LAB CYCLE: 1 Date:

PROGRAM: 1.1

AIM: Review of python programming.

```
1. print("Demo of basic datatypes:Numbers")
   x=3
   y=2.5
   print("x=",x)
   print("y=",y)
   print("Datatype of variable x:",type(x))
   print("Datatype of variable y:",type(y))
   print("Addition:",x+y)
   print("Subtraction:",x-y)
   print("Multiplication:",x*2)
   print("Exponentation:",x**2)
```

OUTPUT:

```
Demo of basic datatypes:Numbers
x= 3
y= 2.5
Datatype of variable x: <class 'int'>
Datatype of variable y: <class 'float'>
Addition: 5.5
Subtraction: 0.5
Multiplication: 6
Exponentation: 9
```

2.
```
print("Demo of basic datatypes:Boolean")
t=True
f=False
print("t=",t)
print("f=",f)
print("Datatype of variable t:",type(t))
print("Datatype of variable f:",type(f))
print("Logical AND operation:",t and f)
print("Logical OR operation:",t or f)
print("Logical NOT operation:",not t)
print("Logical XOR operation:",t != f)
```

OUTPUT:

```
Demo of basic datatypes:Boolean
t= True
f= False
Datatype of variable t: <class 'bool'>
Datatype of variable f: <class 'bool'>
Logical AND operation: False
Logical OR operation: True
Logical NOT operation: False
Logical XOR operation: True
```

3.
```
print("Demo of basic datatypes:String")
s="hello"
t="world"
print("string1=",s)
print("string2=",t)
d=s+" "+t
print("String concatenation:",d)
print("Capitalize:",d.capitalize())
print("Convrted to uppercase:",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:

```
Demo of basic datatypes:String
string1= hello
string2= world
String concatenation: hello world
Capitalize: Hello world
Convrted to uppercase: HELLO
Right justify a string:   hello
String at center:  hello
After replacing l with ell: he(ell)(ell)o
String after striping leading and trailing white spaces: world
```

4.
```
print("Containers:Lists")
nums=list(range(5))
print("list 'nums' contains:",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:",nums)
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 assign 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 squares of even numbers from 'nums':",even_squares)
```

OUTPUT:

```
Containers:Lists
list 'nums' contains: [0, 1, 2, 3, 4]
List can contain elements of different types.Example: [0, 1, 2, 3, 'abc']
'nums'after inserting new element at the end: [0, 1, 2, 3, 'abc', 'xyz']
Sublists:
A slice from index 2 to 4: [2, 3]
A slice from index 2 to the end: [2, 3, 'abc', 'xyz']
A slice from the start to index 2: [0, 1]
A slice of the whole list: [0, 1, 2, 3, 'abc', 'xyz']
After assign a new sublist to 'nums':
1:0
2:1
3:2
4:3
5:8
6:9
List of squares of even numbers from 'nums': [0, 4, 64]
```

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

```
Containers:Dictionaries
Dictionary: {'cat': 'cute', 'dog': 'furry'}
Is the dictionary has the key 'cat'? True
After adding new entry to 'd': {'cat': 'cute', 'dog': 'furry', 'fish': 'wet'}
Get an element monkey: N/A
Get an element fish: wet
After deleting the newly added entry from 'd': {'cat': 'cute', 'dog': 'furry'}
Demo of dictionary comprehension:
Squares of integers of range 10:
0 : 0
1 : 1
2 : 4
3 : 9
4 : 16
5 : 25
6 : 36
7 : 49
8 : 64
9 : 81
```

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 & 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:

```
Containers:Sets
Set 'num1': {120, 100, 110}
'num1' after inserting 90: {120, 90, 100, 110}
'num1' after inserting multiple elements: {100, 70, 110, 50, 120, 90, 60}
'num1' after removing 60: {100, 70, 110, 50, 120, 90}
Set comprehension & Set operations:
n1= {0, 1, 2, 3, 4, 5, 6, 7, 8, 9}
n2= {1, 3, 5, 7, 9}
n1 union n2 {0, 1, 2, 3, 4, 5, 6, 7, 8, 9}
n1 intersection n2 {1, 3, 5, 7, 9}
n1 difference n2 {0, 2, 4, 6, 8}
```

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

OUTPUT:

```
Containers:Tuples
Dictionary with tuple keys:
(0, 1) : 0
(1, 2) : 1
(2, 3) : 2
(3, 4) : 3
(4, 5) : 4
(5, 6) : 5
(6, 7) : 6
(7, 8) : 7
(8, 9) : 8
(9, 10) : 9
Tuple t: (5, 6)
5
1
```

8.
```
print("Demo of function: Program 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:

```
Demo of function: Program to find factorial of a number
Enter a number: 5
Factorial: 120
```

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:

```
Hello, Fred
HELLO, FRED!
```

RESULT: The program was executed successfully and the output was obtained.

LAB CYCLE: 1 Date:

PROGRAM: 1.2

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

1.
```
import numpy as np
a = np.array([1, 2, 3])
print("One dimensional array a =", a)
b = np.array([[1, 2, 3], [4, 5, 6]])
print("Two dimensional array b =\n", b)
print("Size of the array a:", a.shape)
print("Element at indices 0, 1, 2:", a[0], a[1], a[2])
a[0] = 5
print("Array after changing the element at index 0:", a)
a = np.zeros((2, 2))
print("An array of all zeroes:\n", a)
b = np.ones((1, 2))
print("An array of all ones:\n", b)
c = np.full((2, 2), 7)
print("A constant array:\n", c)
d = np.eye(2)
print("A 2x2 identity matrix:\n", d)
e = np.random.random((2, 2))
print("An array with random values:\n", 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 zeroes:
 [[0. 0.]
 [0. 0.]]
An array of all ones:
 [[1. 1.]]
A constant array:
 [[7 7]
 [7 7]]
A 2x2 identity matrix:
 [[1. 0.]
 [0. 1.]]
An array with random values: [[0.33366232 0.47710687]
 [0.15905316 0.41523252]]
```

2.
```
print("Array indexing: slicing")
a1 = np.array([[1,2,3,4],[5,6,7,8],[9,10,11,12]])
print("a1=\n",a1)
b = a1[:2,1:3]
print("Subarray consisting of first two rows and columns 1 and
2:\n",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=\n",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],a2[0,1]]))
a3 = 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 eleement 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]
[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.
```
x = np.array([[1,2],[3,4]],dtype=np.float64)
y = np.array([[6,9],[4,4]],dtype=np.float64)
print("x=\n",x)
print("y=\n",y)
print("Element wise addition:\n",np.add(x,y))
print("Element wise subtraction:\n",np.subtract(x,y))
print("Element wise multiplication:\n",np.multiply(x,y))
print("Element wise square root of x:\n",np.sqrt(x))
print("Matrix multiplication:\n",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:\n",x.T)
```

OUTPUT:

```
x=
 [[1. 2.]
 [3. 4.]]
y=
 [[6. 9.]
 [4. 4.]]
Element wise addition:
 [[ 7. 11.]
 [ 7.  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. 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 v to each row of
   matrix x:")
   print(x+v)
   print("Example for broadcasting fails:-")
   print("Adding the vector w to each column of matrix x will generate
   an error")
   print("Solution:Reshape the vector w,then the reult 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]]
[[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 reult 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("Shape of matrix A:",A.shape)
print("")
U,s,VT = svd(A)
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)
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:\n",B)
```

OUTPUT:

```
A= [[1 2]
 [3 4]
 [5 6]]
Shape of matrix 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)
Reconstructed matrix:
 [[1. 2.]
 [3. 4.]
 [5. 6.]]
```

7. #svd for pseudoinverse

```
from numpy import array,diag,zeros
from numpy.linalg import svd
from scipy.linalg import pinv

A = array([[1,2],[3,4],[5,6]])
print("A=",A)
print("Pseudoinverse of matrix A calculated by function pinv is:")
print(pinv(A))

