 5, center\_y = 30) datos\_2 = circulo(num\_datos = 20,R = 10, center\_x = 20, center\_y = 10) datos\_3 = circulo(num\_datos = 20,R = 10, center\_x = 50, center\_y = 50)

data = datos\_1.append(datos\_2).append(datos\_3) data.head()

PRPCEM, Amravati

**C-Lab II (Hardware Lab)**

**OUTPUT:-**



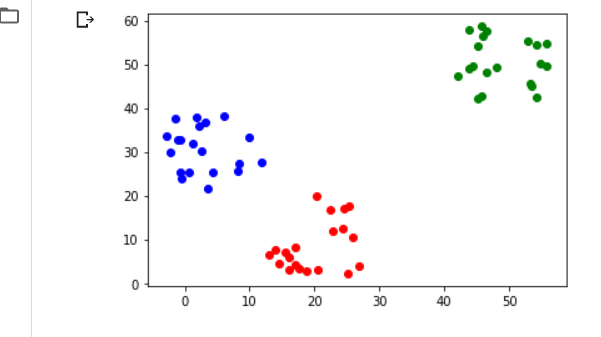
import matplotlib.pyplot as plt

%matplotlib inline

plt.scatter(datos\_1['x'], datos\_1['y'], c = 'b')

plt.scatter(datos\_2['x'], datos\_2['y'], c = 'r')

plt.scatter(datos\_3['x'], datos\_3['y'], c = 'g') plt.show()



**CONCLUSION: -** Thus we have studied implement K-means algorithm Successfully.

PRPCEM, Amravati