WEEK-1

Week : 1 **Program No : 1** **Date : 07/07/2023**

Developed by : G.Sai Hemanth Kumar

Roll No : 21331A0557

Aim : Create a list of 5 elements, pass the list to a function and compute the average of 5 numbers.

Code :

```
def Average(lst) :
    avg = sum = 0
    for i in range (0,5) :
        sum = sum + lst[i]
    avg = sum / 5
    print("Sum :",sum)
    print("Average :",avg)
lst = []
print("Enter 5 integers :")
for i in range (0,5) :
    k = int(input(""))
    lst.append(k)
print("List :",lst)
Average(lst)
```

Output :

```
PS E:\21331A0557\Sem_5\AITT LAB\Codes\Week-1> py W1Q1.py
Enter 5 integers :
1
2
3
4
5
List : [1, 2, 3, 4, 5]
Sum : 15
Average : 3.0
```

Week : 1 **Program No : 2** **Date : 07/07/2023**

Developed by : G.Sai Hemanth Kumar

Roll No : 21331A0557

Aim : Write a program that prompts a user to enter the element of a list and add the element to a list. Write a function “Maximum(list)” and “Minimum(list)” to find maximum and minimum number from the list.

Code :

```
def Maximum(lst) :
    max = 0
    for i in lst :
        if (i > max) :
            max = i
    print("Maximum Value :",max)
def Minimum(lst) :
    min = lst[0]
    for i in lst :
        if (i < min) :
            min = i
    print("Minimum value :",min)
n = int(input("Enter the size of the list : "))
lst = []
print("Enter", n, "integers :")
for i in range (0,n) :
    k = int(input(""))
    lst.append(k)
print("List :",lst)
Maximum(lst)
Minimum(lst)
```

Output :

```
PS E:\21331A0557\Sem_5\AITT LAB\Codes\Week-1> py W1Q2.py
Enter the size of the list : 5
Enter 5 integers :
1
2
3
4
5
List : [1, 2, 3, 4, 5]
Maximum Value : 5
Minimum value : 1
```

Week : 1 **Program No : 3** **Date : 07/07/2023**

Developed by : G.Sai Hemanth Kumar

Roll No : 21331A0557

Aim : Write aa function “Print_Reverse(list)” to reverse the elements of a list.

Code :

```
def Print_Reverse(lst,n) :
    lst2 = []
    for i in range(0,n) :
        k = lst[n-1-i]
        lst2.append(k)
    print("Reverse of the list :",lst2)
n = int(input("Enter the size of the list : "))
lst = []
print("Enter", n ,"integers :")
for i in range (0,n) :
    k = int(input(""))
    lst.append(k)
print("List :",lst)
Print_Reverse(lst,n)
```

Output :

```
PS E:\21331A0557\Sem_5\AITT LAB\Codes\Week-1> py W1Q3.py
Enter the size of the list : 5
Enter 5 integers :
1
2
3
4
5
List : [1, 2, 3, 4, 5]
Reverse of the list : [5, 4, 3, 2, 1]
```

Week : 1 **Program No : 4** **Date : 07/07/2023**

Developed by : G.Sai Hemanth Kumar

Roll No : 21331A0557

Aim : Write a program to return Prime Numbers from a list.

Code :

```
def Prime(lst) :
    lst3 = []
    for i in lst :
        k = i
        count = 0
        for j in range (1,k) :
            if (i%j == 0) :
                count = count + 1
        if (count == 1) :
            lst3.append(i)
    return lst3
n = int(input("Enter the size of the list : "))
lst = []
print("Enter", n ,"integers :")
for i in range (0,n) :
    k = int(input(""))
    lst.append(k)
print("List :",lst)
dummy = []
dummy = Prime(lst)
print("Prime numbers in the list :",dummy)
```

Output :

```
PS E:\21331A0557\Sem_5\AITT LAB\Codes\Week-1> py W1Q4.py
Enter the size of the list : 5
Enter 5 integers :
1
2
3
4
5
List : [1, 2, 3, 4, 5]
Prime numbers in the list : [2, 3, 5]
```

END OF THE WEEK : 1

Concepts Covered :

1. Creating a List.
2. Passing a List into a function.
3. Returning the data from a function.

GRADE : _____

Signature of Lab-Incharge : _____

WEEK-2

Week : 2 **Program No : 1** **Date : 14/07/2023**

Developed by : G.Sai Hemanth Kumar

Roll No : 21331A0557

Aim : Implement Breadth First Search algorithm using python.

Code :

```
def bfs(visited, graph, node):
  visited.append(node)
  queue.append(node)
  while queue:
    m = queue.pop(0)
    print (m, end = " ")
    for neighbour in graph[m]:
       if neighbour not in visited:
          visited.append(neighbour)
          queue.append(neighbour)
visited = []
queue = []
graph = {}
n = int(input("Enter the Number of Nodes : "))
for i in range(0,n) :
    key = input("Enter Node-{} : ".format(i+1))
    value = list(input("Enter Neighbours of Node-{} : ".format(key)))
    graph[key] = value
print("\nGraph :",graph)
k = input("\nEnter the Starting Node : ")
print("\nBFS :",end = " ")
bfs(visited, graph , k)
```

Output :

```
Enter the Number of Nodes : 7
Enter Node-1 : A
Enter Neighbours of Node-A : BDE
Enter Node-2 : B
Enter Neighbours of Node-B : ACG
Enter Node-3 : C
Enter Neighbours of Node-C : B
Enter Node-4 : D
Enter Neighbours of Node-D : A
Enter Node-5 : E
Enter Neighbours of Node-E : AF
Enter Node-6 : F
Enter Neighbours of Node-F : E
Enter Node-7 : G
Enter Neighbours of Node-G : B

Graph : {'A': ['B', 'D', 'E'], 'B': ['A', 'C', 'G'], 'C': ['B'], 'D': ['A'], 'E': ['A', 'F'], 'F': ['E'], 'G': ['B']}

Enter the Starting Node : D

BFS : D A B E C G F
```

Week : 2 **Program No : 2** **Date : 14/07/2023**

Developed by : G.Sai Hemanth Kumar

Roll No : 21331A0557

Aim : Implement Depth First Search algorithm using python.

Code :

```
def dfs(visited, graph, node):
  visited.append(node)
  queue.append(node)
  while queue:
    m = queue.pop(-1)
    print (m, end = " ")
    for neighbour in graph[m]:
       if neighbour not in visited:
          visited.append(neighbour)
          queue.append(neighbour)
visited = []
queue = []
graph = {}
n = int(input("Enter the Number of Nodes : "))
for i in range(0,n) :
    key = input("Enter Node-{} : ".format(i+1))
    value = list(input("Enter Neighbours of Node-{} : ".format(key)))
    graph[key] = value
print("\nGraph :",graph)
k = input("\nEnter the Starting Node : ")
print("\nDFS :",end = " ")
dfs(visited, graph , k)
```

Output :

```
Enter the Number of Nodes : 7
Enter Node-1 : A
Enter Neighbours of Node-A : BDE
Enter Node-2 : B
Enter Neighbours of Node-B : ACG
Enter Node-3 : C
Enter Neighbours of Node-C : B
Enter Node-4 : D
Enter Neighbours of Node-D : A
Enter Node-5 : E
Enter Neighbours of Node-E : AF
Enter Node-6 : F
Enter Neighbours of Node-F : E
Enter Node-7 : G
Enter Neighbours of Node-G : B

Graph : {'A': ['B', 'D', 'E'], 'B': ['A', 'C', 'G'], 'C': ['B'], 'D': ['A'], 'E': ['A', 'F'], 'F': ['E'], 'G': ['B']}

Enter the Starting Node : A

DFS : A E F D B G C
```

Week : 2 **Program No : 3** **Date : 14/07/2023**

Developed by : G.Sai Hemanth Kumar

Roll No : 21331A0557

Aim : Implementing Depth Limited search algorithm using python.

Code :

```
def DLS(start,goal,path,level,maxD):
  print("\nCurrent level-->",level)
  print("Goal node testing for",start)
  path.append(start)
  if start == goal:
    print("Goal test successful")
    return path
  print('Goal node testing failed')
  if level==maxD:
    return False
  print('\nExpanding the current node',start)
  for child in graph[start]:
    if DLS(child,goal,path,level+1,maxD):
      return path
    path.pop()
  return False
graph = {}
n = int(input("Enter the Number of Nodes : "))
for i in range(0,n) :
    key = input("Enter Node-{} : ".format(i+1))
    value = list(input("Enter Neighbours of Node-{} : ".format(key)))
    graph[key] = value
```

```
print("\nGraph :",graph)
for i in graph :
    start = i
    break
goal = input("\nEnter the Goal node : ")
maxD = int(input("Enter the Maximum Depth Limit : "))
print()
path = list()
res = DLS(start,goal,path,0,maxD)
if(res):
    print("Path to goal node available")
    print("Path",path)
else:
    print("No path available for the goal node in given depth limit")
```

Output :

```
Enter the Number of Nodes : 7
Enter Node-1 : S
Enter Neighbours of Node-S : AB
Enter Node-2 : A
Enter Neighbours of Node-A : CD
Enter Node-3 : B
Enter Neighbours of Node-B : IJ
Enter Node-4 : C
Enter Neighbours of Node-C :
Enter Node-5 : D
Enter Neighbours of Node-D :
Enter Node-6 : I
Enter Neighbours of Node-I :
Enter Node-7 : J
Enter Neighbours of Node-J :
```

Graph : {'S': ['A', 'B'], 'A': ['C', 'D'], 'B': ['I', 'J'], 'C': [], 'D': [], 'I': [], 'J': []}

Enter the Goal node : J

Enter the Maximum Depth Limit : 2

Current level--> 0

Goal node testing for S

Goal node testing failed

Expanding the current node S

Current level--> 1

Goal node testing for A

Goal node testing failed

Expanding the current node A

Current level--> 2

Goal node testing for C

Goal node testing failed

Current level--> 2

Goal node testing for D

Goal node testing failed

Current level--> 1

Goal node testing for B

Goal node testing failed

Expanding the current node B

Current level--> 2

Goal node testing for I

Goal node testing failed

Current level--> 2

Goal node testing for J

Goal test successful

Path to goal node available

Path ['S', 'B', 'J']

END OF THE WEEK : 2

Concepts Covered :

1. Breadth First Search.
2. Depth First Search.
3. Depth Limited Search.

GRADE : _____

Signature of Lab-Incharge : _____

WEEK-3

Week : 3 **Program No : 1** **Date : 21/07/2023**

Developed by : G.Sai Hemanth Kumar

Roll No : 21331A0557

Aim : Implement Best First Search Algorithm using Python.

