GUJARAT TECHNOLOGICAL UNIVERSITY

Government Engineering College, Bhavnagar

Subject: **Python for Data Science**

B.E. C.E. Semester-5th

(Computer Branch)

Submitted By:

Name: Vankani Arjun BakulBhai

Enrollment: 180210107060

Prof. Chinmay Vyas
(Faculty Guide)

Prof. KARSHAN KANDORIYA
(Head of the Department)

Academic Year (2020-21)

Python Program (Ass-1)

1) Count the vowel in given string

Input: A String to count all vowels

Program :->

```
class Vowelchar():
    def vowel(self):
        vowels = ('a', 'A', 'e', 'E', 'i', 'I', 'o', 'O', 'u', 'U')
        string = input("Enter the string:")
        output = {}
        for char in vowels:
          x = string.count(char)
          output[char] = x
        print(output)

Vowelchar().vowel()
```

Output: => Enter the string:A String to count all vowels

{'a': 1, 'A': 1, 'e': 1, 'E': 0, 'i': 1, 'I': 0, 'o': 3, 'O': 0, 'u': 1, 'U': 0}

2)Compare both side halves string..

Program:->

```
class HalvesStr():
    def check(self):
        string = input("Enter String :").lower()
        middle = len(string)//2
        leftSide = string[0:middle]

        if(len(string) % 2 == 0):
            rightSide = string[middle:]
        else:
            rightSide = string[middle+1:]
        for char in leftSide:
            if(char not in rightSide):
                print("NO")
                exit()
        print("YES")
HalvesStr().check()
```

Output: 1) Enter String :xyzayzax

YES

2) Enter String :xyZayzax

YES

3) Enter String :xyzzxxyz

No

4) Enter String :xyzaayzax

YES

3) Password check:

Program=>

```
def check(passwd):
    val = True

    if len(passwd) < 9:
        print('length should be at least 9')
        val = False

    if not any(char.isdigit() for char in passwd):
        print('Password should have at least one numeral')
        val = False

    if not any(char.isupper() for char in passwd):
        print('Password should have at least one uppercase
letter')
        val = False

    if not any(char.islower() for char in passwd):
```

```
        print('Password should have at least one lowercase
letter')
        val = False


    Specialchar =['', '%', '#', '$']
    if not any(char in Specialchar for char in passwd):
        print('Password should have at least one of the
symbols _ Or % Or * Or $')
        val = False
    if val:
        return val


print("Enter password ...")
passwd = input()
if (check(passwd)):
        print("Password Accepted")
else:
        print("Invalid Password !!")
```

Outputs: 1) Enter password ...

demoPassword6$

Password Accepted

2) Enter password ...

demoPassword6&

Password should have at least one of the symbols _ Or % Or * Or $

Invalid Password !!

3) Enter password ...

demopassword6$

Password should have at least one uppercase letter

Invalid Password !!

4) Enter password ...

deMopassword$

Password should have at least one numeral

Invalid Password !!

4)Caesar cipher encryption:

```
def Encryption(s,k):
    encstr=""
    for i in s:
        if(ord(i))>=65 and (ord(i)<=90):
            temp=(ord(i)+k)
            if temp>90:
                temp=temp%90+64
            encstr=encstr+chr(temp)
        elif(ord(i))>=97 and (ord(i)<=122):
            temp=(ord(i)+k)
            if temp>122:
                temp=temp%122+96
            encstr=encstr+chr(temp)
        else:
            encstr=encstr+chr(ord(i)+k)
    return encstr
def Decryption(k):
    p=Encryption(s,k)
```

```
    decstr=""
    for i in p:
        if((ord(i))>=65) and (ord(i))<=90:
            decstr=decstr+chr((ord(i) - k-65) % 26 + 65)
        elif((ord(i))>=97) and (ord(i))<=122:
            decstr=decstr+chr((ord(i) - k - 97) % 26 + 97)
        else:
            decstr=decstr+chr(ord(i)-k)
    return decstr
print("Enter the string to Encrypt and decrypt : ")
s=input()
k=9
k=k%26
print("Encrypted String : ",Encryption(s,k))
print("Decrypted String : ",Decryption(k))
```

Outputs: =>Enter the string to Encrypt and decrypt :

This is DUMMY String

Encrypted String : Cqrb)rb)MDVVH)Bcarwp

Decrypted String : This is DUMMY String

Python Practical Set- 2

1) **Merger two list with tuple order:**

Program->

```
class MergeEle():
    def init(self):
        list1 = []
        list2 = []
        list3 = []
        l1_length = int(input("Enter first list length: "))
        l2_length = int(input("Enter second list length: "))

        for i in range(l1_length) :
            l1_data = input()
            list1.append(l1_data)
        print(list1)

        for i in range(l2_length) :
            l2_data = input()
            list2.append(l2_data)
        print(list2)

        for i in range(max(l1_length, l2_length)) :
            while True :
                try :
                    l3_data = (list1[i], list2[i])
                except IndexError :
                    if l1_length > l2_length :
                        list2.append('')
                        l3_data = (list1[i], list2[i])
                    elif l1_length < l2_length :
                        list1.append('')
                        l3_data = (list1[i], list2[i])

                list3.append(l3_data)
                break

        print(list3)

MergeEle.init(0)
```

Input/Output:

```
Enter first list length: 3
Enter second list length: 3
a
b
c
['a', 'b', 'c']
x
y
z
['x', 'y', 'z']
[('a', 'x'), ('b', 'y'), ('c', 'z')]

Enter first list length: 5
Enter second list length: 3
a
b
c
p
q
['a', 'b', 'c', 'p', 'q']
x
y
z
['x', 'y', 'z']
[('a', 'x'), ('b', 'y'), ('c', 'z'), ('p', ''), ('q', '')]

Enter first list length: 3
Enter second list length: 5
a
b
c
['a', 'b', 'c']
x
y
z
r
s
['x', 'y', 'z', 'r', 's']
[('a', 'x'), ('b', 'y'), ('c', 'z'), ('', 'r'), ('', 's')]
```

2) Find Element from list:

Program->

```
class FindEle:
    def init(self):
        lenn = int(input("enter the lenght: "))
        a = []
        output = []
        for _ in range(0, lenn):
            inp = tuple(input().split(" "))
            a.append(inp)
        key = input("enter the key: ")
        for i in a:
            if (key in i):
                ans = i
                output.append(ans)
        print(output)
FindEle.init(0)
```

Input/Output:

```
enter the lenght: 5
3,5
2,a
b,4
a,q
5,15
enter the key: a
[('2', 'a'), ('a', 'q')]
enter the lenght: 5
```

3,5

2,a

b,4

a,q

5,15

enter the key: a

[('3', '5'), ('5', '15')]

3) Remove duplicate Element:

Program->

```
class duplicateRemove:
    def init(self):
        lenn = int(input("enter the length: "))
        a = []
        for _ in range(0, lenn):
            inpt = tuple(input().split(" "))
            a.append(inpt)
        a = set(a)
        a = list(a)
        print(a)
duplicateRemove.init(0)
```

Input/Output:

enter the length: 5
1,2
4,7
6,8
6,8
1,2
[('4,7',), ('1,2',), ('6,8',)]

enter the length: 5
'a','v'
'x','z'
'b','g'
'a','v'
'p','q'
[("'b','g'",), ("'p','q'",), ("'x','z'",), ("'a','v'",)]

Python Assignment-3

1.write a program to find common element from the given array using set.

Program:

```
def Intersection(A, B, C):
    s1 = set(A)
    s2 = set(B)
    s3 = set(C)
    set1 = s1.intersection(s2)
    output_set = set1.intersection(s3)
    output_list = list(output_set)
    print(output_list)

if __name__ == '__main__' :
   A=list()
n=int(input("Enter the size of the List :"))
print("Enter the number  ")
for i in range(int(n)):
    p=input()
```

```
    A.append(p)
print (A)



B=list()
n1=int(input("Enter the size of the List: "))
print("Enter the number  ")
for i in range(int(n1)):
 p=input()
 B.append(p)
print (B)

C=list()
n2=int(input("Enter the size of the List:"))
print("Enter the number  ")
for i in range(int(n2)):
 p=input()
 C.append(p)
print (C)
```

```
print("Common element from of given array")
Intersection(A, B, C)
```

Input/Output:

1)%Run python_1.py

```
Enter the size of the List :3
Enter the number
a
b
c
['a', 'b', 'c']
Enter the size of the List: 3
Enter the number
c
b
d
```

```
['c', 'b', 'd']
Enter the size of the List:5
Enter the number
x
h
b
c
e
['x', 'h', 'b', 'c', 'e']
Common element from of given array
['b', 'c']
```

2)%Run python_1.py

```
Enter the size of the List3
Enter the number
1
2
3
['1', '2', '3']
Enter the size of the List4
```

Enter the number

1

2

5

6

['1', '2', '5', '6']

Enter the size of the List5

Enter the number

1

5

4

2

['1', '5', '4', '2', '']

Intersection of given array

['2', '1']

2.Implement program 1 using dictionary.

Program:

```
dict1 = {}
dict2 = {}
dict3 = {}
temp = []
common = []

dict1_length = int(input("Enter first dictionary length: "))
for i in range(dict1_length) :
    dict1_data = input()
    if(dict1_data.isdigit()) :
        dict1.update({i:int(dict1_data)})
    else :
        dict1.update({i:dict1_data})

dict2_length = int(input("Enter second dictionary length:
"))
for i in range(dict2_length) :
```

```
    dict2_data = input()
    if(dict2_data.isdigit()) :
        dict2.update({i:int(dict2_data)})
    else :
        dict2.update({i:dict2_data})

dict3_length = int(input("Enter third dictionary length: "))
for i in range(dict3_length) :
    dict3_data = input()
    if(dict3_data.isdigit()) :
        dict3.update({i:int(dict3_data)})
    else :
        dict3.update({i:dict3_data})

for i in dict1.values() :
    if i in dict2.values() :
        temp.append(i)
for j in temp :
    if j in dict3.values() :
```

```
        common.append(j)

print("Input: ")
print(list(dict1))
print(list(dict2))
print(list(dict3))

print("Output: ")
print(common)
```

Input/Output:

1)%Run python_2.py

```
Enter first dictionary length: 3
a
b
c
```

Enter second dictionary length: 3

b

c

d

Enter third dictionary length: 5

x

y

b

c

e

Input:

[0, 1, 2]

[0, 1, 2]

[0, 1, 2, 3, 4]

Output:

['b', 'c']

2)%Run python_2.py

Enter first dictionary length: 5

1

3

4

2

6

Enter second dictionary length: 5

2

4

6

8

9

Enter third dictionary length: 6

3

1

4

6

7

2

Input:

[0, 1, 2, 3, 4]

[0, 1, 2, 3, 4]

[0, 1, 2, 3, 4, 5]

Output:

[4, 2, 6]

3. Consider ‘reflection’ of an alphabet as explained here. ‘reflection’ of ‘a’ is ‘z’ , reflection of ‘b’ is ‘y’,……and so on . write a program to take a string S and index N as input and print reflection of the character is S starting from index N in the output string using dictionary.

Program:

```
def reflection(string,N):
```

```
original='abcdefghijklmnopqrstuvwxyzABCDEFGHIJK
LMNOPQRSTUVWXYZ'

reflective='zyxwvutsrqponmlkjihgfedcbaZYXWVUTS
RQPONMLKJIHGFEDCBA'
    dict1 = dict(zip(original,reflective))
    ch = string[N-1:]
    re = string[0:N-1]
    new = ""
    for i in range(0,len(ch)):
        new=new + dict1[ch[i]]
    print(re + new)
if __name__ == '__main__' :
    print("Enter a string::")
    string = input()
    print("Enter a index number you want to start
reflection::")
```

```
N = int(input())
reflection(string,N)
```

Input/Output:

1) %Run python_3.py

Enter a string:

aABbCcDd

Enter a index number you want to start reflection::

3

aAYyXxWw

2)%Run python_3.py

Enter a string::

ComPutEr

Enter a index number you want to start reflection::

2

ClnKfgVi

3)%Run python_3.py

Enter a string::

SaMPleINPUt

Enter a index number you want to start reflection::

1

HzNKovRMKFg

4. Write a program to perform below operation on two sets.

A. Union

B. Intersection

C. Difference

D. Symmetric difference

Program:

```
A=list()
B=list()
n=int(input("Enter the size of the List ::"))
print("Enter the Element of first list::")
for i in range(int(n)):
    k=int(input(""))
```

```
    A.append(k)
n1 = int(input("Enter the size of the List ::"))
print("Enter the Element of second list::")
for i in range(int(n1)):
    k=int(input(""))
    B.append(k)
C=set(list(A))
print(C)
D=set(list(B))
print(D)
print("Union :", C|D)
print("Intersection :", C&D)
print("Difference :", C - D)
print("Symmetric difference :", C ^ D)
```

<u>Input/Output:</u>

1) %Run python_4.py

```
 Enter the size of the List ::5
Enter the Element of first list::
1
2
3
4
5
Enter the size of the List ::4
Enter the Element of second list::
4
5
6
7
{1, 2, 3, 4, 5}
{4, 5, 6, 7}
Union : {1, 2, 3, 4, 5, 6, 7}
Intersection : {4, 5}
Difference : {1, 2, 3}
Symmetric difference : {1, 2, 3, 6, 7}
```

Python (Practical list 4)

1.Learn to Create 1-D, 2-Dand 3-D array using numpy in python.

Program:

```
import numpy as np

print("\n1-D array")
a = np.array([1,2,3,4,5,6])
print(a)

print("\n1-D array with arange function to interval ")
a1 = np.arange(2,12,3)
print(a1)

print("\n1-D array with linspace ")
a2 = np.linspace(1,10,4)
print(a2)


print("\n2-D array")
b = np.array([[1,2,3],[4,5,6]])
print(b)

print("\n2-D array with ones element ")
```

```
b1 = np.ones((2,2))
print(b1)

print("\n2-D array with random array ")
b2 = np.random.rand(2,2)
print(b2)

print("\n2-D array with identity")
b3 = np.identity(2)
print(b3)



print("\n3-D array")
c = np.array([[[1,2,3],
               [4,5,6]],
              [[7,8,9],
               [10,11,12]]])
print(c)

print("\n3-D array with zeros ")
c1 = np.zeros((2,3,4))
print(c1)
```

2.Perform mathematical operations on 2-D arrays like addition, multiplication, subtraction and division using a single integer(b = a+5). Perform the same operations using two arrays as operands(c = a*b).

Program:

```
import numpy as np


b = np.array([[1,2,3],[4,5,6]])


print("2-D array of ..\n",b)
print("Addition of array..\n",b+5)
print("Multiplication of array..\n",b*5)
print("Subtraction of array..\n",b-5)
print("Division of array..\n",b / 5)
```

```
print(" Two element using operation ")
a = np.array([[10,20,30],[40,50,60]])
print("2-D array of a",a)
print("Addition of array..\n",a+b)
print("Multiplication of array..\n",a*b)
print("Subtraction of array..\n",a-b)
print("Division of array..\n",a/b)
```

3.Perform different relational operations like >,>=,<,<=,== and !=on two arrays

Program:

```
a = input("Enter first element ")
b = input("Enter second element ")


print("Relation operation on array")
```

```
print("Result of a>b")
print(a>b)
print("Result of a>=b")
print(a>=b)
print("Result of a<=b")
print(a<=b)
print("Result of a==b")
print(a==b)
print("Result of a!=b")
print(a!=b)
```

4.Perform operations like view(), copy(), shape(), reshape(), flatten() on arrays

Program:

```
import numpy as np
```

```
print("Copy array operation ")
a = np.array([1,2,3])
copy1 = a.copy()
a[2] = 10
print(a)
print(copy1)

print("View array operation ...It's one type of cloning")
view1 = a.view()
a[2] = 10
print(a)
print(view1)


print("Shape array operation ")
arr1 = np.array([[1, 2, 3, 4], [5, 6, 7, 8]])
print("Shape of array :",arr1.shape)


print("Reshape array operation ")
arr = np.array([1,2,3,4,5,6,7,8,9,10,11,12])
reshape_array = arr.reshape(3,4)
reshape_array1 = arr.reshape(3,2,2)
```

```
print("2-D array..\n",reshape_array)
print("3-D array ..\n",reshape_array1)
```

```
print("Flatten array operation..")
print("Given array\n",reshape_array1)
print("Flatten array\n ", reshape_array1.flatten())
```

```
print("Flatten array operation.. example ")
array = np.arange(15).reshape(3, 5)
print("Given array\n",array)
print("Flatten array \n", array.flatten())
```

5.Create matrix in numpy and perform operations like transpose, sort, addition, multiplication, find maximum and minimum element and find sum, average and product of all elements in the matrix.

Program:

```
import numpy as np
print("Matrix operation ..\n")
matrix = np.array( [ [ 4, 5, 6 ],
                     [ 7, 8, 9 ],
                     [ 10, 11, 12 ] ] )
matrix1 = np.array( [ [ 1, 2, 3 ],
                     [ 4, 5, 6 ],
                     [ 7, 8, 9 ] ] )
mat = np.random.rand(4)
print(matrix)
print(matrix1)
print("\nTranspot of matrix\n",matrix.transpose())
print("Array of random ",mat)
mat.sort()
print("\n sort all element from matrix\n",mat)
print("\nAddition of two matrix \n",matrix+matrix1)
print("\nMultiplication of two matrix \n",matrix*matrix1)
print("\nMinimum element from matrix \n",matrix.max())
print("\nMultiplication of from matrix \n",matrix.min())
```

```
print("\nSum of the matrix \n",matrix.sum())
print("\nAverage of matrix \n",matrix.mean())
print("\nProduct of matrix \n",matrix.prod())
```

Python for Data Science (Lab Session-14 - 1)

In [ ]:

Working with array and matrix

Q-1) NumPy array manipulation

A => Creare a Numpy Array of dimension 5 x 4 using random number generator.

Explore about different distributions of random numbers

In [2]:
```
from numpy import random
import numpy as np
x = random.randint(100, size=(5, 4))

print(x)
```

```
[[74 30  2 79]
 [81 33 62 57]
 [33  8  2 62]
 [24 55 56 91]
 [30 81 30  2]]
```

B => Get the values of attributes like ndim, shape, size, itemsize, dtype, nbytes

In [2]:
```
#dimension
print(x.ndim)
```

```
2
```

In [3]:
```
#Shape of data
print(x.shape)
```

```
(5, 4)
```

In [4]:
```
#Size of it
print(x.size)
```

```
20
```

In [5]:
```
# item size or give column
print(x.itemsize)
```

```
4
```

In [6]:
```
#data type
print(x.dtype)
```

```
int32
```

In [7]:
```
#Give total byte
print(x.nbytes)
```

```
80
```

C => Sort the array using 2nd row and then sort it again using 4th column

In [8]:
```
snd_row = x[:,x[1].argsort()]
snd_row
# second tow sort
```

Out[8]:
```
array([[59, 35,  7, 18],
       [23, 24, 43, 67],
       [79,  4, 11, 13],
       [63, 90, 71, 30],
       [81, 10, 44,  7]])
```

In [9]:
```
col4 = snd_row[np.argsort(snd_row[:,3])]
col4
```

Out[9]:
```
array([[81, 10, 44,  7],
       [79,  4, 11, 13],
       [59, 35,  7, 18],
       [63, 90, 71, 30],
       [23, 24, 43, 67]])
```

D => Perform slicing of rows and columns and dicing

In [10]:
```
print("Extract Second Row to End\n")
print(x[2:])
```

```
Extract Second Row to End

[[ 4 11 79 13]
 [90 71 63 30]
 [10 44 81  7]]
```

In [11]:
```
print("Extract Second Row and All Column\n")
print(x[1])
```

```
Extract Second Row and All Column

[24 43 23 67]
```

In [12]:
```
print("Extract All Row with Second column.")
x[:,[2]]
```

```
Extract All Row with Second column.
```

Out[12]:
```
array([[59],
       [23],
       [79],
       [63],
       [81]])
```

In [13]:
```
print("Extract Third Row of fourth element")
x[3:4,[3]]
```

```
Extract Third Row of fourth element
```

Out[13]:
```
array([[30]])
```

E => Perform searching of any element from the array

In [3]:
```
result = np.where(x==74)
print("Row Index",result[0],"And Column Index",result[1],"Found Element 74")
```

```
Row Index [0] And Column Index [0] Found Element 74
```

In [ ]:

In [ ]:

Python Lab session - 5

(Q-1) Explain Python Ecosystem for Data Science.

```
===>> For the Data Science ,We know given librery is used for his functionality
or easy to used.
    That's why going futher one by one lib.

    1) Numpy  --> It is stand for Numric Python.
                NumPy is a python library used for working with arrays.
                It also has functions for working in domain of linear
               algebra, fourier transform, and matrices.
```

In [1]:
```
import numpy as np
a = np.array([1, 2, 3], dtype = complex)
print(a)
```

```
[1.+0.j 2.+0.j 3.+0.j]
```

```
      2) Pandas -->Pandas is an open-source library that is built on top of
NumPy library. It is a Python package that offers various data structures and
operations for manipulating numerical data and time series. It is mainly
popular for importing and analyzing data much easier. Pandas is fast and it has
high-performance & productivity for users.
```

In [3]:
```
import pandas as pd
import numpy as np
data = np.array(['a','b','c','d'])
s = pd.Series(data)
print(s)
```

```
0    a
1    b
2    c
3    d
dtype: object
```

In [4]:
```
data = pd.read_csv('https://archive.ics.uci.edu/ml/machine-learning-databases/iri
data.head()
```

Out[4]:

	sepal_length	sepal_width	petal_length	petal_width	class
0	5.1	3.5	1.4	0.2	Iris-setosa
1	4.9	3.0	1.4	0.2	Iris-setosa
2	4.7	3.2	1.3	0.2	Iris-setosa
3	4.6	3.1	1.5	0.2	Iris-setosa
4	5.0	3.6	1.4	0.2	Iris-setosa

```
     3)Matplotlib  -->Matplotlib is one of the most popular Python packages
used for data visualization. It is a cross-platform library for making 2D plots
from data in arrays. It provides an object-oriented API that helps in embedding
plots in applications using Python GUI toolkits such as PyQt,
WxPythonotTkinter. It can be used in Python and IPython shells, Jupyter
notebook and web application servers also.
```

In [5]:
```
import matplotlib.pyplot as plt

data = {'apple': 10, 'orange': 15, 'lemon': 5, 'lime': 20}
names = list(data.keys())
values = list(data.values())

fig, axs = plt.subplots(1, 3, figsize=(9, 3), sharey=True)
axs[0].bar(names, values)
axs[1].scatter(names, values)
axs[2].plot(names, values)
fig.suptitle('Categorical Plotting')
```

Out[5]: Text(0.5, 0.98, 'Categorical Plotting')

4) Seaborn --> Seaborn is a Python data visualization library based on matplotlib. It provides a high-level interface for drawing attractive and informative statistical graphics. For a brief introduction to the ideas behind the library, you can read the introductory notes. Visit the installation page to see how you can download the package.

