Exno:1 Basic Python Program

1 Program to print the Number of occurance of any element in list:

```
def Count x (list, x)
    Count = 0
    for (ele in list :
        if (ele == x)
            Count = Count + 1
    return Count
List = [8, 6, 8, 10, 8, 20, 20, 30, 8, 8]
x = 8
print ("{} has occured {} times.". format (x = Count x (List. x)))
```

Output:

8 has occured 5 times

2 Program to print the index of element in list:

```
my_list = [10, 20, 30, 40, 50]
element_to_find = 30
if element_to_find in list :
    index = my_list. index (element_to_find)
    print (f" The index of {element_to_find} is {index}.")
else
    print (" Not in list");
```

Output:

The index of 30 is 2

3 Program to Return the String is palindrome or Not

```
def ispalin(text)
    text = text.lower().replace(" ","")
    return text == text[::-1]
text = "hoh"
if ispalin(text)
    print("it is palindrome!");
else:
    print("it is Not palindrome!");
```

output

it is palindrome

4) Remove the element in the list.

```
my_list = [10,20,30,40,50]
remove = 30
if remove in my_list
    my_list.remove(remove)
    print(f"The element {remove} has Removed")
    print("updated list", my_list)
else:
    print(f"The element {remove} is Not in list")
```

Output

The element 30 has Removed

updated list : [10,20,40,50]

5] Check the greater Number

```
a = 10
b = 20
if (a>b):
    print(f"{a} is greater than {b}")
elif (b>a):
    print(f"{b} is greater than {a}")
else:
    print("Both are equal")
```

Output:

20 is greater than 10.

6] Program to add the two list and Return Count.

```
List1 = [1,2,3]
list2 = [4,5,6,7]
Combained list = list1 + list2
Count = len (Combained list)
print ("Combained list :" Combained list)
print ("Count of element:" Count)
```

Output:

Combained list: [1,2,3,4,5,6,7]

Count of list: 6

7] Program to add the list value and return

```
list 1 = [1, 2, 3]
list 2 = [4, 5, 6]
result = []
for i in range (len (list 1)):
    result.append (list 1[i] + list 2[i])
print ("Sum of element of corresponding position :", result)
```

Output

Sum of element at corresponding position : [5, 7, 9]

8] Swapping the element between two list.

```
list 1 = [1, 2 3]
list 2 = [4, 5, 6]
list 1, list 2 = list 2, list 1
print ("list 1 :", list 1)
print ("list 2 :", list 2)
```

Output

list 1 = [4, 5, 6]
list 2 = [1, 2, 3]

9] Reverse the array :

```
my_list = [10, 20, 30, 40, 50]
my_list.Reverse()
print ("Reverse list :", my_list)
```

Output

Reverse list [50, 40, 30, 20, 10]

10] Vowels Count:

```
text = "This is a Sample text"
Vowels = "aeiou AEIOU"
Vowels_Count = 0
for char in text
    if char in Vowels:
        Vowels_Count += 1
print ("Number of Vowels", Vowels_Count)
```

Output:

Number of Vowels: 6

Architecture Recognization & Design Recommendation

Project overview:

AI based system to recognize architectural styles from image and provide recommendation design and Estimating the cost of the design and how to implement.

Data Collection

- → Architecture style dataset
- → Design Elements
- → Estimation Value for implement design.

Model training

- → Recognization
- → Design Recommendation

Target user

- → Architecture Engineer
- → CAD designer
- → Estimator for Design
- → Educational use.

Algorithm:

Convolutional Neural Network (CNN)

Domain: Architecture

Architecture design Recommendation and Recognization using ML algorithm.

Objective :

To develope an AI based system that accurately recognizes architectural styles from image and provide design recommendation based on user performance and Recognized style.

Problem :

Architecture styles are defined by distinctive visual and structure element that can be challenging to identify manually. especially given the variety and evolution of styles. Further More translating style recognition into actional design recommendation requires an intelligent system capable of understanding user preference and providing design solution.