Code :

```
import heapq #For priority queue
def best_first_search(graph, start, goal, heuristic):
    que = [(heuristic[start], start)]
    came_from = {}
    came_from[start] = None
    while que:
        _, peak_node = heapq.heappop(que)
        if peak_node == goal:
            break
        for neighbor in graph[peak_node]:
            if neighbor not in came_from:
                heapq.heappush(que, (heuristic[neighbor], neighbor))
                came_from[neighbor] = peak_node
    return came_from
graph = {}
hc = {}
n = int(input("Enter the Number of Nodes : "))
print()
for i in range(0,n) :
    key = input("Enter Node-{} : ".format(i+1))
    value = list(input("Enter Neighbours of Node-{} : ".format(key)))
    graph[key] = value
print("\nGraph :",graph,"\n\nEnter the Heuristic costs of nodes :\n")
for i in graph :
    HC = int(input("HC of {} : ".format(i)))
```

```
    hc[i] = HC
print("\nHeuristic Costs of nodes :",hc)
g = input("\nEnter the goal node : ")
for i in graph :
    s = i
    break
came_from = best_first_search(graph, s, g, hc)
node = g
path = [node]
while (node != s) :
    node = came_from[node]
    path.append(node)
path.reverse()
print("\nBFS path from",s,"to",g,":",path)
```

Output :

Enter the Number of Nodes : 7

Enter Node-1 : S

Enter Neighbours of Node-S : ABC

Enter Node-2 : A

Enter Neighbours of Node-A : DEG

Enter Node-3 : B

Enter Neighbours of Node-B : G

Enter Node-4 : C

Enter Neighbours of Node-C : G

Enter Node-5 : D

Enter Neighbours of Node-D :

Enter Node-6 : E

Enter Neighbours of Node-E :

Enter Node-7 : G

Enter Neighbours of Node-G :

Graph : {'S': ['A', 'B', 'C'], 'A': ['D', 'E', 'G'], 'B': ['G'], 'C': ['G'], 'D': [], 'E': [], 'G': []}

Enter the Heuristic costs of nodes :

HC of S : 8

HC of A : 8

HC of B : 4

HC of C : 3

HC of D : -1

HC of E : -1

HC of G : 0

Heuristic Costs of nodes : {'S': 8, 'A': 8, 'B': 4, 'C': 3, 'D': -1, 'E': -1, 'G': 0}

Enter the goal node : G

BFS path from S to G : ['S', 'C', 'G']

Week : 3 **Program No : 2** **Date : 21/07/2023**

Developed by : G.Sai Hemanth Kumar

Roll No : 21331A0557

Aim : perform operations with numpy package in python. (type, len, ndim, shape, reshape, arrange, itemsize, dtype).

Code :

```
import numpy as np
arr = np.array([1,2,3,4,5,6])
print("Type of array is :",type(arr))
length = len(arr)
print("Length of the array is :",length)
print("Dimension of the array is :",arr.ndim)
print("Shape of the array is :",arr.shape)
reshaped = arr.reshape(2,3)
print("Reshaped array is :\n",reshaped)
print("Itemsize of array is :",arr.itemsize)
print("Datatype of array is :",arr.dtype)
print("Example of arange is :",end = " ")
np.arange(1,6,2)
```

Output :

```
Type of array is : <class 'numpy.ndarray'>
Length of the array is : 6
Dimenstion of the array is : 1
Shape of the array is : (6,)
Reshaped array is :
 [[1 2 3]
 [4 5 6]]
Itemsize of array is : 8
Datatype of array is : int64
Example of arange is : array([1, 3, 5])
```

Week : 3 **Program No : 3** **Date : 21/07/2023**

Developed by : G.Sai Hemanth Kumar

Roll No : 21331A0557

Aim : Write a program in python to initialise numpy arrays from nested list.

Code :

```
import numpy as np
a = [1,2,3,4,5]
b = [6,7,8,9,10]
c = [3,4,5,6,7]
array_example = np.array([a,b,c])
print(array_example)
```

Output :

```
[[ 1  2  3  4  5]
 [ 6  7  8  9 10]
 [ 3  4  5  6  7]]
```

Week : 3 **Program No : 4** **Date : 21/07/2023**

Developed by : G.Sai Hemanth Kumar

Roll No : 21331A0557

Aim : Write a program in python to perform numpy Arithmetic Operations. (add, subtract, multiple, divide, dot, sum, axis).

Code :

```
import numpy as np
a = np.array([[1, 2], [3, 4]])
b = np.array([[4, 1], [2, 2]])
sum = np.add(a,b)
print("a + b :\n",sum)
sub = np.subtract(a, b)
print("a - b :\n",sub)
mul = np.multiply(a, b)
print("a * b :\n",mul)
div = np.divide(a, b)
print("a / b :\n",div)
sum_a = np.sum(a)
print("Sum of elements of a :",sum_a)
print("Matrix multiplication :\n",np.dot(a, b))
print("Sum of elements in columns of a :", a.sum(axis = 0))
print("Sum of elements in rows of b :",b.sum(axis = 1))
```

Output :

```
a + b :
 [[5 3]
 [5 6]]
a - b :
 [[-3  1]
 [ 1  2]]
a * b :
 [[4 2]
 [6 8]]
a / b :
 [[0.25 2.  ]
 [1.5  2.  ]]
Sum of a : 10
Matrix multiplication :
 [[ 8  5]
 [20 11]]
Sum of elements in columns of a : [4 6]
Sum of elements in rows of b : [5 4]
```

END OF THE WEEK : 3

Concepts Covered :

1. Best First Search Algorithm.
2. Basic operations with numpy package.
3. Initialising numpy arrays from nested list.
4. numpy Arithmetic Operations.

GRADE : _____

Signature of Lab-Incharge : _____

WEEK-4

Week : 4 **Program No : 1** **Date : 28/07/2023**

Developed by : G.Sai Hemanth Kumar

Roll No : 21331A0557

Aim : Create 4*2 integer array and print its attributes

-The shape of an array

-Array dimensions.

-The Length of each element of the array in bytes. [The element must be a type of unsigned int16]

Code :

```
import numpy as np
a1 = [1,2]
a2 = [3,4]
a3 = [5,6]
a4 = [7,8]
b = np.array((a1,a2,a3,a4),dtype = np.int16)
print("array :\n",b)
print("Shape of the array :",b.shape)
print("Dimensions of the array :",b.ndim)
print("Length of each element of the array in bytes :",b.itemsize)
```

Output :

```
array :
 [[1 2]
 [3 4]
 [5 6]
 [7 8]]
Shape of the array : (4, 2)
Dimensions of the array : 2
Length of each element of the array in bytes : 2
```

Week : 4 **Program No : 2** **Date : 28/07/2023**

Developed by : G.Sai Hemanth Kumar

Roll No : 21331A0557

Aim : Create a 5×2 integer array from a range between 100 to 200 such that the difference between each element is 10.

Code :

```
import numpy as np
lst = []
for i in range(100,200,10) :
  lst.append(i)
a = np.array(lst)
print("array :",a)
print("Shape of array :",a.shape)
a = a.reshape(5,2)
print("After reshaping to (5,2), the array is :\n",a)
```

Output :

```
array : [100 110 120 130 140 150 160 170 180 190]
Shape of array : (10,)
After reshaping to (5,2), the array is :
 [[100 110]
 [120 130]
 [140 150]
 [160 170]
 [180 190]]
```

Week : 4 **Program No : 3** **Date : 28/07/2023**

Developed by : G.Sai Hemanth Kumar

Roll No : 21331A0557

Aim : Create a 2D array by using numpy and return array of items by taking the third column from all rows.

Code :

```
import numpy as np
a = [1,2,3]
b = [4,5,6]
dummy = np.array([a,b])
print("array :\n",dummy)
print("Dimension of array :",dummy.ndim)
print("Column at 3rd index :",dummy[:, 2])
```

Output :

```
array :
 [[1 2 3]
 [4 5 6]]
Dimention of array : 2
Column at 3rd index : [3 6]
```

Week : 4 **Program No : 4** **Date : 28/07/2023**

Developed by : G.Sai Hemanth Kumar

Roll No : 21331A0557

Aim : Create 5×4 matrix and return array of odd rows and even columns from the given numpy array.

