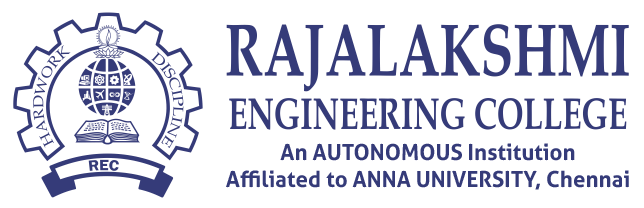
**RAJALAKSHMI ENGINEERING COLLEGE (Autonomous)**

**RAJALAKSHMI NAGAR, THANDALAM, CHENNAI-602105**

**DEPARTMENT OF COMPUTER SCIENCE AND**

**ENGINEERING**



**AI19341**

**PRINCIPLES OF ARTIFICIAL INTELLIGENCE LAB**

**THIRD YEAR**

**FIFTH SEMESTER**

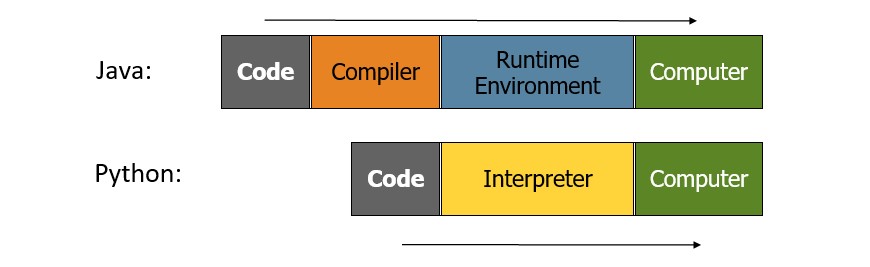
**INDEX**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **S.NO** | **DATE** | **EXP NAME** | **VIVA MARK** | **SIGNATURE** |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

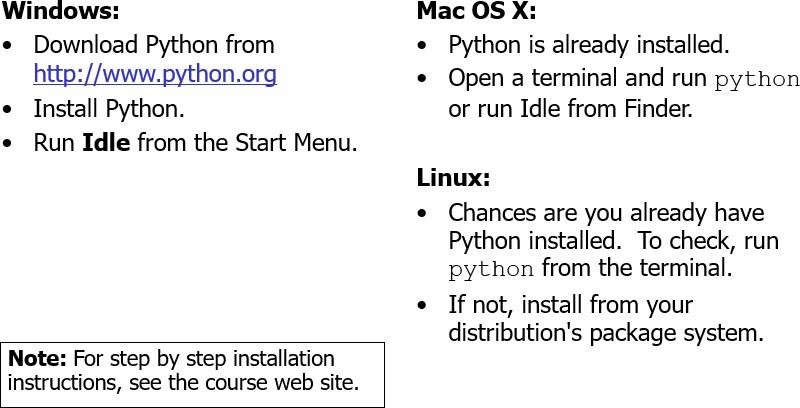
**Working Tools and Language**

PYTHON LANGUAGE

* Interpreter Languages
* Interpreted
* Not compiled like Java
* Code is written and then directly executed by an interpreter
* Type commands into interpreter and see immediate results



# Python Installation Steps

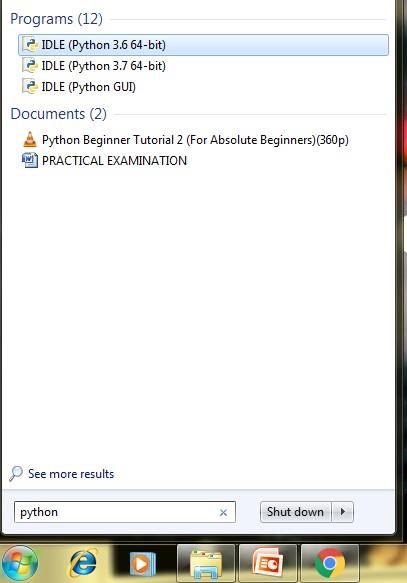


Step 1: Download the Python 3 Installer

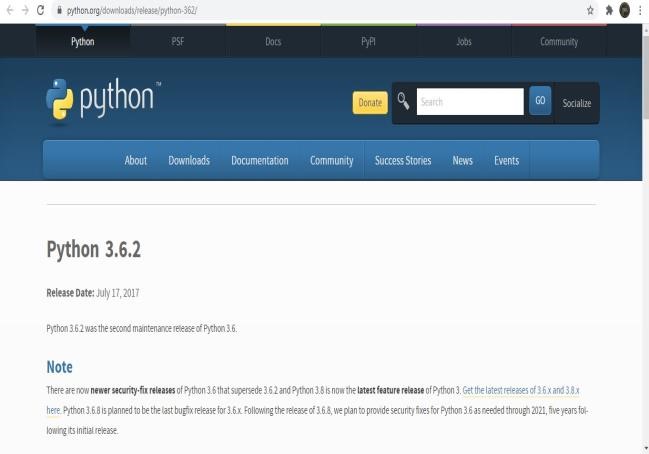
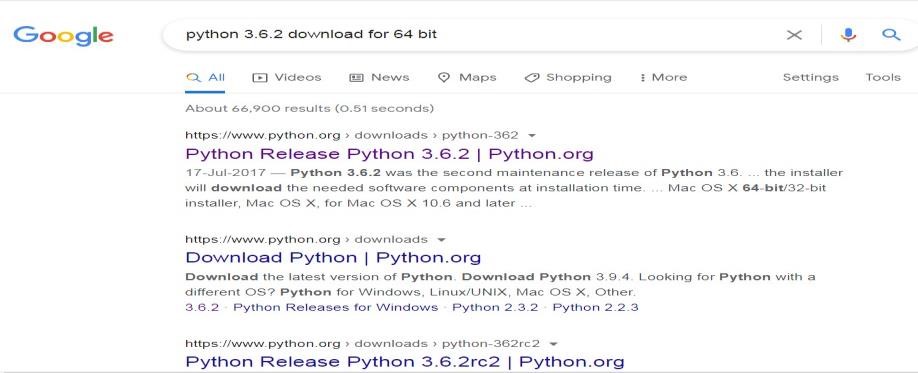
* [Open a browser window and navigate to the Download page for Windows at](https://www.python.org/downloads/windows/) [python.or](https://www.python.org/)[g.](https://www.python.org/downloads/windows/)
* Underneath the heading at the top that says Python Releases for Windows, click on the link for the Latest Python 3 Release – Python 3.x.x. (As of this writing, the latest version is Python 3.7.2.)
* Scroll to the bottom and select either Windows x86-64 executable installer for 64-bit or Windows x86 executable installer for 32-bit.

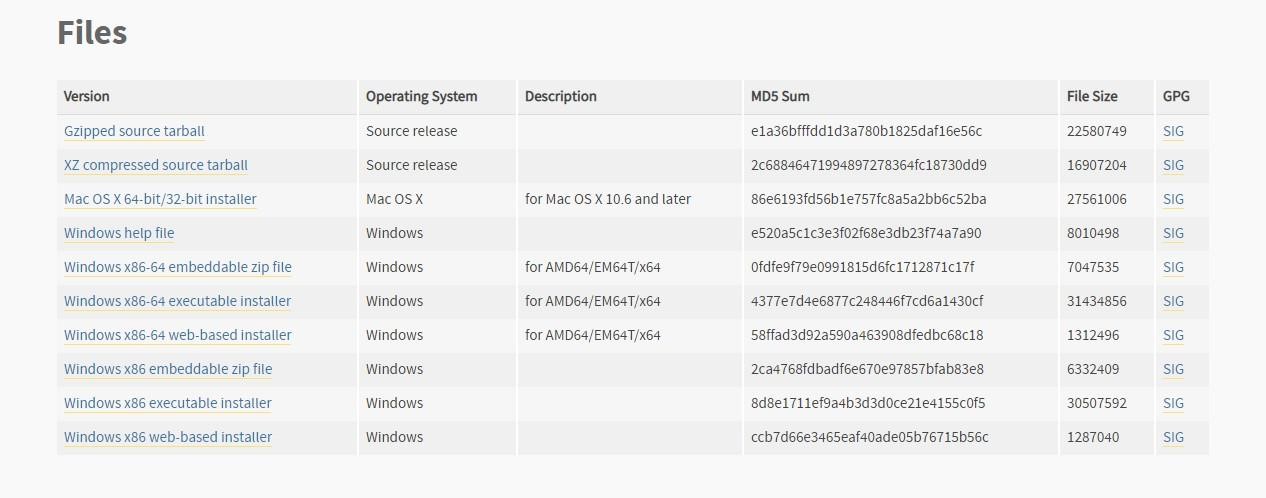
Step 2: Run the Installer

* Once you have chosen and downloaded an installer, simply run it by double-clicking on the downloaded file. A dialog should appear that looks something like this:

