Artificial Intelligence (KCA-351)
Lab Manual

1. Program must be written in Python
2. Algorithm and program both are compulsory.
3. One pilot project is mandatory for individual student example Mini Alexa/Gamming project

S.No.	Program List	Due Date	Remarks
Ex-1	Installation of Python/Anaconda and introduction of Google Colab	First Week	
Ex-2	Write a program to implement water jug problem'	First Week	
Ex-3	Write a program to implement missionary and cannibal algorithm	Second Week	
Ex-4	Write a program to implement breadth first search	Second Week	
Ex-5	Write a program to implement depth first search	Third Week	
Ex-6	Write a program to implement 8 Queen Problems	Third Week	
Ex-7	Write a program to implement first order logic	Fourth Week	
Ex-8	Write a program to demonstrate the working of Bayesian network.	Fourth Week	
Ex-9	Write a program to Implement pattern recognition problems of handwritten character/ digit recognition	Fifth Week	
Ex-10	Write a program to Implement pattern recognition problems of speech recognition	Sixth Week	
Ex-11	Write a program to implement naive bayes classification problem	Seventh Week	
Ex-12	Write a program to implement k-mean clustering problem	Eight Week	
Ex-13	Write a program to convert text to speech using NLP tool	Ninth Week	
Ex-14	Pilot Project in AI using Python **(Contents Beyond Syllabus)**	Tenth Week	

Ex-1	Installation of Python/Anaconda and introduction of Google Colab
Expected Date	First Week
CO	CO-1

Python is a widely used high-level programming language. To write and execute code in python, we first need to install Python on our system.

Installing Python on Windows takes a series of few easy steps.

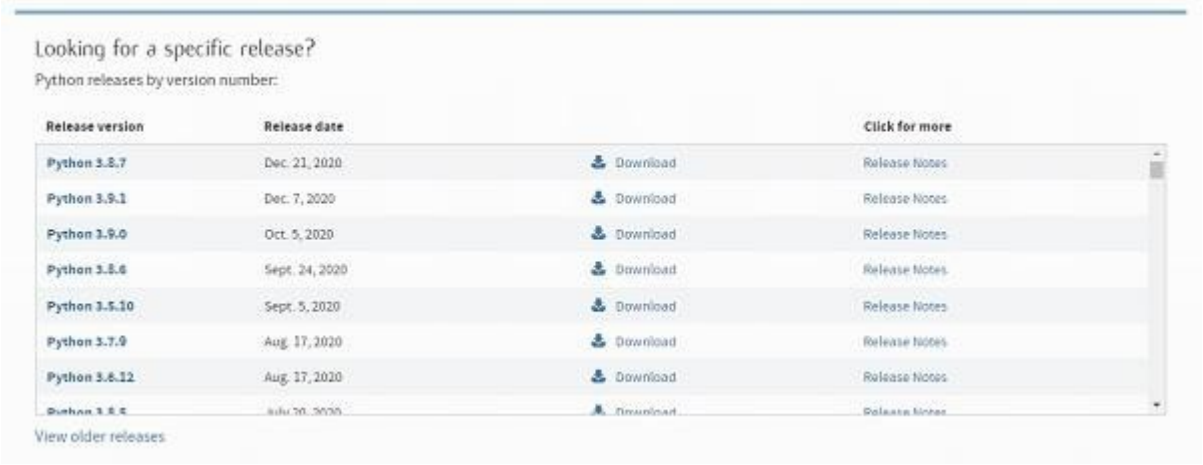
Step 1 − Select Version of Python to Install

Python has various versions available with differences between the syntax and working of different versions of the language. We need to choose the version which we want to use or need. There are different versions of Python 2 and Python 3 available.

Step 2 − Download Python Executable Installer

On the web browser, in the official site of python (www.python.org), move to the Download for Windows section.

All the available versions of Python will be listed. Select the version required by you and click on Download. Let suppose, we chose the Python 3.9.1 version.



On clicking download, various available executable installers shall be visible with different operating system specifications. Choose the installer which suits your system operating system and download the instlaller. Let suppose, we select the Windows installer(64 bits).

The download size is less than 30MB.

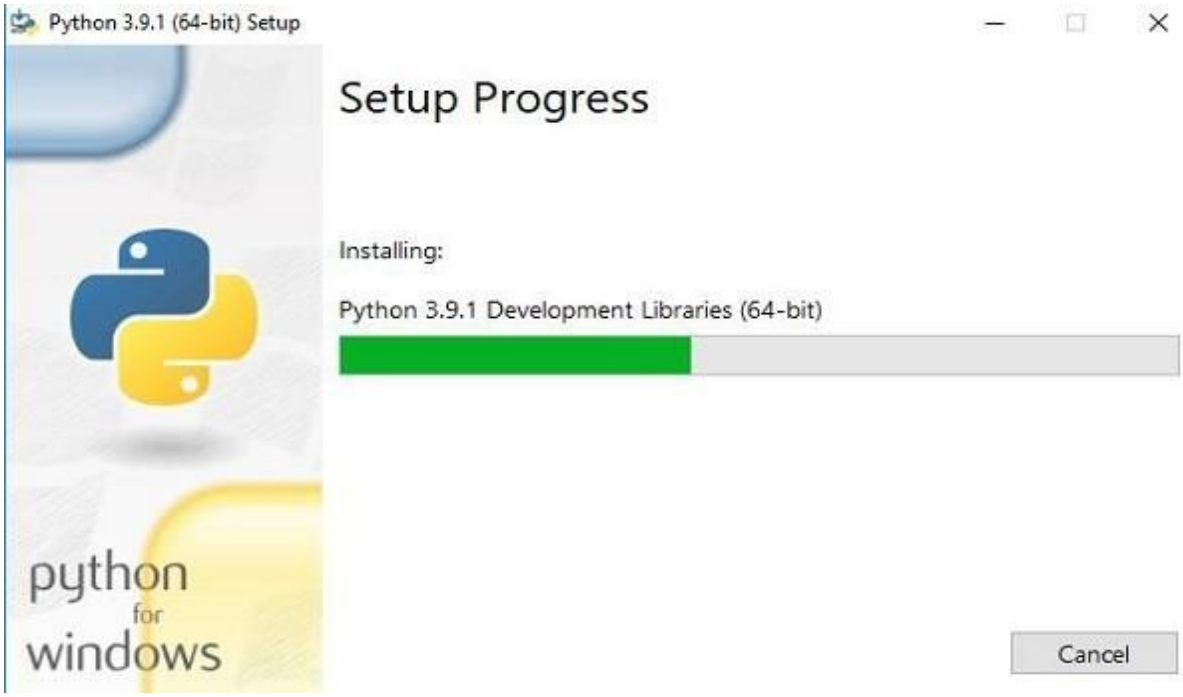
Version	Operating System	Description	MD5 Sum	File Size	GPG
Gzipped source tarball	Source release		429ae95d24227f8fa1560684fad6fca7	25372998	SIG
XZ compressed source tarball	Source release		61981498e75ac8f00adcb908281fadb6	18897104	SIG
macOS 64-bit Intel installer	Mac OS X	for macOS 10.9 and later	74f5cc5b5783ce8fb2ca55f11f3f0699	29795899	SIG
macOS 64-bit universal2 installer	Mac OS X	for macOS 10.9 and later, including macOS 11 Big Sur on Apple Silicon (experimental)	8b19748473609241e60aa3618bbaf3ed	37451735	SIG
Windows embeddable package (32-bit)	Windows		96c6fa81fe8b650e68c3dd41258ae317	7571141	SIG
Windows embeddable package (64-bit)	Windows		e70e5c22432d8f57a497cde5ec2e5ce2	8402333	SIG
Windows help file	Windows		c49d0b6ef88c0831ed0e2d39bc42b316	8787443	SIG
Windows installer (32-bit)	Windows		dde210ea04a31c27488605a9e7cd297a	27126136	SIG
Windows installer (64-bit)	Windows	Recommended	b3fce2ed8bc315ad2bc49eae48a94487	28204528	SIG

