EX.NO:01(A)	PYTHON PROGRAMMING USING SIMPLE STATEMENTS ANDEXPRESSIONS FOR EXCHANGING THE VALUE OF TWO VARIABLES
DATE:	

AIM:

To write a python program using simple statements and expressions for exchanging the values of two variables.

ALGORITHM:

Step 1: start

Step 2: enter x, y

Step 3: print x, y

Step 4: x = x + y

Step 5: y= x - y

Step 6: x =x - y

Step 7: print x, y

Step 8: end

PROGRAM:

```
x = int(input("Enter the value of x: "))
y = int(input("Enter the value of y: "))
print("before swapping numbers: %d %d\n" %(x,y))
#swapping#
x = x +y
y = x - y
x = x - y
print("After swapping: %d  %d\n"%(x,y))
```

OUTPUT:

```
Enter the value of x: 5
Enter the value of y: 6
before swapping numbers: 5  6
After swapping: 6  5
```

RESULT

The python program using simple statements and expressions for exchanging the values of two variables has been executed successfully.

EX.NO:01(B)	PYTHON PROGRAMMING USING SIMPLE STATEMENTS ANDEXPRESSIONS FOR CIRCULATING THE VALUES OF N VARIABLES
DATE:	

AIM:

To write a python program using simple statements and expressions for circulating the values of n variables.

ALGORITHM:

Step1: Start

Step2: Define a List named a[]
Step3: Get the input ‘n’ value to rotate the values.
Step4: Perform the following step till b=a[n:]+a[:n] ,using slice operator
Step5:b Print the rotated values
Step6: Stop

PROGRAM:

```
a=[1,2,3,4,5,6,7,8,9,10]
n=int(input())
b=a[n:]+a[:n]
print(b)
```

OUTPUT:

```
Enter the number:6
[7, 8, 9, 10, 1, 2, 3, 4, 5, 6]
```

RESULT

The python program using simple statements and expressions for circulating the values of n variables has been executed successfully.

EX.NO:01(C) DATE:	PYTHON PROGRAMMING USING SIMPLE STATEMENTS ANDEXPRESSIONS FOR DISTANCE BETWEEN TWO POINTS

AIM:

To write a python program using simple statements and expressions for distance between two points.

ALGORITHM:

Step1: Start
Step2: import the module ‘math’

Step3: Calculate the difference of the ‘x’ coordinates as ‘a’.

Step4: Calculate the difference of the ‘y’ coordinates as ‘b’

Step5: Calculate the sum of ‘x’ and ‘y’ coordinates; c=(a**2)+(b**2)

Step6: Calculate the square root of ‘c’

Step7: Print the distance between two points.

Step8: Stop.

PROGRAM:

```
import math
x1=int(input("x coordinate of first point:"))
y1=int(input("y coordinate of first point:"))
x2=int(input("x coordinate of second point:"))
y2=int(input("y coordinate of second point:"))
distance=math.sqrt(((x1-x2)**2)+((y1-y2)**2))
print(distance)
```

OUTPUT:

```
x coordinate of first point:7
y coordinate of first point:9
x coordinate of second point:8
y coordinate of second point:5
4.123105625617661
```

RESULT

The python program using simple statements and expressions for distance between two points has been executed successfully.

EX.NO: 02(A)	SCIENTIFIC PROBLEMS USING CONDITIONAL STATEMENT
DATE:	

AIM:

To write a python program for scientific problems using conditional statement.

ALGORITHM:

Step1: Start
Step2: Read the value of n.
Step3: If n>1
If n%2==0

Print the value of n.”Even number”

Else

Print the value of n,”Odd number”

Step4: Else
Step5: Print “invalid number”
Step6: Stop

PROGRAM:

```
n=int (input (“Enter the numbers :”)
if n>1:
   if(n%2)==0:
        print(n,”Even number”)
   else:
        print(n,”Odd number”)
else:
   print (“)nvalid number”)
```

OUTPUT:

```
Enter the number: 16
16 Even number

Enter the number: 13
13 Odd number
```

RESULT

The python program for scientific problems using conditional statement has been executed successfully.

EX.NO: 02(B)	SCIENTIFIC PROBLEMS USING ITERATIVE LOOPS
DATE:	

AIM:

To write a python program for scientific problems using iterative loops.