Code :

```
import numpy as np
a = [1,2,3,4]
b = [5,6,7,8]
c = [9,10,11,12]
d = [13,14,15,16]
e = [17,18,19,20]
dummy = np.array([a,b,c,d,e])
print("array :\n",dummy)
print("Shape of array :",dummy.shape)
a,b = dummy.shape
print(“\nOdd Rows :\n”)
for i in range (0,a) :
  if (i%2 == 0) :
    print("Row at {}th index : {}".format(i+1,dummy[i]))
print(“\nEven Columns :\n”)
for i in range (0,b) :
  if (i%2 != 0) :
    print("Column at {}th index : {}".format(i+1,dummy[:,i]))
```

Output :

```
array :
 [[ 1  2  3  4]
 [ 5  6  7  8]
 [ 9 10 11 12]
 [13 14 15 16]
 [17 18 19 20]]
Shape of array : (5, 4)

Odd Rows :

Row at 1th index : [1 2 3 4]
Row at 3th index : [ 9 10 11 12]
Row at 5th index : [17 18 19 20]

Even Columns :

Column at 2th index : [ 2  6 10 14 18]
Column at 4th index : [ 4  8 12 16 20]
```

Week : 4 **Program No : 5** **Date : 28/07/2023**

Developed by : G.Sai Hemanth Kumar

Roll No : 21331A0557

Aim : Addition of two 2D arrays. Modify the result array by calculating the square of each element.

Code :

```
import numpy as np
a = [10,20,36]
b = [25,63,89]
a1 = np.array([a,b])
a2 = np.array([b,a])
print("array-1 :\n",a1)
print("array-2 :\n",a2)
print("Dimension of array-1 :",a1.ndim)
print("Dimension of array-2 :",a2.ndim)
a3 = np.array((a1+a2))
print("\nAddition of two arrays :\n",a3)
print("\nSquare of each elements :\n",a3**2)
```

Output :

```
array-1 :
 [[10 20 36]
 [25 63 89]]
array-2 :
 [[25 63 89]
 [10 20 36]]
Dimension of array-1 : 2
Dimension of array-2 : 2

Addition of two arrays :
 [[ 35  83 125]
 [ 35  83 125]]

Square of each elements :
 [[ 1225  6889 15625]
 [ 1225  6889 15625]]
```

Week : 4 **Program No : 6** **Date : 28/07/2023**

Developed by : G.Sai Hemanth Kumar

Roll No : 21331A0557

Aim : Split the array into four equal-sized sub-arrays. Note: Create an 8×3 integer array from a range between 10 to 34 such that the difference between each element is 1 and then Split the array into four equal-sized sub-arrays.

Code :

```
import numpy as np
lst = []
for i in range (10,34) :
  lst.append(i)
dummy = np.array([lst])
dummy = dummy.reshape(8,3)
print("array :\n",dummy)
print("Shape of array :",dummy.shape)
equalparts = np.split(dummy,4)
print("\nThe array is divided into 4 equal Sub-arrays and they are :")
count = 0
for parts in equalparts :
  count = count+1
  print("\nSub-array{} :\n{}".format(count,parts))
```

Output :

```
array :
 [[10 11 12]
 [13 14 15]
 [16 17 18]
 [19 20 21]
 [22 23 24]
 [25 26 27]
 [28 29 30]
 [31 32 33]]
Shape of array : (8, 3)

The array is divided into 4 equal parts and they are :

Sub-array1 :
[[10 11 12]
 [13 14 15]]

Sub-array2 :
[[16 17 18]
 [19 20 21]]

Sub-array3 :
[[22 23 24]
 [25 26 27]]

Sub-array4 :
[[28 29 30]
 [31 32 33]]
```

Week : 4 **Program No : 7** **Date : 28/07/2023**

Developed by : G.Sai Hemanth Kumar

Roll No : 21331A0557

Aim : Create a 3×3 array and do the following sorting.

Sort array by the second row.

Sort the array by the second column.

Code :

```
import numpy as np
a = [9,8,6]
b = [121,505,340]
c = [15,10,12]
dummy = np.array([a,b,c])
print("array :\n",dummy)
print("Shape of array :",dummy.shape)
dummy[1].sort() #row
print("\narray after sorting the row-2  :\n",dummy)
dummy[:,1].sort() #column
print("\narray after sorting the column-2 :\n",dummy)
```

Output :

```
array :
 [[  9   8   6]
 [121 505 340]
 [ 15  10  12]]
Shape of array : (3, 3)

array after sorting the row-2:
 [[  9   8   6]
 [121 340 505]
 [ 15  10  12]]

array after sorting the column-2 :
 [[  9   8   6]
 [121  10 505]
 [ 15 340  12]]
```

Week : 4 **Program No : 8** **Date : 28/07/2023**

Developed by : G.Sai Hemanth Kumar

Roll No : 21331A0557

Aim : Create a 2D array and print max from axis 0 and min from axis 1.

Code :

```
import numpy as np
a = [10,26,95,2]
b = [12,3,62,1]
dummy = np.array([a,b])
print("array :\n",dummy)
print("Dimensions of array :",dummy.ndim)
print("\nMax of axis-0 of array :",dummy.max(axis=0))
print("Min of axis-1 of array :",dummy.min(axis=1))
```

Output :

```
array :
 [[10 26 95  2]
 [12  3 62  1]]
Dimensions of array : 2

Max of axis-0 of array : [12 26 95  2]
Min of axis-1 of array : [2 1]
```

Week : 4 **Program No : 9** **Date : 28/07/2023**

Developed by : G.Sai Hemanth Kumar

Roll No : 21331A0557

Aim : Create a 3×3 array and delete the second column from a given array and insert the following new column in its place.

Code :

```
import numpy as np
a = [1,2,3]
b = [4,5,6]
c = [7,8,9]
d = [10,11,12]
dummy = np.array([a,b,c])
print("array :\n",dummy)
print("Shape of array :",dummy.shape)
dummy = np.delete(dummy, 1, 1)
print("\nAfter deleting the 2nd column :\n",dummy)
dummy1 = np.array(d)
print("\nThe elements need to be added in the array at 2nd column :", dummy1)
dummy = np.insert(dummy,1,dummy1,1)
print("\nAfter insert the 2nd column in the array :\n",dummy)
```

Output :

```
array :
 [[1 2 3]
 [4 5 6]
 [7 8 9]]
Shape of array : (3, 3)

After deleting the 2nd column :
 [[1 3]
 [4 6]
 [7 9]]

The elements need to be added in the array at 2nd column : [10 11 12]

After insert the 2nd column in the array :
 [[ 1 10  3]
 [ 4 11  6]
 [ 7 12  9]]
```

END OF THE WEEK : 4

Concepts Covered :

1. Displaying the attributes of an array(shape, dimension, length of each element) .
2. Creating an array with certain input(dimension, data with range and jump/skip).
3. Displaying the data from a particular column in an array.
4. Printing odd rows & even columns from an array.
5. Addition of two arrays and calculating the square of elements and display as output.
6. Splitting an array into equal parts.
7. Sort an array by particular row & column.
8. Usage of axis and min(), max() using axis in array.
9. Deleting and adding a particular column in an array.

GRADE : _____

Signature of Lab-Incharge : _____

WEEK-5

Week : 5 **Program No : 1** **Date : 04/08/2023**

Developed by : G.Sai Hemanth Kumar

Roll No : 21331A0557

Aim : Using pandas library, create the following :

-> Empty Series.

-> Series using Arrays.

-> Series using Lists.

-> Series using Dictionaries.

Code :

```
import pandas as pd
import numpy as np
ES = pd.Series()
print("Empty Series :",ES)
arry = np.array([10,20,30,40,50])
ind = []
n = len(arry)
for i in range (1,n+1) :
  ind.append(i)
SUA = pd.Series(arry, index = ind)
print("\nArray :",arry)
print("Series using Array :")
print(SUA)
lst = [10,20,30]
id = []
m = len(lst)
for i in range (1,m+1) :
  id.append(i)
SUL = pd.Series(lst, index = id)
```

```
print("\nList :",lst)

print("Series using List :")

print(SUL)

dic = {"A" : 1, "B" : 2, "C" : 3}

SUD = pd.Series(dic)

print("\nDictionary :",dic)

print("Series using Dictionary :")

print(SUD)
```

Output :

```
Empty Series : Series([], dtype: float64)

Array : [10 20 30 40 50]
Series using Array :
1    10
2    20
3    30
4    40
5    50
dtype: int64

List : [10, 20, 30]
Series using List :
1    10
2    20
3    30
dtype: int64

Dictionary : {'A': 1, 'B': 2, 'C': 3}
Series using Dictionary :
A    1
B    2
C    3
dtype: int64
<ipython-input-79-897f55a18557>:3: FutureWarning: The default dtype for empty Series will be 'object' instead of 'float64' in a future version. Specify a dtype explicitly to silence this warning.
  ES = pd.Series()
```

Week : 5 **Program No : 2** **Date : 04/08/2023**

Developed by : G.Sai Hemanth Kumar

Roll No : 21331A0557

Aim : Create the following using pandas library :

-> Create dataframe using Dictionary.

-> Create dataframe using List of Tuples.

-> Load the data from .xlsx file.

-> Load the data from .csv file.