Step 3 − Run Executable Installer

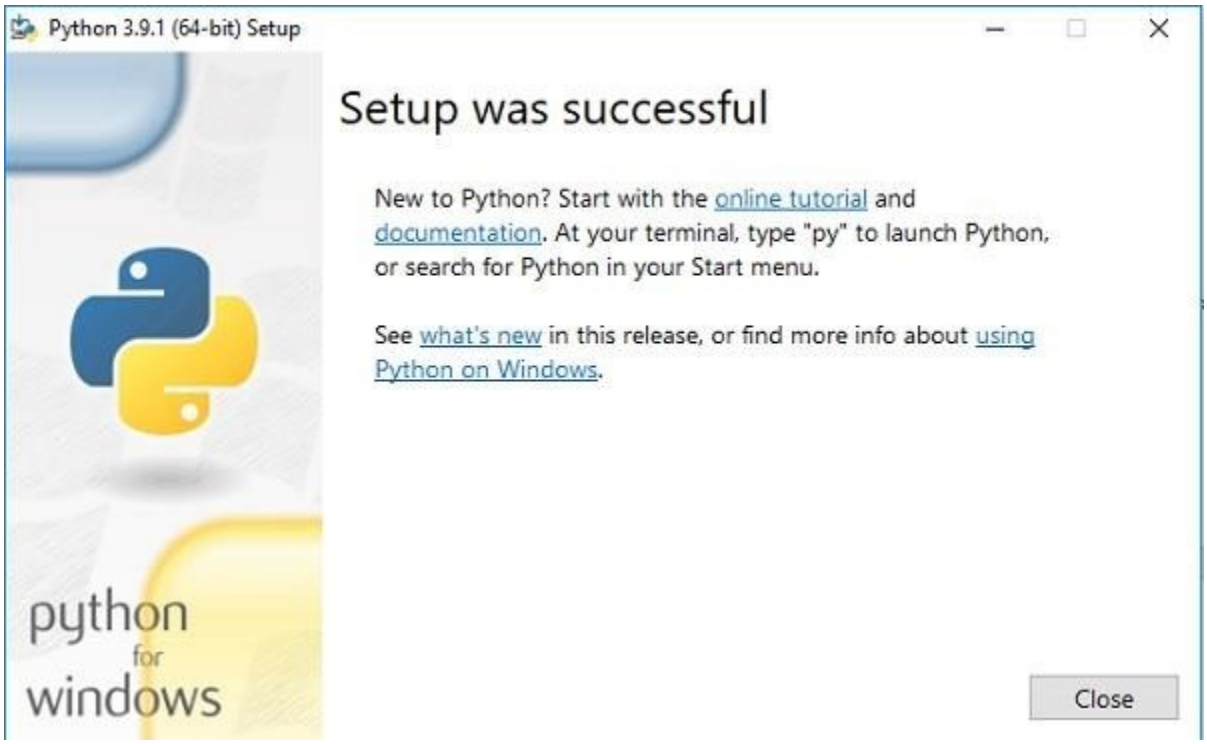
We downloaded the Python 3.9.1 Windows 64 bit installer.

Run the installer. Make sure to select both the checkboxes at the bottom and then click Install New.

On clicking the Install Now, The installation process starts.



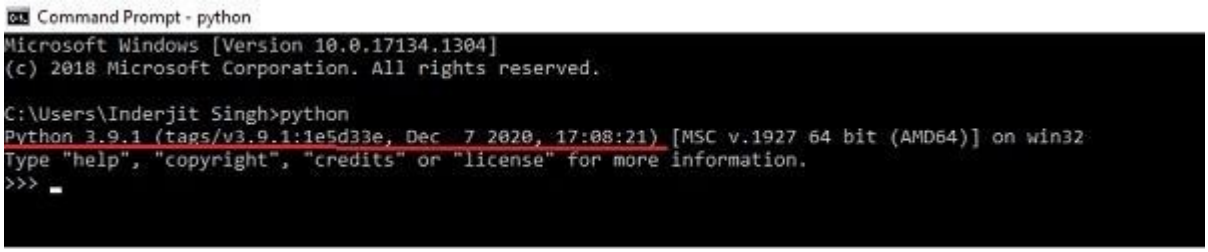
The installation process will take few minutes to complete and once the installation is successful, the following screen is displayed.



Step 4 − Verify Python is installed on Windows

To ensure if Python is succesfully installed on your system. Follow the given steps −

- Open the command prompt.
- Type 'python' and press enter.
- The version of the python which you have installed will be displayed if the python is successfully installed on your windows.

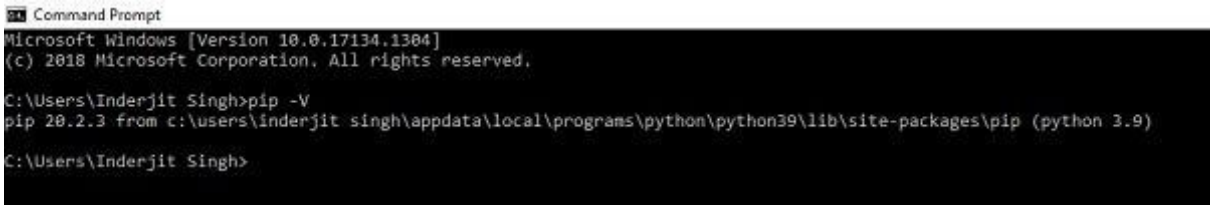


Step 5 − Verify Pip was installed

Pip is a powerful package management system for Python software packages. Thus, make sure that you have it installed.

To verify if pip was installed, follow the given steps −

- Open the command prompt.
- Enter pip –V to check if pip was installed.
- The following output appears if pip is installed successfully.



We have successfully installed python and pip on our Windows system.

Ex-2	Write a program to implement water jug problem
Expected Date	First Week
CO	CO-1

A **Water Jug Problem**: You are given two jugs, a 4-gallon one and a 3-gallon one, a pump which has unlimited water which you can use to fill the jug, and the ground on which water may be poured. Neither jug has any measuring markings on it. How can you get exactly 2 gallons of water in the 4-gallon jug?

Here the initial state is (0, 0). The goal state is (2, n) for any value of n.

State Space Representation: we will represent a state of the problem as a tuple (x, y) where x represents the amount of water in the 4-gallon jug and y represents the amount of water in the 3-gallon jug. Note that $0 \leq x \leq 4$, and $0 \leq y \leq 3$.

To solve this we have to make some assumptions not mentioned in the problem. They are:

- We can fill a jug from the pump.
- We can pour water out of a jug to the ground.
- We can pour water from one jug to another.
- There is no measuring device available.

Operators — we must define a set of operators that will take us from one state to another.

