INDEX

Ex. No. 1(a)	**Electricity Bill Preparation**
Date :	

Aim

To implement a python program for preparing Electricity Bill.

Algorithm

```
1. Start
2. Read the previous month reading and current month reading
3. if units<=100 then
       amttopay=0
   elif units>100 and units<=200:
       amttopay=(units-100)*0.8
   elif units>200 and units<=300:
       amttopay=(units-200)*1+80
   elif units>300 and units<=400:
       amttopay=(units-300)*1.5+100+80
   else:
       amttopay=(units-400)*2+150+100+80
4. print units and amount to pay
5. Stop
```

```
#Program to calculate EB Bill
prev=int(input("Enter previous month reading : "))
curr=int(input("Enter present month reading : "))
units=curr-prev
if units<=100:
   amttopay=0
   print("You have consumed less electricity. No amount to pay for this month")
elif units>100 and units<=200:
     amttopay=(units-100)*0.8
elif units>200 and units<=300:
     amttopay=(units-200)*1+80
elif units>300 and units<=400:
    amttopay=(units-300)*1.5+100+80
else:
     amttopay=(units-400)*2+150+100+80
print("You have consumed ",units," units")
print("Your Electricity Bill for this month : ",amttopay)
```

Output

```
Enter previous month reading : 2430
Enter present month reading : 2645
You have consumed 215 units
Your Electricity Bill for this month : 95
```

Result:

Thus the program for Electricity Bill preparation executed successfully.

Ex. No. 1(b)	Retail Shop Billing
Date :	

Aim

To implement a python program for preparing Retail shop billing
.

Algorithm

1. Start
2. Read the number of items purchased
3. Create empty list itemname , units and price
4. Read the name of the item , no of units and price and append to respective list
5. Compute the amount for each item and the total amount to pay
6. Generate the bill with the particulars item name , units , price and amount
7. Print total amount to pay
8. Stop

#Program to prepare a shopping bill

```
n=int(input("Enter no. of items purchased : "))
itemname=[]
units=[]
price=[]
totamt=0
for i in range(1,n+1):
  item=input("Enter name of the item purchased : ")
  u=int(input("Enter no. of units purchased : "))
  p=int(input("Enter price of the item purchased : "))
  itemname.append(item)
  units.append(u)
  price.append(p)
count=len(units)
print("\t\t\t\tXYZ Super Market\n\n")
print("\tItem Name\t\tUnits\tPrice\t\tAmount")
for i in range(count):
  print("\t",itemname[i],"\t\t\t",units[i],"\t",price[i],"\t\t",units[i]*price[i]) totamt=totamt+units[i]*price[i]
print("\n\n\t\t\t\tTotal amount to pay : ",totamt)

print("\n\n\n\t\t\t\t Thank You , Visit Again")
```

Output

```
Enter no. of items purchased : 3
Enter name of the item purchased : soap
Enter no. of units purchased : 5
Enter price of the item purchased : 60
Enter name of the item purchased : oil
Enter no. of units purchased : 2
Enter price of the item purchased : 220
Enter name of the item purchased : Biscuit
Enter no. of units purchased : 8
Enter price of the item purchased : 20

                              XYZ Super Market

        Item Name              Units   Price            Amount
         soap                   5       60               300
         oil                    2       220              440
         Biscuit                8       20               160

                         Total amount to pay : 900


                          Thank You , Visit Again
```

Result

Thus the program for retail shopping bill executed successfully.

Ex. No. 1(c)	**Sine Series**
Date :	

Aim

To generate sine series for the given input.

Algorithm

1. Start
2. Read 2 values x and n
3. Initialize *sum* to 0 , sign to -1 , fact and i to 1.
4. Calculate p and x for range 1 to i*2
5. Change the sign , multiply sign*p/fact and add it to sum
6. Print the sine series value
7. Stop

```
#program to find sum of sine series x-x3/3!+x5/5!........
x=int(input("Enter value of x"))
n=int(input("Enter value for n"))
sign=-1
fact=i=1
sum=0
while i<=n:
    p=1
    fact=1
    for j in range(1,i*2):
        p=p*x
        fact=fact*j
    #print(p,"    ",fact)
    sign=-1*sign
    sum=sum+sign*p/fact
    i=i+1
print("Sin( ",x," ) = ",sum)
```

Output

```
Enter value of x2
Enter value for n2
Sin( 2  ) =  0.6666666666666667
```

Result

Thus the sine series for the given input generated successfully.

Ex. No. 1(d)	**Weight of a motor bike**
Date :	

Aim

To find the weight of a motor bike from the list of bikes stored.

Algorithm

1. Start
2. Create a list bikebrand and bikewt to store the name of the bikes and the weight of the bikes
3. Get the vehicle name to know the weight
4. Compare the name of the vehicle from the bikebrand and print the corresponding bike weight
5. Stop

#Program to calculate the weight of a motor bike

```
Bikebrand=[“TVS Jupiter”,”Honda Activa”,”Suzuki Access”, “Passion Pro”,”Apache
RTR”]
Bikewt=[“109 Kgs”,”105 Kgs”,”103 Kgs”,”118 Kgs”,”138 Kgs”]
Bn=input(“Enter the vehicle name to know the weight”)
for i in range(0,len(bikebrand)):
    if bn==bikebrand[i]:
        print(“Weight of the Bike “,bikebrand[i],” is “,bikewt[i])
```

Output

```
Enter the vehicle name to know the weight
TVS Jupiter
Weight of the bike TVS Jupiter is 109 Kgs
```

Result

Thus the weight of a motorbike given by user generated successfully.

Ex. No. 1(e)	**Finding weight of a steel bar**
Date :	

Aim

To write a python program for finding the weight of a steel bar**.**

Algorithm

1. Start
2. Read the diameter and length of the steel bar
3. Find the wt of the steel bar using the formula
   wt=((d**2)/162.2)*l
4. Print the weight
5. Stop

```
#Program to calculate the weight of a steel bar
d=int(input("Enter diameter of the steel bar "))
l= int(input("Enter length of the steel bar "))
wt=((d**2)/162.2)*l
print("Weight of the steel bar with diameter ",d, " and length ",l, " is ",wt)
```

Output

```
Enter diameter of the steel bar 20
Enter length of the steel bar 5
Weight of the steel bar with diameter 20 and length 5 is 12.33045
```

Result

Thus weight of the steel bar was calculated.