Solution :

The solution involves creating a two part system : one for architecture style recognition and another for design recommendation. The recognition uses Machine learning to classify architecture style from image. The recommendation system generate a design based on user input.

Abstract:

In Contemporary Architecture, the fusion of Artificial intelligence (AI) with design processes hold the potential to revolutionize how architecture styles are identified and how design recommendation are generate. This paper proposes a novel It propose a novel AI-based system designed to recognize architecture styles from image and provide tailored design recommendation it integrated with advanced Machine learning algorithm and Computer Vision techniques.

Data Set: Archdaily Dataset
Building Dataset for Architecture Styles.

Architectural Style Recognization algorithm:

i) Convolutional Neural Network (CNN)
It is used in training a large data set and to classify the image.

ii) Vision Transformers (ViTs)
used to process the image which divide the image into patches which helps to predict accurately.

Ex no: 2

N Queen Problem

Aim: To write a python Program to place the Queen on Safe

def ip Safe (board, row, Col, n):

for Side from all the direction.

Program

```
def ip - safe (board, row, Col, n):
    for i in range (Col):
        if board [row][i] == 1:
            return false

    for i, j in zip (range (row, -1, -1), range (Col, -1, -1)):
        if board [i][j] == 1:
            return False.
    return True.

def solve . n (board, Col, n):
    if Col >= n:
        return True

    for i in range (n):
        if ip - safe (board, i, Col, n):
            board [i][col] = 1
            if Solve n (board, Col+1, n):
                return True
            board (i)(Col] = 0
    return False
```

```
def print_board(board, n):
    print("\n Solution:")
    for row in board:
        for cell in row:
            if cell == 1:
                print("Q", end=" ")
            else:
                print(".", end=" ")
        print()

def Solve_n_queen(n):
    board = [[0 for _ in range(n)] for _ in range(n)]
    if not Solve_n_queen_util(board, 0, n):
        print("No Solution exist")
        return False

    print_board(board, n)
    return True

n = int(input("Enter the value of N:"))
Solve_n_queen(n)
```

Result: Thus the N-queen Problem is Executed Successfully

Enter the Value: 4

Solution:

. . Q .
Q . . .
. . . Q
. Q . .

True

Exno: 3

DFS.

Aim:

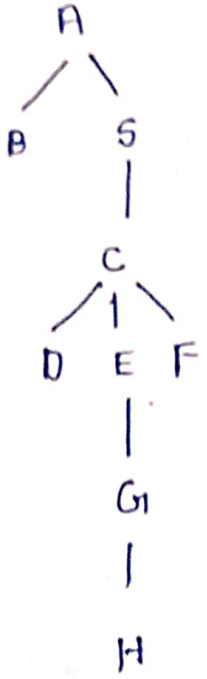
To write a python Program for DFS

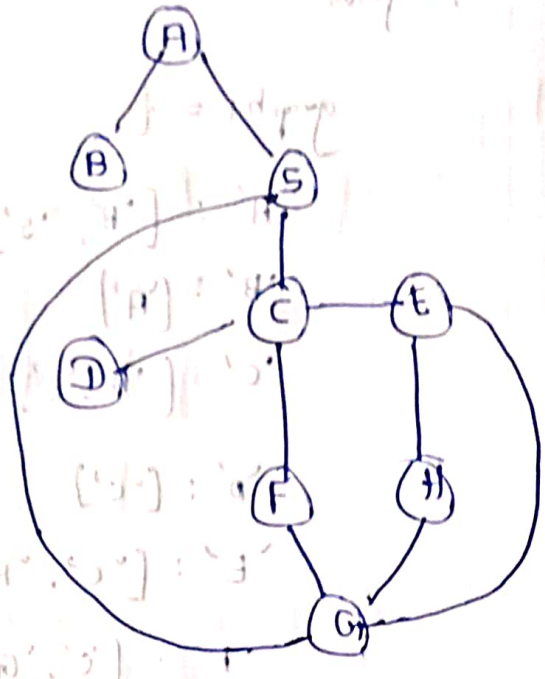
Program

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

def dfs (graph, node, visited = None):
    if visited is None:
        visited = []
    if Node Not in Visited:
        Visited.append (node)
        for neighbour in graph [node]:
            if neighbour not in Visited:
                dfs (graph, neighbour, visited)

return Visited.
Visited = node = dfs (graph 'A'). print (visit).
```

Output

['A', 'B', 'S', 'C', 'D', 'E', 'H', 'G1', 'F']





Result.

Thus the python program for DFS is Executed successfully.

Exno: 4

A* Algorithm:

Aim:

To write a python Code for A* Algorithm

Code:

```
def - astar (start_node, stop_node)
    open_set = set (start_node)
    closed_set = set ()
    g = { }
    parent { }
    g [start_node] = 0
    parent [start_node] = start_node
    while [len (open_set) > 0
        n = none
    for v in open set
        if n = none or g(v) + heuristic [v] < g(n)
    if n == stop_node or Graph_nodes[n]

else
    for (m, weight) in get_neighbour (n)
        not in close_set
            open_set.add (m).
    parent [m] = n
        g[m] = g(n] + weight.
else
    if g(m] > g(n] + weight
        g[m] = g(n] + weight.
        parent [m] = n
if n == none:
    print ("path does not exist !")
    return None
```

```
if n == stop_node:
    path = []
    while parent[n] != n:
        path.append(n)
        n = parent[n]
    path.append(start_node)
    path.reverse()
    print("Path found:", path)
    return
open_set.add(n)
return None

else:
    return None

def heuristic(n):
    H_dist = {
        'A': 11,
        'B': 6,
        'C': 9,
        'D': 1,
        'E': 7,
        'G': 0
    }
    return H_dist[n]

Graph_nodes = {
    'A': [('B', 2), ('E', 3)],
    'B': [('C', 1), ('G', 9)],
    'C': None,
    'E': [('D', 6)],
    'D': [('G', 1)],
}
aStar('A', 'G')
```

Output

path found : ['A' 'B' 'G']

Result

Thus the program for A* is executed Successfully.

Lab 5: Water Jug problem using DFS

Aim:

To create a python code for water jug Problem using DFS

Code:

```
def water_jug (jug1, jug2, target)
    def dfs (j1, j2, seq, visited):
        if j1 == target or j2 == target
            action seq
            visited.add ((j1, j2))
        action = [ ('fill', 1) ('fill', 2) ('empty 1') ('empty 2')]
('pour', 1, 2) ('pour', 2, 1)
        for (action in action:
            if action [0] == "fill"
                if action [1] == 1:
                    next state = (jug1, jug2)
                else:
                    next state = (j1, jug2)
            else
                next state (j1, jug2)
            if action [1] == 1
                next state (0, j2)
```

```
else:
    if action[1] == 1
        amount = min(j1, jug, -j2)
        next state (j1 - amount, j2 + amount)

    else
        amount = min(j2, jug2, j2)
        next state (j1 + amount, j2 - amount)

if next_state not in visited:
    next_seq = seq + (action)
    next_seq.visited
    if result:
        return result.

return None
visited = set()
return dfs(0,0,[], visited)

result = water_jug_dfs(4,3,2)
print(result)
```

Output:

```
[('fill',1), ('fill',2), ('empty',1), ('pour',2,1), ('fill',2)
('pour',2,1)]
```

Result:

Thus the program for water jug problem using DFS is executed Successfully.

Exno. 6

Implementation of Decision Tree classification Technique

Aim:

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

Explanation:

* Import tree from sklearn
* Call the function Decision Tree classifier() from tree
* Assign value for x and y
* Call the function predict for predicting on basic of given random for each given features
* Display the output.

Code:

```
import pandas as pd
from sklearn.tree import Decision tree classifier
data = {
'height' :[152, 155, 172, 185, 167, 180, 157, 180, 164, 177]
'weight' :[45, 57, 72, 85, 68, 78, 22, 90, 66, 88],
'Gender' ['Female', 'Female', 'male', 'Female', 'Male', Female'
'Male', Female', 'Male']
}
df = pd. Data Frame(data)
x = df [['height', 'weight']]
y = df ['Gender']
```

