EXPERIMENT NO :- 01

AIM:
To create a list of numbers 1–100 that are either divisible by 5 or 6

SOFTWARE REQUIRED:
Python

THEORY:
A number when divided by 5 leaves a remainder zero is said to be divisible by 5 .
A number when divided by 6 leaves a remainder zero is said to be divisible by 6.

PROGRAM:
```
n=[]
for i in range(101):
    if ((i%5==0)or(i%6==0))and(i!=0):
        n.append(i)
print('the count of numbers divisible either by 6 or 5 is: ',len(n))
print('the numbers are: ')
for i in range(len(n)):
    print(n[i],end=” “)
```

OUTPUT:

RESULT:
Hence,the list of numbers which are divisible by either 5 or 6 is obtained.

AIM:
To read a list containing integers. And generate a list containing square of those integers and concatenate the 2 lists

SOFTWARE REQUIRED:
Python

THEORY:
Two strings can be concatenated in python by simply using the ‘+’ operator between them. More than 2 strings can be concatenated using ‘+’ operator.

PROGRAM:
```
 list=[]
sql=[]
n=int(input("no.of integers: "))
for i in range(n):
    a=int(input())
    list.append(a)
print('created list: ',list)
```

```
for i in range(n):
    sql.append(list[i]**2)
print('square of  integers: ',sql)
sql=list+sql
print('the concatenated list: ',sql)
```

OUTPUT:
RESULT:

Hence,a list of numbers is created ,their squares are found and the two lists are

Concatenated

EXPERIMENT NO :- 02

AIM: linear search
PROGRAM:

```
l=[]
n=int(input("Enter no of elements in the list "))
for i in range(0,n):
    print("Enter the value")
    l.append(int(input()))
key = int(input("Enter the element to search "))
found=0
for i in range(0,n):
    if l[i]==key:
        found=found+1
if found>0:
    print("Element Found in the list ")
else:
    print("Element Not Found in the list ")
```

OUTPUT:

AIM: binary search
PROGRAM:

```
def binary_search():
  n=int (input("enter the number of elements : "))
  data =[]
  for i in range(n):
    v=int(input())
    data.append(v)
  data.sort()
  key=int(input("enter the element to be found :"))
  low=0;high=len(data)-1;mid=0
  while low<=high:
    mid=(low+high)//2
    if data[int(mid)]==key:
      print("element found at index :",mid)
      break
    elif data[int(mid)]<key:
      low=mid+1
    else:
      high=mid-1
  else:
    print("element not found")

binary_search()
```

OUTPUT:

AIM: merge sort

PROGRAM:

```
def mergeSort(nlist):
    print("Splitting ",nlist)
    if len(nlist)>1:
        mid = len(nlist)//2
        lefthalf = nlist[:mid]
        righthalf = nlist[mid:]

        mergeSort(lefthalf)
        mergeSort(righthalf)
        i=j=k=0
        while i < len(lefthalf) and j < len(righthalf):
            if lefthalf[i] < righthalf[j]:
                nlist[k]=lefthalf[i]
                i=i+1
            else:
                nlist[k]=righthalf[j]
                j=j+1
            k=k+1

        while i < len(lefthalf):
            nlist[k]=lefthalf[i]
            i=i+1
            k=k+1

        while j < len(righthalf):
            nlist[k]=righthalf[j]
            j=j+1
            k=k+1
    print("Merging ",nlist)

nlist = [14,46,43,27,57,41,45,21,70]
mergeSort(nlist)
print(nlist)
```

OUTPUT:

AIM: insertion sort

PROGRAM:

```
def insertionSort(nlist):
   for index in range(1,len(nlist)):

     currentvalue = nlist[index]
     position = index

     while position>0 and nlist[position-1]>currentvalue:
         nlist[position]=nlist[position-1]
         position = position-1

     nlist[position]=currentvalue

nlist = [14,46,43,27,57,41,45,21,70]
insertionSort(nlist)
print(nlist)
```

OUTPUT:

AIM: bubble sort

PROGRAM:

```
def bubbleSort(nlist):
    for passnum in range(len(nlist)-1,0,-1):
        for i in range(passnum):
            if nlist[i]>nlist[i+1]:
                temp = nlist[i]
                nlist[i] = nlist[i+1]
                nlist[i+1] = temp

nlist = [14,46,43,27,57,41,45,21,70]
bubbleSort(nlist)
print(nlist)
```

OUTPUT:

AIM: selection sort

PROGRAM:

```
def selectionSort(nlist):
   for fillslot in range(len(nlist)-1,0,-1):
       maxpos=0
       for location in range(1,fillslot+1):
           if nlist[location]>nlist[maxpos]:
               maxpos = location

       temp = nlist[fillslot]
```

```
        nlist[fillslot] = nlist[maxpos]
        nlist[maxpos] = temp

nlist = [14,46,43,27,57,41,45,21,70]
selectionSort(nlist)
print(nlist)
```
OUTPUT:

AIM: heap sort
PROGRAM:
```
def heap_data(nums, index, heap_size):
    largest_num = index
    left_index = 2 * index + 1
    right_index = 2 * index + 2
    if left_index < heap_size and nums[left_index] > nums[largest_num]:
        largest_num = left_index

    if right_index < heap_size and nums[right_index] > nums[largest_num]:
        largest_num = right_index
    if largest_num != index:
        nums[largest_num], nums[index] = nums[index], nums[largest_num]
        heap_data(nums, largest_num, heap_size)
def heap_sort(nums):
    n = len(nums)
    for i in range(n // 2 - 1, -1, -1):
        heap_data(nums, i, n)
    for i in range(n - 1, 0, -1):
        nums[0], nums[i] = nums[i], nums[0]
        heap_data(nums, 0, i)
    return nums
user_input = input("Input numbers separated by a comma:\n").strip()
nums = [int(item) for item in user_input.split(',')]
heap_sort(nums)
print(nums)
```
OUTPUT:

EXPERIMENT NO :- 03

AIM:
To Create a list of strings, and find the length of each string. And print the list containing length of each string.

SOFTWARE REQUIRED:
Python

THEORY:

Like many other popular programming languages, strings in Python are arrays of bytes representing unicode characters. Python does not have a character data type, a single character is simply a string with a length of 1.Square brackets can be used to access elements of the string.