ALGORITHM:

Step1: Start

Step2: Initialize the value of fact=1

Step3: Read the value of n

Step4: For i in range of n+1

Step5: Print (fact , " is factorial')

Step6: End of the program.

PROGRAM:

```
n =int(input("Enter the number :"))
fact=1
for i in range (1,n+1):
        fact=fact+i
print("Factorial is ', fact)
```

OUTPUT:

```
Enter the number:5
Factorial is 120
```

RESULT

The python program for scientific problems using iterative loops has been executed successfully.

EX.NO: 03(A)	LINEAR SEARCH
DATE:	

AIM:

To write a python program to search an element in an array using Linear search technique.

ALGORITHM:

Step 1: Start

Step 2: Get list of elements from the user

Step 3: Get the number to be searched

Step 4: Set i to 1

Step 5: if i>n then go to step 7

Step 6: if A[i]=x then go to step 6

Step 7: Set I to i+1

Step 8: Go to Step 2

Step 9: Print Element x found at index I and step 8

Step 10: Print element not found

Step 11: Stop

PROGRAM:

```
def search(arr, n, x):
 for i in range(0, n):
        if (arr[i] == x):
            return i
 return -1
arr = []
a = int(input("Enter number of elements : "))for
i in range(0, a):
 ele = int(input())

 arr.append(ele)
print(arr)
x=int(input("enter the value(x):"))n
= len(arr)
result = search(arr, n, x)
if(result == -1):
 print("Element is not present in array")
else:
 print("Element is present at index", result)
```

OUTPUT:

```
Enter number of elements : 45

8

7

36

[5, 8, 7, 36]

enter the value(x):8

Element is present at index 1
```

RESULT

The python program to search an element in an array using linear search technique has been executed successfully.

EX.NO: 03(B)	BINARY SEARCH
DATE:	

AIM:

To write a python program to search an element in an array using binary search technique.

ALGORITHM:

Step1: Start

Step2: Compare x with the middle element.

Step3: If x matches with the middle element, we return the mid index.

Step4: Else if x is greater than the mid element, then x can only lie in the right (greater) half subarray after the mid element. Then we apply the algorithm again for the right half.

Step5: Else if x is smaller, the target x must lie in the left (lower) half. So we apply the algorithm for the left half.

Step6: End

PROGRAM:

```
def binarySearch(arr, l, r, x):
    if r >= l:
        mid = l + (r - l) // 2
        if arr[mid] == x:
            return  mid
        elif arr[mid] > x:
            return binarySearch(arr, l, mid-1, x)
        else:
            return binarySearch(arr, mid + 1, r, x)
    else:
        return -1
arr = []
a = int(input("Enter number of elements : "))
for i in range(0, a):
    ele = int(input())
    arr.append(ele)
print(arr)
x=int(input("enter the element value:"))
result = binarySearch(arr, 0, len(arr)-1, x)
if result != -1:
   print("Element is present at index % d" % result)
else:
   print("Element is not present in array")
```

OUTPUT:

```
Enter number of elements : 4
5
6
12
3
[5, 6, 12, 3]
enter the element value:12
Element is present at index  2
```

RESULT

The python program to search an element in an array using binary search technique has been executed successfully.

EX.NO: 04(A)	SELECTION SORT
DATE:	

AIM:

To write a Python program to sort the elements using selection sort.

ALGORITHM:

Step 1: Start

Step 2: Get the elements from the user

Step 3: Set MIN to location 0

Step 4: Search the minimum element in the list.

Step 5: Swap with value at location MIN

Step 6: Increment MIN to point to next element

Step 7: Repeat until list is sorted.

Step 8: Stop

PROGRAM:

```
def selectionSort(a): for i in
   range(len(a)):
      least=i
      for k in range(i+1,len(a)):if
         a[k]<a[least]:
            least=k
      temp=a[least]
      a[least]=a[i]
      a[i]=temp
a=[50,30,10,20,40,70,60]
print ("Original list",a)
selectionSort(a)       #calling to the function
print("Selection Sort:",a)
```

OUTPUT:

Original list [50, 30, 10, 20, 40, 70, 60]

Selection Sort: [10, 20, 30, 40, 50, 60, 70]

RESULT

The Python programs to sort elements using selection sort method was created and executed successfully.

EX.NO: 04(B)	INSERTION SORT
DATE:	

AIM:

To write a Python program to sort the elements using insertion sort.

ALGORITHM:

Step 1: Start

Step 2: Get the elements from the user

Step 3a: The second element in the list is compared with the elements that appear before it(only first element in this case).

