LABCYCLE :1

PROGRAM NO:1.1 DATE:

AIM: REVIEW OF PYTHON PROGRAMMING

1. print("Demo of basic data types:Numbers")

   x=3

   y=2.5

   print("x=",x)

   print("y=",y)

   print("Data type of variable x:",type(x))

   print("Data type of variable y:",type(y))

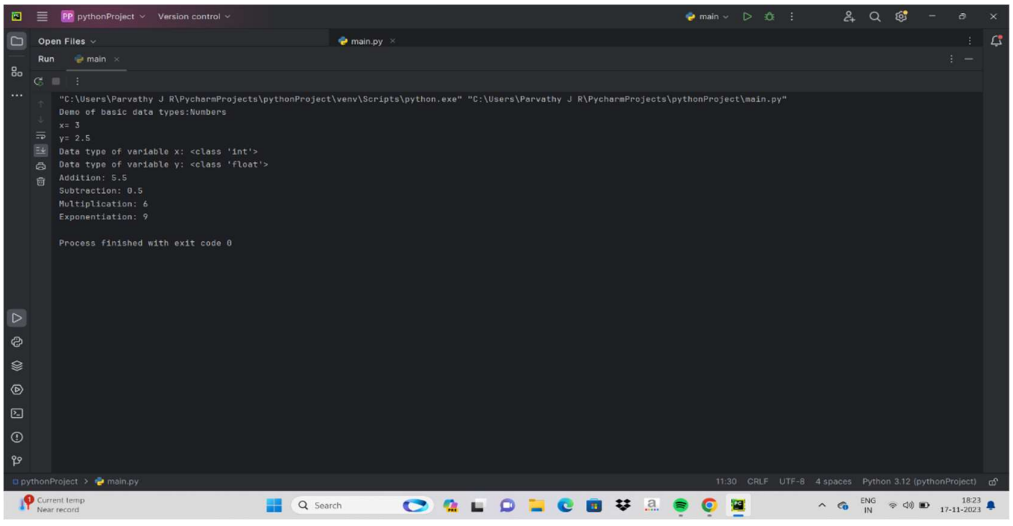
   print("Addition:",x+y)

   print("Subtraction:",x-y)

   print("Multiplication:",x*2)

   print("Exponentiation:",x**2)

OUTPUT



2. print("Demo of basic types :Boolean")

   t=True

   f=False

   print("t=",t)

   print("f=",f)

   print("Data types of variable t:",type(t))

   print("Data types of variable f:",type(f))

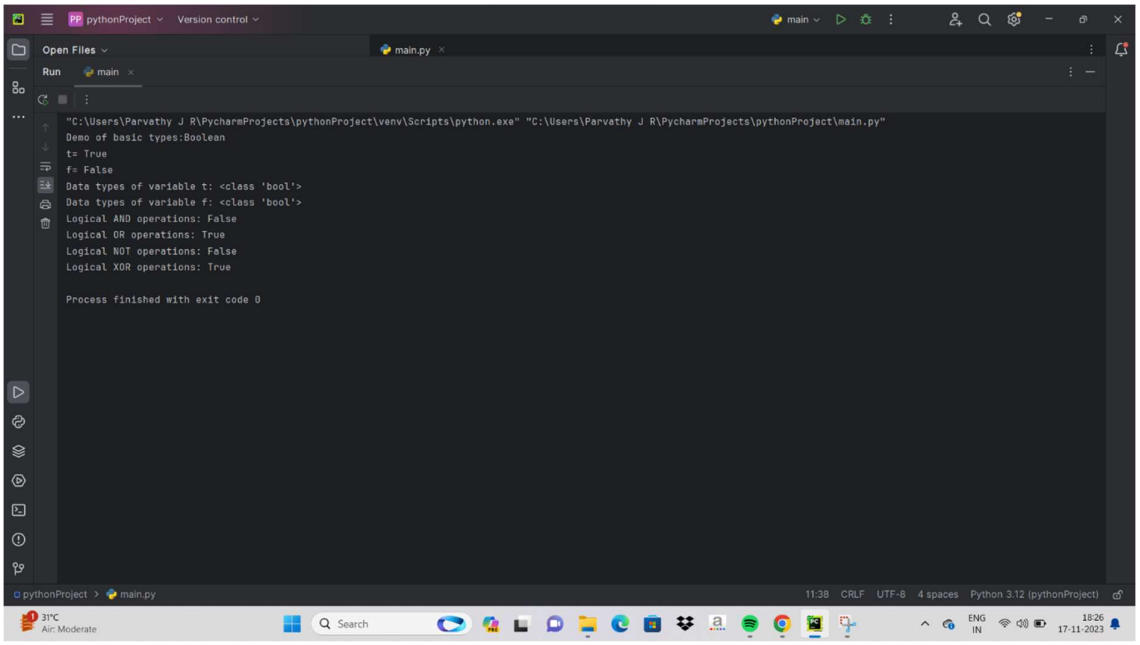
   print("Logical AND operations:",t and f)

   print("Logical OR operations:",t or f)

   print("Logical NOT operations:",not t)

   print("Logical XOR operations:",t!=f)

OUTPUT



3. print("Demo of basic data types:String")

   s="hello"

   t="world"

   print("string1:",s)

   print("string2:",t)

   d=s+""+t

   print("String concatenation:",d)

   print("Capitalize:",d.capitalize())

   print("Converted to upper case:",s.upper())

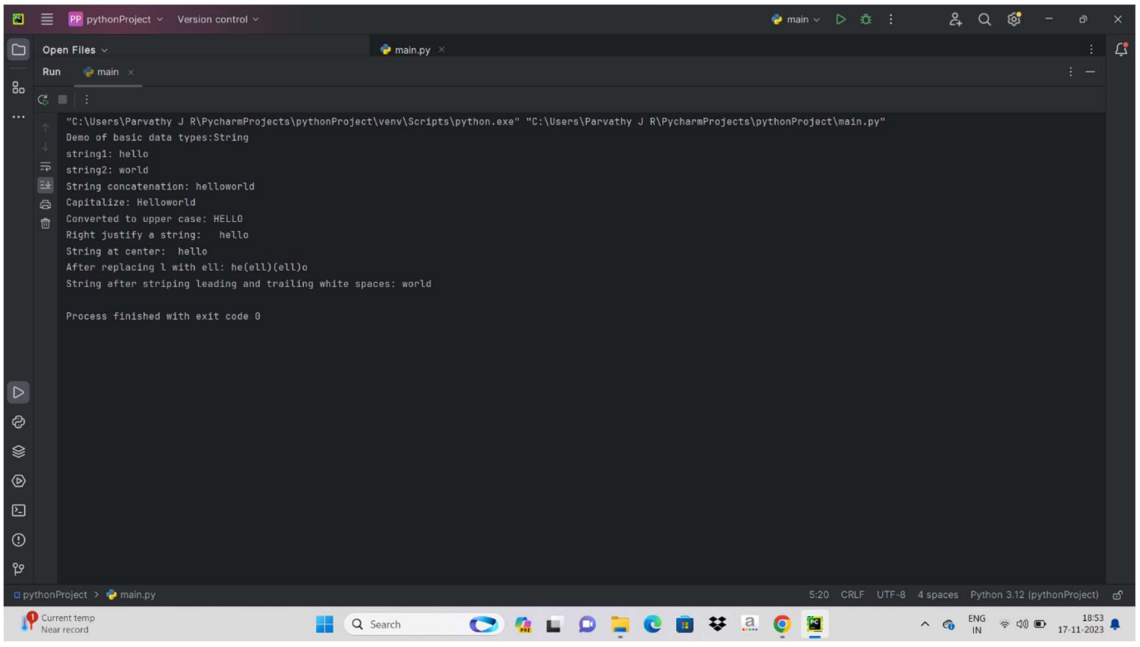
   print("Right justify a string:",s.rjust(7))

   print("String at center:",s.center(7))

   print("After replacing l with ell:",s.replace('l','(ell)'))

   print("String after striping leading and trailing white spaces:",'world'.strip())

OUTPUT



4. print("Containers:Lists")

   nums=list(range(5))

   print("list 'nums'contain:",nums)

   nums[4]="abc"

   print("List can contain elements of different types.Example:",nums)

   nums.append("xyz")

   print("'nums'after inserting new element at the end:")

   print("sublists:")

   print("A slice from index 2 to 4:",nums[2:4])

   print("A slice from index 2 to the end:",nums[2:])

   print("A slice from the start to index 2 :",nums[:2])

   print("A slice of the whole list:",nums[:])

   nums[4:]=[8,9]#Assign a new sublist to a slice

   print("After Assigning a new sublist to 'nums':")

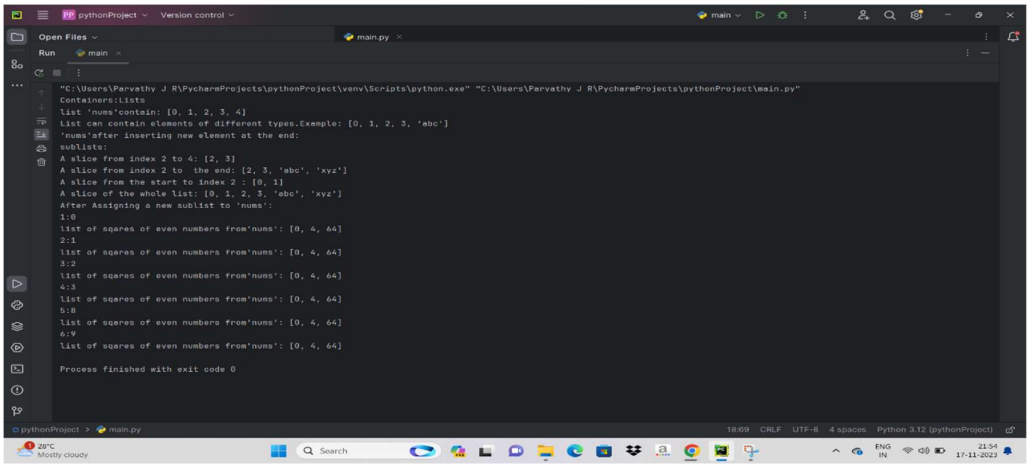
   for idx,i in enumerate(nums):

   print('%d:%s'%(idx + 1,i))

   even_squares=[x**2 for x in nums if x%2==0]

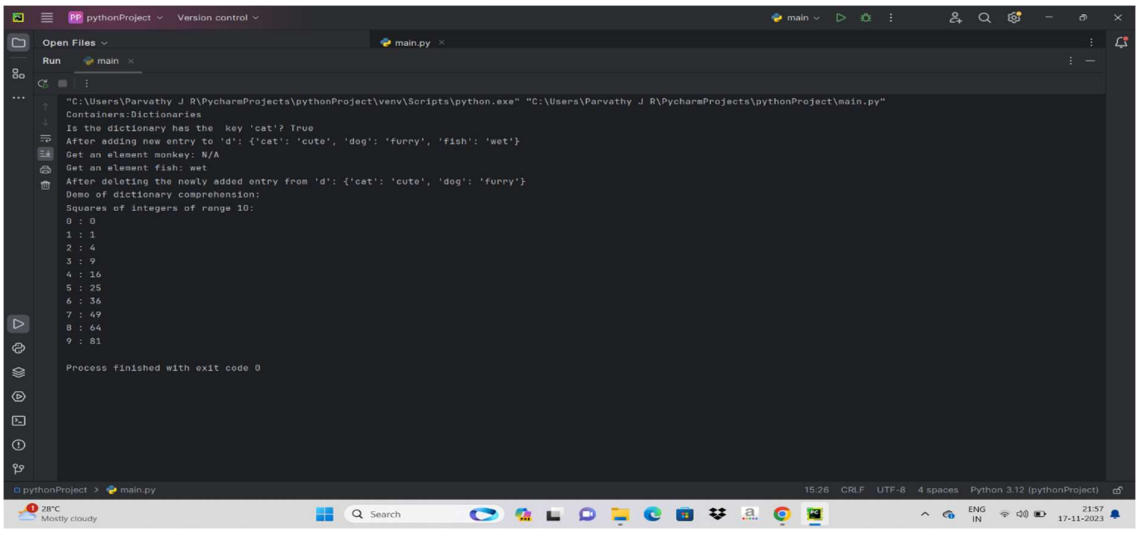
   print("list of sqares of even numbers from'nums':",even_squares)

OUTPUT



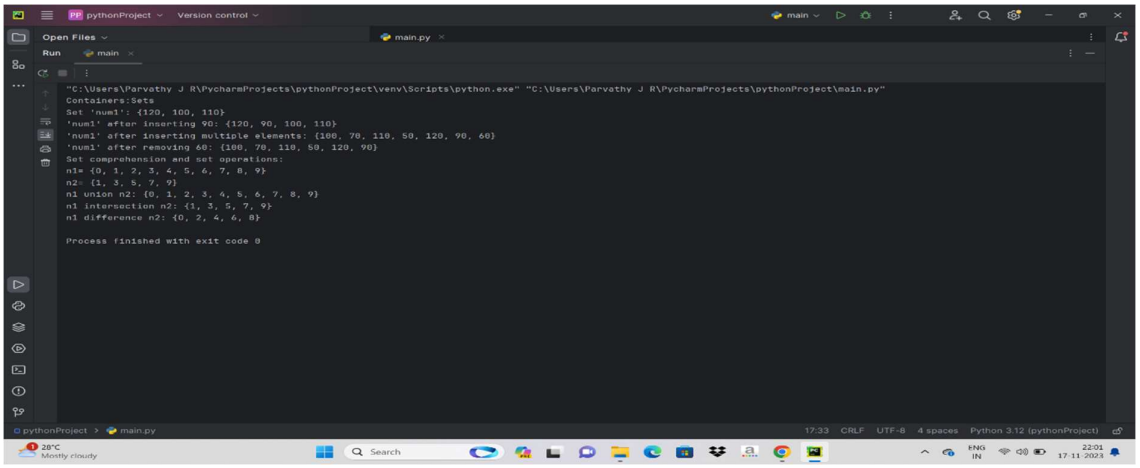
5.
```
print("Containers:Dictionaries")
d=dict()
d={'cat':'cute','dog':'furry'}

print("Is the dictionary has the  key 'cat'?",'cat' in d)

d['fish']='wet'

print("After adding new entry to 'd':",d)

print("Get an element monkey:",d.get('mokey','N/A'))

print("Get an element fish:",d.get('fish','N/A'))

del d['fish']

print("After deleting the newly added entry from 'd':",d)

print("Demo of dictionary comprehension:")

squares={x:x*x for x in range(10)}

print("Squares of integers of range 10:")

for k,v in squares.items():

print(k,":",v)
```