Code :

```
import pandas as pd
dic = {"Name" : ["Varshit","Hemanth","Amani"], "Age" : [19,20,18], "Gender" : ["Male","Male","Female"]}
DUD = pd.DataFrame(dic, index = [1,2,3])
print("Dictionary :",dic)
print("Created dataframe using Dictionary :")
print(DUD)
lst = [("Varshit",19,"Male"),("Hemanth",20,"Male"),("Amani",18,"Female")]
length = len(lst)
lists = []
for i in range (1,length+1) :
  lists.append(i)
DUT = pd.DataFrame(lst, columns = ['Name','Age','Gender'], index = lists)
print("\nList of tuples :",lst)
print("Created dataframe using List of tuples :")
print(DUT)
data_from_excel = pd.read_excel("/Student_Marks.xlsx")
DXF = pd.DataFrame(data_from_excel)
print("\nData in .xlsx file :\n",DXF)
data_from_csv = pd.read_csv("/Student_Marks_Manipulation.csv")
DCF = pd.DataFrame(data_from_csv)
```

```
print("\nData in .csv file :\n",DCF)
```

Output :

```
Dictionary : {'Name': ['Varshit', 'Hemanth', 'Amani'], 'Age': [19, 20, 18], 'Gender': ['Male', 'Male', 'Female']}
Created dataframe using Dictionary :
      Name  Age  Gender
1  Varshit   19    Male
2  Hemanth   20    Male
3    Amani   18  Female

List of tuples : [('Varshit', 19, 'Male'), ('Hemanth', 20, 'Male'), ('Amani', 18, 'Female')]
Created dataframe using List of tuples :
      Name  Age  Gender
1  Varshit   19    Male
2  Hemanth   20    Male
3    Amani   18  Female

Data in .xlsx file :
       Roll No                                Name  WT  AITT  JAVA  DS
0  18331A0501                         Aalla SaiPavan  21    43    34  54
1  18331A0502                     HEMANTH KUMAR AKASAPU  22    33    34  54
2  18331A0504              Vishnu Vardhan Varma Alluri  23    55    34  54
3  18331A0508                        Arisetty Pavitra  24    21    34  54
4  18331A0511                     Bankupalli Srinidhi  25    67    34  54
5  18331A0512                Nikhita V Naga S L Batchu  26    89    34  54
6  18331A0513                        Venkatesh Bevara  27    34    34  54
7  18331A0514  Bhogapurapu Giridhar Gouri Sri Prasad  28    23    23  54
8  18331A0517                     YASHWANTH BOGGARAPU  29    23    23  54
9  18331A0518                           Adarsh Bongi  30    23    23  78

Data in .csv file :
       Roll No                                Name  Marks
0  18331A0501                         Aalla SaiPavan    152
1  18331A0502                     HEMANTH KUMAR AKASAPU    143
2  18331A0504              Vishnu Vardhan Varma Alluri    166
3  18331A0508                        Arisetty Pavitra    133
4  18331A0511                     Bankupalli Srinidhi    180
5  18331A0512                Nikhita V Naga S L Batchu    203
6  18331A0513                        Venkatesh Bevara    149
7  18331A0514  Bhogapurapu Giridhar Gouri Sri Prasad    128
8  18331A0517                     YASHWANTH BOGGARAPU    129
9  18331A0518                           Adarsh Bongi    154
```

END OF THE WEEK : 5

Concepts Covered :

1. Empty Series.
2. Series using arrays.
3. Series using Lists.
4. Series using Dictionaries.
5. Create dataframe using Dictionary.
6. Create dataframe using List of Tuples.
7. Load the data from .xlsx file.
8. Load the data from .csv file.

GRADE : _____

Signature of Lab-Incharge : _____

WEEK-6

Week : 6 **Program No : 1** **Date : 11/08/2023**

Developed by : G.Sai Hemanth Kumar

Roll No : 21331A0557

Aim : Perform the following DataFrames functions in python(Basic Information and Exploration).

-> df.head(n)

-> df.tail(n)

-> df.shape

-> df.info()

-> df.describe()

Code :

```
import pandas as pd
dic = {"Name" : ["Varshit","Hemanth","Amani"], "Age" : [19,20,18], "Gender" : ["Male","Male","Female"]}
DUD = pd.DataFrame(dic, index = [1,2,3])
print("Dictionary :",dic)
print("\nCreated DataFrame using Dictionary :")
print(DUD)
print("\nThe First 2 rows of the DataFrame is :")
print(DUD.head(2))
print("\nThe Last 2 rows of the DataFrame is :")
print(DUD.tail(2))
print("\nThe Shape of the DataFrame is :", DUD.shape)
print("\nThe Information about the DataFrame's Datatypes and Memory usage is :\n")
print(DUD.info())
print("\nThe summary statistics of numerical columns is :")
print(DUD.describe())
```

Output :

```
Dictionary : {'Name': ['Varshit', 'Hemanth', 'Amani'], 'Age': [19, 20, 18], 'Gender': ['Male', 'Male', 'Female']}

Created DataFrame using Dictionary :
      Name  Age  Gender
1  Varshit   19    Male
2  Hemanth   20    Male
3    Amani   18  Female

The First 2 rows of the DataFrame is :
      Name  Age Gender
1  Varshit   19   Male
2  Hemanth   20   Male

The Last 2 rows of the DataFrame is :
      Name  Age  Gender
2  Hemanth   20    Male
3    Amani   18  Female

The Shape of the DataFrame is : (3, 3)

The Information about the DataFrame's Datatypes and Memory usage is :

<class 'pandas.core.frame.DataFrame'>
Int64Index: 3 entries, 1 to 3
Data columns (total 3 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   Name    3 non-null      object
 1   Age     3 non-null      int64
 2   Gender  3 non-null      object
dtypes: int64(1), object(2)
memory usage: 96.0+ bytes
None

The summary statistics of numerical columns is :
        Age
count   3.0
mean   19.0
std     1.0
min    18.0
25%    18.5
50%    19.0
75%    19.5
max    20.0
```

Week : 6 **Program No : 2** **Date : 11/08/2023**

Developed by : G.Sai Hemanth Kumar

Roll No : 21331A0557

Aim : Perform the following DataFrames functions in python(Indexing and Selection).

-> df[column]

-> df[[col1, col2]]

-> df.loc[row_lable, col_lable]

-> df.iloc[row_index, col_index]

-> df.query(condition)

Code :

```
import pandas as pd
dic = {"Name" : ["Varshit","Hemanth","Amani"], "Age" : [19,20,18], "Gender" : ["Male","Male","Female"]}
DUD = pd.DataFrame(dic, index = [1,2,3])
print("Dictionary :",dic)
print("\nCreated DataFrame using Dictionary :")
print(DUD)
print("\nThe Name (Single)column as a series is :")
print(DUD["Name"])
print("\nThe Name,Age (Multiple)column as a new DataFrame is :")
print(DUD[["Name","Age"]])
print("\nAccesses rows and columns by labels as a DataFrame is :")
DUD.set_index("Name", inplace = True)
print(DUD.loc[["Hemanth","Amani"]])
print("\nThe value at index [1][0] is :",DUD.iloc[1,0])
print("\nThe records where Age is equal to 20 is :")
print(DUD.query('Age == 20'))
```

Output :

```
Dictionary : {'Name': ['Varshit', 'Hemanth', 'Amani'], 'Age': [19, 20, 18], 'Gender': ['Male', 'Male', 'Female']}

Created DataFrame using Dictionary :
      Name  Age  Gender
1  Varshit   19    Male
2  Hemanth   20    Male
3    Amani   18  Female

The Name (Single)column as a series is :
1    Varshit
2    Hemanth
3      Amani
Name: Name, dtype: object

The Name,Age (Multiple)column as a new DataFrame is :
      Name  Age
1  Varshit   19
2  Hemanth   20
3    Amani   18

Accesses rows and columns by labels as a DataFrame is :
         Age  Gender
Name
Hemanth   20    Male
Amani     18  Female

The value at index [1][0] is : 20

The records where Age is equal to 20 is :
         Age Gender
Name
Hemanth   20   Male
```

Week : 6 **Program No : 3** **Date : 11/08/2023**

Developed by : G.Sai Hemanth Kumar

Roll No : 21331A0557

Aim : Perform the following DataFrames functions in python(Filtering and Cleaning).

-> df.drop(columns=[‘col1’, ‘col2’])

-> df.dropna()

-> df.fillna(value)

-> df.drop_duplicates()

-> df.rename(columns={‘old_name’: ‘new_name’})

Code :

```
import pandas as pd
import numpy as np
dic = {"Name" : [np.nan,"Hemanth","Amani","Hemanth"], "Age" : [np.nan,np.nan,18,np.nan], "Gender" : [np.nan,"Male","Female","Male"]}
DUD = pd.DataFrame(dic, index = [1,2,3,4])
print("Dictionary :",dic)
print("\nCreated DataFrame using Dictionary :")
print(DUD)
print("\nAfter removing the Gender column, the DataFrame becomes :")
print(DUD.drop(columns = ["Gender"]))
print("\nDrops rows with missing values, the DataFrame becomes :")
print(DUD.dropna())
print("\nReplace missing values with 10, the DataFrame becomes :")
print(DUD.fillna(10))
print("\nDrops rows with duplicate values, the DataFrame becomes :")
print(DUD.drop_duplicates())
print("\nAfter renaming the column Age to Ages, the DataFrame becomes :")
```

```
print(DUD.rename(columns={"Age":"Ages"}))
```

Output :

```
Dictionary : {'Name': [nan, 'Hemanth', 'Amani', 'Hemanth'], 'Age': [nan, nan, 18, nan], 'Gender': [nan, 'Male', 'Female', 'Male']}

Created DataFrame using Dictionary :
      Name   Age  Gender
1      NaN   NaN     NaN
2  Hemanth   NaN    Male
3    Amani  18.0  Female
4  Hemanth   NaN    Male

After removing the Gender column, the DataFrame becomes :
      Name   Age
1      NaN   NaN
2  Hemanth   NaN
3    Amani  18.0
4  Hemanth   NaN

Drops rows with missing values, the DataFrame becomes :
    Name   Age  Gender
3  Amani  18.0  Female

Replace missing values with 10, the DataFrame becomes :
      Name   Age  Gender
1       10  10.0      10
2  Hemanth  10.0    Male
3    Amani  18.0  Female
4  Hemanth  10.0    Male

Drops rows with duplicate values, the DataFrame becomes :
      Name   Age  Gender
1      NaN   NaN     NaN
2  Hemanth   NaN    Male
3    Amani  18.0  Female

After renaming the column Age to Ages, the DataFrame becomes :
      Name  Ages  Gender
1      NaN   NaN     NaN
2  Hemanth   NaN    Male
3    Amani  18.0  Female
4  Hemanth   NaN    Male
```

Week : 6 **Program No : 4** **Date : 11/08/2023**

Developed by : G.Sai Hemanth Kumar

Roll No : 21331A0557

Aim : Perform the following DataFrames functions in python(Aggregation and Grouping).