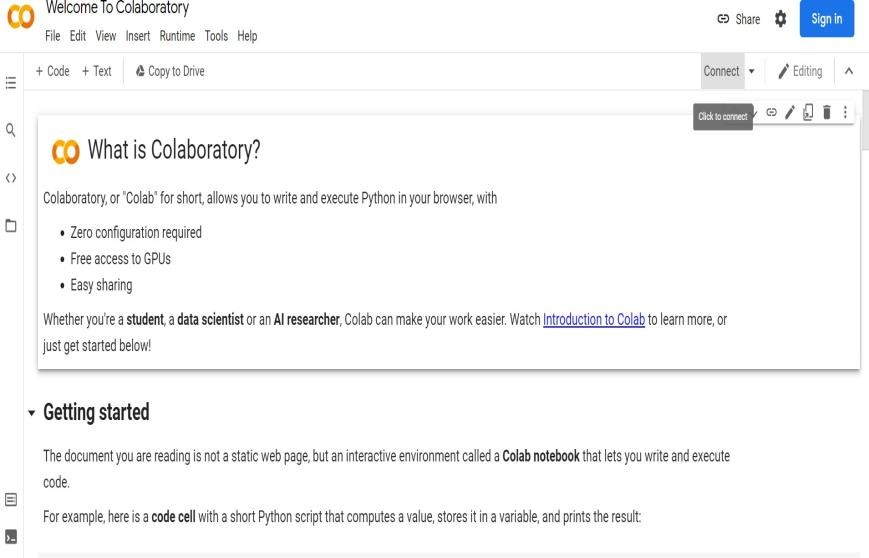
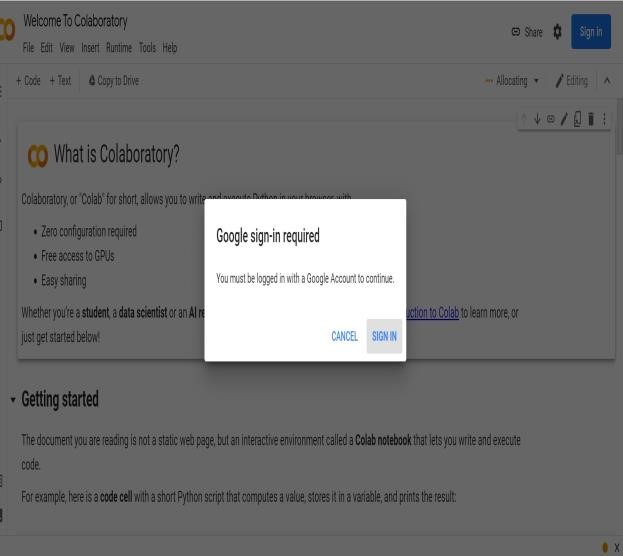
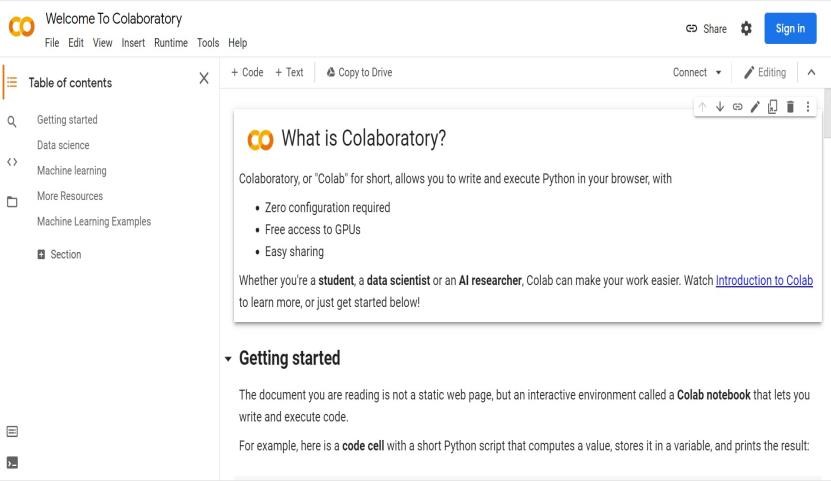
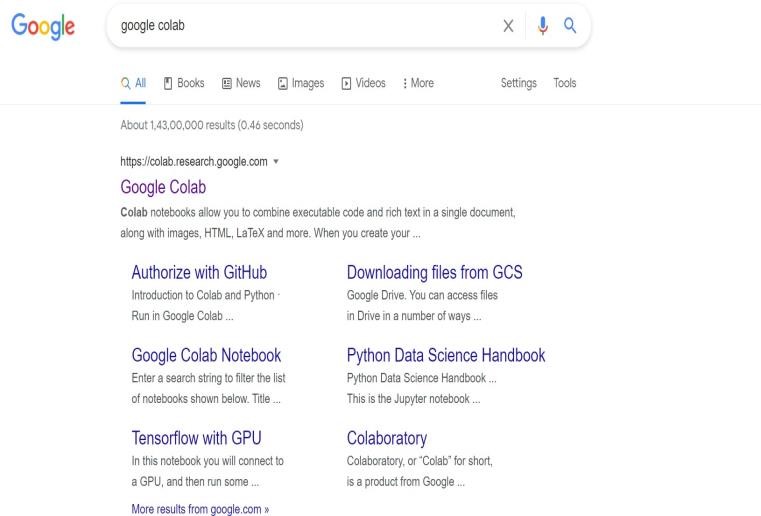