In [6]:
```
import seaborn as sns
sns.set(style="ticks")

df = sns.load_dataset("iris")
sns.pairplot(df, hue="species")
```

Out[6]: <seaborn.axisgrid.PairGrid at 0x29a29e35198>

5) Sckit-Learn --> Scikit-learn (Sklearn) is the most useful and robust library for machine learning in Python. It provides a selection of efficient tools for machine learning and statistical modeling including classification, regression, clustering and dimensionality reduction via a consistence interface in Python.

In [8]:
```
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
feature_names = iris.feature_names
target_names = iris.target_names
print("Feature names:", feature_names)
print("Target names:", target_names)
print("\nFirst 10 rows of X:\n", X[:10])  # 30% for Testing and 70% for training

X_train, X_test, y_train, y_test = train_test_split(
   X, y, test_size = 0.3, random_state = 1
)

print(X_train.shape)
print(X_test.shape)

print(y_train.shape)
print(y_test.shape)
```

```
Feature names: ['sepal length (cm)', 'sepal width (cm)', 'petal length (cm)',
'petal width (cm)']
Target names: ['setosa' 'versicolor' 'virginica']

First 10 rows of X:
 [[5.1 3.5 1.4 0.2]
 [4.9 3.  1.4 0.2]
 [4.7 3.2 1.3 0.2]
 [4.6 3.1 1.5 0.2]
 [5.  3.6 1.4 0.2]
 [5.4 3.9 1.7 0.4]
 [4.6 3.4 1.4 0.3]
 [5.  3.4 1.5 0.2]
 [4.4 2.9 1.4 0.2]
 [4.9 3.1 1.5 0.1]]
(105, 4)
(45, 4)
(105,)
(45,)
```

In [12]:
```
classifier_knn = KNeighborsClassifier(n_neighbors = 3)
classifier_knn.fit(X_train, y_train)
y_pred = classifier_knn.predict(X_test)
# Finding accuracy by comparing actual response values(y_test)with predicted resp
print("Accuracy:", metrics.accuracy_score(y_test, y_pred)*100)
# Providing sample data and the model will make prediction out of that data
```

```
Accuracy: 97.77777777777777
```

```
    6)Keras  --> Keras is an open-source neural-network library written in
Python. It is capable of running on top of TensorFlow, Microsoft Cognitive
Toolkit, R, Theano, or PlaidML. Designed to enable fast experimentation with
deep neural networks, it focuses on being user-friendly, modular, and
extensible.
```

```
   7)Tensorflow  -->TensorFlow is an open source machine learning framework for
all developers. It is used for implementing machine learning and deep learning
applications. To develop and research on fascinating ideas on artificial
intelligence, Google team created TensorFlow. TensorFlow is designed in Python
programming language, hence it is considered an easy to understand framework.
```

In [ ]:

```

```

(Q-2) Expain program in jupyter and perform it

1)Count the vowel in given string

In [13]:

```
class Vowelchar():
     def vowel(self):
         vowels = ('a', 'A', 'e', 'E', 'i', 'I', 'o', 'O', 'u', 'U')
         string = input("Enter the string:")
         output = {}
         for char in vowels:
            x = string.count(char)
            output[char] = x
         print(output)

Vowelchar().vowel()
```

```
Enter the string:This is Dummy String
{'a': 0, 'A': 0, 'e': 0, 'E': 0, 'i': 3, 'I': 0, 'o': 0, 'O': 0, 'u': 1, 'U':
0}
```

2) Password Validation ...

In [14]:
```
def check(passwd):
    val = True

    if len(passwd) < 9:
        print('length should be at least 9')
        val = False

    if not any(char.isdigit() for char in passwd):
        print('Password should have at least one numeral')
        val = False

    if not any(char.isupper() for char in passwd):
        print('Password should have at least one uppercase letter')
        val = False

    if not any(char.islower() for char in passwd):
        print('Password should have at least one lowercase letter')
        val = False

    Specialchar =['', '%', '#', '$']
    if not any(char in Specialchar for char in passwd):
        print('Password should have at least one of the symbols _ Or % Or * Or $'
        val = False
    if val:
        return val

print("Enter password ...")
passwd = input()
if (check(passwd)):
        print("Password Accepted")
else:
        print("Invalid Password !!")
```

```
Enter password ...
M!croSoft$987
Password Accepted
```

3) Caesar cipher encryption:

In [15]:
```
def Encryption(s,k):
    encstr=""
    for i in s:
        if(ord(i))>=65 and (ord(i)<=90):
            temp=(ord(i)+k)
            if temp>90:
                temp=temp%90+64
            encstr=encstr+chr(temp)
        elif(ord(i))>=97 and (ord(i)<=122):
            temp=(ord(i)+k)
            if temp>122:
                temp=temp%122+96
            encstr=encstr+chr(temp)
        else:
            encstr=encstr+chr(ord(i)+k)
    return encstr
def Decryption(k):
    p=Encryption(s,k)
    decstr=""
    for i in p:
        if((ord(i))>=65) and (ord(i))<=90:
            decstr=decstr+chr((ord(i) - k-65) % 26 + 65)
        elif((ord(i))>=97) and (ord(i))<=122:
            decstr=decstr+chr((ord(i) - k - 97) % 26 + 97)
        else:
            decstr=decstr+chr(ord(i)-k)
    return decstr
print("Enter the string to Encrypt and decrypt : ")
s=input()
k=9
k=k%26
print("Encrypted String : ",Encryption(s,k))
print("Decrypted String : ",Decryption(k))
```

```
Enter the string to Encrypt and decrypt :
This is DUMMY String
Encrypted String :  Cqrb)rb)MDVVH)Bcarwp
Decrypted String :  This is DUMMY String
```

4) Numpy using Example

In [1]:
```
import numpy as np


a = np.arange(20)
print(a)
print(a[8:12])
print(a[1:100:3])


cosvalue = np.cos(4)
print("Cos value" ,cosvalue)
sinvalue = np.sin(4)
print("Cos value" ,sinvalue)


print(np.random.rand())
print(np.random.ranf())


print("\n1-D array")
a = np.array([1,2,3,4,5,6])
print(a)

print("\n1-D array with arange function to interval ")
a1 = np.arange(2,12,3)
print(a1)

print("\n1-D array with linspace ")
a2 = np.linspace(1,10,4)
print(a2)


print("\n2-D array")
b = np.array([[1,2,3],[4,5,6]])
print(b)

print("\n2-D array with ones element ")
b1 = np.ones((2,2))
print(b1)

print("\n2-D array with random array ")
b2 = np.random.rand(2,2)
print(b2)

print("\n2-D array with identity")
b3 = np.identity(2)
print(b3)


print("\n3-D array")
c = np.array([[[1,2,3],
               [4,5,6]],
              [[7,8,9],
               [10,11,12]]])
print(c)
```

```
print("\n3-D array with zeros ")
c1 = np.zeros((2,3,4))
print(c1)


print("B\n", np.linspace(1.0, 3.0, num=5, retstep=True), "\n")



arr = np.array([[14, 70], [42, 60]],
                 dtype = np.float64)

print(c.dtype)

print("\nAddition of Array elements: ")
print(np.sum(arr))

print("\nSquare root of Array1 elements: ")
print(np.sqrt(arr))

print("\nTranspose of Array: ")
print(arr.T)


# Printing type of arr object
print("Array is of type: ", type(arr))

# Printing array dimensions (axes)
print("No. of dimensions: ", arr.ndim)

# Printing shape of array
print("Shape of array: ", arr.shape)

# Printing size (total number of elements) of array
print("Size of array: ", arr.size)

# Printing type of elements in array
print("Array stores elements of type: ", arr.dtype)  # complex
```

```
[ 0  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19]
[ 8  9 10 11]
[ 1  4  7 10 13 16 19]
Cos value -0.6536436208636119
Cos value -0.7568024953079282
0.37463574140319633
0.8037711756216795

1-D array
[1 2 3 4 5 6]

1-D array with arange function to interval
[ 2  5  8 11]
```

```
1-D array with linspace
[ 1.  4.  7. 10.]

2-D array
[[1 2 3]
 [4 5 6]]

2-D array with ones element
[[1. 1.]
 [1. 1.]]

2-D array with random array
[[0.6169639  0.50275067]
 [0.23072728 0.49661341]]

2-D array with identity
[[1. 0.]
 [0. 1.]]

3-D array
[[[ 1  2  3]
  [ 4  5  6]]

 [[ 7  8  9]
  [10 11 12]]]

3-D array with zeros
[[[0. 0. 0. 0.]
  [0. 0. 0. 0.]
  [0. 0. 0. 0.]]

 [[0. 0. 0. 0.]
  [0. 0. 0. 0.]
  [0. 0. 0. 0.]]]
B
 (array([1. , 1.5, 2. , 2.5, 3. ]), 0.5)

int32

Addition of Array elements:
186.0

Square root of Array1 elements:
[[3.74165739 8.36660027]
 [6.4807407  7.74596669]]

Transpose of Array:
[[14. 42.]
 [70. 60.]]
Array is of type:  <class 'numpy.ndarray'>
No. of dimensions:  2
Shape of array:  (2, 2)
Size of array:  4
Array stores elements of type:  float64
```

In [2]:
```
import numpy as np

print("Copy array operation ")
a = np.array([1,2,3])
copy1 = a.copy()
a[2] = 10
print(a)
print(copy1)

print("View array operation ...It's one type of cloning")
view1 = a.view()
a[2] = 10
print(a)
print(view1)


print("Shape array operation ")
arr1 = np.array([[1, 2, 3, 4], [5, 6, 7, 8]])
print("Shape of array :",arr1.shape)


print("Reshape array operation ")
arr = np.array([1,2,3,4,5,6,7,8,9,10,11,12])
reshape_array = arr.reshape(3,4)
reshape_array1 = arr.reshape(3,2,2)
print("2-D array..\n",reshape_array)
print("3-D array ..\n",reshape_array1)


print("Flatten array operation..")
print("Given array\n",reshape_array1)
print("Flatten array\n ", reshape_array1.flatten())

print("Flatten array operation.. example ")
array = np.arange(15).reshape(3, 5)
print("Given array\n",array)
print("Flatten array \n", array.flatten())
```

```
Copy array operation
[ 1  2 10]
[1 2 3]
View array operation ...It's one type of cloning
[ 1  2 10]
[ 1  2 10]
Shape array operation
Shape of array : (2, 4)
Reshape array operation
2-D array..
 [[ 1  2  3  4]
 [ 5  6  7  8]
 [ 9 10 11 12]]
3-D array ..
 [[[ 1  2]
  [ 3  4]]
```

```
 [[ 5  6]
  [ 7  8]]

 [[ 9 10]
  [11 12]]]
Flatten array operation..
Given array
 [[[ 1  2]
  [ 3  4]]

 [[ 5  6]
  [ 7  8]]

 [[ 9 10]
  [11 12]]]
Flatten array
  [ 1  2  3  4  5  6  7  8  9 10 11 12]
Flatten array operation.. example
Given array
 [[ 0  1  2  3  4]
 [ 5  6  7  8  9]
 [10 11 12 13 14]]
Flatten array
 [ 0  1  2  3  4  5  6  7  8  9 10 11 12 13 14]
```

In [3]:
```
import numpy as np
print("Matrix operation ..\n")
matrix = np.array( [ [ 4, 5, 6 ],
                     [ 7, 8, 9 ],
                     [ 10, 11, 12 ] ] )
matrix1 = np.array( [ [ 1, 2, 3 ],
                     [ 4, 5, 6 ],
                     [ 7, 8, 9 ] ] )

mat = np.random.rand(4)

print(matrix)
print(matrix1)
print("\nTranspot of matrix\n",matrix.transpose())
print("Array of random ",mat)
mat.sort()
print("\n sort all element from matrix\n",mat)
print("\nAddition of two matrix \n",matrix+matrix1)
print("\nMultiplication of two matrix \n",matrix*matrix1)
print("\nMinimum element from matrix \n",matrix.max())
print("\nMultiplication of from matrix \n",matrix.min())
print("\nSum of the matrix \n",matrix.sum())
print("\nAverage of matrix \n",matrix.mean())
print("\nProduct of matrix \n",matrix.prod())
```

```
Matrix operation ..

[[ 4  5  6]
 [ 7  8  9]
 [10 11 12]]
[[1 2 3]
 [4 5 6]
 [7 8 9]]

Transpot of matrix
 [[ 4  7 10]
 [ 5  8 11]
 [ 6  9 12]]
Array of random  [0.56406907 0.84831072 0.35068213 0.2041597 ]

 sort all element from matrix
 [0.2041597  0.35068213 0.56406907 0.84831072]

Addition of two matrix
 [[ 5  7  9]
 [11 13 15]
 [17 19 21]]

Multiplication of two matrix
 [[  4  10  18]
 [ 28  40  54]
 [ 70  88 108]]
```

```
Minimum element from matrix
 12

Multiplication of from matrix
 4

Sum of the matrix
 72

Average of matrix
 8.0

Product of matrix
 79833600
```

In [ ]:

Working with Dataset

1. Import the load_iris dataset

Analyze the dataset and print below values. Learn what is the me values.

In [1]:
```
import numpy as np
import pandas as pd
from sklearn.datasets import load_iris
df=load_iris()
```

In [2]:
```
df  #It's load data frame (iris _flower)
```

```
m\n        - class:\n                - Iris-Setosa\n
rsicolour\n                - Iris-Virginica\n
tatistics:\n\n    ============== ==== ==== ======= =====
\n                    Min  Max   Mean    SD   Class Corr
====== ==== ==== ======= ===== ====================\n
7.9   5.84   0.83    0.7826\n    sepal width:    2.0  4.
4194\n    petal length:   1.0  6.9   3.76   1.76    0.94
l width:    0.1  2.5   1.20   0.76    0.9565  (high!)\n
= ==== ======= ===== ====================\n\n    :Missir
one\n    :Class Distribution: 33.3% for each of 3 classe
A. Fisher\n    :Donor: Michael Marshall (MARSHALL%PLU@ic
Date: July, 1988\n\nThe famous Iris database, first usec
The dataset is taken\nfrom Fisher\'s paper. Note that it
but not as in the UCI\nMachine Learning Repository, whic
points.\n\nThis is perhaps the best known database to be
n recognition literature.  Fisher\'s paper is a classic
referenced frequently to this day.  (See Duda & Hart, fc
a set contains 3 classes of 50 instances each, where eac
type of iris plant.  One class is linearly separable frc
atter are NOT linearly separable from each other.\n\n..
```

In [3]:
```
print(df.data)    #Only print it's data value
```

```
[[5.1 3.5 1.4 0.2]
 [4.9 3.  1.4 0.2]
 [4.7 3.2 1.3 0.2]
 [4.6 3.1 1.5 0.2]
 [5.  3.6 1.4 0.2]
 [5.4 3.9 1.7 0.4]
 [4.6 3.4 1.4 0.3]
 [5.  3.4 1.5 0.2]
 [4.4 2.9 1.4 0.2]
 [4.9 3.1 1.5 0.1]
 [5.4 3.7 1.5 0.2]
 [4.8 3.4 1.6 0.2]
 [4.8 3.  1.4 0.1]
 [4.3 3.  1.1 0.1]
 [5.8 4.  1.2 0.2]
 [5.7 4.4 1.5 0.4]
 [5.4 3.9 1.3 0.4]
 [5.1 3.5 1.4 0.3]
 [5.7 3.8 1.7 0.3]
```

In [4]:
```
df.data.shape  #That is shape or dimention of data model
```

Out[4]:
```
(150, 4)
```

In [5]:
```
df.feature_names       # Here by , Total 4 column which
```

Out[5]:
```
['sepal length (cm)',
 'sepal width (cm)',
 'petal length (cm)',
 'petal width (cm)']
```

In [6]:
```
df.target_names    # this respresent the Three diffrent
```

Out[6]:
```
array(['setosa', 'versicolor', 'virginica'], dtype='<U16
```

In [7]:
```
df.data  # It's data given by obesarvation
```

```
       [6.1, 3. , 4.9, 1.8],
       [6.4, 2.8, 5.6, 2.1],
       [7.2, 3. , 5.8, 1.6],
       [7.4, 2.8, 6.1, 1.9],
       [7.9, 3.8, 6.4, 2. ],
       [6.4, 2.8, 5.6, 2.2],
       [6.3, 2.8, 5.1, 1.5],
       [6.1, 2.6, 5.6, 1.4],
       [7.7, 3. , 6.1, 2.3],
       [6.3, 3.4, 5.6, 2.4],
       [6.4, 3.1, 5.5, 1.8],
       [6. , 3. , 4.8, 1.8],
       [6.9, 3.1, 5.4, 2.1],
       [6.7, 3.1, 5.6, 2.4],
       [6.9, 3.1, 5.1, 2.3],
       [5.8, 2.7, 5.1, 1.9],
       [6.8, 3.2, 5.9, 2.3],
       [6.7, 3.3, 5.7, 2.5],
       [6.7, 3. , 5.2, 2.3],
       [6.3, 2.5, 5. , 1.9],
```

In [8]:
```
df.target          # Target value in numpy as given bel
```

Out[8]:
```
array([0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0
       0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2, 2, 2, 2, 2
       2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2
       2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2
```