Sr.	Current State	Next State	Descriptions
1	(x, y) if x < 4	(4, y)	Fill the 4 gallon jug
2	(x, y) if y < 3	(x, 3)	Fill the 3 gallon jug
3	(x, y) if x > 0	(x - d, y)	Pour some water out of the 4 gallon jug
4	(x, y) if y > 0	(x, y - d)	Pour some water out of the 3 gallon jug
5	(x, y) if y > 0	(0, y)	Empty the 4 gallon jug
6	(x, y) if y > 0	(x, 0)	Empty the 3 gallon jug on the ground
7	(x, y) if x + y >= 4 and y > 0	(4, y - (4 - x))	Pour water from the 3 gallon jug into the 4 gallon jug until the 4 gallon jug is full
8	(x, y) if x + y >= 3 and x > 0	(x - (3 - y), 3)	Pour water from the 4 gallon jug into the 3-gallon jug until the 3 gallon jug is full
9	(x, y) if x + y <= 4 and y > 0	(x + y, 0)	Pour all the water from the 3 gallon jug into the 4 gallon jug
10	(x, y) if x + y <=3 and x > 0	(0, x + y)	Pour all the water from the 4 gallon jug into the 3 gallon jug
11	(0, 2)	(2, 0)	Pour the 2 gallons from 3 gallon jug into the 4 gallon jug
12	(2, y)	(0, y)	Empty the 2 gallons in the 4 gallon jug on the ground

There are several sequences of operations that will solve the problem. One of the possible solutions is given as:

Gallons in the 4-gallon jug	Gallons in the 3-gallon jug	Rule applied
0	0	2
0	3	9
3	0	2
3	3	7
4	2	5 or 12
0	2	9 or 11
2	0	--

Python Program

```
x = 0
y = 0
m = 4
n = 3
print("Initial state = (0,0)")
print("Capacities = (4,3)")
print("Goal state = (2,y)")
while x != 2:
  r = int(input("Enter rule"))
  if(r == 1):
    x = m
  elif(r == 2):
  y = n
  elif(r == 3):
    x = 0
  elif(r == 4):
    y = 0
  elif(r == 5):
    t = n - y
    y = n
    x -= t
  elif(r == 6):
    t = m - x
    x = m
    y -= t
  elif(r == 7):
    y += x
    x = 0
  elif(r == 8):
    x += y
    y = 0
  print (x, y)
```

Output:

```
Initial state = (0,0)
Capacities = (4,3)
Goal state = (2,y)
Enter rule1
4 0
Enter rule5
1 3
Enter rule4
1 0
Enter rule7
0 1
Enter rule1
4 1
Enter rule5
2 3
```

Ex-3	Write a program to implement missionary and cannibal algorithm
Expected Date	Second Week
CO	CO-1

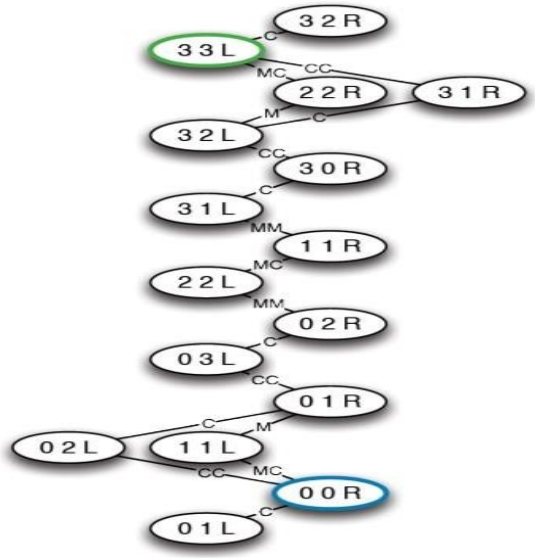
Missionaries and Cannibals: Missionaries and Cannibals is a problem in which 3 missionaries and 3 cannibals want to cross from the left bank of a river to the right bank of the river. There is a boat on the left bank, but it only carries at most two people at a time (and can never cross with zero people). If cannibals ever outnumber missionaries on either bank, the cannibals will eat the missionaries.

A state can be represented by a triple, (m c b), where m is the number of missionaries on the left, c is the number of cannibals on the left, and b indicates whether the boat is on the left bank or right bank. For example, the initial state is (3 3 L) and the goal state is (3 3 R).

Operators are:

- MM: 2 missionaries cross the river
- CC: 2 cannibals cross the river
- MC: 1 missionary and 1 cannibal cross the river
- M: 1 missionary crosses the river
- C: 1 cannibal crosses the river

Draw a diagram showing all the legal states and transitions from states corresponding to all legal operations.



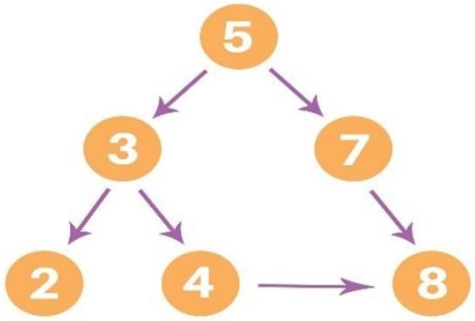
Python program

```
M1=3
C1=3
M2=0
C2=0

print("Initial state = (3M,3C,Bank-1)")

print("Goal state = (3M,3C,Bank-2)")
while ((M2 != 3) or (C2!=3)):
  r = int(input("Enter rule"))
  if(r == 1):  #(1M,0C) from left to right
    M1-=1
    M2+=1

  elif(r == 2): #(2M,0C) from left to right
    M1-=2
    M2+=2
  elif(r == 3):#(1M,1C) from left to right
    M1-=1
    C1-=1
    M2+=1
    C2+=1
  elif(r == 4):#(0M,1C) from left to right
    C1-=1
    C2+=1
  elif(r == 5):#(0M,2C) from left to right
    C1-=2
    C2+=2
  elif(r == 6):#(1M,0C) from right to left
    M1+=1
    M2-=1
  elif(r == 7):#(2M,0C) from right to left
    M1+=2
    M2-=2
  elif(r == 8):#(1M,1C) from right to left
    M1+=1
    C1+=1
    M2-=1
    C2-=1
  elif(r==9):#(0M,1C) from right to left
    C1+=1
    C2-=1
  elif(r==10):#(0M,2C) from right to left
    C1+=2
    C2-=2
  print (M1, C1)
  print (M2, C2)
  if((M1>0 and M1<C1) or (M2>0 and
M2<C2)):
    print("dead")
    break
```

Ex-4	Write a program to implement breadth first search
Expected Date	Second Week
CO	CO-1

Breadth First Search: Breadth-first search (BFS) is an algorithm for traversing or searching tree or graph data structures. It starts at the tree root (or some arbitrary node of a graph, sometimes referred to as a 'search key'), and explores all of the neighbor nodes at the present depth prior to moving on to the nodes at the next depth level.

Question. Write a program to implement breadth first search for the following state space graph:



Time complexity: Equivalent to the number of nodes traversed in BFS until the shallowest solution.
Space complexity: Equivalent to how large can the fringe get.
Completeness: BFS is complete, meaning for a given search tree, BFS will come up with a solution if it exists.