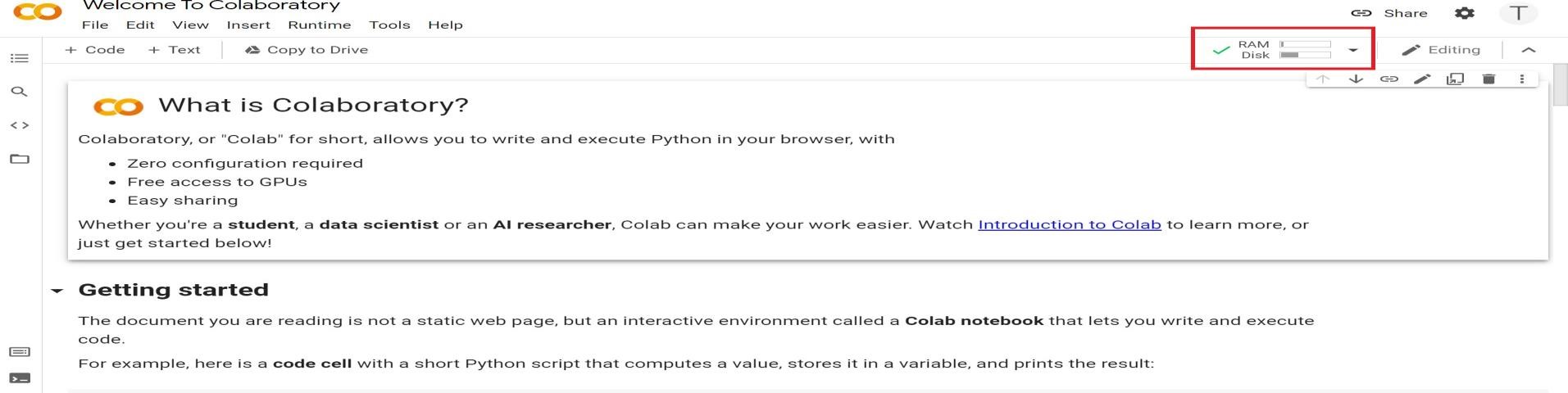
# INSTALLATION STEPS ON PYTHON

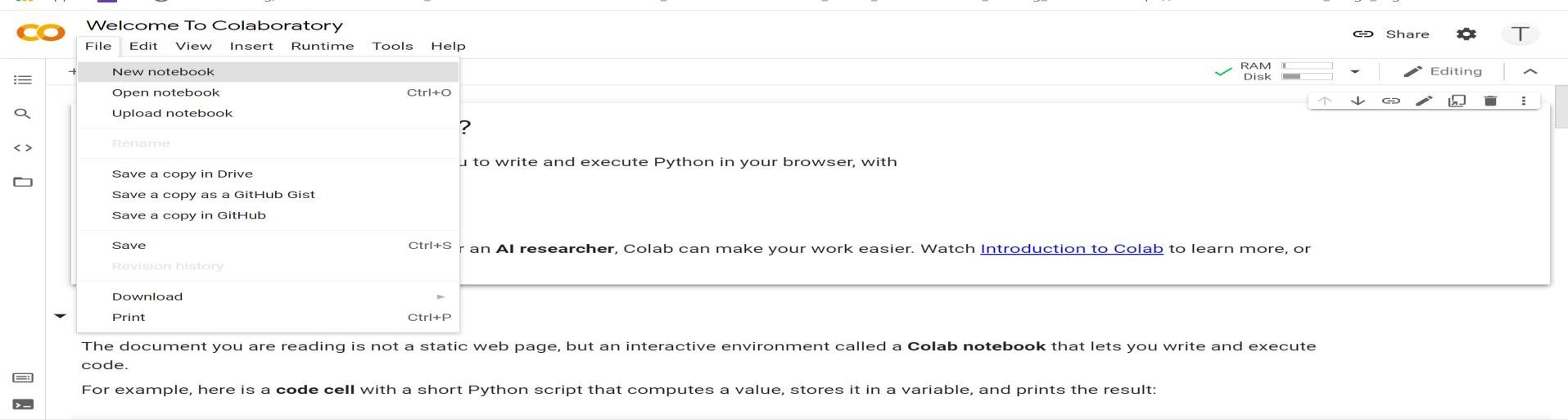


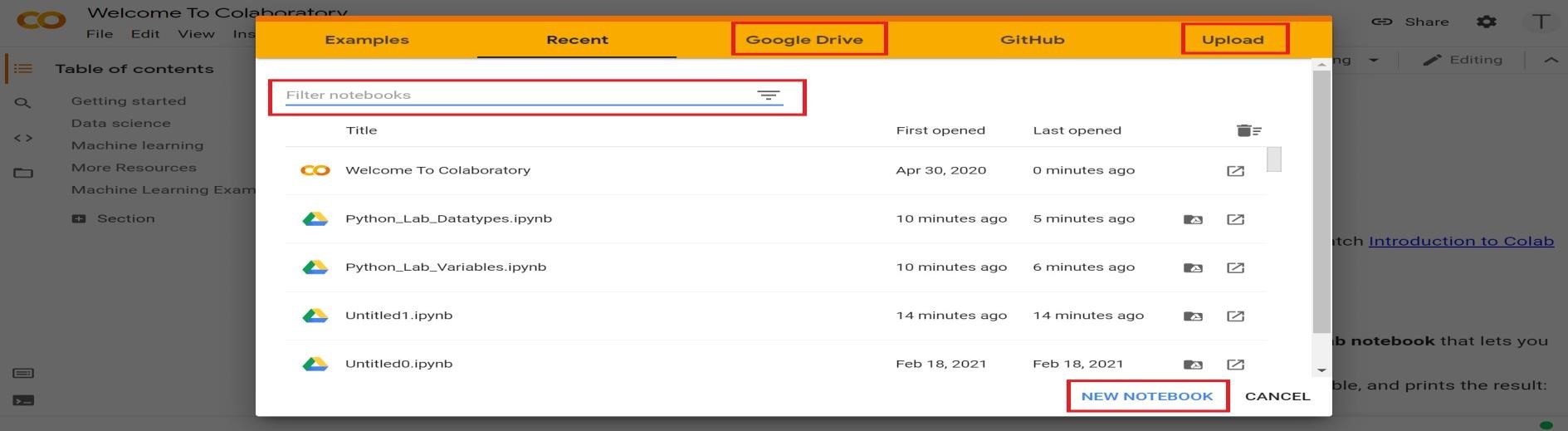


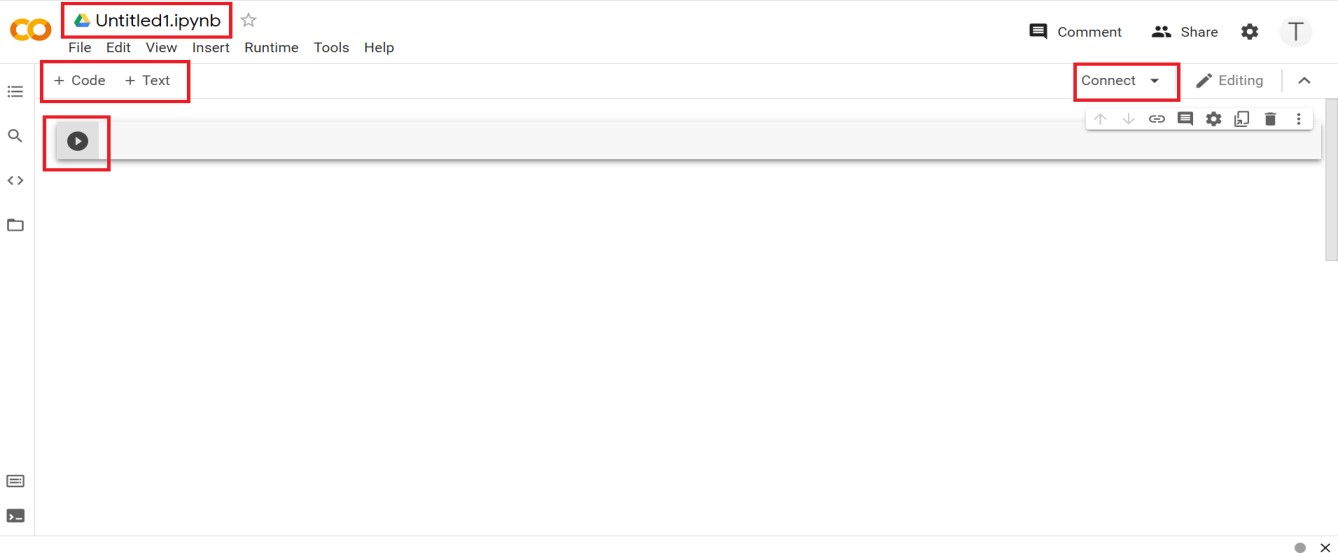
**WORKING WITH GOOGLE COLABORATORY**











**EX.NO: 01 DATE:**

## 8- QUEENS PROBLEM

**AIM :**

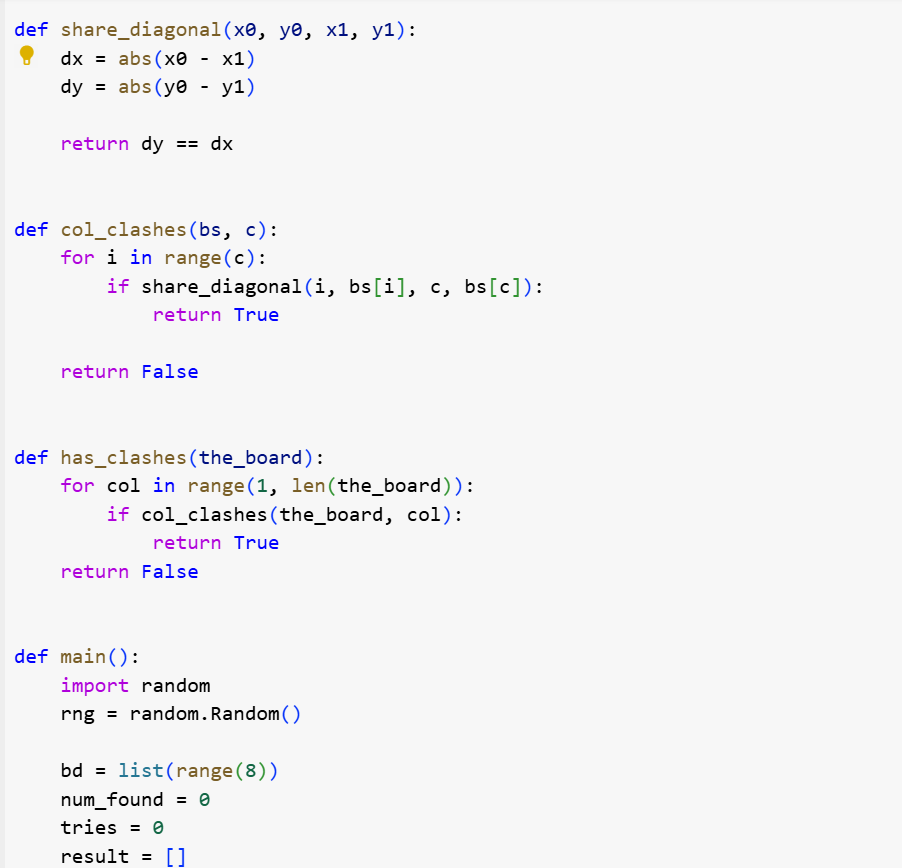
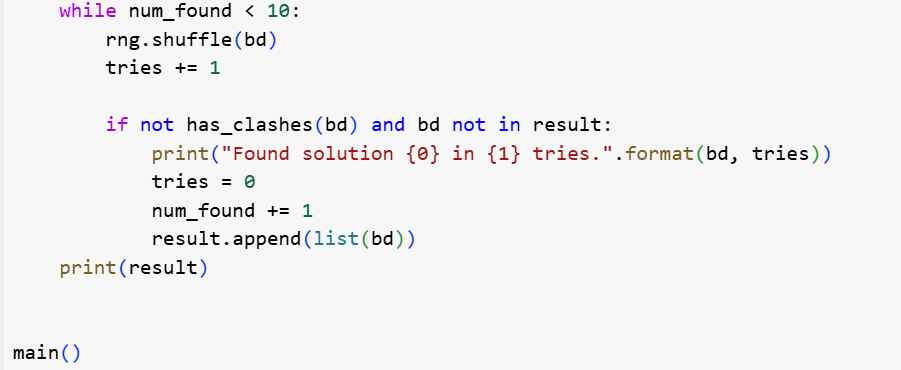
To implement an 8-Queesns problem using Python.

You are given an 8x8 board; find a way to place 8 queens such that no queen can attack any other queen on the chessboard. A queen can only be attacked if it lies on the same row, same column, or the same diagonal as any other queen. Print all the possible configurations.

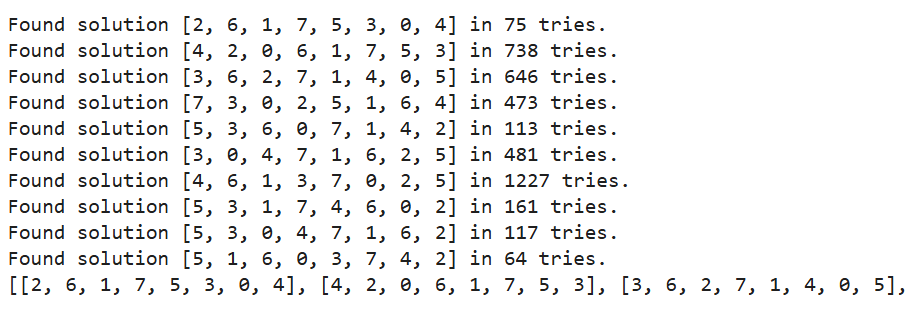
To solve this problem, we will make use of the Backtracking algorithm. The backtracking algorithm, in general checks all possible configurations and test whether the required result is obtained or not. For the given problem, we will explore all possible positions the queens can be relatively placed at. The solution will be correct when the number of placed queens = 8.



**CODE:**



**OUTPUT:**



**RESULT:**

Thus, the 8-Queesns program has been implemented successfully.

**EX.NO:** **02**

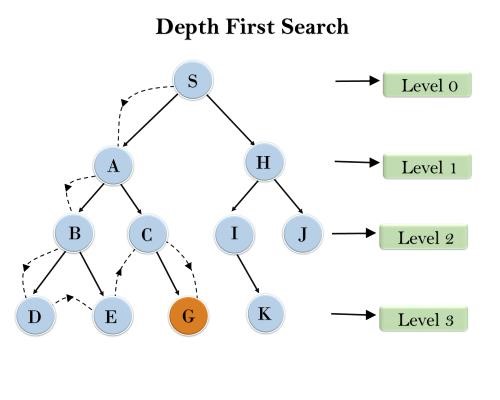
**DATE:**

## DEPTH-FIRST SEARCH

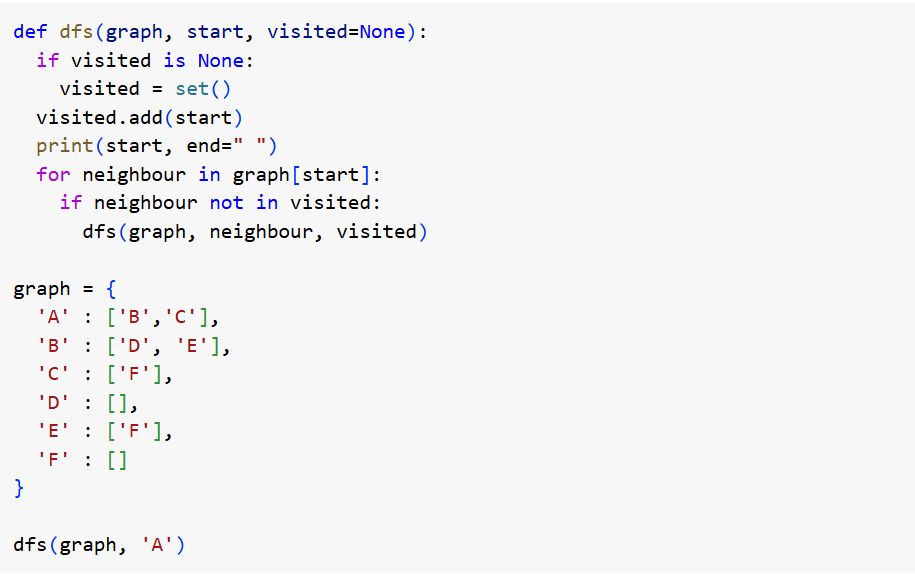
**AIM :**

To implement a depth-first search problem using Python.

