import math

# Creating a super class called Shapes. This is an abstract class

class Shapes:

pass #pass means I am not doing anything in that class

# Creating subclass called Square inheriting from Shapes. See Shapes is in brackets to indicate inheritance

class Square(Shapes):

# This is a constructor which is used to initialise fields (variables)

def \_\_init\_\_(self, side):

self.side1 = side

# This is a method to calculate area

def getArea1(self):

answer = self.side1 \* self.side1

return answer

# Side1 is a field or variable

side1 = int(input("please enter side of a square\n"))

# objRectangle is the object from the class Square

objRectangle = Square(side1)

#Printinting value of an attribute side1

print("The attribute" , objRectangle.side1)

# Printing my results & call the method getArea1 using the object

print("Area of a square is", objRectangle.getArea1(), "cm2")

# Creating subclass called Square inheriting from Shapes. See Shapes is in brackets to indicate inheritance

class Circle(Shapes):

# This is a constructor which is used to initialise fields (variables)

def \_\_init\_\_(self, radius):

self.rad = radius

# This is a method to calculate area of a circle

def getAreaCircle(self):

answer1 = self.rad \* self.rad \* math.pi

return answer1

# This is a method to calculate area of a circle

def getPerimeterCircle(self):

answer1 = 2\* math.pi \* self.rad

return answer1

# Side1 is a field or variable

rad = float(input("please enter radius of a circle\n"))

# objCirle is the object from the class Square

objCircle = Circle(rad)

# Printing my results & calling the method getAreaCircle using the object objCircle

print("Area of a circle whose radius is", rad, "cm is", round(objCircle.getAreaCircle(), 2), "cm2")

# Printing my results & calling the method getPerimeterCircle using the object objCircle

print("The perimeter of a circle whose radius is" , rad, "cm is", round(objCircle.getPerimeterCircle(), 2), "cm2" )

#How to show the subclassess found in a super class. This is another way of demonstarting inheritance

print(Shapes.\_\_subclasses\_\_())

Tomorrow we will post the solution for today and work on OOP again. good night