-> df.groupby('column').agg(func)

-> df.grroupby('column').mean()

-> df.groupby('column').sum()

Code :

```
import pandas as pd
Dict = {'Category': ['A', 'B', 'B', 'A', 'B', 'A'], 'Value': [1, 2, 3, 4, 5, 6]}
DUD = pd.DataFrame(Dict, index = [1,2,3,4,5,6])
print("The data in the Dictionary is :",Dict)
print("\nCreated DataFrame using Dictionary :")
print(DUD)
print("\nThe aggregate function ")
tot = DUD.groupby("Category").aggregate(["sum","min","max"])
print(tot)
print("\nThe mean of numeric columns for each group is :")
mean = DUD.groupby("Category").mean()
print(mean)
print("\nThe sum of numeric columns for each group is :")
total_sum = DUD.groupby("Category").sum()
print(total_sum)
```

Output :

```
The data in the Dictionary is : {'Category': ['A', 'B', 'B', 'A', 'B', 'A'], 'Value': [1, 2, 3, 4, 5, 6]}

Created DataFrame using Dictionary :
  Category  Value
1        A      1
2        B      2
3        B      3
4        A      4
5        B      5
6        A      6

The aggregate function
         Value
           sum min max
Category
A           11   1   6
B           10   2   5

The mean of numeric columns for each group is :
             Value
Category
A         3.666667
B         3.333333

The sum of numeric columns for each group is :
          Value
Category
A            11
B            10
```

END OF THE WEEK : 6

Concepts Covered :

1. Basic Information and Exploration functions of DataFrame.
2. Indexing and Selection functions of DataFrame.
3. Filtering and Cleaning functions of DataFrame.
4. Aggregation and Grouping functions of DataFrame.

GRADE : _____

Signature of Lab-Incharge : _____

WEEK-7

Week : 7 **Program No : 1** **Date : 18/08/2023**

Developed by : G.Sai Hemanth Kumar

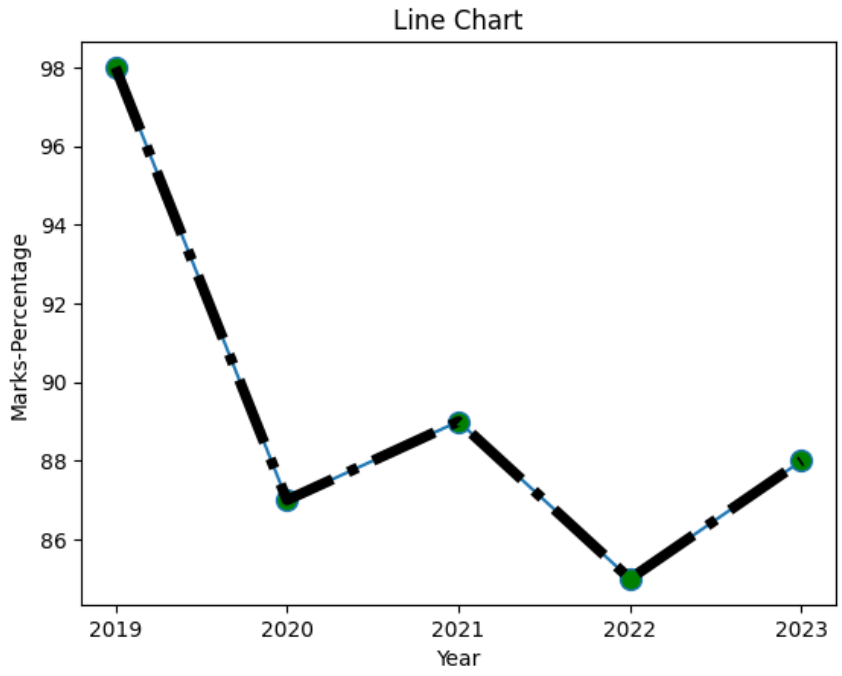
Roll No : 21331A0557

Aim : Plot a Line Chart using matplotlib.

Code :

```
import matplotlib.pyplot as plt
x = ["2019","2020","2021","2022","2023"]
y = [98,87,89,85,88]
plt.xlabel("Year")
plt.ylabel("Marks-Percentage")
plt.title("Line Chart")
plt.plot(x,y,marker="o",ms="10",mfc="g")
plt.plot(x,y,ls="dashdot",lw="5",c="black")
plt.show()
```

Output :



Week : 7 **Program No : 2** **Date : 18/08/2023**

Developed by : G.Sai Hemanth Kumar

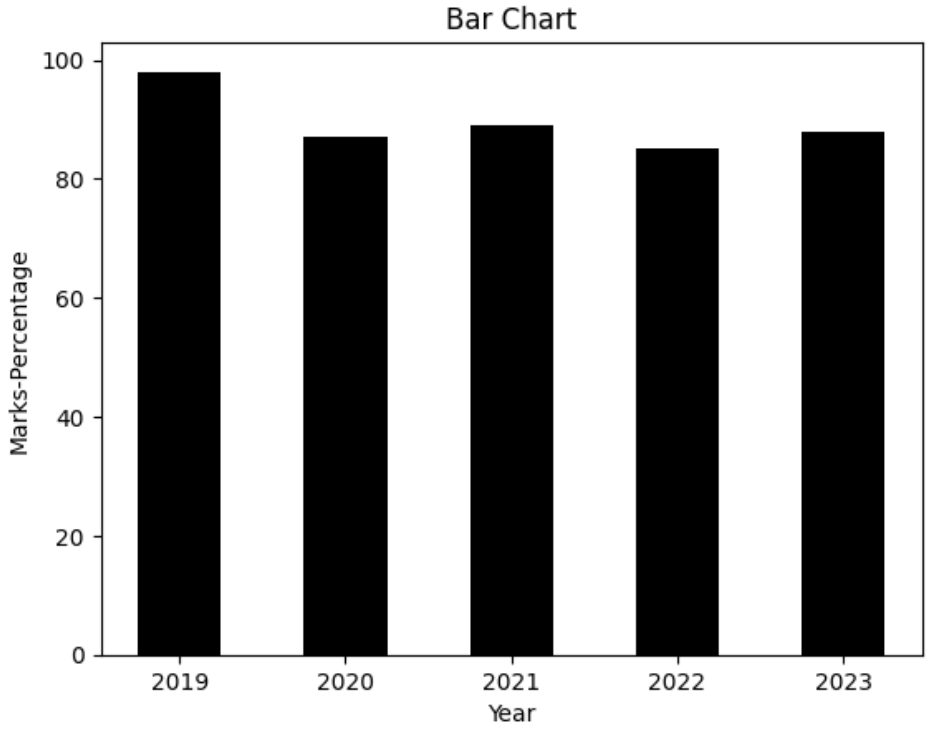
Roll No : 21331A0557

Aim : Plot a Bar Chart using matplotlib.

Code :

```
import matplotlib.pyplot as plt
x = ["2019","2020","2021","2022","2023"]
y = [98,87,89,85,88]
plt.bar(x,y,color = "black",width = 0.5)
plt.xlabel("Year")
plt.ylabel("Marks-Percentage")
plt.title("Bar Chart")
plt.show()
```

Output :



Week : 7 **Program No : 3** **Date : 18/08/2023**

Developed by : G.Sai Hemanth Kumar

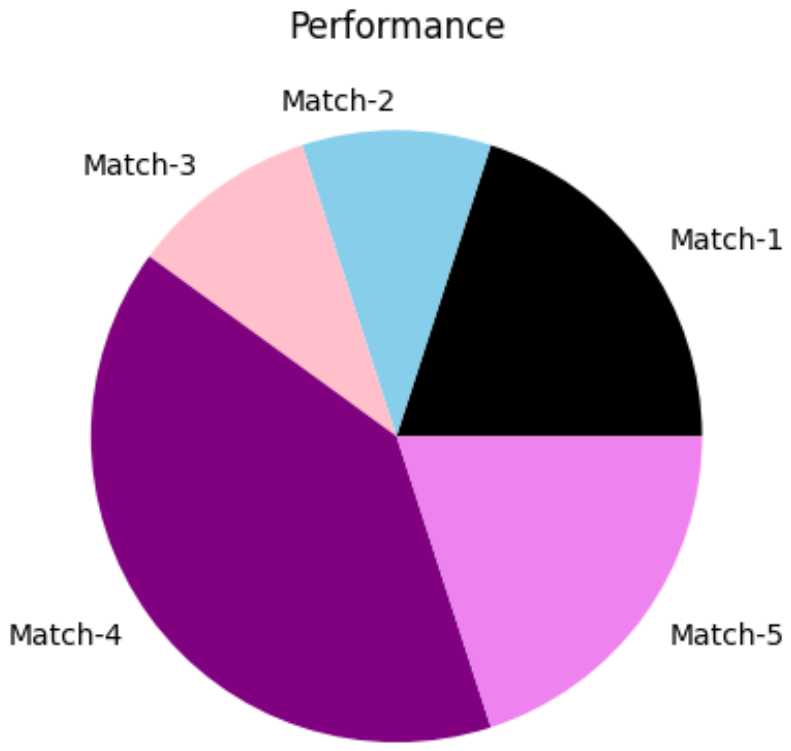
Roll No : 21331A0557

Aim : Plot a Pie Chart using matplotlib.

