华 中 科 技 大 学

《Python 程序设计》论文报告

姓名: 吴志渊

学号: U202010603

班级: 机卓 2001

日期: 2023.11.14

Python 面向对象编程：学校成员管理系统的设计与实现

一. 代码功能介绍

1.大作业选题

本次大作业选题为第三题——面向对象编程内容，具体要求如下：

编写学校成员类，具有成员姓名、年龄和总人数属性，教师类继承学校成员类，具有工号和工资属性，学生类继承学校成员类，具有学号和成绩属性。并实现如下功能：（1）创建对象时，总人数加 1，对象注销时，总人数减 1；（2）输出学生的平均成绩。

具体的代码与实现效果见附录。

2.编程思路

大作业的主要目标是创建一个学校成员管理系统，该系统能够跟踪和管理学校的教师和学生。为了实现这个目标，代码采用了面向对象的编程方法。

首先根据作业要求，可以分析得到需要三种类便可以完成既定的任务。基于这些分析，可以梳理得到三个类的基本结构：

- 名为 `SchoolMember` 的基类，它包含了所有学校成员共有的属性和方法。这个类有三个属性：`name`（姓名）、`age`（年龄）和 `num_total`（总人数）。其中，`name` 和 `age` 是实例属性，每个 `SchoolMember` 对象都有自己的 `name` 和 `age`；`num_total` 是类属性，所有的 `SchoolMember` 对象共享同一个 `num_total`。
- 名为 `Teacher` 的类是继承自 `SchoolMember` 的子类，有两个额外的属性：`id`（工号）和 `salary`（工资）；

- 名为 `Student` 的类也是继承自 `SchoolMember` 的子类，有两个额外的属性：`id`（学号）和 `grade`（成绩）。

在这三个类的定义中，使用了类方法和静态方法。类方法是绑定到类的方法，它的第一个参数是类本身（通常命名为 `cls`）；静态方法则既不绑定到类也不绑定到实例。在这份代码中，类方法主要用来修改类属性，静态方法主要用来打印人员状态情况。

为了尽可能的提高使用时的封装程度，我将所有的与状态打印相关的打印指令封装在了方法 `SchoolMember.showMember()`中与子类的构造函数与析构函数中。

进一步的，为了实现在增加或者删除人员时总人数能够自动变化，我使用`@classmethod` 关键词修饰创建了增加人员与删除人员的函数，使得子类调用这两个函数时可以直接修改类 `SchoolMember` 中对应的人数。

但是在运行代码时我发现，由于将所有的与人员修改有关的状态打印值都封装进了子类的析构函数中，当程序执行完成后，由于系统内存自动释放，导致在代码中没有被“人为删除”的对象也会在程序结束运行时打印出数量状态的改变，这与实际使用的要求显然不符合。

为此我也尝试了多种办法来解决这个问题，包括自己封装一个`delete()`函数来在代码中调用时打印一些状态并使用类似`del self`的方式删除自身，但是后续查阅资料发现这样是不可行的，我也通过 debug 发现在类内部调用 del 是无法删除自身的。

最后，我选择了加入一个标志位来解决这个问题，也即在

SchoolMember 中人为加入了一个默认值为 true 的逻辑值，只有逻辑值为真时才打印输出当前学校的人数状态。因此，现在只需要在程序末尾加上 SchoolMember.printFlag = False 即可解决程序结束释放实例化的类时打印的问题。

最后，这份代码还使用了 gc 模块来获取所有已经实例化的对象。这是因为在 Python 中，当一个对象没有任何变量引用它时，垃圾收集器就会删除它。通过 gc.get_objects()，可以获取所有的对象，然后通过 isinstance 函数过滤出指定类型的对象。我将这样的功能思路封装进了 Student 对象中的一个函数，只要调用该函数即可获取当前已经实例化的所有指定类型的对象，也即 Student 对象，简单的使用列表遍历功能即可输出他们的平均成绩。

二．Python 的语言特点

1.可变与不可变对象

在 Python 编程语言中，有两种基本类型的对象：可变对象和不可变对象。通过之前上机的任务，我也对这两个关键概念有了更清晰的认识。

不可变对象是在程序运行过程中无法更改其状态的对象。像整数、字符串和元组这样的数据类型在 Python 中属于不可变对象。一旦创建了这样的对象，它们的值将不能再进行更改。例如，当创建一个字符串变量并将值分配给该变量时，您无法以任何形式修改原始字符串的值。如果试图这样做，Python 解释器会创建一个新的字符串而不是更改原始值。

相比之下，可变对象则允许我们在程序运行期间改变它们的状态。这包括列表、字典和集合这些数据类型。我们可以添加、删除或者修改这些对象的内容。比如，你可以往一个列表中加入新的元素，从一个字典中删除一个键值对，或者更新一个集合中的元素。这些操作都会直接影响到原来的对象，而不会生成新的对象。

