WEEK-1

1) Exception Handling and User defined exception(s)

a) Write a python program to catch following exception i) Value Error ii) Index Error iii) Name Error iv) Type Error v) DivideZero Error

Code:

Divivsion by zero error:

```
 n=int(input("Enter a number:"))
try:
 print(n/0)
except:
 print("Division by zero is undefined")
```

Index error:

```
 l=[]
for i in range(0,4):
 a=int(input())
 l.append(a)
try:
 print(l[4])
except:
 print("Index is out of range")
```

Value error:

```
 import math
n=int(input("Enter a number:"))
try:
 print("The square root of number is",math.sqrt(n))
except:
 print("Enter a positive value")
```

Type error:

```
 n=input("Enter a string:")
try:
  print(n+2)
except:
  print("Type error encountered")
```

Name error:

```
a=int(input("Enter a number:"))
try:
  print("Value of a is",b)
except:
  print("Name error is encountered")
```

All errors:

```
try :
  a = int(input("enter a number"))
  b=int(input("enter other number"))
  c= str(input("enter a string"))
  list_0=["gvp","college","cse","engineering"]
  print(a+b)
  print(list_0[6])
  print(a/b)
  printf(lsit_0)
except TypeError:
    print("TypeError-operation cannot be performed")
except ZeroDivisionError:
    print("Zerodivisionerror- operation cannot be performed")
except NameError:
    print("NameError- operation cannot be performed")
```

```
except ValueError:
    print("valueError-operation cannot be performed")
except IndexError:
    print("indexerror-operation cannot be performed")
else:
 print("NO ERRORS FOUND")
```

b) Write a python program to create user defined exceptions.

Code:

```
class Error(Exception):
  pass
class ValueTooSmallError(Error):
  pass
class ValueTooLargeError(Error):
  pass
number = 10
while True:
    try:
        num = int(input("Enter a number: "))
        if num < number:
            raise ValueTooSmallError
        elif num > number:
            raise ValueTooLargeError
        break
    except ValueTooSmallError:
        print("This value is too small, try again!")
    except ValueTooLargeError:
        print("This value is too large, try again!")


print("Congratulations! You guessed it correctly.")
```

c) Write a python program to understand the use of else and finally block with try block.

```
 import math
try:
  a = int(input("Enter num1: "))
  b = int(input("Enter num2: "))
  print(math.sqrt(a))
  num = 5
  print(a/b)
  print("Value of num is" , num)
  list1 = [123,23,43,32,65]
  print(list1[2])
  print(a)
except ValueError:
  print("Value Error is handled")
except IndexError:
  print("Index Error is handled")
except NameError:
  print("Name Error is handled")
except TypeError:
  print("Type Error is handled")
except ZeroDivisionError:
  print("Zero Division Error is handled")
else:
  print("NO errors")
finally:
  print("END")
```

d) Write a python program that uses raise and exception class to throw an exception.

```
try :
 a = int(input("enter a number"))
 b=int(input("enter other number"))
 c= str(input("enter a string"))
 x=int(input("enter your age"))
 if x<18:
   raise exception("you are not eligible")
 if x>18:
   print("you are eligible")
 print(a+b)
 print(a/b)
except TypeError:
   print("TypeError-operation cannot be performed")
except ZeroDivisionError:
   print("Zerodivisionerror- operation cannot be performed")
except NameError:
   print("NameError- operation cannot be performed")
except ValueError:
   print("valueError-operation cannot be performed")
except IndexError:
   print("indexerror-operation cannot be performed")
else:
 print("NO ERRORS FOUND")
```

WEEK-3

NUMPY LIBRARY

a) Create a numpy array from list, tuple with float type

```
import numpy as np
l1 = [9,8,7,6]
arr1 =np.array(l1,dtype='f')
t1 = (1,2,3,4)
arr2 = np.array(t1,dtype = 'f')
print(arr1)
print(arr2)
print(arr1.dtype)
```

b) Python program to demonstrate slicing, integer and boolean array indexing

```
import numpy as np
l1 = [11,22,33,44,50,55,60,65,70,75,80,85,90]
arr1 =np.array(l1)
#slicing with integer array indexing
print(arr1[1:3])
#boolean array indexing
print(arr1[arr1>50])
```