Ex. No. 1(f)	**Compute Electric current in 3 phase AC circuit**
Date :	

Aim

To write a python program to compute electric current in 3 phase AC circuit

Algorithm

1. Start
2. Read voltage , resistance , current and power factor from the user
3. Compute electric current=3*Voltage*resistance*Current*Power factor
4. Stop

#Program to compute electric current in 3 phase AC circuit

```
v=int(input(“Enter the voltage of the AC Circuit”))

c=int(input(“Enter the current value of the AC Circuit”))

pf=float(input(“Enter the power factor of the AC Circuit”))

ec=√3*v*c*pf

print(“Electric current in 3 phase AC circuit”,ec)
```

Output

```
Enter the voltage of the AC Circuit 220

Enter the current value of the AC Circuit 20

Enter the power factor of the AC Circuit 0.6

Electric current in 3 phase AC circuit 4572.48
```

Result

Thus the program for computing electric current in 3 phase AC circuit executed successfully.

Ex. No. 2(a)	**Exchange the values of two variables**
Date :	

Aim

To write a python program to exchange values of 2 variables

Algorithm

1. Start
2. Read 2 numbers as a and b
3. print the values got from user
4. call the function swap by passing the 2 parameters a and b
5. Assign the value of a to b and b to a
6. print the values after exchange
7. Stop

#Program to exchange the value of 2 variables

```
def swap(a,b):
      a,b=b,a
      print(“After swapping”)
      print(“Value of a = “,a, “ and value of b = “,b)
a=int(input(“Enter first number”))
b=int(input(“Enter second number”))
print(“Before swapping”)
print(“Value of a = “,a, “ and value of b = “,b)
swap(a,b)
```

Output

```
Enter first number25
Enter second number45
Before swapping
Value of a =  25  and value of b =  45
After swapping
Value of a = 45  and value of b =  25
```

Result

Thus the values of 2 variables exchanged successfully**.**

Ex. No. 2(b)	**Circulate the values of n variables**
Date :	

Aim

To write a python program to circulate the values of the elements stored in a list.

Algorithm

1. Start
2. Read the no. of items to store in the list as *n*
3. Get the n element values and store it in the list origlist
4. Call function rotate by passing the list and nooftimes as argument
5. Get the items from the list from index number nooftimes to end of the list and append it with the elements upto index number nooftimes.
6. print the updated list
7. Repeat steps 5 and 6 based on no. of elements in the list
8. Stop

#Program to circulate values of n variables

```
def rotate(listname,nooftimes):
        updatelist=listname[nooftimes:]+listname[:nooftimes]
        return updatelist
origlist=[]
n=int(input(“Enter no. of items to store in list”))
for i in range(n):
       item=int(input(“Enter item value”))
       origlist.append(item)

print(“Original list”)
print(origlist)
for i in range(1,len(origlist)):
        rotlist=rotate(origlist,i)
        print(“rotated list after “,i, “ rotation”,rolist)
```

Output

```
Enter no. of items to store in list4
Enter item value34
Enter item value45
Enter item value22
Enter item value88
Original list
[34, 45, 22, 88]
rotated list after  1  rotation  [45, 22, 88, 34]
rotated list after  2  rotation  [22, 88, 34, 45]
rotated list after  3  rotation  [88, 34, 45, 22]
```

Result

Thus the program for circulating the values of n variables executed successfully.

Ex. No. 2(c)	**Finding distance between 2 points**
Date :	

Aim

To write a python program to find the distance between 2 points.

Algorithm

1. Start
2. Read the values of the first point and store it as x1,y1
3. Read the values of the second point and store it as x2,y2
4. Find the distance between 2 points using the formula

   $$\textbf{Distance}=\sqrt{(X_2-X_1)^2+(Y_2-Y_1)^2}$$
5. print the distance computed
6. Stop

#Program to find distance between 2 points

$$\textbf{Distance}=\sqrt{(X_2-X_1)^2+(Y_2-Y_1)^2}$$

```
import math
x1=int(input("Enter values for x1 : "))
y1=int(input("Enter values for y1 : "))
x2= int(input("Enter values for x2 : "))
y2= int(input("Enter values for y2 : "))
dist=math.sqrt(((x2-x1)**2)+((y2-y1)**2))
print(dist)
```

Output

```
Enter values for x1 : 20
Enter values for y1 : 28
Enter values for x2 : 30

Enter values for y2 : 30
10.198039027185569
```

Result

Thus the distance between 2 points computed successfully.

Ex. No. 3(a)	**Number Series**
Date :	

Aim

To generate a number series upto a given number.

Algorithm

1. Start
2. Read maximum number for the series *n*
3. Initialize the value of i as 1 and create an empty list lst.
4. Print value of i and append the i value to lst
5. Print the sum.
6. Repeat the step 4 , 5 till it reaches n+1
7. Stop

#Program to generate number series

```
n=int(input("Enter the maximum no. for the series : "))
lst=[]
for i in range(1,n+1):
    print(i,"+ =",end=" ")
    lst.append(i)
    print(sum(lst))
```

Output

```
Enter the maximum no. for the series5
1 + = 1
2 + = 3
3 + = 6
4 + = 10
5 + = 15
```

Result

Thus the number series for the given range printed successfully.

Ex. No. 3(b)	**Number Pattern**
Date :	

Aim

To print the number pattern based on the input given by the user.

Algorithm

1. Start
2. Read maximum number for the series *n*
3. Initialize the value of i as 1 and j as 1.
4. Print the number pattern with the value of i printed j times
5. Repeat the step 4 till it reaches n+1
6. Print the number pattern with the value of j printed i times
7. Repeat the step 6 till it reaches n+1
8. Stop

#Program to generate number pattern

```
n=int(input("Enter the maximum no. for the series : "))
print("Number pattern 1")
for i in range(1,n+1):
    for j in range(1,i+1):
        print(i,"  ",end=" ")
    print("\n")

print("Number pattern 2")
for i in range(1,n+1):
    for j in range(1,i+1):
        print(j,"  ",end=" ")
    print("\n")
```

Output

```
Enter the maximum no. for the series : 5
Number pattern 1
1
2    2
3    3    3
4    4    4    4
5    5    5    5    5


Number pattern 2
1
1    2
1    2    3
1    2    3    4
1    2    3    4    5
```

Result

Thus the number pattern in 2 different formats printed successfully.

Ex. No. 3(c)	**Number Pyramid**
Date :	

Aim

To print the pyramid format pattern based on the input given by the user.