In [9]:
```
df.DESCR     # Description of DATA
```

Out[9]:
```
'.. _iris_dataset:\n\nIris plants dataset\n----------
haracteristics:**\n\n    :Number of Instances: 150 (50 i
s)\n    :Number of Attributes: 4 numeric, predictive att
:Attribute Information:\n        - sepal length in cm\n
cm\n        - petal length in cm\n        - petal width
s:\n                - Iris-Setosa\n                - Iri
- Iris-Virginica\n                \n    :Summary Statist
=== ==== ==== ======= ===== ====================\n
Mean    SD   Class Correlation\n    ============== ====
================\n    sepal length:   4.3  7.9   5.84
al width:    2.0  4.4   3.05   0.43   -0.4194\n    petal
3.76   1.76    0.9490  (high!)\n    petal width:    0.1
0.9565  (high!)\n    ============== ==== ==== ======= ==
=\n\n    :Missing Attribute Values: None\n    :Class Dis
ch of 3 classes.\n    :Creator: R.A. Fisher\n    :Donor:
HALL%PLU@io.arc.nasa.gov)\n    :Date: July, 1988\n\nThe
irst used by Sir R.A. Fisher. The dataset is taken\nfrom
that it\'s the same as in R, but not as in the UCI\nMach
y, which has two wrong data points.\n\nThis is perhaps t
to be found in the\npattern recognition literature.  Fis
ic in the field and\nis referenced frequently to this da
or example.)  The\ndata set contains 3 classes of 50 ins
class refers to a\ntype of iris plant.  One class is lir
e other 2; the\nlatter are NOT linearly separable from e
c:: References\n\n   - Fisher, R.A. "The use of multiple
mic problems"\n     Annual Eugenics, 7, Part II, 179-188
ibutions to\n     Mathematical Statistics" (John Wiley,
R.O., & Hart, P.E. (1973) Pattern Classification and Sce
7.D83) John Wiley & Sons.  ISBN 0-471-22361-1.  See page
B.V. (1980) "Nosing Around the Neighborhood: A New Syste
lassification Rule for Recognition in Partially Exposed\
EEE Transactions on Pattern Analysis and Machine\n     I
2, No. 1, 67-71.\n   - Gates, G.W. (1972) "The Reduced N
IEEE Transactions\n     on Information Theory, May 1972,
o: 1988 MLC Proceedings, 54-64.  Cheeseman et al"s AUTOC
l clustering system finds 3 classes in the data.\n   - M
```

In [ ]:

2. Write a logic to Filter the rows of iris data that h
and sepallength < 5.0

In [10]:
```
for i in df.data:
    for j in i:
      if i[0] < 5 and i[2] >1.5:
            print(i)
print("\n")
```

```
[4.8 3.4 1.6 0.2]
[4.8 3.4 1.6 0.2]
[4.8 3.4 1.6 0.2]
[4.8 3.4 1.6 0.2]
[4.8 3.4 1.9 0.2]
[4.8 3.4 1.9 0.2]
[4.8 3.4 1.9 0.2]
[4.8 3.4 1.9 0.2]
[4.7 3.2 1.6 0.2]
[4.7 3.2 1.6 0.2]
[4.7 3.2 1.6 0.2]
[4.7 3.2 1.6 0.2]
[4.8 3.1 1.6 0.2]
[4.8 3.1 1.6 0.2]
[4.8 3.1 1.6 0.2]
[4.8 3.1 1.6 0.2]
[4.9 2.4 3.3 1. ]
[4.9 2.4 3.3 1. ]
[4.9 2.4 3.3 1. ]
[4.9 2.4 3.3 1. ]
[4.9 2.5 4.5 1.7]
[4.9 2.5 4.5 1.7]
[4.9 2.5 4.5 1.7]
[4.9 2.5 4.5 1.7]
```

3. Calculate the mean, median and standard deviat

In [11]:
```
Mean = np.mean(df.data)
Mean                        #Mean of data
```

Out[11]: 3.4644999999999997

In [12]:
```
Median = np.median(df.data)
Median                   #Median of data
```

Out[12]: 3.2

In [13]:
```
STd = np.std(df.data)
STd                     #Starndard deviation
```

Out[13]: 1.9738430577598278

4. Use the petal length (3rd) column of iris data to categorical data, such that if petal length is:

a. Less than 3 --> 'Small'

b. Between 3to 5 --> 'Medium'

c. Greater than 5 --> 'Large'

In [14]:
```
for i in df.data:
    for j in i:
        if i[2] < 3 :
            print("Smalller : {} ".format(i[2]))
        elif i[2] >= 3 and i[2] <= 5 :
            print("Medium : {} ".format(i[2]))
        else :
            print("Larger : {}".format(i[2]))
```

```
Larger : 5.1
Larger : 5.1
Larger : 5.1
Larger : 5.1
Larger : 5.9
Larger : 5.9
Larger : 5.9
Larger : 5.9
Larger : 5.7
Larger : 5.7
Larger : 5.7
Larger : 5.7
Larger : 5.2
Larger : 5.2
Larger : 5.2
Larger : 5.2
Medium : 5.0
Medium : 5.0
Medium : 5.0
```

In [ ]:

Python for Data Science LAB Session

->Working with Dataset using pandas

In [1]:
```
import numpy as np
import pandas as pd
```

Q-1) Load the data from elements.txt, titanic.csv, Values.xls into pandas datafram

In [2]:
```
txt = pd.read_fwf('elements.txt')   # fixed width file
txt
```

Out[2]:

	A_Wght\tSymbol\tName
0	1.008\tH\tHydrogen
1	4.003\tHe\tHelium
2	6.941\tLi\tLithium
3	9.012\tBe\tBeryllium
4	10.811\tB\tBoron
5	12.011\tC\tCarbon
6	14.007\tN\tNitrogen
7	15.999\tO\tOxygen
8	18.998\tF\tFluorine
9	20.180\tNe\tNeon

In [3]:
```
txt1 = pd.read_table('elements.txt')  # table format
txt1
```

Out[3]:

	A_Wght	Symbol	Name
0	1.008	H	Hydrogen
1	4.003	He	Helium
2	6.941	Li	Lithium
3	9.012	Be	Beryllium
4	10.811	B	Boron
5	12.011	C	Carbon
6	14.007	N	Nitrogen
7	15.999	O	Oxygen
8	18.998	F	Fluorine
9	20.180	Ne	Neon

In [4]:
```
df = pd.read_csv('titanic.csv')  #csv file for comma seprated value
df
```

Out[4]:

	Unnamed: 0	pclass	survived	sex	age	sibsp	parch
0	1	1st	survived	female	29.0000	0	0
1	2	1st	survived	male	0.9167	1	2
2	3	1st	died	female	2.0000	1	2
3	4	1st	died	male	30.0000	1	2
4	5	1st	died	female	25.0000	1	2
...	...	...	...	...	...	...	...
1304	1305	3rd	died	female	14.5000	1	0
1305	1306	3rd	died	female	9999.0000	1	0
1306	1307	3rd	died	male	26.5000	0	0
1307	1308	3rd	died	male	27.0000	0	0
1308	1309	3rd	died	male	29.0000	0	0

1309 rows × 7 columns

In [5]:
```
data = pd.read_excel('Values.xls')
data
```

Out[5]:

	Angle (Degrees)	Sine	Cosine	Tangent
0	138.550574	0.661959	-0.749540	-0.883153
1	305.535745	-0.813753	0.581211	-1.400100
2	280.518695	-0.983195	0.182556	-5.385709
3	216.363795	-0.592910	-0.805269	0.736289
4	36.389247	0.593268	0.805005	0.736974
...	...	...	...	...
67	324.199562	-0.584964	0.811059	-0.721234
68	187.948172	-0.138277	-0.990394	0.139619
69	270.678249	-0.999930	0.011837	-84.472139
70	270.779159	-0.999908	0.013598	-73.530885
71	200.213513	-0.345520	-0.938412	0.368196

72 rows × 4 columns

Q-2) Load the image Colorblk.jpg, check the size of original image, convert it to the 1D data and check the size now. Resize the image to 50x50 grey-scale image, convert to 1D and check the size of it.

In [6]:
```
from PIL import Image

img = Image.open('colorblk.jpg')


print(img.size) # find the size of image
```

```
(100, 100)
```

In [7]:
```
import matplotlib.image as img
import matplotlib.pyplot as plt
from skimage.io import imread
from skimage.transform import resize

# Read Images
img1 = img.imread('colorblk.jpg')

plt.imshow(img1)
print(img1.shape)  # size of image



I1 = img1.flatten()
print(I1.shape)  # shape of imahe


# resize imag in  50*50
resize1 = resize(img1, (50,50))  # resize image
plt.imshow(resize1)
print(resize1.shape)
```

```
(100, 100, 3)
(30000,)
(50, 50, 3)
```

In [8]:
```
from skimage.color import rgb2gray

gray  = rgb2gray(img1)
plt.imshow(gray, cmap = 'gray')   # give color from cmap =>
#plt.imshow(gray, cmap='Greys_r')

print(gray.shape)
gray = rgb2gray(gray)
gray = gray.flatten()  # reshape in 1d array
print(gray.shape)
```

```
(100, 100)
(10000,)
```

```
c:\users\arjun vankani\appdata\local\programs\python\python37\lib\site-packag
es\ipykernel_launcher.py:8: FutureWarning: The behavior of rgb2gray will chan
ge in scikit-image 0.19. Currently, rgb2gray allows 2D grayscale image to be
passed as inputs and leaves them unmodified as outputs. Starting from version
0.19, 2D arrays will be treated as 1D images with 3 channels.
```

3. From titanic.csv, find the number of male passengers who were travelling in 2nd class that survived.

In [9]:
```
df
```

Out[9]:

	Unnamed: 0	pclass	survived	sex	age	sibsp	parch
0	1	1st	survived	female	29.0000	0	0
1	2	1st	survived	male	0.9167	1	2
2	3	1st	died	female	2.0000	1	2
3	4	1st	died	male	30.0000	1	2
4	5	1st	died	female	25.0000	1	2
...	...	...	...	...	...	...	...
1304	1305	3rd	died	female	14.5000	1	0
1305	1306	3rd	died	female	9999.0000	1	0
1306	1307	3rd	died	male	26.5000	0	0
1307	1308	3rd	died	male	27.0000	0	0
1308	1309	3rd	died	male	29.0000	0	0

1309 rows × 7 columns

In [10]:
```
# find the 2nd class passenger
class_2 = df["pclass"]=="2nd"
df=df.loc[class_2]      # c1 => class_2 a => df  c2 = c_2 c=>df1   c3 => c_2nd
b = df_final


# which are survided male
c_2=df["survived"]=="survived"
df1 = df.loc[c_2]
c_2nd = df1["sex"]=="male"
df_final=df1.loc[c_2nd]
df_final
```

Out[10]:

	Unnamed: 0	pclass	survived	sex	age	sibsp	parch
336	337	2nd	survived	male	32.0000	1	0
339	340	2nd	survived	male	1.0000	2	1
343	344	2nd	survived	male	34.0000	0	0
359	360	2nd	survived	male	0.8333	0	2
360	361	2nd	survived	male	26.0000	1	1
376	377	2nd	survived	male	24.0000	0	0
385	386	2nd	survived	male	8.0000	1	1
398	399	2nd	survived	male	8.0000	0	2
427	428	2nd	survived	male	0.6667	1	1
432	433	2nd	survived	male	62.0000	0	0
454	455	2nd	survived	male	42.0000	0	0
492	493	2nd	survived	male	1.0000	0	2
503	504	2nd	survived	male	19.0000	0	0
514	515	2nd	survived	male	2.0000	1	1
515	516	2nd	survived	male	3.0000	1	1
520	521	2nd	survived	male	20.0000	0	0
523	524	2nd	survived	male	22.0000	0	0
524	525	2nd	survived	male	9999.0000	0	0
526	527	2nd	survived	male	29.0000	0	0
538	539	2nd	survived	male	30.0000	0	0
548	549	2nd	survived	male	0.8333	1	1
549	550	2nd	survived	male	3.0000	1	1
587	588	2nd	survived	male	2.0000	1	1
596	597	2nd	survived	male	31.0000	0	0
597	598	2nd	survived	male	9999.0000	0	0

Q-4) From Values.xls, create the dataframe that contain the cosine values of the angles between 150 degree to 200 degree.

In [11]:
```
data
```

Out[11]:

	Angle (Degrees)	Sine	Cosine	Tangent
0	138.550574	0.661959	-0.749540	-0.883153
1	305.535745	-0.813753	0.581211	-1.400100
2	280.518695	-0.983195	0.182556	-5.385709
3	216.363795	-0.592910	-0.805269	0.736289
4	36.389247	0.593268	0.805005	0.736974
...	...	...	...	...
67	324.199562	-0.584964	0.811059	-0.721234
68	187.948172	-0.138277	-0.990394	0.139619
69	270.678249	-0.999930	0.011837	-84.472139
70	270.779159	-0.999908	0.013598	-73.530885
71	200.213513	-0.345520	-0.938412	0.368196

72 rows × 4 columns

In [12]:
```
# find cosine value in table which > 150
col1=data["Angle (Degrees)"]>150
data1=data.loc[col1]
col2=data1["Angle (Degrees)"]<200
ans = data1.loc[col2]
ans.loc[:,("Cosine")]
```

Out[12]:
```
8    -0.999998
14   -0.994096
24   -0.955297
30   -0.981569
33   -0.998929
37   -0.999200
38   -0.954926
41   -0.984736
47   -0.965127
52   -0.951402
54   -0.997873
62   -0.978785
64   -0.954533
68   -0.990394
Name: Cosine, dtype: float64
```

Q-5) Create a dataframe from last 5 elements from elements.txt and find their average atomic weight

In [13]:
```
print(txt1)  #data
print(txt1.tail(5))    #last 5 element
last5 = txt1.tail(5)
np.average(last5["A_Wght"]) # average
```

```
   A_Wght Symbol       Name
0   1.008      H   Hydrogen
1   4.003     He     Helium
2   6.941     Li    Lithium
3   9.012     Be  Beryllium
4  10.811      B      Boron
5  12.011      C     Carbon
6  14.007      N   Nitrogen
7  15.999      O     Oxygen
8  18.998      F   Fluorine
9  20.180     Ne       Neon
   A_Wght Symbol      Name
5  12.011      C    Carbon
6  14.007      N  Nitrogen
7  15.999      O    Oxygen
8  18.998      F  Fluorine
9  20.180     Ne      Neon
```

Out[13]: 16.238999999999997

In [ ]:

Python for Data Science (LAB Session-8)

Working with Dataset using pandas

Q-1)Load the data from ufo_sighting_data_complete_NUFORC.csv into pandas dataframe and print the shape and columns of the data.

In [1]:
```
import numpy as np
import pandas as pd
```

In [2]:
```
df = pd.read_csv('ufo_sighting_data_complete_NUFORC.csv')
df
```

```
c:\users\arjun vankani\appdata\local\programs\python\python37\lib\site-packag
es\IPython\core\interactiveshell.py:3146: DtypeWarning: Columns (4) have mixe
d types.Specify dtype option on import or set low_memory=False.
  interactivity=interactivity, compiler=compiler, result=result)
```

Out[2]:

	datetime	state	country	shape	duration (seconds)	latitude	longitude	Unnamed: 7
0	10-10-1949 20:30	tx	us	cylinder	2700	29.8830556	-97.941111	NaN
1	10-10-1949 21:00	tx	NaN	light	7200	29.38421	-98.581082	NaN
2	10-10-1955 17:00	NaN	gb	circle	20	53.2	-2.916667	NaN
3	10-10-1956 21:00	tx	us	circle	20	28.9783333	-96.645833	NaN
4	10-10-1960 20:00	hi	us	light	900	21.4180556	-157.803611	NaN
...	...	...	...	...	...	...	...	...
88870	09-09-2013 22:00	ca	us	other	1200	38.2972222	-122.284444	NaN
88871	09-09-2013 22:20	va	us	circle	5	38.9011111	-77.265556	NaN
88872	09-09-2013 23:00	ok	us	cigar	1020	35.6527778	-97.477778	NaN
88873	09-09-2013 23:00	sc	us	diamond	0	34.3769444	-82.695833	NaN
88874	09-09-2013 23:30	fl	us	oval	0	26.1219444	-80.143611	NaN

88875 rows × 8 columns

In [3]:
```
print(df)  # print data frame
```

```
               datetime state country     shape duration (seconds)  \
0      10-10-1949 20:30    tx      us  cylinder               2700
1      10-10-1949 21:00    tx     NaN     light               7200
2      10-10-1955 17:00   NaN      gb    circle                 20
3      10-10-1956 21:00    tx      us    circle                 20
4      10-10-1960 20:00    hi      us     light                900
...                 ...   ...     ...       ...                ...
88870  09-09-2013 22:00    ca      us     other               1200
88871  09-09-2013 22:20    va      us    circle                  5
88872  09-09-2013 23:00    ok      us     cigar               1020
88873  09-09-2013 23:00    sc      us   diamond                  0
88874  09-09-2013 23:30    fl      us      oval                  0

         latitude   longitude  Unnamed: 7
0      29.8830556  -97.941111         NaN
1        29.38421  -98.581082         NaN
2            53.2   -2.916667         NaN
3      28.9783333  -96.645833         NaN
4      21.4180556 -157.803611         NaN
...           ...         ...         ...
88870  38.2972222 -122.284444         NaN
88871  38.9011111  -77.265556         NaN
88872  35.6527778  -97.477778         NaN
88873  34.3769444  -82.695833         NaN
88874  26.1219444  -80.143611         NaN

[88875 rows x 8 columns]
```

In [4]:
```
df.shape  #shape of data frame  (row,col)
```

Out[4]:
```
(88875, 8)
```

In [ ]:
```

```

Q-2)Calculate the percentage of null value entries present into each column and display the data by sorting the data.

In [5]:
```
df.describe()  # describe all the value
```

Out[5]:

	longitude	Unnamed: 7
count	88875.000000	196.0
mean	-84.834334	0.0
std	41.567822	0.0
min	-176.658056	0.0
25%	-112.046944	0.0
50%	-87.650000	0.0
75%	-77.615833	0.0
max	178.441900	0.0

In [23]:
```
per = df.isnull().sum() *100 / len(df)    # find percant
dataframe = pd.DataFrame({'Percentage' : per})
dataframe
```

Out[23]:

	Percentage
datetime	0.000000
state	8.460197
country	14.133333
shape	3.508298
duration (seconds)	0.002250
latitude	0.000000
longitude	0.000000
Unnamed: 7	99.779466

In [24]:
```
dataframe.sort_values(by=['Percentage'], inplace=True, ascending=False)
dataframe  # arrange in ascending
```

Out[24]:

	Percentage
Unnamed: 7	99.779466
country	14.133333
state	8.460197
shape	3.508298
duration (seconds)	0.002250
datetime	0.000000
latitude	0.000000
longitude	0.000000

Q-3) Remove the null values from the dataset. Our intention here is to keep the row if at least 6 values in the row and not-null , else we should remove the row . (Hint: check dropna() documentation). print the shape of the data now.(it should

In [17]:
```
data = pd.DataFrame(df.dropna(thresh=6))   # used dropna() function
data
```

Out[17]:

	datetime	state	country	shape	duration (seconds)	latitude	longitude	Unnamed: 7
0	10-10-1949 20:30	tx	us	cylinder	2700	29.8830556	-97.941111	NaN
1	10-10-1949 21:00	tx	NaN	light	7200	29.38421	-98.581082	NaN
2	10-10-1955 17:00	NaN	gb	circle	20	53.2	-2.916667	NaN
3	10-10-1956 21:00	tx	us	circle	20	28.9783333	-96.645833	NaN
4	10-10-1960 20:00	hi	us	light	900	21.4180556	-157.803611	NaN
...	...	...	...	...	...	...	...	...
88870	09-09-2013 22:00	ca	us	other	1200	38.2972222	-122.284444	NaN
88871	09-09-2013 22:20	va	us	circle	5	38.9011111	-77.265556	NaN
88872	09-09-2013 23:00	ok	us	cigar	1020	35.6527778	-97.477778	NaN
88873	09-09-2013 23:00	sc	us	diamond	0	34.3769444	-82.695833	NaN
88874	09-09-2013 23:30	fl	us	oval	0	26.1219444	-80.143611	NaN

83651 rows × 8 columns

In [19]:
```
data.shape  # shape of data
```

Out[19]: (83651, 8)

Q-4) Find the countries where the most UFO sighting occurs.

