1. IMPLEMENTATION OF SIMPLE CLASS PROGRAM

Aim :

To Write a Python program to implement the simple class program using inheritance.

Algorithm :

```
Step 1 : Start
Step 2 :  Create a class
Step 3 :  Create a built-in-function with reference parameter and get the no.of sides
for the polygon
Step 4 :  Create a user-define function to print the sides
Step 5 :  Create a derived class
Step 6 :  Print
Step 7 : Stop.
```

Program :

```
class polygon :

    def __init__(self,no_of_sides) :
        self.n=no_of_sides
        self.sides=[0 for i in range(no_of_sides)]

    def inputside(self) :
        self.sides=[float(input("Enter side"+str(i+1)+" : "))for i in range(self.n)]

    def dispsides(self) :
        for i in range(self.n) :
            print("Side",i+1,"is",self.sides[i])


class triangle(polygon) :

    def __init__(self) :
        polygon.__init__(self,3)

    def findarea(self) :
        a, b, c = self.sides
        s = (a + b + c) / 2
        area = (s * (s - a) * (s - b) * (s - c)) ** 0.5
        print(f"The area of the triangle is  :  {area}")

t=triangle()
t.inputside()
t.dispsides()
t.findarea()
```

Output :

Result :

Thus the Python class program was implemented using inheritance concept were executed successfully.

----

2. IMPLEMENTATION OF RECURSION ALGORITHM

Aim :

To Write a Python program
a) To find the fibonacci series using recursion.
b) To find the factorial of number using recursion.

a)To find the fibonacci series using recursion.

a)Algorithm :

```
Step 1 :  Start
Step 2 :  Define a function
Step 3 :  Check the number is less than or equal to 1
Step 4 :  If it is equal to 1 return it
Step 5 :  Else use the formula (recur_fibo(n1)+recur_fibo(n-2))
Step 6 :  Given the number of terms to be print
Step 7 :  Print
Step 8 :  Stop.
```

Program :

```
def recur_fibo(n) :
   if n<=1 :
      return n
   else :
      return(recur_fibo(n-1)+recur_fibo(n-2))

nterms=10
if nterms<=0 :
   print("Plese enter a positive integer")

else :
   print("Fibonacci sequence  : ")

for i in range(nterms) :
   print(recur_fibo(i))
```

Output :

```
Fibonacci sequence  :
0
1
```

```
1
2
3
5
8
13
21
34
```

b)Algorithm :

```
b)To find the factorial of number using recursion
Step1 : Start
Step2 : Define a function
Step3 :  Check the number is equal to 1
Step4 : If it is equal to 1
Step5 : Else use the formula(X*factorial(X-1))
Step6 :  Give the number to be factorial
Step7 : Print
Step8 : Stop.
```

Program :

```
def factorial(x) :
    """This is a recursive function to find the factorial of an integer"""
    if x == 1 :
        return 1
    else :
        return (x * factorial(x - 1))

num = 3
print("The factorial of", num, "is", factorial(num))
```

Output :

```
The factorial of 3 is 6
```

Result :

Thus the recursion algorithm was implemented using Python program were executed successfully.

----

3. IMPLEMENT THE PYTHON ARRAY USING LIST

Aim :

To write a Python program for implement the array to insert,delete and traverse the data element .

Algorithm :

```
Step 1 :  Start
Step 2 :  Import array library
Step 3 :  Print the array values and then accessing the array element from index 2
Step 4 :  Insert an element in the array of index 3 & 4 (400,150) respectively and then print the inserted element
Step 5 :  Delete an element from the array (150) and then print
Step 6 :  Traversing the elements from the array and then print
Step 7 :  Stop.
```

Program :

```
# IMPLEMENT THE PYTHON ARRAY USING LIST

import array

# Create an array of integers
balance = array.array('i', [100, 200, 300])

# Print the array
print("The array with given values : ")
print(balance)

# Accessing an array element at index 2
print("Accessing an array element from index[2] : ")
print(balance[2])

# Inserting elements into the array
balance.insert(3, 400)
balance.insert(4, 150)

# Print the array after insertion
print("After Insertion : ")
print(balance)

# Accessing the index value of an element (e.g., 400)
print("Accessing the index value of an element (e.g., 400) : ")
print(balance.index(400))

# Deleting an element from the array (e.g., 150)
balance.remove(150)

# Print the array after deletion
print("After Deleting 150 : ")
print(balance)

# Traverse the array and print its elements
print("Traverse the array : ")
for x in balance :
    print("Array Element : ", x)
```

Output :

Result :

Thus the Python program executed successfully.

----

4. IMPLEMENTATION OF LINKED LIST

Aim :

To write a python program to implement the linked list.

Algorithm :

```
Step 1 :  Start .
Step 2 : Creating a class as Node.
Step 3 :  Again creating a class as Linked List.
Step 4  :  Defining a insert function.
Step 5 :  Defining printLL .
Step 6 :  Print the data element in the linked list.
Step 7 :  Creating an object for the class Linked List .
Step 8 :  With the help of the object calling the insert function to insert the element .
Step 9 :  Print the data element.
```

Program :

```
# IMPLEMENTATION OF LINKED LIST
class Node() :
    def __init__(self, data=None, next=None) :
        self.data = data
        self.next = next

class LinkedList() :
    def __init__(self) :
        self.head = None

    def insert(self, data) :
        new_node = Node(data)
        if self.head :
            current = self.head
            while current.next :
                current = current.next
            current.next = new_node
        else :
            self.head = new_node

    def printLL(self) :
        print("The data elements in the linked list are : ")
        current = self.head
        while current :
            print(current.data)
            current = current.next



# Create a linked list and insert elements
LL = LinkedList()
LL.insert(1)
LL.insert(2)
LL.insert(3)
LL.insert(4)

# Print the linked list
LL.printLL()
```

Output :

Result :

Thus the Python program executed successfully.

----

5. IMPLEMENTATION OF STACK ADT

Aim :

To write a Python program to implement of stack ADT using list.

Algorithm :

Step 1 : Start.

Step 2 : Creating a class as Node

Step 3 : Again creating a class as Stack

Step 4 : Defining push function

Step 5 : Defining pop function

Step 6 : Defining traverse function

Step 7 : Define Menu function

Step 8 : Print

Step 9 : Creating an object for the class Stack

Step 10 : Print the stack

Step 11 : Get the input from the user for choice and then print

Step 12 : Stop.

Program:

Output:

Result:

Thus the above Python program executed successfully.
