**Topic: Inheritance**

**Question 1:** *Create a class Vehicle with attributes make and model. Then, create a subclass Car that inherits from Vehicle and has an additional attribute year. Implement a method display\_info to print the car's details.*

**Example:**

car = Car("Toyota", "Camry", 2022)

car.display\_info()

**Sample Input:**

make = "Toyota"

model = "Camry"

year = 2022

**Sample Output:**

"Make: Toyota, Model: Camry, Year: 2022"

**Question 2:** *Create a base class Shape with a method area() and then create two subclasses, Rectangle and Circle, that inherit from Shape. Implement the area method in each subclass to calculate their respective areas.*

**Example:**

rectangle = Rectangle(5, 4)

circle = Circle(3)

print(rectangle.area())

print(circle.area())

**Sample Input:**

For Rectangle: Length = 5, Width = 4

For Circle: Radius = 3

**Sample Output:**

Rectangle Area: 20

Circle Area: 28.27 (rounded to two decimal places)

**Question 3:** *Create a class Person with attributes name and age. Then, create a subclass Student that inherits from Person and has an additional attribute student\_id. Implement a method to display the student's details.*

**Example:**

student = Student("Alice", 20, "S12345")

student.display\_details()

**Sample Input:**

name = "Alice"

age = 20

student\_id = "S12345"

**Sample Output:**

"Name: Alice, Age: 20, Student ID: S12345"

**Question 4*:*** *Create a class Animal with attributes name and species. Then, create a subclass Dog that inherits from Animal and has an additional attribute breed. Implement a method to display the dog's details.*

**Example:**

dog = Dog("Buddy", "Canine", "Golden Retriever")

dog.display\_info()

**Sample Input:**

name = "Buddy"

species = "Canine"

breed = "Golden Retriever"

**Sample Output:**

"Name: Buddy, Species: Canine, Breed: Golden Retriever"

**Question 5:** *Create a class BankAccount with attributes account\_number and balance. Then, create a subclass SavingsAccount that inherits from BankAccount and has a method calculate\_interest to compute the interest earned on the balance.*

**Example:**

savings = SavingsAccount("123456", 1000)

interest = savings.calculate\_interest(0.05)

print(interest)

**Sample Input:**

account\_number = "123456"

balance = 1000

interest\_rate = 0.05

**Sample Output:**

Interest Earned: 50.0

**Topic: Polymorphism**

**Question 6:** *Create a class Shape with a method area(). Then, create subclasses Rectangle and Circle that override the area() method to calculate their respective areas. Write a function that calculates the total area of a list of shapes, regardless of their specific types.*

**Example:**

shapes = [Rectangle(5, 4), Circle(3), Rectangle(2, 3)]

total\_area = calculate\_total\_area(shapes)

print(total\_area)

**Sample Input:**

Shapes List: [Rectangle(5, 4), Circle(3), Rectangle(2, 3)]

**Sample Output:**

Total Area: 38.27 (rounded to two decimal places)

**Question 7:** *Create a class Animal with a method speak(). Then, create subclasses Dog and Cat that override the speak() method to produce their respective sounds. Write a function that takes a list of animals and makes them speak.*

**Example:**

animals = [Dog("Buddy"), Cat("Whiskers"), Dog("Max")]

make\_animals\_speak(animals)

**Sample Input:**

Animals List: [Dog("Buddy"), Cat("Whiskers"), Dog("Max")]

**Sample Output:**

"Buddy says Woof!"

"Whiskers says Meow!"

"Max says Woof!"

**Question 8:** *Create a class Person with a method greet(). Then, create subclasses Teacher and Student that override the greet() method to provide their own greetings. Write a function that greets a list of people.*

**Example:**

people = [Teacher("Mr. Smith"), Student("Alice"), Teacher("Ms. Johnson")]

greet\_people(people)

**Sample Input:**

People List: [Teacher("Mr. Smith"), Student("Alice"), Teacher("Ms. Johnson")]

**Sample Output:**

"Mr. Smith says Hello, I'm a teacher."

"Alice says Hi, I'm a student."

"Ms. Johnson says Hello, I'm a teacher."

**Question 9:** *Create a class Shape with a method description(). Then, create subclasses Rectangle and Circle that override the description() method to provide a description of the shape. Write a function that generates descriptions for a list of shapes.*

**Example:**

shapes = [Rectangle(5, 4), Circle(3), Rectangle(2, 3)]

describe\_shapes(shapes)

**Sample Input:**

Shapes List: [Rectangle(5, 4), Circle(3), Rectangle(2, 3)]

**Sample Output:**

"Rectangle with length 5 and width 4."

"Circle with radius 3."

"Rectangle with length 2 and width 3."

**Question 10:** *Create a class Vehicle with a method fuel\_efficiency(). Then, create subclasses Car and Motorcycle that override the fuel\_efficiency() method to provide the fuel efficiency of each vehicle type in miles per gallon (MPG). Write a function that calculates the total fuel consumption for a list of vehicles.*

**Example:**

vehicles = [Car("Sedan"), Motorcycle("Sportbike"), Car("SUV")]

total\_fuel\_consumption = calculate\_total\_fuel\_consumption(vehicles, 100)

print(total\_fuel\_consumption)

**Sample Input:**

Vehicles List: [Car("Sedan"), Motorcycle("Sportbike"), Car("SUV")]

Distance Traveled: 100 miles

**Sample Output:**

Total Fuel Consumption: 10.5 gallons