PROGRAM:

```
from array import *
strs=[“ “ “ “]
for i in range(len(strs)):
   print(strs[i],'=',len(strs[i]),'characters')
```

OUTPUT:

RESULT:

Hence, different strings are created and the length of each string is found

AIM :-

To write a program to convert given number of days into year, weeks and days.

SOFTWARE REQUIRED :-

Python

THEORY :-

A non-leap year contains 365 days (or) 52 weeks. A leap year contains 366 days (or) 52 weeks.

No.of years = (given no.of days)/365

No.of weeks = (given no.of days % 365)/7

No.of days = (given no.of days % 365)%7

PROGRAM :-

```
number_of_days = int(input("Enter number of days: "))

year = int(number_of_days / 365)

week = int((number_of_days % 365) /7)

days = (number_of_days % 365) % 7

    print("years = ",year,

        "\nweeks = ",week,

        "\ndays = ",days)
```

OUTPUT :-

RESULT :-

Hence the given number of days are converted into number of years, weeks and days.

AIM :-

To write a program to demonstrate oct() casting function.

SOFTWARE REQUIRED :-

Python

THEORY :-

The octal number system, oct for short, is the base-8 number system. It uses the digits 0 to 7. To convert a given decimal number into octal number, we divide the number by 8 and write the remainders in the reverse order.

PROGRAM :-

```
int_to_oct=oct(int(input(‘Enter a number=’)))
print(f"After integer to hex casting the result is{int_to_oct}")
```

OUTPUT :-

RESULT :-

Hence the given number is converted into octal number by using oct() casting function.

AIM :-

To check whether the given number is positive or negative.

SOFTWARE REQUIRED :-

Python

THEORY :-

A number which is greater than zero is called a positive number. A number which is less than zero is called a negative number.

PROGRAM :-

```
num=int(input('Enter a number='))
if num>0:
   print("positive number")
elif num==0:
   print("zero")
else:
      print("negative number")
```

OUTPUT :-

RESULT :-

Hence the given number is classified as positive or negative or zero.

AIM :-

To find the factorial of a given number.

SOFTWARE REQUIRED :-

Python

THEORY :-

The factorial of a given number ‘n’ can be calculated by using the following formula

n! = n(n-1)(n-2)(n-3)…

Factorial of a negative number does not exist. Factorial of the number ‘0’ is ‘1’.

PROGRAM :-

```
num = int(input("Enter a number: "))
factorial = 1
if num < 0:
  print(" Factorial does not exist for negative numbers")
elif num == 0:
  print("The factorial of 0 is 1")
else:
  for i in range(1,num + 1):
     factorial = factorial*i
  print("The factorial of",num,"is",factorial)
```

OUTPUT :-

RESULT :-

Hence the factorial of a given number is calculated.

AIM :-

To find a fibonacci series for the given number of terms.

SOFTWARE REQUIRED :-

Python

THEORY :-

The Fibonacci series is a series in which each number is the sum of the two preceding ones. The series commonly starts from 0 and 1. The common fibonacci series is 0,1,1,2,3,5,8,13,21,34,55,…

PROGRAM :-

```
def fibonacci(n):
    a, b = 0,1
    while a < n:
        print(a ,end=" ")
        a, b = b, a+b
def main():
    num=eval(input("Enter the no of terms to be displayed in series:"))
    fibonacci(num)
main()
```

OUTPUT :-

RESULT :-

Hence the Fibonacci series is formed for the given number of terms.

AIM :-

To write a program to display the prime numbers within a given range.

SOFTWARE REQUIRED :-

Python

THEORY :-

A prime number is a number whose only factors are 1 and itself. The smallest prime number is ‘2’. The number ‘1’ is neither prime nor composite number. The few prime numbers are 2,3,5,7,11,13,17,…

PROGRAM :-

```
min = int(input("Enter the min : "))

max = int(input("Enter the max : "))

for n in range(min,max + 1):

   if n > 1:

       for i in range(2,n):

           if (n % i) == 0:

               break

       else:

           print(n)
```

OUTPUT :-

RESULT :-Hence the prime numbers within a given range are displayed by using python program.

AIM :-

To find whether a given number is EVEN or ODD

SOFTWARE USED :-

Python

THEORY :-

When an even number is divided by 2 we get reminder 0,

When an odd number is divided by 2 we get reminder 1.

PROGRAM :-

```
list=[]

even=[]

odd=[]

n=int(input("Enter numbers: "))

for i in range(n):

    list.append(int(input()))
```

```
print('the numbers are: ',list)
for i in range(n):
    if list[i]%2==0:
        even.append(int(list[i]))
for i in range(n):
    if list[i]%2==1:
        odd.append(int(list[i]))
print('Even Numbers are: ',even)
print('odd numbers are: ',odd)
```

OUTPUT :-

RESULT :-

Hence two separate lists of Even and Odd numbers are formed from a given list of natural numbers

EXPERIMENT NO :- 04

(Need to be updated)

EXPERIMENT NO-5

AIM:
To find the Mean, Variance, and Standard Deviation of a List of Numbers.

SOFTWARE REQUIRED:
Python

THEORY:
The mean, variance and standard deviation of a list of numbers can be calculated by using the following formulae
MEAN(μ) = (∑ 〚X)〛)/N
VARIANCE = $(\sum X^2 / N) - \mu^2$
STANDARD DEVIATION = $[\text{VARIANCE}]^{1/2}$

PROGRAM:

```
import math
list=[]
n=int(input("max no.of integers: "))
for i in range(n):
    list.append(int(input()))
print('the numbers are: ',list)
mean=sum(list)/len(list)
print('the mean of list is : ',mean)
var=sum((i-1)**2 for i in list)/len(list)
print('the variance is: ',var)
print('the standard deviation is: ',math.sqrt(var))
```

OUTPUT:

RESULT:

Hence, mean, variance and standard deviation of given list of numbers are calculated.

