1 Python程序设计#1作业

班级: 305

学号: 2022211683

姓名: 张晨阳

1.1 作业题目

每人独立设计并实现一个小型python程序（功能不限），代码需要涉及：class类、对象实例化、继承、对象方法(self参数)、类方法(@classmethod)、静态方法(@staticmethod)、对象属性、类属性、多态。

1.2 作业内容

程序源代码嵌入下方的code block中。

```
class Vehicle:
    class_attribute = "This is a vehicle class"

    def __init__(self, brand, model, year):
        self.brand = brand
        self.model = model
        self.year = year
        self.mileage = 0

    def description(self):
        return f"This vehicle is a {self.year} {self.brand} {self.model} with
{self.mileage} miles."

    # 类方法
    @classmethod
    def show_class_info(cls):
        return f"Class Info: {cls.class_attribute}"

    # 静态方法
    @staticmethod
    def vehicle_sound():
        return "Vehicles can make sounds."

    # 对象方法，更新里程数
    def update_mileage(self, miles):
        if miles >= self.mileage:
            self.mileage = miles
        else:
            print("Mileage cannot be rolled back.")

# Car类继承Vehicle类
class Car(Vehicle):
    def __init__(self, brand, model, year, num_doors):
```

```
        super().__init__(brand, model, year)
        self.num_doors = num_doors
        self.fuel_level = 100

    def description(self):
        return f"This car is a {self.year} {self.brand} {self.model} with
{self.num_doors} doors and {self.mileage} miles."

    # 对象方法，描述燃油状态
    def fuel_status(self):
        return f"The car has {self.fuel_level}% fuel remaining."

    # 对象方法，加油
    def refuel(self, amount):
        if 0 <= self.fuel_level + amount <= 100:
            self.fuel_level += amount
        else:
            print("Fuel level must be between 0 and 100.")

# Motorcycle类继承Vehicle类
class Motorcycle(Vehicle):
    def __init__(self, brand, model, year, engine_type):
        super().__init__(brand, model, year)
        self.engine_type = engine_type
        self.helmet_on = False

    def description(self):
        return f"This motorcycle is a {self.year} {self.brand} {self.model} with a
{self.engine_type} engine and {self.mileage} miles."

    def wear_helmet(self):
        self.helmet_on = True
        return "Helmet is on."

    def check_helmet(self):
        return "Helmet is on." if self.helmet_on else "Helmet is off."

if __name__ == '__main__':
    car = Car("Toyota", "Camry", 2021, 4)
    motorcycle = Motorcycle("Yamaha", "MT-07", 2022, "parallel-twin")

    print(car.description())
    print(motorcycle.description())

    # 更新车辆里程数
    car.update_mileage(5000)
    motorcycle.update_mileage(3000)
    print(car.description())
    print(motorcycle.description())

    print(Vehicle.show_class_info())
    print(Vehicle.vehicle_sound())

    # Car类的特有方法演示
```

```
    print(car.fuel_status())
    car.refuel(-10)
    print(car.fuel_status())

    # Motorcycle类的特有方法演示
    print(motorcycle.check_helmet())
    print(motorcycle.wear_helmet())
    print(motorcycle.check_helmet())
```

1.3 代码说明

1. **类的定义**:

   代码中有三个类：`Vehicle` (父类)，`Car` 和 `Motorcycle` (子类)。`Vehicle` 是一个通用的车辆类，包含了基本的车辆信息和方法。`Car` 和 `Motorcycle` 继承了 `Vehicle`，并且各自增加了一些特有的属性和方法。

   - `Vehicle` 类具有一个类属性 `class_attribute`，它可以被所有实例共享。
   - `Car` 类和 `Motorcycle` 类通过继承 `Vehicle` 类，重用了父类中的代码。

2. **对象属性**:

   `Vehicle` 类的构造方法 `__init__()` 初始化了 `brand`、`model` 和 `year` 作为对象属性，每个实例都有自己独立的品牌、型号和年份信息。

   `Car` 类添加了 `num_doors` 属性，用于描述车门的数量，并添加了 `fuel_level` 属性，用于描述燃油水平。

   `Motorcycle` 类添加了 `engine_type` 属性，用于描述发动机类型，并增加了 `helmet_on` 属性，用于描述头盔的状态。

3. **对象方法**:

   `Vehicle` 类中定义了一个对象方法 `description()`，用于返回车辆的描述信息。这个方法在 `Car` 和 `Motorcycle` 子类中被重写，以提供更具体的描述。另外，`Vehicle` 类还有一个 `update_mileage()` 方法，用于更新车辆的里程数。

4. **类方法**:

   使用 `@classmethod` 装饰器定义了类方法 `show_class_info()`，它通过 `cls` 参数访问类属性，返回有关类的信息。类方法可以通过类名直接调用，也可以通过实例调用。

5. **静态方法**:

   使用 `@staticmethod` 装饰器定义了静态方法 `vehicle_sound()`，它与任何特定实例无关，可以直接通过类名调用。

   静态方法通常用于定义逻辑上与类相关，但不需要访问类或实例的任何属性的方法。

6. **继承和多态**:

`Car` 和 `Motorcycle` 类继承了 `Vehicle` 类，并重写了 `description()` 方法，这实现了多态。通过调用 `description()` 方法，不同的子类可以返回不同的描述信息，表现出不同的行为。

7. **对象实例化和方法调用**:

   在 `if __name__ == '__main__':` 块中，实例化了 `Car` 和 `Motorcycle` 对象，并调用了它们的 `description()` 方法。

   还调用了 `Vehicle` 类的类方法和静态方法，演示了它们的使用方法。

8. **特有方法**:

   `Car` 类有一个 `fuel_status()` 方法，用于返回当前的燃油水平，并有一个 `refuel()` 方法，用于给车辆加油。`Motorcycle` 类有 `wear_helmet()` 和 `check_helmet()` 方法，用于管理和检查头盔的状态。