Python Program

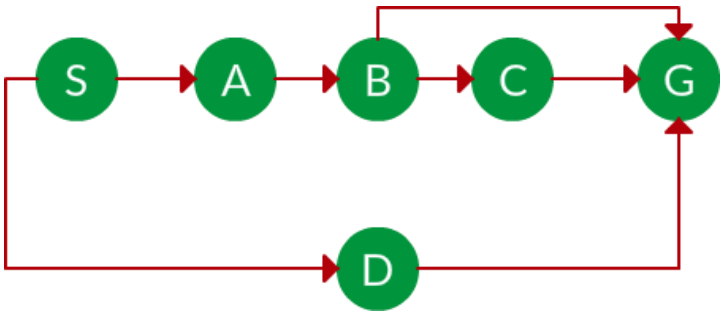
```
graph = {
  'A' : ['B','C'],
  'B' : ['D', 'E'],
  'C' : ['F'],
  'D' : [],
  'E' : ['F'],
  'F' : []
}
```

```
visited = [] # List to keep track of visited nodes.
queue = []     #Initialize a queue
def bfs(visited, graph, node):
  visited.append(node)
  queue.append(node)
  while queue:
    s = queue.pop(0)
    print (s, end = " ")
    for neighbour in graph[s]:
      if neighbour not in visited:
        visited.append(neighbour)
        queue.append(neighbour)
# Driver Code
bfs(visited, graph, 'A')
```

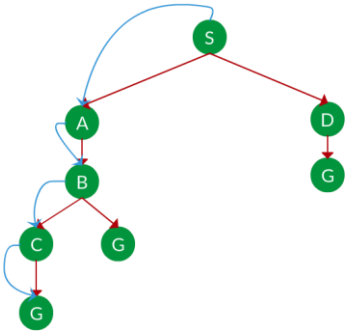
Ex-5	Write a program to implement depth first search
Expected Date	Third Week
CO	CO-1

Depth First Search: Depth-first search (DFS) is an algorithm for traversing or searching tree or graph data structures. The algorithm starts at the root node (selecting some arbitrary node as the root node in the case of a graph) and explores as far as possible along each branch before backtracking.

Question: Write a program to implement depth first search for the following state space graph:



Solution. The equivalent search tree for the above graph is as follows. As DFS traverses the tree "deepest node first", it would always pick the deeper branch until it reaches the solution (or it runs out of nodes, and goes to the next branch). The traversal is shown in blue arrows.



Path: S -> A -> B -> C -> G = the depth of the search tree = the number of levels of the search tree.

Time complexity: Equivalent to the number of nodes traversed in DFS.

Space complexity: Equivalent to how large can the fringe get.

Completeness: DFS is complete if the search tree is finite, meaning for a given finite search tree, DFS will come up with a solution if it exists.

Optimality: DFS is not optimal, meaning the number of steps in reaching the solution, or the cost spent in reaching it is high.

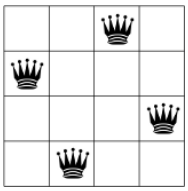
Python Program:

```
#non recursive Depth First search with source and target with 4 puzzle
adj_list = {
  'A' : ['B','C'],
  'B' : ['A'],
  'C' : ['A','D'],
  'D' : ['C','E'],
  'E' : ['F','D'],
  'F' : []
}
traversal_path = []    # empty list
stack = []     #Initialize a stack
visited=[]
def Dfs(adj_list, start, target, path):
    stack.append(start)      #to insert into list   like a push
    while stack:
        s=stack.pop()       # like a pop will run on last element
        path.append(s)       # to create path list
        visited.append(s)    #all vistied node
        if s == target:
            return path
        for neighbour in adj_list[s]:
            if neighbour not in visited:
                stack.append(neighbour)     # all child of out node has been pushed
                    traversal_path    =    Dfs(adj_list,    'A',    'F', traversal_path)
print(traversal_path)
```

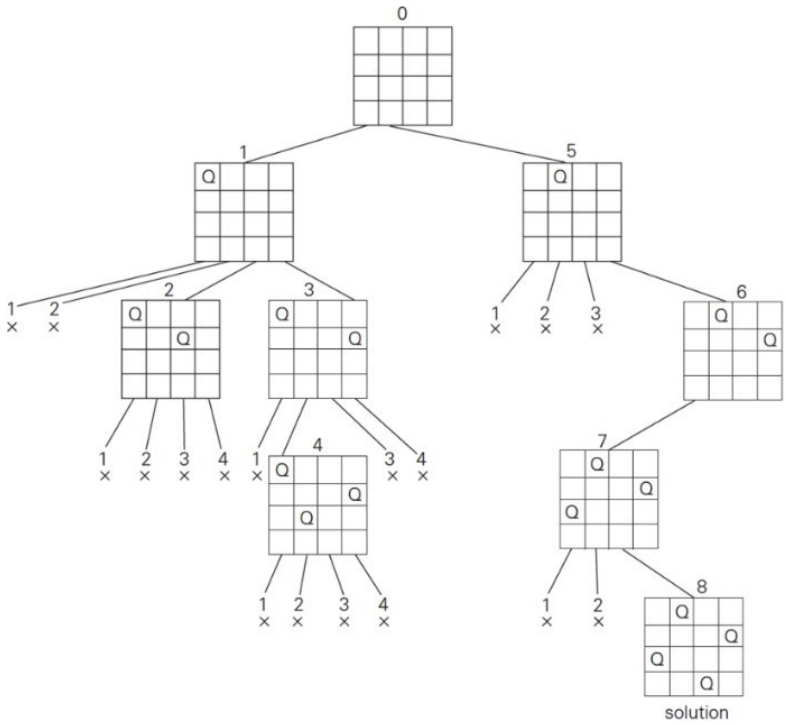
Ex-6	Write a program to implement 4/8 Queen Problems
Expected Date	Third Week
CO	CO-1

The **4-Queens Problem** consists in placing four queens on a 4 x 4 chessboard so that no two queens can capture each other. That is, no two queens are allowed to be placed on the same row, the same column or the same diagonal.

The following figure illustrates a solution to the 4-Queens Problem: none of the 4 queens can capture each other.



Although this particular problem isn't very impressive, keep in mind that you can generalize it to chessboards with.



Python Program:

```
#non recursive Depth First search with source and target with 4 queen
#4 Queen Problem
adj_list = {
 'A' : ['B','C','D','E'],
 'B' : ['F','G'],
 'C' : ['H'],
 'D' : ['I'],
 'E' : ['J','K'],
 'F' : [],
  'G' : ['M'],
  'H' : ['N'],
  'I' : ['O'],
  'J' : ['P'],
  'K' : [],
  'M' : [],
  'N' : ['S'],#GOAL STATE
  'O' : ['T'],#GOAL STATE
  'P' : [],
  'T' : []
  'S'
}
traversal_path = [] # empty list
stack = []     #Initialize a stack
visited=[]
def Dfs(adj_list, start, target, path):
   stack.append(start)      #to insert into list like a push
   while stack:
      s=stack.pop(0)       # like a pop will run on last element
      print(s)
      path.append(s)       # to create path list
      visited.append(s)   #all  vistied  node
      if s == target:
         return path
      if adj_list[s] == []:
         print("dead end")
      for neighbour in adj_list[s]:
      if neighbour not in visited:
         stack.append(neighbour)      # all child of out node has been pushed
      if neighbour == []:
        print("dead end")
traversal_path = Dfs(adj_list, 'A', 'T', traversal_path)
#print(traversal_path)
```

Ex-7	Write a program to implement first order logic
Expected Date	Fourth Week
CO	CO-2

Python library that enables using logic programming in python. The aim of the library is to explore ways to use symbolic reasoning with machine learning. Pytholog supports probabilities. Pytholog gives facts indices (first term) and uses binary search to search for relevant facts instead of looping over all knowledge base. So when defining rules, make sure that the main search terms are in the first position to speed up the search queries