Algorithm

1. Start
2. Read number of rows for the series r
3. Calculate number of spaces to be printed before character.
4. Print the character pattern
5. Repeat the step 4 till condition is satisfied
6. Stop

#Program to print pyramid pattern

```
r=int(input("Enter no.of.rows:"))
k=0
for i in range(1,r+1):
        for space in range(1,(r-i)+1):
                print(end="")
                 while k!=(2*i-1):
                        print("*",end="")
                        k+=1
        k=0
        print()
```

Output

```
Enter no.of.rows:5
            *
           ***
          *****
         *******
        *********
```

Result

The character pattern in pyramid format printed successfully

Ex. No. 4(a)	**Items present in a library using list**
Date :	

Aim

To print the items present in a library using different list operations.

Algorithm

1. Start
2. Create a list in the name library and store the elements in it.
3. Print the entire list using print function.
4. Print the 1st element and the 4th element using index number of the list.
5. Print elements from index 0 to 4
6. Add a new element in the list using append function
7. Print the updated list of items in the library
8. Stop

#Program to print the items present in a library

```
# declaring a list of items in a Library
library=['Books','Periodicals','Newspaper','Manuscripts','Maps','Prints','Documents','Ebooks
']
# printing the complete list
print('Library:  ',library)
# printing first element
print('first element: ',library[0])
# printing fourth element
print('fourth element:  ',library[3])
# printing list elements from 0th index to 4th index
print('Items in Library from 0 to 4 index: ',library[0: 5])
# printing list -7th or 3rd element from the list
print('3rd or -7th element: ',library[-7])
# appending an element to the list
library.append('Audiobooks')
print('Library list after append():  ',library)
```

Output

```
Library:  ['Books', 'Periodicals', 'Newspaper', 'Manuscripts', 'Maps', 'Prints', 'Documents',
'Ebooks']
first element:  Books
fourth element:  Manuscripts
Items in Library from 0 to 4 index:  ['Books', 'Periodicals', 'Newspaper', 'Manuscripts',
'Maps']
3rd or -7th element:  Periodicals
Library list after append():  ['Books', 'Periodicals', 'Newspaper', 'Manuscripts', 'Maps',
'Prints', 'Documents', 'Ebooks', 'Audiobooks']
```

Result

Thus the number of items present in the library using list operations printed.

Ex. No. 4(b)	**Components of a car using tuple**
Date :	

Aim

To demonstrate tuple operations by storing the components required to assemble a car.

Algorithm

1. Start
2. Create 2 tuples in the name car1 and car2 and store the elements
3. Print values of a tuple car2 at index 1
4. Print number of components in each tuple using len()
5. Join contents of both tuple using + and print
6. Repeat the content of car1 twice using * operator
7. Check a component name is available or not using membership operator in and notin
8. Print a part of content in tuple using range of index numbers from and to
9. Stop

#Program to store spare parts required to assemble a car

```
car1=("steering","wheels","brake","engine","seats")
car2=("accelerator","clutch","gear","horn","indicator","battery")
print("Indexing position of car2 : ",car2[1])
print("Numbers of components in car 1 : ",len(car1))
print("Numbers of components in car 2 : ",len(car2))
print("Concatenation of two car : ",car1+car2)
print("Repetition of car1 : ",car1*2)
print("Membership operator of car1 : ","engine" in car1)
print("Membership operator of car2 : ","components" not in car2)
print("Slicing of car2:",car2[0:2])
```

Output

```
Index 2 content of car 2 : clutch
Numbers of components in car1 5
Numbers of components in car 2 6
Concatenation of two car ('steering', 'wheels', 'brake', 'engine', 'seats', 'accelerator', 'clutch',
'gear', 'horn', 'indicator', 'battery')
Repetition of car1 ('steering', 'wheels', 'brake', 'engine', 'seats', 'steering', 'wheels', 'brake',
'engine', 'seats')
Membership operator of car1: True
Membership operator of car2: True
Slicing of car2: ('accelerator', 'clutch')
```

Result

Components of automobile to assemble a car using tuple operations executed

Ex. No. 4(c)	**Materials required for construction of a building using tuple**
Date :	

Aim

To demonstrate tuple operations by storing the components required to assemble a car.

Algorithm

1. Start
2. Create 2 tuples in the name building1 and building2 and store the elements
3. Print values of a tuple building2 at index 1
4. Print number of components in each tuple using len()
5. Join contents of both tuple using + and print
6. Repeat the content of car1 twice using * operator
7. Check a component name is available or not using membership operator in and not in
8. Print a part of content in tuple using range of index numbers from and to
9. Stop

#Program to store materials required for building construction

```
building1=("bricks","sand","cement")
building2=("tiles","paint","wood","pipes")
print("Index 1 of building1 : ",building1[1])
print("Numbers of components in building1 : ",len(building1))
print("Numbers of components in building2 : ",len(building2))
print("Concatenation of two building ",building1+building2)
print("Repetition of building1 ",building1*2)
print("Membership operator of building1: ","components" in building1)
print("Membership operator of building2: ","paint" in building2)
print("Slicing of building1:",building1[0:1])
```

Output

```
Index 1 of building1 : sand
Numbers of components in building1 : 3
Numbers of components in building2 : 4
Concatenation of two building  ('bricks', 'sand', 'cement', 'tiles', 'paint', 'wood', 'pipes')
Repetition of building1  ('bricks', 'sand', 'cement', 'bricks', 'sand', 'cement')
Membership operator of building1:  False
Membership operator of building2:  True
Slicing of building1: ('bricks',)
```

Result

Materials required for construction of a building are stored and printed using different tuple operations executed.

Ex. No. 5(a)	**Languages stored in a set**
Date :	

Aim

To demonstrate set operations by storing the computer language details in different sets

Algorithm

1. Start
2. Create 2 sets in the name langset1 and langset2 and store the elements
3. Print contents in each set
4. Combine the sets and print the contents using union operator |
5. Print the common languages in both set using intersection operator &
6. Print contents of languages in set1 but not in set2 using operator –
7. Stop

Program to print languages stored in a set

```
langset1={"C","C++","Java",".Net","Python"}
langset2={"PHP","MySQL","HTML","Python"}
print("Languages in set 1 : ",langset1)
print("Languages in set 2 : ",langset2)
print("Languages in both sets : ",langset1|langset2)
print("Languages common in both sets : ",langset1&langset2)
print("Languages in set 1 but not in set 2 are : ",langset1-langset2)
```

Output

```
Languages in set 1 : {'C++', '.Net', 'C', 'Python', 'Java'}
Languages in set 2 :  {'PHP', 'HTML', 'Python', 'MySQL'}
Languages in both sets :  {'Python', 'PHP', '.Net', 'HTML', 'MySQL', 'C++', 'C', 'Java'}
Languages common in both sets :  {'Python'}
Languages in set 1 but not in set 2 are :  {'C++', 'Java', 'C', '.Net'}
```

Result

Thus languages are stored in sets and executed using different set operations.