U,s,VT = svd(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]]
```

RESULT: The program was executed successfully and the output was obtained.

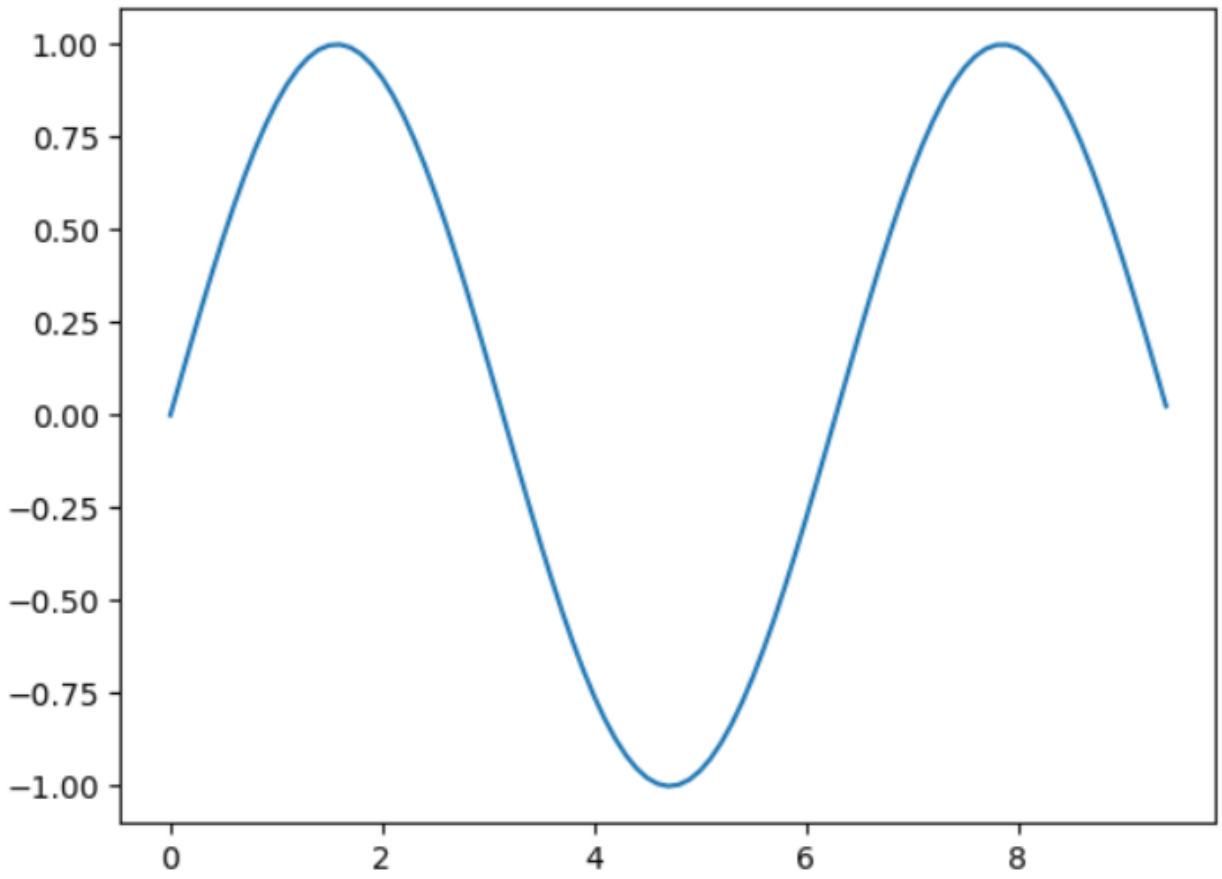
LAB CYCLE: 1 Date:

PROGRAM: 1.3

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

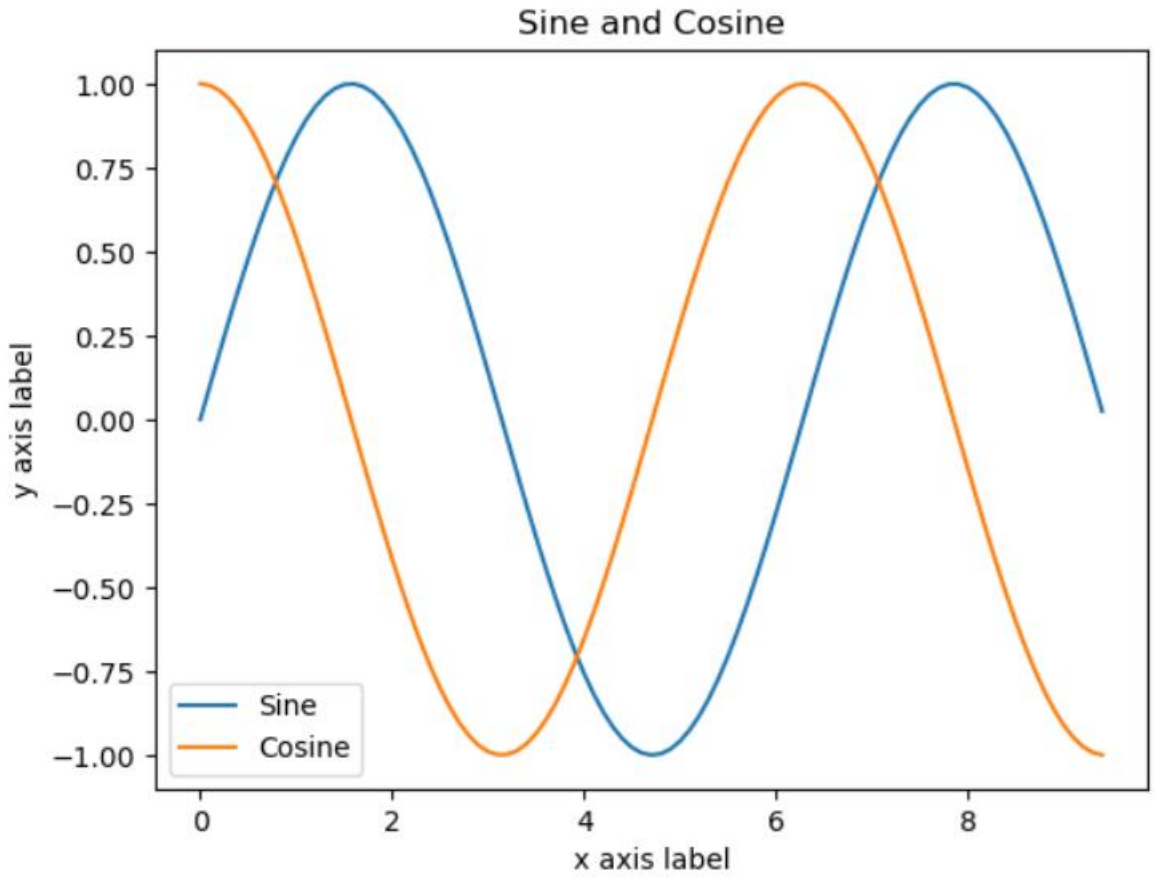
1.
```
import numpy as np
import matplotlib.pyplot as plt

x = np.arange(0,3*np.pi,0.1)
y = np.sin(x)
plt.plot(x,y)
plt.show()
```

OUTPUT:



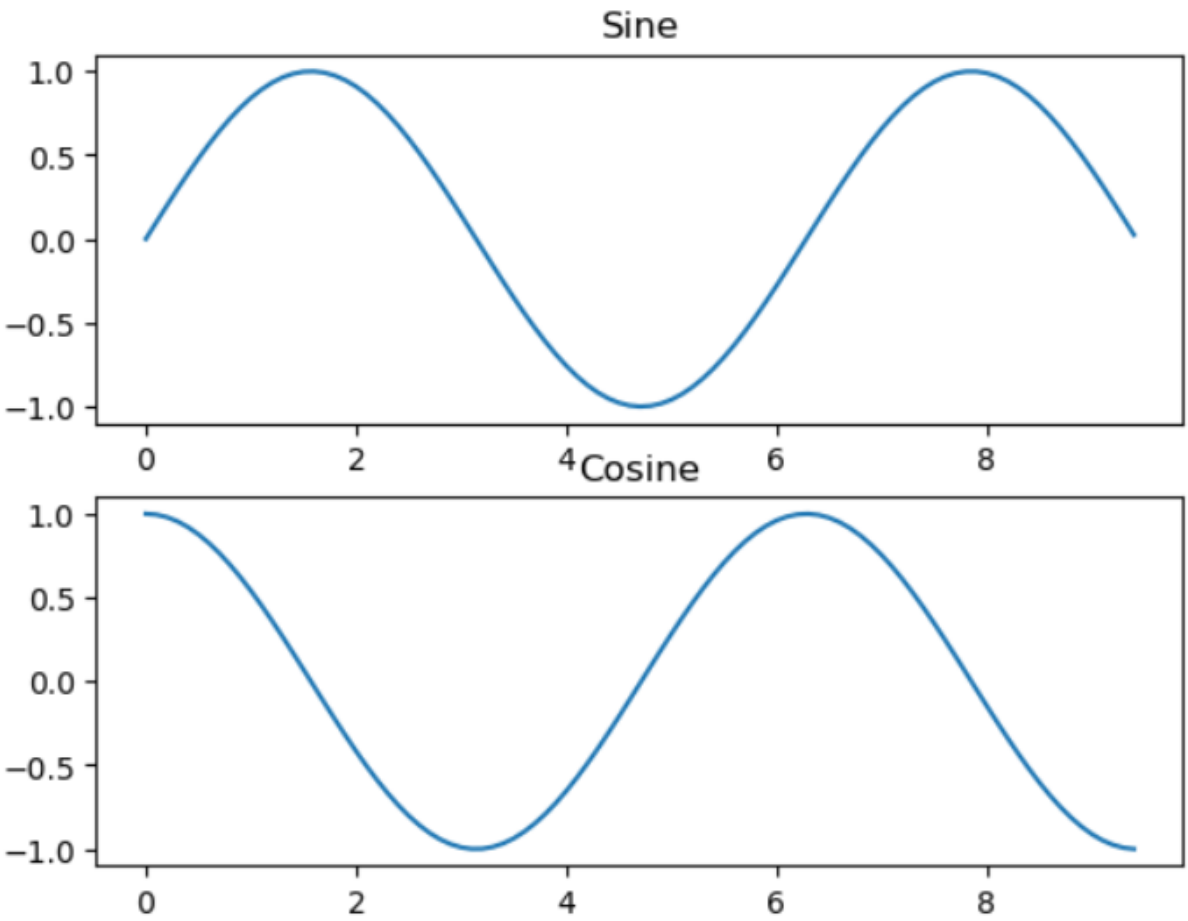
```
2. x = np.arange(0,3*np.pi,0.1)
   y_sin = np.sin(x)
   y_cos = np.cos(x)
   plt.plot(x,y_sin)
   plt.plot(x,y_cos)
   plt.xlabel('x axis label')
   plt.ylabel('y axis label')
   plt.title('Sine and Cosine')
   plt.legend(['Sine','Cosine'])
   plt.show()
```

OUTPUT:



3.
```
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('Sine')
plt.subplot(2,1,2)
plt.plot(x,y_cos)
plt.title('Cosine')
plt.show()
```

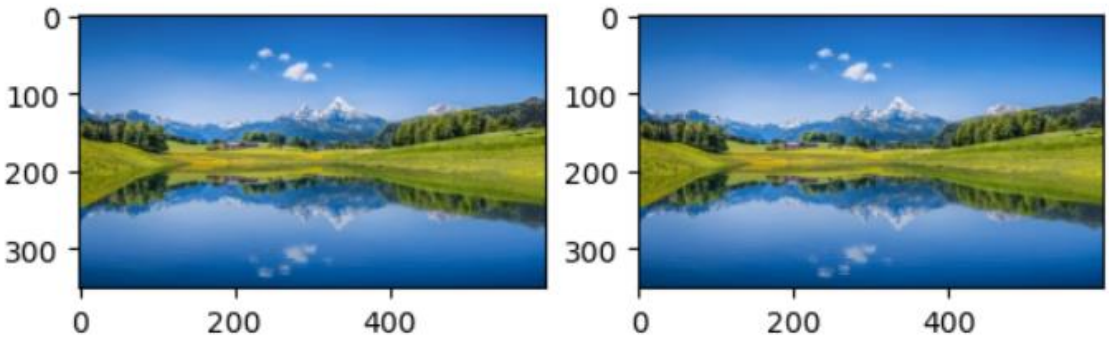
OUTPUT:



```
4. import matplotlib.image as img
   from IPython import display
   from PIL import Image
   tstimg = Image.open('scenery.jpg')
   plt.subplot(1,2,1)
   plt.imshow(tstimg)
   plt.subplot(1,2,2)
   plt.imshow(tstimg)
```

OUTPUT:

```
<matplotlib.image.AxesImage at 0x7f128a6f04d0>
```



RESULT: The program was executed successfully and the output was obtained.

LAB CYCLE: 1 Date:

PROGRAM: 1.4

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)

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.
```
ufo = pd.read_csv("http://bit.ly/uforeports")
print("Overview of UFO data reports:")
print(ufo.head())
#series
print("City series(sorted):")
print(ufo.City.sort_values())
ufo['Location']=ufo.City + ',' + ufo.State
print("After creating a new 'Location' series:")
print(ufo.head())
print("\nCalculate summary statistics:")
print(ufo.describe())
print("\nColumn names of ufo dataframe:\n", ufo.columns)
#rename two of the columns by using the 'rename' method
ufo.rename(columns={'Colors Reported':'Colors_Reported','Shape
Reported':'Shape_Reported'},inplace=True)
print("\nColumn name of ufo dataframe after renaming two column
names:\n",ufo.columns)
#rename multiple columns at once
ufo.drop(['City','State'], axis=1, inplace=True)
print("\nColumn name of ufo dataframe after removing two
columns(city,state):\n",ufo.columns)
#remove multiple rows at once(axis=0 refers to rows)
ufo.drop([0,1], axis=0, inplace=True)
print("\nufo dataframe after deleting first two rows:\n",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
City series(sorted):
1761     Abbeville
17809     Aberdeen
2297      Aberdeen
9404      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     18215            2882          15597  18241             18241
unique     6475              27             27     52             16145
top     Seattle             RED          LIGHT     CA  11/16/1999 19:00
freq        187             780           2803   2529                27

          Location
count        18215
unique        8028
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(['Colors_Reported', 'Shape_Reported', 'Time', 'Location'], dtype='object')

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

3.
```
#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("\nDifferent 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 elements 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("2.Broadcasting:-")
print(movies[movies.duration>=200])
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    False
1    False
2     True
3    False
4    False
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...
```

4.
```
#read a dataset of Chipotle orders into a DataFrame
orders = pd.read_table('http://bit.ly/chiporders')
print("Dataframe:")
print(orders.head())
print("\nString methods in pandas:-")
print("\nitem_name series(in uppercase):")
print(orders.item_name.str.upper().head())
print("\nCheck 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("replace []")
print(orders.choice_description.str.replace(r"[\[\]]",'',regex=True)
.head())
print()
print("Examine the data type of each Series:")
print(orders.dtypes)
print()
print("Dataframe after replacing 'S' and converting string to float
of 'item_price' series:")
print(orders.item_price.str.replace('$',"").astype(float))
```

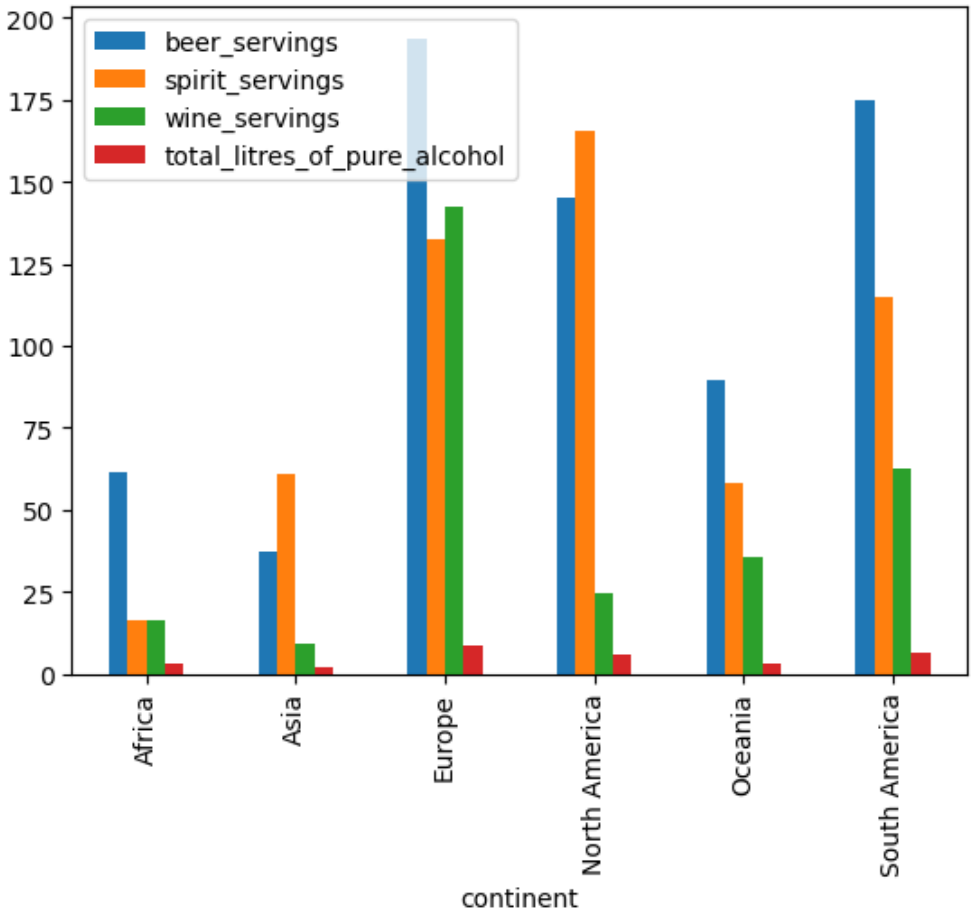
OUTPUT:

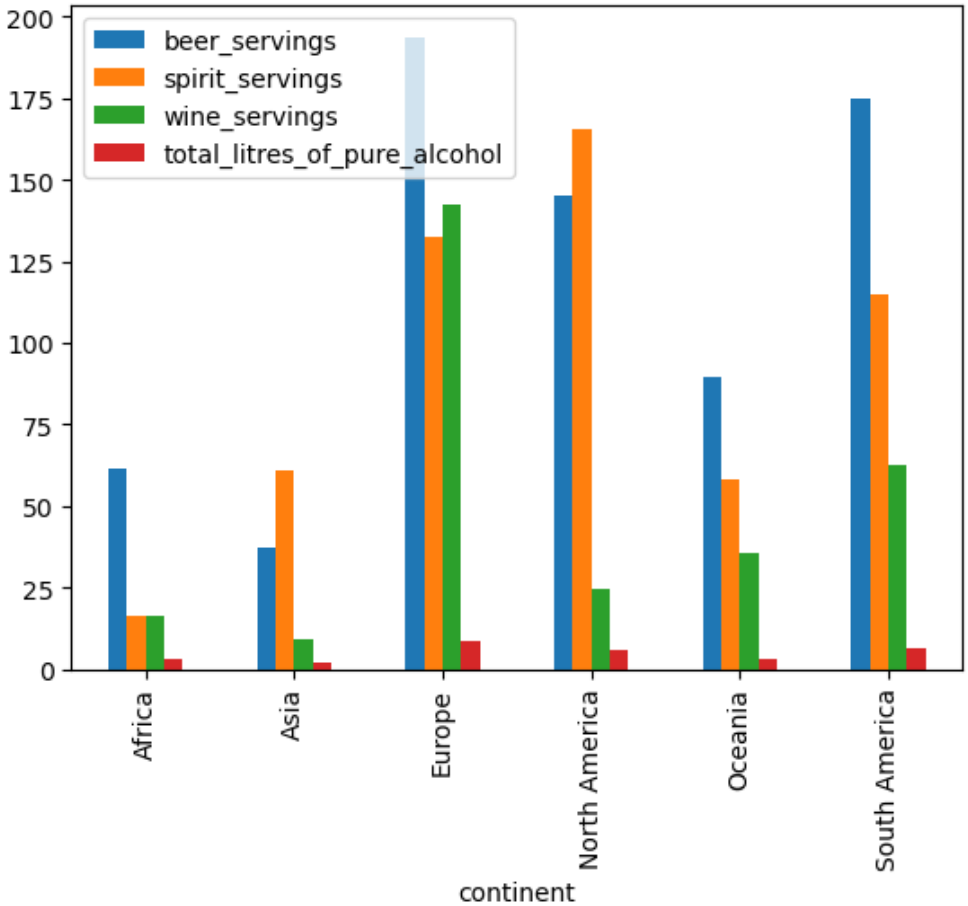
```
Dataframe:
   order_id  quantity                              item_name  \
0         1         1           Chips and Fresh Tomato Salsa
1         1         1                                   Izze
2         1         1                       Nantucket Nectar
3         1         1  Chips and Tomatillo-Green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

replace []
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 'S' 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.
```
#read a dataset of Chipotle orders into a DataFrame

drinks = pd.read_table('http://bit.ly/drinksbycountry',sep=',')

print("Dataframe:")

print(drinks.columns)

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 not required

drinks.groupby('continent').mean(numeric_only=True)

#allow plots to appear in the notebook

%matplotlib inline

#side-by-side bar plot of the DataFrame directly above

drinks.groupby('continent').mean(numeric_only=True).plot(kind='bar')

drinks.groupby('continent').mean(numeric_only=True).plot(kind='bar')
```

OUTPUT:

```
Dataframe:
Index(['country', 'beer_servings', 'spirit_servings', 'wine_servings',
       'total_litres_of_pure_alcohol', 'continent'],
      dtype='object')

Mean beer servings across the entire dataset: 106.16062176165804
Mean beer servings just for countries in Africa: 61.471698113207545

Aggregate functions used with groupby:

Mean beer servings for each continent: continent
Africa          61.471698113207545
Asia             37.04545454545455
Europe          193.77777777777777
North America   145.43478260869566
Oceania                   89.6875
South America   175.08333333333334
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 61.471698113207545    0  376
Asia              44  37.04545454545455    0  247
Europe            45 193.77777777777777    0  361
North America     23 145.43478260869566    1  285
Oceania           16            89.6875    0  306
South America     12 175.08333333333334   93  333
```


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


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. 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())
   #fll in missing values with a specifed value
   print(ufo['Shape Reported'].fillna(value='VARIOUS', inplace=True))
   # confrm that the missing values were flled 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 defne them
   print(pd.read_table('http://bit.ly/movieusers', header=None,
   sep='|').head())
   # identifcation: index remains with each row when fltering 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
  print(ufo['Shape Reported'].fillna(value='VARIOUS', inplace=True))
       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. ufo = pd.read_csv('http://bit.ly/uforeports')
   print("Dataframe: ")
   print(ufo.head(3))
   print()
   print("Selecting multiple rows and columns from a pandas Data Frame
   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, :])
```

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 Data Frame 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
```

```
8. 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 frst dummy variable ('female') using the 'iloc' method
   print(pd.get_dummies(train.Sex).iloc[:, 1:].head())
   print()
   # add a prefx 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 frst 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()
   # 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 numpy as np

# create a DataFrame from a dictionary (keys become column names, values become data)

#optionally specify the order of columns and defne 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)16 29 29

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 fll it with random numbers between0&1

arr = np.random.rand(4, 2)

print("Numpy array: ")


print(arr)

print()

print("DataFrame from the above defned 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:
[[0.66738598 0.09491928]
 [0.55293294 0.22016802]
 [0.96767014 0.00565765]
 [0.08032915 0.83041176]]

DataFrame from the above defned NumPy array:
        one       two
0  0.667386  0.094919
1  0.552933  0.220168
2  0.967670  0.005658
3  0.080329  0.830412

DataFrame of student IDs (100 through 109) and test scores (random integers between 60 and 100:
   student  test
0      100    82
1      101    96
2      102    80
```

```
3      103    78
4      104    67
5      105    67
6      106    99
7      107    62
8      108    97
9      109    74

         test
student
100        62
101        78
102        84
103        61
104        90
105        69
106        91
107        80
108        90
109        67

c     round
b    square
Name: shape, dtype: object

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

10.
```
#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()
# 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  4900.0
2                           0.7    Africa   14000   700.0
3                          12.4    Europe  312000 12400.0
4                           5.9    Africa   45000  5900.0
   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.
```
# 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)

# confrm 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)

# confrm that the 'City' column was actually removed

print(ufo.columns)

print()

print(ufo.shape)

print()

#drop a row if any value is missing from that row (doesn't
affect the DataFrame since inplace=False)

ufo.dropna(how='any')

# confrm 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 fll' strategy:
                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 fll' 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:
                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
```

RESULT: The program was executed successfully and the output was obtained.

LAB CYCLE: 2 Date:

PROGRAM: 2.1

AIM: Implementation of KNN classification algorithm.

```
1. 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']
```

RESULT: The program was executed successfully and the output was obtained.

LAB CYCLE: 2 Date:

PROGRAM: 2.2

AIM: Implementation of Naive Bayes classification algorithm.

```
2. 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)
   xtrain,xtest,ytrain,ytest =
   train_test_split(x,y,test_size=0.5,random_state=0)
   gnb = GaussianNB()
   ypred = gnb.fit(xtrain,ytrain).predict(xtest)
   print(ypred)
   xnew = [[5,5,4,4]]
   ynew = gnb.fit(xtrain,ytrain).predict(xnew)
   print("Predicted Output for [[5,5,4,4]]:", ynew)
   print("Naive Bayes score:",gnb.score(xtest,ytest))
```

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
```

RESULT: The program was executed successfully and the output was obtained.

LAB CYCLE: 2 Date:

PROGRAM: 2.3

AIM: Implementation of Decision Tree classification algorithm.