Step 3b: If the second element is smaller than first element, second element isinserted in theposition of first element. After first step, first two elements of an array will be sorted.

Step 3c: Pick next element

Step 4: Repeat step 3 until all the elements in the list is sortedStep

5: Stop

PROGRAM:

```
def insertion(a):
    for i in range(1,len(a)):
        currentvalue=a[i]
        position=i
        while position>0 and a[position-1]>currentvalue:
            a[position]=a[position-1]
            position=position-1
        a[position]=currentvalue
a=[]       #define empty list
n=int(input("Enter the upper limit"))
for i in range (n):
    a.append(int(input()))
insertion(a)
print ("Insertion list",a)
```

OUTPUT:

```
Enter the upper limit710
09
05
04
12
16
18
Insertion list [4, 5, 9, 10, 12, 16, 18]
```

RESULT

The Python program to sort the elements using insertion sort method wascreated andexecuted successfully.

EX.NO: 05(A)	MERGE SORT
DATE:	

AIM:

To write a Python program to sort elements using merge sort method.

ALGORITHM:

Step 1: Start

MERGE-SORT(A,p,r)

Step 2: if r>p

Step 3: Find the middle point to divide the array into two halves:thenq=FLOOR

[(p+r)/2] //Divide step

Step 4: Call merge Sort for first half:Call

MERGE (A,p,q)

Step 5: Call merge Sort for second half:Call

MERGE (A,q+1,r)

Step 6: Merge the two halves sorted in step 2 and 3:Call

MERGE (A,p,q,r) //Conquer step

Step 7: Stop

PROGRAM:

```
def mergeSort(a):
   print("Splitting ",a)
   if len(a)>1:
      mid = len(a)//2
      lefthalf = a[:mid]
      righthalf = a[mid:]
      mergeSort(lefthalf)
      mergeSort(righthalf)
      i=0
      j=0
      k=0
      while i < len(lefthalf) and j <
len(righthalf):
         if lefthalf[i] < righthalf[j]:
            a[k]=lefthalf[i]
            i=i+1
         else:
            a[k]=righthalf[j]
            j=j+1
            k=k+1
      while i < len(lefthalf):
         a[k]=lefthalf[i]
         i=i+1
         k=k+1
      while j < len(righthalf):
```

```
            a[k]=righthalf[j]
            j=j+1
            k=k+1
        print("Merging ",a)
    a = [54,26,93,17,77,31,44,55,20]
    mergeSort(a)
    print(a)
```

OUTPUT:

```
Splitting  [54, 26, 93, 17, 77, 31, 44, 55, 20]
Splitting  [54, 26, 93, 17]
Splitting  [54, 26]
Splitting  [54]
Merging  [54]
Splitting  [26]
Merging  [26]
Merging  [26, 54]
Splitting  [93, 17]
Splitting  [93]
Merging  [93]
Splitting  [17]
Merging  [17]
Merging  [17, 93]
Merging  [17, 93, 93, 17]
Splitting  [77, 31, 44, 55, 20]
Splitting  [77, 31]
Splitting  [77]
```

Merging [77]

Splitting [31]

Merging [31]

Merging [31, 77]

Splitting [44, 55, 20]

Splitting [44]

Merging [44]

Splitting [55, 20]

Splitting [55]

Merging [55]

Splitting [20]

Merging [20]

Merging [20, 55]

Merging [20, 55, 20]

Merging [20, 55, 20, 77, 20]

Merging [20, 55, 20, 77, 20, 93, 93, 17, 20]

[20, 55, 20, 77, 20, 93, 93, 17, 20]

RESULT

The Python program to sort the elements using merge sort method was created and executed successfully.

EX.NO: 05(B)	QUICK SORT
DATE:	

AIM:

To write a Python program to sort elements using quick sort method.

ALGORITHM:

Step1: Start

Step2: Create a function quicksort that takes a list and two variables start and end as arguments.

Step3: If end – start is not greater than 1, return.

Step4: Find the index of the pivot, p by calling the function partition with the list and variables start and end as arguments.

Step5: Call quicksort with the list and the variables start and p as arguments to sort the list from start to p – 1.

Step6: Call quicksort with the list, the expression p + 1 and end as arguments to sort the list from p + 1 to end – 1.

Step7: Define the function partition that takes a list and two variables start and end as arguments.