Python Program

```
import pytholog as pl
new_kb = pl.KnowledgeBase("flavor")
new_kb(["likes(noor, sausage)",
        "likes(melissa, pasta)",
        "likes(dmitry, cookie)",
        "likes(nikita, sausage)",
        "likes(assel, limonade)",
        "food_type(gouda, cheese)",
        "food_type(ritz, cracker)",
        "food_type(steak, meat)",
        "food_type(sausage, meat)",
        "food_type(limonade, juice)",
        "food_type(cookie, dessert)",
        "flavor(sweet, dessert)",
        "flavor(savory, meat)",
        "flavor(savory, cheese)",
        "flavor(sweet, juice)",
        "food_flavor(X, Y) :- food_type(X, Z), flavor(Y, Z)",
        "dish_to_like(X, Y) :- likes(X, L), food_type(L, T), flavor(F, T), food_flavor(Y, F), neq(L, Y)"])
```

```
new_kb.query(pl.Expr("likes(noor, sausage)")) ['Yes']

new_kb.query(pl.Expr("likes(noor, pasta)"))  ['No']

print(new_kb.query(pl.Expr("food_flavor(What, sweet)")))

from time import time

start = time()

print(new_kb.query(pl.Expr("food_flavor(What, sweet)")))

print(time() - start)
```

```
#Program 2

kb1 = pl.KnowledgeBase("friendship")

kb1(["friend(krishna,sudama)", "friend(arjun, krishna)"])

kb1.query(pl.Expr("friend(sudama, arjun)"))

kb1.query(pl.Expr("friend(sudama, krishna)"))

kb1.query(pl.Expr("friend(krishna,sudama)"))

#End of Program 2
```

Ex-8	Write a program to demonstrate the working of Bayesian network.
Expected Date	Fourth Week
CO	CO-2

Problem Statement: Write a program to demonstrate the working of Bayesian network using heart disease dataset "https://archive.ics.uci.edu/ml/machine-learning-databases/heart-disease/processed.cleveland.data". Calculate the probability of model = BayesianModel([("A","B"),("B","C"),("C","RESULT")]) while considering the following 14 attributes

A. #3 (age)
B. #4 (sex)
C. #9 (cp)
D. #10 (trestbps)
E. #12 (chol)
F. #16 (fbs)
G. #19 (restecg)
H. #32 (thalach)
I. #38 (exang)
J. #40 (oldpeak)
K. #41 (slope)
L. #44 (ca)
M. #51 (thal)
N. #58 (num) (the predicted attribute)

Python Program:

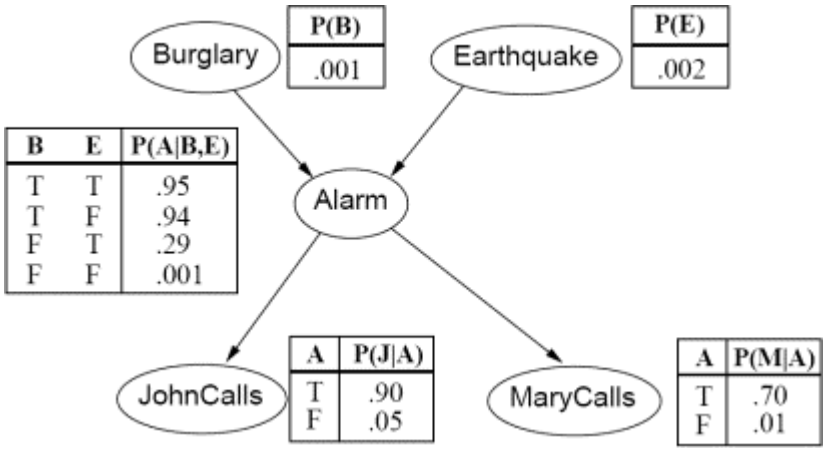
```
import numpy as np

import pandas as pd

import csv

from pgmpy.estimators import MaximumLikelihoodEstimator

from pgmpy.models import BayesianModel

from pgmpy.inference import VariableElimination

heartDisease = pd.read_csv('heart.csv')

heartDisease = heartDisease.replace('?',np.nan)

print('Sample instances from the dataset are given below'
)

print(heartDisease.head())

print('\n Attributes and datatypes')

print(heartDisease.dtypes)
```

```
model= BayesianModel([('age','heartdisease'),('sex','heartdisease'),('exang',
'heartdisease'),('cp','heartdisease'),('heartdisease','restecg'),('heartdisea
se','chol')])

print('\nLearning CPD using Maximum likelihood estimators')

model.fit(heartDisease,estimator=MaximumLikelihoodEstimator)

print('\n Inferencing with Bayesian Network:')

HeartDiseasetest_infer = VariableElimination(model)

print('\n 1. Probability of HeartDisease given evidence= restecg')

q1=HeartDiseasetest_infer.query(variables=['heartdisease'],evidence={'restecg
':1})

print(q1)

print('\n 2. Probability of HeartDisease given evidence= cp ')

q2=HeartDiseasetest_infer.query(variables=['heartdisease'],evidence={'cp':2})

print(q2)
```

Output

```
Learning CPD using Maximum likelihood estimators

 Inferencing with Bayesian Network:

 1. Probability of HeartDisease given evidence= restecg
+-----------------+----------------------+
| heartdisease    |   phi(heartdisease)  |
+=================+======================+
| heartdisease(0) |               0.1012 |
+-----------------+----------------------+
| heartdisease(1) |               0.0000 |
+-----------------+----------------------+
| heartdisease(2) |               0.2392 |
+-----------------+----------------------+
| heartdisease(3) |               0.2015 |
+-----------------+----------------------+
| heartdisease(4) |               0.4581 |
+-----------------+----------------------+

2. Probability of HeartDisease given evidence= cp
+-----------------+----------------------+
| heartdisease    |   phi(heartdisease)  |
+=================+======================+
| heartdisease(0) |               0.3610 |
+-----------------+----------------------+
| heartdisease(1) |               0.2159 |
+-----------------+----------------------+
| heartdisease(2) |               0.1373 |
+-----------------+----------------------+
| heartdisease(3) |               0.1537 |
+-----------------+----------------------+
| heartdisease(4) |               0.1321 |
+-----------------+----------------------+
```

Problem Statement-2: Write a program to demonstrate the working of Bayesian network for the following graph:



- Calculate the probability of a burglary if John and Mary calls (0: True, 1: False)
- Calculate the probability of alarm starting if there is a burglary and an earthquake (0: True, 1: False)
- Calculate the probability of alarm starting if there is a burglary and an earthquake (0: True, 1: False)

