Q1class Animal:

def makeSound(self):

print("This animal makes a sound.")

class Dog(Animal):

def makeSound(self):

print("Dog barks!")

class Cat(Animal):

def makeSound(self):

print("Cat meows!")

class Cow(Animal):

def makeSound(self):

print("Cow moos!")

# Testing the classes

if \_\_name\_\_ == "\_\_main\_\_":

# Create instances of each class and call makeSound() method

animal = Animal()

animal.makeSound() # Output: This animal makes a sound.

dog = Dog()

dog.makeSound() # Output: Dog barks!

cat = Cat()

cat.makeSound() # Output: Cat meows!

cow = Cow()

cow.makeSound() # Output: Cow moos!

Q2.from abc import ABC, abstractmethod

import math

class Shape(ABC):

@abstractmethod

def calculateArea(self):

pass

class Rectangle(Shape):

def \_\_init\_\_(self, length, width):

self.length = length

self.width = width

def calculateArea(self):

return self.length \* self.width

class Circle(Shape):

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

self.radius = radius

def calculateArea(self):

return math.pi \* self.radius\*\*2

class Triangle(Shape):

def \_\_init\_\_(self, base, height):

self.base = base

self.height = height

def calculateArea(self):

return 0.5 \* self.base \* self.height

class ShapeCalculator:

def printArea(self, shape):

if not isinstance(shape, Shape):

raise ValueError("Invalid shape instance.")

area = shape.calculateArea()

print(f"The area of the shape is: {area:.2f}")

# Testing the classes

if \_\_name\_\_ == "\_\_main\_\_":

shape\_calculator = ShapeCalculator()

# Create instances of each class and pass them to printArea() method

rectangle = Rectangle(5, 10)

shape\_calculator.printArea(rectangle) # Output: The area of the shape is: 50.00

circle = Circle(7)

shape\_calculator.printArea(circle) # Output: The area of the shape is: 153.94

triangle = Triangle(6, 8)

shape\_calculator.printArea(triangle) # Output: The area of the shape is: 24.00

Q3.class Person:

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

self.\_name = ""

self.\_age = 0

self.\_address = ""

def get\_name(self):

return self.\_name

def set\_name(self, name):

self.\_name = name

def get\_age(self):

return self.\_age

def set\_age(self, age):

self.\_age = age

def get\_address(self):

return self.\_address

def set\_address(self, address):

self.\_address = address

# Testing the class

if \_\_name\_\_ == "\_\_main\_\_":

person = Person()

# Set values using the setter methods

person.set\_name("John Doe")

person.set\_age(30)

person.set\_address("123 Main Street, City")

# Display values using the getter methods

print("Name:", person.get\_name()) # Output: Name: John Doe

print("Age:", person.get\_age()) # Output: Age: 30

print("Address:", person.get\_address()) # Output: Address: 123 Main Street, City

Q4.from abc import ABC, abstractmethod

import math

class Drawable(ABC):

@abstractmethod

def draw(self):

pass

class Circle(Drawable):

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

self.radius = radius

def draw(self):

print(f"Drawing a circle with radius {self.radius}")

class Rectangle(Drawable):

def \_\_init\_\_(self, length, width):

self.length = length

self.width = width

def draw(self):

print(f"Drawing a rectangle with length {self.length} and width {self.width}")

# Testing the classes

if \_\_name\_\_ == "\_\_main\_\_":

circle = Circle(5)

circle.draw() # Output: Drawing a circle with radius 5

rectangle = Rectangle(4, 8)

rectangle.draw() # Output: Drawing a rectangle with length 4 and width 8