EXPERIMENT NO-6

AIM :-

To perform auto correlation on a given signal

SOFTWARE USED :- Python

THEORY :- it is defined as the measure of similarity between a signal and its delayed version

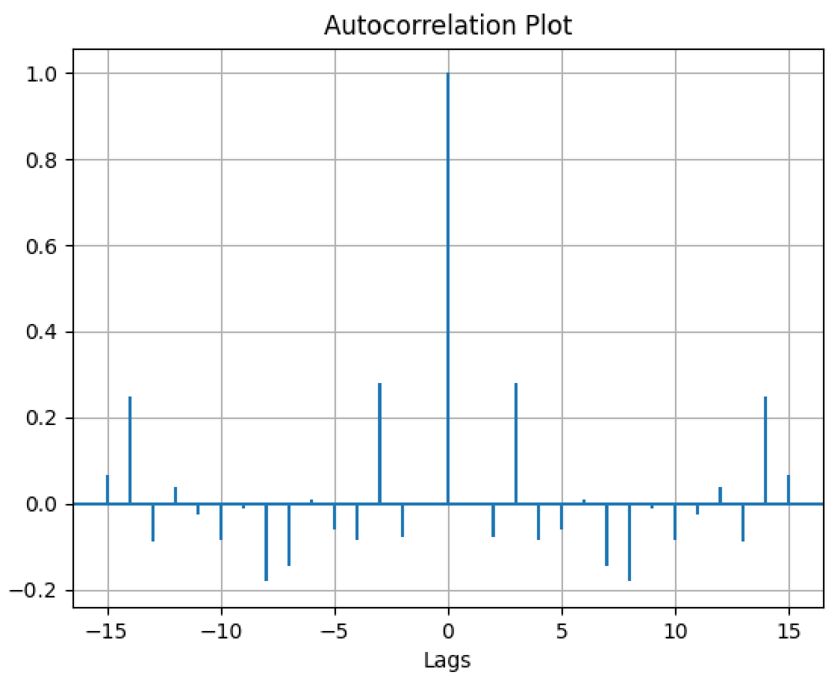
the auto-correlation of a signal x(n) with x(n) is defined as

$$r_{xx}(l) = \sum_{n=-\infty}^{\infty} x(n)x(n - l)$$

PROGRAM :-

```
#Autocorrelation

import matplotlib.pyplot as plt

import numpy as np

np.random.seed(40)

data = np.random.randn(30)

plt.title("Autocorrelation Plot")

plt.xlabel("Lags")

plt.acorr(data, maxlags=15)

print("The Autocorrelation plot for the data is:")

plt.grid(True)

plt.show()
```

OUTPUT :-



RESULT :-

Hence autocorrelation of given signal is done by using python software

AIM :-

To perform cross correlation on two given signals

SOFTWARE USED :-

Python

THEORY :-

Cross correlation tells the similarity of two signals

Cross correlation of two signals $x_1(n)$ and $x_2(n)$

$$R_{x_1x_2}(k) = \sum_{n=-\infty}^{\infty} x_1(n)x_2(n-k)$$

PROGRAM :-

import numpy as np

sig1 = np.sin(np.r_[-1:1:0.1])

```
sig2 = np.sin(np.r_[-1:0:0.1] + np.pi/4)

corr = (len(sig1) - len(sig2) + 1) * [0]

for l in range(len(corr)):

    corr[l] = sum([sig1[i+l] * sig2[i] for i in range(len(sig2))])

print(corr)
```

OUTPUT :-

```
[-0.471998494510103, -0.24686753498102817, -0.019269956645980538,
0.20852016072607304, 0.4342268135797527, 0.6555948156484444,
0.8704123310300105, 1.076532974119988, 1.271897255587048, 1.4545531601096169,
1.62267565026772]
```

RESULT :-

Hence cross correlation of given signal is done by using python software

EXPERIMENT NO-7

AIM :-

Write a program to create, display, append, insert, and reverse the order of items in the array.

SOFTWARE REQUIRED :-

Python

THEORY :-

An array is a collection of similar types of data. An array can hold many values under a single name, and you can access the values by referring to an index number. Adding a new element to the end of the array is called appending and adding a specific element at a specified index of the array is called inserting.

PROGRAM :-

```
# Create array

array = [ ,  ,  ,  ,  ]

# display array

print(f'Created Array : {array}')

# Append  ,  to array

array.append( )

array.append( )

print(f'Array after appending is {array}')
```

```
# Insert  ,  at  ,  places in array
array.insert( ,  )
array.insert( ,  )
print(f'Array after inserting is {array}')
# reverse the array
array.reverse()
print(f'Array after reversing is {array}')
```

OUTPUT :-

RESULT :-

Hence an array is created, displayed, appended, inserted and the items in the array are reversed.

AIM :-

To perform addition of two matrices by using numpy.

SOFTWARE REQUIRED :-

Python

THEORY :-

A matrix is a set of numbers arranged in rows and columns so as to form a rectangular array. Addition of matrices is possible only when the matrices are of same order. To add two matrices, we create two input arrays of same length.

PROGRAM:-

```
import numpy as np
in_arr1 = np.array([[   ], [    ]])
in_arr2 = np.array([[     ], [     ]])
print("1st Input array : ", in_arr1)
print("2nd Input array : ", in_arr2)
out_arr = np.add(in_arr1, in_arr2)
print("output added array : ", out_arr)
```

OUTPUT :-

RESULT :-

Hence the two matrices are added by using numpy.

AIM :-

To perform multiplication of matrices by using numpy.

SOFTWARE REQUIRED:-

Python

THEORY :-

Multiplication of matrices is only possible when the matrices are of same order. To multiply two matrices, we create two input arrays of same length. To multiply the matrices we use dot() function. The numpy module of python provides a function to perform the dot product of two arrays.

PROGRAM :-

```
import numpy as n

A = [[   ], [   ]]

B = [[   ],[    ]]

Print(“A*B is”)

print(np.dot(A, B))

print(“B*A is”)

print(np.dot(B, A))
```

OUTPUT :-

RESULT :-

Hence the two matrices are multiplied by using numpy.

AIM :-

To perform transpose of a matrix by using numpy.

SOFTWARE REQUIRED :-

Python

THEORY :-

The transpose of a matrix is obtained by interchanging its rows into columns and columns into rows. To transpose a matrix, we create an input array. To transpose the given matrix in array ,we use transpose() function in python. This function reverses the dimension of the given array and returns the modified array.

PROGRAM :-

```
import numpy as np
A = np.array([[    ], [     ], [     ]])
A_T = A.transpose()
Print("the transposed matrix is")
print(A_T)
```

OUTPUT :-

RESULT :-

Hence the transpose of a matrix is obtained by using numpy.