OUTPUT



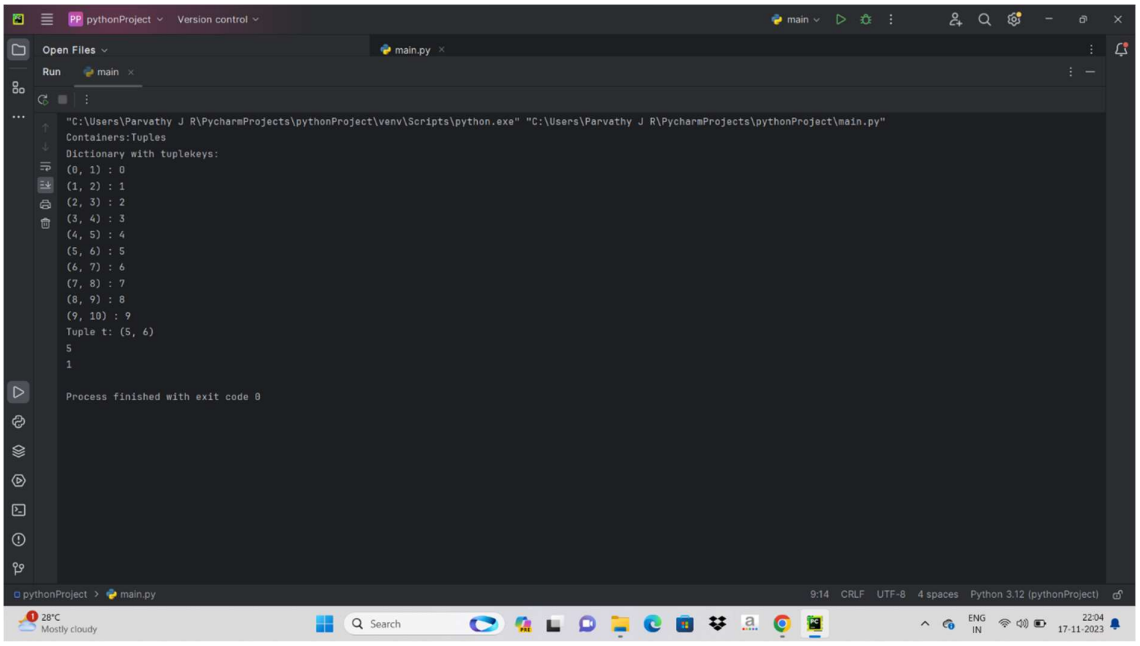
6.
```
print("Containers:Sets")
num1={100,110,120}
print("Set 'num1':",num1)
num1.add(90)
print("'num1' after inserting 90:",num1)
num1.update([50,60,70])
print("'num1' after inserting multiple elements:",num1)
num1.remove(60)
print("'num1' after removing 60:",num1)
print("Set comprehension and set operations:")
n1={x for x in range(10)}
print("n1=",n1)
n2={x for x in range(10) if x%2!=0}
print("n2=",n2)
print("n1 union n2:",n1|n2)
print("n1 intersection n2:",n1&n2)
print("n1 difference n2:",n1-n2)
```

OUTPUT



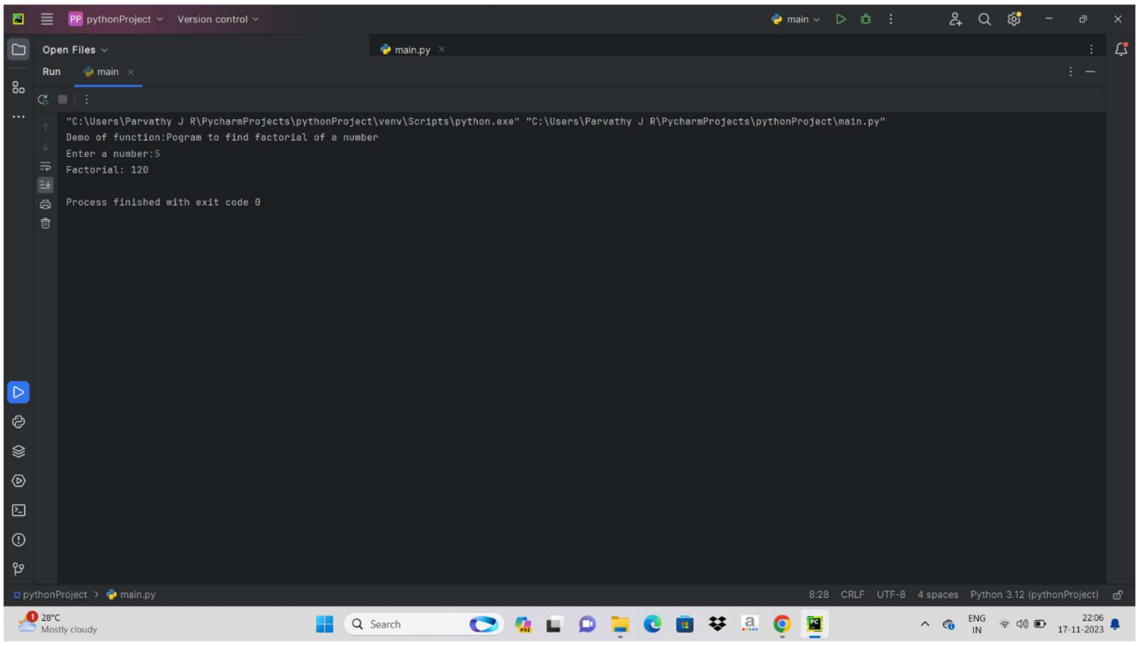
7.
```
print("Containers:Tuples")
d={(x,x+1):x for x in range(10)}
print("Dictionary with tuplekeys:")
for k,v in d.items():
print(k,":",v)
t=(5,6)
print("Tuple t:",t)
print(d[t])
print(d[1,2])
```

OUTPUT



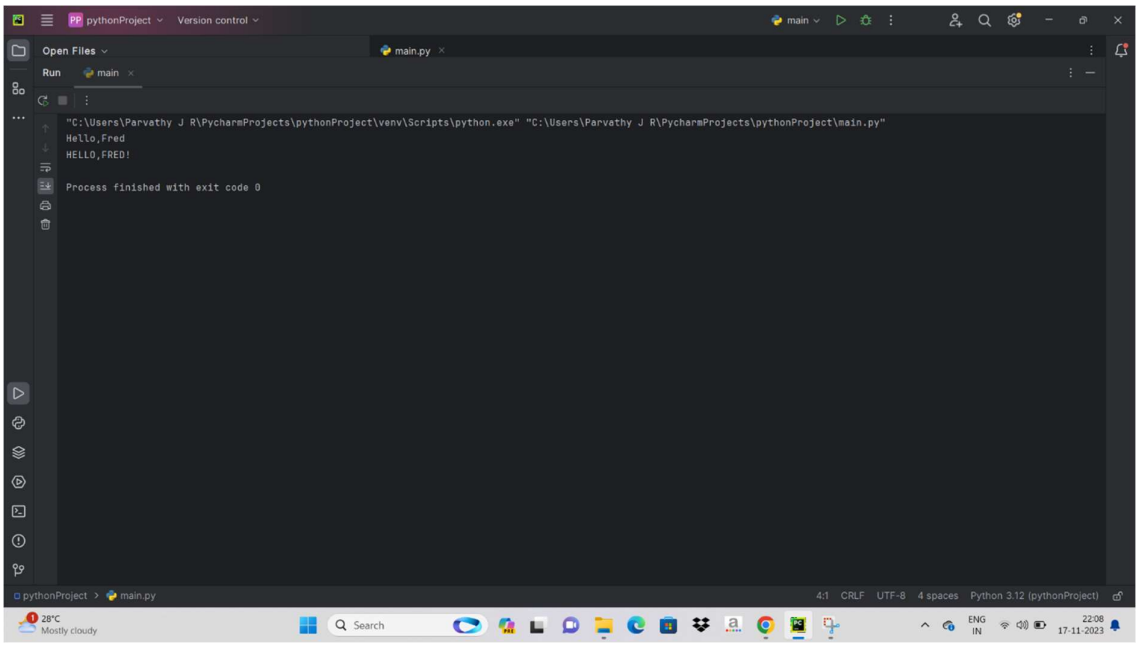
8. print("Demo of function:Pogram to find factorial of a number")
   def fact(n):
   if n==1:
   return 1
   else:
   return(n*fact(n-1))
   n=int(input("Enter a number:"))
   print("Factorial:",fact(n))

OUTPUT



9. class Greeter:
   def __init__(self, name):
   self.name = name
   def greet(self, loud=False):
   if loud:
   print('HELLO,%s!' % self.name.upper())
   else:
   print('Hello,%s' % self.name)
   g = Greeter('Fred')
   g.greet()
   g.greet(loud=True)

OUTPUT



RESULT : The program has been executed successfully and output obtained.

PROGRAM NO: 1.2 DATE:

AIM: Matrix operations(using vectorization) and transformation using python and SVD using python.

1.
```
import numpy as np
a = np.array([1, 2, 3]) # Create a rank 1 array
print("One dimensional array a = ",a)
b = np.array([[1,2,3],[4,5,6]])
print("Two dimensional array b = ",b)
print("Size of the array: ",a.shape)
print("Element at indices 0,1,2 : ",a[0], a[1], a[2])
a[0] = 5 # Change an element of the array
print("Array after changing the element at index 0 : ",a)
a = np.zeros((2,2)) # Create an array of all zeros
print("An array of all zeros : ",a)
b = np.ones((1,2)) # Create an array of all ones
print("An array of all ones : ",b)
c = np.full((2,2), 7) # Create a constant array
print("A constant array : ",c)
d = np.eye (2) # Create a 2x2 identity matrix
print("A 2*2 identity matrix : ",d)
e = np.random.random((2,2)) # Create an array filled with random values
print("An array with random values : ",e)
```

OUTPUT

```
One dimensional array a =  [1 2 3]
Two dimensional array b =  [[1 2 3]
 [4 5 6]]
Size of the array:  (3,)
Element at indices 0,1,2 :  1 2 3
Array after changing the element at index 0 :  [5 2 3]
An array of all zeros :  [[0. 0.]
 [0. 0.]]
An array of all ones :  [[1. 1.]]
A constant array :  [[7 7]
 [7 7]]
A 2*2 identity matrix :  [[1. 0.]
 [0. 1.]]
An array with random values :  [[0.19686022 0.85363223]
 [0.73018189 0.40954629]]
```

2.
```
import numpy as np
print("Array indexing: slicing")
a1 = np.array([[1,2,3,4], [5,6,7,8], [9,10,11,12]])
print("a1 = ",a1)
b = a1[:2, 1:3]
print("Subarray consisting of first two rows and columns 1 and 2: ",b)
b=a1[1:2,:]
print("Subarray consists of second row: ",b)
print("Accessing columns: ")
b=a1[:,1]
print(b,b.shape)
c=a1[:,1:2]
print(c,c.shape)
print("Array integer indexing: ")
a2 = np.array([[1,2], [3, 4], [5, 6]])
print("a2 = ",a2)
print("Example of array integer indexing: ",a2[[0, 1, 2], [0, 1, 0]])
# When using integer array indexing, you can reuse the same element from the source
#array
print(a2[[0, 0], [1, 1]])
# Equivalent to the previous integer array indexing example print(np.array([a2[0, 1],
a[0, 1]]))
a3=a = np.array([[1,2,3], [4,5,6], [7,8,9], [10, 11, 12]])
print("a3 = ",a3)
# Create an array of indices
b = np.array([0, 2, 0, 1])
print("b = ",b)
# Select one element from each row of a using the indices in b
print("a3",a3[np.arange(4), b]) # Mutate one element from each row of a using the
indices in b
a3[np.arange(4), b] += 10
print("a3 = ",a3)
print("Boolean array indexing: ")
a = np.array([[1,2], [3, 4], [5, 6]])
print("a = ",a)
bool_idx= (a > 2)
print("Elements greater than 2: ",a[bool_idx])
```

OUTPUT

```
Array indexing: slicing
a1 =  [[ 1  2  3  4]
 [ 5  6  7  8]
 [ 9 10 11 12]]
Subarray consisting of first two rows and columns 1 and 2:  [[2 3]
 [6 7]]
Subarray consists of second row:  [[5 6 7 8]]
Accessing columns:
[ 2  6 10] (3,)
[[ 2]
 [ 6]
 [10]] (3, 1)
Array integer indexing:
a2 =  [[1 2]
 [3 4]
 [5 6]]
Example of array integer indexing:  [1 4 5]
[2 2]
a3 =  [[ 1  2  3]
 [ 4  5  6]
 [ 7  8  9]
 [10 11 12]]
b =  [0 2 0 1]
a3 [ 1  6  7 11]
a3 =  [[11  2  3]
 [ 4  5 16]
 [17  8  9]
 [10 21 12]]
Boolean array indexing:
a =  [[1 2]
 [3 4]
 [5 6]]
Elements greater than 2:  [3 4 5 6]
```

3. import numpy as np

```
x = np.array([[1,2], [3, 4]], dtype=np.float64)
y = np.array([[6,9], [4, 4]], dtype=np.float64)
print("x = ",x)
print("y=",y)
print("Element wise addition: ",np.add(x, x))
print("Element wise subtraction: ",np.subtract(x, y))
print("Element wise multiplication: ",np.multiply(x, y))
print("Element wise square root of x: ",np.sqrt(x))
print("Matrix multiplication: ",np.dot(x, y))
print("Sum of all elements of matrix x: ",np.sum(x))
print("Sum of elements in each column of matrix y: ",np.sum(y,axis=0))
print("Sum of elements in each row of matrix y: ",np.sum(y,axis=1))
print("Transpose of matrix x: ",x.T)
```

OUTPUT

```
x =  [[1. 2.]
 [3. 4.]]
y= [[6. 9.]
 [4. 4.]]
Element wise addition:  [[2. 4.]
 [6. 8.]]
Element wise subtraction:  [[-5. -7.]
 [-1.  0.]]
Element wise multiplication:  [[ 6. 18.]
 [12. 16.]]
Element wise square root of x:  [[1.         1.41421356]
 [1.73205081 2.        ]]
Matrix multiplication:  [[14. 17.]
 [34. 43.]]
Sum of all elements of matrix x:  10.0
Sum of elements in each column of matrix y:  [10. 13.]
Sum of elements in each row of matrix y:  [15.  8.]
Transpose of matrix x:  [[1. 3.]
 [2. 4.]]
```

4.
```
import numpy as np
print("Example for broadcasting: ")
v=np.array([1,2,3])
w=np.array([4,5])
print("v = ",v)
print("w = ",w)
print("Outer product of above vectors: ")
print(np.reshape(v,(3,1))*w)
x=np.array([[1,2,3],[4,5,6]])
print("x = ",x)
print("Resultant matrix after adding the vector x to each row of matrix v: ")
print(x+v)
print("Example for broadcasting fails: ")
print("Adding the vector x to each column of matrix w will generate an error")
print("Solution: Reshape the matrix w then the result will be: ")
print(x+np.reshape(w,(2,1)))
```

OUTPUT

```
Example for broadcasting:
v =  [1 2 3]
w =  [4 5]
Outer product of above vectors:
[[ 4  5]
 [ 8 10]
 [12 15]]
x =  [[1 2 3]
 [4 5 6]]
Resultant matrix after adding the vector x to each row of matrix v:
[[2 4 6]
 [5 7 9]]
Example for broadcasting fails:
Adding the vector x to each column of matrix w will generate an error
Solution: Reshape the matrix w then the result will be:
[[ 5  6  7]
 [ 9 10 11]]
```

5.
```
from numpy import array
from scipy.linalg import svd
# define a matrix
A = array([[1, 2], [3, 4], [5, 6]])
print("A=",A)
print("Shape of array A: ",A.shape)
print("")
U, s, VT = svd (A)# SVD
print("U = ",U)
print("Shape of matrix U: ",U.shape)
print("")
print("Sigma (diagonal matrix), s = ",s)
print("Shape of matrix sigma : ",s.shape)
print("")
print("Transpose Matrix, VT = ",VT)
print("Shape of matrix VT: ",VT.shape)
```

OUTPUT

```
A= [[1 2]
 [3 4]
 [5 6]]
Shape of array A:  (3, 2)

U =  [[-0.2298477   0.88346102  0.40824829]
 [-0.52474482  0.24078249 -0.81649658]
 [-0.81964194 -0.40189603  0.40824829]]
Shape of matrix U:  (3, 3)

Sigma (diagonal matrix), s =  [9.52551809 0.51430058]
Shape of matrix sigma :  (2,)

Transpose Matrix, VT =  [[-0.61962948 -0.78489445]
 [-0.78489445  0.61962948]]
Shape of matrix VT:  (2, 2)
```

6.
```
#Reconstruct matrix from svd
from numpy import array,diag,dot,zeros
from scipy.linalg import svd
A = array([[1, 2], [3, 4], [5, 6]])
print("A = ",A)
print(A.shape)
U,s,VT = svd (A)
print("U = ",U)
print(U.shape)
print("s = ",s)
print(s.shape)
print("VT = ",VT)
print(VT.shape)
sigma=zeros((A.shape[0],A.shape[1]))
sigma[:A.shape[1],:A.shape[1]]=diag(s)
B=U.dot(sigma.dot(VT))
print("Reconstructed matrix: ",B)
```