Step8: The function partition uses Hoare's partition scheme to partition the list.

Step9: End

PROGRAM:

```
def quicksort(alist, start, end):
   '''Sorts the list from indexes start to end - 1 inclusive.'''if
   end - start > 1:
      p = partition(alist, start, end)
      quicksort(alist, start, p)
      quicksort(alist, p + 1, end)


def partition(alist, start, end):
   pivot = alist[start]
   i = start + 1
   j = end - 1

   while True:
      while (i <= j and alist[i] <= pivot):i
         = i + 1
      while (i <= j and alist[j] >= pivot):j
         = j - 1

      if i <= j:
         alist[i], alist[j] = alist[j], alist[i]
      else:
         alist[start], alist[j] = alist[j], alist[start]
         return j


alist = input('Enter the list of numbers: ').split()alist
= [int(x) for x in alist]
quicksort(alist, 0, len(alist))
print('Sorted list: ', end='')
print(alist)
```

OUTPUT:

Enter the list of numbers: 25 12 47 10

Sorted list: [10, 12, 25, 47]

RESULT

The Python program to sort the elements using quick sort method wascreated and executed successfully.

EX.NO: 06(A)	IMPLEMENTING APPLICATION USING LISTS
DATE:	

AIM:

To write a python program to perform various operations of list.

ALGORITHM:

Step 1: Start the program
Step 2: Declare the components of construction using lists
Step 3: Perform the operations of tuples
Step 4: Display the output
Step 5: Stop

PROGRAM:

```
thislist = ['bricks', 'cement', 'brush','sand','Paint']
print(thislist)
print(len(thislist))
print(thislist[1])
print(thislist[-1])
print(thislist[1:3])
print(type(thislist)) if
'cement' in thislist:
   print('Yes, cement is in the list')
   thislist[1] = 'Tiles' print(thislist)
   thislist.insert(2, 'bricks') print(thislist)
   thislist.append('butterfly tiles')
   print(thislist)
   tropical = ['floor', 'marbel', 'granite']
   thislist.extend(tropical) print(thislist)
   thislist.remove('floor') print(thislist)
   thislist.pop(1)
   print(thislist)
i = 0
while i < len(thislist):
   print(thislist[i])
   i = i + 1
   thislist.sort()
   print(thislist)
   thislist.sort(reverse = True)
   print(thislist)
```

OUTPUT:

```
['bricks', 'cement', 'brush', 'sand', 'Paint']5
cement
Paint
['cement', 'brush']
<class 'list'>
Yes, cement is in the list
['bricks', 'Tiles', 'brush', 'sand', 'Paint']
['bricks', 'Tiles', 'bricks', 'brush', 'sand', 'Paint']
['bricks', 'Tiles', 'bricks', 'brush', 'sand', 'Paint', 'butterfly tiles']
['bricks', 'Tiles', 'bricks', 'brush', 'sand', 'Paint', 'butterfly tiles', 'floor', 'marbel', 'granite']
['bricks', 'Tiles', 'bricks', 'brush', 'sand', 'Paint', 'butterfly tiles', 'marbel', 'granite']
['bricks', 'bricks', 'brush', 'sand', 'Paint', 'butterfly tiles', 'marbel', 'granite']bricks
['Paint', 'bricks', 'bricks', 'brush', 'butterfly tiles', 'granite', 'marbel', 'sand']
['sand', 'marbel', 'granite', 'butterfly tiles', 'brush', 'bricks', 'bricks', 'Paint']
marbel
['Paint', 'bricks', 'bricks', 'brush', 'butterfly tiles', 'granite', 'marbel', 'sand']
['sand', 'marbel', 'granite', 'butterfly tiles', 'brush', 'bricks', 'bricks', 'Paint']
granite
['Paint', 'bricks', 'bricks', 'brush', 'butterfly tiles', 'granite', 'marbel', 'sand']
['sand', 'marbel', 'granite', 'butterfly tiles', 'brush', 'bricks', 'bricks', 'Paint']butterfly tiles
['Paint', 'bricks', 'bricks', 'brush', 'butterfly tiles', 'granite', 'marbel', 'sand']
['sand', 'marbel', 'granite', 'butterfly tiles', 'brush', 'bricks', 'bricks', 'Paint']brush
['Paint', 'bricks', 'bricks', 'brush', 'butterfly tiles', 'granite', 'marbel', 'sand']
['sand', 'marbel', 'granite', 'butterfly tiles', 'brush', 'bricks', 'bricks', 'Paint']bricks
['Paint', 'bricks', 'bricks', 'brush', 'butterfly tiles', 'granite', 'marbel', 'sand']
['sand', 'marbel', 'granite', 'butterfly tiles', 'brush', 'bricks', 'bricks', 'Paint']bricks
['Paint', 'bricks', 'bricks', 'brush', 'butterfly tiles', 'granite', 'marbel', 'sand']
['sand', 'marbel', 'granite', 'butterfly tiles', 'brush', 'bricks', 'bricks', 'Paint']
Paint
['Paint', 'bricks', 'bricks', 'brush', 'butterfly tiles', 'granite', 'marbel', 'sand']
['sand', 'marbel', 'granite', 'butterfly tiles', 'brush', 'bricks', 'bricks', 'Paint']
```