* Depth-first search (DFS) algorithm or searching technique starts with the root node of graph G, and then travel deeper and deeper until we find the goal node or the node which has no children by visiting different node of the tree.
* The algorithm, then backtracks or returns back from the dead end or last node towards the most recent node that is yet to be completely unexplored.
* The data structure (DS) which is being used in DFS Depth-first search is stack. The process is quite similar to the BFS algorithm.
* In DFS, the edges that go to an unvisited node are called discovery edges while the edges that go to an already visited node are called block edges.



**CODE:**

****

**OUTPUT:**

****

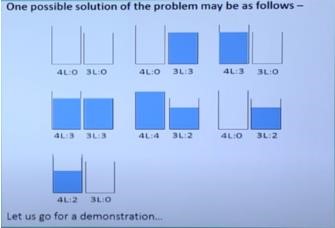
**RESULT:**

Thus, the depth-first search program has been implemented successfully.

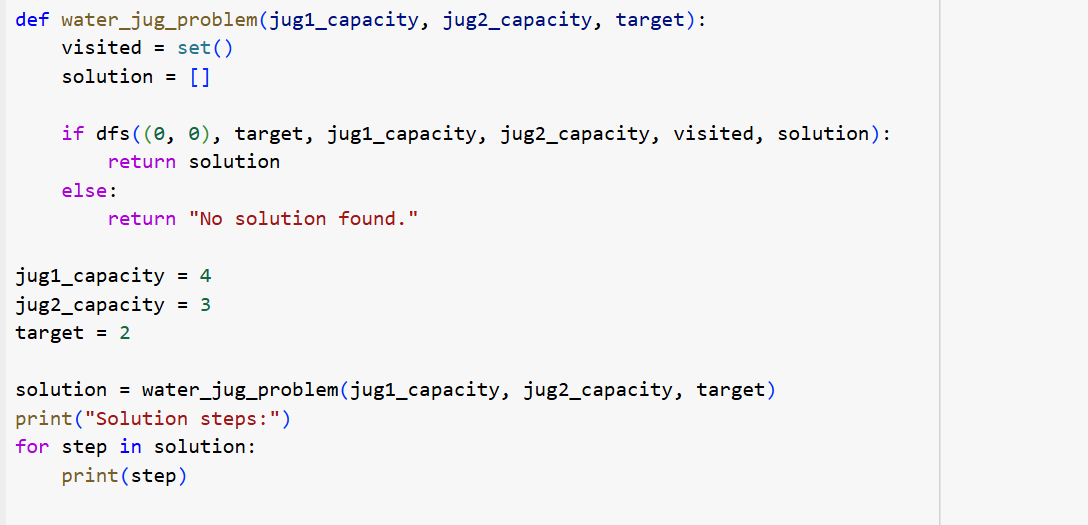
**EX.NO:** **03** **DATE:**

## DEPTH-FIRST SEARCH – WATER JUG PROBLEM

In the **water jug problem in Artificial Intelligence**, we are provided with two jugs: one having the capacity to hold 3 gallons of water and the other has the capacity to hold 4 gallons of water. There is no other measuring equipment available and the jugs also do not have any kind of marking on them. So, the agent’s task here is to fill the 4-gallon jug with 2 gallons of water by using only these two jugs and no other material. Initially, both our jugs are empty.



**CODE:**

****

**OUTPUT:**

****

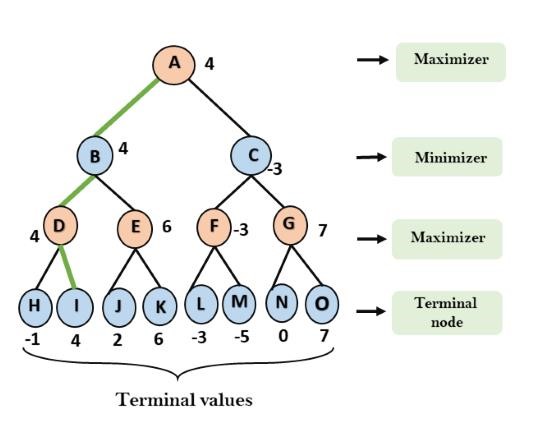
**RESULT:**

Thus, the water jugprogram has been implemented successfully.

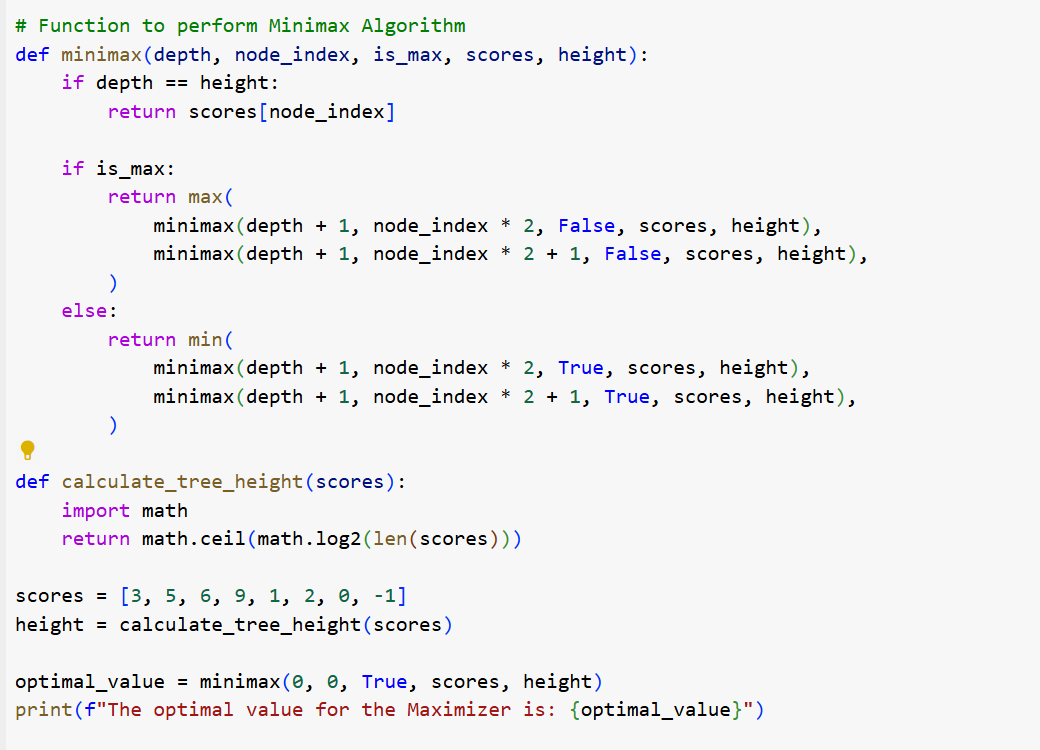
**EX.NO:**  **04**  **DATE:**

## MINIMAX ALGORITHM

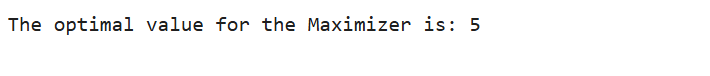
* A simple example can be used to explain how the minimax algorithm works. We've included an example of a game-tree below, which represents a two-player game.
* There are two players in this scenario, one named Maximizer and the other named Minimizer.
* Maximizer will strive for the highest possible score, while Minimizer will strive for the lowest possible score.
* Because this algorithm uses DFS, we must go all the way through the leaves to reach the terminal nodes in this game-tree.
* The terminal values are given at the terminal node, so we'll compare them and retrace the tree till we reach the original state.



**CODE:**



**OUTPUT:**



**RESULT:**

Thus, the minmax algorithm program has been implemented successfully.