Ex. No. 5(b)	**Components of an automobile using set**
Date :	

Aim

To demonstrate set operations by storing the components of automobile cat and bike in different sets

Algorithm

1. Start
2. Create 2 sets in the name carcomp and bikecomp.
3. Print contents in each set
4. Combine the sets and print the contents using union operator |
5. Print the common components in both set using intersection operator &
6. Print contents of components in set1 but not in set2 using operator –
7. Stop

```
#Program to print components of an automobile using set
carcomp={"chassis","engine","suspension","axle","doors","Wiper","fuel tank"}
print("components of a car are ",end="")
print(carcomp)
bikecomp={"engine","brake","fuel tank"}
print("components of a bike are ",end="")
print(bikecomp)
print("components common to car and bike are ")
print(carcomp&bikecomp)
print("components of car and bike combined are ")
print(carcomp|bikecomp)
print("components of car but not for bike are ")
print(carcomp-bikecomp)
```

Output

```
components of a car are {'engine', 'axle', 'doors', 'fuel tank', 'chassis', 'suspension', 'Wiper'}
components of a bike are {'engine', 'brake', 'fuel tank'}
components common to car and bike are
{'engine', 'fuel tank'}
components of car and bike combined are
{'suspension', 'Wiper', 'engine', 'doors', 'chassis', 'axle', 'brake', 'fuel tank'}
components of car but not for bike are
{'Wiper', 'suspension', 'doors', 'chassis', 'axle'}
```

Result

Thus components of automobile car and bike are stored in sets and executed using different set operations.

Ex. No. 5(c)	**Elements of a civil structure**
Date :	

Aim

To demonstrate dictionary operations by storing the elements of a civil structure.

Algorithm

1. Start
2. Create a dictionary ele with 7 elements of key and value.
3. Print contents in the dictionary
4. Add a new key floor and print the dictionary content
5. Change the value of the key floor and print the updated content in the dictionary
6. Remove the key window and show updated content
7. Stop

```
#Program to store elements of a civil structure using dictionary
ele={"Foundation":"Spreadfooting","Column":"structural","roof":"flat","Lintel":"RCC","d
oors":"Teak Wood","Window":"Rose wood","wall":"bricks"}
print("Elements of a civil structure are : ")
print(ele)
ele["floor"]="Tiles"
print("Elements after adding floor")
print(ele)
ele["roof"]="sloped"
print("Elements after changing roof value ")
print(ele)
del ele["Window"]
print("Elements after deleting component window ")
print(ele)
```

Output

```
Elements of a civil structure are :
{'Foundation': 'Spreadfooting', 'Column': 'structural', 'roof': 'flat', 'Lintel': 'RCC', 'doors':
'Teak Wood', 'Window': 'Rose wood', 'wall': 'bricks'}
Elements after adding floor
{'Foundation': 'Spreadfooting', 'Column': 'structural', 'roof': 'flat', 'Lintel': 'RCC', 'doors':
'Teak Wood', 'Window': 'Rose wood', 'wall': 'bricks', 'floor': 'Tiles'}
Elements after changing roof value
{'Foundation': 'Spreadfooting', 'Column': 'structural', 'roof': 'sloped', 'Lintel': 'RCC', 'doors':
'Teak Wood', 'Window': 'Rose wood', 'wall': 'bricks', 'floor': 'Tiles'}
Elements after deleting component window
{'Foundation': 'Spreadfooting', 'Column': 'structural', 'roof': 'sloped', 'Lintel': 'RCC', 'doors':
'Teak Wood', 'wall': 'bricks', 'floor': 'Tiles'}
```

Result

Thus elements of a civil structure are stored in dictionary and executed using different dictionary operations.

Ex. No. 6(a)	**Factorial of a number using function**
Date :	

Aim

To compute the factorial of a given number using function.

Algorithm

1. Start
2. Define a function fact and the value to compute factorial as argument
3. If argument value is 0 , it prints the result as 1.
4. If argument value is a positive number , it calculates the factorial value by calling the function fact() recursively and print the factorial
5. Call the function fact() first by passing value 0 and then by passing value 3
6. Result returned back to called statement and printed.
7. Stop

#Program to print factorial of given number using function

```
def fact(n):
    if n==0:
      return 1
    else
        return n*fact(n-1)
print("Factorial of value 0 : ",fact(0))
print("Factorial of value 3 : ",fact(3))
```

Output

```
Factorial of value 0 : 1
Factorial of value 3 : 6
```

Result

Thus factorial for the given input is computed using function and the results were printed.

Ex. No. 6(b)	**Finding the largest element in the list using function**
Date :	

Aim

To find the largest element in the list using function.

Algorithm

1. Start
2. Read the no. of elements to store nos
3. Create an empty list num
4. Add values to that by getting the input from user and append to the list num.
5. Print contents of list
6. Define a function maxi and pass the list num as argument
7. Assume index 0 as maximum and compare with the elements from index 1 till the end of list.
8. Change the value of mx if there is a value greater than the already stored mx
9. Pass the mx value to the called statement and print the maximum number.
10. Stop

```
#Program to print maximum element in the list
def maxi(n):
    mx=n[0]
    for i in range(1,len(n)):
        if n[i]>mx:
            mx=n[i]
    return mx

nos=int(input("Enter no. of elements to store : "))
num=[]
for i in range(0,nos):
    inp=int(input("Enter a number : "))
    num.append(inp)
print("Elements stored in the list are ")
print(num)
m=maxi(num)
print("Maximum value stored in the list : ",m)
```

Output

```
Enter no. of elements to store : 5
Enter a number : 44
Enter a number : 890
Enter a number : 2345
Enter a number : 2
Enter a number : 588
Elements stored in the list are
[44, 890, 2345, 2, 588]
Maximum value stored in the list :  2345
```

Result

Thus the largest number in the list was identified using function and printed.