EXPERIMENT NO-8

Select your project from here

EXPERIMENT NO-9

Matplotlib is easy to use and an amazing visualizing library in Python. It is built on NumPy arrays and designed to work with the broader SciPy stack and consists of several plots like line, bar, scatter, histogram, etc.

Pyplot is a Matplotlib module that provides a MATLAB-like interface. Pyplot provides functions that interact with the figure i.e. creates a figure, decorates the plot with labels, and creates a plotting area in a figure.

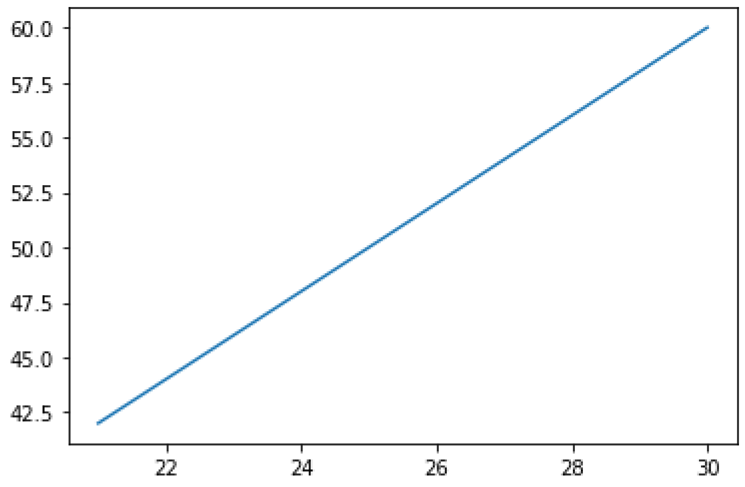
Syntax:

```
matplotlib.pyplot.plot(*args, scalex=True, scaley=True, data=None, **kwargs)
```

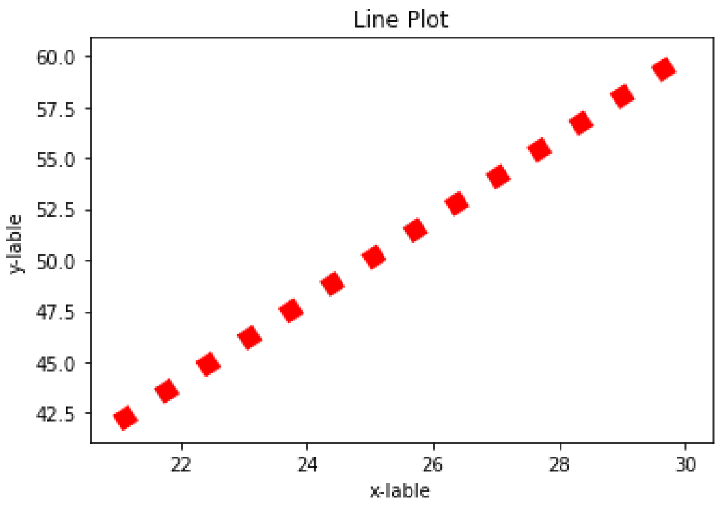
AIM: Simple Line plots
PROGRAM:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
x=np.arange(21,31)
y=x*2
plt.plot(x,y)
```

OUTPUT:



AIM: Adjusting the Plot: Line Colors and Styles, Axes Limits, Labeling Plots,
PROGRAM:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
x=np.arange(21,31)
y=x*2
plt.plot(x,y,linewidth=10,color="red",linestyle=":") #- , -- , -. , :
plt.title("Line Plot")
plt.xlabel("x-label")
plt.ylabel("y-label")
plt.show()
```
OUTPUT:

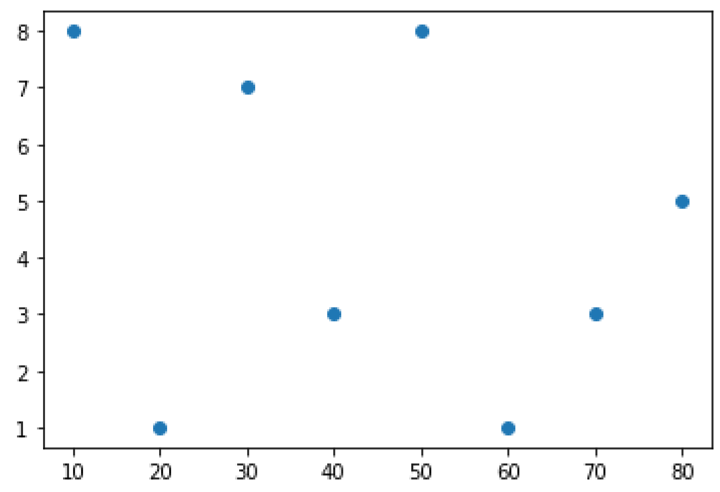


AIM: Simple Scatter Plots

Theory: Scatter plots are useful for showing the relationship between two variables. Any correlation between variables or outliers in the data can be easily spotted using scatter plots.
PROGRAM:

```
x=[10,20,30,40,50,60,70,80]
y=[8,1,7,3,8,1,3,5]
plt.scatter(x,y,marker="o") #s,^,*,o
plt.show()
```

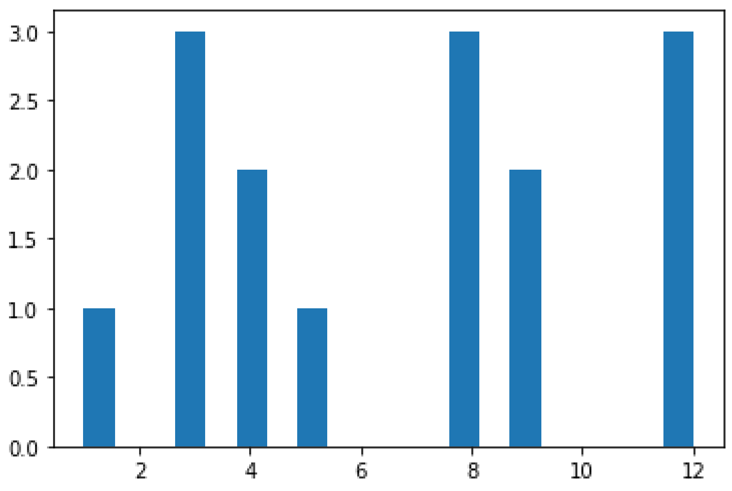
OUTPUT:



AIM: Histograms
THEORY: A histogram shows the distribution of numeric data through a continuous interval by segmenting data into different bins. Useful for inspecting skewness in the data.
PROGRAM:

```
Age=[1,3,3,3,9,9,5,4,4,8,8,8,12,12,12]
plt.hist(Age,bins=20)
plt.show()
```

OUTPUT:



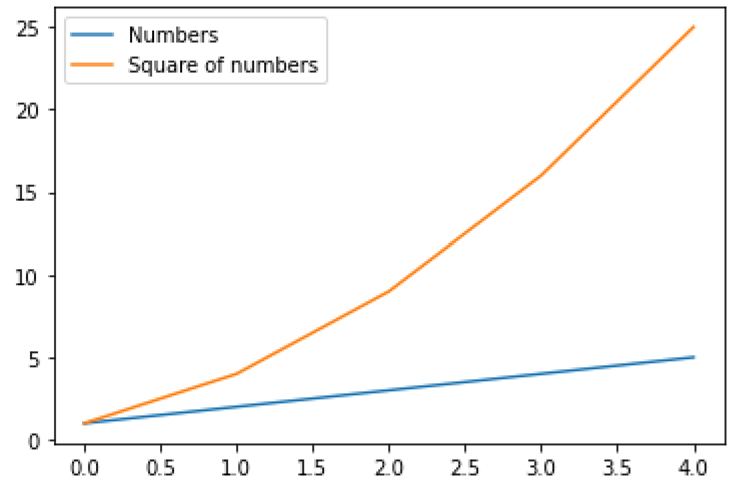
AIM: Customizing Plot Legends

THEORY: A legend is an area describing the elements of the graph. In the matplotlib library, there’s a function called legend() which is used to Place a legend on the axes.

PROGRAM:

```
import numpy as np
import matplotlib.pyplot as plt
x = np.arange(5)
y1 = [1, 2, 3, 4, 5]
y2 = [1, 4, 9, 16, 25]
plt.plot(x, y1, label ='Numbers')
plt.plot(x, y2, label ='Square of numbers')
plt.legend()
plt.show()
```

OUTPUT:

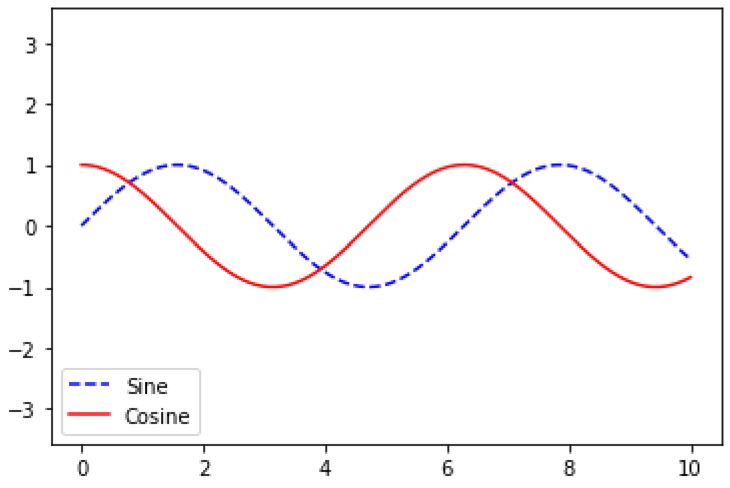


AIM: Choosing Elements for the Legend

PROGRAM:

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

x = np.linspace(0, 10, 1000)
fig, ax = plt.subplots()

ax.plot(x, np.sin(x), '--b', label ='Sine')
ax.plot(x, np.cos(x), c ='r', label ='Cosine')
ax.axis('equal')

leg = ax.legend(loc ="lower left");
```

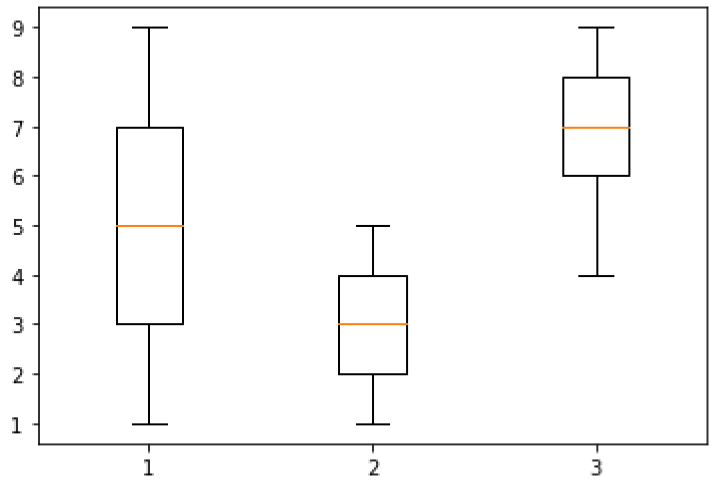
OUTPUT:



AIM: Boxplot
THEORY: Box plot gives statistical information about the distribution of numeric data divided into different groups. It is useful for detecting outliers within each group
PROGRAM:

```
x=[1,2,3,4,5,6,7,8,9]
y=[1,2,3,4,5,4,3,2,1]
z=[6,7,8,9,8,7,6,5,4]
data=list([x,y,z])
plt.boxplot(data)
plt.show()
```

OUTPUT:

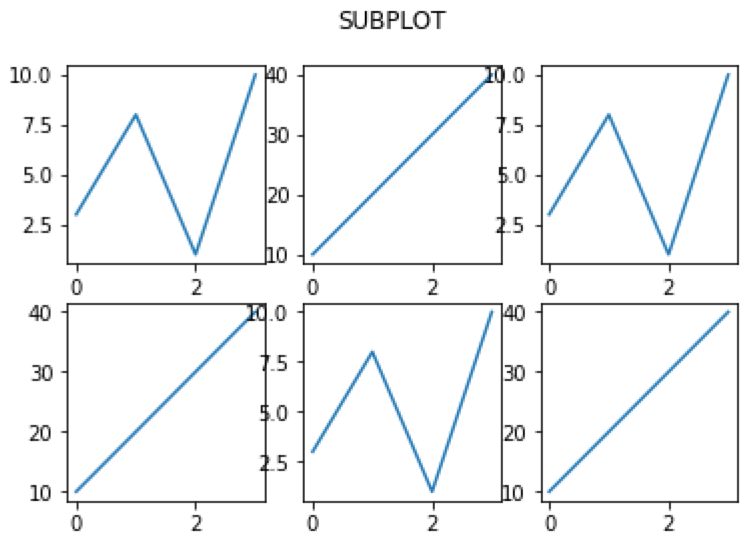


AIM: Multiple sub-plots
PROGRAM:

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