RESULT

The python program to perform various operations of list was created and executedsuccessfully.

EX.NO: 06(B)	IMPLEMENTING APPLICATION USING TUPLES
DATE:	

AIM:

To write a python program to perform various operations of tuples.

ALGORITHM:

Step 1: Start the program
Step 2: Declare the components of automobile using tuplesStep 3:
Perform the operations of tuples
Step 4: Display the output
Step 5: Stop

PROGRAM:

```
Tuple1 = (0,1,2,3,4,5,6)
Tuple2 = ('python','java')
Tuple3 = (Tuple1,Tuple2)
print("\nTuple with nested tuples:\n")
print(Tuple3)
Tuple3 = Tuple1 + Tuple2 print("\nTuples
after Concatenation:\n")print(Tuple3)
print("\nRemoval of First Element: \n")
print(Tuple1[1:])
print("\nTuple after sequence of Element is reversed: \n")
print(Tuple1[::-1])
print("\nPrinting elements between Range 4-9: \n")
print(Tuple1[3:6])
```

OUTPUT:

Tuple with nested tuples:

((0, 1, 2, 3, 4, 5, 6), ('python', 'java'))

Tuples after Concatenation:

(0, 1, 2, 3, 4, 5, 6, 'python', 'java')

Removal of First Element:

(1, 2, 3, 4, 5, 6)

Tuple after sequence of Element is reversed:

(6, 5, 4, 3, 2, 1, 0)

Printing elements between Range 4-9:

(3, 4, 5)

RESULT

The python program to perform various operations of tuples was created and executedsuccessfully.

EX.NO: 07(A)	IMPLEMENTING APPLICATION USING SETS
DATE:	

AIM:

To write a python program to perform various operations of sets.

ALGORITHM:

Step 1: Start the program
Step 2: Declare the components of automobile using setsStep 3: Perform the operations of sets
Step 4: Display the output
Step 5: Stop

PROGRAM:

```
A = {0, 2, 4, 6, 8}
B = {1, 2, 3, 4, 5}
print("Union :", A | B)
print("Intersection :", A & B)
print("Difference :", A - B)
print("Symmetric difference :", A ^ B)
```

OUTPUT:

```
Union : {0, 1, 2, 3, 4, 5, 6, 8}
Intersection : {2, 4}
Difference : {0, 8, 6}
Symmetric difference : {0, 1, 3, 5, 6, 8}
```

RESULT

The python program to perform various operations of sets was created and executedsuccessfully.

EX.NO: 07(B)	IMPLEMENTING APPLICATION USING DICTIONARY
DATE:	

AIM:

To write a python program to perform various operations of dictionary.

ALGORITHM:

Step 1: Start the program
Step 2: Declare the components of automobile using dictionaryStep 3:
Perform the operations of dictionary
Step 4: Display the output
Step 5: Stop

PROGRAM:

```
Dict = {}
print("Empty Dictionary: ")
print(Dict)
Dict[0] = 'python'
Dict[2] = 'java'
Dict[3] = 1
print("\nDictionary after adding 3 elements: ")
print(Dict)
Dict['Value_set'] = 2, 3, 4
print("\nDictionary after adding 3 elements: ")
print(Dict)
Dict[2] = 'Welcome'
print("\nUpdated key value: ")
print(Dict)
Dict[5] = {'Nested' :{'1' : 'Life', '2' : 'python'}}
print("\nAdding a Nested Key: ")
print(Dict)
print("\nAccessing a element using key:")
print(Dict[0])
print(Dict[5]['Nested'])
del Dict[3]
print("\nDeleting a specific key: ")
print(Dict)
pop_ele = Dict.pop(5)
print('\nDictionary after deletion: ' + str(Dict)) print('\nValue
associated to poped key is: ' + str(pop_ele))
```