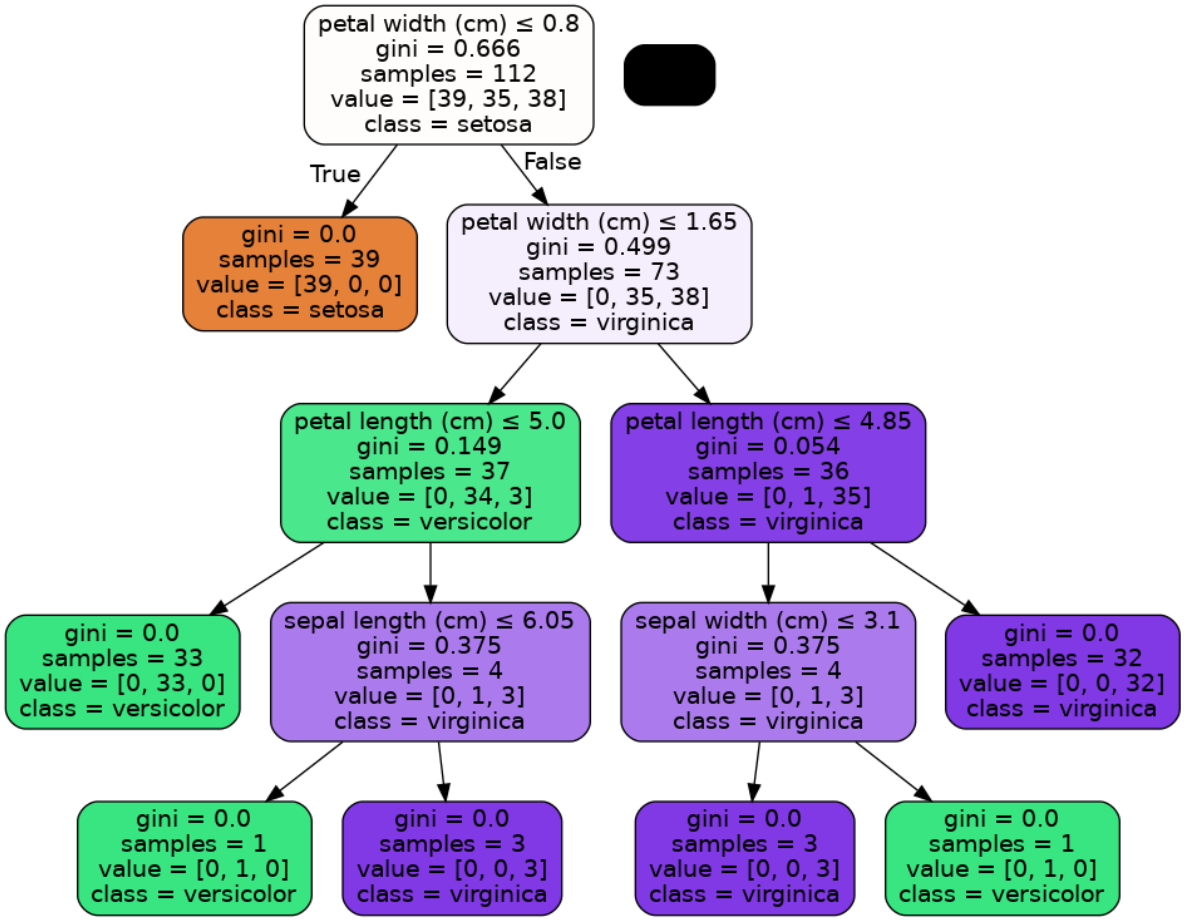
3.
```
import pandas as pd
import numpy as np
from sklearn.datasets import load_iris
data = load_iris()
print(data.data.shape)
print("Classes to predict:",data.target_names)
print("Features:",data.feature_names)
x = data.data
y = data.target
display(x.shape,y.shape)
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
xtrain,xtest,ytrain,ytest =
train_test_split(x,y,random_state=50,test_size=0.25)
#default criterion is Gini
classifier = DecisionTreeClassifier()
classifier.fit(xtrain,ytrain)
ypred = classifier.predict(xtest)
from sklearn.metrics import accuracy_score
print("Accuracy on train data using Gini:",
accuracy_score(y_true=ytrain,y_pred=classifier.predict(xtrain)))
print("Accuracy on test data using Gini:",
accuracy_score(y_true=ytest,y_pred=ypred))
#change criterion to entropy
classifier_entropy = DecisionTreeClassifier(criterion='entropy')
classifier_entropy.fit(xtrain,ytrain)
```

```
ypred_entropy = classifier_entropy.predict(xtest)

print("Accuracy on train data using entropy:",
accuracy_score(y_true=ytrain,y_pred=classifier_entropy.predict(xtrai
n)))

print("Accuracy on test data using entropy:",
accuracy_score(y_true=ytest,y_pred=ypred_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(xtrain,ytrain)

ypred_entropy1 = classifier_entropy1.predict(xtest)

print("Accuracy on train data using enhttp://localhost:8888/tropy:",
accuracy_score(y_true=ytrain,y_pred=classifier_entropy1.predict(xtra
in)))

print("Accuracy on test data using entropy:",
accuracy_score(y_true=ytest,y_pred=ypred_entropy1))


#visualise the decision tree

from sklearn.tree import export_graphviz

from six import StringIO

from IPython.display import Image

import pydotplus

dotdata = StringIO()


#INCASE OF MODULE-MISSING ERRORS; DO FOLLOWING 2 STEPS;

#------CREATE NEW CELL -> TYPE AND RUN: pip install pydotplus   ----
--

#------OPEN NEW TERMINAL -> TYPE AND ENTER:  sudo apt-get install
graphviz   ------
```

```
#the students can try using classifier, classifier_entropy &
classifier_entropy1

#as first parameter below.

export_graphviz(classifier,out_file =
dotdata,filled=True,rounded=True,special_characters=True,feature_nam
es=data.feature_names,class_names=data.target_names)

graph = pydotplus.graph_from_dot_data(dotdata.getvalue())

Image(graph.create_png())
```

OUTPUT:

```
(150, 4)
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 was executed successfully and the output was obtained.

LAB CYCLE: 3 Date:

PROGRAM: 3.1

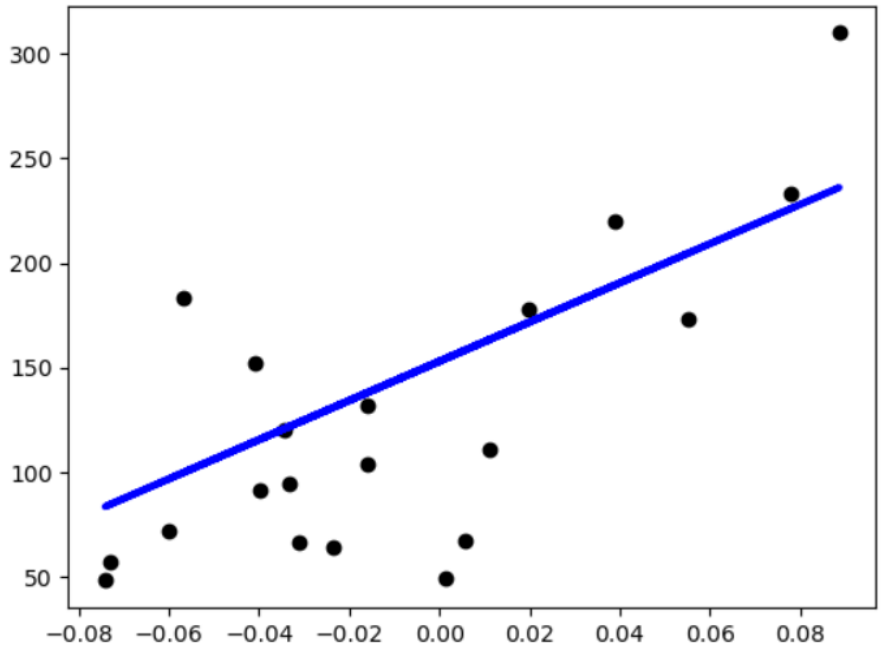
AIM: Implementation of Linear Regression techniques.

1.
```
#------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 diabetes dataset
diabetesX, diabetesY = datasets.load_diabetes(return_X_y = True)
#------Split Datasets------
#use only one feature
diabetesX = diabetesX[:,np.newaxis,2]
#split the data into training/testing sets
diabetesXtrain = diabetesX[:-20]
diabetesXtest = diabetesX[-20:]
#split the targets into training/testing sets
diabetesYtrain = diabetesY[:-20]
diabetesYtest = diabetesY[-20:]
#------Creating the model------
#create linear regression object
regr = linear_model.LinearRegression()
#train the model using training sets
regr.fit(diabetesXtrain,diabetesYtrain)
#------Make prediction------
#make predictions using the testing set
diabetesYpred = regr.predict(diabetesXtest)
```

```
#------finding coefficients and mean square error------

#coefficients

print("Coefficients:\n",regr.coef_)

#mean squared error

print("Mean squared
error:\n",mean_squared_error(diabetesYtest,diabetesYpred))

#coefficient of determination: 1 is perfect prediction

print("Coefficient of
determination:",r2_score(diabetesYtest,diabetesYpred))

plt.scatter(diabetesXtest,diabetesYtest,color='black')

plt.plot(diabetesXtest,diabetesYpred,color='blue',linewidth=3)

plt.xticks()

plt.yticks()

plt.show()
```

OUTPUT:

```
Coefficients:
 [938.23786125]
Mean squared error:
 2548.07239872597
Coefficient of determination: 0.47257544798227147
```



RESULT: The program was executed successfully and the output was obtained.

LAB CYCLE: 3 Date:

PROGRAM: 3.2

AIM: Image classification using convolutional neural networks.

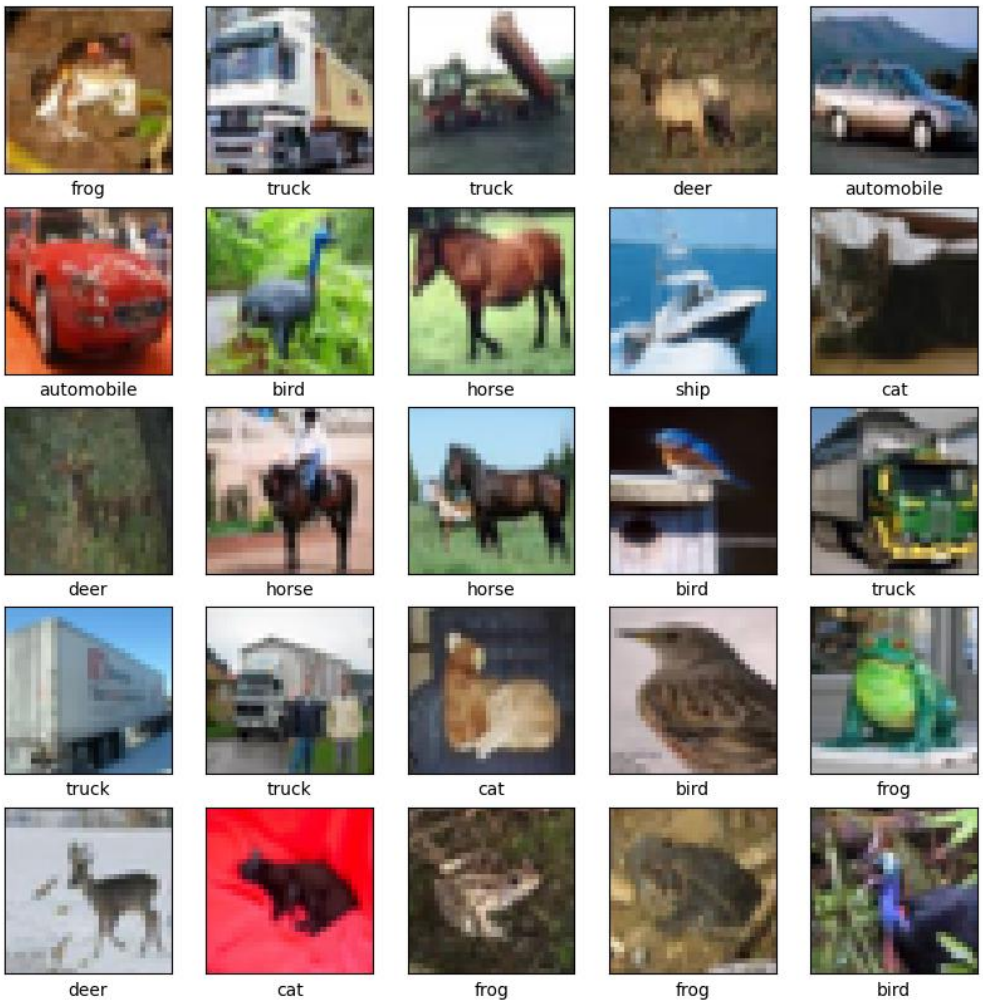
2. #------CNN------

```
# FOR MODULE ERROR; ADD NEW CELL -> TYPE AND RUN:-  pip install
tensorflow

import tensorflow as tf

from tensorflow import keras

from keras import datasets, layers, models

import matplotlib.pyplot as plt

# Load and normalize the CIFAR-10 dataset

(train_images, train_labels), (test_images, test_labels) =
datasets.cifar10.load_data()

train_images, test_images = train_images / 255.0, test_images /
255.0

# Class names for CIFAR-10

class_names = ['airplane', 'automobile', 'bird', 'cat', 'deer',
'dog', 'frog', 'horse', 'ship', 'truck']

# Plotting the first 25 images from the training set

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]])

plt.show()
```

```
# Building the CNN model

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'))

# Adding Dense layers

model.add(layers.Flatten())

model.add(layers.Dense(64, activation='relu'))

model.add(layers.Dense(10))

print("Architecture of the model:\n")

model.summary()

# Compiling the model

model.compile(optimizer='adam',
loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True)
, metrics=['accuracy'])

# Training the model

history = model.fit(train_images, train_labels, epochs=10,
validation_data=(test_images, test_labels))

# Evaluating the model

test_loss, test_acc = model.evaluate(test_images, test_labels,
verbose=2)

print("Test loss:", test_loss)

print("Test accuracy:", test_acc)

# Plotting training and validation accuracy/loss

plt.figure(figsize=(12, 4))

plt.subplot(1, 2, 1)

plt.plot(history.history['accuracy'], label='accuracy')

plt.plot(history.history['val_accuracy'], label='val_accuracy')
```

```
plt.xlabel('Epoch')
plt.ylabel('Accuracy')
plt.legend(loc='lower right')
plt.title('Training and Validation Accuracy')
plt.subplot(1, 2, 2)
plt.plot(history.history['loss'], label='loss')
plt.plot(history.history['val_loss'], label='val_loss')
plt.xlabel('Epoch')
plt.ylabel('Loss')
plt.legend(loc='upper right')
plt.title('Training and Validation Loss')
plt.show()
```

OUTPUT:



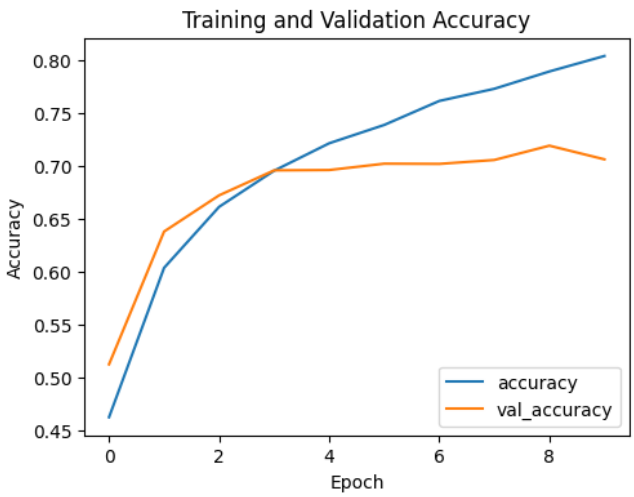
Architecture of the model:

Model: "sequential"

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 30, 30, 32)	896
max_pooling2d (MaxPooling2D)	(None, 15, 15, 32)	0
conv2d_1 (Conv2D)	(None, 13, 13, 64)	18496
max_pooling2d_1 (MaxPooling2D)	(None, 6, 6, 64)	0
conv2d_2 (Conv2D)	(None, 4, 4, 64)	36928
flatten (Flatten)	(None, 1024)	0
dense (Dense)	(None, 64)	65600
dense_1 (Dense)	(None, 10)	650

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

```
1563/1563 [==============================] - 12s 7ms/step - loss: 1.4840 - accuracy: 0.4629 - val_loss: 1.3304 - val_accuracy: 0.5129
Epoch 2/10
1563/1563 [==============================] - 11s 7ms/step - loss: 1.1147 - accuracy: 0.6040 - val_loss: 1.0307 - val_accuracy: 0.6383
Epoch 3/10
1563/1563 [==============================] - 12s 8ms/step - loss: 0.9643 - accuracy: 0.6616 - val_loss: 0.9492 - val_accuracy: 0.6725
Epoch 4/10
1563/1563 [==============================] - 12s 8ms/step - loss: 0.8650 - accuracy: 0.6958 - val_loss: 0.8869 - val_accuracy: 0.6961
Epoch 5/10
1563/1563 [==============================] - 13s 8ms/step - loss: 0.8005 - accuracy: 0.7217 - val_loss: 0.8782 - val_accuracy: 0.6964
Epoch 6/10
1563/1563 [==============================] - 13s 8ms/step - loss: 0.7430 - accuracy: 0.7390 - val_loss: 0.8576 - val_accuracy: 0.7024
Epoch 7/10
1563/1563 [==============================] - 12s 8ms/step - loss: 0.6861 - accuracy: 0.7617 - val_loss: 0.8879 - val_accuracy: 0.7022
Epoch 8/10
1563/1563 [==============================] - 12s 8ms/step - loss: 0.6415 - accuracy: 0.7731 - val_loss: 0.8777 - val_accuracy: 0.7059
Epoch 9/10
1563/1563 [==============================] - 12s 8ms/step - loss: 0.5993 - accuracy: 0.7895 - val_loss: 0.8398 - val_accuracy: 0.7194
Epoch 10/10
1563/1563 [==============================] - 13s 8ms/step - loss: 0.5533 - accuracy: 0.8041 - val_loss: 0.8950 - val_accuracy: 0.7065
313/313 - 1s - loss: 0.8950 - accuracy: 0.7065 - 866ms/epoch - 3ms/step
Test loss: 0.8950179815292358
Test accuracy: 0.7064999938011169
```



RESULT: The program was executed successfully and the output was obtained.

LAB CYCLE: 4 Date:

PROGRAM: 4.1

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

1.
```
import nltk

nltk.download('punkt')

nltk.download('stopwords')

nltk.download('wordnet')

nltk.download('omw-1.4')

nltk.download('averaged_perceptron_tagger')

nltk.download('maxent_ne_chunker')


nltk.download('punkt_tab')


from nltk.tokenize import sent_tokenize, word_tokenize

from nltk.corpus import stopwords

from string import punctuation

from nltk.stem import PorterStemmer, WordNetLemmatizer

from nltk import RegexpParser


# Sample text

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")


# Tokenizing by sentence

sent_tokenized = sent_tokenize(text)
```

```
print("Tokenizing by sentence: \n", sent_tokenized, "\n")


# Tokenizing by word
word_tokenized = word_tokenize(text)
print("Tokenizing by word: \n", word_tokenized, "\n")


# Removing stop words and punctuation
stop_words = set(stopwords.words('english'))
punctuation_set = set(punctuation)


print("After filtering the stop words and punctuation: ")
filtered_words = [word for word in word_tokenized if word.casefold()
not in stop_words and word.casefold() not in punctuation_set]
for word in filtered_words:
    print(word)


# Stemming
ps = PorterStemmer()
words = ["reading", "globalization", "Being", "Went", "gone",
"going"]
print("\nGiven words: ", words)
stemm = [ps.stem(i) for i in words]
print("After stemming: ", stemm, "\n")


# Lemmatization
lem = WordNetLemmatizer()
print("rocks:", lem.lemmatize("rocks"))
print("corpora:", lem.lemmatize("corpora"))
print("better:", lem.lemmatize("better"))
print("believes:", lem.lemmatize("believes"), "\n")
```

```
# Lemmatization with POS tag
print("went as adjective:", lem.lemmatize("went", pos="a"))
print("went as verb:", lem.lemmatize("went", pos="v"))
print("went as noun:", lem.lemmatize("went", pos="n"), "\n")

# POS tagging
postag = nltk.pos_tag(word_tokenized)
print("POS tagging: \n")
for i in postag:
    print(i)
print("\n")

# Chunking
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 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 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', '.']