c) Write a python program to find min, max, sum, cumulative sum of array.

```
import numpy as np
arr = np.array([[19,9,72],[9,8,80],[15,9,3],[3,5,5]])
print(arr)
#maximum
print("Maximum element is: ",np.max(arr))
#min
print("Minimum element is: ",arr.min())
#sum
print("Sum of elements in array is: ",arr.sum())
```

```
#cumulative sum
print("Cumulative sum of elements in array is:\n",arr.cumsum(axis=1))
print("Cumulative sum of elements in array is:\n",np.cumsum(arr,axis=1))
```

d) Write a python program to demonstrate use of ndim, shape, size, dtype.

```
 import numpy as np
arr = np.array([[19,9,72],[9,8,80],[15,9,3],[3,5,5]])
print(arr)
#ndim(no.of dimensions of array)
print("No.of dimensions in array: ",arr.ndim)
#shape(shape of array)
print("Shape of an array is: ",arr.shape)
#size(total no.of elements in array)
print("Size of an array: ",arr.size)
#dtype
print("Data type of elements in array: ",arr.dtype)
```

WEEK-4

Numpy Library: Linear Algebra

a) Write a python program to find rank, determinant, and trace of an array

```
 import numpy as np
r=int(input("Enter the number of Rows:"))
c=int(input("Enter the number of Columns:"))
matrix=[]
print("Enter the Elements Rowwise:")
for i in range(r):
  a=[]
  for j in range(c):
    a.append(int(input()))
  matrix. append(a)
m=np.array(matrix)
```

```
print("The original matrix is:")
for i in range(r):
  for j in range(c):
    print(matrix[i][j],end=" ")
  print()
print("Rank of given matrix is:",np.linalg.matrix_rank(m))
print("\nTrace of given matrix is:",np.trace(m))
print("\nDeterminent of give matrix is:",np.linalg.det(m))
```

b) Write a python program to find eigenvalues of matrices

```
 import numpy as np
r=int(input("Enter the number of Rows of Square matrix:"))
c=int(input("Enter the number of Columns of Square matrix:"))
matrix=[]
print("Enter the Elements Rowwise:")
for i in range(r):
  a=[]
  for j in range(c):
    a.append(int(input()))
  matrix. append(a)
m=np.array(matrix)
print("The original matrix is:")
for i in range(r):
  for j in range(c):
    print(matrix[i][j],end=" ")
  print()
w,v=np.linalg.eig(m)
print("The Eigen values of the given Square matrix are:\n",w)
print("The Eigen vectors of the given Square matrix are:\n",v)
```

c) Write a python program to find matrix and vector products (dot, inner, outer,product), matrix exponentiation.

```
import numpy as np
a = np.array([4, 6])
b = np.array([5, 11])
print("The Given Vectors are :")
print("a = ", a)
print("\nb = ", b)

print("\nInner product of vectors a and b =")
print(np.inner(a, b))
print("\nOuter product of vectors a and b =")
print(np.outer(a, b))
print("\nDot product of vectors a and b =")
print(np.dot(a, b))




print("----------------------------------")

x = np.array([[4, 3], [5, 2]])
y = np.array([[1, 5],[8,5]])
print("\nThe Given Matrices are:")
print("x =", x)
print("\ny =", y)

print("\nInner product of matrices x and y =")
print(np.inner(x, y))
print("\nOuter product of matrices x and y =")
print(np.outer(x, y))
```

```
print("\nDot product of matrices x and y =")
print(np.dot(x, y))
```

Matrix Exponentiation:

```
import numpy as np
r=int(input("Enter the number of Rows:"))
c=int(input("Enter the number of Columns:"))
matrix=[]
print("Enter the Elements Rowwise:")
for i in range(r):
 a=[]
 for j in range(c):
  a.append(int(input()))
 matrix. append(a)
m=np.array(matrix)
print("The original matrix is:")
for i in range(r):
 for j in range(c):
  print(matrix[i][j],end=" ")
 print()
n=int(input("Enter the Power you want to raise:"))
print("The Matrix Exponentiation is:")
print(np.linalg.matrix_power(m,n))
```

d) Write a python program to solve a linear matrix equation, or system of linear scalar equations.

```
import numpy as np
r=int(input("Enter the number of variables in the Equation:"))
c=int(input("Enter the number of Equations:"))
matrix=[]
print("Enter the Elements Rowwise:")
for i in range(r):
 a=[]
```

```
 for j in range(c):
   a.append(int(input()))
 matrix. append(a)
m=np.array(matrix)
n=[]
b=int(input("Enter the number of Equations:"))
print("Enter the values of constants:")
for i in range(0,b):
 ele=int(input())
 n.append(ele)
x=np.linalg.solve(m,n)
print("The Solution for the given Equations is:")
print(x)
```