Code :

```
import matplotlib.pyplot as plt
y = ["Match-1","Match-2","Match-3","Match-4","Match-5"]
z = [20,10,10,40,20]
c = ["Black","Skyblue","Pink","Purple","Violet"]
plt.pie(z, labels = y, colors = c)
plt.title("Performance")
plt.show()
```

Output :



Week : 7 **Program No : 4** **Date : 18/08/2023**

Developed by : G.Sai Hemanth Kumar

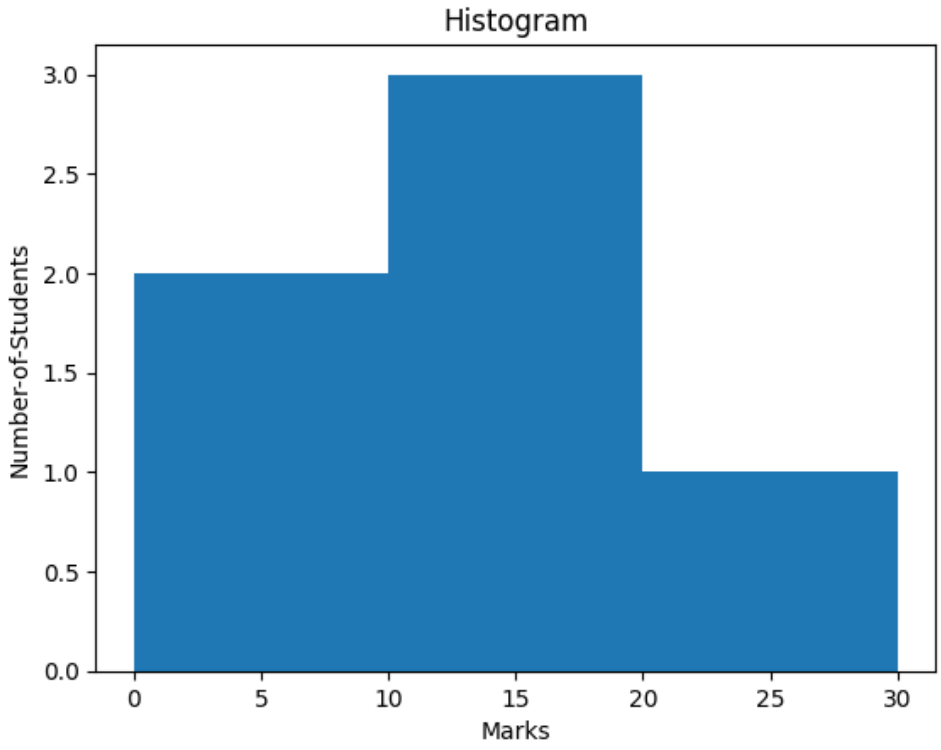
Roll No : 21331A0557

Aim : Plot a Histogram using matplotlib.

Code :

```
import matplotlib.pyplot as plt
x = [0,10,20,30]
y = [1,3,30,19,14,12]
plt.hist(y,x)
plt.ylabel("Number-of-Students")
plt.xlabel("Marks")
plt.title("Histogram")
plt.show()
```

Output :



Week : 7 **Program No : 5** **Date : 18/08/2023**

Developed by : G.Sai Hemanth Kumar

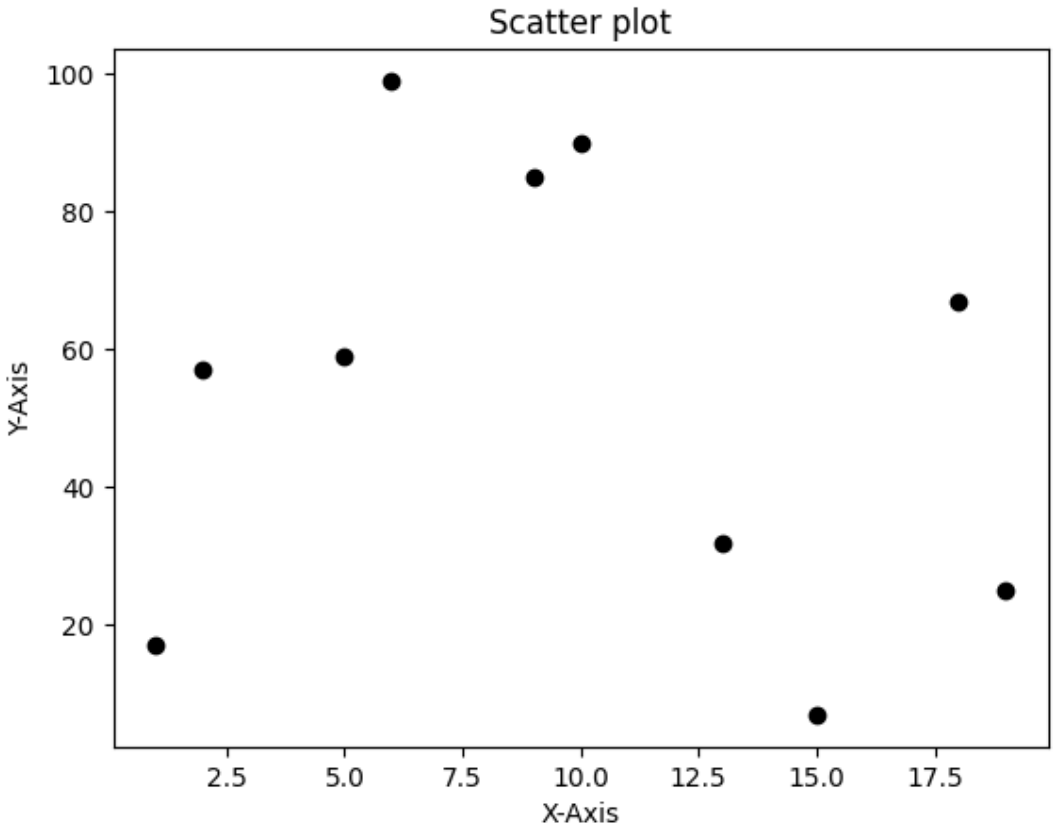
Roll No : 21331A0557

Aim : Plot a Scatter plot using matplotlib.

Code :

```
import matplotlib.pyplot as plt
x = [15,6,9,18,2,13,19,1,5,10]
y = [7,99,85,67,57,32,25,17,59,90]
plt.scatter(x,y,color = "black")
plt.xlabel("X-Axis")
plt.ylabel("Y-Axis)
plt.title("Scatter plot")
plt.show()
```

Output :



Week : 7 **Program No : 6** **Date : 18/08/2023**

Developed by : G.Sai Hemanth Kumar

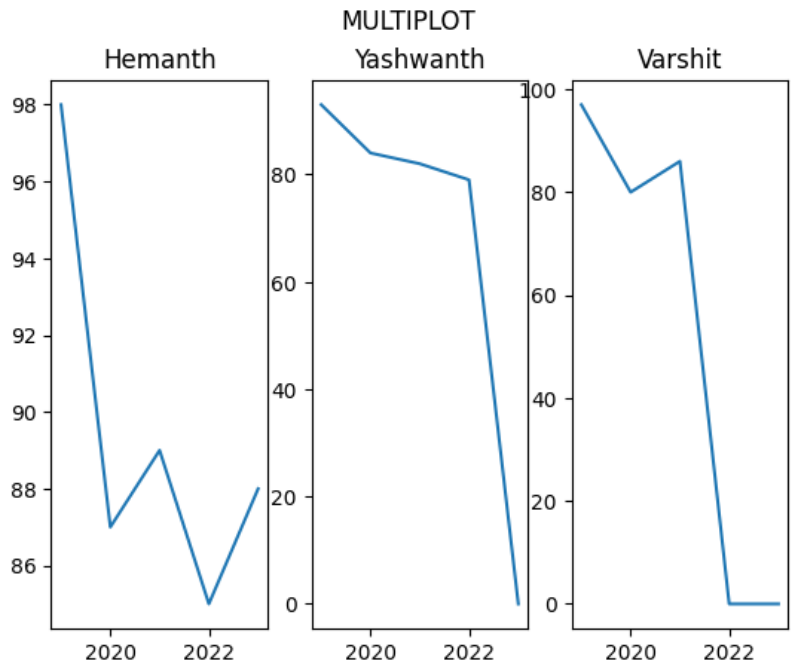
Roll No : 21331A0557

Aim : Plot a Multi plot using matplotlib.

Code :

```
import matplotlib.pyplot as plt
x=[2019,2020,2021,2022,2023]
y1=[98,87,89,85,88]
y2=[93,84,82,79,0]
y3 = [97,80,86,0,0]
plt.suptitle("MULTIPLOT")
plt.subplot(1,3,1)
plt.plot(x,y1)
plt.title("Hemanth")
plt.subplot(1,3,2)
plt.plot(x,y2)
plt.title("Yashwanth")
plt.subplot(1,3,3)
plt.plot(x,y3)
plt.title("Varshit")
plt.show()
```

Output :


MULTIPLOT
Hemanth
Yashwanth
Varshit
98
96
94
92
90
88
86
80
60
40
20
0
100
80
60
40
20
0
2020
2022
2020
2022
2020
2022

END OF THE WEEK : 7

Concepts Covered :

1. Line chart using matplotlib.pyplot.
2. Bar chart using matplotlib.pyplot.
3. Pie chart using matplotlib.pyplot.
4. Histogram using matplotlib.pyplot.
5. Scatter plot using matplotlib.pyplot.
6. Multi plot using matplotlib.pyplot.