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

Given words:  ['reading', 'globalization', 'Being', 'Went', 'gone', 'going']
After stemming:  ['read', 'global', 'Be', 'Went', 'gon', 'go']

rocks: rock
corpora: corpus
better: better
believes: believe

went as adjective: went
went as verb: go
went as noun: went
```

```
POS tagging:
('A', 'DT')
('newspaper', 'NN')
('is', 'VBZ')
('the', 'DT')
('strongest', 'JJS')
('medium', 'NN')
('for', 'IN')
('news', 'NNS')
('.', '.')
('People', 'NNS')
('are', 'VBP')
('reading', 'VBG')
('newspapers', 'NNS')
('for', 'IN')
('decades', 'NNS')
('.', '.')
('It', 'PRP')
('has', 'VBZ')
('a', 'DT')
('huge', 'JJ')
('contribution', 'NN')
('to', 'TO')
('globalization', 'NN')
('.', '.')
('Right', 'RB')
('now', 'RB')
('because', 'IN')
('of', 'IN')
('easy', 'JJ')
('internet', 'NN')
('connection', 'NN')
(',', ',')
('people', 'NNS')
('don\'t', 'VB')
('read', 'VB')
('printed', 'VBD')
('newspapers', 'NNS')
('often', 'RB')
('.', '.')
('They', 'PRP')
('read', 'VBP')
('the', 'DT')
('online', 'JJ')
('version', 'NN')
('.', '.')
```

```
After Chunking:
(S
  (NP A newspaper)
  is
  the
  (NP strongest medium)
  for
  (NP news)
  .
  (NP People)
  are
  (VBG reading)
  (NP newspapers)
  for
  (NP decades)
  .
  It
  has
  a
  huge
  (NP contribution)
  to
  (NP globalization)
  .
  (NP Right now)
  because
  of
  easy
  (NP internet connection)
  ,
  (NP people)
  don't
  read
  (NP printed newspapers)
  often
  .
  (NP They)
  read
  the
  (NP online version)
  .)
```

RESULT: The program was executed successfully and the output was obtained.

LAB CYCLE: 4 Date:

PROGRAM: 4.2

AIM: Text classification using support vector machines.

```
2. 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 1 1 0 1 1 1 0 1 1 0 0 1 0
 0 0 1 1 1 1 1 1 0 0 1 0 1 1 1 1 1 1 1 1 1 0 1 1 1]
Predicted values: [1 1 0 0 1 0 1 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 1 0 0 1 0 1 1 1 1 1 1 1 1 1 0 1 1 1]
Accuracy: 0.9649122807017544
Precision: 0.9811320754716981
Recall: 0.9629629629629629
```

RESULT: The program was executed successfully and the output was obtained.

LAB CYCLE: 4 Date:

PROGRAM: 4.3

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

3.
```
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.string)
print()
print("All the links: ")
links = [link.get('href') for link in ritsoup.find_all('a')]
print(links, "\n")
print(ritsoup.head)
print("For over 150 years, generations of Boilermakers have left
their mark in small steps and giant leaps. Today, we continue in
those footsteps as we bring our best and learn to build a better
world, together.")



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()
if ritsoup.h1:
    print("Obtaining strings: ", ritsoup.h1.string)
print()
if ritsoup.h1:
    print("Obtaining strings using contents method: ",
ritsoup.h1.contents)
print()
if len(paragraphs) > 2:
    print(paragraphs[2], "\n")
    print(paragraphs[2].string, "\n")
if len(paragraphs) > 2:
    para2 = paragraphs[2].contents
    print(para2, "\n")
if len(paragraphs) > 5:
    para7 = paragraphs[5].contents
    print(para7, "\n")
if len(paragraphs) > 3:
    print("Raw text from the 4th paragraph:")
    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__list .purdue-home-news-events__title{
min-height:3.5rem;
}
}
.purdue-home-hot-spot {
    overflow: visible !important;
}
.purdue-home-slide__hot-spot-mobile{
    overflow: hidden;
}

/***
.purdue-home-button-list li:last-child .purdue-home-button{
    background: #000;
    color: #fff;
}
.purdue-home-button-list li:last-child .purdue-home-button:after{
    background-image: url("data:image/svg+xml,%3Csvg
xmlns=%27http://www.w3.org/2000/svg%27 viewBox=%270 0 448 512%27%3E%3C%21--%21 Font
Awesome Pro 6.4.0 by @fontawesome - https://fontawesome.com License -
https://fontawesome.com/license %28Commercial License%29 Copyright 2023 Fonticons,
Inc. --%3E%3Cpath fill=%27%23cfb991%27 d=%27M438.6 278.6c12.5-12.5 12.5-32.8 0-
45.3l-160-160c-12.5-12.5-32.8-12.5-45.3 0s-12.5 32.8 0 45.3L338.8 224 32 224c-17.7
0-32 14.3-32 32s14.3 32 32 32l306.7 0L233.4 393.4c-12.5 12.5-12.5 32.8 0 45.3s32.8
12.5 45.3 0l160-160z%27/%3E%3C/svg%3E");
}
.purdue-home-button-list li:last-child .purdue-home-button:hover{
color: #000;
background: #cfb991 !important;
}
.purdue-home-button-list li:last-child .purdue-home-button:hover:after{
background-image: url("data:image/svg+xml,%3Csvg
xmlns=%27http://www.w3.org/2000/svg%27 viewBox=%270 0 448 512%27%3E%3C%21--%21 Font
Awesome Pro 6.4.0 by @fontawesome - https://fontawesome.com License -
https://fontawesome.com/license %28Commercial License%29 Copyright 2023 Fonticons,
Inc. --%3E%3Cpath fill=%27%23000000%27 d=%27M438.6 278.6c12.5-12.5 12.5-32.8 0-
45.3l-160-160c-12.5-12.5-32.8-12.5-45.3 0s-12.5 32.8 0 45.3L338.8 224 32 224c-17.7
0-32 14.3-32 32s14.3 32 32 32l306.7 0L233.4 393.4c-12.5 12.5-12.5 32.8 0 45.3s32.8
12.5 45.3 0l160-160z%27/%3E%3C/svg%3E");
}
***/</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>
<!-- End Google Tag Manager for WordPress by gtm4wp.com -->
<!-- The SEO Framework by Sybre Waaijer -->
<meta content="max-snippet:-1,max-image-preview:large,max-video-preview:-1"
name="robots">
```