WEEK-5

Numpy Advanced

a) Create a white image using NumPy in Python

```
from PIL import Image
import numpy as np
a = np.full((512, 256, 3), 255, dtype=np.uint8)
image = Image.fromarray(a, "RGB")
image.save("white.png", "PNG")
```

b) Convert a NumPy array to an image and Convert images to NumPy array

```
 from PIL import Image
from numpy import *
Image=Image.open('images.png')
data=asarray(Image)
print(data)
m=Image.fromarray(data,"RGB")
m.save('pic.png')
```

c) Perform Sorting, Searching and Counting using Numpy methods.

```
import numpy as np
a = np.array([[23, 65], [90, 33]])
arr1 = np.sort(a, axis = 0)
print ("Along first axis : \n", arr1)
a = np.array([[70, 86], [42, 18]])
arr2 = np.sort(a, axis = 1)
print ("\nAlong first axis : \n", arr2)
a = np.array([[36, 45], [81, 99]])
arr1 = np.sort(a, axis = None)
print ("\nAlong none axis : \n", arr1)
```

d) Write a program to demonstrate the use of the reshape() method.

```
import numpy as np
array = np.array([1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15,16])
print("Array : " + str(array))
reshaped1 = array.reshape((4, 4))
print("First Reshaped Array : ")
print(reshaped1)
reshaped2 = np.reshape(array, (8, 2))
print("Second Reshaped Array : ")
print(reshaped2)
```

WEEK-6

Pandas Library

a) Write a python program to implement Pandas Series with labels

```
import pandas as pd
import numpy as np
n=int(input("Enter the number of Elements:"))
```

```
m=[]
print("Enter the Elements:")
for i in range(0,n):
  ele=int(input())
  m.append(ele)
ar1=np.array(m)
index=[]
print("Enter the Labels:")
for i in range(0,n):
  lb=input()
  index.append(lb)
ar2=np.array(index)
s=pd.Series(ar1,index=ar2)
print(s)
```

b) Create a Pandas Series from a dictionary.

```
 import pandas as pd
dictionary ={ 'a':2,'b':3,'c':4,'d':5}
series=pd.Series(dictionary)
print(series)
```

c) Create Pandas Dataframe

```
import pandas as pd
data={"groceries":["soap","shampoo","deodrant"]  , "price":[250,200,100]}
bill=pd.DataFrame(data)
print(bill)
```

d) Write a program which make use of following Pandas methods

i) describe():

```
import pandas as pd
data = [[10, 18, 11], [13, 15, 8], [9, 20, 3]]
df = pd.DataFrame(data)
print(df.describe())
```

ii) head():

```
import pandas as pd
data = {'Name': ['Rohit', 'Mohit', 'Shubh'],
        'Class': ['10', '09', '11'], 'Roll no': ['25', '37', '48'], 'Fav Subject': ['C++', 'Python', 'Kotlin'], 'Favourite Sports': ['Football', 'Basketball', 'Hockey']}
data_frame=pd.DataFrame(data)
print(data_frame)
print(data_frame.head(1))
```

iii) tail():

```
import pandas as pd
data = {'Name': ['Rohit', 'Mohit', 'Shubh'],
        'Class': ['10', '09', '11'], 'Roll no': ['25', '37', '48'], 'Fav Subject': ['C++', 'Python', 'Kotlin'], 'Favourite Sports': ['Football', 'Basketball', 'Hockey']}
data_frame=pd.DataFrame(data)
 print(data_frame)
print(data_frame.tail(1))
```