Ex. No. 6(c)	**Finding the area of a shape using function**
Date :	

Aim

To find the area of a circle using function.

Algorithm

1. Start
2. Read the radius of the circle rad
3. Define a function circle that accepts an argument radius
4. Return the values area and circumference to the called statement
5. Print area and circumference of the circle
6. Stop

Program to print area of a circle using function

```
pi=3.14
def circle(r):
    return(pi*r*r,2*pi*r)
rad=int(input(“Enter the radius of the circle”))
ar,circum=circle(rad)
print(“Radius of circle : “,rad)
print(“Area of circle : “,ar)
print(“Circumference of circle : “,circum)
```

Output

```
Enter the radius of the circle20
Radius of circle : 20
Area of circle : 1256.0
Circumference of circle :  125.60000000000001
```

Result

Thus the area and circumference of a circle was computed using function and printed.

Ex. No. 7(a)	**Reverse a String**
Date :	

Aim

To write a python program for reversing a string.

Algorithm

1. Start
2. Read the string from the user
3. Find the number of characters in string using len() function
4. Start reading from the reverse index -1 till the beginning of the string
5. Print reversed string
6. Stop

Program to print reverse of a string

```
str1=input(“Enter a string”)
index=-1
print(“The string entered is : “,str1)
print(“The reversed string is : “)
while index>-len(str1):
    print(str1[index],end=“ “)
    index+=-1
```

Output

```
Enter a string
python
The string entered is : python
The reversed string is :
nohtyp
```

Result

Thus the string entered by user was reversed and printed.

Ex. No. 7(b)	**Checking the string is Palindrome**
Date :	

Aim

To write a python program for reversing a string, checking the original and reverse are the same.

Algorithm

1. Start
2. Read the string from the user
3. Find the number of characters in string using len() function
4. Start reading from the reverse index -1 till the beginning of the string
5. Check reversed string and original string are same. If true , print string is palindrome otherwise print string is not a palindrome.
6. Stop

#Program to check given string is palindrome

```
str1=input("Enter a string : ")
index=-1
str2=""
while  index>=-len(str1):
      str2=str2+str1[index]
      index+=-1
print("Reversed string : ",str2)
if(str1==str2):
    print("String entered {} is a palindrome".format(str1))
else:
    print("String entered {} is not a palindrome".format(str1))
```

Output

```
Enter a string : mum
Reversed string :  mum
String entered mum is a palindrome

Enter a string : liril
Reversed string :  liril
String entered liril is a palindrome

Enter a string : sets
Reversed string :  stes
String entered sets is not a palindrome
```

Result

Thus the string entered by user is a palindrome or not and printed.

Ex. No. 7(c)	**Counting characters in a String**
Date :	

Aim

To write a python program for counting the number of characters in a string.

Algorithm

1. Start
2. Read the string from the user
3. Find the number of characters in string using len() function
4. Initialize count as 0 and index as 0
5. Increment the count once a character for each index
6. Repeat step 5 till the end of the string is reached
7. Print the number of characters stored in count.
8. Stop

#Program to count number of characters in a string

```
str1=input("Enter a string : ")
index=0
count=0
while index<len(str1):
    count+=1
    index+=1
print("Number of characters in the string are : ",count)
```

Output

```
Enter a string : aravinda choudry
Number of characters in the string are : 16
```

Result

Thus the number of characters given in a string printed.

Ex. No. 7(d)	**Replace characters in a String**
Date :	

Aim

To write a python program for counting the number of characters in a string.

Algorithm

1. Start
2. Read the string from the user
3. Replace the string given by using replace() function and specify the character to find , replace and optionally number of times to replace
4. Print the string with the replaced characters.
5. Stop

#Program to replace characters in a string

```
str1=input("Enter a string : ")
print("String given as input : ",str1)
print("String replaced with the given character only once ")
print("The changed string is : ",str1.replace("a","v",1))
print("String replaced with the given character upt0 3 occurrences of that character : ")
print("The changed string is : ",str1.replace("a","v",3))
```

Output

```
Enter a string : aravindan
String given as input :  aravindan
String replaced with the given character only once
The changed string is : vravindan
String replaced with the given character upto 3 occurences of that character :
The changed string is :  vrvvindvn
```

Result

Thus the characters given in a string is replaced with another character and printed.

Ex. No. 8(a)	**Program using pandas Dataframes**
Date :	

Aim

To implement program in python for working with Pandas data frame.

Algorithm

1. Start
2. import pandas with an aliased name as pd.
3. Create a given list and assign it to variable data.
4. Call the data Frame function (data), and assign it to variable t.
5. Call the Print function to print the Pandas data frame(t).
6. Stop

#Program to work with pandas Dataframes

```
import pandas as pd
data=={"Name":["Ram","Subash","Raghul","Arun","Deepak"],"Age":[24,25,24,
 26,25],"CGPA":[9.5,9.3,9.0,8.5,8.8]}
 t=pd.DataFrame(data)
t.index+=1
print(t)
```

Output

```
     Name    Age    CGPA
1     Ram    24       9.5
2   Subash     25     9.3
3   Raghul     24     9.0
4    Arun    26       8.5
5   Deepak     25   8.8
```

Result

Thus the program pandas with dataframes executed successfully.

Ex. No. 8(b)	**Program using numpy**
Date :	

Aim

To implement program in python for working with numpy arrays.

Algorithm

1. Start
2. **i**mport numpy with an aliased name as np.
3. Create an array of values with 2 rows and 3 columns.
4. Print the values and the dimension of the array.
5. Reshape the dimension of the array using reshape.
6. Print the values and dimension
7. Stop

#Program using numpy module

```
import numpy as np
a = np.array([(8,9,10),(11,12,13)])
print("Original Dimension of the array : ",a.shape)
print(a)
print("Modified Dimension of the array : ",end="")
a=a.reshape(3,2)
print(a.shape)
print(a)
```

Output

```
Original Dimension of the array : (2, 3)
[[ 8  9 10]
 [11 12 13]]
Modified Dimension of the array : (3, 2)
[[ 8  9]
 [10 11]
 [12 13]]
```

Result

Thus the program numpy with array executed successfully.

Ex. No. 8(c)	**Program using matplotlib**
Date :	

Aim

To implement program in python for working with matplotlib to plot charts.