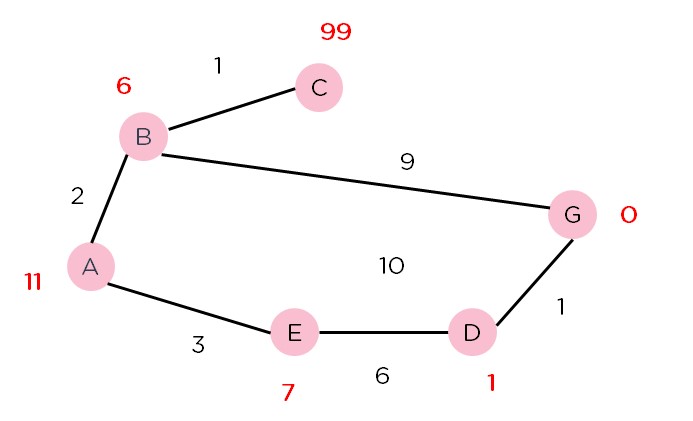
**EX.No :**  **05**  **DATE :**

## A\* SEARCH ALGORITHM

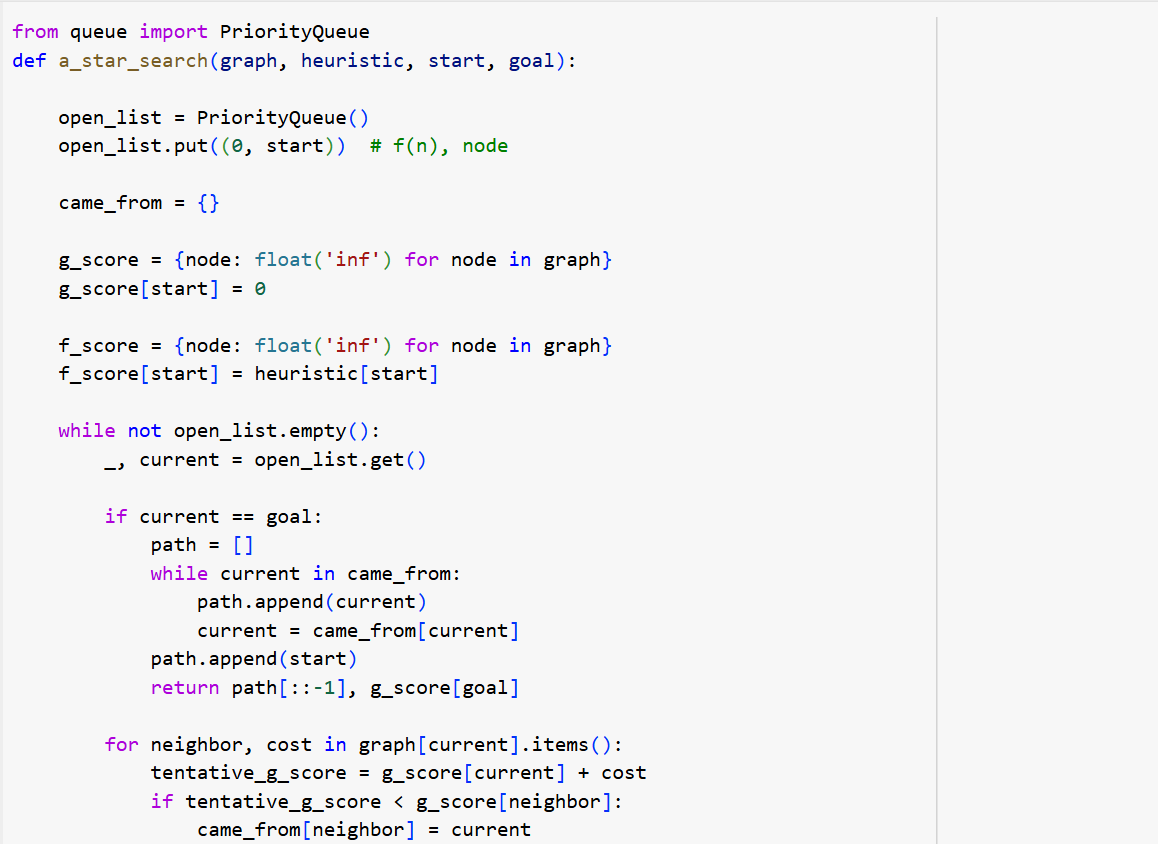
A heuristic algorithm sacrifices optimality, with precision and accuracy for speed, to solve problems faster and more efficiently.

All graphs have different nodes or points which the algorithm has to take, to reach the final node. The paths between these nodes all have a numerical value, which is considered as the weight of the path. The total of all paths transverse gives you the cost of that route.

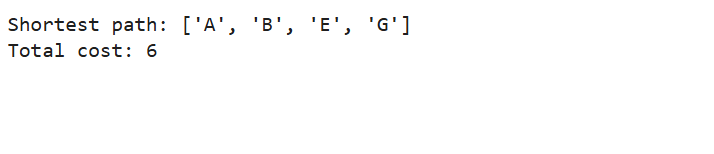
Initially, the Algorithm calculates the cost to all its immediate neighboring nodes,n, and chooses the one incurring the least cost. This process repeats until no new nodes can be chosen and all paths have been traversed. Then, you should consider the best path among them. If f(n) represents the final cost, then it can be denoted as : f(n) = g(n) + h(n), where : g(n) = cost of traversing from one node to another. This will vary from node to node h(n) = heuristic approximation of the node's value. This is not a real value but an approximation cost.



**CODE:**



**OUTPUT:**



**RESULT:**

Thus, the A\* search program has been implemented successfully.

**EX.NO :** **06 DATE:**

**INTRODUCTION TO PROLOG**

**AIM**

To learn PROLOG terminologies and write basic programs.

**TERMINOLOGIES**

1. Atomic Terms: -

Atomic terms are usually strings made up of lower- and uppercase letters, digits, and the underscore, starting with a lowercase letter.

Ex:

dog

ab\_c\_321

1. Variables: -

Variables are strings of letters, digits, and the underscore, starting with a capital letter or an underscore.

Ex:

Dog

Apple\_420

1. Compound Terms: -

Compound terms are made up of a PROLOG atom and a number of arguments (PROLOG terms, i.e., atoms, numbers, variables, or other compound terms) enclosed in parentheses and separated by commas.

Ex:

is\_bigger(elephant,X) f(g(X,\_),7) 4. Facts: -

A fact is a predicate followed by a dot.

Ex: bigger\_animal(whale).

life\_is\_beautiful.

5. Rules: -

A rule consists of a head (a predicate) and a body (a sequence of predicates separated by commas).

Ex:

is\_smaller(X,Y):-is\_bigger(Y,X).

aunt(Aunt,Child):-sister(Aunt,Parent),parent(Parent,Child).

**SOURCE CODE:**

**KB1:**

woman(mia). woman(jody). woman(yolanda). playsAirGuitar(jody).

party.

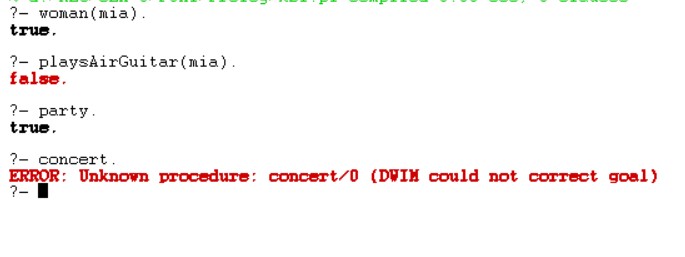
Query 1: ?-woman(mia).

Query 2: ?-playsAirGuitar(mia).

Query 3: ?-party.

Query 4: ?-concert.

**OUTPUT: -**

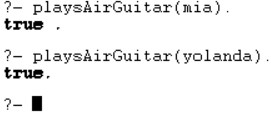


**KB2:**

happy(yolanda). listens2music(mia).

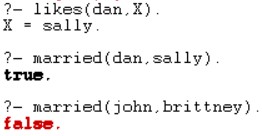
Listens2music(yolanda):-happy(yolanda). playsAirGuitar(mia):-listens2music(mia). playsAirGuitar(Yolanda):-listens2music(yolanda).

**OUTPUT: -**



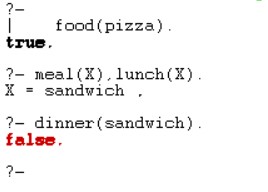
**KB3:** likes(dan,sally). likes(sally,dan). likes(john,brittney). married(X,Y) :- likes(X,Y) , likes(Y,X). friends(X,Y) :- likes(X,Y) ; likes(Y,X).

**OUTPUT: -**



**KB4:** food(burger). food(sandwich). food(pizza). lunch(sandwich). dinner(pizza). meal(X):-food(X).

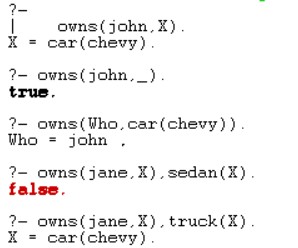
**OUTPUT:**



**KB5:**

owns(jack,car(bmw)). owns(john,car(chevy)). owns(olivia,car(civic)). owns(jane,car(chevy)). sedan(car(bmw)). sedan(car(civic)). truck(car(chevy)).

**OUTPUT:**



**RESULT:**

Thus, the Prolog terminologies and basic program has been implemented successfully.

**EX.NO: 07 DATE:**

**PROLOG- FAMILY TREE**

**AIM**

To develop a family tree program using PROLOG with all possible facts, rules, and queries.

**SOURCE CODE:**

**KNOWLEDGE BASE:**

/\*FACTS :: \*/ male(peter). male(john). male(chris). male(kevin).

female(betty). female(jeny). female(lisa). female(helen).

parentOf(chris,peter). parentOf(chris,betty). parentOf(helen,peter). parentOf(helen,betty). parentOf(kevin,chris). parentOf(kevin,lisa). parentOf(jeny,john). parentOf(jeny,helen).

/\*RULES :: \*/

/\* son,parent

\* son,grandparent\*/

father(X,Y):- male(Y), parentOf(X,Y).