WEEK-7

Pandas Library: Selection

a) Write a program that converts Pandas DataFrame and Series into numpy.array.

```
import pandas as pd
data = {'Age': [25,47,38],
      'Birth Year': [1995,1973,1982],
      'Graduation Year': [2016,2000,2005]
      }
df = pd.DataFrame(data, columns = ['Age','Birth Year','Graduation Year'])
my_array = df.to_numpy()
print(my_array)
print(type(my_array))
```

b) Write a program that demonstrates the column selection, column addition, and column deletion.

```
 d = {'one' : pd.Series([1, 2, 3], index=['a', 'b', 'c']),
  'two' : pd.Series([1, 2, 3], index=['a', 'b', 'c'])}
df = pd.DataFrame(d)
```

```
print ("Adding a new column by passing as Series:")
df['three']=pd.Series([10,20,30],index=['a','b','c'])
print (df)
print ("Adding a new column using the existing columns in DataFrame:")
df['four']=df['one']+df['three']
print (df)
print ("Deleting the first column using DEL function:")
del df['one']
print (df)
```

c) Write a program that demonstrates the row selection, row addition, and row deletion.

```
import pandas as pd
d = {'One':pd.Series([1,2,3],index=['a','b','c']),'Two':pd.Series([4,5,6],index=['a','b','c'])}
df = pd.DataFrame(d)
df['Three']=pd.Series([7,7,8],index=['a','b','c'])
print(df)
print (df.loc['b'])
print (df.loc['a'])
print(df[2:3])
df1 = pd.DataFrame([[1,2,5],[3,4,6]],columns=['a','b','c'],index=[1,2])
df2 = pd.DataFrame([[11,12,15],[13,14,16]],columns=['a','b','c'],index=[3,4])
df1 = df1.append(df2)
df3=df1.drop(2)
```

d) Get n-largest and n-smallest values from a particular column in Pandas dataFrame

```
 import pandas as pd
 df = pd.DataFrame([[10, 20, 30, 40], [7, 14, 21, 28], [55, 15, 8, 12],
                    [15, 14, 1, 8], [7, 1, 1, 8], [5, 4, 9, 2]],
                   columns=['Apple', 'Orange', 'Banana', 'Pear'],
                   index=['Basket1', 'Basket2', 'Basket3', 'Basket4',
                          'Basket5', 'Basket6'])
 print("n smallest ")
print(df.nsmallest(2, ['Apple']))
```

```
print(" n largest ")
print(df.nlargest(2, ['Apple']))
```

WEEK-8

Pandas Library: Visualization

a) Write a program which use pandas inbuilt visualization to plot following graphs: i. Bar plots ii. Histograms iii. Line plots iv. Scatter plots

```
-> import matplotlib.pyplot as plt
import numpy as np
x = np.array(['A','B','C','D'])
y = np.array([2,4,6,8])
plt.bar(x,y,width=0.4)
plt.show()
-> import matplotlib.pyplot as plt
import numpy as np
x = np.array(['A','B','C','D'])
y = np.array([2,4,6,8])
plt.bar(x,y,width=0.4)
plt.show()
-> import matplotlib.pyplot as plt
import numpy as np
y = np.array([35, 25, 25, 15])
mylabels = ["Apples", "Bananas", "Cherries", "Dates"]
myexplode = [0.2, 0, 0, 0]
plt.pie(y, labels = mylabels, explode = myexplode, shadow = True)
plt.show()
-> import numpy as np
x = np.array([5,7,8,7,2,17,2,9,4,11,12,9,6])
y = np.array([99,86,87,88,111,86,103,87,94,78,77,85,86])
colors = np.array([0, 10, 20, 30, 40, 45, 50, 55, 60, 70, 80, 90, 100])
plt.scatter(x, y, c=colors, cmap='viridis')
```

```
plt.colorbar()
plt.show()
-> import matplotlib.pyplot as plt
import numpy as np
y1 = np.array([6, 9, 3, 14])
y2 = np.array([8, 5, 7, 13])
plt.plot(y1)
plt.plot(y2)
plt.show()
```

b) Write a program to demonstrate use of groupby() method.

```
import pandas as pd
data = {'Name': ['Chandler', 'Joey', 'Monica', 'Ross','Phoebe'],'Percentage': [92, 88, 88, 86,90],'Course': ['Sarcasm','Innocency','Perfectionist','Dinosaurs','Beauty']}
df = pd.DataFrame(data)
print(df)
grp=df.groupby(['Course','Name'])
print(grp)
print(grp.groups)
```

Pandas Library: Visualization

a) Write a program which use pandas inbuilt visualization to plot following graphs: i. Bar plots ii. Histograms iii. Line plots iv. Scatter plots

[ ]

```
import matplotlib.pyplot as plt
import numpy as np
x = np.array(['A','B','C','D'])
y = np.array([2,4,6,8])
plt.bar(x,y,width=0.4)
plt.show()
```