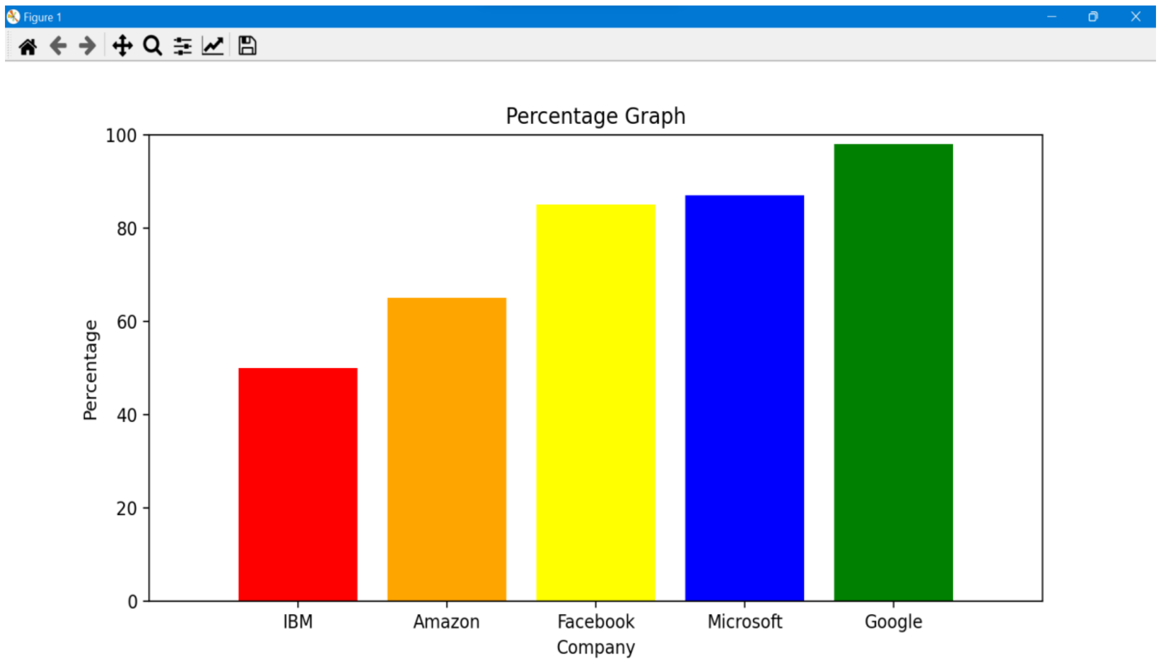
Algorithm

1. Store in x=[1,2,3,4,5]
2. Store in y=[50,65,85,87,98]
3. Store in text=["IBM","Amazon","Facebook","Microsoft","Google"]
4. Store in colors=["red","orange","yellow","blue","green"]
5. Using xlim() and ylim() we can set the points as 0 to 6 on x-axis and 0 to100 on y-axis respectively.
6. Using bar() we can create a bar graph with x,y with label as text andcolor=colors and line width of the graph as 0.5.
7. show the x-axis and y-axis of the plot as ‘Company' and 'Percentage',show the title as Percentage Graph.

#Program to plot a chart using matplotlib module

```
import matplotlib.pyplot as mpl
x=[1,2,3,4,5]
y=[50,65,85,87,98]
text=["IBM","Amazon","Facebook","Microsoft","Google"]
colors=["red","orange","yellow","blue","green"]
mpl.xlim(0,6)
mpl.ylim(0,100)
mpl.bar(x,y,tick_label=text,color=colors,linewidth=0.5)
mpl.xlabel("Company")
mpl.ylabel("Percentage")
mpl.title("Percentage Graph")
mpl.show()
```

Output



Result

Thus the program matplotlib for plotting a bar chart executed successfully.

Ex. No. 8(d)	**Program using Scipy**
Date :	

Aim

To implement program in python for working with Scipy to find determinant of a matrix.

Algorithm

1. Start
2. import package linalg from module scipy
3. import module numpy
4. Generate a 2D array
5. Using function det() find the determinant of the matrix and print
6. Stop.

```
#Program using scipy module to find determinant of the matrix
from scipy import linalg
import numpy as np
two_darray=np.array([[4.5],[3,2]])
linalg.det(two_darray)
```

Output

```
-7.0
```

Result

Thus the program using module scipy for finding determinant of matrix executed successfully.

Ex. No. 9(a)	**File Handling – Copy one file into another**
Date :	

Aim

To implement program in python for copying contents of one file into another.

Algorithm

1. Start
2. Read the name of the source file and destination file
3. Open source file in read mode and destination file in write mode
4. Read one line from source file and write it into destination file
5. Print File Copied Successfully
6. Open the destination file in read mode and print the contents of the file using read()
7. Stop.

#Program to copy the contents of one file into another

```
sfile = input("Enter Source File's Name: ")
tfile = input("Enter Target File's Name: ")
sourfile=open(sfile, "r")
destfile= open(sfile, "w")
for line in sourfile:
        destfile.write(line)
print("File successfully copied")
sourfile.close()
destfile.close()
print("Contents of destination file")
destfile= open(sfile, "r")
print(destfile.read())
destfile.close()
```

Output

```
Enter Source File's Name: aa.txt
Enter Target File's Name: sa.txt
File successfully copied
Contents of destination file
I am writing
this file to
check the working
of copy file
```

Result

Thus the program for copying the contents of one file into another executed successfully.

Ex. No. 9(b)	**File Handling – Word count**
Date :	

Aim

To implement program in python for copying contents of one file into another.

Algorithm

1. Start
2. Read the name of the source file
3. Open source file in read mode
4. Use split() to take out a line from the file
5. Create a dictionary wcount , store the new words and increment the count the existing words
6. After completing the reading of the entire file , Print the count of each word
7. Stop.

```
#Program to count the number of words in a file
sfile = input("Enter Source File's Name: ")
sourfile=open(sfile, "r")
wcount={}
for line in sourfile:
    word_list=line.split()
    for word in word_list:
        if word not in wcount:
            wcount[word]=1
        else:
            wcount[word]=wcount[word]+1
print("Count of each word in file : ")
print("{:15}{:3}".format("Word","Frequency"))
print("-"*25)
for (word,count) in wcount.items():
    print("{:15}{:3}".format(word,count))
```

Output

```
Enter Source File's Name: aa.txt
Count of each word in file :
Word          Frequency
----------------------------
I              1
am               1
in             2
aa.txt           1
this            1
file            1
order            1
to             1
check            2
the             2
working          1
of              1
word             1
and             1
count            1
```

Contents of aa.txt

```
I am in aa.txt
this file in order to
check the working
of the word check and count
```

Result

Thus the program for counting the number of words in a file executed successfully.