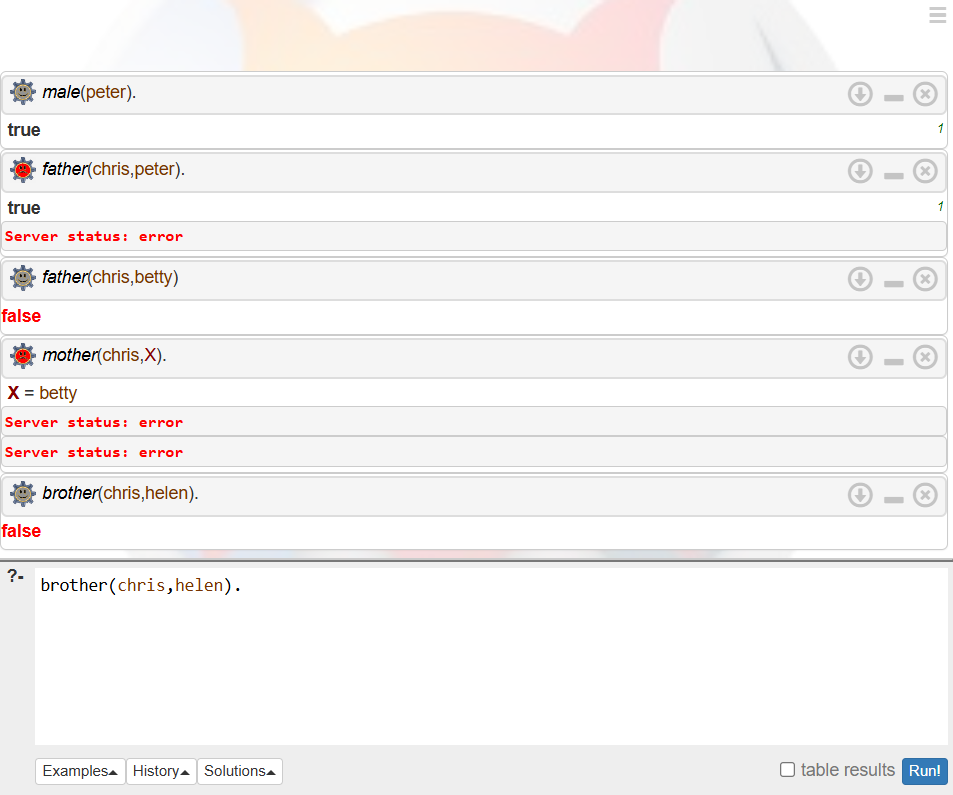
mother(X,Y):- female(Y), parentOf(X,Y).

grandfather(X,Y):- male(Y),parentOf(X,Z),parentOf(Z,Y).

grandmother(X,Y):- female(Y),parentOf(X,Z),parentOf(Z,Y).

brother(X,Y):- male(Y), father(X,Z), father(Y,W),Z==W.

sister(X,Y):- female(Y), father(X,Z),father(Y,W),Z==W.



**RESULT:**

Thus the family tree program using PROLOG with all possible facts, rules, and queries has been implemented successfully.

**EX.NO :** **08** **DATE :**

**IMPLEMENTING ARTIFICIAL NEURAL NETWORKS FOR AN**

**APPLICATION USING PYTHON - REGRESSION**

**AIM :**

To implementing artificial neural networks for an application in Regression using python.

**Regression using Artificial Neural Networks**

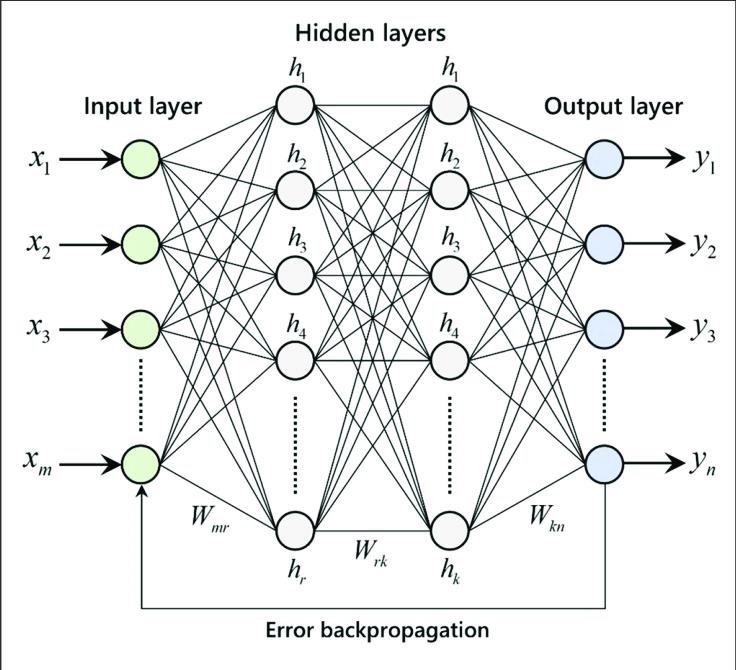
Why do we need to use Artificial Neural Networks for Regression instead of simply using Linear Regression?

The purpose of using Artificial Neural Networks for Regression over Linear Regression is that the linear regression can only learn the linear relationship between the features and target and therefore cannot learn the complex non-linear relationship. In order to learn the complex non-linear relationship between the features and target, we are in need of other techniques. One of those techniques is to use Artificial Neural Networks. Artificial Neural Networks have the ability to learn the complex relationship between the features and target due to the presence of activation function in each layer. Let’s look at what are Artificial Neural Networks and how do they work.

**Artificial Neural Networks**

Artificial Neural Networks are one of the deep learning algorithms that simulate the workings of neurons in the human brain. There are many types of Artificial Neural Networks, Vanilla Neural Networks, Recurrent Neural Networks, and Convolutional Neural Networks. The Vanilla Neural Networks have the ability to handle structured data only, whereas the Recurrent Neural Networks and Convolutional Neural Networks have the ability to handle unstructured data very well. In this post, we are going to use Vanilla Neural Networks to perform the Regression Analysis.

**Structure of Artificial Neural Networks**



The Artificial Neural Networks consists of the Input layer, Hidden layers, Output layer. The hidden layer can be more than one in number. Each layer consists of n number of neurons. Each layer will be having an Activation Function associated with each of the neurons. The activation function is the function that is responsible for introducing non-linearity in the relationship. In our case, the output layer must contain a linear activation function. Each layer can also have regularizers associated with it. Regularizers are responsible for preventing overfitting.

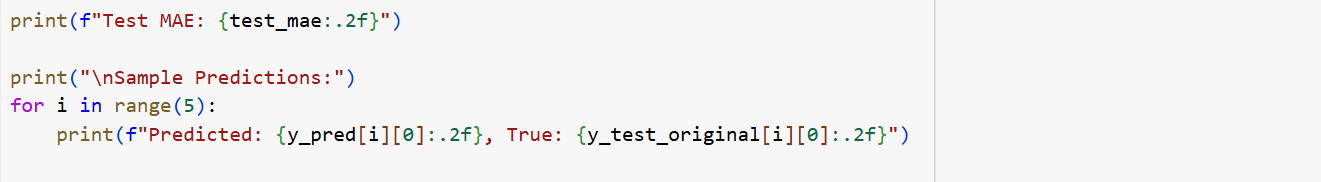
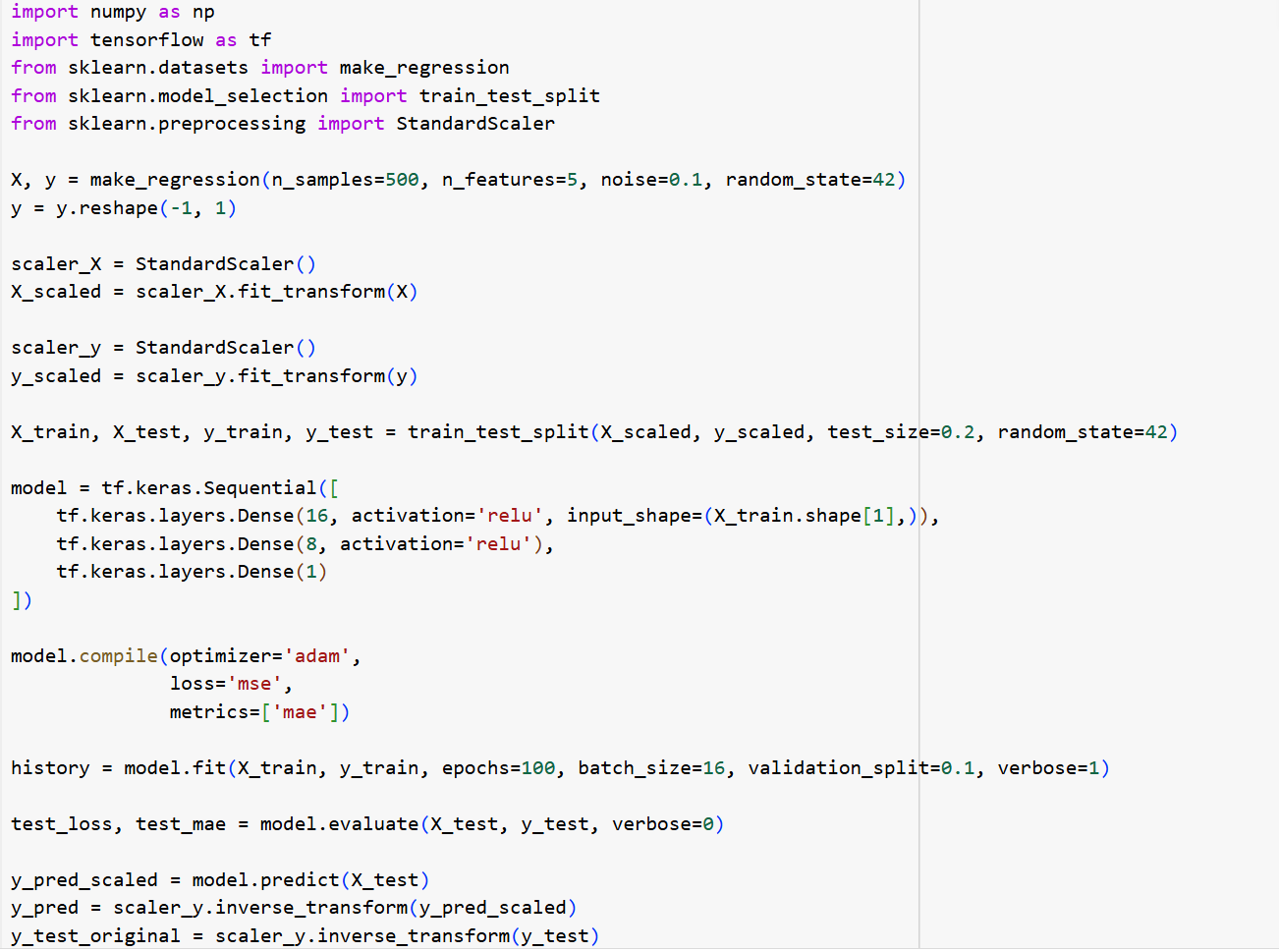
Artificial Neural Networks consists of two phases,