```
classifier = DecisionTreeClassifier()
classifier.fit(x, y)
height = float(input("Enter height (in cm) for prediction: "))
weight = float(input("Enter weight in kg"))
RandomVariable = pd.DataFrame([[height, weight]], columns=
                 [height, weight])
predict_gender = classifier.predict(randomVariable)
print("predict gender for height {height} in cm and weight
      {weight}kg = {predicted_gender[0]}")
```

Output:

Array.

0 99860217 97 8185

Enter the height : 152

Enter the age : 45

predicted gender :- Female

Result:

Thus the program for the DTC was executed successfully.

Exno: 7 Implementation of Artifical Neural Network for an Application using Python - Regression

Aim:

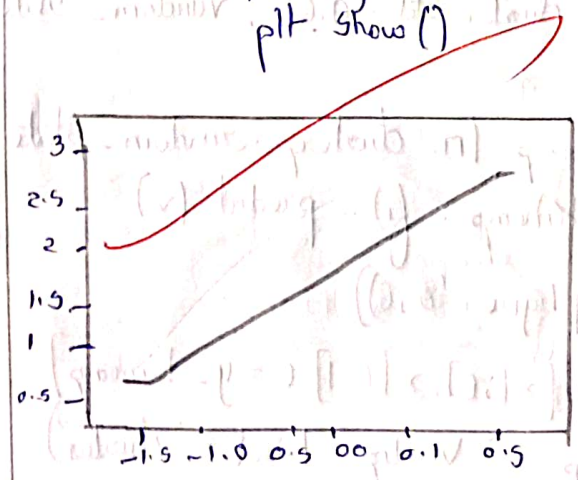
To implement of Artifical Neural Network for an Application using python Regression

Code:

```
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScale
from keras.layer import dense
from keras.optimizers import adamp
import matplotlib.pyplot as plt

np.random.seed(42)
x = np.random.rand(1000,3)
y = 3* x[:,0]+2* x[:,1]**2+1.5*
    np.sin(x[:,2]*np.pi)+np.random.normal
    (0,0.1,1000)

x_train, x_test, y_train, y_test =
train_test_split(x, y, test_size=0.2, random_state=42)
scaler = StandardScaler()
x_train = scaler.fit_transform(x_train)
y_test = scaler.transform(x_test)

Model = Sequential().
model.add(Dense(10, input_dim = x_train.shape[1],
          activation='relu'));
```

Building the ANN Model

```
Model.add(Dense(10, activation='relu'))
Model.add(Dense(1, activation='linear'))
Model.compile(optimizer='adam', learning_rate=0.01, loss='mean_squared_error')
history = model.fit(X_train, y_train, epochs=50, batch_size=32, validation_split=0.2)
y_pred = model.predict(X_test)
mse = np.mean((y_test - y_pred.flatten())**2)
print(f'mean Squared Error {mse:.4f}')
plt.figure(figsize=(12,6))
plt.plot(history, history['loss'], label='Training loss')
plt.plot(history history['val_loss'], label='Validation loss')
plt.title('Training & Validation loss')
plt.xlabel('Epoch')
plt.ylabel('loss')
plt.legend()
plt.show()
```