在 Python 的使用过程中，我认为掌握可变对象和不可变对象的概念是非常基础且重要的。利用不可变对象时，便不必担心它们会被意外更改。在处理可变对象时，需要更加小心，以确保您的改动不会波及到其他部分的代码。

2.命名空间与作用域

在我学习 Python 基础课程的过程中，我对 Python 中的命名空间和作用域有了一定的认识。我发现这两者是 Python 编程中非常核心的组成部分，它们不仅决定着变量名的管理与查找方式，同时也为编写清晰、易读的代码提供了保障。

所谓的命名空间，其实就是一个用来储存变量名称的地方。每一个变量名都有着自己的唯一位置，这就是它的命名空间。我们可以将命名空间看作是一个庞大的字典，里面包含了所有的变量名及其对应的值。当我们需要使用一个变量时，就可以在这个“字典”里找到相应的变量名，并获取其值。

而作用域则是指命名空间的生效范围。在 Python 中，主要有两种类型的作用域，即局部作用域和全局作用域。局部作用域是指函数内部的命名空间，而全局作用域则是指整个程序文件的命名空间。当

我们在一个特定的作用域内定义一个变量时，只有在这个作用域内的代码才能访问到这个变量。

我自己就曾在编程过程中，因为不熟悉命名空间和作用域的概念，犯下过严重的错误。我在一个函数内部定义了一个与全局变量同名的局部变量，结果函数运行时，竟然改变了全局变量的值，使得程序的运行结果出现严重偏差。这件事让我深刻认识到，充分理解并正确使用命名空间和作用域，是十分重要的。

Python 中的命名空间和作用域为我们提供了良好的变量管理机制，让我们能够更清晰地组织代码，避免潜在的错误，提升代码的可读性和效率。因此，掌握好这两个概念，对于学习 Python 编程而言，具有至关重要的意义。

3.面向对象编程

我发现面向对象编程是一种强大而又灵活的设计方法论，可以帮助我们将复杂的代码结构化，提高代码的复用性。

在面向对象编程的思想中，一切皆对象。我们可以通过定义类来抽象出一组相关的属性和行为。类就像是现实世界中的模板或蓝图，对象则是基于这些模板或蓝图创建出来的具体实例。通过继承，子类可以从父类那里获得所有属性和方法，并根据自身的特殊需求进行扩展或覆盖。多态则可以让同一接口支持多种不同的实现方式。

通过面向对象编程，我可以更好地组织我的代码。当我面对一个复杂的任务时，我会先分析其中有哪些对象以及他们之间的关系，然后设计出相应的类以及他们的交互方式。这样不仅可以提高代码的可

读性，而且也方便我在后续开发中对其进行维护和扩展。

然而，面向对象编程也不是一蹴而就的事情。比如，要想正确理解类、对象、继承、多态这些概念，还是需要花费一番功夫的。而且，在编写面向对象的代码时，还需要注意降低类之间的耦合度，防止过度设计等问题的发生。

虽然面向对象编程有一定的挑战性，但我相信通过对面向对象编程的学习，我相信我可以编写出更加高质量、易于维护的代码。

三．Python 与机械专业联系的思考

Python 是非常强大的工具，它将在我的学术研究中发挥重要作用。对于机械专业研究而言，我认为 Python 在数据分析和可视化，人工智能，计算机辅助设计，机器人技术等方面都有广泛的应用前景。

在数据分析和可视化方面，Python 的许多第三方库（如 NumPy，SciPy 和 Matplotlib）可以使我们有效地处理大量数据，并将数据直观地展示出来。这对于机械设计和模拟过程尤为重要，因为它可以让我们快速验证我们的设计理念，提高设计效率。

在人工智能领域，Python 也有很多高级的库（如 TensorFlow，Keras 等），让我们可以构建深度神经网络，进行图像识别，语音识别等各种任务。这对于现在的智能机械设计和控制领域有很大的帮助。

在计算机辅助设计（CAD）中，Python 也可以扮演重要角色。例如，它可以帮助我们自动化繁琐的设计步骤，让我们可以专注于更有价值的创新活动。此外，随着 3D 打印技术的兴起，Python 在制造领域也有广阔的应用前景。

最后，对于机器人技术，Python 也是一个理想的选择。因为 Python 有许多优秀的机器人框架（如 ROS），可以简化机器人系统的开发流程。同时 Python 强大的矩阵计算功能也使得在求解机器人相关问题是可以很方便的求解矩阵等。

Python 以其简洁，易学和强大的特性为我提供了无限的可能性。无论是在理论上还是实践上，我都期待在未来进一步挖掘 Python 在机械专业的潜力。也感谢刘老师的辛勤付出。