GRADE : _____

Signature of Lab-Incharge : _____

WEEK-8

Week : 8 **Program No : 1** **Date : 15/09/2023**

Developed by : G.Sai Hemanth Kumar

Roll No : 21331A0557

Aim : Implementation of Linear Regression algorithm using Python.

Code :

```
from sklearn.linear_model import LinearRegression
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd

data = pd.read_csv("/content/Linear Regression.csv")
df = pd.DataFrame(data)
res = []
for column in df.columns :
  li = df[column].tolist()
  res.append(li)
X = np.array(res[0]).reshape((-1,1))
Y = np.array(res[1])
print("X-Values :\n", X)
print("Y-Values :", Y)

model = LinearRegression()
model.fit(X,Y)
c = model.intercept_
m = model.coef_[0]
print("\nSlope :",m)
print("Intercept :",c)
```

```
plt.plot(X, Y, '*')
plt.plot(X, m*X+c)
plt.xlabel('X')
plt.ylabel('Y')
plt.title("Linear Regression")

a = float(input("\nEnter X for prediction : "))
Z = m * a + c
print("The Value of Y would be :",Z)
print("\nThe Graph of Linear Regression is :")
plt.show()
```

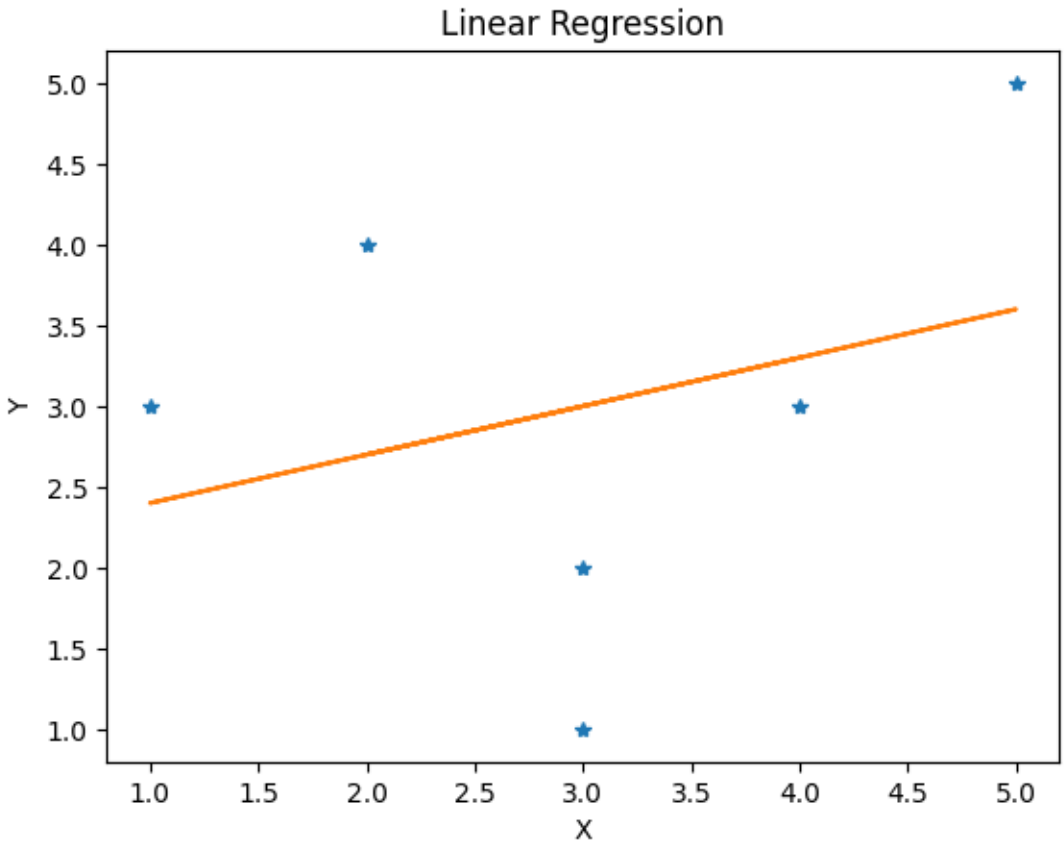
Output :

```
X-Values :
 [[4]
 [2]
 [3]
 [5]
 [1]
 [3]]
Y-Values : [3 4 2 5 3 1]

Slope : 0.3
Intercept : 2.1

Enter X for prediction : 10
The Value of Y would be : 5.1

The Graph of Linear Regression is :
```


Linear Regression
5.0
4.5
4.0
3.5
3.0
2.5
2.0
1.5
1.0
Y
1.0
1.5
2.0
2.5
3.0
3.5
4.0
4.5
5.0
X

Week : 8 **Program No : 2** **Date : 15/09/2023**

Developed by : G.Sai Hemanth Kumar

Roll No : 21331A0557

Aim : Implementation of Logistic Regression using Python.

Code :

```
from sklearn.linear_model import LogisticRegression
import numpy as np
import matplotlib.pyplot as plt

X = np.array([1,2,3,4,5,6,7,8]).reshape((-1,1))
Y = np.array([0,0,0,1,1,1,0,1])
print("X-Values :\n",X)
print("\nY-Values :\n",Y)

model = LogisticRegression()
model.fit(X,Y)
c = model.intercept_
m = model.coef_
print("Intersept :",c)
print("Slope :",m)

plt.scatter(X,Y,marker = '*')
X_range = np.linspace(min(X), max(X), 10)
Logistic_Curve = 1 / (1 + np.exp(-m * X_range - c))
plt.plot(X_range, Logistic_Curve, color = 'red', linewidth = 1)
plt.xlabel("X")
plt.ylabel("Probability")
```

```
plt.title("Logistic Regression")


a = float(input("\nEnter X for Prediction : "))
p = 1 / (1 + np.exp(-m * a - c))
print("The Probability would be :",p)
print("\nThe Graph of Logistic Regression is :")
plt.show()
```

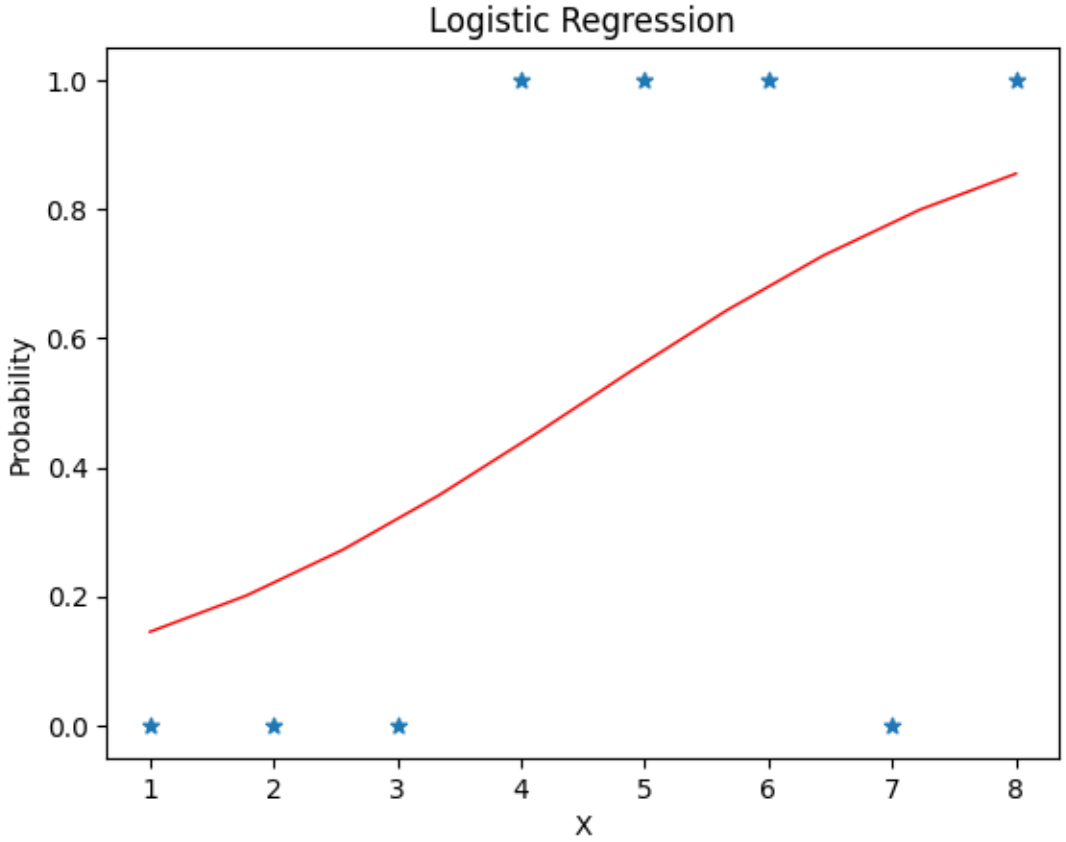
Output :

```
X-Values :
 [[1]
 [2]
 [3]
 [4]
 [5]
 [6]
 [7]
[8]]

Y-Values :
 [0 0 0 1 1 1 0 1]
Intersept : [-2.28081078]
Slope : [[0.50684726]]

Enter X for Prediction : 4.2
The Probability would be : [[0.46206]]

The Graph of Logistic Regression is :
```


Logistic Regression
Probability
X
1.0
0.8
0.6
0.4
0.2
0.0
1
2
3
4
5
6
7
8

END OF THE WEEK : 8

Concepts Covered :

1. Linear Regression.
2. Logistic Regression.

GRADE : _____

Signature of Lab-Incharge : _____

WEEK-9

Week : 9 **Program No : 1** **Date : 22/09/2023**

Developed by : G.Sai Hemanth Kumar

Roll No : 21331A0557

Aim : Using cv2 write a program in python to read and write the image.