```
<link href="https://www.purdue.edu/home/" rel="canonical">
<meta content="Purdue University is a world-renowned, public research university
that advances discoveries in science, technology, engineering and math."
name="description">
<meta content="website" property="og:type"/>
<meta content="en_US" property="og:locale"/>
<meta content="Purdue University" property="og:site_name"/>
<meta content="Purdue University" property="og:title"/>
<meta content="Purdue University is a world-renowned, public research university
that advances discoveries in science, technology, engineering and math."
property="og:description"/>
<meta content="https://www.purdue.edu/home/" property="og:url"/>
<meta content="https://www.purdue.edu/home/wp-content/uploads/2023/09/cropped-Home-
Cover-2023_KAL_1419.jpg" property="og:image"/>
<meta content="1499" property="og:image:width"/>
<meta content="787" property="og:image:height"/>
<meta content="Purdue University students laughing and sitting on the Engineering
Fountain at the main campus of Purdue University in West Lafayette, Indiana."
property="og:image:alt"/>
<meta content="summary_large_image" name="twitter:card"/>
<meta content="Purdue University" name="twitter:title"/>
<meta content="Purdue University is a world-renowned, public research university
that advances discoveries in science, technology, engineering and math."
name="twitter:description"/>
<meta content="https://www.purdue.edu/home/wp-content/uploads/2023/09/cropped-Home-
Cover-2023_KAL_1419.jpg" name="twitter:image"/>
<meta content="Purdue University students laughing and sitting on the Engineering
Fountain at the main campus of Purdue University in West Lafayette, Indiana."
name="twitter:image:alt"/>
<script type="application/ld+json">{"@context":"https://schema.org","@graph":
[{"@type":"WebSite","@id":"https://www.purdue.edu/home/#/schema/
WebSite","url":"https://www.purdue.edu/home/","name":"Purdue
University","description":"Indiana&#039;s Land Grant University","inLanguage":"en-
US","potentialAction":{"@type":"SearchAction","target":
{"@type":"EntryPoint","urlTemplate":"https://www.purdue.edu/home/search/
{search_term_string}/"},"query-input":"required
name=search_term_string"},"publisher":{"@id":"https://www.purdue.edu/home/#/
schema/Organization"}},{"@type":"WebPage","@id":"https://www.purdue.edu/
home/","url":"https://www.purdue.edu/home/","name":"Purdue University &#x2d;
Indiana&#039;s Land Grant University","description":"Purdue University is a
world&#x2d;renowned, public research university that advances discoveries in
science, technology, engineering and math.","inLanguage":"en-US","isPartOf":
{"@id":"https://www.purdue.edu/home/#/schema/WebSite"},"breadcrumb":
{"@type":"BreadcrumbList","@id":"https://www.purdue.edu/home/#/schema/
BreadcrumbList","itemListElement":{"@type":"ListItem","position":1,"name":"Purdue
University"}},"potentialAction":{"@type":"ReadAction","target":"https://
www.purdue.edu/home/"},"about":{"@id":"https://www.purdue.edu/home/#/schema/
Organization"}},{"@type":"Organization","@id":"https://www.purdue.edu/home/#/
schema/Organization","name":"Purdue
University","url":"https://www.purdue.edu/home/"}]}</script>
<!-- / The SEO Framework by Sybre Waaijer | 41.02ms meta | 13.37ms boot -->
<link href="//use.typekit.net" rel="dns-prefetch"/>
<link href="//fonts.googleapis.com" rel="dns-prefetch"/>
<link href="//use.fontawesome.com" rel="dns-prefetch"/>
<script type="text/javascript">
/* <![CDATA[ */
window._wpemojiSettings =
{"baseUrl":"https:\/\/s.w.org\/images\/core\/emoji\/15.0.3\/72x72\/","ext":".png","
svgUrl":"https:\/\/s.w.org\/images\/core\/emoji\/15.0.3\/
```

```
<script data-cfasync="false" data-pagespeed-no-defer="">
      var dataLayer_content = {"pagePostType":"frontpage","pagePostType2":"single-
page","pagePostAuthor":"Nicole Gilles"};
      dataLayer.push( dataLayer_content );
</script>
<script data-cfasync="false">
(function(w,d,s,l,i){w[l]=w[l]||[];w[l].push({'gtm.start':
new Date().getTime(),event:'gtm.js'});var f=d.getElementsByTagName(s)[0],
j=d.createElement(s),dl=l!='dataLayer'?'&l='+l:'';j.async=true;j.src=
'//www.googletagmanager.com/gtm.js?id='+i+dl;f.parentNode.insertBefore(j,f);
})(window,document,'script','dataLayer','GTM-T863X92Z');
</script>
<!-- End Google Tag Manager for WordPress by gtm4wp.com
--></meta></link></meta></head>
For over 150 years, generations of Boilermakers have left their mark in small steps
and giant leaps. Today, we continue in those footsteps as we bring our best and
learn to build a better world, together.
<meta charset="utf-8"/>

Get all <p> elements:  [<p class="purdue-home-cta-card__cta-text">GIVE TO
PURDUE</p>, <p class="cta-link purdue-home-cta-card__link"
href="https://giving.purdue.edu/support/?appealcode=20733" target="_blank">Discover
ways to donate and make your gift before Dec. 31.</p>, <p class="purdue-home-cta-
card__cta-text">GIVE TO PURDUE</p>, <p class="purdue-home-cta-card__tag">Stadium
Mall  </p>, <p class="purdue-home-cta-card__title">Neil Armstrong Statue  </p>, <p
class="purdue-home-cta-card__content">Take your photo with the alumnus depicted as
he was during his undergraduate days: ready to go with a stack of books and a slide
rule.  </p>, <p class="cta-link">Learn More</p>, <p class="purdue-home-cta-
card__tag">Stadium Mall  </p>, <p class="purdue-home-cta-card__title">Gateway to
the Future  </p>, <p class="purdue-home-cta-card__content">Step into new beginnings
under the arch at the north end of Stadium Mall. It symbolizes heading toward the
future for Boilermakers.  </p>, <p class="cta-link">Learn More</p>, <p
class="purdue-home-cta-card__tag">France A. Córdova Recreational Sports Center</p>,
<p class="purdue-home-cta-card__title">GroupX</p>, <p class="purdue-home-cta-
card__content">Join a group fitness class, work out on your schedule or try
something new in a state-of-the-art facility.</p>, <p class="cta-link">Learn
More</p>, <p class="purdue-home-cta-card__tag">France A. Córdova Recreational
Sports Center</p>, <p class="purdue-home-cta-card__title">Climbing and
Bouldering</p>, <p class="purdue-home-cta-card__content">Scale a 55-foot-tall roped
climbing wall and test your skills on the bouldering wall with more than 60
horizontal feet of terrain.</p>, <p class="cta-link">Learn More</p>, <p
class="purdue-home-cta-card__tag">Architecture</p>, <p class="purdue-home-cta-
card__title">Purdue Bell Tower</p>, <p class="purdue-home-cta-card__content">Hear
the four bells with a history stretching back more than 100 years — plus, a special
speaker for songs like “Hail Purdue!”</p>, <p class="cta-link">Learn More</p>, <p
class="purdue-home-cta-card__tag">Architecture</p>, <p class="purdue-home-cta-
card__title">Engineering Fountain</p>, <p class="purdue-home-cta-
card__content">Explore everything we have to offer on campus and be where some of
our most treasured traditions take place.</p>, <p class="cta-link">Watch Video</p>,
<p class="purdue-home-cta-card__tag">Purdue University in Indianapolis</p>, <p
class="purdue-home-cta-card__title">In-demand majors </p>, <p class="purdue-home-
cta-card__content">Hands-on learning opportunities and direct admission to
engineering programs drive Boilermakers to innovate and grow here.   </p>, <p
class="cta-link">Explore majors</p>, <p class="purdue-home-cta-card__tag">Purdue
University in Indianapolis</p>, <p class="purdue-home-cta-card__title">Career-ready
students</p>, <p class="purdue-home-cta-card__content">Our STEM-focused urban
campus is preparing students for the jobs of tomorrow in areas such as AI,
biomedical engineering, data science, cybersecurity and more.  </p>, <p class="cta-
link">Learn More</p>, <p class="purdue-home-cta-card__tag">Stadium Mall  </p>, <p
```

class="purdue-home-cta-card__title">Neil Armstrong Statue , Take your photo with the alumnus depicted as he was during his undergraduate days: ready to go with a stack of books and a slide rule. , Learn More , Stadium Mall , Gateway to the Future , Step into new beginnings under the arch at the north end of Stadium Mall. It symbolizes heading toward the future for Boilermakers. , Learn More , France A. Córdova Recreational Sports Center , GroupX , Join a group fitness class, work out on your schedule or try something new in a state-of-the-art facility. , Learn More , France A. Córdova Recreational Sports Center , Climbing and Bouldering , Scale a 55-foot-tall roped climbing wall and test your skills on the bouldering wall with more than 60 horizontal feet of terrain. , Learn More , Architecture , Purdue Bell Tower , Hear the four bells with a history stretching back more than 100 years — plus, a special speaker for songs like “Hail Purdue!” , Learn More , Architecture , Engineering Fountain , Explore everything we have to offer on campus and be where some of our most treasured traditions take place. , Watch Video , Purdue University in Indianapolis , In-demand majors , Hands-on learning opportunities and direct admission to engineering programs drive Boilermakers to innovate and grow here. , Explore majors , Purdue University in Indianapolis , Career-ready students , Our STEM-focused urban campus is preparing students for the jobs of tomorrow in areas such as AI, biomedical engineering, data science, cybersecurity and more. , Learn More , Purdue, Google will gather business, education and government leaders to explore the power of AI ,
October 29, 2024
, Purdue accelerates research through adopting novel AI tools, increasing supercomputing prominence ,
October 31, 2024
, WGHI-supported research could lead to blood tests for early breast cancer diagnoses ,
October 29, 2024
, ICYMI: Purdue alum and pilot Capt. Sully Sullenberger of ‘Miracle on the Hudson’ fame highlights Purdue Presidential Lecture ,
November 8, 2024
,
Purdue University
610 Purdue Mall
West Lafayette, IN 47907 ,
765-494-4600]

Gets all the 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 with One Small
Step']

<p class="purdue-home-cta-card__cta-text">GIVE TO PURDUE</p>

GIVE TO PURDUE

['GIVE TO PURDUE']

['Take your photo with the alumnus depicted as he was during his undergraduate
days: ready to go with a stack of books and a slide rule.  ']

Raw text from the 4th paragraph:
==================================================
Stadium Mall
```

RESULT: The program was executed successfully and the output was obtained.