[ ]

```
import·matplotlib·as·mb
import·pandas·as·pd
import·numpy·as·np
ts·=·pd.Series(np.random.randn(1000),·index=pd.date_range("1/1/2000",·periods=1000))
ts·=·ts.cumsum()
ts.plot();
```

[ ]

```
import·matplotlib.pyplot·as·plt
import·numpy·as·np
y·=·np.array([35,·25,·25,·15])
mylabels·=·["Apples",·"Bananas",·"Cherries",·"Dates"]
myexplode·=·[0.2,·0,·0,·0]
plt.pie(y,·labels·=·mylabels,·explode·=·myexplode,·shadow·=·True)
plt.show()
```

[ ]

```
import·numpy·as·np
x·=·np.array([5,7,8,7,2,17,2,9,4,11,12,9,6])
y·=·np.array([99,86,87,88,111,86,103,87,94,78,77,85,86])
colors·=·np.array([0,·10,·20,·30,·40,·45,·50,·55,·60,·70,·80,·90,·100])
```

```
plt.scatter(x,·y,·c=colors,·cmap='viridis')
plt.colorbar()
plt.show()
```

```
[ ]
import·matplotlib.pyplot·as·plt
import·numpy·as·np
y1·=·np.array([6,·9,·3,·14])
y2·=·np.array([8,·5,·7,·13])
plt.plot(y1)
plt.plot(y2)
plt.show()
```

b) Write a program to demonstrate use of groupby() method.

```
import pandas as pd
data = {'Name': ['Chandler', 'Joey', 'Monica', 'Ross','Phoebe'],'Percentage': [92, 88, 88, 86,90],'Course': ['Sarcasm','Innocency','Perfectionist','Dinosaurs','Beauty']}
df = pd.DataFrame(data)
print(df)
grp=df.groupby(['Course','Name'])
print(grp)
print(grp.groups)
```

c) Write a program to demonstrate pandas Merging, Joining and Concatenating

```
import pandas as pd
dataframe_1 = pd.DataFrame({"Common": ["A", "B", "C", "D", "E"],
  "Name": ["Ram", "Pavan", "Mohan", "Ravi", "Bala"], "Age": [19, 18, 20, 19,18]})
dataframe_2 = pd.DataFrame({"Common": ["A", "B", "C", "D", "E"], "Sport": ["Basketball", "Table tennis", "Cricket", "Volleyball", "Ball badminton "], "Movie": ["Friends", "Mirchi", "Spiderman", "Rangastalam", "Avengers"]})
new_df = pd.merge(dataframe_1, dataframe_2, how="left", on="Common")
```

```
concat_df = pd.DataFrame(dataframe_1,index=[0, 1, 2, 3])
 print(new_df)
print(concat_df)
```

d) Creating dataframes from csv and excel files.

```
from google.colab import files
uploaded=files.upload()
-> import pandas as pd
df = pd.read_csv('california_housing_test.csv')
print(df.to_string())
```

WEEK-9

Object Oriented Programming: basic

a) Write a Python class named Person with attributes name, age, weight (kgs), height (ft) and takes them through the constructor and exposes a method get_bmi_result() which returns one of "underweight", "healthy", "obese"

```
class Person:
 def __init__(self,name,age,weight,height):
  self.name=name
  self.age=age
  self.weight=weight
  self.height=height
 def get_bmi_result(self):
  self.bmi=(self.weight)/(self.height/100)**2
  if(self.bmi<18.5) :
   print(self.bmi)
   print(self.name,"is underweight")
  elif(18.5<self.bmi<24):
   print(self.name,'is healthy')
  else:
   print(self.name,'is obese')
x=Person('Tej',19,73,178)
```

```
x.get_bmi_result()
```