* Forward Propagation
* Backward Propagation

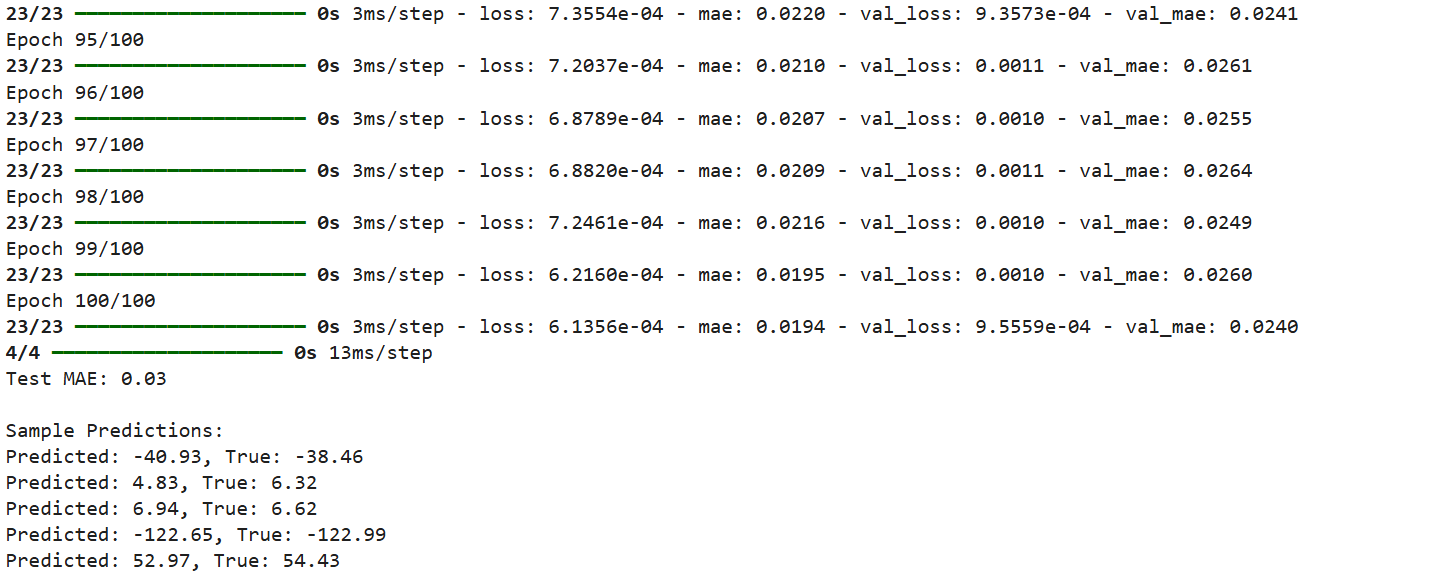
Forward propagation is the process of multiplying weights with each feature and adding them. The bias is also added to the result. Backward propagation is the process of updating the weights in the model.

Backward propagation requires an optimization function and a loss function.

**CODE:**

****

**OUTPUT:**



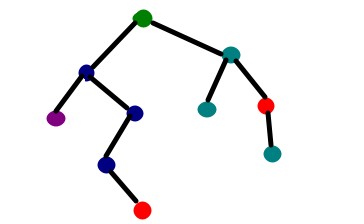
**RESULT:**

Thus, the artificial neural network application in regression program has been implemented successfully.

**EX.NO :**  **09**  **DATE :**

**IMPLEMENTATION OF DECISION TREE CLASSIFICATION TECHNIQUES**

[Decision Tree](https://www.geeksforgeeks.org/decision-tree/) is one of the most powerful and popular algorithm. Decision-tree algorithm falls under the category of supervised learning algorithms. It works for both continuous as well as categorical output variables.



**AIM:**

To implement a decision tree classification technique for gender classification using python.

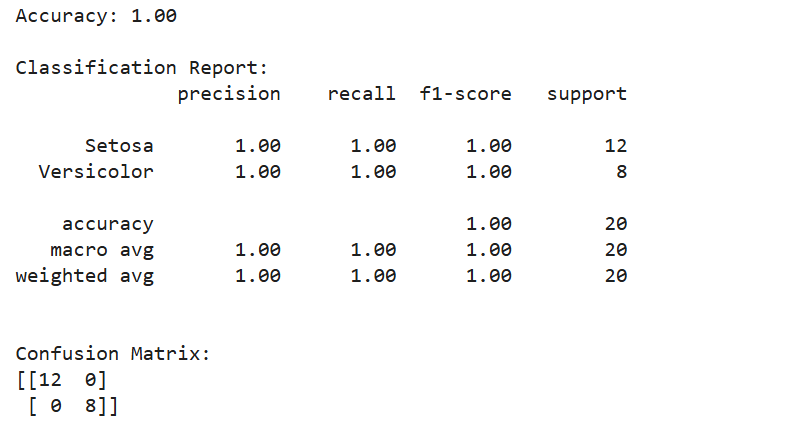
**EXPLANATION:**

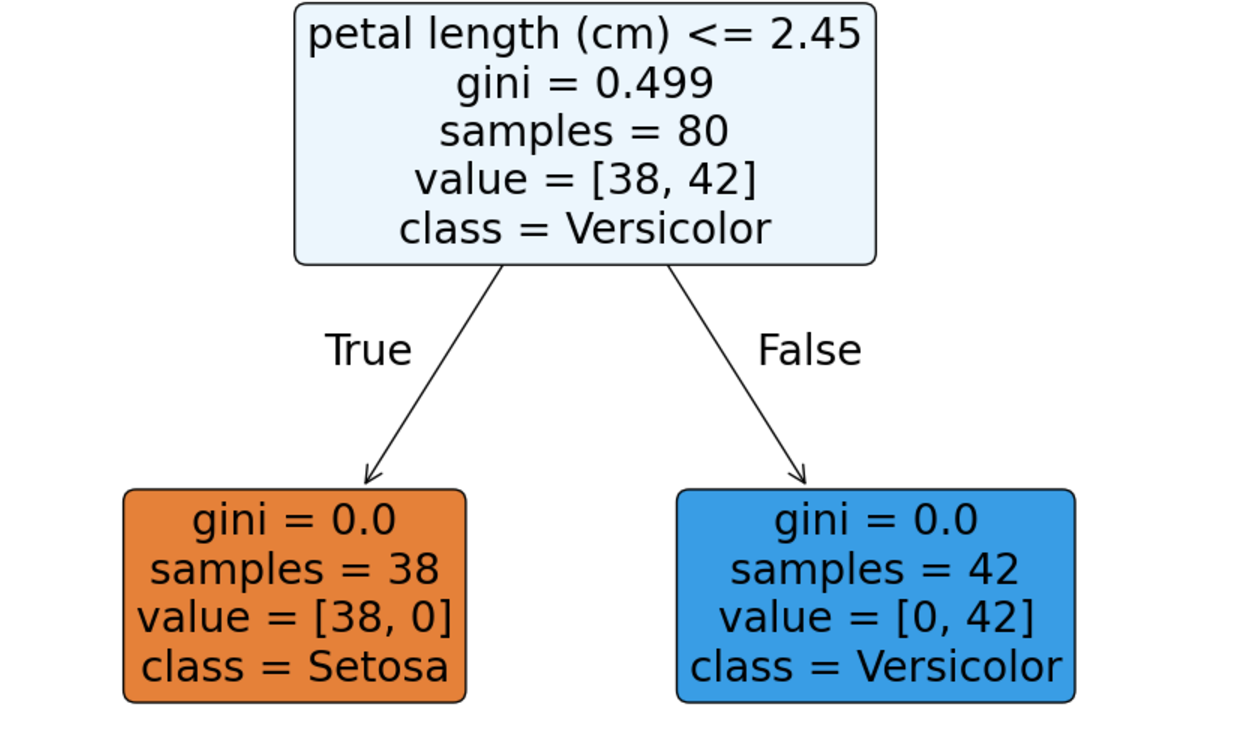
* + Import tree from sklearn.
  + Call the function DecisionTreeClassifier() from tree
  + Assign values for X and Y.
  + Call the function predict for Predicting on the basis of given random values for each given feature.
  + Display the output.

**CODE:**



**OUTPUT:**

****



**RESULT:**

Thus, the Decision Tree Classification program has been implemented successfully.

**EX NO :** **10** **DATE :**

## IMPLEMENTATION OF CLUSTERING TECHNIQUES K - MEANS

The ***k*-means clustering** method is an [unsupervised machine learning](https://en.wikipedia.org/wiki/Unsupervised_learning) technique used to identify clusters of data objects in a dataset. There are many different types of clustering methods, but *k*means is one of the oldest and most approachable. These traits make implementing *k*-means clustering in Python reasonably straightforward, even for novice programmers and data scientists.

If you’re interested in learning how and when to implement *k*-means clustering in Python, then this is the right place. You’ll walk through an end-to-end example of *k*-means clustering using Python, from preprocessing the data to evaluating results.

How does it work?

First, each data point is randomly assigned to one of the K clusters. Then, we compute the centroid (functionally the center) of each cluster, and reassign each data point to the cluster with the closest centroid. We repeat this process until the cluster assignments for each data point are no longer changing.

K-means clustering requires us to select K, the number of clusters we want to group the data into. The elbow method lets us graph the inertia (a distance-based metric) and visualize the point at which it starts decreasing linearly. This point is referred to as the "eblow" and is a good estimate for the best value for K based on our data.