Python Program:

```
pip install pgmpy

import pgmpy. inference

import pgmpy. models

import networkx as nx

import pylab as plt

model = pgmpy. models . BayesianNetwork ([( ' Burglary ' , 'Alarm ') , ( ' Ea
rthquake ' , 'Alarm ') ,

                                          ( 'Alarm ' , ' JohnCalls ') , ( 'Al
arm ' , 'MaryCalls ' ) ] )

# Define conditional probability distributions (CPD)

# Probability of burglary (True , False )
```

```
cpd burglary = pgmpy. factors . discrete .TabularCPD(" Burglary " , 2 , [[0.0
01] , [0.999]])

# Probability of earthquake (True , False )

cpd earthquake = pgmpy. factors . discrete .TabularCPD("Earthquake" , 2 , [[0
.002] , [0.998]])

# Probability of alarm going of (True , False ) given a burglary and/or earth
quake

cpd alarm = pgmpy. factors . discrete .TabularCPD( 'Alarm ' , 2 , [[0.95 , 0.
94 , 0.29 , 0.001] , [0.05 , 0.06 , 0.71 , 0.999]] ,

                                                    evidence =['Burglary ' , 'E
arthquake '] , evidence card =[2, 2])

# Probability that John calls (True , False ) given that the alarm has sounde
d

cpd john = pgmpy. factors . discrete .TabularCPD( ' JohnCalls ' , 2 , [[0.90
, 0.05] , [0.10 , 0.95]] , evidence =['Alarm '] , evidence card =[2])

# Probability that Mary calls (True , False ) given that the alarm has sounde
d

cpd mary = pgmpy. factors . discrete .TabularCPD( ' MaryCalls ' , 2 , [[0.70
, 0.01] , [0.30 , 0.99]] , evidence =['Alarm '] , evidence card =[2])

# Add CPDs to the network structure

model . add cpds ( cpd burglary , cpd earthquake , cpd alarm , cpd john , cpd
 mary)

# Check i f the model is valid , throw an exception otherwise

model . check model ()

# Print probability distributions

print ( ' Probability distribution , P( Burglary ) ')

print ( cpd burglary )

print ()

print ( ' Probability distribution , P(Earthquake ) ')

print ( cpd earthquake )
```

```
print ()

print ( ' Joint probability distribution , P(Alarm | Burglary , Earthquake )
')

print ( cpd alarm )

print ()

print ( ' Joint probability distribution , P( JohnCalls | Alarm) ')

print ( cpd john )

print ()

print ( ' Joint probability distribution , P(MaryCalls | Alarm) ')

print (cpd mary) print () # Plot the model

nx. draw(model , with labels=True) plt . savefig ( ' alarm1 .png ') plt . clo
se ()

# Perform variable elimination for inference

# Variable elimination (VE) is a an exact inference algorithm in bayesian net
works

infer = pgmpy. inference . VariableElimination (model)

# Calculate the probability of a burglary i f John and Mary calls (0: True ,
1: False )

posterior probability = infer . query ([ ' Burglary '] , evidence={'JohnCalls
 ' : 0 , 'MaryCalls ' : 0})

# Print posterior probability

print ( ' Posterior probability of Burglary i f JohnCalls (True) and MaryCall
s(True ) ')

print ( posterior probability )

print ()

# Calculate the probability of alarm starting i f there is a burglary and an
earthquake (0: True , 1: False )

posterior probability = infer . query ([ ' Alarm '] , evidence= {' Burglary '
 : 0 , 'Earthquake ' : 0})
```

```
# Print posterior probability

print ( ' Posterior probability of Alarm sounding i f Burglary (True) and Ear
thquake(True ) ')

print ( posterior probability )

print ()
```

Output:

```
Probability distribution, P(Burglary)
+-------------+-------+
| Burglary(0) | 0.001 |
+-------------+-------+
| Burglary(1) | 0.999 |
+-------------+-------+

Probability distribution, P(Earthquake)
+---------------+-------+
| Earthquake(0) | 0.002 |
+---------------+-------+
| Earthquake(1) | 0.998 |
+---------------+-------+

Joint probability distribution, P(Alarm | Burglary, Earthquake)
+------------+---------------+---------------+---------------+---------------+
| Burglary   | Burglary(0)   | Burglary(0)   | Burglary(1)   | Burglary(1)   |
+------------+---------------+---------------+---------------+---------------+
| Earthquake | Earthquake(0) | Earthquake(1) | Earthquake(0) | Earthquake(1) |
+------------+---------------+---------------+---------------+---------------+
| Alarm(0)   | 0.95          | 0.94          | 0.29          | 0.001         |
+------------+---------------+---------------+---------------+---------------+
| Alarm(1)   | 0.05          | 0.06          | 0.71          | 0.999         |
+------------+---------------+---------------+---------------+---------------+

Joint probability distribution, P(JohnCalls | Alarm)
+--------------+----------+----------+
| Alarm        | Alarm(0) | Alarm(1) |
+--------------+----------+----------+
| JohnCalls(0) | 0.9      | 0.05     |
+--------------+----------+----------+
| JohnCalls(1) | 0.1      | 0.95     |
+--------------+----------+----------+

Joint probability distribution, P(MaryCalls | Alarm)
+--------------+----------+----------+
| Alarm        | Alarm(0) | Alarm(1) |
+--------------+----------+----------+
| MaryCalls(0) | 0.7      | 0.01     |
+--------------+----------+----------+
| MaryCalls(1) | 0.3      | 0.99     |
+--------------+----------+----------+

Posterior probability of Burglary if JohnCalls(True) and MaryCalls(True)
+-------------+-----------------+
| Burglary    |   phi(Burglary) |
+=============+=================+
| Burglary(0) |          0.2842 |
+-------------+-----------------+
| Burglary(1) |          0.7158 |
+-------------+-----------------+

Finding Elimination Order: :    0/0 [00:00<?, ?it/s]

    0/0 [00:00<?, ?it/s]

Posterior probability of Alarm sounding if Burglary(True) and Earthquake(True)
+----------+--------------+
| Alarm    |   phi(Alarm) |
+==========+==============+
| Alarm(0) |       0.9500 |
+----------+--------------+
| Alarm(1) |       0.0500 |
+----------+--------------+
```

Ex-9	Write a program to Implement pattern recognition problems of handwritten character/ digit recognition
Expected Date	Fifth Week
CO	CO-2

Problem Statement: Write a program to implement pattern recognition problems of handwritten digit recognition using dataset as "https://scikit-learn.org/stable/modules/generated/sklearn.datasets.load_digits.html"

[Hint: From the dataset, given attributes ('DESCR', 'data', 'feature_names', 'frame', 'images', 'target', 'target_names'), consider only 'images' and 'target']

Python Program:

```
from sklearn import datasets

digits = datasets.load digits ()

dir ( digits )

digits . images . shape

print ( digits . images [0])

import matplotlib . pyplot as plt plt . imshow( digits . images [0] ,cmap='bi
nary ') plt . show()

def plot multi ( i ):

  ' ' ' Plots 16 digits , starting with digit i ' ' '

   nplots = 16 fig = plt . figure ( figsize =(15 ,15))

   for j in range ( nplots ):

     plt . subplot (4 ,4 , j+1)

     plt . imshow( digits . images [ i+j ] , cmap='binary ')

     plt . t i t l e ( digits . target [ i+j ])

     plt . axis ( ' off ') plt . show() plot multi (0)
```

Output:

*'DESCR','data','feature_names','frame','images','target','target_names'+

Ex-10	Write a program to Implement pattern recognition problems of speech recognition
Expected Date	Sixth Week
CO	CO-3

Recognition of Spoken Words: Speech recognition means that when humans are speaking, a machine understands it. Here we are using Google Speech API in Python to make it happen. We need to install the following packages for this −

- Pyaudio − It can be installed by using pip install Pyaudio command.
- SpeechRecognition − This package can be installed by using pip install SpeechRecognition.
- Google-Speech-API − It can be installed by using the command pip install google-api-python-client.
- Pyaudio − It can be installed by using pip install Pyaudio command.
- SpeechRecognition − This package can be installed by using pip install SpeechRecognition.
- Google-Speech-API − It can be installed by using the command pip install google-api-python-client.