OUTPUT:

```
Empty Dictionary:
{}

Dictionary after adding 3 elements:
{0: 'python', 2: 'java', 3: 1}

Dictionary after adding 3 elements:
{0: 'python', 2: 'java', 3: 1, 'Value_set': (2, 3, 4)}

Updated key value:
{0: 'python', 2: 'Welcome', 3: 1, 'Value_set': (2, 3, 4)}

Adding a Nested Key:
{0: 'python', 2: 'Welcome', 3: 1, 'Value_set': (2, 3, 4), 5: {'Nested': {'1': 'Life', '2': 'python'}}}

Accessing a element using key:
python
{'1': 'Life', '2': 'python'}

Deleting a specific key:
{0: 'python', 2: 'Welcome', 'Value_set': (2, 3, 4), 5: {'Nested': {'1': 'Life', '2': 'python'}}}

Dictionary after deletion: {0: 'python', 2: 'Welcome', 'Value_set': (2, 3, 4)}

Value associated to poped key is: {'Nested': {'1': 'Life', '2': 'python'}}
```

RESULT

The python program to perform various operations of dictonary was created andexecuted successfully.

EX.NO: 08 DATE:	IMPLEMENTING PROGRAMS USING FUNCTIONS

AIM:

To write a python program to find the factorial of a number using function.

ALGORITHM:

Step 1: Start
Step 2: Define factorial (num)
Step 2.1: Initialize fact=1
Step 2.2: Set a for loop with i in range(1,n+1)
Step 2.2.1: Calculate fact = fact * i
Step 2.3: return fact
Step 3: Read num
Step 4: Print factorial(num)
Step 5: Stop

PROGRAM:

```
def factorial(num):
   fact=1
   for i in range(1,num+1):
      fact=fact*i
   return fact
num=int(input("Enter The number: ")) print("The
factorial of",num,"is",factorial(num))
```

OUTPUT:

```
Enter The number: 5
The factorial of 5 is 120
```

RESULT

The write a python program to find the factorial of a number using function was createdand executed successfully.

EX.NO: 09 DATE:	IMPLEMENTING PROGRAMS USING STRINGS

AIM:

To write a python program to check whether the string is palindrome or not.

ALGORITHM:

Step 1: Start
Step 2: Read str
Step 3: Set rev=str[::-1]
Step 4: Check If rev==str, the go to next step, otherwise go to step 5
Step 4.1: Print(" Palindrome")
Step 5: Print(" Not a palindrome")
Step 6: Stop

PROGRAM:

```
def isPalindrome(s):
    return s == s[::-1]
s = input("enter word to check palindrome or not:\n")ans
= isPalindrome(s)
if ans:
    print("Yes, it is palindrome")
else:
    print("No, it is not palindrome")
```

OUTPUT:

```
enter word to check palindrome or not:
level
Yes, it is palindrome
```

RESULT

The write a python program to check whether the string is palindrome or not wascreated and executed successfully.

EX.NO: 10(A)	IMPLEMENTING PROGRAMS USING WRITTEN MODULESAND PYTHON STANDARD LIBRARIES (PANDAS)
DATE:	

AIM:

To write a python program to print the series using an inbuilt package pandas.

ALGORITHM:

Step 1: Start
Step 2: Import the pandas package
Step 3: Using pd.series operation from pandas print the seriesStep 4: Display the output
Step 5:Stop

PROGRAM:

```
import pandas as pd
A=['python','maths','physics','chemistry','english']
df=pd.Series(A)
print(df)
```

OUTPUT:

```
0    python
1    maths
2    physics
3    chemistry
4    english
```

RESULT

Thus to write a python program to print the series using an inbuilt package pandas was created and executed successfully.

EX.NO: 10(B)	IMPLEMENTING PROGRAMS USING WRITTEN MODULESAND PYTHON STANDARD LIBRARIES (NUMPY)
DATE:	

AIM:

To write a python program to create an array using NumPy module.