Code :

```
import cv2
from google.colab.patches import cv2_imshow
im = cv2.imread('/content/Forest.jpeg')
print("Image :")
cv2_imshow(im)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

Output :

Image :

Week : 9 **Program No : 2** **Date : 22/09/2023**

Developed by : G.Sai Hemanth Kumar

Roll No : 21331A0557

Aim : Using cv2 write a program in python to convert and print original image into grayscale image.

Code :

```
import cv2
from google.colab.patches import cv2_imshow

image = cv2.imread('/content/Forest.jpeg')
gray_image = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)

print("Original Image :")
cv2_imshow(image)

print("\nGrayScale Image :")
cv2_imshow(gray_image)

cv2.waitKey(0)
cv2.destroyAllWindows()
```

Output :

Original Image :

GrayScale Image :

Week : 9 **Program No : 3** **Date : 22/09/2023**

Developed by : G.Sai Hemanth Kumar

Roll No : 21331A0557

Aim : Using cv2 write a program in python to convert and print original image into grayscale image, RGB image, HSV image.

Code :

```
import cv2
from google.colab.patches import cv2_imshow

image = cv2.imread('/content/Forest.jpeg')
print("Original Image :")
cv2_imshow(image)

gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
print("\nGrayScale Image :")
cv2_imshow(gray)

rgb = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)
print("\nRGB Image :")
cv2_imshow(rgb)

hsv = cv2.cvtColor(image, cv2.COLOR_BGR2HSV)
print("\nHSV Image :")
cv2_imshow(hsv)

cv2.waitKey(0)
cv2.destroyAllWindows()
```

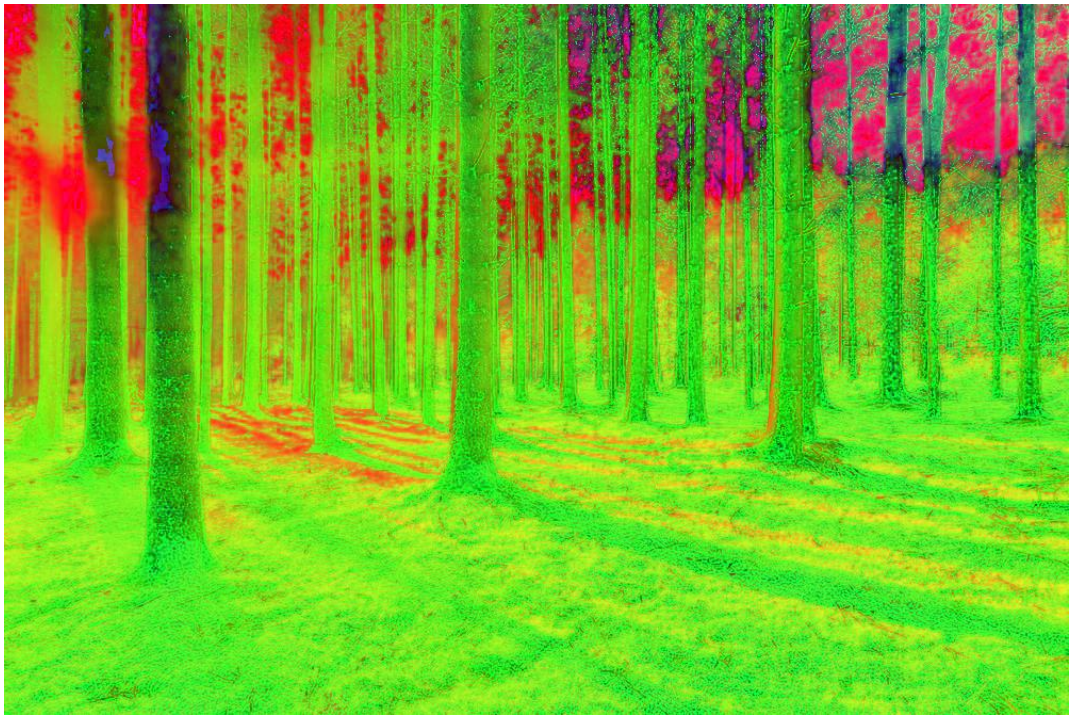
Output :

Original Image :

GrayScale Image :

RGB Image :

HSV Image :



Week : 9 **Program No : 4** **Date : 22/09/2023**

Developed by : G.Sai Hemanth Kumar

Roll No : 21331A0557

Aim : Write a program in python to print dimensions of the image.

Code :

```
import cv2
from google.colab.patches import cv2_imshow
im = cv2.imread('/content/Forest.jpeg')
print("Image :")
cv2_imshow(im)
print("\nDimensions of the Image is :",im.ndim)
print("Shape of the Image is :",im.shape)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

Output :

Image :

Dimensions of the Image is : 3

Shape of the Image is : (667, 1000, 3)

Week : 9 **Program No : 5** **Date : 22/09/2023**

Developed by : G.Sai Hemanth Kumar

Roll No : 21331A0557

Aim : Using cv2 write a program in python to rotate image in different angles.

Code :

```
import cv2
from google.colab.patches import cv2_imshow

image = cv2.imread('/content/Forest.jpeg')
height, width = image.shape[:2]
center = (width/2, height/2)

rotate_matrix = cv2.getRotationMatrix2D(center=center, angle=45, scale=1)
rotated_image = cv2.warpAffine(src=image, M=rotate_matrix, dsize=(width, height))

print("Original Image :")
cv2_imshow(image)

print("\nRotated Image :")
cv2_imshow(rotated_image)

cv2.waitKey(0)
cv2.destroyAllWindows()
```

.

Output :

Original Image :

Rotated Image :

END OF THE WEEK : 9

Concepts Covered :

1. Read an Image.
2. Write an Image.
3. Image Color Convertions.
4. GrayScale Image.
5. RGB Image.
6. HSV Image.
7. Dimensions of an Image.
8. Image Rotation.

GRADE : _____

Signature of Lab-Incharge : _____

WEEK-10

Week : 10 **Program No : 1** **Date : 29/09/2023**

Developed by : G.Sai Hemanth Kumar

Roll No : 21331A0557

Aim : Program for Face detection using Haar cascade classifier in OpenCV.

Code :

```
import cv2
import numpy as np
from google.colab.patches import cv2_imshow

face_cascade = cv2.CascadeClassifier('haarcascade_frontalface_default.xml')
image_path = '/content/Group.jpg'

image = cv2.imread(image_path)
print("Original Image :")
cv2_imshow(image)
print("\nAfter detection of faces in Original Image :")

faces = face_cascade.detectMultiScale(image, scaleFactor=1.5, minNeighbors=5, minSize=(30, 30))
for (x, y, w, h) in faces:
  cv2.rectangle(image, (x, y), (x + w, y + h), (0, 255, 0), 2)

cv2_imshow(image)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

Output :

Original Image :

After detection of faces in Original Image :

Week : 10 **Program No : 2** **Date : 29/09/2023**

Developed by : G.Sai Hemanth Kumar

Roll No : 21331A0557

Aim : Predict whether two images are similar using SIFT for feature detection and BF matcher algorithms.

Code :

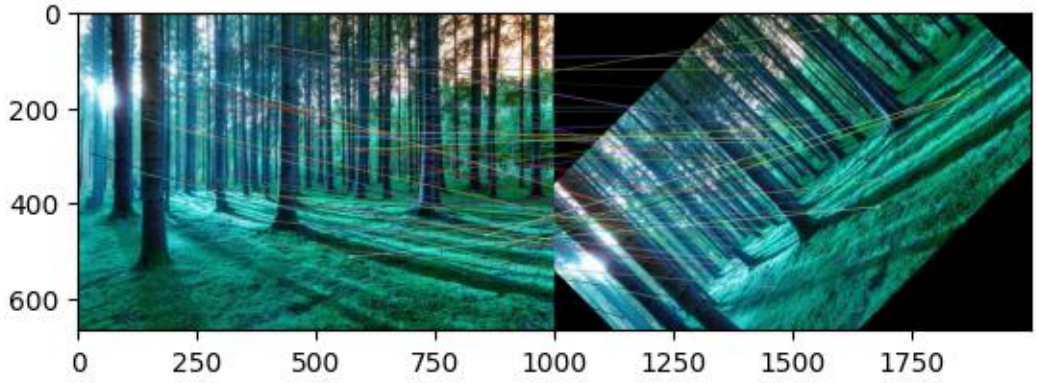
```
import numpy as np
import cv2
import matplotlib.pyplot as plt
from google.colab.patches import cv2_imshow
img1 = cv2.imread('/content/Forest.png')
img2 = cv2.imread('/content/Rotated_Image.png')
print("Original Image :")
cv2_imshow(img1)
print("\nRotated Image :")
cv2_imshow(img2)
print("\nMatching both the images :")
sift = cv2.SIFT_create()
kp1, des1 = sift.detectAndCompute(img1,None)
kp2, des2 = sift.detectAndCompute(img2,None)
bf = cv2.BFMatcher()
matches = bf.match(des1,des2)
matches = sorted(matches, key=lambda val: val.distance)
out = cv2.drawMatches(img1, kp1, img2, kp2, matches[:50], None, flags=2)
plt.imshow(out)
plt.show()
cv2.waitKey(0)
cv2.destroyAllWindows()
```

Output :

Original Image :

Rotated Image :

Matching both the images :



END OF THE WEEK : 10

Concepts Covered :

1. Face detection using Haar cascade classifier in OpenCV.
2. Predict whether two images are similar using SIFT.

GRADE : _____

Signature of Lab-Incharge : _____