Python Program:

```
import speech_recognition as sr

r= sr.Recognizer()

with sr.Microphone() as source:

   print("Speak anything :")

   audio = r.listen(source)

   try:

      text = r.recognize_google(audio)

      if text =='exit':

         flag=0

      print("You said : {}".format(text))

   except:

      print("Sorry could not recognize what you said")
```

Output:

```
Speak anything :
You said : hello hello
```

```
flag = 1
while(flag):
    with sr.Microphone() as source:
        print("Speak anything :")
        audio = r.listen(source)
        try:
            text = r.recognize_google(audio)
            if text =='exit':
                flag=0
            print("You said : {}".format(text))
        except:
            printf("Sorry could not recognize what you said")
print("Exit form the loop")
```

Output:

```
Speak anything :
You said : hello everyone hi hello everyone
Speak anything :
You said : speed correction
Speak anything :
You said : exit
Exit form the loop
```

Ex-11	Write a program to implement Naive Bayes classification problem
Expected Date	Sixth Week
CO	CO-3

Naive Bayes classification: Naive Bayes classifiers are a collection of classification algorithms based on Bayes' Theorem. It is not a single algorithm but a family of algorithms where all of them share a common principle, i.e. every pair of features being classified is independent of each other.

Imported Libraries:

- sklearn import preprocessing
- sklearn.naive_bayes import GaussianNB

Python Program:

```
weather =['sunny','sunny','overcast','rainy','rainy','rainy','overcast','sunn
y','rainy','sunny','overcast','overcast','rainy']
temp = ['hot','hot','hot','mild','cool','cool','cool','mild','cool','mild','m
ild','hot','hot']
play = ['no','no','yes','yes','no','yes','no','yes','yes','no','yes','no','no
']
from sklearn import preprocessing
le = preprocessing.LabelEncoder()
weather_encoded = le.fit_transform(weather)
print(weather_encoded)
type(weather_encoded)
temp_encoded = le.fit_transform(temp)
play_encoded = le.fit_transform(play)
print(temp_encoded)
print(play_encoded)
features = list(zip(weather_encoded,temp_encoded))
print(features)
from sklearn.naive_bayes import GaussianNB
model = GaussianNB()
model.fit(features,play_encoded)
predicted = model.predict([[0,2]])
print("Predicted Value = ",predicted)
```

Output: Predicted Value = [1]

Ex-12	Write a program to implement k-mean clustering problem
Expected Date	Sixth Week
CO	CO-3

K MEANS ALGORITHM:-

1. Specify number of clusters K.
2. Initialize centroids by first shuffling the dataset and then randomly selecting K data points for the centroids without replacement.
3. Keep iterating until there is no change to the centroids. i.e assignment of data points to clusters isn't changing.
   - Compute the sum of the squared distance between data points and all centroids.
   - Assign each data point to the closest cluster (centroid).
   - Compute the centroids for the clusters by taking the average of the all data points that belong to each cluster.

Python Program:

```
[15] import matplotlib.pyplot as plt
     from sklearn import datasets
     from sklearn.cluster import KMeans
```

```
[17] x=iris.data

     y=iris.target
     print(x)
     print(y)
```

```
[6.9 3.1 4.9 1.5]
[5.5 2.3 4.  1.3]
[6.5 2.8 4.6 1.5]
[5.7 2.8 4.5 1.3]
[6.3 3.3 4.7 1.6]
[4.9 2.4 3.3 1. ]
[6.6 2.9 4.6 1.3]
[5.2 2.7 3.9 1.4]
[5.  2.  3.5 1. ]
[5.9 3.  4.2 1.5]
[6.  2.2 4.  1. ]
[6.1 2.9 4.7 1.4]
[5.6 2.9 3.6 1.3]
[6.7 3.1 4.4 1.4]
[5.6 3.  4.5 1.5]
[5.8 2.7 4.1 1. ]
```

```
+ Code  + Text  Copy to Drive

[17]
[6.7 3.1 4.4 1.4]
[5.6 3.  4.5 1.5]
[5.8 2.7 4.1 1. ]
[6.2 2.2 4.5 1.5]
[5.6 2.5 3.9 1.1]
[5.9 3.2 4.8 1.8]
[6.1 2.8 4.  1.3]
[6.3 2.5 4.9 1.5]
[6.1 2.8 4.7 1.2]
[6.4 2.9 4.3 1.3]
[6.6 3.  4.4 1.4]
[6.8 2.8 4.8 1.4]
[6.7 3.  5.  1.7]
[6.  2.9 4.5 1.5]
[5.7 2.6 3.5 1. ]
[5.5 2.4 3.8 1.1]
[5.5 2.4 3.7 1. ]
[5.8 2.7 3.9 1.2]
[6.  2.7 5.1 1.6]
[5.4 3.  4.5 1.5]
[6.  3.4 4.5 1.6]
[6.7 3.1 4.7 1.5]
[6.3 2.3 4.4 1.3]
[5.6 3.  4.1 1.3]
[5.5 2.5 4.  1.3]
[5.5 2.6 4.4 1.2]
[6.1 3.  4.6 1.4]
[5.8 2.6 4.  1.2]
```

+ Code + Text Copy to Drive

```
[17] [6.  2.7 5.1 1.6]
     [5.4 3.  4.5 1.5]
     [6.  3.4 4.5 1.6]
     [6.7 3.1 4.7 1.5]
     [6.3 2.3 4.4 1.3]
     [5.6 3.  4.1 1.3]
     [5.5 2.5 4.  1.3]
     [5.5 2.6 4.4 1.2]
     [6.1 3.  4.6 1.4]
     [5.8 2.6 4.  1.2]
     [5.  2.3 3.3 1. ]
     [5.6 2.7 4.2 1.3]
     [5.7 3.  4.2 1.2]
     [5.7 2.9 4.2 1.3]
     [6.2 2.9 4.3 1.3]
     [5.1 2.5 3.  1.1]
     [5.7 2.8 4.1 1.3]
     [6.3 3.3 6.  2.5]
     [5.8 2.7 5.1 1.9]
     [7.1 3.  5.9 2.1]
     [6.3 2.9 5.6 1.8]
     [6.5 3.  5.8 2.2]
     [7.6 3.  6.6 2.1]
     [4.9 2.5 4.5 1.7]
     [7.3 2.9 6.3 1.8]
     [6.7 2.5 5.8 1.8]
     [7.2 3.6 6.1 2.5]
     [6.5 3.2 5.1 2. ]
```

+ Code + Text Copy to Drive

```
iris=datasets.load_iris()
print(iris)
```

```
[5.7, 2.8, 4.1, 1.3],
[6.3, 3.3, 6. , 2.5],
[5.8, 2.7, 5.1, 1.9],
[7.1, 3. , 5.9, 2.1],
[6.3, 2.9, 5.6, 1.8],
[6.5, 3. , 5.8, 2.2],
[7.6, 3. , 6.6, 2.1],
[4.9, 2.5, 4.5, 1.7],
[7.3, 2.9, 6.3, 1.8],
[6.7, 2.5, 5.8, 1.8],
[7.2, 3.6, 6.1, 2.5],
[6.5, 3.2, 5.1, 2. ],
[6.4, 2.7, 5.3, 1.9],
[6.8, 3. , 5.5, 2.1],
[5.7, 2.5, 5. , 2. ],
[5.8, 2.8, 5.1, 2.4],
[6.4, 3.2, 5.3, 2.3],
[6.5, 3. , 5.5, 1.8],
[7.7, 3.8, 6.7, 2.2],
[7.7, 2.6, 6.9, 2.3],
[6. , 2.2, 5. , 1.5],
[6.9, 3.2, 5.7, 2.3],
[5.6, 2.8, 4.9, 2. ],
[7.7, 2.8, 6.7, 2. ],
```