OUTPUT

```
A =  [[1 2]
 [3 4]
 [5 6]]
(3, 2)
U =  [[-0.2298477   0.88346102  0.40824829]
 [-0.52474482  0.24078249 -0.81649658]
 [-0.81964194 -0.40189603  0.40824829]]
(3, 3)
s =  [9.52551809 0.51430058]
(2,)
VT =  [[-0.61962948 -0.78489445]
 [-0.78489445  0.61962948]]
(2, 2)
Reconstructed matrix:  [[1. 2.]
 [3. 4.]
 [5. 6.]]
```

7.
```
#svd for pseudoinverse
from numpy import array, diag, zeros
from scipy.linalg import pinv
from numpy.linalg import svd
A = array([[1, 2], [3, 4], [5, 6]])
print("A = ",A)
print("Pseudoinverse of matrix A calculated by function pinv is: ")
print(pinv(A))
d=1.0/s
D=zeros(A.shape)
D[:A.shape[1], :A.shape[1]]=diag(d)
B=VT.T.dot(D.T).dot(U.T)
print("Pseudoinverse of matrix A calculated by using svd is: ")
print(B)
```

OUTPUT

```
A =  [[1 2]
 [3 4]
 [5 6]]
Pseudoinverse of matrix A calculated by function pinv is:
[[-1.33333333 -0.33333333  0.66666667]
 [ 1.08333333  0.33333333 -0.41666667]]
Pseudoinverse of matrix A calculated by using svd is:
[[-1.33333333 -0.33333333  0.66666667]
 [ 1.08333333  0.33333333 -0.41666667]]
```

8.
```
#svd for dimensionality reduction
from numpy import array, diag,zeros
from sklearn.decomposition import TruncatedSVD
from numpy.linalg import svd
A = array([
[1,2,3,4,5,6,7,8,9,10],
[11,12,13,14,15,16,17,18,19,20], [21,22,23,24,25,26,27,28,29,30]])
print("A = ",A)
print("")
U,s,VT = svd(A)
sigma=zeros((A.shape[0],A.shape[1]))
sigma[:A.shape[0],:A.shape[0]]=diag(s)
sigma=sigma[:,:2]
VT=VT[:2,:]
T=U.dot(sigma)
T=A.dot(VT.T)
print("Transform of the original matrix using svd is: ")
print(T)
print("")
svd=TruncatedSVD(2)
svd.fit(A)
result=svd.transform(A)
print("Transformed version : ")
print(result)
```

OUTPUT

```
A =  [[ 1  2  3  4  5  6  7  8  9 10]
 [11 12 13 14 15 16 17 18 19 20]
 [21 22 23 24 25 26 27 28 29 30]]

Transform of the original matrix using svd is:
[[-18.52157747   6.47697214]
 [-49.81310011   1.91182038]
 [-81.10462276  -2.65333138]]

Transformed version :
[[18.52157747  6.47697214]
 [49.81310011  1.91182038]
 [81.10462276 -2.65333138]]
```

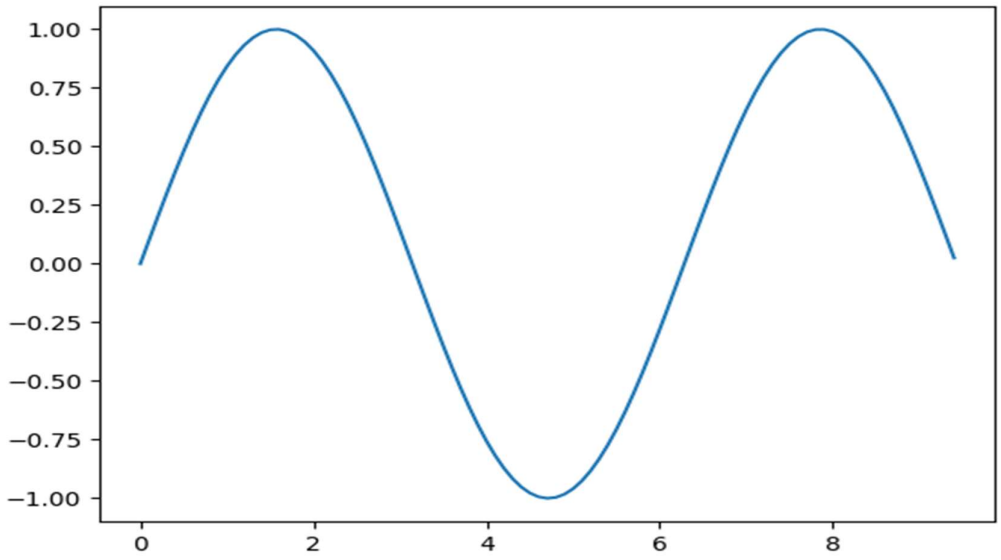
RESULT : The program has been executed successfully and output obtained.

PROGRAM NO:1. 3 DATE:

AIM : Programs using matplotlib/plotly/bokeh/seaborn for data visualization.

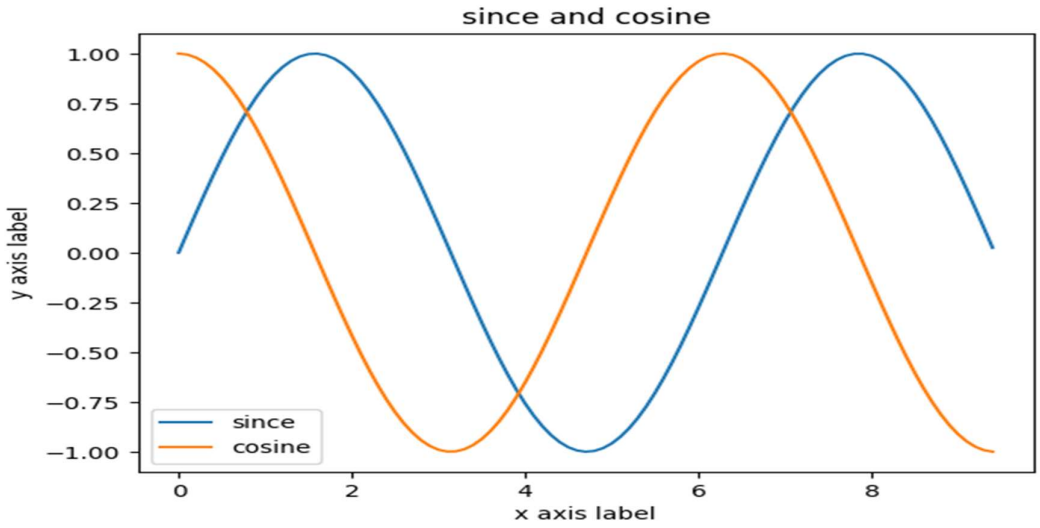
1.
```
import numpy as np
import matplotlib.pyplot as plt
x=np.arange (0,3*np.pi,0.1)
y=np.sin(x)
plt.plot(x,y)
plt.show()
```

OUTPUT



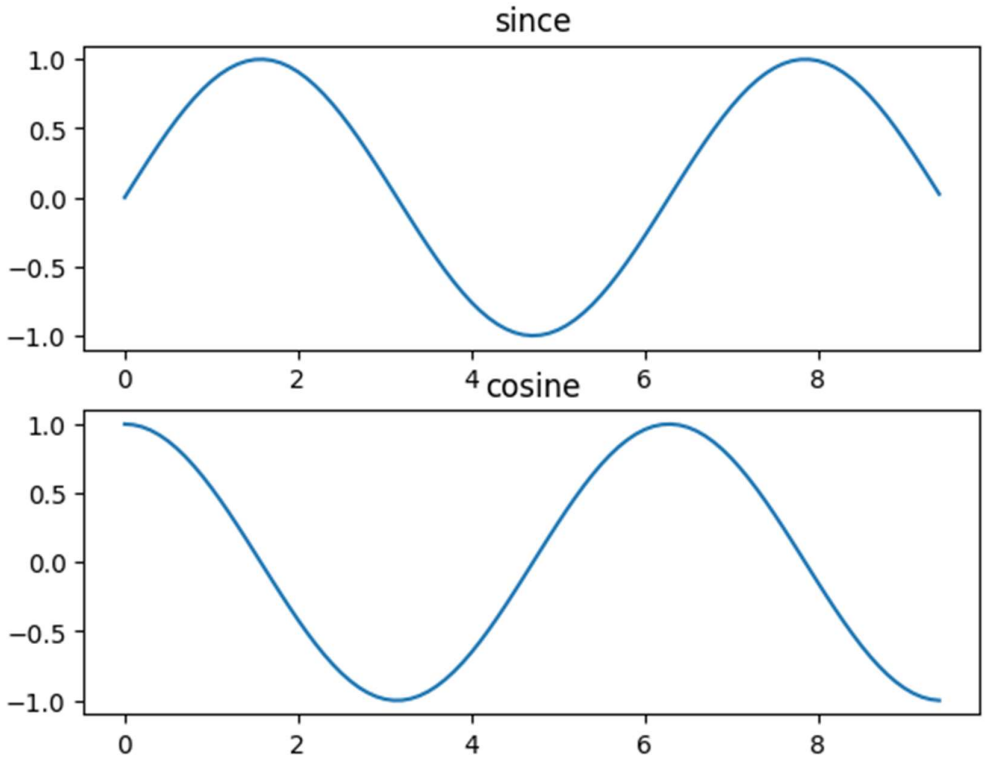
2.
```
import numpy as np
import matplotlib.pyplot as plt
x=np.arange(0,3*np.pi,0.1)
y_sin=np.sin(x)
y_cos=np.cos(x)
plt.plot(x,y_sin)
plt.plot(x,y_cos)
plt.xlabel('x axis label')
plt.ylabel('y axis label')
plt.title('since and cosine')
plt.legend(['since','cosine'])
plt.show()
```

OUTPUT



3. # subplots
   import numpy as np
   import matplotlib.pyplot as plt
   x=np.arange(0,3*np.pi,0.1)
   y_sin=np.sin(x)
   y_cos=np.cos(x)
   plt.subplot(2,1,1)
   plt.plot(x,y_sin)
   plt.title('since')
   plt.subplot(2,1,2)
   plt.plot(x,y_cos)
   plt.title('cosine')
   plt.show()

OUTPUT



4. #images
   import numpy as np
   import matplotlib.image as img
   import matplotlib.pyplot as plt
   tstimg = img.imread('/content/cat.jpeg')
   plt.subplot(1,2,1)
   plt.imshow(tstimg)
   plt.subplot(1,2,2)
   plt.imshow(tstimg)

OUTPUT

RESULT: The program has been executed successfully and output obtained.

PROGRAM NO: 1.4 DATE:

AIM: Programs to handle data using pandas.

1.
```
import pandas as pd
orders = pd.read_table('http://bit.ly/movieusers')
print("Overview of dataframe : ")
print(orders.head())

print("Shape : ",orders.shape)
print()

user_cols = ['user_id', 'age', 'gender', 'occupation', 'zip_code']
users = pd.read_table('http://bit.ly/movieusers', sep='|', header=None,
names=user_cols)

print("Dataframe after modifying the default parameter values for read_table: ")
print(users.head())
```

OUTPUT

```
Overview of dataframe :
   1|24|M|technician|85711
0       2|53|F|other|94043
1      3|23|M|writer|32067
2  4|24|M|technician|43537
3       5|33|F|other|15213
4   6|42|M|executive|98101
Shape :  (942, 1)

Dataframe after modifying the default parameter values for read_table:
   user_id  age gender  occupation zip_code
0        1   24      M  technician    85711
1        2   53      F       other    94043
2        3   23      M      writer    32067
3        4   24      M  technician    43537
4        5   33      F       other    15213
```

2. import pandas as pd #read a csv file

```
ufo = pd.read_csv('http://bit.ly/uforeports')
print("Overview of UFO data reports: ")
print(ufo.head())

print()

#series

print("Cityseries(sorted):")
print(ufo.City.sort_values())
print()

ufo['Location'] = ufo.City + ', ' + ufo.State
print("After creating a new 'Location' Series : ")
print(ufo.head())

print()

print("Calculate summary statistics : ")
print(ufo.describe())

print()

print("Column names of ufo dataframe : ",ufo.columns)
print()

# rename two of the columns by using the 'rename' method
ufo.rename(columns={'Colors Reported':'Colors_Reported', 'Shape
Reported':'Shape_Reported'},inplace=True)

print("Column name of ufo dataframe after renaming two column names :
",ufo.columns)

print()

# remove multiple columns at once ufo.drop(['City', 'State'], axis=1, inplace=True)

print("Column name of ufo dataframe after removing two columns(city,state) :
",ufo.columns)

print()

# remove multiple rows at once (axis=0 refers to rows)

ufo.drop([0, 1], axis=0, inplace=True)
```

```
print("ufo dataframe after deleting first two rows: ")
print(ufo.head())
```

OUTPUT

```
Overview of UFO data reports:
                City Colors Reported Shape Reported State            Time
0               Ithaca             NaN       TRIANGLE    NY   6/1/1930 22:00
1          Willingboro             NaN          OTHER    NJ  6/30/1930 20:00
2              Holyoke             NaN           OVAL    CO  2/15/1931 14:00
3              Abilene             NaN           DISK    KS   6/1/1931 13:00
4  New York Worlds Fair             NaN          LIGHT    NY  4/18/1933 19:00

Cityseries(sorted):
1761     Abbeville
4553      Aberdeen
16167     Aberdeen
14703     Aberdeen
389       Aberdeen
           ...
12441          NaN
15767          NaN
15812          NaN
16054          NaN
16608          NaN
Name: City, Length: 18241, dtype: object

After creating a new 'Location' Series :
                City Colors Reported Shape Reported State            Time  \
0               Ithaca             NaN       TRIANGLE    NY   6/1/1930 22:00
1          Willingboro             NaN          OTHER    NJ  6/30/1930 20:00
2              Holyoke             NaN           OVAL    CO  2/15/1931 14:00
3              Abilene             NaN           DISK    KS   6/1/1931 13:00
4  New York Worlds Fair             NaN          LIGHT    NY  4/18/1933 19:00

                Location
0               Ithaca, NY
1          Willingboro, NJ
2              Holyoke, CO
3              Abilene, KS
4  New York Worlds Fair, NY
```

```
Calculate summary statistics :
          City Colors Reported Shape Reported  State             Time  \
count     18216            2882          15597  18241            18241
unique     6476              27             27     52            16145
top     Seattle             RED          LIGHT     CA  11/16/1999 19:00
freq        187             780           2803   2529               27

          Location
count        18216
unique        8029
top     Seattle, WA
freq            187

Column names of ufo dataframe :  Index(['City', 'Colors Reported', 'Shape Reported', 'State', 'Time',
       'Location'],
      dtype='object')

Column name of ufo dataframe after renaming two column names :  Index(['City', 'Colors Reported', 'Shape Reported', 'State',
'Time',
       'Location'],
      dtype='object')

Column name of ufo dataframe after removing two columns(city,state) :  Index(['City', 'Colors Reported', 'Shape Reported', 'St
ate', 'Time',
       'Location'],
      dtype='object')

ufo dataframe after deleting first two rows:
                City Colors Reported Shape Reported State            Time  \
2              Holyoke             NaN           OVAL    CO  2/15/1931 14:00
3              Abilene             NaN           DISK    KS   6/1/1931 13:00
4  New York Worlds Fair             NaN          LIGHT    NY  4/18/1933 19:00
5           Valley City             NaN           DISK    ND  9/15/1934 15:30
6           Crater Lake             NaN         CIRCLE    CA   6/15/1935 0:00

                Location
2              Holyoke, CO
3              Abilene, KS
```

```
2                Holyoke, CO
3                Abilene, KS
4  New York Worlds Fair, NY
5            Valley City, ND
6            Crater Lake, CA
```