In [21]:
```
data1 = pd.DataFrame(df.groupby("country").count().iloc[:,0:1])
data1    # group data into one frame
```

Out[21]:

country	datetime
au	593
ca	3266
de	112
gb	2050
us	70293

In [22]:
```
data1.sort_values(by=['datetime'], inplace=True, ascending=False)
data1  #sortind in ascending
```

Out[22]:

country	datetime
us	70293
ca	3266
gb	2050
au	593
de	112

Q-5) Create a sample dataframe the contains 5% random records(of original records) from the original dataset. Remove the rows with null values from the newly created dataframe and perform below operations. (consider the data in your dataframe as the dataset for below programs

In [26]:
```
data = df.sample(frac=0.05)
data    # random 5% data
```

Out[26]:

	datetime	state	country	shape	duration (seconds)	latitude	longitude	Unnamed: 7
68527	07-04-1999 15:30	ca	us	other	480	38.5816667	-121.493333	NaN
48584	5/22/1996 24:00	oh	us	NaN	0	40.0333333	-83.158333	NaN
14004	11/23/2013 16:45	ca	us	NaN	10	34.3541667	-119.058333	NaN
78707	08-03-1998 17:53	tx	us	fireball	2	31.7586111	-106.486389	NaN
79825	08-07-2005 03:45	wy	us	light	2040	41.8955556	-106.204167	NaN
...	...	...	...	...	...	...	...	...
55233	6/17/2001 22:20	tn	us	fireball	120	36.595	-82.188889	NaN
2554	10/16/2002 20:45	tx	us	light	900	31.5491667	-97.146389	NaN
87981	09-06-2001 06:00	md	us	cigar	1800	39.4141667	-77.410833	NaN
47475	5/16/2011 00:30	mn	us	light	240	45.6091667	-94.451389	NaN
85589	9/24/1998 22:35	ak	us	sphere	600	64.7511111	-147.349444	NaN

4444 rows × 8 columns

In [27]:
```
data1 = pd.DataFrame(data.dropna(thresh=6))
data1
```

Out[27]:

	datetime	state	country	shape	duration (seconds)	latitude	longitude	Unnamed: 7
68527	07-04-1999 15:30	ca	us	other	480	38.5816667	-121.493333	NaN
48584	5/22/1996 24:00	oh	us	NaN	0	40.0333333	-83.158333	NaN
14004	11/23/2013 16:45	ca	us	NaN	10	34.3541667	-119.058333	NaN
78707	08-03-1998 17:53	tx	us	fireball	2	31.7586111	-106.486389	NaN
79825	08-07-2005 03:45	wy	us	light	2040	41.8955556	-106.204167	NaN
...	...	...	...	...	...	...	...	...
55233	6/17/2001 22:20	tn	us	fireball	120	36.595	-82.188889	NaN
2554	10/16/2002 20:45	tx	us	light	900	31.5491667	-97.146389	NaN
87981	09-06-2001 06:00	md	us	cigar	1800	39.4141667	-77.410833	NaN
47475	5/16/2011 00:30	mn	us	light	240	45.6091667	-94.451389	NaN
85589	9/24/1998 22:35	ak	us	sphere	600	64.7511111	-147.349444	NaN

4167 rows × 8 columns

Q-5(A) Write a program to count year wise frequency of reporting dates of unidentified flying object(UFO)

In [33]:
```
data1['datetime'] = pd.to_datetime(data1['datetime'] , errors = 'coerce')
data1['datetime']  # fetch data of datetime
```

Out[33]:
```
68527   1999-07-04 15:30:00
48584                   NaT
14004   2013-11-23 16:45:00
78707   1998-08-03 17:53:00
79825   2005-08-07 03:45:00
                ...
55233   2001-06-17 22:20:00
2554    2002-10-16 20:45:00
87981   2001-09-06 06:00:00
47475   2011-05-16 00:30:00
85589   1998-09-24 22:35:00
Name: datetime, Length: 4167, dtype: datetime64[ns]
```

In [37]:
```
data1['year'] = data1["datetime"].dt.year
data1['year']   # only fetch value of year
```

Out[37]:
```
68527    1999.0
48584       NaN
14004    2013.0
78707    1998.0
79825    2005.0
          ...
55233    2001.0
2554     2002.0
87981    2001.0
47475    2011.0
85589    1998.0
Name: year, Length: 4167, dtype: float64
```

In [36]:
```
print(data1['year'].value_counts().sort_values(ascending = True))
# sorting value of year by UFO values
```

```
1953.0      1
1944.0      1
1961.0      1
1954.0      1
1950.0      1
         ...
2005.0    227
2008.0    246
2011.0    271
2013.0    374
2012.0    395
Name: year, Length: 68, dtype: int64
```

Q-5(B) Write a program to get the current date , oldest date and number of days between current date and oldest date of sighting from UFO dataset.

In [48]:
```
print("Current Date : ",data1.datetime.max())  # current date for dataset
current = data1.datetime.max()
current
```

```
Current Date :  2014-05-06 21:00:00
```

Out[48]:
```
Timestamp('2014-05-06 21:00:00')
```

In [47]:
```
print("\nOldest Date: ", data1.datetime.min())   # oldest date for dataset
old = data1.datetime.min()
old
```

```
Oldest Date:  1936-07-15 00:00:00
```

Out[47]:
```
Timestamp('1936-07-15 00:00:00')
```

In [49]:
```
print("Diiffrence of days between oldest date and current date of dataset : ",
(current - old).days)
```

```
Diiffrence of days between oldest date and current date of dataset :  28419
```

Q-5(C) Write a program to get all the sighting dates of the uidentified flying object(UFO) from last 20 years (last 365*20 days)

In [53]:
```
import datetime

current1 = pd.to_datetime('today')
current1                        # today's time
```

Out[53]:
```
Timestamp('2020-08-31 16:04:32.630486')
```

In [58]:
```
timedur = datetime.timedelta(days = 365*20)
timedur    # 365*20 days
```

Out[58]:
```
datetime.timedelta(days=7300)
```

In [57]:
```
print("Result : ")
print(data1[current1-data1['datetime'] <= timedur])
```

```
Result :
                 datetime state country     shape duration (seconds)  \
14004 2013-11-23 16:45:00    ca      us       NaN                 10
79825 2005-08-07 03:45:00    wy      us     light               2040
15938 2003-11-04 19:30:00    mo      us     light                 60
31652 2002-02-25 12:00:00    ca      us    sphere                180
77098 2010-08-25 05:30:00    co      us    sphere               3600
...                     ...   ...     ...       ...                ...
16722 2011-01-15 22:40:00    vt      us  fireball                600
55233 2001-06-17 22:20:00    tn      us  fireball                120
2554  2002-10-16 20:45:00    tx      us     light                900
87981 2001-09-06 06:00:00    md      us     cigar               1800
47475 2011-05-16 00:30:00    mn      us     light                240

         latitude   longitude  Unnamed: 7    year
14004  34.3541667 -119.058333         NaN  2013.0
79825  41.8955556 -106.204167         NaN  2005.0
15938  37.1272222  -90.450000         NaN  2003.0
31652  39.2191667 -121.060000         NaN  2002.0
77098  38.4783333 -107.875556         NaN  2010.0
...           ...         ...         ...     ...
16722  44.1719444  -72.651389         NaN  2011.0
55233      36.595  -82.188889         NaN  2001.0
2554   31.5491667  -97.146389         NaN  2002.0
87981  39.4141667  -77.410833         NaN  2001.0
47475  45.6091667  -94.451389         NaN  2011.0

[3205 rows x 9 columns]
```

Python for Data Science LAB Session-9

Working with Dataset using pandas

Q-1) Write a program to fill an intermittent time series so all missing dates show up with values of previous non-missing date.

In [1]:
```
import numpy as np
import pandas as pd
```

In [2]:
```
ser = pd.Series([1,10,3, np.nan], index=pd.to_datetime(['2000-05-20', '2000-05-23', '2000-05-26', '2000-05-28']))


print("This series will look like:\n",ser)


# Solution
print("\n\nThe Output of the code should be:\n")
ser.resample('D').ffill()  # fill with previous value
```

```
This series will look like:
 2000-05-20     1.0
2000-05-23    10.0
2000-05-26     3.0
2000-05-28     NaN
dtype: float64


The Output of the code should be:
```

Out[2]:
```
2000-05-20     1.0
2000-05-21     1.0
2000-05-22     1.0
2000-05-23    10.0
2000-05-24    10.0
2000-05-25    10.0
2000-05-26     3.0
2000-05-27     3.0
2000-05-28     NaN
Freq: D, dtype: float64
```

Q-2)Write a sample program to explain how to find missing values from a dataset, how to remove the missing values and how to fill the missing values using fillna() and Imputer.

In [3]:
```
df = pd.read_csv("heart.csv")
```

In [4]:
```
df # dataframe
```

Out[4]:

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target
0	63.0	1	3	145.0	233.0	1.0	0.0	150	0.0	2.3	0.0	0.0	1.0	1.0
1	NaN	1	2	130.0	250.0	0.0	1.0	187	0.0	3.5	0.0	0.0	2.0	NaN
2	41.0	0	1	130.0	NaN	NaN	0.0	172	0.0	NaN	2.0	0.0	2.0	1.0
3	56.0	1	1	120.0	236.0	0.0	1.0	178	0.0	0.8	2.0	0.0	2.0	1.0
4	57.0	0	0	NaN	354.0	0.0	1.0	163	1.0	0.6	2.0	NaN	2.0	1.0
5	NaN	1	0	140.0	192.0	0.0	NaN	148	0.0	0.4	NaN	0.0	1.0	1.0
6	56.0	0	1	140.0	294.0	0.0	0.0	153	0.0	1.3	1.0	0.0	NaN	1.0
7	44.0	1	1	120.0	NaN	0.0	1.0	173	NaN	0.0	2.0	0.0	3.0	1.0
8	52.0	1	2	172.0	199.0	1.0	1.0	162	0.0	0.5	2.0	0.0	3.0	NaN
9	57.0	1	2	150.0	168.0	0.0	1.0	174	0.0	1.6	2.0	0.0	2.0	1.0
10	54.0	1	0	140.0	239.0	0.0	1.0	160	0.0	1.2	2.0	0.0	2.0	1.0
11	48.0	0	2	130.0	275.0	0.0	1.0	139	0.0	0.2	2.0	0.0	2.0	1.0
12	49.0	1	1	130.0	266.0	0.0	1.0	171	0.0	0.6	2.0	0.0	2.0	1.0
13	64.0	1	3	110.0	211.0	0.0	0.0	144	1.0	1.8	1.0	0.0	2.0	1.0
14	58.0	0	3	150.0	283.0	1.0	0.0	162	0.0	1.0	2.0	0.0	2.0	1.0
15	50.0	0	2	120.0	219.0	0.0	1.0	158	0.0	1.6	1.0	0.0	2.0	1.0
16	58.0	0	2	120.0	340.0	0.0	1.0	172	0.0	0.0	2.0	0.0	2.0	1.0
17	66.0	0	3	150.0	226.0	0.0	1.0	114	0.0	2.6	0.0	0.0	2.0	1.0
18	43.0	1	0	150.0	247.0	0.0	1.0	171	0.0	1.5	2.0	0.0	2.0	1.0
19	69.0	0	3	140.0	239.0	0.0	1.0	151	0.0	1.8	2.0	2.0	2.0	1.0
20	59.0	1	0	135.0	234.0	0.0	1.0	161	0.0	0.5	1.0	0.0	3.0	1.0
21	44.0	1	2	130.0	233.0	0.0	1.0	179	1.0	0.4	2.0	0.0	2.0	1.0
22	42.0	1	0	140.0	226.0	0.0	1.0	178	0.0	0.0	2.0	0.0	2.0	1.0
23	61.0	1	2	150.0	243.0	1.0	1.0	137	1.0	1.0	1.0	0.0	2.0	1.0
24	40.0	1	3	140.0	199.0	0.0	1.0	178	1.0	1.4	2.0	0.0	3.0	1.0
25	71.0	0	1	160.0	302.0	0.0	1.0	162	0.0	0.4	2.0	2.0	2.0	1.0
26	59.0	1	2	150.0	212.0	1.0	1.0	157	0.0	1.6	2.0	0.0	2.0	1.0
27	51.0	1	2	110.0	175.0	0.0	1.0	123	0.0	0.6	2.0	0.0	2.0	1.0
28	65.0	0	2	140.0	417.0	1.0	0.0	157	0.0	0.8	2.0	1.0	2.0	1.0
29	53.0	1	2	130.0	197.0	1.0	0.0	152	0.0	1.2	0.0	0.0	2.0	1.0
30	41.0	0	1	105.0	198.0	0.0	1.0	168	0.0	0.0	2.0	1.0	2.0	1.0
31	65.0	1	0	120.0	177.0	0.0	1.0	140	0.0	0.4	2.0	0.0	3.0	1.0
32	44.0	1	1	130.0	219.0	0.0	0.0	188	0.0	0.0	2.0	0.0	2.0	1.0
33	54.0	1	2	125.0	273.0	0.0	0.0	152	0.0	0.5	0.0	1.0	2.0	1.0
34	51.0	1	3	125.0	213.0	0.0	0.0	125	1.0	1.4	2.0	1.0	2.0	1.0
35	46.0	0	2	142.0	177.0	0.0	0.0	160	1.0	1.4	0.0	0.0	2.0	1.0
36	54.0	0	2	135.0	304.0	1.0	1.0	170	0.0	0.0	2.0	0.0	2.0	1.0
37	54.0	1	2	150.0	232.0	0.0	0.0	165	0.0	1.6	2.0	0.0	3.0	1.0
38	65.0	0	2	155.0	269.0	0.0	1.0	148	0.0	0.8	2.0	0.0	2.0	1.0
39	65.0	0	2	160.0	360.0	0.0	0.0	151	0.0	0.8	2.0	0.0	2.0	1.0
40	51.0	0	2	140.0	308.0	0.0	0.0	142	0.0	1.5	2.0	1.0	2.0	1.0
41	48.0	1	1	130.0	245.0	0.0	0.0	180	0.0	0.2	1.0	0.0	2.0	1.0
42	45.0	1	0	104.0	208.0	0.0	0.0	148	1.0	3.0	1.0	0.0	2.0	1.0
43	53.0	0	0	130.0	264.0	0.0	0.0	143	0.0	0.4	1.0	0.0	2.0	1.0
44	39.0	1	2	140.0	321.0	0.0	0.0	182	0.0	0.0	2.0	0.0	2.0	1.0
45	52.0	1	1	120.0	325.0	0.0	1.0	172	0.0	0.2	2.0	0.0	2.0	1.0
46	44.0	1	2	140.0	235.0	0.0	0.0	180	0.0	0.0	2.0	0.0	2.0	1.0
47	47.0	1	2	138.0	257.0	0.0	0.0	156	0.0	0.0	2.0	0.0	2.0	1.0
48	53.0	0	2	128.0	216.0	0.0	0.0	115	0.0	0.0	2.0	0.0	0.0	1.0

NA -> FOR NOT AVAILABLE NaN -> Not a Number None -> Python singleton object that is often used for missing data in Python code

In [6]:
```
data = df.iloc[1:10] # fetch only 10 value
```

In [7]:
```
data
```

Out[7]:

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target
1	NaN	1	2	130.0	250.0	0.0	1.0	187	0.0	3.5	0.0	0.0	2.0	NaN
2	41.0	0	1	130.0	NaN	NaN	0.0	172	0.0	NaN	2.0	0.0	2.0	1.0
3	56.0	1	1	120.0	236.0	0.0	1.0	178	0.0	0.8	2.0	0.0	2.0	1.0
4	57.0	0	0	NaN	354.0	0.0	1.0	163	1.0	0.6	2.0	NaN	2.0	1.0
5	NaN	1	0	140.0	192.0	0.0	NaN	148	0.0	0.4	NaN	0.0	1.0	1.0
6	56.0	0	1	140.0	294.0	0.0	0.0	153	0.0	1.3	1.0	0.0	NaN	1.0
7	44.0	1	1	120.0	NaN	0.0	1.0	173	NaN	0.0	2.0	0.0	3.0	1.0
8	52.0	1	2	172.0	199.0	1.0	1.0	162	0.0	0.5	2.0	0.0	3.0	NaN
9	57.0	1	2	150.0	168.0	0.0	1.0	174	0.0	1.6	2.0	0.0	2.0	1.0

we use isnull() function this function return dataframe of Boolean values which are True for NaN values

In [8]:
```
data.isnull()  # True return where missing value otherwise false
```

Out[8]:

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target
1	True	False	False	False	False	False	False	False	False	False	False	False	False	True
2	False	False	False	False	True	True	False	False	False	True	False	False	False	False
3	False	False	False	False	False	False	False	False	False	False	False	False	False	False
4	False	False	False	True	False	False	False	False	False	False	False	True	False	False
5	True	False	False	False	False	False	True	False	False	False	True	False	False	False
6	False	False	False	False	False	False	False	False	False	False	False	False	True	False
7	False	False	False	False	True	False	False	False	True	False	False	False	False	False
8	False	False	False	False	False	False	False	False	False	False	False	False	False	True
9	False	False	False	False	False	False	False	False	False	False	False	False	False	False

In [9]:
```
data.notnull() # it retuen True where vaule exits otherwise False for missing vaule
```

Out[9]:

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target
1	False	True	True	True	True	True	True	True	True	True	True	True	True	False
2	True	True	True	True	False	False	True	True	True	False	True	True	True	True
3	True	True	True	True	True	True	True	True	True	True	True	True	True	True
4	True	True	True	False	True	True	True	True	True	True	True	False	True	True
5	False	True	True	True	True	True	False	True	True	True	False	True	True	True
6	True	True	True	True	True	True	True	True	True	True	True	True	False	True
7	True	True	True	True	False	True	True	True	False	True	True	True	True	True
8	True	True	True	True	True	True	True	True	True	True	True	True	True	False
9	True	True	True	True	True	True	True	True	True	True	True	True	True	True

In [10]:
```
data.fillna('Empty')   # fillna retuen Null vaule to fill EMPTY value
```

Out[10]:

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target
1	Empty	1	2	130	250	0	1	187	0	3.5	0	0	2	Empty
2	41	0	1	130	Empty	Empty	0	172	0	Empty	2	0	2	1
3	56	1	1	120	236	0	1	178	0	0.8	2	0	2	1
4	57	0	0	Empty	354	0	1	163	1	0.6	2	Empty	2	1
5	Empty	1	0	140	192	0	Empty	148	0	0.4	Empty	0	1	1
6	56	0	1	140	294	0	0	153	0	1.3	1	0	Empty	1
7	44	1	1	120	Empty	0	1	173	Empty	0	2	0	3	1
8	52	1	2	172	199	1	1	162	0	0.5	2	0	3	Empty
9	57	1	2	150	168	0	1	174	0	1.6	2	0	2	1

In [11]:
```
data.fillna(method ='pad') # it's return Null vaule to fill with previous vaule
```

Out[11]:

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target
1	NaN	1	2	130.0	250.0	0.0	1.0	187	0.0	3.5	0.0	0.0	2.0	NaN
2	41.0	0	1	130.0	250.0	0.0	0.0	172	0.0	3.5	2.0	0.0	2.0	1.0
3	56.0	1	1	120.0	236.0	0.0	1.0	178	0.0	0.8	2.0	0.0	2.0	1.0
4	57.0	0	0	120.0	354.0	0.0	1.0	163	1.0	0.6	2.0	0.0	2.0	1.0
5	57.0	1	0	140.0	192.0	0.0	1.0	148	0.0	0.4	2.0	0.0	1.0	1.0
6	56.0	0	1	140.0	294.0	0.0	0.0	153	0.0	1.3	1.0	0.0	1.0	1.0
7	44.0	1	1	120.0	294.0	0.0	1.0	173	0.0	0.0	2.0	0.0	3.0	1.0
8	52.0	1	2	172.0	199.0	1.0	1.0	162	0.0	0.5	2.0	0.0	3.0	1.0
9	57.0	1	2	150.0	168.0	0.0	1.0	174	0.0	1.6	2.0	0.0	2.0	1.0

In [12]:
```
data.fillna(method ='bfill')   # it's return Null vaule to fill with next vaule
```

Out[12]:

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target
1	41.0	1	2	130.0	250.0	0.0	1.0	187	0.0	3.5	0.0	0.0	2.0	1.0
2	41.0	0	1	130.0	236.0	0.0	0.0	172	0.0	0.8	2.0	0.0	2.0	1.0
3	56.0	1	1	120.0	236.0	0.0	1.0	178	0.0	0.8	2.0	0.0	2.0	1.0
4	57.0	0	0	140.0	354.0	0.0	1.0	163	1.0	0.6	2.0	0.0	2.0	1.0
5	56.0	1	0	140.0	192.0	0.0	0.0	148	0.0	0.4	1.0	0.0	1.0	1.0
6	56.0	0	1	140.0	294.0	0.0	0.0	153	0.0	1.3	1.0	0.0	3.0	1.0
7	44.0	1	1	120.0	199.0	0.0	1.0	173	0.0	0.0	2.0	0.0	3.0	1.0
8	52.0	1	2	172.0	199.0	1.0	1.0	162	0.0	0.5	2.0	0.0	3.0	1.0
9	57.0	1	2	150.0	168.0	0.0	1.0	174	0.0	1.6	2.0	0.0	2.0	1.0

In [13]:
```
data["age"].fillna("No age ", inplace = True)
data   # just replace age column in null value as No age
```

```
c:\users\arjun vankani\appdata\local\programs\python\python37\lib\site-packages\pandas\core\series.py:4523: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  downcast=downcast,
```

Out[13]:

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target
1	No age	1	2	130.0	250.0	0.0	1.0	187	0.0	3.5	0.0	0.0	2.0	NaN
2	41	0	1	130.0	NaN	NaN	0.0	172	0.0	NaN	2.0	0.0	2.0	1.0
3	56	1	1	120.0	236.0	0.0	1.0	178	0.0	0.8	2.0	0.0	2.0	1.0
4	57	0	0	NaN	354.0	0.0	1.0	163	1.0	0.6	2.0	NaN	2.0	1.0
5	No age	1	0	140.0	192.0	0.0	NaN	148	0.0	0.4	NaN	0.0	1.0	1.0
6	56	0	1	140.0	294.0	0.0	0.0	153	0.0	1.3	1.0	0.0	NaN	1.0
7	44	1	1	120.0	NaN	0.0	1.0	173	NaN	0.0	2.0	0.0	3.0	1.0
8	52	1	2	172.0	199.0	1.0	1.0	162	0.0	0.5	2.0	0.0	3.0	NaN
9	57	1	2	150.0	168.0	0.0	1.0	174	0.0	1.6	2.0	0.0	2.0	1.0

In [14]:
```
data.replace(to_replace = np.nan, value = '--')
```

Out[14]:

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target
1	No age	1	2	130	250	0	1	187	0	3.5	0	0	2	--
2	41	0	1	130	--	--	0	172	0	--	2	0	2	1
3	56	1	1	120	236	0	1	178	0	0.8	2	0	2	1
4	57	0	0	--	354	0	1	163	1	0.6	2	--	2	1
5	No age	1	0	140	192	0	--	148	0	0.4	--	0	1	1
6	56	0	1	140	294	0	0	153	0	1.3	1	0	--	1
7	44	1	1	120	--	0	1	173	--	0	2	0	3	1
8	52	1	2	172	199	1	1	162	0	0.5	2	0	3	--
9	57	1	2	150	168	0	1	174	0	1.6	2	0	2	1

It uses various interpolation technique to fill the missing values rather than hard-coding the value, Linear method ignore the index and treat the values as equally spaced

In [15]:
```
data.interpolate(method ='linear', limit_direction ='forward')
```

Out[15]:

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target
1	No age	1	2	130.0	250.0	0.0	1.0	187	0.0	3.50	0.0	0.0	2.0	NaN
2	41	0	1	130.0	243.0	0.0	0.0	172	0.0	2.15	2.0	0.0	2.0	1.0
3	56	1	1	120.0	236.0	0.0	1.0	178	0.0	0.80	2.0	0.0	2.0	1.0
4	57	0	0	130.0	354.0	0.0	1.0	163	1.0	0.60	2.0	0.0	2.0	1.0
5	No age	1	0	140.0	192.0	0.0	0.5	148	0.0	0.40	1.5	0.0	1.0	1.0
6	56	0	1	140.0	294.0	0.0	0.0	153	0.0	1.30	1.0	0.0	2.0	1.0
7	44	1	1	120.0	246.5	0.0	1.0	173	0.0	0.00	2.0	0.0	3.0	1.0
8	52	1	2	172.0	199.0	1.0	1.0	162	0.0	0.50	2.0	0.0	3.0	1.0
9	57	1	2	150.0	168.0	0.0	1.0	174	0.0	1.60	2.0	0.0	2.0	1.0

In [16]:
```
data.interpolate(method ='linear', limit_direction ='backward', limit = 1)
```

Out[16]:

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target
1	No age	1	2	130.0	250.0	0.0	1.0	187	0.0	3.50	0.0	0.0	2.0	1.0
2	41	0	1	130.0	243.0	0.0	0.0	172	0.0	2.15	2.0	0.0	2.0	1.0
3	56	1	1	120.0	236.0	0.0	1.0	178	0.0	0.80	2.0	0.0	2.0	1.0
4	57	0	0	130.0	354.0	0.0	1.0	163	1.0	0.60	2.0	0.0	2.0	1.0
5	No age	1	0	140.0	192.0	0.0	0.5	148	0.0	0.40	1.5	0.0	1.0	1.0
6	56	0	1	140.0	294.0	0.0	0.0	153	0.0	1.30	1.0	0.0	2.0	1.0
7	44	1	1	120.0	246.5	0.0	1.0	173	0.0	0.00	2.0	0.0	3.0	1.0
8	52	1	2	172.0	199.0	1.0	1.0	162	0.0	0.50	2.0	0.0	3.0	1.0
9	57	1	2	150.0	168.0	0.0	1.0	174	0.0	1.60	2.0	0.0	2.0	1.0

In [17]:
```
data.dropna()  #only show fully filled row , another all are leave
```

Out[17]:

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target
3	56	1	1	120.0	236.0	0.0	1.0	178	0.0	0.8	2.0	0.0	2.0	1.0
9	57	1	2	150.0	168.0	0.0	1.0	174	0.0	1.6	2.0	0.0	2.0	1.0

In [18]:
```
data.dropna(how = 'all')  # we drop a rows whose all data is missing or contain null value
```

Out[18]:

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target
1	No age	1	2	130.0	250.0	0.0	1.0	187	0.0	3.5	0.0	0.0	2.0	NaN
2	41	0	1	130.0	NaN	NaN	0.0	172	0.0	NaN	2.0	0.0	2.0	1.0
3	56	1	1	120.0	236.0	0.0	1.0	178	0.0	0.8	2.0	0.0	2.0	1.0
4	57	0	0	NaN	354.0	0.0	1.0	163	1.0	0.6	2.0	NaN	2.0	1.0
5	No age	1	0	140.0	192.0	0.0	NaN	148	0.0	0.4	NaN	0.0	1.0	1.0
6	56	0	1	140.0	294.0	0.0	0.0	153	0.0	1.3	1.0	0.0	NaN	1.0
7	44	1	1	120.0	NaN	0.0	1.0	173	NaN	0.0	2.0	0.0	3.0	1.0
8	52	1	2	172.0	199.0	1.0	1.0	162	0.0	0.5	2.0	0.0	3.0	NaN
9	57	1	2	150.0	168.0	0.0	1.0	174	0.0	1.6	2.0	0.0	2.0	1.0

In [19]:
```
data.dropna(axis = 1)   # drop null column
```

Out[19]:

	age	sex	cp	thalach
1	No age	1	2	187
2	41	0	1	172
3	56	1	1	178
4	57	0	0	163
5	No age	1	0	148
6	56	0	1	153
7	44	1	1	173
8	52	1	2	162
9	57	1	2	174

In [20]:
```
data.dropna(axis = 0, how ='any')  # if only one vaule missing in any row then remove it's row
```

Out[20]:

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target
3	56	1	1	120.0	236.0	0.0	1.0	178	0.0	0.8	2.0	0.0	2.0	1.0
9	57	1	2	150.0	168.0	0.0	1.0	174	0.0	1.6	2.0	0.0	2.0	1.0

Q-3) Write a sample program to explain working with categorical variables:

a. Define categorical variables with at least 5 categories. Take sample input that has values from the categories and some values which are out of the categories and observe the output

b. Rename the categorical values to some new values and observer the output

c. Combine any two categories with each other (combining the levels) so that now there are only three categories. Apply the new categories to the dataset and observe the output.

In [64]:
```
m = pd.Categorical(['a', 'k', 'l', 'e', 'a', 'b','e'])
m
```

Out[64]:
```
['a', 'k', 'l', 'e', 'a', 'b', 'e']
Categories (5, object): ['a', 'b', 'e', 'k', 'l']
```

In [27]:
```
e = pd.Categorical(['a', 'b', 'c', 'e', 'a', 'b','e'])
e
```

Out[27]:
```
['a', 'b', 'c', 'e', 'a', 'b', 'e']
Categories (4, object): ['a', 'b', 'c', 'e']
```

In [87]:
```
from pandas.api.types import CategoricalDtype
e == CategoricalDtype(['b', 'c', 'a'])

# yes it return true bcz it belong to this catogary
```

Out[87]:
```
True
```

In [88]:
```
m == CategoricalDtype(['k', 'l', 'a'], ordered=True)
```

Out[88]:
```
array([False, False, False, False, False, False, False])
```

In [30]:
```
c.describe()
```

Out[30]:
```
count     5
unique    5
top       e
freq      1
dtype: object
```

In [29]:
```
e.describe()
```

Out[29]:

	counts	freqs
categories		
a	2	0.285714
b	2	0.285714
c	1	0.142857
e	2	0.285714

In [90]:
```
#renameing
```

In [94]:
```
d1 = pd.Series(["k","l","m","p","k"],dtype="category")
d1
```

Out[94]:
```
0    k
1    l
2    m
3    p
4    k
dtype: category
Categories (4, object): ['k', 'l', 'm', 'p']
```

In [95]:
```
d1.cat.categories = ["Group %s" % g for g in d1.cat.categories] # group
```

In [96]:
```
d1
```

Out[96]:
```
0    Group k
1    Group l
2    Group m
3    Group p
4    Group k
dtype: category
Categories (4, object): ['Group k', 'Group l', 'Group m', 'Group p']
```

In [101]:
```
d1 = d1.cat.rename_categories([1, 2, 3, 4])
d1  #rename
```

Out[101]:
```
0    1
1    2
2    3
3    4
4    1
dtype: category
Categories (4, int64): [1, 2, 3, 4]
```

In [ ]:

In [106]:
```
d1 = d1.cat.add_categories([5])
d1
```

Out[106]:
```
0    1
1    2
2    3
3    4
4    1
dtype: category
Categories (6, object): [1, 2, 3, 4, 'a', 5]
```

In [107]:
```
d1.cat.categories
```

Out[107]:
```
Index([1, 2, 3, 4, 'a', 5], dtype='object')
```

In [110]:
```
d1 = d1.cat.remove_categories([4])
d1
```

Out[110]:
```
0      1
1      2
2      3
3    NaN
4      1
dtype: category
Categories (5, object): [1, 2, 3, 'a', 5]
```

In [ ]:

In [115]:
```
from pandas.api.types import union_categoricals
ser1 = pd.Series(['k', 'l'], dtype='category')
ser2 = pd.Series(['m', 'b', 'a'], dtype='category')
print(ser1)
print(ser2)
pd.concat([ser1,ser2])
```

```
0    k
1    l
dtype: category
Categories (2, object): ['k', 'l']
0    m
1    b
2    a
dtype: category
Categories (3, object): ['a', 'b', 'm']
```

Out[115]:
```
0    k
1    l
0    m
1    b
2    a
dtype: object
```

In [123]:
```
ser3 = union_categoricals([ser1.array, ser2.array])
ser3
```

Out[123]:
```
['k', 'l', 'm', 'b', 'a']
Categories (5, object): ['k', 'l', 'a', 'b', 'm']
```

Q-4) Write a sample program to demonstrate how to sort and shuffle the data from a dataset.

In [126]:
```
data
```

Out[126]:

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target
1	No age	1	2	130.0	250.0	0.0	1.0	187	0.0	3.5	0.0	0.0	2.0	NaN
2	41	0	1	130.0	NaN	NaN	0.0	172	0.0	NaN	2.0	0.0	2.0	1.0
3	56	1	1	120.0	236.0	0.0	1.0	178	0.0	0.8	2.0	0.0	2.0	1.0
4	57	0	0	NaN	354.0	0.0	1.0	163	1.0	0.6	2.0	NaN	2.0	1.0
5	No age	1	0	140.0	192.0	0.0	NaN	148	0.0	0.4	NaN	0.0	1.0	1.0
6	56	0	1	140.0	294.0	0.0	0.0	153	0.0	1.3	1.0	0.0	NaN	1.0
7	44	1	1	120.0	NaN	0.0	1.0	173	NaN	0.0	2.0	0.0	3.0	1.0
8	52	1	2	172.0	199.0	1.0	1.0	162	0.0	0.5	2.0	0.0	3.0	NaN
9	57	1	2	150.0	168.0	0.0	1.0	174	0.0	1.6	2.0	0.0	2.0	1.0

In [134]:
```
data.sort_values("trestbps", axis = 0, ascending = True, inplace = True)
data  # sort by perticular column
```

```
c:\users\arjun vankani\appdata\local\programs\python\python37\lib\site-packages\ipykernel_launcher.py:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  """Entry point for launching an IPython kernel.
```

Out[134]:

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target
3	56	1	1	120.0	236.0	0.0	1.0	178	0.0	0.8	2.0	0.0	2.0	1.0
7	44	1	1	120.0	NaN	0.0	1.0	173	NaN	0.0	2.0	0.0	3.0	1.0
1	No age	1	2	130.0	250.0	0.0	1.0	187	0.0	3.5	0.0	0.0	2.0	NaN
2	41	0	1	130.0	NaN	NaN	0.0	172	0.0	NaN	2.0	0.0	2.0	1.0
5	No age	1	0	140.0	192.0	0.0	NaN	148	0.0	0.4	NaN	0.0	1.0	1.0
6	56	0	1	140.0	294.0	0.0	0.0	153	0.0	1.3	1.0	0.0	NaN	1.0
9	57	1	2	150.0	168.0	0.0	1.0	174	0.0	1.6	2.0	0.0	2.0	1.0
8	52	1	2	172.0	199.0	1.0	1.0	162	0.0	0.5	2.0	0.0	3.0	NaN
4	57	0	0	NaN	354.0	0.0	1.0	163	1.0	0.6	2.0	NaN	2.0	1.0

In [136]:
```
data.sort_values("thalach", axis = 0, ascending = True, inplace = True)
data
```

```
c:\users\arjun vankani\appdata\local\programs\python\python37\lib\site-packages\ipykernel_launcher.py:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  """Entry point for launching an IPython kernel.
```

Out[136]:

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target
5	No age	1	0	140.0	192.0	0.0	NaN	148	0.0	0.4	NaN	0.0	1.0	1.0
6	56	0	1	140.0	294.0	0.0	0.0	153	0.0	1.3	1.0	0.0	NaN	1.0
8	52	1	2	172.0	199.0	1.0	1.0	162	0.0	0.5	2.0	0.0	3.0	NaN
4	57	0	0	NaN	354.0	0.0	1.0	163	1.0	0.6	2.0	NaN	2.0	1.0
2	41	0	1	130.0	NaN	NaN	0.0	172	0.0	NaN	2.0	0.0	2.0	1.0
7	44	1	1	120.0	NaN	0.0	1.0	173	NaN	0.0	2.0	0.0	3.0	1.0
9	57	1	2	150.0	168.0	0.0	1.0	174	0.0	1.6	2.0	0.0	2.0	1.0
3	56	1	1	120.0	236.0	0.0	1.0	178	0.0	0.8	2.0	0.0	2.0	1.0
1	No age	1	2	130.0	250.0	0.0	1.0	187	0.0	3.5	0.0	0.0	2.0	NaN

In [137]:
```
data.sample(frac=1).reset_index(drop=True) #shuffle frame
```

Out[137]:

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target
0	No age	1	0	140.0	192.0	0.0	NaN	148	0.0	0.4	NaN	0.0	1.0	1.0
1	57	0	0	NaN	354.0	0.0	1.0	163	1.0	0.6	2.0	NaN	2.0	1.0
2	No age	1	2	130.0	250.0	0.0	1.0	187	0.0	3.5	0.0	0.0	2.0	NaN
3	57	1	2	150.0	168.0	0.0	1.0	174	0.0	1.6	2.0	0.0	2.0	1.0
4	52	1	2	172.0	199.0	1.0	1.0	162	0.0	0.5	2.0	0.0	3.0	NaN
5	41	0	1	130.0	NaN	NaN	0.0	172	0.0	NaN	2.0	0.0	2.0	1.0
6	56	1	1	120.0	236.0	0.0	1.0	178	0.0	0.8	2.0	0.0	2.0	1.0
7	56	0	1	140.0	294.0	0.0	0.0	153	0.0	1.3	1.0	0.0	NaN	1.0
8	44	1	1	120.0	NaN	0.0	1.0	173	NaN	0.0	2.0	0.0	3.0	1.0

In [142]:
```
data.sample(frac=1)
```

Out[142]:

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target
4	57	0	0	NaN	354.0	0.0	1.0	163	1.0	0.6	2.0	NaN	2.0	1.0
8	52	1	2	172.0	199.0	1.0	1.0	162	0.0	0.5	2.0	0.0	3.0	NaN
5	No age	1	0	140.0	192.0	0.0	NaN	148	0.0	0.4	NaN	0.0	1.0	1.0
2	41	0	1	130.0	NaN	NaN	0.0	172	0.0	NaN	2.0	0.0	2.0	1.0
6	56	0	1	140.0	294.0	0.0	0.0	153	0.0	1.3	1.0	0.0	NaN	1.0
3	56	1	1	120.0	236.0	0.0	1.0	178	0.0	0.8	2.0	0.0	2.0	1.0
7	44	1	1	120.0	NaN	0.0	1.0	173	NaN	0.0	2.0	0.0	3.0	1.0
9	57	1	2	150.0	168.0	0.0	1.0	174	0.0	1.6	2.0	0.0	2.0	1.0
1	No age	1	2	130.0	250.0	0.0	1.0	187	0.0	3.5	0.0	0.0	2.0	NaN

In [143]:
```
data.sort_index()
```

Out[143]:

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target
1	No age	1	2	130.0	250.0	0.0	1.0	187	0.0	3.5	0.0	0.0	2.0	NaN
2	41	0	1	130.0	NaN	NaN	0.0	172	0.0	NaN	2.0	0.0	2.0	1.0
3	56	1	1	120.0	236.0	0.0	1.0	178	0.0	0.8	2.0	0.0	2.0	1.0
4	57	0	0	NaN	354.0	0.0	1.0	163	1.0	0.6	2.0	NaN	2.0	1.0
5	No age	1	0	140.0	192.0	0.0	NaN	148	0.0	0.4	NaN	0.0	1.0	1.0
6	56	0	1	140.0	294.0	0.0	0.0	153	0.0	1.3	1.0	0.0	NaN	1.0
7	44	1	1	120.0	NaN	0.0	1.0	173	NaN	0.0	2.0	0.0	3.0	1.0
8	52	1	2	172.0	199.0	1.0	1.0	162	0.0	0.5	2.0	0.0	3.0	NaN
9	57	1	2	150.0	168.0	0.0	1.0	174	0.0	1.6	2.0	0.0	2.0	1.0

In [145]:
```
data.sort_index(ascending=False)
```

Out[145]:

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target
9	57	1	2	150.0	168.0	0.0	1.0	174	0.0	1.6	2.0	0.0	2.0	1.0
8	52	1	2	172.0	199.0	1.0	1.0	162	0.0	0.5	2.0	0.0	3.0	NaN
7	44	1	1	120.0	NaN	0.0	1.0	173	NaN	0.0	2.0	0.0	3.0	1.0
6	56	0	1	140.0	294.0	0.0	0.0	153	0.0	1.3	1.0	0.0	NaN	1.0
5	No age	1	0	140.0	192.0	0.0	NaN	148	0.0	0.4	NaN	0.0	1.0	1.0
4	57	0	0	NaN	354.0	0.0	1.0	163	1.0	0.6	2.0	NaN	2.0	1.0
3	56	1	1	120.0	236.0	0.0	1.0	178	0.0	0.8	2.0	0.0	2.0	1.0
2	41	0	1	130.0	NaN	NaN	0.0	172	0.0	NaN	2.0	0.0	2.0	1.0
1	No age	1	2	130.0	250.0	0.0	1.0	187	0.0	3.5	0.0	0.0	2.0	NaN

In [146]:
```
data.sort_index(axis=1)
```

Out[146]:

	age	ca	chol	cp	exang	fbs	oldpeak	restecg	sex	slope	target	thal	thalach	trestbps
5	No age	0.0	192.0	0	0.0	0.0	0.4	NaN	1	NaN	1.0	1.0	148	140.0
6	56	0.0	294.0	1	0.0	0.0	1.3	0.0	0	1.0	1.0	NaN	153	140.0
8	52	0.0	199.0	2	0.0	1.0	0.5	1.0	1	2.0	NaN	3.0	162	172.0
4	57	NaN	354.0	0	1.0	0.0	0.6	1.0	0	2.0	1.0	2.0	163	NaN
2	41	0.0	NaN	1	0.0	NaN	NaN	0.0	0	2.0	1.0	2.0	172	130.0
7	44	0.0	NaN	1	NaN	0.0	0.0	1.0	1	2.0	1.0	3.0	173	120.0
9	57	0.0	168.0	2	0.0	0.0	1.6	1.0	1	2.0	1.0	2.0	174	150.0
3	56	0.0	236.0	1	0.0	0.0	0.8	1.0	1	2.0	1.0	2.0	178	120.0
1	No age	0.0	250.0	2	0.0	0.0	3.5	1.0	1	0.0	NaN	2.0	187	130.0

In [150]:
```
data.sort_values(by='trestbps',kind='mergesort')
```

Out[150]:

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target
7	44	1	1	120.0	NaN	0.0	1.0	173	NaN	0.0	2.0	0.0	3.0	1.0
3	56	1	1	120.0	236.0	0.0	1.0	178	0.0	0.8	2.0	0.0	2.0	1.0
2	41	0	1	130.0	NaN	NaN	0.0	172	0.0	NaN	2.0	0.0	2.0	1.0
1	No age	1	2	130.0	250.0	0.0	1.0	187	0.0	3.5	0.0	0.0	2.0	NaN
5	No age	1	0	140.0	192.0	0.0	NaN	148	0.0	0.4	NaN	0.0	1.0	1.0
6	56	0	1	140.0	294.0	0.0	0.0	153	0.0	1.3	1.0	0.0	NaN	1.0
9	57	1	2	150.0	168.0	0.0	1.0	174	0.0	1.6	2.0	0.0	2.0	1.0
8	52	1	2	172.0	199.0	1.0	1.0	162	0.0	0.5	2.0	0.0	3.0	NaN
4	57	0	0	NaN	354.0	0.0	1.0	163	1.0	0.6	2.0	NaN	2.0	1.0

In [ ]:

Python for Data Science (LAB Session - 10)

Basics of NLP

Q-1)Explore NLP with example of The 20NewsGroup

In [1]:
```
from sklearn.datasets import fetch_20newsgroups
from nltk.tokenize import sent_tokenize, word_tokenize
#importing librery
```

In [2]:
```
data = fetch_20newsgroups() # assign value of dataset
a=data["data"]
df = a[0]
print(df) # print data of frist index
```

```
From: lerxst@wam.umd.edu (where's my thing)
Subject: WHAT car is this!?
Nntp-Posting-Host: rac3.wam.umd.edu
Organization: University of Maryland, College Park
Lines: 15

 I was wondering if anyone out there could enlighten me on this car I saw
the other day. It was a 2-door sports car, looked to be from the late 60s/
early 70s. It was called a Bricklin. The doors were really small. In additio
n,
the front bumper was separate from the rest of the body. This is
all I know. If anyone can tellme a model name, engine specs, years
of production, where this car is made, history, or whatever info you
have on this funky looking car, please e-mail.

Thanks,
- IL
   ---- brought to you by your neighborhood Lerxst ----
```

Q-A) Sentence Tokenization

In [3]:
```
print(sent_tokenize(df))
```

```
["From: lerxst@wam.umd.edu (where's my thing)\nSubject: WHAT car is this!?",
'Nntp-Posting-Host: rac3.wam.umd.edu\nOrganization: University of Maryland, C
ollege Park\nLines: 15\n\n I was wondering if anyone out there could enlighte
n me on this car I saw\nthe other day.', 'It was a 2-door sports car, looked
to be from the late 60s/\nearly 70s.', 'It was called a Bricklin.', 'The door
s were really small.', 'In addition,\nthe front bumper was separate from the
rest of the body.', 'This is \nall I know.', 'If anyone can tellme a model na
me, engine specs, years\nof production, where this car is made, history, or w
hatever info you\nhave on this funky looking car, please e-mail.', 'Thanks,\n
- IL\n   ---- brought to you by your neighborhood Lerxst ----']
```

Q-B) Word Tokenization

In [10]:
```
print(word_tokenize(df))
word_data = word_tokenize(df)
```

```
['From', ':', 'lerxst', '@', 'wam.umd.edu', '(', 'where', "'s", 'my', 'thin
g', ')', 'Subject', ':', 'WHAT', 'car', 'is', 'this', '!', '?', 'Nntp-Posting
-Host', ':', 'rac3.wam.umd.edu', 'Organization', ':', 'University', 'of', 'Ma
ryland', ',', 'College', 'Park', 'Lines', ':', '15', 'I', 'was', 'wondering',
'if', 'anyone', 'out', 'there', 'could', 'enlighten', 'me', 'on', 'this', 'ca
r', 'I', 'saw', 'the', 'other', 'day', '.', 'It', 'was', 'a', '2-door', 'spor
ts', 'car', ',', 'looked', 'to', 'be', 'from', 'the', 'late', '60s/', 'earl
y', '70s', '.', 'It', 'was', 'called', 'a', 'Bricklin', '.', 'The', 'doors',
'were', 'really', 'small', '.', 'In', 'addition', ',', 'the', 'front', 'bumpe
r', 'was', 'separate', 'from', 'the', 'rest', 'of', 'the', 'body', '.', 'Thi
s', 'is', 'all', 'I', 'know', '.', 'If', 'anyone', 'can', 'tellme', 'a', 'mod
el', 'name', ',', 'engine', 'specs', ',', 'years', 'of', 'production', ',',
'where', 'this', 'car', 'is', 'made', ',', 'history', ',', 'or', 'whatever',
'info', 'you', 'have', 'on', 'this', 'funky', 'looking', 'car', ',', 'pleas
e', 'e-mail', '.', 'Thanks', ',', '-', 'IL', '--', '--', 'brought', 'to', 'yo
u', 'by', 'your', 'neighborhood', 'Lerxst', '--', '--']
```

Q-C)Text Lemmatization

In [22]:
```
from nltk.stem import WordNetLemmatizer
lemmatizer = WordNetLemmatizer()
lemmatize_data = ""
list1 = []
list2 = []
for i in word_data:
    list2.append(i)
    l=lemmatizer.lemmatize(i)
    list1.append(l)

print("Lammatizer all list out word...\n")
for i in zip(list2,list1):
    print(i)
```

```
Lammatizer all list out word...

('From', 'From')
(':', ':')
('lerxst', 'lerxst')
('@', '@')
('wam.umd.edu', 'wam.umd.edu')
('(', '(')
('where', 'where')
("'s", "'s")
('my', 'my')
('thing', 'thing')
(')', ')')
('Subject', 'Subject')
(':', ':')
('WHAT', 'WHAT')
('car', 'car')
('is', 'is')
('this', 'this')
('!', '!')
('?', '?')
('Nntp-Posting-Host', 'Nntp-Posting-Host')
(':', ':')
('rac3.wam.umd.edu', 'rac3.wam.umd.edu')
('Organization', 'Organization')
(':', ':')
('University', 'University')
('of', 'of')
('Maryland', 'Maryland')
(',', ',')
('College', 'College')
('Park', 'Park')
('Lines', 'Lines')
(':', ':')
('15', '15')
('I', 'I')
('was', 'wa')
('wondering', 'wondering')
('if', 'if')
('anyone', 'anyone')
('out', 'out')
('there', 'there')
('could', 'could')
('enlighten', 'enlighten')
('me', 'me')
('on', 'on')
('this', 'this')
('car', 'car')
('I', 'I')
('saw', 'saw')
('the', 'the')
('other', 'other')
('day', 'day')
('.', '.')
('It', 'It')
('was', 'wa')
('a', 'a')
```

```
('2-door', '2-door')
('sports', 'sport')
('car', 'car')
(',', ',')
('looked', 'looked')
('to', 'to')
('be', 'be')
('from', 'from')
('the', 'the')
('late', 'late')
('60s/', '60s/')
('early', 'early')
('70s', '70')
('.', '.')
('It', 'It')
('was', 'wa')
('called', 'called')
('a', 'a')
('Bricklin', 'Bricklin')
('.', '.')
('The', 'The')
('doors', 'door')
('were', 'were')
('really', 'really')
('small', 'small')
('.', '.')
('In', 'In')
('addition', 'addition')
(',', ',')
('the', 'the')
('front', 'front')
('bumper', 'bumper')
('was', 'wa')
('separate', 'separate')
('from', 'from')
('the', 'the')
('rest', 'rest')
('of', 'of')
('the', 'the')
('body', 'body')
('.', '.')
('This', 'This')
('is', 'is')
('all', 'all')
('I', 'I')
('know', 'know')
('.', '.')
('If', 'If')
('anyone', 'anyone')
('can', 'can')
('tellme', 'tellme')
('a', 'a')
('model', 'model')
('name', 'name')
(',', ',')
('engine', 'engine')
('specs', 'spec')
```

```
(',', ',')
('years', 'year')
('of', 'of')
('production', 'production')
(',', ',')
('where', 'where')
('this', 'this')
('car', 'car')
('is', 'is')
('made', 'made')
(',', ',')
('history', 'history')
(',', ',')
('or', 'or')
('whatever', 'whatever')
('info', 'info')
('you', 'you')
('have', 'have')
('on', 'on')
('this', 'this')
('funky', 'funky')
('looking', 'looking')
('car', 'car')
(',', ',')
('please', 'please')
('e-mail', 'e-mail')
('.', '.')
('Thanks', 'Thanks')
(',', ',')
('-', '-')
('IL', 'IL')
('--', '--')
('--', '--')
('brought', 'brought')
('to', 'to')
('you', 'you')
('by', 'by')
('your', 'your')
('neighborhood', 'neighborhood')
('Lerxst', 'Lerxst')
('--', '--')
('--', '--')
```

In [15]:

```
lemmatizer = WordNetLemmatizer()
print("rocks :", lemmatizer.lemmatize("rocks"))
print("corpora :", lemmatizer.lemmatize("corpora"))
print("better :", lemmatizer.lemmatize("better", pos ="a"))
```

```
rocks : rock
corpora : corpus
better : good
```

In [17]:
```
sentence = ["This","sentence","was","transformed", "using", "WordNet", "Lemmat
izer"]

lemmatizer = WordNetLemmatizer()

print (" ".join([lemmatizer.lemmatize(word) for word in sentence]))
```

```
This sentence wa transformed using WordNet Lemmatizer
```

Q-D)Stemming

In [27]:
```
from nltk.stem import PorterStemmer
stem = PorterStemmer()
res_str = ""
list1 = []
list2 = []
for i in word_data:
    list2.append(i)
    l=stem.stem(i)
    list1.append(l)

print("Stemming Text all list out word...\n")
for i in zip(list2,list1):
    print(i)
```

```
Stemming Text all list out word...

('From', 'from')
(':', ':')
('lerxst', 'lerxst')
('@', '@')
('wam.umd.edu', 'wam.umd.edu')
('(', '(')
('where', 'where')
("'s", "'s")
('my', 'my')
('thing', 'thing')
(')', ')')
('Subject', 'subject')
(':', ':')
('WHAT', 'what')
('car', 'car')
('is', 'is')
('this', 'thi')
('!', '!')
('?', '?')
('Nntp-Posting-Host', 'nntp-posting-host')
(':', ':')
('rac3.wam.umd.edu', 'rac3.wam.umd.edu')
('Organization', 'organ')
(':', ':')
('University', 'univers')
('of', 'of')
('Maryland', 'maryland')
(',', ',')
('College', 'colleg')
('Park', 'park')
('Lines', 'line')
(':', ':')
('15', '15')
('I', 'I')
('was', 'wa')
('wondering', 'wonder')
('if', 'if')
('anyone', 'anyon')
('out', 'out')
('there', 'there')
('could', 'could')
('enlighten', 'enlighten')
('me', 'me')
('on', 'on')
('this', 'thi')
('car', 'car')
('I', 'I')
('saw', 'saw')
('the', 'the')
('other', 'other')
('day', 'day')
('.', '.')
('It', 'It')
('was', 'wa')
('a', 'a')
```

```
('2-door', '2-door')
('sports', 'sport')
('car', 'car')
(',', ',')
('looked', 'look')
('to', 'to')
('be', 'be')
('from', 'from')
('the', 'the')
('late', 'late')
('60s/', '60s/')
('early', 'earli')
('70s', '70')
('.', '.')
('It', 'It')
('was', 'wa')
('called', 'call')
('a', 'a')
('Bricklin', 'bricklin')
('.', '.')
('The', 'the')
('doors', 'door')
('were', 'were')
('really', 'realli')
('small', 'small')
('.', '.')
('In', 'In')
('addition', 'addit')
(',', ',')
('the', 'the')
('front', 'front')
('bumper', 'bumper')
('was', 'wa')
('separate', 'separ')
('from', 'from')
('the', 'the')
('rest', 'rest')
('of', 'of')
('the', 'the')
('body', 'bodi')
('.', '.')
('This', 'thi')
('is', 'is')
('all', 'all')
('I', 'I')
('know', 'know')
('.', '.')
('If', 'If')
('anyone', 'anyon')
('can', 'can')
('tellme', 'tellm')
('a', 'a')
('model', 'model')
('name', 'name')
(',', ',')
('engine', 'engin')
('specs', 'spec')
```

```
(',', ',')
('years', 'year')
('of', 'of')
('production', 'product')
(',', ',')
('where', 'where')
('this', 'thi')
('car', 'car')
('is', 'is')
('made', 'made')
(',', ',')
('history', 'histori')
(',', ',')
('or', 'or')
('whatever', 'whatev')
('info', 'info')
('you', 'you')
('have', 'have')
('on', 'on')
('this', 'thi')
('funky', 'funki')
('looking', 'look')
('car', 'car')
(',', ',')
('please', 'pleas')
('e-mail', 'e-mail')
('.', '.')
('Thanks', 'thank')
(',', ',')
('-', '-')
('IL', 'IL')
('--', '--')
('--', '--')
('brought', 'brought')
('to', 'to')
('you', 'you')
('by', 'by')
('your', 'your')
('neighborhood', 'neighborhood')
('Lerxst', 'lerxst')
('--', '--')
('--', '--')
```

Q-E)Stop Words

In [31]:
```
from nltk.corpus import stopwords
word=stopwords.words('english')
word_data=word_tokenize(df)
remove_stop_str = ""
list1 = []
for i in word_data:
    if( i not in word):
        lst.append(i)
for i in lst:
    remove_stop_str += "".join(i)
print(remove_stop_str)
```

```
From:lerxst@wam.umd.edu(where'smything)Subject:WHATcaristhis!?Nntp-Posting-Ho
st:rac3.wam.umd.eduOrganization:UniversityofMaryland,CollegeParkLines:15Iwawo
nderingifanyoneouttherecouldenlightenmeonthiscarIsawtheotherday.Itwaa2-doorsp
ortcar,lookedtobefromthelate60s/early70.ItwacalledaBricklin.Thedoorwerereally
small.Inaddition,thefrontbumperwaseparatefromtherestofthebody.ThisisallIknow.
Ifanyonecantellmeamodelname,enginespec,yearofproduction,wherethiscarismade,hi
story,orwhateverinfoyouhaveonthisfunkylookingcar,pleasee-mail.Thanks,-IL----b
roughttoyoubyyourneighborhoodLerxst----From:lerxst@wam.umd.edu('sthing)Subjec
t:WHATcar!?Nntp-Posting-Host:rac3.wam.umd.eduOrganization:UniversityMaryland,
CollegeParkLines:15IwonderinganyonecouldenlightencarIsawday.It2-doorsportsca
r,lookedlate60s/early70s.ItcalledBricklin.Thedoorsreallysmall.Inaddition,fron
tbumperseparaterestbody.ThisIknow.Ifanyonetellmemodelname,enginespecs,yearspr
oduction,carmade,history,whateverinfofunkylookingcar,pleasee-mail.Thanks,-IL-
---broughtneighborhoodLerxst----From:lerxst@wam.umd.edu('sthing)Subject:WHATc
ar!?Nntp-Posting-Host:rac3.wam.umd.eduOrganization:UniversityMaryland,College
ParkLines:15IwonderinganyonecouldenlightencarIsawday.It2-doorsportscar,looked
late60s/early70s.ItcalledBricklin.Thedoorsreallysmall.Inaddition,frontbumpers
eparaterestbody.ThisIknow.Ifanyonetellmemodelname,enginespecs,yearsproductio
n,carmade,history,whateverinfofunkylookingcar,pleasee-mail.Thanks,-IL----brou
ghtneighborhoodLerxst----
```

In [57]:
```
data = "All work and no play makes jack dull boy. All work and no play makes j
ack a dull boy."
stopWords = set(stopwords.words('english'))
words = word_tokenize(data)
wordsFiltered = []

for w in words:
    if w not in stopWords:
        wordsFiltered.append(w)
print(wordsFiltered)
```

```
['All', 'work', 'play', 'makes', 'jack', 'dull', 'boy', '.', 'All', 'work',
'play', 'makes', 'jack', 'dull', 'boy', '.']
```

Q-F)RegEx

In [42]:
```
import re

print("Before Remove The punctuation in string\n-->")
print(df)
print("-----------------------------------------------")
res = re.sub(r'[^\w\s]', '', df)
print("\nAfter Remove The punctuation in string\n-->")
print(res)
```

```
Before Remove The punctuation in string
-->
From: lerxst@wam.umd.edu (where's my thing)
Subject: WHAT car is this!?
Nntp-Posting-Host: rac3.wam.umd.edu
Organization: University of Maryland, College Park
Lines: 15

 I was wondering if anyone out there could enlighten me on this car I saw
the other day. It was a 2-door sports car, looked to be from the late 60s/
early 70s. It was called a Bricklin. The doors were really small. In additio
n,
the front bumper was separate from the rest of the body. This is
all I know. If anyone can tellme a model name, engine specs, years
of production, where this car is made, history, or whatever info you
have on this funky looking car, please e-mail.

Thanks,
- IL
   ---- brought to you by your neighborhood Lerxst ----




-----------------------------------------------

After Remove The punctuation in string
-->
From lerxstwamumdedu wheres my thing
Subject WHAT car is this
NntpPostingHost rac3wamumdedu
Organization University of Maryland College Park
Lines 15

 I was wondering if anyone out there could enlighten me on this car I saw
the other day It was a 2door sports car looked to be from the late 60s
early 70s It was called a Bricklin The doors were really small In addition
the front bumper was separate from the rest of the body This is
all I know If anyone can tellme a model name engine specs years
of production where this car is made history or whatever info you
have on this funky looking car please email

Thanks
 IL
    brought to you by your neighborhood Lerxst
```

Q-G)Bag-of-Words

In [46]:
```
import pandas as pd
from sklearn.feature_extraction.text import CountVectorizer

s_tok = sent_tokenize(df)
cv = CountVectorizer()
cv_data = cv.fit([i for i in s_tok])
print("Bag OF Words:")
print()
print(cv_data.vocabulary_)
cv_tr = cv.transform([i for i in se])
pd.DataFrame(cv_tr.toarray(),columns=cv.get_feature_names())
```

```
Bag OF Words:

{'from': 24, 'lerxst': 38, 'wam': 79, 'umd': 77, 'edu': 21, 'where': 84, 'm
y': 47, 'thing': 74, 'subject': 69, 'what': 82, 'car': 14, 'is': 34, 'this':
75, 'nntp': 50, 'posting': 59, 'host': 29, 'rac3': 61, 'organization': 54, 'u
niversity': 78, 'of': 51, 'maryland': 44, 'college': 15, 'park': 57, 'lines':
39, '15': 0, 'was': 80, 'wondering': 85, 'if': 30, 'anyone': 5, 'out': 56, 't
here': 73, 'could': 16, 'enlighten': 23, 'me': 45, 'on': 52, 'saw': 64, 'th
e': 72, 'other': 55, 'day': 17, 'it': 35, 'door': 18, 'sports': 68, 'looked':
40, 'to': 76, 'be': 6, 'late': 37, '60s': 1, 'early': 20, '70s': 2, 'called':
12, 'bricklin': 8, 'doors': 19, 'were': 81, 'really': 62, 'small': 66, 'in':
32, 'addition': 3, 'front': 25, 'bumper': 10, 'separate': 65, 'rest': 63, 'bo
dy': 7, 'all': 4, 'know': 36, 'can': 13, 'tellme': 70, 'model': 46, 'name': 4
8, 'engine': 22, 'specs': 67, 'years': 86, 'production': 60, 'made': 42, 'his
tory': 28, 'or': 53, 'whatever': 83, 'info': 33, 'you': 87, 'have': 27, 'funk
y': 26, 'looking': 41, 'please': 58, 'mail': 43, 'thanks': 71, 'il': 31, 'bro
ught': 9, 'by': 11, 'your': 88, 'neighborhood': 49}
```

Out[46]:

	15	60s	70s	addition	all	anyone	be	body	bricklin	brought	...	wam	was	were	what
0	0	0	0	0	0	0	0	0	0	0	...	1	0	0	1
1	1	0	0	0	0	1	0	0	0	0	...	1	1	0	0
2	0	1	1	0	0	0	1	0	0	0	...	0	1	0	0
3	0	0	0	0	0	0	0	0	1	0	...	0	1	0	0
4	0	0	0	0	0	0	0	0	0	0	...	0	0	1	0
5	0	0	0	1	0	0	0	1	0	0	...	0	1	0	0
6	0	0	0	0	1	0	0	0	0	0	...	0	0	0	0
7	0	0	0	0	0	1	0	0	0	0	...	0	0	0	0
8	0	0	0	0	0	0	0	0	0	1	...	0	0	0	0