Cannot save changes

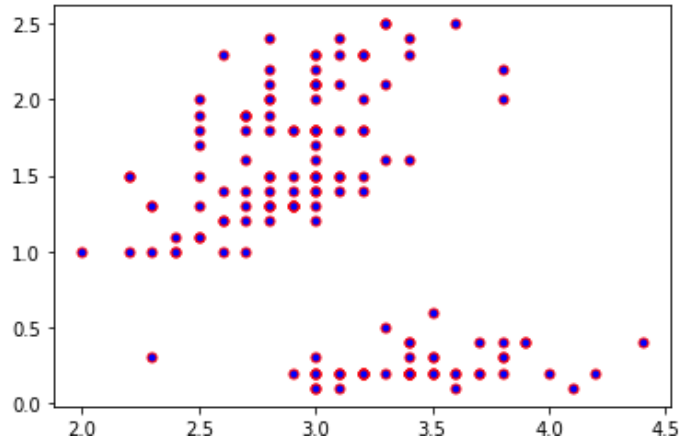
Code + Text Copy to Drive RAM Disk Editing

```
[7.2, 3. , 5.8, 1.6],
[7.4, 2.8, 6.1, 1.9],
[7.9, 3.8, 6.4, 2. ],
[6.4, 2.8, 5.6, 2.2],
[6.3, 2.8, 5.1, 1.5],
[6.1, 2.6, 5.6, 1.4],
[7.7, 3. , 6.1, 2.3],
[6.3, 3.4, 5.6, 2.4],
[6.4, 3.1, 5.5, 1.8],
[6. , 3. , 4.8, 1.8],
[6.9, 3.1, 5.4, 2.1],
[6.7, 3.1, 5.6, 2.4],
[6.9, 3.1, 5.1, 2.3],
[5.8, 2.7, 5.1, 1.9],
[6.8, 3.2, 5.9, 2.3],
[6.7, 3.3, 5.7, 2.5],
[6.7, 3. , 5.2, 2.3],
[6.3, 2.5, 5. , 1.9],
[6.5, 3. , 5.2, 2. ],
[6.2, 3.4, 5.4, 2.3],
[5.9, 3. , 5.1, 1.8]]), 'target': array([0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2]), 'frame': None, 'target_names': array(['setosa', 'versicol
```

+ Code + Text Copy to Drive

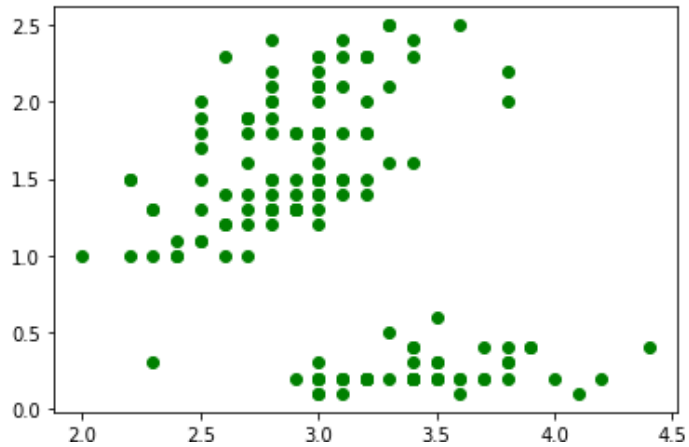
[18] plt.scatter(x[:,1],x[:,3],color="b",marker="o",edgecolors="red",s=25)

```
<matplotlib.collections.PathCollection at 0x7f0414e76ed0>
```



[21] plt.scatter(x[:,1],x[:,3],color="G")

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:1: MatplotlibDepre
  """Entry point for launching an IPython kernel.
<matplotlib.collections.PathCollection at 0x7f0414309690>
```



```
kmc=KMeans(n_clusters=3,init="random",n_init=20,max_iter=10,tol=1e-04,random_state=0)
```

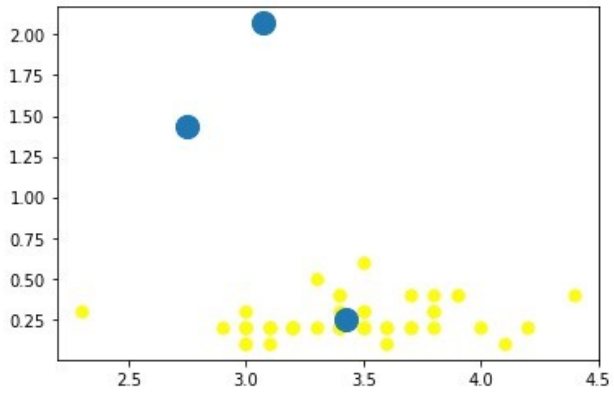
```
y_kmc=kmc.fit_predict(x)
y_kmc
```

```
array([0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 1, 1, 2, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2, 1, 2, 2, 2, 2, 1, 2, 2, 2,
       2, 2, 2, 1, 1, 2, 2, 2, 2, 1, 2, 1, 2, 1, 2, 2, 1, 1, 2, 2, 2, 2,
       2, 1, 2, 2, 2, 2, 1, 2, 2, 2, 1, 2, 2, 2, 1, 2, 2, 1], dtype=int32)
```

```
plt.scatter(x[y_kmc==0,1],x[y_kmc==0,3],s=50,c="pink",label="cluster1")
plt.scatter(x[y_kmc==0,1],x[y_kmc==0,3],s=50,c="white",label="cluster1")
plt.scatter(x[y_kmc==0,1],x[y_kmc==0,3],s=50,c="yellow",label="cluster1")
plt.scatter(kmc.cluster_centers_[:,1],kmc.cluster_centers_[:,3],s=200)
```

```
[25] plt.scatter(x[y_kmc==0,1],x[y_kmc==0,3],s=50,c="pink",label="cluster1")
     plt.scatter(x[y_kmc==0,1],x[y_kmc==0,3],s=50,c="white",label="cluster1")
     plt.scatter(x[y_kmc==0,1],x[y_kmc==0,3],s=50,c="yellow",label="cluster1")
     plt.scatter(kmc.cluster_centers_[:,1],kmc.cluster_centers_[:,3],s=200)
```

```
<matplotlib.collections.PathCollection at 0x7f0414216d90>
```



Ex-13	Write a program to convert text to speech using NLP tool
Expected Date	Sixth Week
CO	CO-3

Python Program:

```
import speech_recognition as sr
import pyttsx3
# Initialize the recognizer
r = sr.Recognizer()
# Function to convert text to
# speech
def SpeakText(command):

    # Initialize the engine
    engine = pyttsx3.init()
    engine.say(command)
    engine.runAndWait()
text=" hi how are you, today is sunny day"
SpeakText(text)
# Loop infinitely for user to
# speak

while(1):

    # Exception handling to handle
    # exceptions at the runtime
    try:
```

```
    # use the microphone as source for input.
    with sr.Microphone() as source2:

        # wait for a second to let the recognizer
        # adjust the energy threshold based on
        # the surrounding noise level
        r.adjust_for_ambient_noise(source2, duration=0.2)

        #listens for the user's input
        audio2 = r.listen(source2)

        # Using ggogle to recognize audio
        MyText = r.recognize_google(audio2)
        MyText = MyText.lower()

        print("Did you say "+MyText)
        SpeakText(MyText)

except sr.RequestError as e:
    print("Could not request results; {0}".format(e))

except sr.UnknownValueError:
    print("unknown error occured")
```

End