3. import pandas as pd

```
# read a dataset of top-rated IMDb movies into a DataFrame
movies = pd.read_csv('http://bit.ly/imdbratings')
print("Dataframe of top-rated IMDb movies: ")
print(movies.head())

print()

print("Different ways to filter rows of a pandas DataFrame by column value: ")
print("Example : Filter rows to only show movies with a duration of atleast 200
minutes")
print("1.using for loop")

# create a list in which each element refers to a DataFrame row: True if the row
satisfies the condition,False otherwise

booleans = []

for length in movies.duration:

 if length >= 200:

 booleans.append(True)

else:

booleans.append(False)

is_long = pd.Series(booleans)
print(is_long.head())

print()
print("2.broadcasting")

print(movies[movies.duration >= 200])
print()

print("3.using 'loc' method")
print(movies.loc[movies.duration >= 200])
```

OUTPUT

```
Dataframe of top-rated IMDb movies:
   star_rating                     title content_rating   genre  duration  \
0          9.3  The Shawshank Redemption              R   Crime       142
1          9.2             The Godfather              R   Crime       175
2          9.1    The Godfather: Part II              R   Crime       200
3          9.0           The Dark Knight          PG-13  Action       152
4          8.9              Pulp Fiction              R   Crime       154

                                      actors_list
0  [u'Tim Robbins', u'Morgan Freeman', u'Bob Gunt...
1    [u'Marlon Brando', u'Al Pacino', u'James Caan']
2  [u'Al Pacino', u'Robert De Niro', u'Robert Duv...
3  [u'Christian Bale', u'Heath Ledger', u'Aaron E...
4  [u'John Travolta', u'Uma Thurman', u'Samuel L....

Different ways to filter rows of a pandas DataFrame by column value:
Example : Filter rows to only show movies with a duration of atleast 200 minutes
1.using for loop
0    True
1    True
2    True
3    True
4    True
dtype: bool

2.broadcasting
     star_rating                                      title  \
2            9.1                     The Godfather: Part II
7            8.9  The Lord of the Rings: The Return of the King
17           8.7                                Seven Samurai
78           8.4              Once Upon a Time in America
85           8.4                        Lawrence of Arabia
142          8.3        Lagaan: Once Upon a Time in India
157          8.2                        Gone with the Wind
204          8.1                                   Ben-Hur
445          7.9                    The Ten Commandments
476          7.8                                   Hamlet



630          7.7                                 Malcolm X
767          7.6          It's a Mad, Mad, Mad, Mad World

    content_rating      genre  duration  \
2                R      Crime       200
7            PG-13  Adventure       201
17         UNRATED      Drama       207
78               R      Crime       229
85              PG  Adventure       216
142             PG  Adventure       224
157              G      Drama       238
204              G  Adventure       212
445       APPROVED  Adventure       220
476          PG-13      Drama       242
630          PG-13  Biography       202
767       APPROVED     Action       205

                                     actors_list
2    [u'Al Pacino', u'Robert De Niro', u'Robert Duv...
7    [u'Elijah Wood', u'Viggo Mortensen', u'Ian McK...
17   [u'Toshir\xf4 Mifune', u'Takashi Shimura', u'K...
78   [u'Robert De Niro', u'James Woods', u'Elizabet...
85   [u"Peter O'Toole", u'Alec Guinness', u'Anthony...
142  [u'Aamir Khan', u'Gracy Singh', u'Rachel Shell...
157  [u'Clark Gable', u'Vivien Leigh', u'Thomas Mit...
204  [u'Charlton Heston', u'Jack Hawkins', u'Stephe...
445  [u'Charlton Heston', u'Yul Brynner', u'Anne Ba...
476  [u'Kenneth Branagh', u'Julie Christie', u'Dere...
630  [u'Denzel Washington', u'Angela Bassett', u'De...
767  [u'Spencer Tracy', u'Milton Berle', u'Ethel Me...

3.using 'loc' method
     star_rating                                      title  \
2            9.1                     The Godfather: Part II
7            8.9  The Lord of the Rings: The Return of the King
17           8.7                                Seven Samurai
78           8.4              Once Upon a Time in America
```

```
85          8.4                          Lawrence of Arabia
142         8.3            Lagaan: Once Upon a Time in India
157         8.2                          Gone with the Wind
204         8.1                                    Ben-Hur
445         7.9                       The Ten Commandments
476         7.8                                    Hamlet
630         7.7                                 Malcolm X
767         7.6             It's a Mad, Mad, Mad, Mad World

    content_rating      genre  duration  \
2                R      Crime       200
7            PG-13  Adventure       201
17         UNRATED      Drama       207
78               R      Crime       229
85              PG  Adventure       216
142             PG  Adventure       224
157              G      Drama       238
204              G  Adventure       212
445       APPROVED  Adventure       220
476          PG-13      Drama       242
630          PG-13  Biography       202
767       APPROVED     Action       205

                                    actors_list
2    [u'Al Pacino', u'Robert De Niro', u'Robert Duv...
7    [u'Elijah Wood', u'Viggo Mortensen', u'Ian McK...
17   [u'Toshir\xf4 Mifune', u'Takashi Shimura', u'K...
78   [u'Robert De Niro', u'James Woods', u'Elizabet...
85   [u"Peter O'Toole", u'Alec Guinness', u'Anthony...
142  [u'Aamir Khan', u'Gracy Singh', u'Rachel Shell...
157  [u'Clark Gable', u'Vivien Leigh', u'Thomas Mit...
204  [u'Charlton Heston', u'Jack Hawkins', u'Stephe...
445  [u'Charlton Heston', u'Yul Brynner', u'Anne Ba...
476  [u'Kenneth Branagh', u'Julie Christie', u'Dere...
630  [u'Denzel Washington', u'Angela Bassett', u'De...
767  [u'Spencer Tracy', u'Milton Berle', u'Ethel Me...
```

3.using 'loc' method

```
     star_rating                                        title  \
2            9.1                        The Godfather: Part II
7            8.9  The Lord of the Rings: The Return of the King
17           8.7                                Seven Samurai
78           8.4                  Once Upon a Time in America
85           8.4                        Lawrence of Arabia
142          8.3            Lagaan: Once Upon a Time in India
157          8.2                          Gone with the Wind
204          8.1                                    Ben-Hur
445          7.9                       The Ten Commandments
476          7.8                                    Hamlet
630          7.7                                 Malcolm X
767          7.6             It's a Mad, Mad, Mad, Mad World

    content_rating      genre  duration  \
2                R      Crime       200
7            PG-13  Adventure       201
17         UNRATED      Drama       207
78               R      Crime       229
85              PG  Adventure       216
142             PG  Adventure       224
157              G      Drama       238
204              G  Adventure       212
445       APPROVED  Adventure       220
476          PG-13      Drama       242
630          PG-13  Biography       202
767       APPROVED     Action       205

                                    actors_list
2    [u'Al Pacino', u'Robert De Niro', u'Robert Duv...
7    [u'Elijah Wood', u'Viggo Mortensen', u'Ian McK...
17   [u'Toshir\xf4 Mifune', u'Takashi Shimura', u'K...
78   [u'Robert De Niro', u'James Woods', u'Elizabet...
85   [u"Peter O'Toole", u'Alec Guinness', u'Anthony...
142  [u'Aamir Khan', u'Gracy Singh', u'Rachel Shell...
157  [u'Clark Gable', u'Vivien Leigh', u'Thomas Mit...
```

```
157  [u'Clark Gable', u'Vivien Leigh', u'Thomas Mit...
204  [u'Charlton Heston', u'Jack Hawkins', u'Stephe...
445  [u'Charlton Heston', u'Yul Brynner', u'Anne Ba...
476  [u'Kenneth Branagh', u'Julie Christie', u'Dere...
630  [u'Denzel Washington', u'Angela Bassett', u'De...
767  [u'Spencer Tracy', u'Milton Berle', u'Ethel Me...
```

4. import pandas as pd

```
# read a dataset of Chipotle orders into a DataFrame
orders = pd.read_table('http://bit.ly/chiporders')
print("Dataframe : ")

print(orders.head())
print()

print("String methods in pandas: ")
print()

print("'item_name' series(in uppercase) : ")
print(orders.item_name.str.upper().head())
print()

print("Checks for a substring 'Chicken' in the given dataframe: ")
print(orders[orders.item_name.str.contains('Chicken')].head())
print()

# many pandas string methods support regular expressions (regex)
print(orders.choice_description.str.replace('[]', '').head())
print()

print("Examine the data type of each Series: ")
print(orders.dtypes)

print()

print("Dataframe after replacing '$' and converting string to float of 'item_price'
series: ")
print(orders.item_price.str.replace('$', '').astype(float))

print()
```

OUTPUT

```
Dataframe :
   order_id  quantity                          item_name  \
0         1         1              Chips and Fresh Tomato Salsa
1         1         1                                      Izze
2         1         1                         Nantucket Nectar
3         1         1  Chips and Tomatillo-Green Chili Salsa
4         2         2                           Chicken Bowl

                                choice_description item_price
0                                              NaN     $2.39
1                                    [Clementine]     $3.39
2                                        [Apple]      $3.39
3                                              NaN     $2.39
4  [Tomatillo-Red Chili Salsa (Hot), [Black Beans...    $16.98

String methods in pandas:

'item_name' series(in uppercase) :
0                CHIPS AND FRESH TOMATO SALSA
1                                        IZZE
2                           NANTUCKET NECTAR
3    CHIPS AND TOMATILLO-GREEN CHILI SALSA
4                             CHICKEN BOWL
Name: item_name, dtype: object

Checks for a substring 'Chicken' in the given dataframe:
    order_id  quantity           item_name  \
4         2         2        Chicken Bowl
5         3         1        Chicken Bowl
11        6         1  Chicken Crispy Tacos
12        6         1   Chicken Soft Tacos
13        7         1        Chicken Bowl

                                choice_description item_price
4   [Tomatillo-Red Chili Salsa (Hot), [Black Beans...    $16.98
5   [Fresh Tomato Salsa (Mild), [Rice, Cheese, Sou...    $10.98
11  [Roasted Chili Corn Salsa, [Fajita Vegetables,...     $8.75
12  [Roasted Chili Corn Salsa, [Rice, Black Beans,...     $8.75
13  [Fresh Tomato Salsa, [Fajita Vegetables, Rice,...    $11.25

0                                              NaN
1                                    [Clementine]
2                                        [Apple]
3                                              NaN
4   [Tomatillo-Red Chili Salsa (Hot), [Black Beans...
Name: choice_description, dtype: object
```

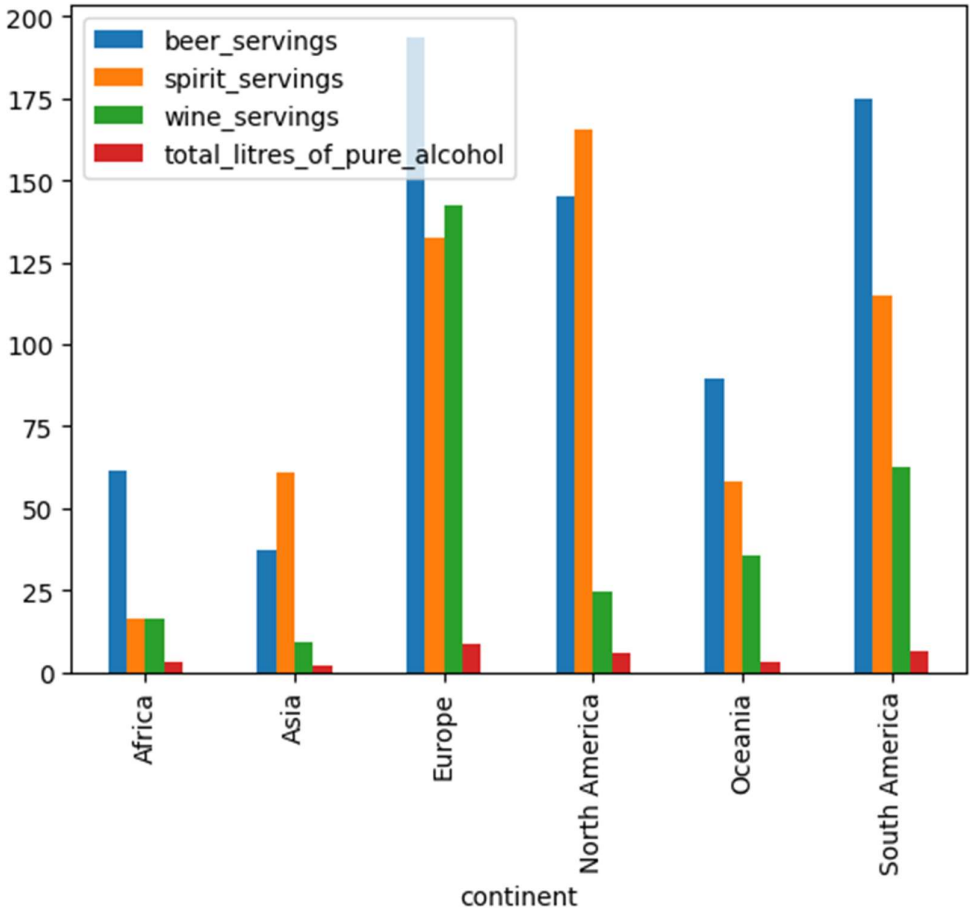
Examine the data type of each Series:
order_id int64
quantity int64
item_name object
choice_description object
item_price object
dtype: object

Dataframe after replacing '$' and converting string to float of 'item_price' series:
0 2.39
1 3.39
2 3.39
3 2.39
4 16.98
...
4617 11.75
4618 11.75
4619 11.25
4620 8.75
4621 8.75
Name: item_price, Length: 4622, dtype: float64

5.
```
import pandas as pd
#read a dataset of alcohol consumption into a DataFrame
drinks = pd.read_csv('http://bit.ly/drinksbycountry')
print("Dataframe : ")
print(drinks.head())
print()
print("Mean beer servings across the entire dataset: ",drinks.beer_servings.mean())
print("Mean beer servings just for countries in
Africa:",drinks[drinks.continent=='Africa'].beer_servings.mean())
print()
print("Aggregate functions used with groupby: ")
print()
print("Mean beer servings for each
continent:",drinks.groupby('continent').beer_servings.mean())
print("Maximum beer servings for each
continent:",drinks.groupby('continent').beer_servings.max())
print("Multiple aggregation functions can be applied simultaneously: ")
print(drinks.groupby('continent').beer_servings.agg(['count', 'mean', 'min', 'max']))
# specifying a column to which the aggregation function should be applied is
notrequired
drinks.groupby('continent').mean()
# allow plots to appear in the notebook
%matplotlib inline
# side-by-side bar plot of the DataFrame directly above
drinks.groupby('continent').mean().plot(kind='bar')
```

OUTPUT

```
Europe           193.777778
North America    145.434783
Oceania           89.687500
South America    175.083333
Name: beer_servings, dtype: float64
Maximum beer servings for each continent: continent
Africa           376
Asia             247
Europe           361
North America    285
Oceania          306
South America    333
Name: beer_servings, dtype: int64
Multiple aggregation functions can be applied simultaneously:
               count        mean  min  max
continent
Africa            53   61.471698    0  376
Asia              44   37.045455    0  247
Europe            45  193.777778    0  361
North America     23  145.434783    1  285
Oceania           16   89.687500    0  306
South America     12  175.083333   93  333
<ipython-input-2-bb9293c93057>:17: FutureWarning: The default value of numeric_only in
DataFrameGroupBy.mean is deprecated. In a future version, numeric_only will default to
False. Either specify numeric_only or select only columns which should be valid for the
function.
  drinks.groupby('continent').mean()
<ipython-input-2-bb9293c93057>:22: FutureWarning: The default value of numeric_only in
DataFrameGroupBy.mean is deprecated. In a future version, numeric_only will default to
False. Either specify numeric_only or select only columns which should be valid for the
function.
  drinks.groupby('continent').mean().plot(kind='bar')
<Axes: xlabel='continent'>
```


beer_servings
spirit_servings
wine_servings
total_litres_of_pure_alcohol
200
175
150
125
100
75
50
25
0
Africa
Asia
Europe
North America
Oceania
South America
continent

