MCKV INSTITUTE OF ENGINEERING

Computer Science and Engineering – Data Science

***Name - Abhinaba Sarkar***

***Roll No. - BTECH/CSE-DS/2020/47***

**Assignment Number:** 9

**Assignment Number: 09(A)**

**Problem Statement:**

Write a program to create a class Circle. Perform the following operations on it.

i) Define the attribute/variable radius.

ii) Define the constructor with one argument containing radius.

iii) Define the method named get\_radius() which returns the radius of the circle.

iv) Define the method named calc\_area() which return the area of the circl

**Source Code:**

*class Circle():*

*def \_\_init\_\_(self,r=None):*

*if r is None:*

*self.radius=eval(input('enter the radius:'))*

*else:*

*self.radius=r*

*print("A circle has been created")*

*def get\_radius(self):*

*return self.radius*

*def calc\_area(self):*

*return 3.14\*self.radius\*\*2*

*def \_\_del\_\_(self):*

*print("Object has been destructed")*

*class Cylinder(Circle):*

*def \_\_init\_\_(self,h=None,r=None):*

*if h is None:*

*self.height=eval(input('enter the height of the cylinder:'))*

*else:*

*self.height=h*

*if r is None:*

*super().\_\_init\_\_()*

*else:*

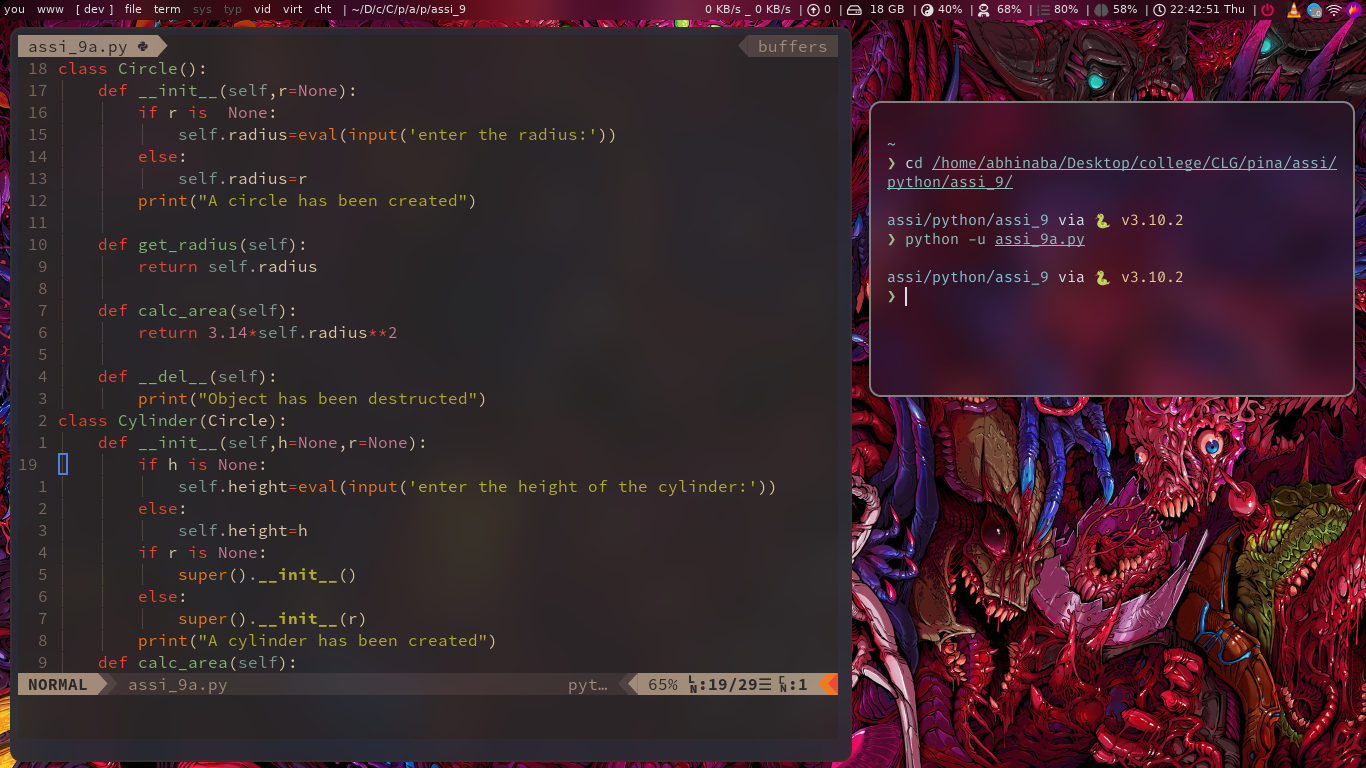
*super().\_\_init\_\_(r)*

*print("A cylinder has been created")*

*def calc\_area(self):*

*return 2\*3.14\*self.radius\*(self.radius+self.height)*

**Output :**



**Assignment Number: 09(B)**

**Problem Statement :**

Write a program to implement single inheritance

i) Create the parent class Circle. Initialize the constructor with the radius of the circle.