Ex. No. 9(c)	**File Handling – Longest word in a file**
Date :	

Aim

To implement program in python for finding the longest word in the file.

Algorithm

1. Start
2.Read the name of the source file
3. Open source file in read mode
4. Use split() to take out a line from the file
5. Create a dictionary wcount , store the new words and increment the count the existing words
6. Find the length of the word and if found to be longer than the exiting longer , replace it with the new word
7. After completing the reading of the entire file , Print the count of each word and also the longest word in the file
8. Stop.

```
#Program to find the longest word in a file
sfile = input("Enter Source File's Name: ")
sourfile=open(sfile, "r")
wcount={}
maxi=0
noofwords=0
for line in sourfile:
    word_list=line.split()
    for word in word_list:
        if word not in wcount:
            wcount[word]=1
            leng=len(word)
            if leng>maxi:
                maxi=leng
                lword=word
        else:
            wcount[word]=wcount[word]+1
        noofwords+=1
print("Words and their occurrences in the file : ")
print(wcount)
print("Number of words in the file : ",noofwords)
print("Longest word : ",lword)
```

Output

```
Enter Source File's Name: aa.txt
Words and their occurrences in the file :
{'I': 1, 'am': 1, 'in': 2, 'aa.txt': 1, 'this': 1, 'file': 1, 'order': 1, 'to': 1, 'check': 2, 'the': 2,
'working': 1, 'of': 1, 'word': 1, 'and': 1, 'count': 1}
Number of words in the file : 18
Longest word :  working
```

Result

Thus the program for finding the longest word in a file executed successfully.

Ex. No. 10(a)	**Exception Handling – Divide by zero error**
Date :	

Aim

To implement program in python for handling the exception divide by zero.

Algorithm

1. Start
2. Read 2 numbers
3. Divide the first number by second number and if second number happens to be zero , divide by zero exception occurs
4. If the second number is other than 0 , it prints the result
5. Stop.

#Program to handle exception Divide by zero

```
try:
    n1=int(input("Enter first number"))
    n2=int(input("Enter second number"))
    x=n1/n2
    print(n1," / ",n2, " =",x)
    y=22/0
    print(y)
except ZeroDivisionError:
    print("Tried to divide by zero")
```

Output

```
58 / 2 = 29.0
Tried to divide by zero
```

Result

Thus the program for handling the exception divide by zero executed successfully.

Ex. No. 10(b)	**Exception Handling – Voters Age validity**
Date :	

Aim

To implement program in python for validating voters age and handling the exception.

Algorithm

1. Start
2. Read age of the person
3. If age>18 print Eligible to vote otherwise print Not Eligible to vote
4. If other than number given , exception ValueError executed and shows Input must be a number
5. If an I/O error occurs , IOError exception executed and shows IOError
6. Stop.

#Multiple exceptions in a try

```
def checkvote():
    try:
        age=int(input("Enter your age : "))
        if age>18:
            print("Eligible to vote")
        else:
            print("Not Eligible to vote")
    except ValueError:
        print("Input must be number")
    except IOError:
        print("IO Error")
        #raise ValueError("Invalid age ")
    print("Rest of the code ....")
checkvote()
```

Output

```
Enter your age : a
Input must be number
Rest of the code....

Enter your age : 20
Eligible to vote
Rest of the code....

Enter your age : 8
Not Eligible to vote
Rest of the code....
```

Result

Thus the program for validating voters age and handling the exception executed successfully.

Ex. No. 10(c)	**Exception Handling – Student mark validity**
Date :	

Aim

To implement program in python for validating student mark and handling the exception.

Algorithm

1. Start
2. Read mark of the student
3. If mark>0 and mark<100 print mark otherwise raise NameError and print Invalid mark
4. If other than number given , exception ValueError executed and shows Input must be a number
5. If an I/O error occurs , IOError exception executed and shows IOError
6. Stop.

```
#Program to check the validity of the student mark entry
def checkmark():
    try:
        mark=int(input("Enter student mark : "))
        if mark>0 and mark<=100:
            print("Marks Entered is ",mark)
        else:
            raise NameError("Invalid mark ")
    except ValueError:
           print("Input must be number")
    except IOError:
           print("IO Error")

    print("Rest of the code ....")
checkmark()
```

Output

```
Enter student mark : ww
Input must be number
Traceback (most recent call last):
  File "C:/Users/HP/AppData/Local/Programs/Python/Python310/stu.py", line 15, in
<module>
    checkmark()
  File "C:/Users/HP/AppData/Local/Programs/Python/Python310/stu.py", line 13, in
checkmark
    raise ValueError("Invalid mark ")
ValueError: Invalid mark

Enter student mark : 54
Marks Entered is 54
Rest of the code....

Enter student mark : -4
Traceback (most recent call last):
  File "C:/Users/HP/AppData/Local/Programs/Python/Python310/stu.py", line 17, in
<module>
    checkmark()
  File "C:/Users/HP/AppData/Local/Programs/Python/Python310/stu.py", line 9, in
checkmark
    raise NameError("Invalid mark ")
NameError: Invalid mark
```

Result

Thus the program for validating student mark and handling the exception executed successfully.

Ex. No. 11	**Exploring pygame tool**
Date :	

Aim

To implement program in python for exploring pygame tool to print text in desired size and font

Algorithm

1. Start
2. Set the display mode using setmode
3. Give the font size and color of the text to display
4. Make the background color as white
5. Set the x , y position of the text using blit() and display the text
6. Stop.