```
plt.suptitle("SUBPLOT")
x = np.array([0, 1, 2, 3])
y = np.array([3, 8, 1, 10])
plt.subplot(2, 3, 1)
plt.plot(x,y)

x = np.array([0, 1, 2, 3])
y = np.array([10, 20, 30, 40])
plt.subplot(2, 3, 2)
plt.plot(x,y)

x = np.array([0, 1, 2, 3])
y = np.array([3, 8, 1, 10])
plt.subplot(2, 3, 3)
plt.plot(x,y)

x = np.array([0, 1, 2, 3])
y = np.array([10, 20, 30, 40])
plt.subplot(2, 3, 4)
plt.plot(x,y)

x = np.array([0, 1, 2, 3])
y = np.array([3, 8, 1, 10])
plt.subplot(2, 3, 5)
plt.plot(x,y)

x = np.array([0, 1, 2, 3])
y = np.array([10, 20, 30, 40])
plt.subplot(2, 3, 6)
plt.plot(x,y)
plt.show()
```

OUTPUT:



AIM:
PROGRAM:
OUTPUT:

EXPERIMENT NO-10

(Need to be updated)

EXPERIMENT NO-11

AIM: Python Program for Compressing data via dimensionality reduction: PCA
THEORY: PCA is a dimensionality - reduction method that is often used to reduce the dimensionality
Of large data sets, by transforming a large set of Variables into a smaller one that still contains most of the information in the large set.
PROGRAM:

```
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np
import seaborn as sns
%matplotlib inline
from sklearn.datasets import load_breast_cancer
cancer = load_breast_cancer()
cancer.keys
dict_keys= (['DESCR', 'data', 'feature_names','target_names','target'])
print(cancer['DESCR'])

df = pd.DataFrame(cancer['data'],columns=cancer['feature_names'])
print(df)
```

OUTPUT:

```
.. _breast_cancer_dataset:

Breast cancer wisconsin (diagnostic) dataset
--------------------------------------------

**Data Set Characteristics:**

    :Number of Instances: 569

    :Number of Attributes: 30 numeric, predictive attributes and the class
    :Summary Statistics:

    ===================================== ====== ======
                                           Min    Max
    ===================================== ====== ======
    radius (mean):                        6.981  28.11
    texture (mean):                       9.71   39.28
    perimeter (mean):                     43.79  188.5
    area (mean):                          143.5  2501.0
    smoothness (mean):                    0.053  0.163
    compactness (mean):                   0.019  0.345
    concavity (mean):                     0.0    0.427
………………

     mean radius  mean texture  mean perimeter  mean area  mean smoothness  \
0          17.99         10.38          122.80     1001.0          0.11840
1          20.57         17.77          132.90     1326.0          0.08474
..           ...           ...             ...        ...              ….
567        20.60         29.33          140.10     1265.0          0.11780
568         7.76         24.54           47.92      181.0          0.05263

     mean compactness  mean concavity  mean concave points  mean symmetry  \
0             0.27760         0.30010              0.14710         0.2419
1             0.07864         0.08690              0.07017         0.1812
 ..               ...             ...                  ...            ...
567           0.27700         0.35140              0.15200         0.2397
568           0.04362         0.00000              0.00000         0.1587

     mean fractal dimension  ...  worst radius  worst texture  \
0                   0.07871  ...        25.380          17.33
1                   0.05667  ...        24.990          23.41
..                      ...  ...           ...            ...
567                 0.07016  ...        25.740          39.42
568                 0.05884  ...         9.456          30.37
```

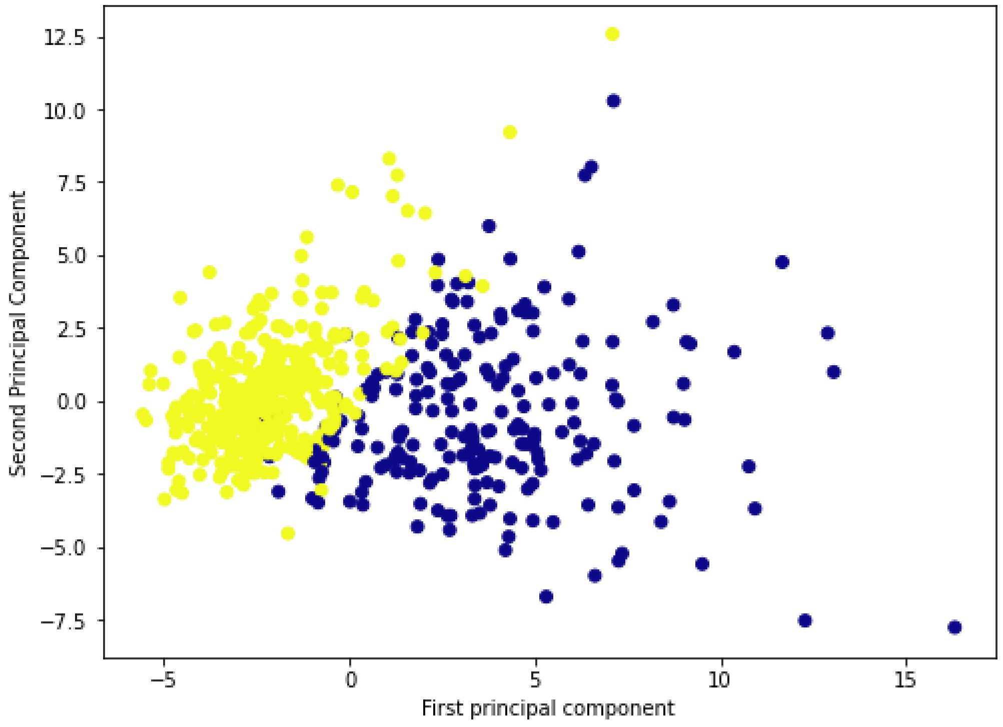
```
   worst perimeter  worst area  worst smoothness  worst compactness  \
0           184.60     2019.0           0.16220            0.66560
1           158.80     1956.0           0.12380            0.18660
 ..            ...        ...               …                ...
567         184.60     1821.0           0.16500            0.86810
568          59.16      268.6           0.08996            0.06444

   worst concavity  worst concave points  worst symmetry  \
0           0.7119                0.2654          0.4601
1           0.2416                0.1860          0.2750
..             ...                   ...             ...
567         0.9387                0.2650          0.4087
568         0.0000                0.0000          0.2871

    worst fractal dimension
0                   0.11890
1                   0.08902
..                      ...
567                 0.12400
568                 0.07039