6.
```
import pandas as pd
ufo = pd.read_csv('http://bit.ly/uforeports')
print(ufo.isnull().tail())
print(ufo.notnull().tail())
print(ufo.isnull().sum())
print(ufo.shape)
# if 'all' values are missing in a row, then drop that row (none are dropped in this case)
print(ufo.dropna(how='all').shape)
print(ufo.dropna(subset=['City', 'Shape Reported'], how='any').shape)
print(ufo['Shape Reported'].value_counts().head())
# fill in missing values with a specified value
print(ufo['Shape Reported'].fillna(value='VARIOUS', inplace=True)) # confirm that
the missing values were filled in
print(ufo['Shape Reported'].value_counts().head())
drinks = pd.read_csv('http://bit.ly/drinksbycountry')
print(drinks.head())
# every DataFrame has an index (sometimes called the "row labels")
print(drinks.index)
# index and columns both default to integers if you don't define them
print(pd.read_table('http://bit.ly/movieusers', header=None, sep='|').head()) #
identification: index remains with each row when filtering the DataFrame
print(drinks[drinks.continent=='South America'])
# selection: select a portion of the DataFrame using the index
print(drinks.loc[23, 'beer_servings'])
# set an existing column as the index
print(drinks.set_index('country', inplace=True))
print(drinks.head())
# you can interact with any DataFrame using its index and columns
print(drinks.describe().loc['25%', 'beer_servings'])
# access the Series index
print(drinks.continent.value_counts().index) # access the Series values
print(drinks.continent.value_counts().values) # any Series can be sorted by its values
print(drinks.continent.value_counts().sort_values())
people = pd.Series([3000000, 85000], index=['Albania', 'Andorra'],
name='population') # concatenate the 'drinks' DataFrame with the 'population' Series
(aligns by the index)
print(pd.concat([drinks, people], axis=1).head()).
```

OUTPUT

```
City  Colors Reported  Shape Reported  State   Time
18236  False           True          False  False  False
18237  False           True          False  False  False
18238  False           True           True  False  False
18239  False          False          False  False  False
18240  False           True          False  False  False
      City  Colors Reported  Shape Reported  State  Time
18236  True          False          True   True  True
```

```
18237  True          False          True   True  True
18238  True          False         False   True  True
18239  True           True          True   True  True
18240  True          False          True   True  True
City               26
Colors Reported    15359
Shape Reported      2644
State                 0
Time                  0
dtype: int64
(18241, 5)
(18241, 5)
(15575, 5)
Shape Reported
LIGHT       2803
DISK        2122
TRIANGLE    1889
OTHER       1402
CIRCLE      1365
Name: count, dtype: int64
None
Shape Reported
VARIOUS     2977
LIGHT       2803
DISK        2122
TRIANGLE    1889
OTHER       1402
Name: count, dtype: int64
       country  beer_servings  spirit_servings  wine_servings  \
0  Afghanistan              0                0              0
1      Albania             89              132             54
2      Algeria             25                0             14
3      Andorra            245              138            312
4       Angola            217               57             45

   total_litres_of_pure_alcohol continent
0                           0.0      Asia
1                           4.9    Europe
2                           0.7    Africa
3                          12.4    Europe
4                           5.9    Africa
RangeIndex(start=0, stop=193, step=1)
   0   1  2          3      4
0  1  24  M  technician  85711
1  2  53  F       other  94043
2  3  23  M      writer  32067
3  4  24  M  technician  43537
4  5  33  F       other  15213
       country  beer_servings  spirit_servings  wine_servings  \
6    Argentina             193               25            221
```

```
20     Bolivia          167            41           8
23      Brazil          245           145          16
35       Chile          130           124         172
37    Colombia          159            76           3
52     Ecuador          162            74           3
72      Guyana           93           302           1
132   Paraguay          213           117          74
133       Peru          163           160          21
163   Suriname          128           178           7
185    Uruguay          115            35         220
188  Venezuela          333           100           3

    total_litres_of_pure_alcohol      continent
6                            8.3  South America
20                           3.8  South America
23                           7.2  South America
35                           7.6  South America
37                           4.2  South America
52                           4.2  South America
72                           7.1  South America
132                          7.3  South America
133                          6.1  South America
163                          5.6  South America
185                          6.6  South America
188                          7.7  South America
245
None
             beer_servings  spirit_servings  wine_servings  \
country
Afghanistan            0              0              0
Albania               89            132             54
Algeria               25              0             14
Andorra              245            138            312
Angola               217             57             45

             total_litres_of_pure_alcohol continent
country
Afghanistan                           0.0      Asia
Albania                               4.9    Europe
Algeria                               0.7    Africa
Andorra                              12.4    Europe
Angola                               5.9    Africa
20.0
Index(['Africa', 'Europe', 'Asia', 'North America', 'Oceania',
       'South America'],
      dtype='object', name='continent')
[53 45 44 23 16 12]
continent
South America    12
Oceania          16
```

```
North America    23
Asia             44
Europe           45
Africa           53
Name: count, dtype: int64
             beer_servings  spirit_servings  wine_servings  \
Afghanistan              0                0              0
Albania                 89              132             54
Algeria                 25                0             14
Andorra                245              138            312
Angola                 217               57             45

             total_litres_of_pure_alcohol continent  population
Afghanistan                           0.0      Asia         NaN
Albania                               4.9    Europe   3000000.0
Algeria                               0.7    Africa         NaN
Andorra                              12.4    Europe     85000.0
Angola                                5.9    Africa         NaN
```

7.
```
import pandas as pd
ufo = pd.read_csv('http://bit.ly/uforeports')
print("Dataframe: ")
print(ufo.head(3))
print()
print("Selecting multiple rows and columns from a pandas DataFrame using 'loc': ")
print()
#loc method is used to select rows and columns by label
print("First row, all columns: ")
print(ufo.loc[0, :])
print()
print("First 3 rows, all columns: ")
print(ufo.loc[[0, 1, 2], :])
print()
# rows 0 through 2 (inclusive), all columns
print(ufo.loc[0:2, :])
print()
# this implies "all columns", but explicitly stating "all columns" is better
print(ufo.loc[0:2])
print()
print("First 3 rows, only one column 'City': ")
print(ufo.loc[0:2, 'City'])
print()
print("First 3 rows, two columns 'City' and 'State': ")
print(ufo.loc[0:2, ['City', 'State']])
print()
print("Accomplish the same thing using double brackets: ")
#using 'loc' is preferred since it's more explicit
print(ufo[['City', 'State']].head(3))
print()
print("First 3 rows, columns 'City' through 'State': ")
print(ufo.loc[0:2, 'City':'State'])
print()
print("Accomplish the same thing using 'head' and 'drop': ")
print(ufo.head(3).drop('Time', axis=1))
print()
print("Rows in which the 'City' is 'Oakland', column 'State': ")
print(ufo.loc[ufo.City=='Oakland', 'State'])
print()
print("Accomplish the same thing using 'chained indexing': ")
#using 'loc' is preferred since chained indexing can cause problems
print(ufo[ufo.City=='Oakland'].State)
print()
print("Selecting multiple rows and columns from a pandas DataFrame using 'iloc': ")
print()
print("Rows in positions 0 and 1, columns in positions 0 and 3: ")
print(ufo.iloc[[0, 1], [0, 3]])
print()
```

```
print("Rows in positions 0 through 2 (exclusive), columns in positions 0 through 4(exclusive): ")
print(ufo.iloc[0:2, 0:4])
print()
print("Rows in positions 0 through 2 (exclusive), all columns: ")
print(ufo.iloc[0:2, :])
print()
```

OUTPUT

```
Dataframe:
        City Colors Reported Shape Reported State           Time
0     Ithaca             NaN       TRIANGLE    NY  6/1/1930 22:00
1  Willingboro           NaN          OTHER    NJ  6/30/1930 20:00
2     Holyoke            NaN           OVAL    CO  2/15/1931 14:00

Selecting multiple rows and columns from a pandas DataFrame using 'loc':

First row, all columns:
City                    Ithaca
Colors Reported            NaN
Shape Reported        TRIANGLE
State                       NY
Time            6/1/1930 22:00
Name: 0, dtype: object

First 3 rows, all columns:
        City Colors Reported Shape Reported State           Time
0     Ithaca             NaN       TRIANGLE    NY  6/1/1930 22:00
1  Willingboro           NaN          OTHER    NJ  6/30/1930 20:00
2     Holyoke            NaN           OVAL    CO  2/15/1931 14:00

        City Colors Reported Shape Reported State           Time
0     Ithaca             NaN       TRIANGLE    NY  6/1/1930 22:00
1  Willingboro           NaN          OTHER    NJ  6/30/1930 20:00
2     Holyoke            NaN           OVAL    CO  2/15/1931 14:00

        City Colors Reported Shape Reported State           Time
0     Ithaca             NaN       TRIANGLE    NY  6/1/1930 22:00
1  Willingboro           NaN          OTHER    NJ  6/30/1930 20:00
2     Holyoke            NaN           OVAL    CO  2/15/1931 14:00

First 3 rows, only one column 'City':
0        Ithaca
1    Willingboro
2       Holyoke
Name: City, dtype: object
```

First 3 rows, two columns 'City' and 'State':

	City	State
0	Ithaca	NY
1	Willingboro	NJ
2	Holyoke	CO

Accomplish the same thing using double brackets:

	City	State
0	Ithaca	NY
1	Willingboro	NJ
2	Holyoke	CO

First 3 rows, columns 'City' through 'State':

	City	Colors Reported	Shape Reported	State
0	Ithaca	NaN	TRIANGLE	NY
1	Willingboro	NaN	OTHER	NJ
2	Holyoke	NaN	OVAL	CO

Accomplish the same thing using 'head' and 'drop':

	City	Colors Reported	Shape Reported	State
0	Ithaca	NaN	TRIANGLE	NY
1	Willingboro	NaN	OTHER	NJ
2	Holyoke	NaN	OVAL	CO

Rows in which the 'City' is 'Oakland', column 'State':

```
1694    CA
2144    CA
4686    MD
7293    CA
8488    CA
8768    CA
10816   OR
10948   CA
11045   CA
12322   CA
12941   CA
16803   MD
17322   CA
Name: State, dtype: object
```

Accomplish the same thing using 'chained indexing':

```
1694    CA
2144    CA
4686    MD
7293    CA
8488    CA
8768    CA
10816   OR
10948   CA
11045   CA
```

```
12322    CA
12941    CA
16803    MD
17322    CA
Name: State, dtype: object
```

Selecting multiple rows and columns from a pandas DataFrame using 'iloc':

Rows in positions 0 and 1, columns in positions 0 and 3:

```
         City State
0       Ithaca    NY
1  Willingboro    NJ
```

Rows in positions 0 through 2 (exclusive), columns in positions 0 through 4(exclusive):

```
         City Colors Reported Shape Reported State
0       Ithaca            NaN       TRIANGLE    NY
1  Willingboro            NaN          OTHER    NJ
```

Rows in positions 0 through 2 (exclusive), all columns:

```
         City Colors Reported Shape Reported State            Time
0       Ithaca            NaN       TRIANGLE    NY   6/1/1930 22:00
1  Willingboro            NaN          OTHER    NJ  6/30/1930 20:00
```

8.
```
import pandas as pd
print("Creating dummy variables in pandas: ")
print()
# read the training dataset from Kaggle's Titanic competition
train = pd.read_csv('http://bit.ly/kaggletrain')
print("Dataframe: ")
print(train.head())
print()
#use 'get_dummies' to create one column for every possible value
print(pd.get_dummies(train.Sex).head())
print()
# drop the first dummy variable ('female') using the 'iloc' method
print(pd.get_dummies(train.Sex).iloc[:, 1:].head())
print()
# add a prefix to identify the source of the dummy variables
print(pd.get_dummies(train.Sex, prefix='Sex').iloc[:, 1:].head())
print()
# use 'get_dummies' with a feature that has 3 possible values
print(pd.get_dummies(train.Embarked, prefix='Embarked').head(10))
print()
# drop the first dummy variable ('C')
print(pd.get_dummies(train.Embarked, prefix='Embarked').iloc[:, 1:].head(10))
print()
#0, 0 means C 1, 0 means Q 0, 1 means S
# reset the DataFrame
train = pd.read_csv('http://bit.ly/kaggletrain')
print("Dataframe: ")
print(train.head())
print()
26# pass the DataFrame to 'get_dummies' and specify which columns to dummy (it
drops
#the original columns)
print(pd.get_dummies(train, columns=['Sex', 'Embarked']).head())
print()
# use the 'drop_first' parameter (new in pandas 0.18) to drop the first dummy variable
#for each feature
print(pd.get_dummies(train, columns=['Sex', 'Embarked'], drop_first=True).head())
```

OUTPUT

Creating dummy variables in pandas:

Dataframe:

	PassengerId	Survived	Pclass	\
0	1	0	3	
1	2	1	1	
2	3	1	3	

```
3          4       1      1
4          5       0      3

                                                Name     Sex   Age  SibSp  \
0                            Braund, Mr. Owen Harris    male  22.0      1
1  Cumings, Mrs. John Bradley (Florence Briggs Th...  female  38.0      1
2                             Heikkinen, Miss. Laina  female  26.0      0
3       Futrelle, Mrs. Jacques Heath (Lily May Peel)  female  35.0      1
4                           Allen, Mr. William Henry    male  35.0      0

   Parch            Ticket     Fare Cabin Embarked
0      0         A/5 21171   7.2500   NaN        S
1      0          PC 17599  71.2833   C85        C
2      0  STON/O2. 3101282   7.9250   NaN        S
3      0            113803  53.1000  C123        S
4      0            373450   8.0500   NaN        S

   female   male
0   False   True
1    True  False
2    True  False
3    True  False
4   False   True

    male
0   True
1  False
2  False
3  False
4   True

   Sex_male
0      True
1     False
2     False
3     False
4      True

   Embarked_C  Embarked_Q  Embarked_S
0       False       False        True
1        True       False       False
2       False       False        True
3       False       False        True
4       False       False        True
5       False        True       False
6       False       False        True
7       False       False        True
8       False       False        True
9        True       False       False
```

```
   Embarked_Q  Embarked_S
0       False        True
1       False       False
2       False        True
3       False        True
4       False        True
5        True       False
6       False        True
7       False        True
8       False        True
9       False       False

Dataframe:
   PassengerId  Survived  Pclass  \
0            1         0       3
1            2         1       1
2            3         1       3
3            4         1       1
4            5         0       3

                                                Name     Sex   Age  SibSp  \
0                            Braund, Mr. Owen Harris    male  22.0      1
1  Cumings, Mrs. John Bradley (Florence Briggs Th...  female  38.0      1
2                             Heikkinen, Miss. Laina  female  26.0      0
3       Futrelle, Mrs. Jacques Heath (Lily May Peel)  female  35.0      1
4                           Allen, Mr. William Henry    male  35.0      0

   Parch            Ticket     Fare Cabin Embarked
0      0         A/5 21171   7.2500   NaN        S
1      0          PC 17599  71.2833   C85        C
2      0  STON/O2. 3101282   7.9250   NaN        S
3      0            113803  53.1000  C123        S
4      0            373450   8.0500   NaN        S

   PassengerId  Survived  Pclass  \
0            1         0       3
1            2         1       1
2            3         1       3
3            4         1       1
4            5         0       3

                                                Name   Age  SibSp  Parch  \
0                            Braund, Mr. Owen Harris  22.0      1      0
1  Cumings, Mrs. John Bradley (Florence Briggs Th...  38.0      1      0
2                             Heikkinen, Miss. Laina  26.0      0      0
3       Futrelle, Mrs. Jacques Heath (Lily May Peel)  35.0      1      0
4                           Allen, Mr. William Henry  35.0      0      0

            Ticket     Fare Cabin  Sex_female  Sex_male  Embarked_C  \
0         A/5 21171   7.2500   NaN       False      True       False
```

```
1         PC 17599  71.2833   C85        True     False       True
2  STON/O2. 3101282   7.9250   NaN        True     False      False
3          113803  53.1000  C123        True     False      False
4          373450   8.0500   NaN       False      True      False

   Embarked_Q  Embarked_S
0       False        True
1       False       False
2       False        True
3       False        True
4       False        True

   PassengerId  Survived  Pclass  \
0            1         0       3
1            2         1       1
2            3         1       3
3            4         1       1
4            5         0       3

                                                Name   Age  SibSp  Parch  \
0                            Braund, Mr. Owen Harris  22.0      1      0
1  Cumings, Mrs. John Bradley (Florence Briggs Th...  38.0      1      0
2                             Heikkinen, Miss. Laina  26.0      0      0
3      Futrelle, Mrs. Jacques Heath (Lily May Peel)  35.0      1      0
4                           Allen, Mr. William Henry  35.0      0      0

          Ticket     Fare Cabin  Sex_male  Embarked_Q  Embarked_S
0       A/5 21171   7.2500   NaN      True       False        True
1        PC 17599  71.2833   C85     False       False       False
2  STON/O2. 3101282   7.9250   NaN     False       False        True
3          113803  53.1000  C123     False       False        True
4          373450   8.0500   NaN      True       False        True
```