```
Epoch 0/100 ...
Epoch 1/100 ...
```

Result:

Thus the ANN using python has been executed successfully.

Exno: 8

K means clustering technique using python.

Aim:

To implement a K - Means clustering technique using python language

Explanation.

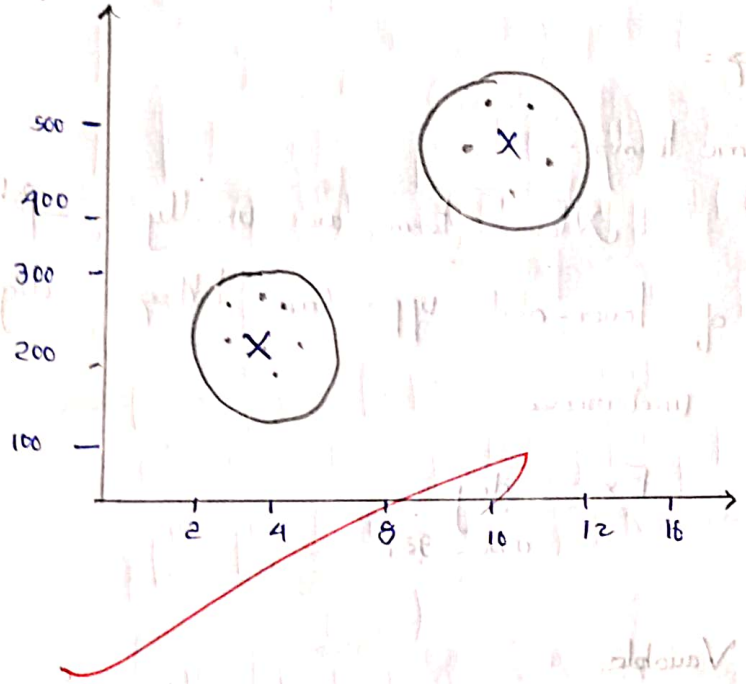
* import K means from Sklearn. cluster
* Assign x and Y
* Call the function K means ().
* perform Scatter operation and display the output.

Code.

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.cluster import Kmeans
from sklearn.datasets import make_blobs
X, y_true = make_blobs (n_samples = 300,
    Center = 3, cluster_std = 0.60, random_state = 0)
K = 3
Kmeans = Kmeans (n_clusters = 3, random_state = 0)
y_Kmeans = Kmeans.fit_predict (x)
plt.figure (figure = (5,6))
plt.Scatter (x[:,0] x[0:1] c = y_Kmeans)
{ = So, Cmpp = 'Viridis' label = 'cluster')
    Center = Kmeans.cluster.Center
    plt Scatter( Center [:,0], Centers [:,1] C = 'red'),
    S = 200, alpha = 0.75 marker = 'x',
```

```
label = 'Centroids')
plt.title('K-means clustering Result')
plt.xlabel('Feature 1')
plt.ylabel('Feature 2')
plt.legend()
plt.show()
```

Output



Result.

Thus the program for K means clustering technique was executed successfully.

Exno: 9

Introduction to PROLOG

Aim:

To learn Prolog terminologies and write the basic programs.

Terminologies:

Atomic terms:

Atomic term are usually strings make up of lower and upper case letters, digit and the underscore.

Ex: dog
abc_321

Variable

Variable are strings of letter, digit and the underscore starting with a capital

Ex:
Dog
Apple_420.

Component Terms

Component terms are made up of a Prolog atom and a number of argument are separated by commas

Ex ?- bigger(elephant, x)
?|g(x,-),1)

Facts:

A fact is a predicate followed by a dot.

Ex:

bigger_animal(whale)
life_is_beautiful.

Rules:

A rule consists of a head (a predicate) and a body (a sequence)

Ex: is_smaller(X,Y) :- is_bigger(Y,X)

Source Code:

```
woman(mia).
woman(jody).
woman(yolanda).
plays_air_guitar(jody).
```

Output:

Query 1 woman(mia).

true

Query 2: play air guitar(mia)

false.

KB2

happy (yolanda)

listens2music (mia)

listens2music (yolanda) :- happy (yolanda)

Output

Query 1 happy (yolanda)

true

Query 2 listens2music (mia)

false

KB3

likes (dan, Sally)

likes (Sally, dan)

likes (John, Brittney)

married (X, Y) :- likes (X, Y), likes (Y, X)

friend (X, Y) :- likes (X, Y); likes (Y, X)

Output

likes (dan, X)

X = Sally

true.

likes (Sally, Sally)

false.

KB 4:

food (burger)

food (Sandwich)

food (Pizza)

lunch (Sandwich)

dinner (Pizza)

Output

food (Pizza)

true

dinner (Sandwich)

false

Result:

Thus the PROLOG has been Executed Successfully