[569 rows x 30 columns]
```

AIM: PCA visualization
THEORY:
It is difficult to visualize high dimensional data, we can use PCA to find the first two principal components, and visualize the data in this new, two-dimensional space, with a single scatter-plot. Before we do this though, we'll need to scale our data so that each feature has single unit variance.
PROGRAM:

```
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
scaler.fit(df)
StandardScaler(copy = True, with_mean= True, with_std= True)
scaled_data = scaler.transform(df)

from sklearn.decomposition import PCA
pca = PCA(n_components=2)
pca.fit(scaled_data)
PCA(copy=True, n_components=2, whiten=False)
x_pca = pca.transform(scaled_data)
Scaled_data.shape #(569, 30)
X_pca.shape #(569, 2)
plt.figure(figsize=(8,6)) plt.scatter(x_pca[:,0],x_pca[:,1],c=cancer['target'],cmap='plasma')
plt.xlabel('First principal component') plt.ylabel('Second Principal Component')
```

OUTPUT:

```
Text(0, 0.5, 'Second Principal Component')
```



EXPERIMENT NO-12

(Need to be updated)

EXPERIMENT NO-13

AIM: Python Programs for Classification

PROGRAM:

(Exclude the first flower diagram)

https://github.com/codebasics/py/blob/master/ML/17_knn_classification/knn_classification_tutorial.ipynb

OUTPUT:

EXPERIMENT NO-14

AIM: Python Programs for Model Evaluation: K-fold cross validation.

PROGRAM:

https://github.com/codebasics/py/blob/master/ML/12_KFold_Cross_Validation/12_k_fold.ipynb

OUTPUT:

–WRITE TILL HERE ONLY DON'T WRITE THE REMAINING PART–

AIM :-

To generate a sinusoidal signal

SOFTWARE USED :-

Python

THEORY :-

A sinusoidal wave also known as sine wave is a trigonometric function defines as

$X(t)=A\sin(\Omega t+ \varphi)$

Where

A=amplitude

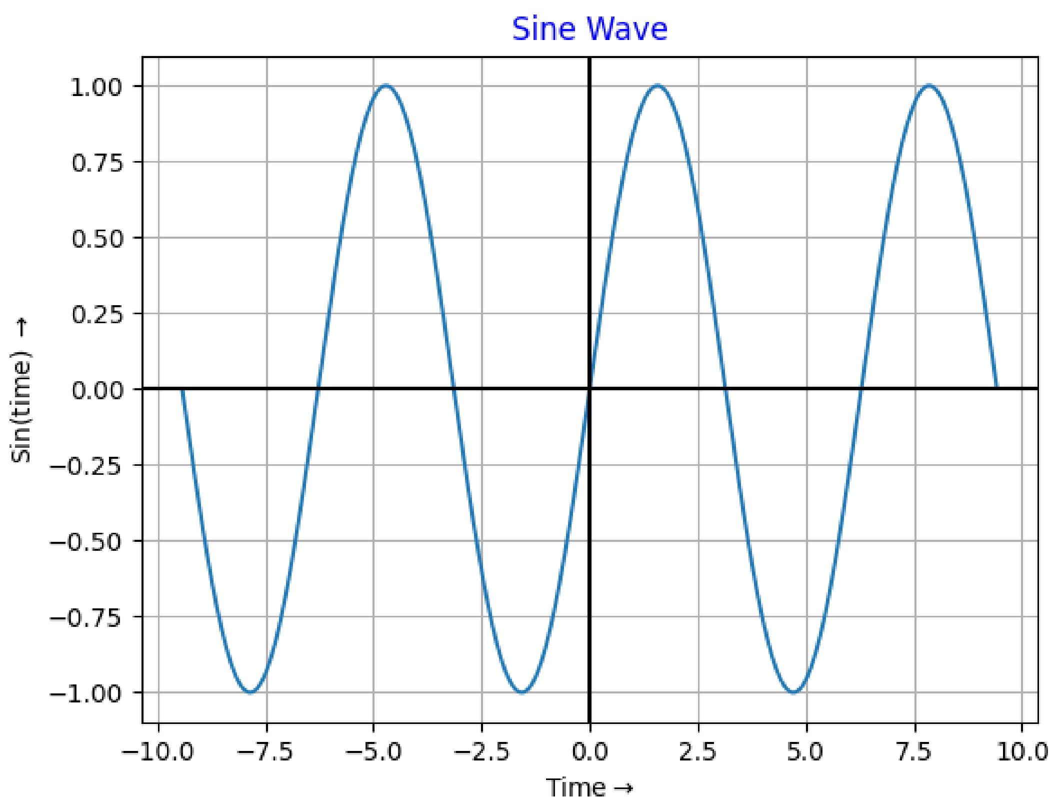
Ω=angular frequency

Φ=phase difference

PROGRAM :-