9.
```
import pandas as pd
import numpy as np
# create a DataFrame from a dictionary (keys become column names, values become
#data) optionally specify the order of columns and define the index
df = pd.DataFrame({'id':[100, 101, 102], 'color':['red', 'blue', 'red']}, columns=['id',
'color'],index=['a', 'b', 'c'])
print("DataFrame from a dictionary: ")
print(df)
print()
# create a DataFrame from a list of lists (each inner list becomes a row)
print("DataFrame from a list of lists: ")
print(pd.DataFrame([[100, 'red'], [101, 'blue'], [102, 'red']], columns=['id', 'color']))
print()
# create a NumPy array (with shape 4 by 2) and fill it with random numbers
between0&1
arr = np.random.rand(4, 2)
print("Numpy array: ")
print(arr)
print()
print("DataFrame from the above defined NumPy array: ")
print(pd.DataFrame(arr, columns=['one', 'two']))
print()
print("DataFrame of student IDs (100 through 109) and test scores (random integers
between 60 and 100: ")
print(pd.DataFrame({'student':np.arange(100, 110, 1), 'test':np.random.randint(60,
101,10)}))
print()
# 'set_index' can be chained with the DataFrame constructor to select an index
print(pd.DataFrame({'student':np.arange(100, 110, 1),
'test':np.random.randint(60,101,10)}).set_index('student'))
print()
# create a new Series using the Series constructor
s = pd.Series(['round', 'square'], index=['c', 'b'], name='shape')
print(s)
print()
# concatenate the DataFrame and the Series (use axis=1 to concatenate columns)
print(pd.concat([df, s], axis=1))
```

OUTPUT

```
DataFrame from a dictionary:
   id color
a  100   red
b  101  blue
c  102   red
```

DataFrame from a list of lists:
```
   id color
0  100   red
1  101  blue
2  102   red
```

Numpy array:
```
[[0.5393908  0.90976723]
 [0.87398901 0.89512585]
 [0.31705152 0.82931947]
 [0.70089941 0.88028066]]
```

DataFrame from the above defined NumPy array:
```
        one       two
0  0.539391  0.909767
1  0.873989  0.895126
2  0.317052  0.829319
3  0.700899  0.880281
```

DataFrame of student IDs (100 through 109) and test scores (random integers between 60 and 100:
```
   student  test
0      100    61
1      101    62
2      102    69
3      103    73
4      104    92
5      105    76
6      106   100
7      107    95
8      108    88
9      109    67
```

```
         test
student
100        76
101        96
102        83
103        66
104        66
105        93
106        90
107        87
108        63
109        92
```

```
c     round
b    square
Name: shape, dtype: object
```

```
   id color   shape
a 100   red     NaN
b 101  blue  square
c 102   red   round
```

10.
```
import pandas as pd
    # change display options in pandas
    # read a dataset of alcohol consumption into a DataFrame
    drinks = pd.read_csv('http://bit.ly/drinksbycountry')
    print("Shape: ",drinks.shape)
    print()
    # check the current setting for the 'max_rows' option
    pd.get_option('display.max_rows')
    print(drinks)
    print()
    # overwrite the current setting so that all rows will be displayed
    pd.set_option('display.max_rows',2)
    print(drinks)
    print()
    # reset the 'max_rows' option to its default
    pd.reset_option('display.max_rows')
    print(drinks)
    print()
    # add two meaningless columns to the drinks DataFrame
    drinks['x'] = drinks.wine_servings * 1000
    drinks['y'] = drinks.total_litres_of_pure_alcohol * 1000
    print(drinks.head())
    print()
    31# use a Python format string to specify a comma as the thousands separator
    pd.set_option('display.float_format', '{:,}'.format)
    print(drinks.head())
    print()
    # read the training dataset from Kaggle's Titanic competition into a DataFrame
    train = pd.read_csv('http://bit.ly/kaggletrain')
    # an ellipsis is displayed in the 'Name' cell of row 1 because of the 'max_colwidth'
    option
    pd.get_option('display.max_colwidth')
    print(train.head())
    print()
    # overwrite the current setting so that more characters will be displayed
    pd.set_option('display.max_colwidth', 1000)
    print(train.head())
    print()
```

OUTPUT

```
Shape:  (193, 6)

       country  beer_servings  spirit_servings  wine_servings  \
0  Afghanistan              0                0              0
1      Albania             89              132             54
2      Algeria             25                0             14
```

```
3       Andorra          245           138          312
4        Angola          217            57           45
..          ...          ...           ...          ...
188    Venezuela          333           100            3
189     Vietnam           111             2            1
190       Yemen             6             0            0
191      Zambia            32            19            4
192    Zimbabwe            64            18            4

    total_litres_of_pure_alcohol      continent
0                             0.0          Asia
1                             4.9        Europe
2                             0.7        Africa
3                            12.4        Europe
4                             5.9        Africa
..                            ...           ...
188                           7.7  South America
189                           2.0          Asia
190                           0.1          Asia
191                           2.5        Africa
192                           4.7        Africa

[193 rows x 6 columns]

        country  beer_servings  spirit_servings  wine_servings  \
0   Afghanistan              0                0              0
..          ...            ...              ...            ...
192    Zimbabwe             64               18              4

    total_litres_of_pure_alcohol continent
0                             0.0      Asia
..                            ...       ...
192                           4.7    Africa

[193 rows x 6 columns]

        country  beer_servings  spirit_servings  wine_servings  \
0   Afghanistan              0                0              0
1       Albania             89              132             54
2       Algeria             25                0             14
3       Andorra            245              138            312
4        Angola            217               57             45
..          ...            ...              ...            ...
188    Venezuela            333              100              3
189     Vietnam             111                2              1
190       Yemen               6                0              0
191      Zambia              32               19              4
192    Zimbabwe              64               18              4

    total_litres_of_pure_alcohol      continent
```

```
0                    0.0         Asia
1                    4.9       Europe
2                    0.7       Africa
3                   12.4        Europe
4                    5.9       Africa
..                   ...          ...
188                  7.7  South America
189                  2.0         Asia
190                  0.1         Asia
191                  2.5       Africa
192                  4.7       Africa

[193 rows x 6 columns]

      country  beer_servings  spirit_servings  wine_servings  \
0  Afghanistan              0                0              0
1      Albania             89              132             54
2      Algeria             25                0             14
3      Andorra            245              138            312
4       Angola            217               57             45

   total_litres_of_pure_alcohol continent      x       y
0                           0.0      Asia      0     0.0
1                           4.9    Europe   54000   4900.0
2                           0.7    Africa   14000    700.0
3                          12.4    Europe  312000  12400.0
4                           5.9    Africa   45000   5900.0

      country  beer_servings  spirit_servings  wine_servings  \
0  Afghanistan              0                0              0
1      Albania             89              132             54
2      Algeria             25                0             14
3      Andorra            245              138            312
4       Angola            217               57             45

   total_litres_of_pure_alcohol continent      x       y
0                           0.0      Asia      0     0.0
1                           4.9    Europe   54000  4,900.0
2                           0.7    Africa   14000    700.0
3                          12.4    Europe  312000 12,400.0
4                           5.9    Africa   45000  5,900.0

   PassengerId  Survived  Pclass  \
0            1         0       3
1            2         1       1
2            3         1       3
3            4         1       1
4            5         0       3

                                                Name     Sex  Age  SibSp  \
```

```
0                            Braund, Mr. Owen Harris    male 22.0      1
1  Cumings, Mrs. John Bradley (Florence Briggs Th...  female 38.0      1
2                             Heikkinen, Miss. Laina  female 26.0      0
3       Futrelle, Mrs. Jacques Heath (Lily May Peel)  female 35.0      1
4                           Allen, Mr. William Henry    male 35.0      0

   Parch            Ticket    Fare Cabin Embarked
0      0         A/5 21171    7.25   NaN        S
1      0          PC 17599 71.2833   C85        C
2      0  STON/O2. 3101282   7.925   NaN        S
3      0            113803    53.1  C123        S
4      0            373450    8.05   NaN        S

   PassengerId  Survived  Pclass  \
0            1         0       3
1            2         1       1
2            3         1       3
3            4         1       1
4            5         0       3

                                                Name     Sex  Age  SibSp  \
0                            Braund, Mr. Owen Harris    male 22.0      1
1  Cumings, Mrs. John Bradley (Florence Briggs Thayer)  female 38.0      1
2                             Heikkinen, Miss. Laina  female 26.0      0
3       Futrelle, Mrs. Jacques Heath (Lily May Peel)  female 35.0      1
4                           Allen, Mr. William Henry    male 35.0      0

   Parch            Ticket    Fare Cabin Embarked
0      0         A/5 21171    7.25   NaN        S
1      0          PC 17599 71.2833   C85        C
2      0  STON/O2. 3101282   7.925   NaN        S
3      0            113803    53.1  C123        S
4      0            373450    8.05   NaN        S
```

11.
```
import pandas as pd
# read a dataset of UFO reports into a DataFrame
print("'inplace'parameter in pandas: ")
print()
ufo = pd.read_csv('http://bit.ly/uforeports')
print("Dataframe: ")
print(ufo.head())
print("Shape : ",ufo.shape)
print()
# remove the 'City' column (doesn't affect the DataFrame since inplace=False)
ufo.drop('City', axis=1)
# confirm that the 'City' column was not actually removed
print(ufo.columns)
print()
# remove the 'City' column (does affect the DataFrame since inplace=True)
ufo.drop('City', axis=1, inplace=True)
# confirm that the 'City' column was actually removed
print(ufo.columns)
print()
print(ufo.shape)
print()
34#drop a row if any value is missing from that row (doesn't affect the DataFrame
since
#inplace=False)
ufo.dropna(how='any')
# confirm that no rows were actually removed
print(ufo.shape)
print()
print("Using an assignment statement instead of the 'inplace' parameter: ")
ufo = ufo.set_index('Time')
print(ufo.tail(3))
print()
print("Fill missing values using 'backward fill' strategy: ")
# doesn't affect the DataFrame since inplace=False
print(ufo.fillna(method='bfill').tail(3))
print()
print("Dataframe: ")
print(ufo.tail(3))
print()
print("Fill missing values using 'forward fill' strategy: ")
#doesn't affect the DataFrame since inplace=False
print(ufo.fillna(method='ffill').tail(3))
print()
print("Dataframe: ")
print(ufo.tail(3))
```

OUTPUT

'inplace'parameter in pandas:

Dataframe:

```
              City Colors Reported Shape Reported State            Time
0           Ithaca            NaN       TRIANGLE    NY   6/1/1930 22:00
1       Willingboro            NaN          OTHER    NJ  6/30/1930 20:00
2          Holyoke            NaN           OVAL    CO  2/15/1931 14:00
3          Abilene            NaN           DISK    KS   6/1/1931 13:00
4  New York Worlds Fair            NaN          LIGHT    NY  4/18/1933 19:00
Shape :  (18241, 5)
```

Index(['City', 'Colors Reported', 'Shape Reported', 'State', 'Time'], dtype='object')

Index(['Colors Reported', 'Shape Reported', 'State', 'Time'], dtype='object')

(18241, 4)

(18241, 4)

Using an assignment statement instead of the 'inplace' parameter:

```
                    Colors Reported Shape Reported State
Time
12/31/2000 23:45             NaN           NaN    WI
12/31/2000 23:45             RED         LIGHT    WI
12/31/2000 23:59             NaN          OVAL    FL
```

Fill missing values using 'backward fill' strategy:

```
                    Colors Reported Shape Reported State
Time
12/31/2000 23:45             RED         LIGHT    WI
12/31/2000 23:45             RED         LIGHT    WI
12/31/2000 23:59             NaN          OVAL    FL
```

Dataframe:

```
                    Colors Reported Shape Reported State
Time
12/31/2000 23:45             NaN           NaN    WI
12/31/2000 23:45             RED         LIGHT    WI
12/31/2000 23:59             NaN          OVAL    FL
```

Fill missing values using 'forward fill' strategy:

```
                    Colors Reported Shape Reported State
Time
12/31/2000 23:45             RED          DISK    WI
12/31/2000 23:45             RED         LIGHT    WI
12/31/2000 23:59             RED          OVAL    FL
```

Dataframe:

Time	Colors Reported	Shape Reported	State
12/31/2000 23:45	NaN	NaN	WI
12/31/2000 23:45	RED	LIGHT	WI
12/31/2000 23:59	NaN	OVAL	FL

1

RESULT: The program has been executed successfully and output obtained.

LABCYCLE :2

PROGRAM NO:5 DATE:

AIM: Implementation of KNN classification algorithm

```
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
from sklearn import metrics
iris=load_iris()
x=iris.data
y=iris.target
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.3,random_state=1)
c_knn=KNeighborsClassifier(n_neighbors=3)
c_knn.fit(x_train,y_train)
y_pred=c_knn.predict(x_test)
print("Accuracy:",metrics.accuracy_score(y_test,y_pred))
sample=[[2,2,2,2]]
pred=c_knn.predict(sample)
pred_v=[iris.target_names[p] for p in pred]
print(pred_v)
```

OUTPUT

```
Accuracy: 0.9777777777777777
['setosa']
```

PROGRAM NO:6 DATE:

AIM: Implementation of Naïve Bayes classification algorithm

```
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import GaussianNB
X,y=load_iris(return_X_y=True)
X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.5,random_state=0)
gnb=GaussianNB()
y_pred=gnb.fit(X_train,y_train).predict(X_test)
print(y_pred)
x_new=[[5,5,4,4]]
y_new=gnb.fit(X_train,y_train).predict(x_new)
print("predicted output for [[5,5,4,4]]:",y_new)
print("Naive bayes score:",gnb.score(X_test,y_test))
```

OUTPUT

```
[2 1 0 2 0 2 0 1 1 1 1 1 1 1 1 0 1 1 0 0 2 1 0 0 2 0 0 1 1 0 2 1 0 2 2 1 0
 1 1 1 2 0 2 0 0 1 2 2 1 2 1 2 1 1 2 1 1 2 1 2 1 0 2 1 1 1 1 2 0 0 2 1 0 0
 1]
predicted output for [[5,5,4,4]]: [2]
Naive bayes score: 0.9466666666666667
```

PROGRAM NO:7 DATE:

AIM: Implementation of Decision Tree classification algorithm

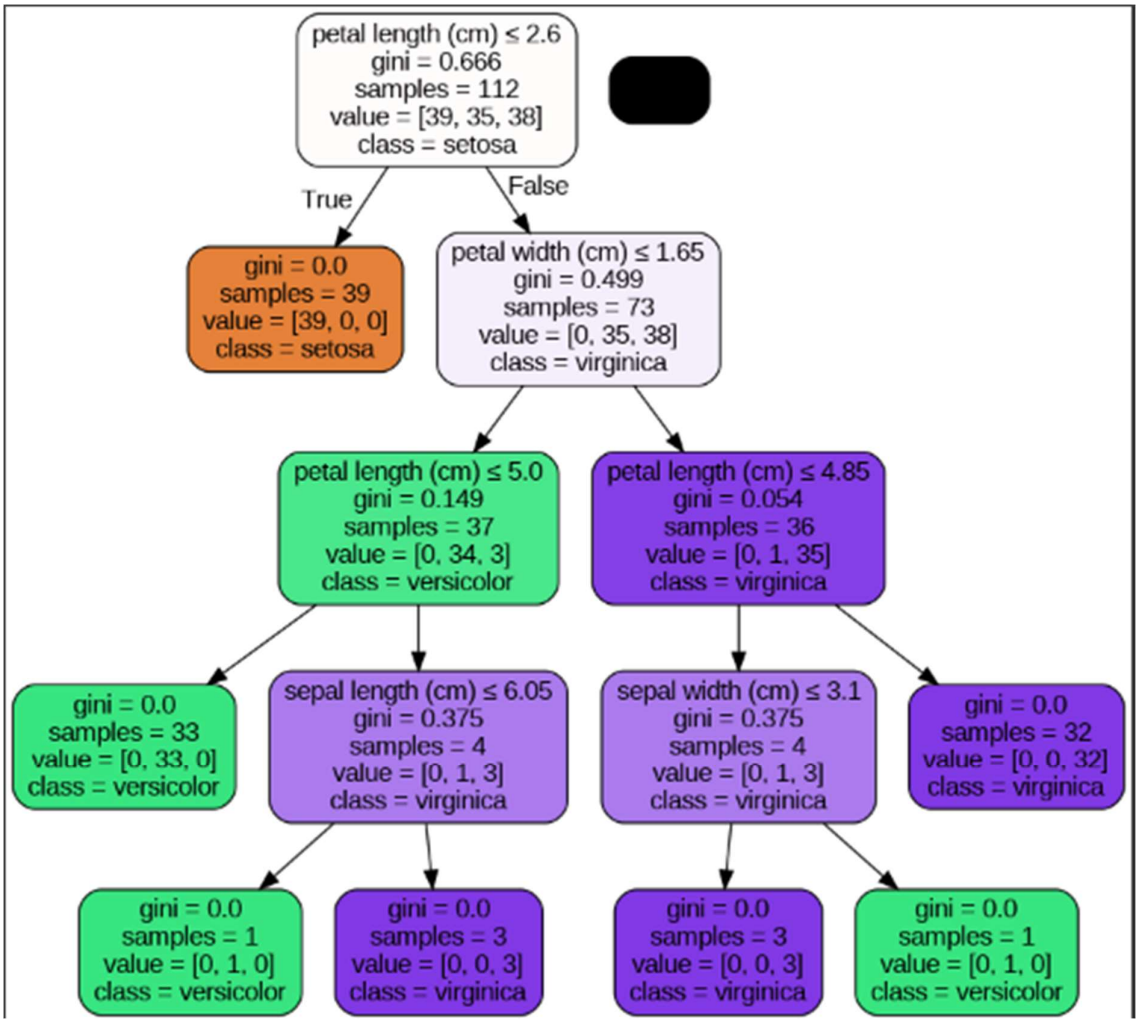
```
import pandas as pd
import numpy as np
from sklearn.datasets import load_iris
#load iris data
data = load_iris()
data.data.shape
print('classes to predict: ',data.target_names)
print('Features: ',data.feature_names)
X = data.data
y = data.target
display (X.shape, y.shape)
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
X_train, X_test, y_train, y_test = train_test_split(X, y,random_state = 50, test_size = 0.25)
#default criterion is Gini
classifier = DecisionTreeClassifier()
classifier.fit(X_train, y_train)
y_pred = classifier.predict(X_test)
from sklearn.metrics import accuracy_score
print('Accuracy on train data using Gini: ',accuracy_score(y_true = y_train, y_pred =
classifier.predict(X_train)))
```

```
print('Accuracy on test data using Gini: ',accuracy_score(y_true = y_test, y_pred = y_pred))

#change criterion to entropy

classifier_entropy = DecisionTreeClassifier(criterion='entropy')

classifier_entropy.fit(X_train, y_train)

y_pred_entropy = classifier_entropy.predict(X_test)

print('Accuracy on train data using entropy', accuracy_score(y_true=y_train, y_pred =
classifier_entropy.predict(X_train)))

print('Accuracy on test data using entropy', accuracy_score(y_true=y_test, y_pred =
y_pred_entropy))

#change criterion to entropy with min_samples_split to 50. Default value is 2

classifier_entropy1 = DecisionTreeClassifier(criterion='entropy', min_samples_split=50)

classifier_entropy1.fit(X_train, y_train)

y_pred_entropy1 = classifier_entropy1.predict(X_test)

print('Accuracy on train data using entropy', accuracy_score(y_true=y_train, y_pred =
classifier_entropy1.predict(X_train)))

print('Accuracy on test data using entropy', accuracy_score(y_true=y_test, y_pred =
y_pred_entropy1))

#visualise the decision tree

from sklearn.tree import export_graphviz

from six import StringIO

from IPython.display import Image

import pydotplus

dot_data = StringIO()

#the students can try using classifier, classifier_entropy and classifier_entropy1
#as first parameter below.
export_graphviz(classifier, out_file = dot_data,filled = True, rounded =
True,special_characters = True, feature_names = data.feature_names, class_names =
data.target_names)
```

```
graph = pydotplus.graph_from_dot_data(dot_data.getvalue())

Image(graph.create_png())
```

OUTPUT

```
classes to predict:  ['setosa' 'versicolor' 'virginica']
Features:  ['sepal length (cm)', 'sepal width (cm)', 'petal length (cm)', 'petal width (cm)']
(150, 4)
(150,)
Accuracy on train data using Gini:  1.0
Accuracy on test data using Gini:  0.9473684210526315
Accuracy on train data using entropy 1.0
Accuracy on test data using entropy 0.9473684210526315
Accuracy on train data using entropy 0.9642857142857143
Accuracy on test data using entropy 0.9473684210526315
```



RESULT: The program has been executed successfully and output obtained.

LABCYCLE :3

PROGRAM NO:8 DATE:

AIM: Implementation of Linear Regression techniques

Load sklearn Libraries:

```
#import libraries
import matplotlib.pyplot as plt
import numpy as np
from sklearn import datasets, linear_model
from sklearn.metrics import mean_squared_error, r2_score
```

Load Data

```
# Load the diabetes dataset
diabetes_X, diabetes_y = datasets.load_diabetes(return_X_y=True)
```

Split Dataset

```
# Use only one feature
diabetes_X = diabetes_X[:, np.newaxis, 2]
# Split the data into training/testing sets
diabetes_X_train = diabetes_X[:-20]
diabetes_X_test = diabetes_X[-20:]
# Split the targets into training/testing sets
diabetes_y_train = diabetes_y[:-20]
diabetes_y_test = diabetes_y[-20:]
```

Creating Model

```
# Create linear regression object
regr = linear_model.LinearRegression()
# Train the model using the training sets
regr.fit(diabetes_X_train, diabetes_y_train)
```

Make Prediction

```
# Make predictions using the testing set
```

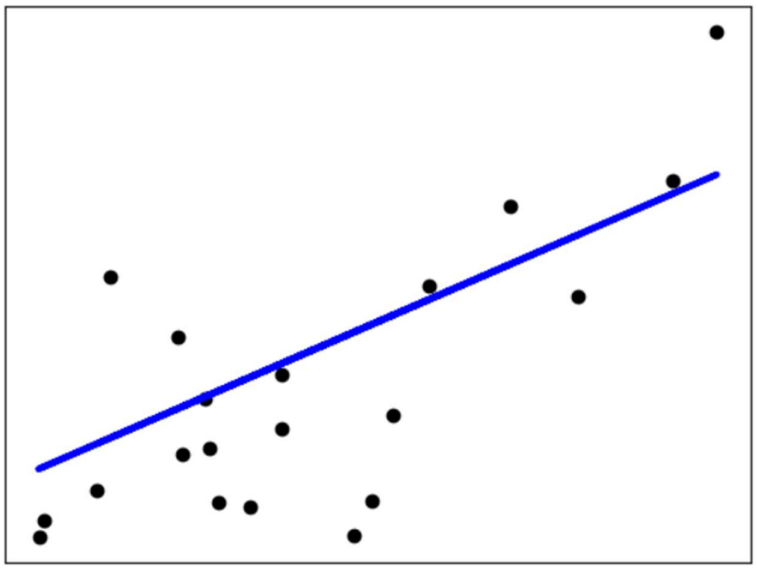
```
diabetes_y_pred = regr.predict(diabetes_X_test)
```

Finding Coefficient And Mean Square Error

```
# The coefficients
print('Coefficients: \n', regr.coef_)
# The mean squared error
print('Mean squared error: %.2f'
      % mean_squared_error(diabetes_y_test, diabetes_y_pred))
# The coefficient of determination: 1 is perfect prediction
print('Coefficient of determination: %.2f'
      % r2_score(diabetes_y_test, diabetes_y_pred))
plt.scatter(diabetes_X_test, diabetes_y_test,  color='black')
plt.plot(diabetes_X_test, diabetes_y_pred, color='blue', linewidth=3)
plt.xticks(())
plt.yticks(())
plt.show()
```

OUTPUT

```
Coefficients:
 [938.23786125]
Mean squared error: 2548.07
Coefficient of determination: 0.47
```



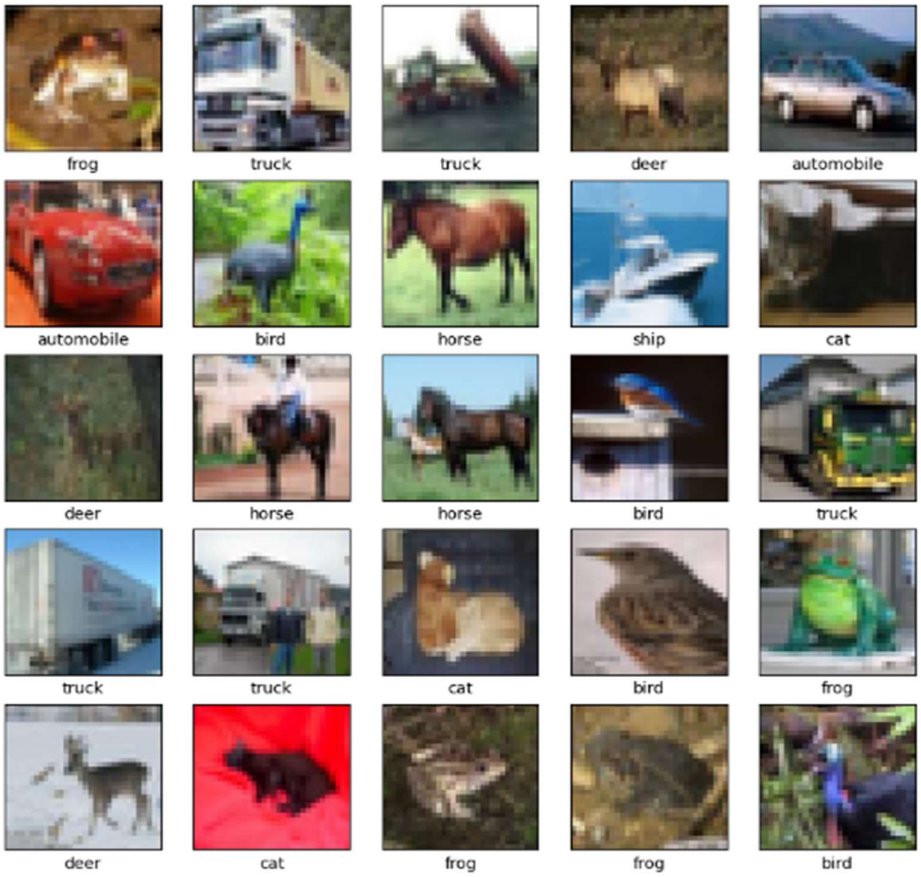
PROGRAM NO:9 DATE:

AIM: Image classification using convolutional neural networks

```
#CNN
import tensorflow as tf
from keras import datasets, layers, models
import matplotlib.pyplot as plt
(train_images, train_labels), (test_images, test_labels) = datasets.cifar10.load_data()
# Normalize pixel values to be between 0 and 1
train_images, test_images = train_images / 255.0, test_images / 255.0
class_names = ['airplane', 'automobile', 'bird', 'cat', 'deer','dog', 'frog', 'horse', 'ship', 'truck']
plt.figure(figsize=(10,10))
for i in range(25):
  plt.subplot(5,5,i+1)
plt.xticks([])
plt.yticks([])
plt.grid(False)
plt.imshow(train_images[i])
plt.xlabel(class_names[train_labels[i][0]])
print("Given dataset : ")
import tensorflow
from tensorflow import keras
plt.show()
model = models.Sequential()
model.add(layers.Conv2D(32, (3, 3), activation='relu', input_shape=(32, 32, 3)))
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Conv2D(64, (3, 3), activation='relu'))
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Conv2D(64, (3, 3), activation='relu'))
print("Architecture of the model: ")
model.summary()
```

```
#Adding dense layer
model.add(layers.Flatten())
model.add(layers.Dense(64, activation='relu'))
model.add(layers.Dense(10))
print("Architecture of the modal : ")
model.summary()
model.compile(optimizer='adam',loss=tf.keras.losses.SparseCategoricalCrossentropy(
 from_logits=True),metrics=['accuracy'])
history = model.fit(train_images, train_labels, epochs=10, validation_data=(test_images,
test_labels))
test_loss, test_acc = model.evaluate(test_images, test_labels, verbose=2)
print("Test accuracy : ",test_acc)
```

OUTPUT



```
Architecture of the model:
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_3 (Conv2D)           (None, 30, 30, 32)        896

 max_pooling2d_2 (MaxPoolin  (None, 15, 15, 32)        0
 g2D)

 conv2d_4 (Conv2D)           (None, 13, 13, 64)        18496

 max_pooling2d_3 (MaxPoolin  (None, 6, 6, 64)          0
 g2D)

 conv2d_5 (Conv2D)           (None, 4, 4, 64)          36928

=================================================================
Total params: 56320 (220.00 KB)
Trainable params: 56320 (220.00 KB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

```
Architecture of the modal :
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_3 (Conv2D)           (None, 30, 30, 32)        896

 max_pooling2d_2 (MaxPoolin  (None, 15, 15, 32)        0
 g2D)

 conv2d_4 (Conv2D)           (None, 13, 13, 64)        18496

 max_pooling2d_3 (MaxPoolin  (None, 6, 6, 64)          0
 g2D)

 conv2d_5 (Conv2D)           (None, 4, 4, 64)          36928

 flatten_1 (Flatten)         (None, 1024)              0

 dense_2 (Dense)             (None, 64)                65600

 dense_3 (Dense)             (None, 10)                650

=================================================================
Total params: 122570 (478.79 KB)
Trainable params: 122570 (478.79 KB)
Non-trainable params: 0 (0.00 Byte)
```

```
Epoch 1/10
1563/1563 [==============================] - 46s 28ms/step - loss: 1.5562 - accuracy: 0.4336 - val_loss: 1.3301 - val_accuracy:
0.5234
Epoch 2/10
1563/1563 [==============================] - 45s 28ms/step - loss: 1.1934 - accuracy: 0.5758 - val_loss: 1.0833 - val_accuracy:
0.6151
Epoch 3/10
1563/1563 [==============================] - 45s 29ms/step - loss: 1.0332 - accuracy: 0.6372 - val_loss: 1.0674 - val_accuracy:
0.6279
Epoch 4/10
1563/1563 [==============================] - 46s 29ms/step - loss: 0.9266 - accuracy: 0.6756 - val_loss: 0.9515 - val_accuracy:
0.6665
Epoch 5/10
1563/1563 [==============================] - 45s 29ms/step - loss: 0.8499 - accuracy: 0.7019 - val_loss: 0.9020 - val_accuracy:
0.6829
Epoch 6/10
1563/1563 [==============================] - 45s 29ms/step - loss: 0.7931 - accuracy: 0.7238 - val_loss: 0.8996 - val_accuracy:
0.6874
Epoch 7/10
1563/1563 [==============================] - 48s 31ms/step - loss: 0.7391 - accuracy: 0.7396 - val_loss: 0.9072 - val_accuracy:
0.6896
Epoch 8/10
1563/1563 [==============================] - 50s 32ms/step - loss: 0.6983 - accuracy: 0.7552 - val_loss: 0.8461 - val_accuracy:
0.7113
Epoch 9/10
1563/1563 [==============================] - 48s 31ms/step - loss: 0.6603 - accuracy: 0.7682 - val_loss: 0.8772 - val_accuracy:
0.7001
Epoch 10/10
1563/1563 [==============================] - 44s 28ms/step - loss: 0.6229 - accuracy: 0.7806 - val_loss: 0.8802 - val_accuracy:
0.7155
```

```
313/313 - 3s - loss: 0.8802 - accuracy: 0.7155 - 3s/epoch - 9ms/step
Test accuracy :  0.715499997138977
```

RESULT: The program has been executed successfully and output obtained.

LABCYCLE :4

PROGRAM NO:10 DATE:

AIM: Implementation of Part of Speech tagging, N-gram, smoothening and Chunking using NLTK.