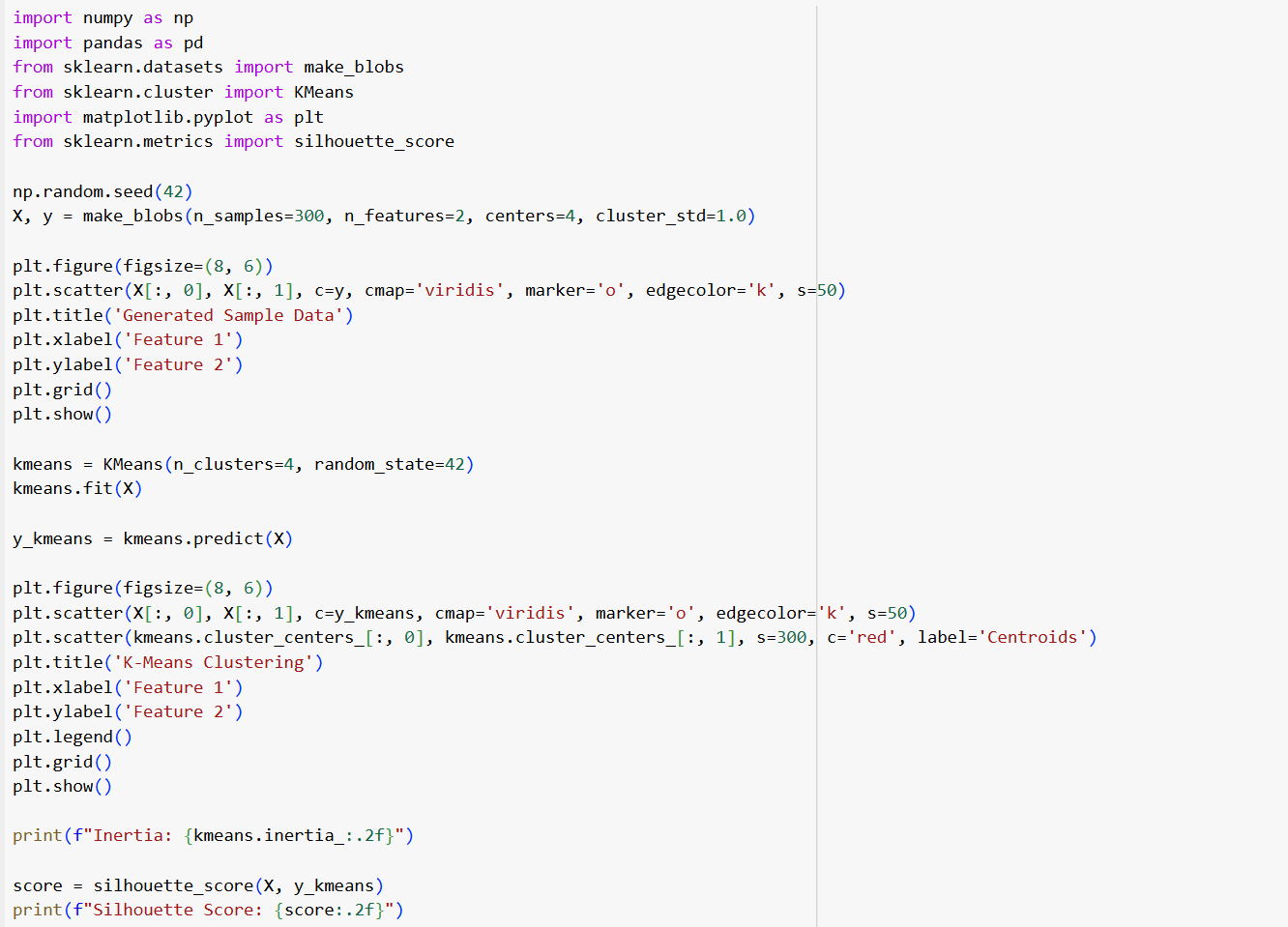
**AIM:**

To implement a K - Means clustering technique using python language.

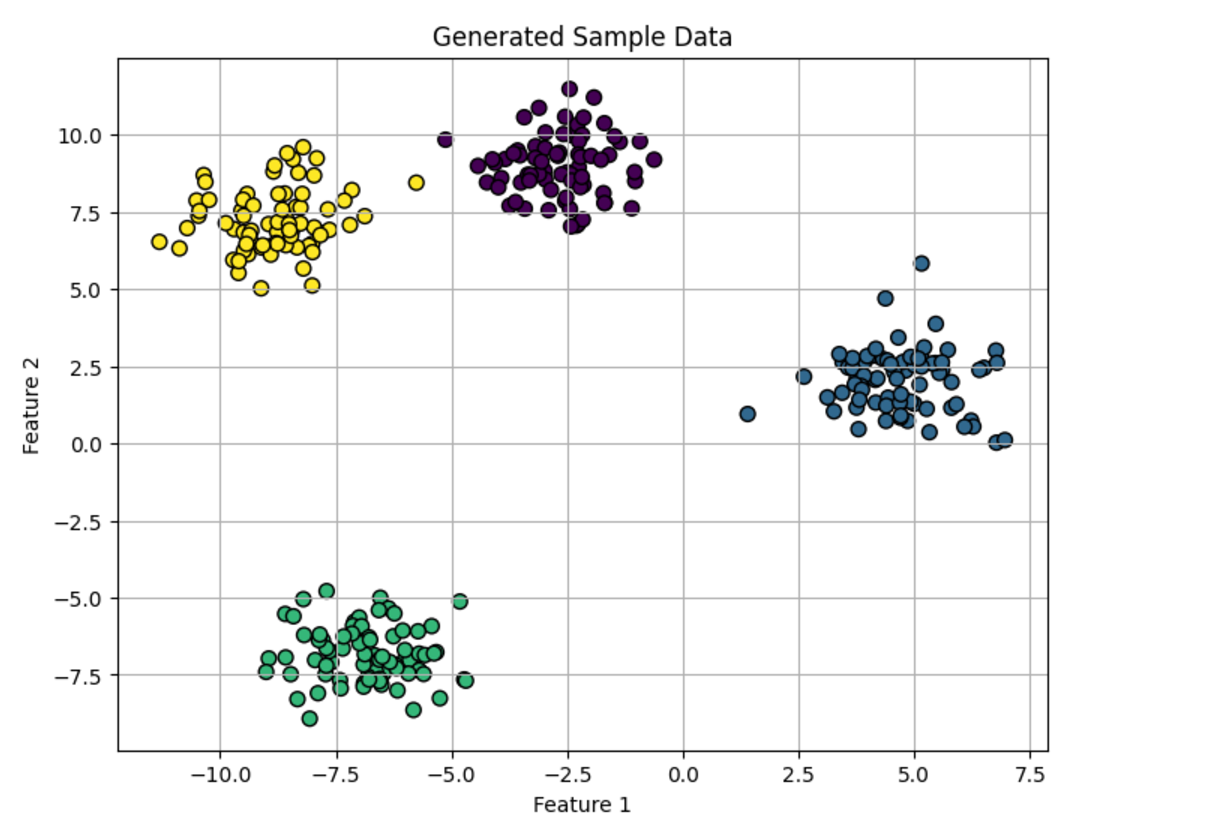
**EXPLANATION:**

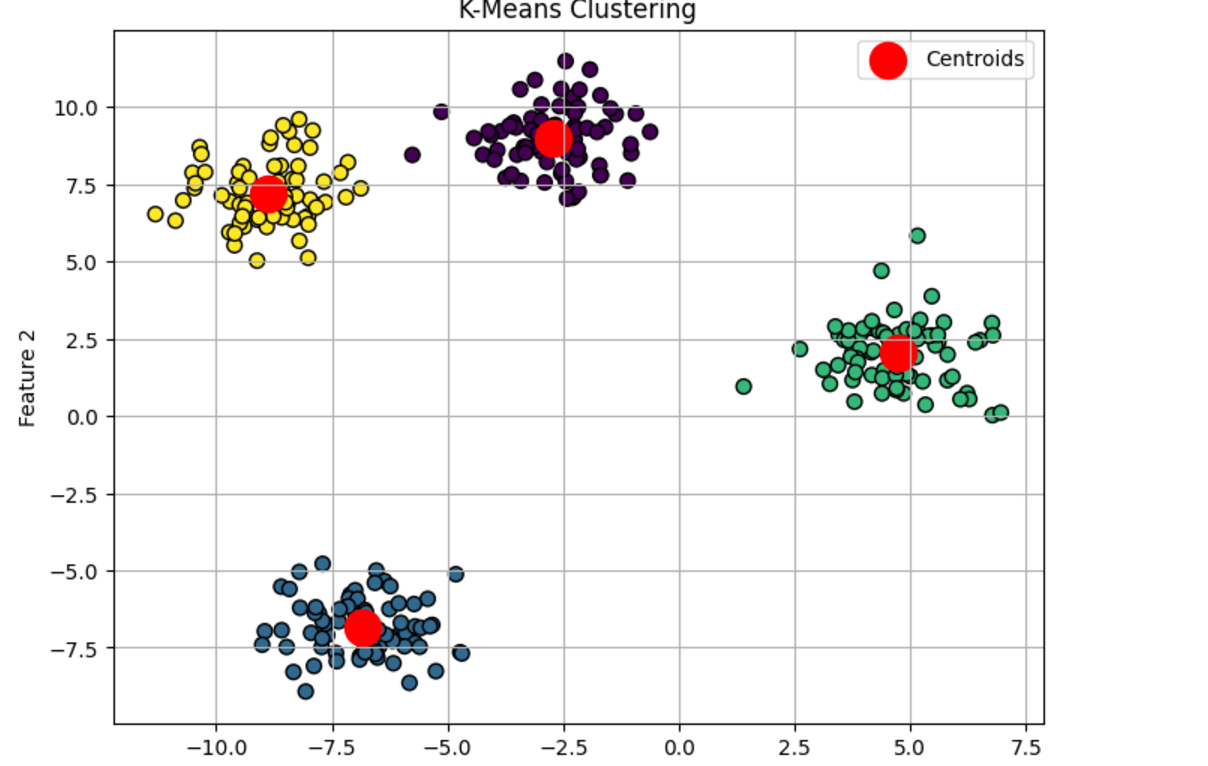
* Import KMeans from sklearn.cluster
* Assign X and Y.
* Call the function KMeans().
* Perform scatter operation and display the output.

**CODE:**

****

**OUTPUT:**

****

****

**RESULT:**

Thus, the K-Means Clustering technique program has been implemented successfully.