b) Write a python program to demonstrate various kinds of inheritance. #SINGLELEVEL INHERITANCE

```
class Parent:
  def func1(self):
    print("This function is in parent class.")
class Child(Parent):
  def func2(self):
    print("This function is in child class.")
  object = Child()
object.func1()
object.func2()
-> #MULTIPLE INHERITANCE
class A:
  def a(self):
      print(self.a)
class B:
   def b(self):
      print(self.b)
class C(A,B):
   def c(self):
     super().__init__()
   def sum(self):
     print(self.a+self.b)
s1 = C()
s1.a = 7
s1.b = 9
s1.sum()
-> #MULTILEVEL INHERITANCE
class A:
  def __init__(self,i):
```

```
    self.i=i
class B(A):
  def __init__(self,j):
    super().__init__()
    self.j=j
class C(B):
  def __init__(self,k)
    self.k=k
  def sum(self):
    print(self.i+self.j+self.k)
x=C(7)
x.i=5
x.j=6
x.sum()
-> #HIERARCHIAL INHERITANCE
class A:
      def func1(self):
          print("this is function 1")
class B(A):
      def func2(self):
          print("this is function 2")
class C(A):
      def func3(self):
          print("this is function 3")
 ob = B()
ob1 = C()
ob.func1()
ob.func2()
ob1.func3()
-> #HYBRID INHERITANCE
class A:
```

```
    def func1(self):
        print("this is function 1")
class B(A):
    def func2(self):
        print("this is function 2")
class C(A):
    def func3(self):
        print(" this is function 3")
class D(B ,C):
    def func4(self):
        print(" this is function 4")
ob = D()
ob.func1()
ob.func2()
ob.func3()
ob.func4()
```

WEEK-10

Object Oriented Programming: advanced

a) Write a python program to demonstrate operator overloading.

```
class X:
    def __init__(self, a):
        self.a= a
    def __add__(self, b):
        return self.a + b.a
ob1 = X(2)
ob2 = X(3)
ob3 = X("BARRY ")
ob4 = X("THE SLICER")
print(ob1 + ob2)
print(ob3 + ob4)
```

b) Write a python program to create abstract classes and abstract methods.

```
from abc import ABC, abstractmethod
class Polygon(ABC):
@abstractmethod
 def no_of_sides(self):
  pass
class Triangle(Polygon):
 def no_of_sides(self):
  print("Triangle:I have 3 sides")
class Pentagon(Polygon):
 def no_of_sides(self):
  print("Penatagon:I have 5 sides")
class Hexagon(Polygon):
 def no_of_sides(self):
  print("Hexagon:I have 6 sides")
class Quadrilateral(Polygon):
 def no_of_sides(self):
  print("Quadilateral:I have 4 sides")
 R = Triangle()
R.no_of_sides()
R = Quadrilateral()
R.no_of_sides()
R = Pentagon()
R.no_of_sides()
R = Hexagon()
R.no_of_sides()
```

WEEK-11

Python Collections:

a) Write a Python program to show different ways to create Counter

```
from collections import Counter
print(Counter(['B','B','A','B','C','A','B',
    'B','A','C']))
print(Counter({'A':3, 'B':5, 'C':2}))
print(Counter(A=3, B=5, C=2))
```

b) Write a Python program to demonstrate working of OrderedDict.

```
from collections import OrderedDict
s = OrderedDict()
s['a'] = 1
s['b'] = 2
s['c'] = 3
s['d'] = 4
print('Before Deleting')
for key, value in s.items():
  print(key, value)
s.pop('a')
s['d'] = 1
print('\nAfter re-inserting')
for key, value in s.items():
  print(key, value)
```

c) Write a Python program to demonstrate working of defaultdict

```
from collections import defaultdict
d = defaultdict(list)
for i in range(11):
  d[i].append(i)
print("Dictionary with values as list:")
print(d)
```

d) Write a python program to demonstrate working of ChainMap

```
from collections import ChainMap
d1 = {'a': 1, 'b': 2}
d2 = {'c': 3, 'd': 4}
```

```
d3 = {'e': 5, 'f': 6}
c = ChainMap(d1, d2, d3)
print(c['a'])
print(c.values())
print(c.keys())
```

WEEK-12

Python collections:

a) Write a Python program to demonstrate the working of namedtuple() and _make(), _asdict().

```
from collections import namedtuple
Employee=namedtuple('Employee',['name','age'])
x=Employee('Tej',20)
print(x[1])#using index
print(x.age)
-> #_make()
Employee=namedtuple('Employee',['name','age'])
lis=['Tej',20]
x=Employee._make(lis)
print(x)
print(x.name)
-> #_asdict
Employee=namedtuple('Employee',['name','age'])
x=Employee('Tej',20)
x1=x._asdict()
x2=dict(x1)
print(x1)
print(x2)
```

b) Write a Python program to demonstrate the working of deque.

```
import collections
```

```
de = collections.deque([1,2,3,3])
de.append(4)
print ("The deque after appending at right is : ")
print (de)
de.appendleft(6)
print ("The deque after appending at left is : ")
print (de)
de.pop()
print ("The deque after deleting from right is : ")
print (de)
de.popleft()
print ("The deque after deleting from left is : ")
print (de)
print("count of 3 in dequeue")
print (de.count(3))
```