```
import nltk
nltk.download('punkt')
#Tokenizing
from nltk.tokenize import *
text="""A newspaper is the strongest medium for news. People are reading newspapers for decades. It has a huge contribution to globalization. Right now because of easy internet
connection, people don’t read printed newspapers often. They read the online version."""
print("Sample text : \n ",text,"\n")
sent_tokenized=sent_tokenize(text)
print("Tokenizing by sentence : \n",sent_tokenized,"\n")
word_tokenized=word_tokenize(text)
print("Tokenizing by word : \n ",word_tokenized,"\n")
#Filtering stop words
import nltk
nltk.download('stopwords')
from nltk.corpus import stopwords
from string import punctuation
stopwords=stopwords.words('english')
punctuation=list(punctuation)
print("After filtering the stop words and punctuation : ")
for word in word_tokenized:
    if word.casefold() not in stopwords and word.casefold() not in punctuation:
        print(word)
#new_list=[word for word in word_tokenized if word.casefold() not in stopwords and word not in punctuation]
#print(new_list,"\n")
#Stemming
```

```
from nltk.stem import PorterStemmer
ps = PorterStemmer()
words = ["reading", "globalization", "Being","Went","gone","going"]
print("Given words : ",words)
stemm=[ps.stem(i) for i in words ]
print("After stemming : ",stemm,"\n")
#Lemmatization
from nltk.stem import WordNetLemmatizer
import nltk
nltk.download('wordnet')
nltk.download('omw-1.4')
lem= WordNetLemmatizer()
print("rocks :", lem.lemmatize("rocks"))
print("corpora :", lem.lemmatize("corpora"))
print("better :", lem.lemmatize("better"))
print("believes :", lem.lemmatize("believes"),"\n")
print("better :", lem.lemmatize("went",pos="a"))
print("better :", lem.lemmatize("went",pos="v"))
print("better :", lem.lemmatize("went",pos="n"),"\n")
nltk.download('averaged_perceptron_tagger')
nltk.download('maxent_ne_chunker')
nltk.download('words')
from nltk import RegexpParser
from nltk.tree import *
#POS Tag
postag=nltk.pos_tag(word_tokenized)
print("POS tagging : \n")
for i in postag:
   print(i)
#Chunking
```

```
print("\n")
grammar = "NP: {<DT>?<JJ>*<NN>}"
chunker = RegexpParser(grammar)
output = chunker.parse(postag)
print("After Chunking:\n",output)
output.pretty_print()
```

OUTPUT

```
Sample text :
  A newspaper is the strongest medium for news. People are reading newspapers
for decades. It has a huge contribution to globalization. Right now because of easy
internet connection, people don’t read printed newspapers often. They read the online
version.

Tokenizing by sentence :
 ['A newspaper is the strongest medium for news.', 'People are reading newspapers\nfor decades.', 'It has a huge contribution t
o globalization.', 'Right now because of easy\ninternet connection, people don’t read printed newspapers often.', 'They read th
e online\nversion.']

Tokenizing by word :
  ['A', 'newspaper', 'is', 'the', 'strongest', 'medium', 'for', 'news', '.', 'People', 'are', 'reading', 'newspapers', 'for',
'decades', '.', 'It', 'has', 'a', 'huge', 'contribution', 'to', 'globalization', '.', 'Right', 'now', 'because', 'of', 'easy',
'internet', 'connection', ',', 'people', 'don', '’', 't', 'read', 'printed', 'newspapers', 'often', '.', 'They', 'read', 'the',
'online', 'version', '.']

[nltk_data] Downloading package punkt to
[nltk_data]     C:\Users\user\AppData\Roaming\nltk_data...
[nltk_data]   Package punkt is already up-to-date!
```

```
After filtering the stop words and punctuation :
newspaper
strongest
medium
news
People
reading
newspapers
decades
huge
contribution
globalization
Right
easy
internet
connection
people
’
read
printed
newspapers
often
read
online
version
```

```
[nltk_data] Downloading package stopwords to
[nltk_data]     C:\Users\user\AppData\Roaming\nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
```

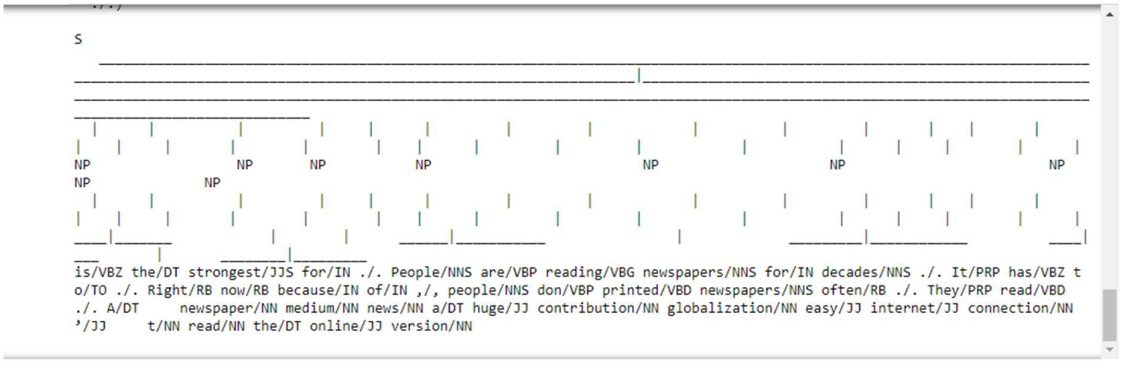
```
Given words :  ['reading', 'globalization', 'Being', 'Went', 'gone', 'going']
After stemming :  ['read', 'global', 'be', 'went', 'gone', 'go']
```

```
[nltk_data] Downloading package wordnet to
[nltk_data]     C:\Users\user\AppData\Roaming\nltk_data...
[nltk_data]   Package wordnet is already up-to-date!
[nltk_data] Downloading package omw-1.4 to
[nltk_data]     C:\Users\user\AppData\Roaming\nltk_data...
[nltk_data]   Package omw-1.4 is already up-to-date!
```

```
rocks : rock
corpora : corpus
better : better
believes : belief

better : went
better : go
better : went
```

```
[nltk_data] Downloading package averaged_perceptron_tagger to
[nltk_data]     C:\Users\user\AppData\Roaming\nltk_data...
[nltk_data]   Package averaged_perceptron_tagger is already up-to-
[nltk_data]       date!
[nltk_data] Downloading package maxent_ne_chunker to
[nltk_data]     C:\Users\user\AppData\Roaming\nltk_data...
[nltk_data]   Package maxent_ne_chunker is already up-to-date!
[nltk_data] Downloading package words to
[nltk_data]     C:\Users\user\AppData\Roaming\nltk_data...
[nltk_data]   Package words is already up-to-date!
```

```
POS tagging :

('A', 'DT')
('newspaper', 'NN')
('is', 'VBZ')
('the', 'DT')
('strongest', 'JJS')
('medium', 'NN')
('for', 'IN')
```



PROGRAM NO:11 DATE:

AIM: Text classification using support vector machines

```
from sklearn.model_selection import train_test_split
from sklearn import datasets
from sklearn import svm
from sklearn import metrics
cancer=datasets.load_breast_cancer()
x_train,x_test,y_train,y_test=train_test_split(cancer.data,cancer.target,test_size=0.3,random_state=109)
clf=svm.SVC(kernel='linear')
clf.fit (x_train,y_train)
y_pred=clf.predict(x_test)
print("actual values",y_test)
print("predicted values",y_pred)
print("accuracy",metrics.accuracy_score(y_test,y_pred))
print("precision",metrics.precision_score(y_test,y_pred))
print("recall",metrics.recall_score(y_test,y_pred))
```

OUTPUT

```
actual values [1 1 0 0 1 0 1 1 1 0 0 0 1 0 1 1 0 0 1 0 1 1 0 0 1 1 0 1
1 1 1 0 1 1 1 1 1
 0 1 1 0 1 0 1 1 1 0 1 0 0 1 1 0 1 1 0 1 1 1 0 0 1 0 1 0 0 1 1 1 1 0 1
1 1
 0 1 0 1 1 1 1 1 1 1 0 1 1 1 1 1 0 0 1 0 1 1 1 1 1 0 0 1 1 0 1 1 0 1 0
1 1
 0 1 1 0 0 0 1 0 1 1 1 1 1 0 1 0 1 0 1 0 0 0 0 0 0 1 1 0 1 1 1 0 1 1 0 0
1 0
 0 0 1 1 1 1 1 0 0 1 0 1 1 1 1 1 1 1 1 0 1 1 1]
predicted values [1 1 0 0 1 0 1 1 1 0 0 0 1 0 0 1 0 0 1 0 1 1 0 0 1 1 0
1 1 1 1 0 1 1 1 1 1
 0 1 1 0 1 0 1 1 1 1 1 0 0 1 1 0 1 1 0 1 1 1 0 0 1 0 1 0 0 1 1 1 1 0 1
1 1
 0 1 0 1 1 1 1 1 1 1 0 0 1 1 1 1 0 0 0 0 1 1 1 1 1 0 0 1 0 0 1 1 0 1 0
1 1
 0 1 1 0 0 0 1 0 1 1 1 1 1 0 1 0 1 0 1 0 0 0 1 0 1 1 0 1 1 1 0 1 1 0 0
1 0
 0 0 1 1 1 1 1 0 0 1 0 1 1 1 1 1 1 1 1 0 1 1 1]
accuracy 0.9649122807017544
precision 0.9811320754716981
recall 0.9629629629629629
```

PROGRAM NO:12 DATE:

AIM: Implementation of a simple web-crawler and scrapping web pages

```
# Imports
import requests
import numpy as np
import pandas as pd
from bs4 import BeautifulSoup
import matplotlib.pyplot as plt
import re
import os
%matplotlib inline
riturl="https://purdue.edu"
webpage = requests.get(riturl)
ritsoup = BeautifulSoup(webpage.content, "html.parser")
print(ritsoup)

print("Title of the parsed page : ",ritsoup.title)
print()
print("All the links : ")
links = [link.get('href') for link in ritsoup.find_all('a')]
print(links,"\n")
print("Accessing elements of the parsed page : ")
print("Accessing heading element : ",ritsoup.h1)
print(ritsoup.head)
print(ritsoup.head.meta)
paragraphs = ritsoup.find_all("p")
print()
print("Get all <p> elements : ",paragraphs)
```

```
print()
print("Gets all the p elements with a class attribute with value hide: ",ritsoup.find_all("p",
attrs={"class": "hide"}))
print()
print("Obtaining strings : ",ritsoup.h1.string)
print()
#The contents method is similar but always returns a list:
print("Obtaining strings using contents method : ", ritsoup.h1.contents)
print()
#If the element contains any tags, then string will return None
print(paragraphs[2],"\n")
print(paragraphs[2].string,"\n")
#However, contents will return a list as before, mixing different kinds of elements:
para2 = paragraphs[2].contents
print(para2,"\n")
para7=paragraphs[5].contents
print(para7,"\n")
#You can also use stripped_strings, which is a generator over all the strings (tags
#removed)
#inside the element; this is a fast way to extract the raw texts, with all tag soup strained
#off:
for s in paragraphs[3].stripped_strings:
    print("="*50)
    print(s)
```

OUTPUT

```
<!DOCTYPE html>

<html class="is-fullheight" lang="en-US">
<head>
<title>Purdue University</title>
<meta charset="utf-8"/>
<meta content="width=device-width, initial-scale=1" name="viewport"/>
<link href="http://gmpg.org/xfn/11" rel="profile"/>
<style type="text/css">@media (min-width:1024px){
.purdue-home-news-events__title{
min-height:3.5rem;
}
}</style>
<!-- Google Tag Manager for WordPress by gtm4wp.com -->
<script data-cfasync="false" data-pagespeed-no-defer="">
	var gtm4wp_datalayer_name = "dataLayer";
	var dataLayer = dataLayer || [];
</script>
\.edu\/home\/wp-includes\/js\/wp-emoji-release.min.js?ver=6.3.2"}};
/*! This file is auto-generated */
if("object"==typeof e&&"number"==typeof e.timestamp&&(new Date).valueOf
()<e.timestamp+604800&&"object"==typeof e.supportTests)return e.):e.wpe
moji&&e.twemoji&&(t(e.twemoji),t(e.wpemoji)))})}((window,document),win
dow._wpemojiSettings);
</script>
<style type="text/css">
img.wp-smiley,
img.emoji {
	display: inline !important;
	border: none !important;
	box-shadow: none !important;
	height: 1em !important;
	width: 1em !important;
	margin: 0 0.07em !important;
	vertical-align: -0.1em !important;
	background: none !important;
	padding: 0 !important;
}
</style>


<link href="https://www.purdue.edu/home/wp-includes/css/dist/block-libr
ary/common.min.css?ver=6.3.2" id="wp-block-library-css" media="all" rel
="stylesheet" type="text/css"/>
<link href="https://www.purdue.edu/home/wp-includes/css/classic-themes.
min.css?ver=6.3.2" id="classic-theme-styles-css" media="all" rel="style
sheet" type="text/css"/>
<link href="https://use.typekit.net/ghc8hdz.css?ver=6.3.2" id="brandfon
ts-css" media="all" rel="stylesheet" type="text/css"/>
<link href="https://fonts.googleapis.com/css2?family=Source+Serif+Pro%3
Awght%40400%3B600%3B700&amp;display=swap&amp;ver=6.3.2" id="sourceserif
-css" media="all" rel="stylesheet" type="text/css"/>
```

```
<link href="https://www.purdue.edu/home/wp-content/plugins/wp-accessibi
lity/css/wpa-style.css?ver=2.0.1" id="wpa-style-css" media="all" rel="s
tylesheet" type="text/css"/>
<style id="wpa-style-inline-css" type="text/css">
:root { --admin-bar-top : 7px; }
</style>
<link href="https://use.fontawesome.com/releases/v6.4.2/css/all.css?ver
=6.3.2" id="load-fa-css" media="all" rel="stylesheet" type="text/css"/>
<link href="https://www.purdue.edu/home/wp-content/themes/purdue-home-t
heme/style.css?ver=6.3.2" id="purdueBrand-style-css" media="all" rel="s
tylesheet" type="text/css"/>
<link href="https://www.purdue.edu/home/wp-content/themes/purdue-home-t
heme/build/app.css?ver=e90763d4084803de069e" id="purdueBrand-brand-styl
e-css" media="" rel="stylesheet" type="text/css"/>


Gets all the p elements with a class attribute with value hide:  []

Obtaining strings :  Every Giant Leap Starts with One Small Step

Obtaining strings using contents method :  ['Every Giant Leap Starts wi
th One Small Step']

<p class="purdue-home-cta-card__cta-text">Give the Gift of Purdue</p>

Give the Gift of Purdue

['Give the Gift of Purdue']

['See the Boilermakers — long known as the “Cradle of Quarterbacks” — t
ake on other Big Ten Conference teams.']

================================================
Ross-Ade Stadium
```

RESULT: The program has been executed successfully and output obtained.