ii) Define the method get\_radius() and calc\_area() to know the radius and area of the circle.

iii) Create the child class named Cylinder. Initialize the value of the height within the constructor and

call the constructor of the parent class to initialize the radius of the cylinder.

iv) Finally define the method calc\_area() in class cylinder to calculate the area of cylinder

(2\*pi\*r\*(h+r)).

**Source Code :**

*class Circle():*

*def \_\_init\_\_(self,r = None):*

*if r is None:*

*self.radius = eval(input('Input the radius'))*

*else:*

*self.radius = r*

*print("A Circle has been created")*

*def get\_radius(self):*

*return self.radius*

*def calc\_area(self):*

*return 3.14\*self.radius\*\*2*

*def \_\_del\_\_(self):*

*print("Object has been Destructed")*

**Output :**



***Assingnment No.: 09(C)***

**Problem Statement :**

Write a program to implement the concept of multilevel inheritance

i) Create the parent class StLine. Initialize the constructor with length the straight line.

ii) Create another class named Square which inherits the properties of parent class StLine. Initialize

the side of the square by using the constructor of the parent class. Define the method calc\_area() toreturn the area of the Square.

iii) Create another class named Cube which inherits the properties of parent class Square. Define that tributes height of Cube class. Also call the constructor of the parent class for the edges of the

cube. Define the method calc\_area() to return the area of the Cube.

iv) Finally, create the instances of StLine, Square, and Cube classes to return the area of the of the