9 rows × 89 columns

Q-H)POS Tagging

In [48]:
```
from nltk import pos_tag

word_data=word_tokenize(res)
tokens_tag = pos_tag(word_data)
for i in tokens_tag:
    print(i)
```

```
('From', 'IN')
('lerxstwamumdedu', 'JJ')
('wheres', 'NNS')
('my', 'PRP$')
('thing', 'NN')
('Subject', 'NNP')
('WHAT', 'NNP')
('car', 'NN')
('is', 'VBZ')
('this', 'DT')
('NntpPostingHost', 'NNP')
('rac3wamumdedu', 'NN')
('Organization', 'NNP')
('University', 'NNP')
('of', 'IN')
('Maryland', 'NNP')
('College', 'NNP')
('Park', 'NNP')
('Lines', 'NNP')
('15', 'CD')
('I', 'PRP')
('was', 'VBD')
('wondering', 'VBG')
('if', 'IN')
('anyone', 'NN')
('out', 'IN')
('there', 'RB')
('could', 'MD')
('enlighten', 'VB')
('me', 'PRP')
('on', 'IN')
('this', 'DT')
('car', 'NN')
('I', 'PRP')
('saw', 'VBD')
('the', 'DT')
('other', 'JJ')
('day', 'NN')
('It', 'PRP')
('was', 'VBD')
('a', 'DT')
('2door', 'JJ')
('sports', 'NNS')
('car', 'NN')
('looked', 'VBD')
('to', 'TO')
('be', 'VB')
('from', 'IN')
('the', 'DT')
('late', 'JJ')
('60s', 'NNS')
('early', 'RB')
('70s', 'CD')
('It', 'PRP')
('was', 'VBD')
('called', 'VBN')
('a', 'DT')
```

```
('Bricklin', 'NNP')
('The', 'DT')
('doors', 'NNS')
('were', 'VBD')
('really', 'RB')
('small', 'JJ')
('In', 'IN')
('addition', 'NN')
('the', 'DT')
('front', 'NN')
('bumper', 'NN')
('was', 'VBD')
('separate', 'JJ')
('from', 'IN')
('the', 'DT')
('rest', 'NN')
('of', 'IN')
('the', 'DT')
('body', 'NN')
('This', 'DT')
('is', 'VBZ')
('all', 'DT')
('I', 'PRP')
('know', 'VBP')
('If', 'IN')
('anyone', 'NN')
('can', 'MD')
('tellme', 'VB')
('a', 'DT')
('model', 'NN')
('name', 'NN')
('engine', 'NN')
('specs', 'CD')
('years', 'NNS')
('of', 'IN')
('production', 'NN')
('where', 'WRB')
('this', 'DT')
('car', 'NN')
('is', 'VBZ')
('made', 'VBN')
('history', 'NN')
('or', 'CC')
('whatever', 'WDT')
('info', 'VBP')
('you', 'PRP')
('have', 'VBP')
('on', 'IN')
('this', 'DT')
('funky', 'NN')
('looking', 'VBG')
('car', 'NN')
('please', 'NN')
('email', 'VBP')
('Thanks', 'NNP')
('IL', 'NNP')
('brought', 'VBD')
```

```
('to', 'TO')
('you', 'PRP')
('by', 'IN')
('your', 'PRP$')
('neighborhood', 'NN')
('Lerxst', 'NN')
```

Q-I)N-grams

In [52]:
```
cv = CountVectorizer(ngram_range=(3, 3))
cv_data = cv.fit([i for i in se])
print("N-grams\n")
print(cv_data.vocabulary_)
cv_tr = cv.transform([i for i in s_tok])
pd.DataFrame(cv_tr.toarray(),columns=cv.get_feature_names())
```

N-grams

```
{'from lerxst wam': 23, 'lerxst wam umd': 41, 'wam umd edu': 88, 'umd edu whe
re': 86, 'edu where my': 20, 'where my thing': 96, 'my thing subject': 49, 't
hing subject what': 78, 'subject what car': 69, 'what car is': 94, 'car is th
is': 11, 'nntp posting host': 51, 'posting host rac3': 61, 'host rac3 wam': 3
0, 'rac3 wam umd': 63, 'umd edu organization': 85, 'edu organization universi
ty': 19, 'organization university of': 58, 'university of maryland': 87, 'of
maryland college': 52, 'maryland college park': 46, 'college park lines': 15,
'park lines 15': 60, 'lines 15 was': 42, '15 was wondering': 0, 'was wonderin
g if': 92, 'wondering if anyone': 98, 'if anyone out': 32, 'anyone out ther
e': 4, 'out there could': 59, 'there could enlighten': 77, 'could enlighten m
e': 16, 'enlighten me on': 22, 'me on this': 47, 'on this car': 55, 'this car
saw': 80, 'car saw the': 14, 'saw the other': 65, 'the other day': 75, 'it wa
s door': 39, 'was door sports': 90, 'door sports car': 17, 'sports car looke
d': 68, 'car looked to': 12, 'looked to be': 43, 'to be from': 83, 'be from t
he': 5, 'from the late': 24, 'the late 60s': 74, 'late 60s early': 40, '60s e
arly 70s': 1, 'it was called': 38, 'was called bricklin': 89, 'the doors wer
e': 72, 'doors were really': 18, 'were really small': 93, 'in addition the':
34, 'addition the front': 2, 'the front bumper': 73, 'front bumper was': 26,
'bumper was separate': 7, 'was separate from': 91, 'separate from the': 66,
'from the rest': 25, 'the rest of': 76, 'rest of the': 64, 'of the body': 54,
'this is all': 82, 'is all know': 36, 'if anyone can': 31, 'anyone can tellm
e': 3, 'can tellme model': 9, 'tellme model name': 70, 'model name engine': 4
8, 'name engine specs': 50, 'engine specs years': 21, 'specs years of': 67,
'years of production': 99, 'of production where': 53, 'production where thi
s': 62, 'where this car': 97, 'this car is': 79, 'car is made': 10, 'is made
history': 37, 'made history or': 45, 'history or whatever': 29, 'or whatever
info': 57, 'whatever info you': 95, 'info you have': 35, 'you have on': 101,
'have on this': 28, 'on this funky': 56, 'this funky looking': 81, 'funky loo
king car': 27, 'looking car please': 44, 'car please mail': 13, 'thanks il br
ought': 71, 'il brought to': 33, 'brought to you': 6, 'to you by': 84, 'you b
y your': 100, 'by your neighborhood': 8, 'your neighborhood lerxst': 102}
```

Out[52]:

	15 was wondering	60s early 70s	addition the front	anyone can tellme	anyone out there	be from the	brought to you	bumper was separate	by your neighborhood	can tellme model
0	0	0	0	0	0	0	0	0	0	0
1	1	0	0	0	1	0	0	0	0	0
2	0	1	0	0	0	1	0	0	0	0
3	0	0	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0	0	0
5	0	0	1	0	0	0	0	1	0	0
6	0	0	0	0	0	0	0	0	0	0
7	0	0	0	1	0	0	0	0	0	1
8	0	0	0	0	0	0	1	0	1	0

9 rows × 103 columns

Q. TF-IDF

In [54]:
```
from sklearn.feature_extraction.text import TfidfVectorizer
tf = TfidfVectorizer()
tf_data = tf.fit_transform([i for i in s_tok])
pd.DataFrame(tf_data.toarray(),columns=tf.get_feature_names())
```

Out[54]:

	15	60s	70s	addition	all	anyone	be	body	bricklin	brou
0	0.000000	0.00000	0.00000	0.000000	0.000000	0.000000	0.00000	0.000000	0.000000	0.000
1	0.194244	0.00000	0.00000	0.000000	0.000000	0.164062	0.00000	0.000000	0.000000	0.000
2	0.000000	0.29842	0.29842	0.000000	0.000000	0.000000	0.29842	0.000000	0.000000	0.000
3	0.000000	0.00000	0.00000	0.000000	0.000000	0.000000	0.00000	0.000000	0.564838	0.000
4	0.000000	0.00000	0.00000	0.000000	0.000000	0.000000	0.00000	0.000000	0.000000	0.000
5	0.000000	0.00000	0.00000	0.285264	0.000000	0.000000	0.00000	0.285264	0.000000	0.000
6	0.000000	0.00000	0.00000	0.000000	0.581208	0.000000	0.00000	0.000000	0.000000	0.000
7	0.000000	0.00000	0.00000	0.000000	0.000000	0.165600	0.00000	0.000000	0.000000	0.000
8	0.000000	0.00000	0.00000	0.000000	0.000000	0.000000	0.00000	0.000000	0.000000	0.350

9 rows × 89 columns

Python for Data Science Lab Session

Basic of Matplot

Q-1) Use Matplotlib package below tasks

In [2]:
```
v1 = [1,6,8,2,5,0,3,10,4,7]
v2 = [3,8,9,2,1,2,4,7,6,6]
```

In [161]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import matplotlib
```

Q-1-> A) Plot both the values using line graph

In [33]:
```
fig, axs = plt.subplots()    # It subplot the figure in array like we also giv
e more than one dimention axis (1,3 ) -> gives Three figure line by
axs.plot(v1,color='red',label='v1')

axs.plot(v2,color='cyan',label='v2')  # set color to cyan and give label
axs.legend()        # it locate the legend means (shows all the values in grap
h by it's row values)
plt.xlabel('X - axis ')  # Give the x label
plt.ylabel('Y - axis')  # Give the y label
fig.suptitle('Plotting')  # set title
plt.grid(True)   # plot  grid
plt.show()
```

In [34]:
```
fig, ax = plt.subplots()

line1, = ax.plot(v1,dashes=[6, 2], label='v1')

line2, = ax.plot(v2, dashes=[6, 2], label='v2')
plt.xlabel('X - axis ')
plt.ylabel('Y - axis')
ax.legend()
plt.grid(True)
plt.show()
```

In [41]:
```
fig, axs = plt.subplots(1, 3, figsize=(9, 3), sharey=True)  #sharey use for si
milar data in all diffrent type visulization Where x axis comman
fig.suptitle('Categorical Plotting')

axs[0].bar(v1,v2)
axs[1].scatter(v1,v2)
axs[2].plot(v1,v2)  # here x and y axis is needed that why just for practise

plt.show()
```

Q-1 -> B) Change the axes limit and ticks and observe the result

In [60]:
```
fig, ax = plt.subplots()
ax.set_xticks([1,2,3,4,5,6,7,8,9,10])  # here we assige x and y values
ax.set_yticks([1,2,3,4,5,6,7,8,9,10])
ax.plot(v1)
ax.plot(v2)

plt.show()
```

Q-1 -> C) Observe the resulting image grid in plot

In [76]:
```
fig, axes = plt.subplots(1,3, figsize = (12,4))

axes[0].plot(v1)
axes[0].grid(True)
axes[0].set_title('Default grid')

axes[1].plot(v1, 'r')
axes[1].grid(color='c', lw = 0.25) #lw for light opecity
axes[1].set_title('custom grid')

axes[2].plot(v1)
axes[2].set_title('no grid')

fig.tight_layout()
plt.show()
```

Q-1 -> D) Use lines of diffrent color to represent each values

In [91]:
```
colors = ['#1f77b4',
          '#ff7f0e',
          '#2ca02c',
          '#d62728',
          '#9467bd',
          '#8c564b',
          '#e377c2',
          '#7f7f7f',
          '#bcbd22',
          '#17becf',
          '#1a55FF']


fig = plt.figure()
ax = fig.add_axes([0.1, 0.1, 0.6, 0.75])

x = np.linspace(0, 2 * np.pi)
offsets = np.linspace(0, 2 * np.pi, 11, endpoint=False)
yy = np.transpose([np.sin(x + phi) for phi in offsets])

for i in range(len(v1)):
    ax.plot(yy[:, i], label=str(v1[i]))
    ax.legend(bbox_to_anchor=(1.05, 1) )

plt.title('Make curve with diffrent color')

plt.show()
```

In [ ]:

Q-1 -> E) Use diffrent markers for both the lines

In [117]:
```
fig, ax = plt.subplots()

line1, = ax.plot(v1,marker='D', label='v1') #use Marker function

line2, = ax.plot(v2, marker='*', label='v2')
plt.xlabel('X - axis ')
plt.ylabel('Y - axis')
ax.legend()
plt.grid(True)
plt.show()
```

In [126]:
```
plt.plot(v1, color='lightcoral', marker='8', markeredgecolor='black')

plt.plot(v2 ,color='#4b0082', linewidth=4,marker='h', markerfacecolor='lightgr
een', markeredgewidth=2,markersize=12, markevery=3)

plt.show()
```

Q-1 -> F) Use diffrent lines styles to represent both lines

In [150]:
```
plt.plot(v1, color='lightcoral', linestyle=':')
```

Out[150]:
```
[<matplotlib.lines.Line2D at 0x1501e2ce7b8>]
```

In [151]:
```
plt.plot(v1,color='lightcoral', linestyle='-.')
```

Out[151]:
```
[<matplotlib.lines.Line2D at 0x1501d07ae80>]
```

In [153]:
```
plt.plot(v1,color='lightcoral', linestyle='--')
```

Out[153]:
```
[<matplotlib.lines.Line2D at 0x1501e535128>]
```

In [156]:
```
plt.plot(v1,color='lightcoral', linestyle='solid')
```

Out[156]:
```
[<matplotlib.lines.Line2D at 0x1501ce1a240>]
```

In [141]:
```
from matplotlib import style
style.use('ggplot')
fig, axs = plt.subplots()    # It subplot the figure in array like we also giv
e more than one dimention axis (1,3 ) -> gives Three figure line by
axs.plot(v1,color='red',label='v1')

axs.plot(v2,color='cyan',label='v2')  # set color to cyan and give label
axs.legend()        # it locate the legend means (shows all the values in grap
h by it's row values)
plt.xlabel('X - axis ')  # Give the x label
plt.ylabel('Y - axis')  # Give the y label
fig.suptitle('Plotting')  # set title
plt.grid(True)   # plot  grid
plt.show()
```

In [142]:
```
style.use('fivethirtyeight')
fig, axs = plt.subplots()    # It subplot the figure in array like we also giv
e more than one dimention axis (1,3 ) -> gives Three figure line by
axs.plot(v1,color='red',label='v1')

axs.plot(v2,color='cyan',label='v2')  # set color to cyan and give label
axs.legend()        # it locate the legend means (shows all the values in grap
h by it's row values)
plt.xlabel('X - axis ')  # Give the x label
plt.ylabel('Y - axis')  # Give the y label
fig.suptitle('Plotting')  # set title
plt.grid(True)   # plot  grid
plt.show()
```

In [145]:
```
style.use('dark_background')
fig, axs = plt.subplots()    # It subplot the figure in array like we also giv
e more than one dimention axis (1,3 ) -> gives Three figure line by
axs.plot(v1,color='red',label='v1')

axs.plot(v2,color='cyan',label='v2')  # set color to cyan and give label
axs.legend()        # it locate the legend means (shows all the values in grap
h by it's row values)
plt.xlabel('X - axis ')  # Give the x label
plt.ylabel('Y - axis')  # Give the y label
fig.suptitle('Plotting')  # set title
plt.grid(True)   # plot  grid
plt.show()
```

Q-1 -> G) Save work in image in hard drive

In [160]:
```
fig.savefig('plot.png') # it save last image
```

In [169]:
```
import matplotlib.image as mpimg
plt.imshow(mpimg.imread('plot.png'))
```

Out[169]:
```
<matplotlib.image.AxesImage at 0x1501cc74438>
```

Python for Data Science (Lab Session 12)

Basics of Data Visualization

Q-1) Read the financial data of Alphabet Inc. between October 3, 2016 to October 7, 2016 from fdata.csv file.

In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
```

Data load

In [2]:
```
df = pd.read_csv("fdata.csv")
df
```

Out[2]:

	Date	Open	High	Low	Close
0	10-03-16	774.250000	776.065002	769.500000	772.559998
1	10-04-16	776.030029	778.710022	772.890015	776.429993
2	10-05-16	779.309998	782.070007	775.650024	776.469971
3	10-06-16	779.000000	780.479980	775.539978	776.859985
4	10-07-16	779.659973	779.659973	770.750000	775.080017

Q-A)Make the date column as the index column

In [3]:
```
date = df.set_index("Date")
date
```

Out[3]:

	Open	High	Low	Close
Date				
10-03-16	774.250000	776.065002	769.500000	772.559998
10-04-16	776.030029	778.710022	772.890015	776.429993
10-05-16	779.309998	782.070007	775.650024	776.469971
10-06-16	779.000000	780.479980	775.539978	776.859985
10-07-16	779.659973	779.659973	770.750000	775.080017

Q-B)Write a Python program to draw line charts of the financial datawith dates on x-axis and open, high, low and close values on y-axis

In [4]:
```
plt.plot(date)
plt.show()
```

Q-C)Give the appropriate title to the graph and give proper labels to the x-axis and y-axis

In [5]:
```
plt.title("Financial data of Alphabet Inc",color="m")
plt.plot(date)
plt.xlabel("Dates")
plt.ylabel("Values")
plt.show()
```

Q-D)Choose different colors for data with different dates

In [6]:
```
plt.title("Financial data of Alphabet Inc")
plt.xlabel("Dates")
plt.ylabel("Values")
date1 = date.loc["10-03-16"]
date2 = date.loc["10-04-16"]
date3 = date.loc["10-05-16"]
date4 = date.loc["10-06-16"]
date5 = date.loc["10-07-16"]

plt.plot(date1, color='r',marker='p',alpha=.2)
plt.plot(date2, color='r',marker='s',alpha=.3)
plt.plot(date3, color='g',marker='h',alpha=.4)
plt.plot(date4, color='b',marker='*',alpha=.5)
plt.plot(date5, color='m',marker='D',alpha=.6)
plt.show()
```

Q-E)Display the legend and give annotations to the chart

In [7]:
```
plt.title("Financial data of Alphabet Inc")
plt.xlabel("Dates")
plt.ylabel("Values")
date1 = date.loc["10-03-16"]
date2 = date.loc["10-04-16"]
date3 = date.loc["10-05-16"]
date4 = date.loc["10-06-16"]
date5 = date.loc["10-07-16"]

plt.plot(date1, color='r',marker='p',alpha=.2,label="10-03-16")
plt.plot(date2, color='r',marker='s',alpha=.3,label="10-04-16")
plt.plot(date3, color='g',marker='h',alpha=.4,label="10-05-16")
plt.plot(date4, color='b',marker='*',alpha=.5,label="10-06-16")
plt.plot(date5, color='m',marker='D',alpha=.6,label="10-07-16")
plt.legend()
plt.show()
```

Q-2)Use below values for visualization Programminglanguages:

Java, Python, PHP, JavaScript, C#, C++

Popularity:22.2, 17.6, 8.8, 8, 7.7, 6.7

Q-2.A) Make a pie chart to display above data

In [8]:
```
lan = ['Java', 'Python', 'PHP', 'JavaScript', 'C#', 'C++']

popularity = [22.2, 17.6, 8.8, 8, 7.7, 6.7]

fig = plt.figure(figsize =(10, 7))
plt.pie(popularity, labels = lan,autopct="%1.1f%%")
plt.show()
```

Q-2.B)Use user defined colors to represent each value

In [9]:
```
colors = ['cyan','red','yellow','blue','gray','purple']

fig = plt.figure(figsize =(10, 7))
plt.pie(popularity, labels = lan,autopct="%1.1f%%",colors=colors)
plt.show()
```

Q-2.C)Give labels to the data

In [10]:
```
fig = plt.figure(figsize =(10, 7))
plt.pie(popularity, labels = lan,autopct="%1.1f%%", colors=colors)
plt.legend(title ="language",loc ="center left",bbox_to_anchor =(1, 0, 0.5, 1
))
plt.show()
```

Q-2.D)Give proper title to the graph

In [11]:
```
fig = plt.figure(figsize =(10, 7))
plt.pie(popularity, labels = lan,autopct="%1.1f%%", colors=colors)
plt.title("Programming languages with Popularity")
plt.show()
```

Q-2.E)Display wedges for Java and Python with different values of explode parameter for both languages.