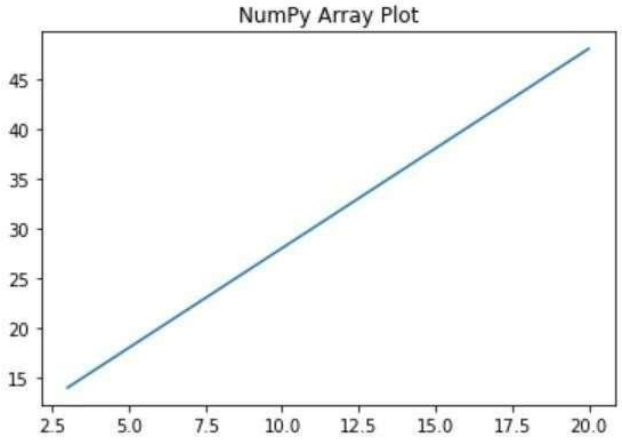
ALGORITHM:

Step 1: Start
Step 2: Import the numpy module
Step 3: Using np.array operation print array shape, size and dimensionsStep 4: Display the output
Step 5: Stop

PROGRAM:

```
import numpy as np
from matplotlib import pyplot as pltx
= np.arange(3,21)
 y = 2 * x + 8
plt.title("NumPy Array Plot")
plt.plot(x,y)
plt.show()
```

OUTPUT:



RESULT

Thus to write a python program to create an array using NumPy module was created and executed successfully.

EX.NO: 10(C) DATE:	IMPLEMENTING PROGRAMS USING WRITTEN MODULESAND PYTHON STANDARD LIBRARIES (MATPLOTLIB)

AIM:

To write a python program to plot a graph using an inbuilt package matplotlib.

ALGORITHM:

Step 1: Start
Step 2: Import the matplotlib package
Step 3: Get the input array to plot the graph
Step 4: Using plt.plot operation plot the numbers in he graphStep
5: Display the graph using plt.show()
Step 6: Stop

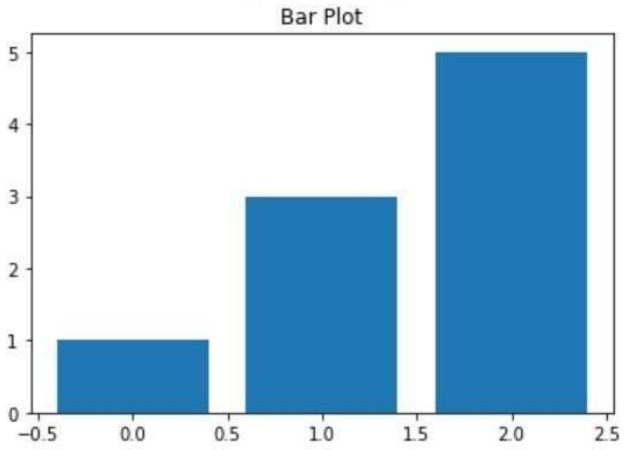
PROGRAM:

```
import matplotlib.pyplot as plt

xdata=['A','B','C']
ydata=[1,3,5]

plt.bar(range(len(xdata)),ydata)
plt.title('Bar Plot')
plt.show()
```

OUTPUT:



RESULT

Thus to write a python program to plot a graph using in built pacakage matplotlib was created and executed successfully.

EX.NO: 10(D)	IMPLEMENTING PROGRAMS USING WRITTEN MODULESAND PYTHON STANDARD LIBRARIES (SCIPY)
DATE:	

AIM:

To write a python program to plot a graph using an inbuilt package scipy .

ALGORITHM:

Step 1: Start
Step 2: Import the scipy package
Step 3: Find the inverse of the matix
Step 4: Using the linalg.inv method operation plot the numbers in he graph
Step 5: Display the matrix using print()
Step 6: Stop

PROGRAM:

```
import numpy as np
from scipy import linalg
A = np.array([[1,2], [4,3]])
B = linalg.inv(A)
print(B)
```

OUTPUT:

```
[[-0.6 0.4]
[ 0.8 -0.2]]
```

RESULT

Thus to write a python program to find the inverse of a matrix was created and executed successfully.

EX.NO: 11(A)	IMPLEMENTING REAL-TIME APPLICATIONS USING FILE HANDLING
DATE:	

AIM:

To write a python program to perform copy of the file using file operations.