```
import numpy as np
import matplotlib.pyplot as plt
time = np.arange(-3*np.pi, 3*np.pi, 0.01)
amplitude = np.sin(time)
plt.plot(time, amplitude)
plt.title('Sine Wave', color='b')
plt.xlabel('Time'+ r'$\rightarrow$')
plt.ylabel('Sin(time) '+ r'$\rightarrow$')
plt.grid()
plt.axhline(y=0, color='k')
plt.axvline(x=0, color='k')
plt.show()
```

OUTPUT :-



RESULT :-

Hence a sine wave is generated using python software.

AIM :- To display digital sinusoidal signal

SOFTWARE USED :- Python

THEORY :-

A digital sinusoidal wave also known as discrete sine wave is a trigonometric function defines as

X(n)=Asin(Ω n+ φ) Where

A=amplitude ; Ω=angular frequency ; Φ=phase difference

PROGRAM :-

import numpy as np

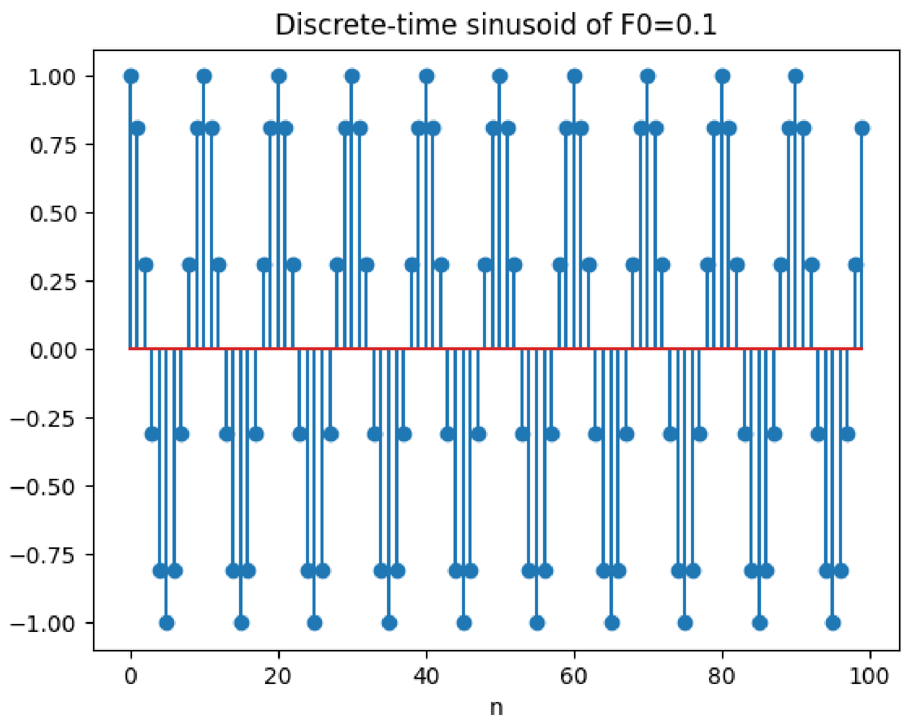
import matplotlib.pyplot as plt

F0 = 0.1

L = 100

```
n = np.arange(L)
x = np.cos(2*np.pi*F0*n)
plt.stem(x)
plt.title('Discrete-time sinusoid of F0=0.1')
plt.xlabel('n')
plt.show()
```

OUTPUT :-



RESULT :-

Hence a discrete sine wave is generated using python software

AIM :-

To write a program to display exponential signal by using numpy in python.

SOFTWARE REQUIRED :-

Python

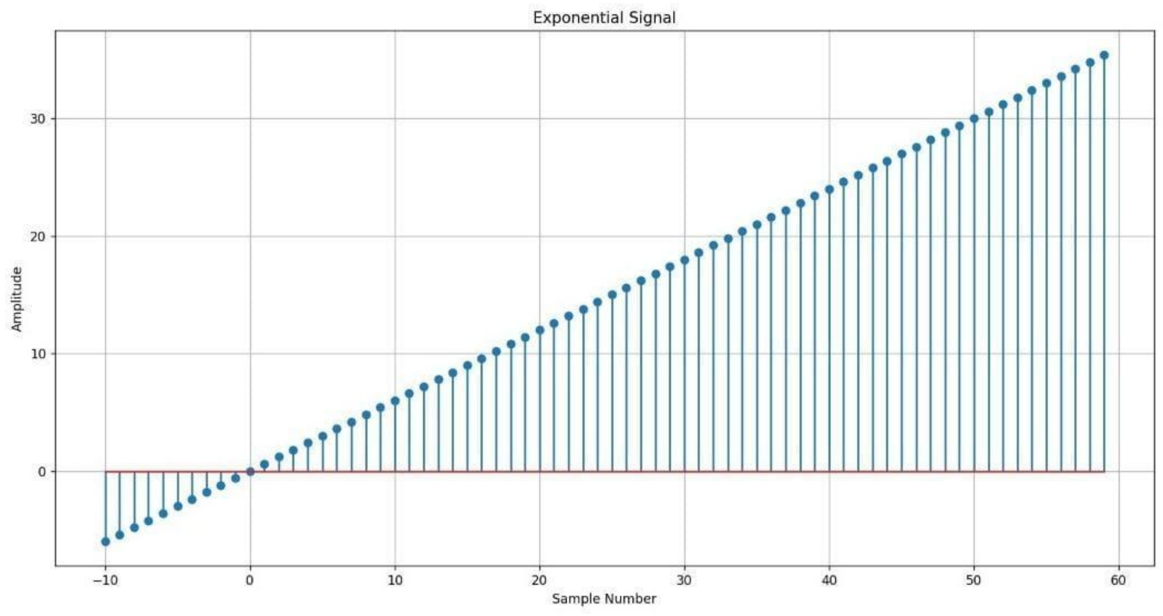
THEORY :-

An exponential signal or exponential function is a function that literally represents an exponentially increasing or decreasing series. To generate an exponential signal in python, we use exp() function. This function helps user to calculate exponential of all the elements in the input array.

PROGRAM :-

```
import matplotlib.pyplot as plt

import numpy as np

from math import pi

plt.close('all')

#Generate Digital Sinusoidal Signal

n=np.arange(-100, 100)

x=np.sin(0.1*n)

plt.stem(n, x); plt.title('Digital Sinusoidal Signal')

plt.xlabel('Sample number');  plt.ylabel('Amplitude')

plt.grid(True)

plt.show()

#Generate Exponential Signal

a = 0.9

n = np.arange(-10, 50)

z = a*n

plt.figure(3)

plt.stem(n, z); plt.title('Exponential Signal')

plt.xlabel('Sample Number'); plt.ylabel('Amplitude')

plt.grid(True)

plt.show()
```

OUTPUT :-



RESULT :-

Hence the exponential signal is displayed by using numpy in python.