四．结语

作为一门初学者性质的编程语言教学课程，“Python 程序设计”这门课程的学习使我受益匪浅，刘老师的讲授方式充满耐心，同时在课程知识的讲授中也与我们之前专业课程时走马观花式的编程语言学习方式大有不同，刘老师细致的通过 Jupter 的方式以多个短小清晰的代码示例逐步像我们展示 Python 语法的实践意义，比传统的刻板规则描述要清晰很多。

同时通过课程的学习，我也更加清晰了编程时的一些概念，不仅仅局限于 Python 这门语言，诸如类与对象、命名空间与作用域、面向对象等编程概念，在提高我们的代码思维与编写逻辑的能力上有很大帮助。我本身的常用语言是 C/C++，但通过 Python 这门课程，更多体会到 Python 语言的优势所在。Python 语言简洁、易读、强大且灵活，让我可以在寥寥数语间编写出高效的程序。

总之，很庆幸学到这样一门课程，同时我认为这门课程也会在我日后的学习与工作中起到很大的帮助作用，能让我更快做一些算法的

验证与实现等。

五．对课程的建议

我是有一些嵌入式的编程经验的，感觉在 Python 的基础课程教学中，可以适当的引入一些“Python 工程”的概念，也即如何构建一个具有一定规模的 Python 代码项目，而不仅仅局限在一个小部分的代码片段上，这样将会有更加完整的从基础知识到实践过程的学习体验。

附录

完整代码：

```
import gc

class SchoolMember:
    num_total = 0
    num_teacher = 0
    printFlag = True
    def __init__(self, name, age):
        self.name = name
        self.age = age
    def showMember(self):
        if self.printFlag:
            print("当前学校有{}位成员，其中教师{}位，学生{}位".format(self.num_total,
self.num_teacher, self.num_total-self.num_teacher))
    @classmethod
    def add_member(cls,isTeacher=False):
        cls.num_total += 1
        if isTeacher:
            cls.num_teacher += 1
    @classmethod
    def del_member(cls,nameIn,isTeacher=False):
        cls.num_total -= 1
        if isTeacher:
            cls.num_teacher -= 1
            if cls.printFlag:
                print("删除教师{}".format(nameIn))
        if not isTeacher:
            if cls.printFlag:
                print("删除学生{}".format(nameIn))

class Teacher(SchoolMember):
    def __init__(self, name, age, id, salary):
        super().__init__(name, age)
        self.id = id
        self.salary = salary
        SchoolMember.add_member(True)
        print("新增教师{}".format(self.name))
        self.showMember()
    def __del__(self):
        SchoolMember.del_member(self.name,True)
        self.showMember()
```

```
        pass


class Student(SchoolMember):
    def __init__(self, name, age, id, grade):
        super().__init__(name, age)
        self.id = id
        self.grade = grade
        SchoolMember.add_member()
        print("新增学生{}".format(self.name))
        self.showMember()
    @staticmethod
    def avg_grade(students):
        total_grade = 0
        for student in students:
            total_grade += student.grade
        return total_grade / len(students)
    def __del__(self):
        SchoolMember.del_member(self.name,False)
        self.showMember()
        pass
    def printMeanGrade(self):
        studentList = [obj for obj in gc.get_objects() if isinstance(obj,Student)]
        length = studentList.__len__()
        grade = sum([obj.grade for obj in studentList])
        print("学生平均成绩为{}".format(grade/length))


zhang3 = Teacher("zhang3", 35, "001", 5000)
li = Teacher("li", 40, "002", 6000)
mike = Student("mike", 18, "003", 90)
llh = Teacher("llh", 40, "004", 6000)
licy = Student("licy", 18, "003", 95)
mike.printMeanGrade()
del zhang3
albort = Student("albort", 18, "003", 100)
licy.printMeanGrade()
del li
del licy
albort.printMeanGrade()
del mike
albort.printMeanGrade()
SchoolMember.printFlag = False
```

代码运行截图：

```
PS V:\College\Python\大作业> python .\Final.py
新增教师zhang3
当前学校有1位成员，其中教师1位，学生0位
新增教师li
当前学校有2位成员，其中教师2位，学生0位
新增学生mike
当前学校有3位成员，其中教师2位，学生1位
新增教师llh
当前学校有4位成员，其中教师3位，学生1位
新增学生licy
当前学校有5位成员，其中教师3位，学生2位
学生平均成绩为92.5
删除教师zhang3
当前学校有4位成员，其中教师2位，学生2位
新增学生albort
当前学校有5位成员，其中教师2位，学生3位
学生平均成绩为95.0
删除教师li
当前学校有4位成员，其中教师1位，学生3位
删除学生licy
当前学校有3位成员，其中教师1位，学生2位
学生平均成绩为95.0
删除学生mike
当前学校有2位成员，其中教师1位，学生1位
学生平均成绩为100.0
```

查重结果：