**respective classes.**

*Source Code : class StLine():*

*def \_\_init\_\_(self,l=None):*

*if l is None:*

*self.length=eval(input('enter the length:'))*

*else:*

*self.length=l*

*print("A straight line has been created")*

*def get\_length(self):*

*return self.length*

*class Square(StLine):*

*def \_\_init\_\_(self,s=None):*

*if s is None:*

*super().\_\_init\_\_()*

*else:*

*super().\_\_init\_\_(s)*

*print("A Square has been created")*

*def get\_side(self):*

*return self.length*

*def calc\_area(self):*

*return self.length\*\*2*

*class Cube(Square):*

*def \_\_init\_\_(self,h=None):*

*if h is None:*

*self.height=eval(input('enter the height of the cube:'))*

*else:*

*self.height=h*

*print("A Cube has been created")*

*def get\_height(self):*

*return self.height*

*def calc\_area(self):*

*return 6\*self.height\*\*2*

*st=StLine()*

*print(st.get\_length())*

*sq=Square()*

*print('Square has been created length:',sq.get\_side())*

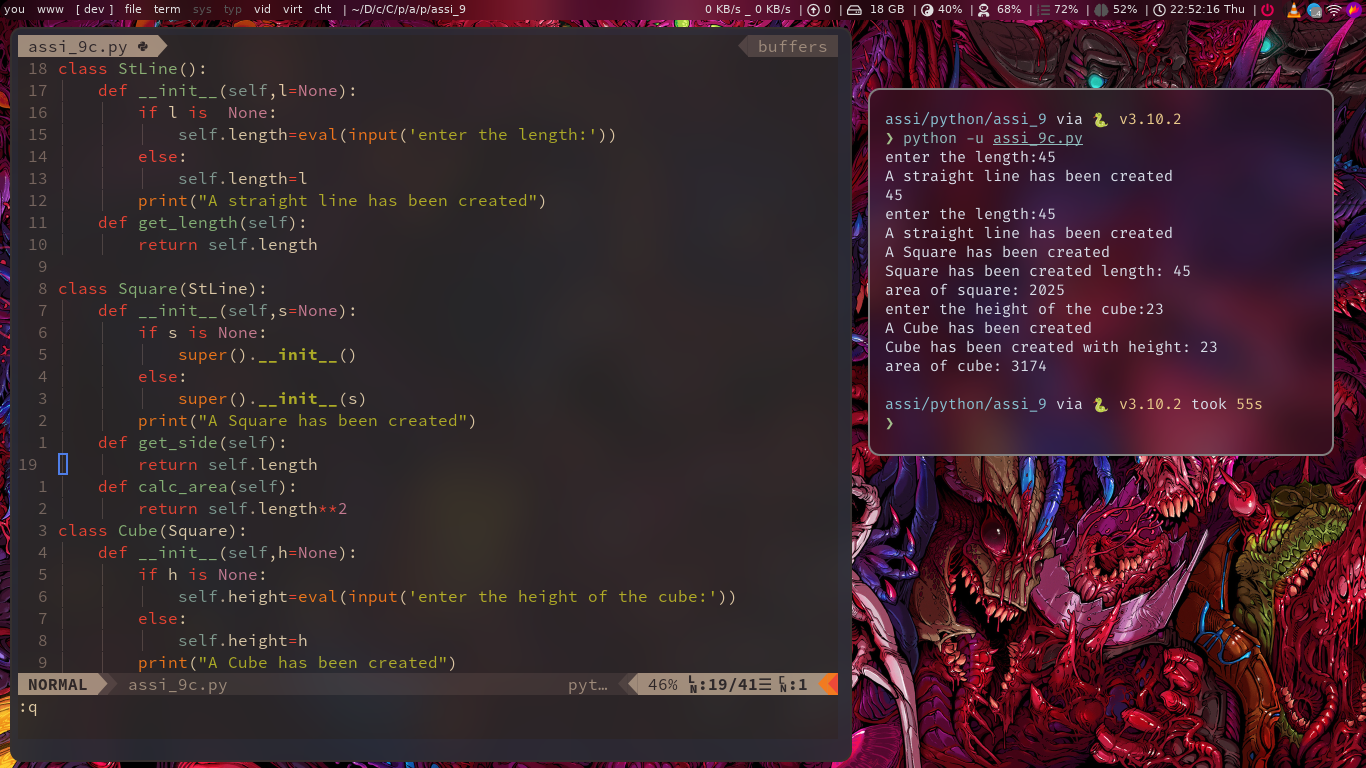
*print('area of square:',sq.calc\_area())*

*cu=Cube()*

*print('Cube has been created with height:',cu.get\_height())*

*print('area of cube:',cu.calc\_area())*

**Output :**



**Assingnment No.: 09(D)**

**Problem Statement :**

Write a program to implement the concept multiple inheritance

i) Define a class SqrArea. With in this class define a method cal\_area() to calculate the area of a

square.

ii) Define a class SqrPeri. With in this class define a method cal\_peri() to calculate the peripheral of asquare.

iii) Define a class Sqr for square by inheriting the SqrArea and SqrPeri class. Initialize the constructor

with length the side. Define a method show() to show the area and the peripheral of the square.

**Source Code :**

*class SqrArea():*

*def calc\_area(self,l):*

*return l\*l*

*def \_\_del\_\_(self):*

*print('Square has been destructed')*

*class SqrPeri():*

*def calc\_peri(self,l):*

*return 4\*l*

*def \_\_del\_\_(self):*

*print('Squareperi has been destructed')*

*class Sqr(SqrArea,SqrPeri):*

*def \_\_init\_\_(self,s=None):*

*if s is None:*

*self.length=eval(input('enter the length of the Square:'))*

*else:*

*self.length=s*

*def show(self):*

*print('area of the square:',super().calc\_area(self.length))*

*print('peripheral of the Square:',super().calc\_peri(self.length))*

*sq=Sqr()*

*print(sq.show())*

**Output :**



**Assignment No.: 09(E)**

**Problem Statement :**

Write a program to implement the concept of module and operator overloading

i) Define a module name powerset, which consist of the PowerSet.

ii) In PowerSet class, overload the +, \*, and - to perform Set Union, Set Intersection and Set

difference respectively.

iii) Import this module to perform the Set Union, Set Intersection and Set difference on Sets.

**Source Code :**

*class Powerset():*

*def \_init\_(self,s):*

*if type(s) == set:*

*self.set1 = s*

*else:*

*print('Argument is not a set')*

*def \_add\_(self,other):*

*print('The 1st set is: ', self.set1)*

*print('The 2nd set is: ', other.set1)*

*print('The reuslt of union is: ',self.set1.union(other.set1))*

*def \_mul\_(self,other):*

*print('The 1st set is: ',self.set1)*

*print('The 2nd set is: ', other.set1)*

*print('The reuslt of intersection is: ',self.set1.intersection(other.set1))*

*def \_sub\_(self,other):*

*print('The 1st set is: ',self.set1)*

*print('The 2nd set is: ', other.set1)*

*print('The reuslt of difference is: ',self.set1.difference(other.set1))*

*print('The reuslt of difference is: ',other.set1.difference(self.set1))*

**Output :**