WEEK-13

REGULAR EXPRESSIONS

a) Given an input file which contains a list of names and phone numbers separated by spaces in the following format: i) Phone Number contains a 3- or 2-digit area code and a hyphen followed by an 8-digit number. ii) Find all names having phone numbers with a 3-digit area code using regular expressions.

```
import re
str1 = input()
match=re.search(r'\d{2,3}-\d{8}',str1)
if match:
  print('It is a Valid number ')
  print('The number is '+match.group())
else:
  print("Invalid")
-> import re
```

```
l=["123-12354678 Pavan","121-90594648 Power","1234-90103157 eswar"]
for i in l:
  pattern = r"\d{3}-\d{8}"
  a = re.match(pattern,i)
  if a:
    x=re.split("\s",i)
    print(x[1])
```

c) Write a python program to check the validity of a password given by the user. The password should satisfy the following criteria: i) Contain at least 1 letter between a and z ii) Contain at least 1 number between 0 and 9 iii) Contain at least 1 letter between A and Z iv) Contain at least 1 character from $, #, @ v) Minimum length of password: 6 vi) Maximum length of password: 12

```
import re
password = input()
a=0
b=0
c=0
d=0
e=0
if len(password)>5 and len(password)<13:
  a=1
for i in password:
  if  re.search(r'\d',i):
    b=1
  elif  re.search(r'[a-z]',i):
    c=1
  elif  re.search(r'[A-Z]',i):
    e=1
  elif  re.search(r'[@|#|$]',i):
    d=1
if a==1 and b==1 and c==1 and d==1 and e==1:
  print('password is valid')
```

```
else:
  print('password is invalid')
```

d) Write a Python program to validate mobile number.

```
s = input()
Pattern = re.search(r'[0|91]?[7|8|9]\d{9}',s)
if Pattern:
  print("Valid Number")
else :
  print("Invalid Number")
```

WEEK-15

OS MODULE

Displaying the name

```
-> import os
print(os.name) # os name
```

code to get current working directory

```
-> print(os.getcwd()) # To get current working directory
```

code to create new directory

```
-> os.mkdir("/content/python") #making dir
```

code to rename a directory

```
-> x="python"
os.rename(x,"lab") #renaming
```

code to remove a existing directory

```
-> os.rmdir("/content/lab") #removing
```

code to change the directory path

```
-> os.chdir("/content/pyi") # To change dir
```

code to display list of files

```
-> print(os.listdir("/content")) # To get list of dirs
```

current path

```
-> print(os.path)
```

popen method

```
-> import os, sys
a = 'mkdir nitin'
b = os.popen(a,'r',1)
print (b)
```

open method

```
-> try:
  fname="Pthon.txt"
  f=open(fname,'r')
  t=f.read()
  f.close()
except IOError:
  print(fname)
```

WEEK-16
CALENDER CLASS

```
-> import calendar as cal
year = int(input("Enter the year in 'YYYY' format = "))
month = int(input("Enter the month in 'MM' format = "))
print(cal.month(year,month))
-> import calendar as cal
year = int(input("Enter the year in 'YYYY' format = "))
b = year % 4
a = year-b
if(b>=0 and b<=2):
    print("The nearest Leap Year to",year,"is",a)
    print("The calendar of the year",a,"is:")
    print(cal.calendar(a,2,1,6))
else:
    a = a + 4
    print("The nearest Leap Year to",year,"is",a)
    print("The calendar of the year",a,"is:")
    print(cal.calendar(a,2,1,6))
```

```
-> import calendar as cal
year = int(input("Enter any year in 'YYYY' format = "))
print("Python Calendar of the year",year,"is:")
print(cal.calendar(year,2,1,6))
t = cal.TextCalendar(firstweekday = 0)
print("Text Calendar of",year,"is:")
print(t.pryear(year,3))
t = cal.HTMLCalendar(firstweekday = 0)
print("HTML Calendar of",year,"is:")
print(t.formatyearpage(year))
-> import calendar as cal
import datetime as dt
date = input("Enter the date in format 'DD MM YYYY': ")
born = dt.datetime.strptime(date, '%d %m %Y').weekday()
print("Given Date:",date)
print(cal.day_name[born])
```