In [12]:
```
ex = [0.1,0,0,0,0,0]  #inwhich perticular , if we want to hilight some part th
en give values of it
fig = plt.figure(figsize =(10, 7))
plt.pie(popularity, labels = lan,autopct="%1.1f%%",explode=ex)
plt.legend(title ="language",loc ="center left",bbox_to_anchor =(1, 0, 0.5, 1
))
plt.title("Programming languages with Popularity")
plt.show()
```

Q-3)Create bar chart from above data

In [13]:
```
plt.bar(lan,popularity)
plt.show()
```

Q-3.A)Change the width of the bar for Python

In [14]:
```
w = [.1,.2,.3,.4,.5,.6]
plt.bar(lan,popularity,width=w)
```

Out[14]: <BarContainer object of 6 artists>

Q-3.B)Give proper title to the graph and labels to the x and y axis

In [15]:
```
plt.bar(lan,popularity,color='lightblue')
plt.title("Programing language with Popularity")
plt.xlabel("Language")
plt.ylabel("Popularity")
plt.show()
```

Q-3.C)Use different colors to represent each bar

In [16]:
```
plt.bar(lan,popularity,color=colors)
plt.title("Programing language with Popularity")
plt.xlabel("Language")
plt.ylabel("Popularity")
plt.show()
```

Python for Data Science LAB Session - 13

Basics of Data Visualization

Q-1) Read the data from company_sales_data.csv file. Concider all the "?" and "n.a." values as NaN values.

a. Read all the profit data of each month and display it using the scatter plot

b. Create a line on the scatterplot that displays the general direction of the profit using polyfit()

In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import networkx as nx
```

In [2]:
```
df = pd.read_csv("company_sales_data.csv", index_col=0, na_values=['?', 'n.a.'])
df = df.fillna(0)   #load the data and it fill as 0 ,where value ? or n.a.
```

In [3]:
```
df
```

Out[3]:

month_number	Date	facecream	facewash	toothpaste	bathingsoap	shampoo	moisturizer	total_units	total_profit
1	05-01-2020	2500	1500	5200	9200.0	1200.0	1500	21100.0	211000.0
2	05-02-2020	2630	1200	5100	6100.0	2100.0	1200	18330.0	183300.0
3	05-03-2020	2140	1340	4550	9550.0	0.0	1340	0.0	224700.0
4	05-04-2020	3400	1130	5870	8870.0	1870.0	1130	22270.0	222700.0
5	05-05-2020	3600	1740	4560	0.0	1560.0	1740	20960.0	209600.0
6	05-06-2020	2760	1555	4890	7490.0	1890.0	1555	20140.0	0.0
7	05-07-2020	2980	1120	4780	8980.0	1780.0	1120	29550.0	295500.0
8	05-08-2020	3700	1400	5860	9960.0	2860.0	1400	36140.0	361400.0
9	05-09-2020	3540	1780	6100	8100.0	2100.0	1780	23400.0	234000.0
10	05-10-2020	1990	1890	8300	0.0	2300.0	1890	0.0	266700.0
11	05-11-2020	2340	2100	7300	13300.0	2400.0	2100	41280.0	412800.0
12	05-12-2020	2900	1760	7400	14400.0	0.0	1760	30020.0	300200.0

In [4]:
```
plt.scatter(df.Date, df.total_profit)      #Scatter plot
plt.xticks(rotation=90)  #It rotate x value to 90
plt.xlabel("Month")
plt.ylabel("Total Profit")
plt.show()
```

In [5]:
```
fig, con = np.polyfit(df.total_units, df.total_profit, 1)
plt.plot(df.total_units, df.total_profit, '*')
plt.plot(df.total_units, fig * df.total_units + con)  #that make's the y= m*x + c line equation
plt.xticks(rotation=90)
plt.xlabel("Total Units")
plt.ylabel("Total Profit")
plt.show()
```

Q-2)Create a dataframe taking facecream and toothpaste sales data from the original dataframe with date as index column.

a. Create a time series graph that shows facecream and toothpaste sales data on each data. Use different style and markers for facecream and toothpaste data

b. Create two columns facecreame-trend and toothpaste-trend that contains the value of general trends over time and display both trends as lines on top of the timesries graph.

In [6]:
```
df1 = pd.DataFrame({"face_cream": df.facecream.to_list(), "toothpaste": df.toothpaste.to_list()}, index=df.Date)
# here we arrange face cream and toothpaste date to dataframe by date wise
```

In [7]:
```
df1
```

Out[7]:

Date	face_cream	toothpaste
05-01-2020	2500	5200
05-02-2020	2630	5100
05-03-2020	2140	4550
05-04-2020	3400	5870
05-05-2020	3600	4560
05-06-2020	2760	4890
05-07-2020	2980	4780
05-08-2020	3700	5860
05-09-2020	3540	6100
05-10-2020	1990	8300
05-11-2020	2340	7300
05-12-2020	2900	7400

In [8]:
```
plt.plot(df1.index, df1.toothpaste, label='Toothpaste',marker='o')  #plor the data
plt.plot(df1.index, df1.face_cream, label='Face Cream',marker='*')
plt.xticks(rotation=90)
plt.legend()
plt.show()
```

In [9]:
```
facecream_inc = np.array(np.random.rand(12) * 2000, dtype=np.int32) # these both varible genrate randome value between 2000
toothpaste_inc = np.array(np.random.rand(12) * 2000, dtype=np.int32)

trend = df1
trend['facecream_inc'] = facecream_inc
trend['toothpaste_inc'] = toothpaste_inc

trend.plot()
plt.xticks(rotation=90)
```

Out[9]:
```
(array([-2.,  0.,  2.,  4.,  6.,  8., 10., 12.]),
 [Text(-2.0, 0, '05-11-2020'),
  Text(0.0, 0, '05-01-2020'),
  Text(2.0, 0, '05-03-2020'),
  Text(4.0, 0, '05-05-2020'),
  Text(6.0, 0, '05-07-2020'),
  Text(8.0, 0, '05-09-2020'),
  Text(10.0, 0, '05-11-2020'),
  Text(12.0, 0, '')])
```

Q-3)Read the total profit of each month and show it using the histogram to see most common profit ranges.

Explore different parameters of hist() and its uses.

In [10]:
```
total = df.total_profit.to_list()
rang = [100000, 105000, 200000, 205000, 300000, 400000, 420000]

plt.hist(total, rang,alpha=0.5)
plt.grid() #grid
plt.show()
```

In [11]:
```
rang0 = [100000, 105000, 200000, 205000, 300000,350000, 400000, 420000]
plt.hist(total, rang, color='lightblue', alpha=0.8) #color used for give color and alpha for opacity
plt.show()
```

In [12]:
```
rang1 = [200000,305000, 346000, 450000, 490000]
plt.hist(total, rang, facecolor='y', alpha=0.75, label="Histrogram 1")
plt.hist(total, rang1, facecolor='b', alpha=0.25, label="Histrogram 2")
plt.grid(True)
plt.legend()
plt.show()
```

In [13]:
```
plt.hist(total, rang, color='lightblue', alpha=0.8, orientation='horizontal') #oreientation based horizantal/vertical
plt.grid()
plt.show()
```

In [14]:
```
fig, axs = plt.subplots(1, 3, figsize=(9, 3), sharey=True)  #shray for all the y axis are same value
axs[0].hist(total, rang, facecolor='r', alpha=.8, align='left')   #alignment from left right center side
axs[1].hist(total, rang, facecolor='g', alpha=.6, align='mid')
axs[2].hist(total, rang, facecolor='b', alpha=.5, align='right')
fig.suptitle('Histrogram alignment')
```

Out[14]:
```
Text(0.5, 0.98, 'Histrogram alignment')
```

Q-4) Create an undirected graph from below adjacency matrix using networkx.

In [15]:
```
Graph = nx.Graph([(1,2), (1,3), (1,4), (2,1), (2,3), (3,1), (3,2), (4,1)])
```

In [16]:
```
nx.draw(Graph, with_labels = True)
```

Q-5) Create below Directed graph with appropriate labels and colors using networkx

In [17]:
```
grp = nx.DiGraph()
colors = ['green', 'green', 'green', 'green', 'purple', 'red', 'purple', 'red']

grp.add_nodes_from([2, 3, 4, 5, 6, "Begin", "Pre-Final", "End"]) # added all value
grp.add_edges_from([(4,5), ("Pre-Final", "End"), ("Begin", 2), ("Begin", 4),(6, "Pre-Final"), (5, 6), (3, 6), (2, 3)]) #show all path

nx.draw(grp, with_labels=True, node_color=colors,node_size=[900, 900, 900, 900, 1000, 1100, 1500, 1100],node_shape='d')#draw graph
```

Q-6)Perform below operations to create a box plot

a. Create a numpy array of size 100 that contains random numbers in the range of 50 to 200

b. Create a numpy array of size 75 that contains all 1s

c. Create a numpy array of size 12 that contains random numbers in the range of 200 to 240

d. Create a numpy array of size 10 that contains random numbers in the range of -10 to -70

e. Concatenate all these arrays and create a single array

f. Draw a default boxplot and display all outliers as red color x mark

g. Create a notched boxplot using same data and observe the difference.

In [18]:
```
arr = np.random.uniform(50, 200, 100)
```

In [19]:
```
arr
```

Out[19]:
```
array([113.15056793, 140.77590247,  60.5034521 , 197.22488   ,
        51.15614227, 170.29391859, 152.91087957, 187.39701841,
        55.86099951, 164.79006063,  69.22318367, 181.03859641,
       199.14759506, 137.23805344, 172.62993564, 173.48442914,
       180.96435542,  53.60544163,  75.13988941, 184.06386345,
       135.47333829, 129.11072049, 114.36612423, 168.75342476,
       190.9918443 , 110.34148697, 123.69300417, 132.35997835,
       164.55178846, 157.01017192,  71.17567739, 153.91356572,
       165.62090968, 102.69938877,  68.1002667 ,  96.70351594,
       142.57762451,  97.86636698, 135.09085239,  97.26792568,
       185.72053692, 117.98275519,  58.63892761, 187.02238867,
       167.89192305, 133.10488793,  72.50736343, 109.53255117,
       148.61216382,  93.06422446,  77.73135218, 181.85093558,
       128.05109076,  58.86842493, 166.11825082, 169.20979258,
       113.30632012, 143.60021659, 161.60294145, 156.70385674,
        50.58784487, 141.16265257, 170.98598691, 112.98759453,
       196.20683907, 184.16194723, 120.28577441, 171.26471206,
       178.28017799, 161.89949743,  91.94606712,  68.37333884,
       156.11848099, 149.47242942,  90.33746749, 129.8902996 ,
       148.95875111, 164.5224345 , 182.5713298 , 115.06778499,
       192.53565123,  57.6974238 ,  60.30473873, 195.7891697 ,
        70.59475063, 157.13744429, 163.16189897, 177.57884183,
       124.93011239, 179.7283863 , 118.70474584, 140.34948659,
       102.20296611, 180.91179053,  84.17927362, 108.3259271 ,
       113.68585379,  71.90073507, 138.66019836, 157.93815018])
```

In [20]:
```
arr1 = np.ones(75) * 100
arr1
```

Out[20]:
```
array([100., 100., 100., 100., 100., 100., 100., 100., 100., 100., 100.,
       100., 100., 100., 100., 100., 100., 100., 100., 100., 100., 100.,
       100., 100., 100., 100., 100., 100., 100., 100., 100., 100., 100.,
       100., 100., 100., 100., 100., 100., 100., 100., 100., 100., 100.,
       100., 100., 100., 100., 100., 100., 100., 100., 100., 100., 100.,
       100., 100., 100., 100., 100., 100., 100., 100., 100., 100., 100.,
       100., 100., 100., 100., 100., 100., 100., 100., 100.])
```

In [21]:
```
arr2 = np.random.uniform(200, 240, 12)
arr2
```

Out[21]:
```
array([207.23703817, 235.26988763, 217.45213723, 230.79327922,
       210.50274499, 213.98164139, 216.14676086, 205.41623759,
       219.33600333, 222.06102462, 227.65764844, 212.00982013])
```

In [22]:
```
arr3 = np.random.uniform(-70, -10, 10)
arr3
```

Out[22]:
```
array([-46.75494237, -13.89462009, -62.3127355 , -61.1265556 ,
       -44.42635847, -53.91531869, -21.99466615, -36.08196174,
       -68.27524554, -14.59624308])
```

In [23]:
```
concate = np.concatenate((arr,arr1,arr2,arr3))
concate
```

Out[23]:
```
array([113.15056793, 140.77590247,  60.5034521 , 197.22488   ,
        51.15614227, 170.29391859, 152.91087957, 187.39701841,
        55.86099951, 164.79006063,  69.22318367, 181.03859641,
       199.14759506, 137.23805344, 172.62993564, 173.48442914,
       180.96435542,  53.60544163,  75.13988941, 184.06386345,
       135.47333829, 129.11072049, 114.36612423, 168.75342476,
       190.9918443 , 110.34148697, 123.69300417, 132.35997835,
       164.55178846, 157.01017192,  71.17567739, 153.91356572,
       165.62090968, 102.69938877,  68.1002667 ,  96.70351594,
       142.57762451,  97.86636698, 135.09085239,  97.26792568,
       185.72053692, 117.98275519,  58.63892761, 187.02238867,
       167.89192305, 133.10488793,  72.50736343, 109.53255117,
       148.61216382,  93.06422446,  77.73135218, 181.85093558,
       128.05109076,  58.86842493, 166.11825082, 169.20979258,
       113.30632012, 143.60021659, 161.60294145, 156.70385674,
        50.58784487, 141.16265257, 170.98598691, 112.98759453,
       196.20683907, 184.16194723, 120.28577441, 171.26471206,
       178.28017799, 161.89949743,  91.94606712,  68.37333884,
       156.11848099, 149.47242942,  90.33746749, 129.8902996 ,
       148.95875111, 164.5224345 , 182.5713298 , 115.06778499,
       192.53565123,  57.6974238 ,  60.30473873, 195.7891697 ,
        70.59475063, 157.13744429, 163.16189897, 177.57884183,
       124.93011239, 179.7283863 , 118.70474584, 140.34948659,
       102.20296611, 180.91179053,  84.17927362, 108.3259271 ,
       113.68585379,  71.90073507, 138.66019836, 157.93815018,
       100.        , 100.        , 100.        , 100.        ,
       100.        , 100.        , 100.        , 100.        ,
       100.        , 100.        , 100.        , 100.        ,
       100.        , 100.        , 100.        , 100.        ,
       100.        , 100.        , 100.        , 100.        ,
       100.        , 100.        , 100.        , 100.        ,
       100.        , 100.        , 100.        , 100.        ,
       100.        , 100.        , 100.        , 100.        ,
       100.        , 100.        , 100.        , 100.        ,
       100.        , 100.        , 100.        , 100.        ,
       100.        , 100.        , 100.        , 100.        ,
       100.        , 100.        , 100.        , 100.        ,
       100.        , 100.        , 100.        , 100.        ,
       100.        , 100.        , 100.        , 100.        ,
       100.        , 100.        , 100.        , 100.        ,
       100.        , 100.        , 100.        , 100.        ,
       100.        , 100.        , 100.        , 100.        ,
       100.        , 100.        , 100.        , 207.23703817,
       235.26988763, 217.45213723, 230.79327922, 210.50274499,
       213.98164139, 216.14676086, 205.41623759, 219.33600333,
       222.06102462, 227.65764844, 212.00982013, -46.75494237,
       -13.89462009, -62.3127355 , -61.1265556 , -44.42635847,
       -53.91531869, -21.99466615, -36.08196174, -68.27524554,
       -14.59624308])
```

In [24]:
```
plt.boxplot(concate, 0, 'b*')
plt.title('Box Plot')
plt.show()
```

In [25]:
```
plt.boxplot(concate, 1, 'gx')
plt.title('Box Plot 1')
plt.show()
```

In [ ]:

Python for Data Science (Lab Session-14 - 2)

In [ ]:

Working With EDA approch

Q-2) EDA Process

A => Create bag of words for any two sentences using CountVectorizer and Hashing Vectorizer.

Compare the performance considering execution time and memory usage

In [15]:
```
from nltk.tokenize import sent_tokenize
import pandas as pd
import time
from sklearn.feature_extraction.text import CountVectorizer,HashingVectorizer

st_time = time.time()
mydata = "I am Arjun Vankani.Here nowday's working with Ml. Wish to do bettter"
se = sent_tokenize(mydata)

for i in se:
    print(i)
cv = CountVectorizer()
cv_data = cv.fit([i for i in se])
print("Bag OF Words:")

cv_tr = cv.transform([i for i in se])
pd.DataFrame(cv_tr.toarray(),columns=cv.get_feature_names())
end_time = time.time()
tot_time = end_time - st_time
print("Totle Execution Time Of CountVectorizer",tot_time)
```

```
I am Arjun Vankani.Here nowday's working with Ml.
Wish to do bettter
Bag OF Words:
Totle Execution Time Of CountVectorizer 0.010001182556152344
```

In [16]:
```
hs_st_time = time.time()
text = ["This is Example sentence."]
vt = HashingVectorizer(n_features=10)
ans = vt.transform(text)
print(ans.toarray())
hs_end_time = time.time()
hs_tot_time=hs_end_time-hs_st_time
print("Totle Execution Time Of Hashing Vectorizer",hs_tot_time)
```

```
[[ 0.   0.   0.5  0.   0.   0.   0.   0.5 -0.5 -0.5]]
Totle Execution Time Of Hashing Vectorizer 0.002001047134399414
```

B => Measure central tendency,variance and range from iris dataset. Also print quantile valuesto display min, max, median, 25% and 75%value of the data distribution

In [17]:
```
from sklearn.datasets import load_iris
iris = load_iris()
iris_data = iris.data
print(iris_data)
```

```
 [5.  3.5 1.3 0.3]
 [4.5 2.3 1.3 0.3]
 [4.4 3.2 1.3 0.2]
 [5.  3.5 1.6 0.6]
 [5.1 3.8 1.9 0.4]
 [4.8 3.  1.4 0.3]
 [5.1 3.8 1.6 0.2]
 [4.6 3.2 1.4 0.2]
 [5.3 3.7 1.5 0.2]
 [5.  3.3 1.4 0.2]
 [7.  3.2 4.7 1.4]
 [6.4 3.2 4.5 1.5]
 [6.9 3.1 4.9 1.5]
 [5.5 2.3 4.  1.3]
 [6.5 2.8 4.6 1.5]
 [5.7 2.8 4.5 1.3]
 [6.3 3.3 4.7 1.6]
 [4.9 2.4 3.3 1. ]
 [6.6 2.9 4.6 1.3]
 [5.2 2.7 3.9 1.4]
```

In [18]:
```
print("Varience Of Iris Data",np.var(iris_data))
print("Range Of Iris Data",np.ptp(iris_data))
print("Minimum Of Iris Dataset Element",np.amin(iris_data))
print("Maximum Of Iris Dataset Element:",np.amax(iris_data))
print("Median Of Iris Dataset Element:",np.median(iris_data))
```

```
Varience Of Iris Data 3.896056416666667
Range Of Iris Data 7.800000000000001
Minimum Of Iris Dataset Element 0.1
Maximum Of Iris Dataset Element: 7.9
Median Of Iris Dataset Element: 3.2
```

In [19]:
```
import pandas as pd
da = pd.DataFrame(iris_data)
da.describe()
(iris_data.sum()*25)/100
print("Covariance Of Iris Dataset Element:\n",np.cov(iris_data))
```

```
Covariance Of Iris Dataset Element:
 [[4.75       4.42166667 4.35333333 ... 2.915      2.475      2.6       ]
 [4.42166667 4.14916667 4.055      ... 2.95583333 2.50416667 2.62833333]
 [4.35333333 4.055      3.99       ... 2.68833333 2.28166667 2.39666667]
 ...
 [2.915      2.95583333 2.68833333 ... 4.18916667 3.65083333 3.835     ]
 [2.475      2.50416667 2.28166667 ... 3.65083333 3.20916667 3.375     ]
 [2.6        2.62833333 2.39666667 ... 3.835      3.375      3.55      ]]
```

C => Calculate covariance and correlation of the dataset

In [20]:
```
print("Covariance Of Iris Dataset Element:\n",np.cov(iris_data))
print("Correlation Of Iris Dataset Element:\n",np.corrcoef(iris_data))
```

```
Covariance Of Iris Dataset Element:
 [[4.75       4.42166667 4.35333333 ... 2.915      2.475      2.6       ]
 [4.42166667 4.14916667 4.055      ... 2.95583333 2.50416667 2.62833333]
 [4.35333333 4.055      3.99       ... 2.68833333 2.28166667 2.39666667]
 ...
 [2.915      2.95583333 2.68833333 ... 4.18916667 3.65083333 3.835     ]
 [2.475      2.50416667 2.28166667 ... 3.65083333 3.20916667 3.375     ]
 [2.6        2.62833333 2.39666667 ... 3.835      3.375      3.55      ]]
Correlation Of Iris Dataset Element:
 [[1.         0.99599866 0.99997391 ... 0.65347343 0.6339168  0.63315839]
 [0.99599866 1.         0.99660709 ... 0.70898277 0.68625679 0.68483481]
 [0.99997391 0.99660709 1.         ... 0.65755616 0.63763128 0.6368058 ]
 ...
 [0.65347343 0.70898277 0.65755616 ... 1.         0.99570813 0.99446012]
 [0.6339168  0.68625679 0.63763128 ... 0.99570813 1.         0.99991588]
 [0.63315839 0.68483481 0.6368058  ... 0.99446012 0.99991588 1.        ]]
```