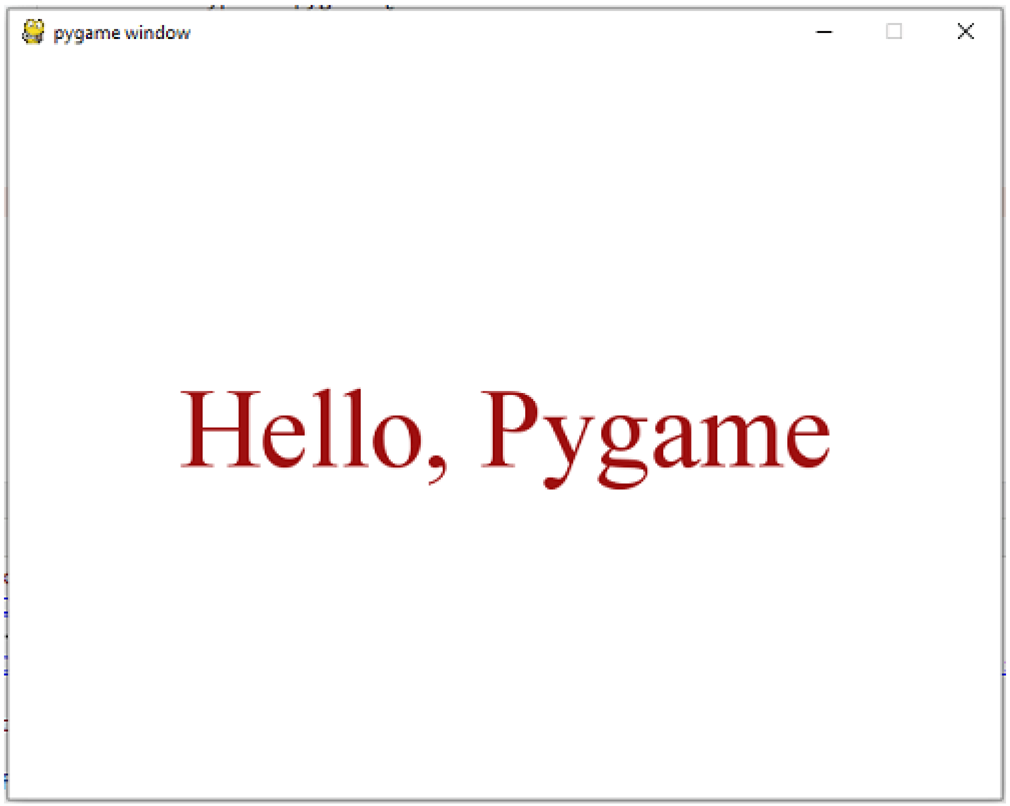
#Program to print text in desired size using pygame tool

```
import pygame
pygame.init()
screen = pygame.display.set_mode((640, 480))
done = False

#load the fonts
font = pygame.font.SysFont("Times new Roman", 72)
text = font.render("Hello, Pygame", True, (158, 16, 16))
while not done:
        for event in pygame.event.get():
            if event.type == pygame.QUIT:
                done = True
            if event.type == pygame.KEYDOWN and event.key == pygame.K_ESCAPE:
                done = True
screen.fill((255, 255, 255))
screen.blit(text,(320 - text.get_width() // 2, 240 - text.get_height() // 2))

pygame.display.flip()
```

Output



```
Hello, Pygame
```

Result

Thus the program for printing the text in desired size and color using pygame tool executed successfully.

Ex. No. 12	**Bouncing Ball game**
Date :	

Aim

To implement program in python for bouncing a ball

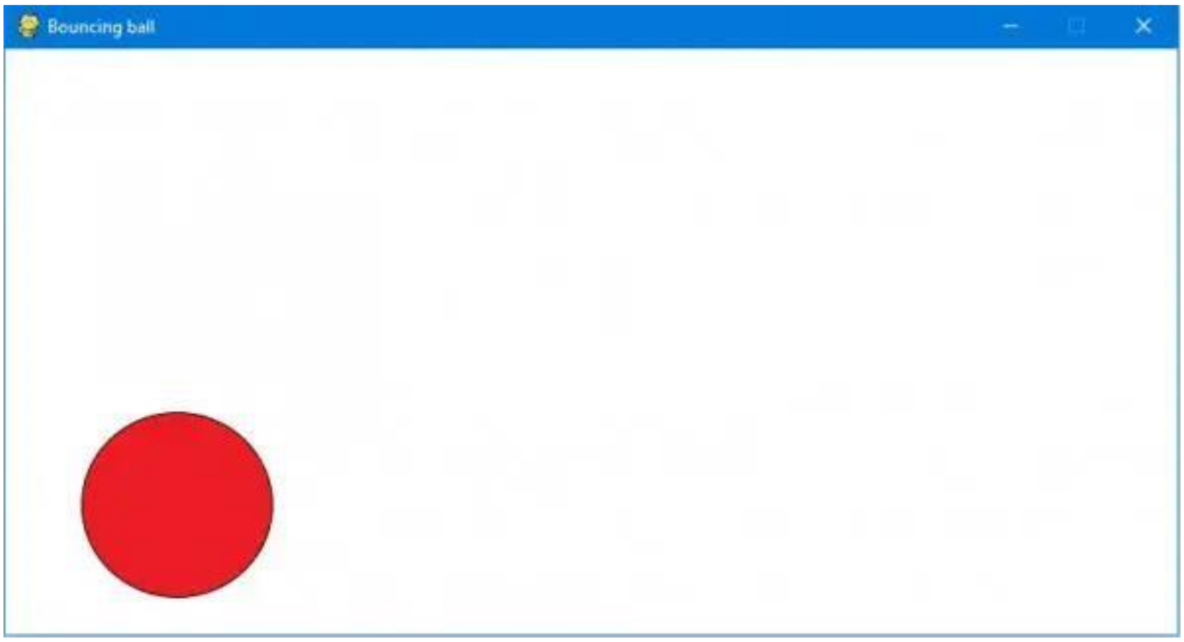
Algorithm

1. Start
2. Set the width and height of the window
3. Make the background color as white
4. Set the caption as Bouncing Ball
5. Select the ball image and make it display on screen
6. Check the endpoints of the window left,right,up and down and make the ball to reset position
7. Stop.

#Program to bounce a ball

```
import sys, pygame
pygame.init()
size = width, height = 800, 400
speed = [1, 1]
background = 255, 255, 255
screen = pygame.display.set_mode(size)
pygame.display.set_caption("Bouncing ball")
ball = pygame.image.load("ball.png")
ballrect = ball.get_rect()
while 1:
    for event  in pygame.event.get():
        if event.type == pygame.QUIT:
            sys.exit()
    ballrect = ballrect.move(speed)
    if ballrect.left < 0 or ballrect.right > width:
        speed[0] = -speed[0]
    if ballrect.top < 0 or ballrect.bottom > height:
        speed[1] = -speed[1]
    screen.fill(background)
    screen.blit(ball, ballrect)
    pygame.display.flip()
```

Output



Result

Thus the program for bouncing ball using pygame tool executed successfully.