ALGORITHM:

Step 1: Start

Step 2: Open a file in read mode using file pointer (f1).
Step 3: Open another file in write mode using file pointer (f2).Step 4: Traverse through all the elements in f1 using for loop Step 5: Apply copy method to copy content from f1 to f2.
Step 6: Stop

PROGRAM:

```
print("Enter the Name of Source File: ")
sFile = input()
print("Enter the Name of Target File: ")
tFile = input()
fileHandle = open(sFile, "r")
texts = fileHandle.readlines()
fileHandle.close()
fileHandle = open(tFile, "w")
for s in texts:
        fileHandle.write(s)
fileHandle.close()
print("\nFile Copied Successfully!")
```

OUTPUT:

First.txt

I am the content from

first pageSecond.txt

I am the content from first page

RESULT

The write a python program to perform copy of the file using file operations was created and executed successfully.

EX.NO: 11(B)	IMPLEMENTING TECHNICAL APPLICATIONS
DATE:	USING FILE HANDLING

AIM:

To write a python program to perform count no of words in a file using fileoperations.

ALGORITHM:

Step 1: Start

Step 2: Open a file in read mode using file pointer.
Step 3: Read a line from file.
Step 4: Split the line into words and store it Step
5: increment count by 1 for each word Step 6:
find count of words using len() functionStep 7:
Print Count
Step 8: Stop

PROGRAM:

```
number_of_words =  0 with
open('s.txt','r') as file:
data = file.read()
# Splitting the data into separate lines using the split() functionlines
= data.split()
number_of_words  += len(lines) #
Printing total number of words
print("The Number of Words :", number_of_words)
```

OUTPUT:

```
The Number of Words : 6
```

RESULT

The write a python program to perform count no of words in a file using file operations was createdand executed successfully.

EX.NO: 12(A)	IMPLEMENTING REAL-TIME APPLICATIONS USING EXCEPTION HANDLING
DATE:	

AIM:

To write a python program to implement divide by zero using Exception handling.

ALGORITHM:

Step 1: Start
Step 2: Take inputs from the user, two numbers.
Step 3: If the entered data is not integer, throw an exception.
Step 4: If the remainder is 0, throw divide by zero exception
Step 5: If no exception is there, return the result.
Step 6: Stop the program.

PROGRAM:

```
a=int(input("Enter the value of a:"))
b=int(input("Enter the value of b:"))
try:
   c=a/b
   print("subraction of a and b is",c)
except ZeroDivisionError:
   print("Error! Division by Zero!")
```

OUTPUT:

```
Enter the value of a:20
Enter the value of b:4
subraction of a and b is 5.0
```

RESULT

The write a python program to implement divide by zero using Exception handling was created and executed successfully.

EX.NO: 12(B)	IMPLEMENTING TECHNICAL APPLICATIONS USING EXCEPTION HANDLING
DATE:	

AIM:

To write a python program to implement voter's age validity using Exception handling

ALGORITHM:

Step 1: Start
Step 2: Declare the variables to get the input age from user
Step 3: Inside the try block print the eligibility to vote based on the conditionStep 4: Catch the exception and display the appropriate message.
Step 5: Stop the program.

PROGRAM:

```
def main():
   try:
      age=int(input("Enter your age:"))if
      age>18:
         print("Eligible to vote")
      else:
         print("Not eligible to vote")
   except:
      print("age must be a valid number")
main()
```

OUTPUT:

```
Enter your age:12
Not eligible to vote

Enter your age:27
Eligible to vote

Enter your age:G
age must be a valid number
```

RESULT

To write a python program to implement divide by zero using Exception handling was created and executed successfully.

EX.NO:13	CREATING AND INSTANTIATING CLASSES
DATE:	

AIM:

To create a program for instantiating classes.

ALGORITHM:

Step 1:Start of the program

Step 2:Define a class ‘person’

Step 3:Pass three arguments name,age,course

Step 4:Declare these parameters with self argument.

Step 5: Use function display () to print the result

Step 6:Create an object p for class Person

Step 7:Call the function display()

Step 8:End of the program.

PROGRAM:

```
class Person:
      def_init_(self,name,age,course):
            self.name=name
            self.age=age
            self.course=course
     def display(self):
           print(“Hello My name is:”,self.name)
           print(“My age is:”,self.age)
           print(“My course is:”,self.course)
p=Person(‘Sarath’,21,’MCA’)
p.display()
```

OUTPUT:

```
Hello My name is:Sarath
My age is:21
My course is:MCA
```

RESULT:

To write the program to create the class instantiation was created and executed successfully.