



## 中山大学计算机学院

### 人工智能

### 本科生实验报告

(2024 学年春季学期)

课程名称: Artificial Intelligence

教学班级	202320346	专业 (方向)	计算机科学与技术
学号	22320107	姓名	饶鉴晟

#### 一、实验题目

给定 `student_data.txt` 文本文件，每一行是一名学生的信息，从左到右分别是该学生的姓名，学号，性别和年龄，每个属性以空格间隔。数据类型如下：

`name: str # 姓名`

`stu_num: str # 学号`

`gender: str # 性别, "M"为男性, "F"为女性`

`age: int # 年龄`

编写 `StuData` 类，须有以下方法：

1. 构造函数 (`__init__`)，以文件名 (`str` 类型，带 `.txt` 后缀) 为输入，读取文件中的学生信息，储存到类成员 `data` 中。`data` 的数据类型为 `list`，其中每一个学生的信息以列表方式存储。

例如，读入一行学生信息 "Aaron 243 M 18"，则 `data` 变为

```
[["Aaron", "243", "M", 18], ["Eric", "249", "M", 19]]
```

2. `AddData` 方法，以单个学生的信息作为输入，储存到 `data` 中。调用该方法的参数形式为学生属性的 4 个关键字实参。例如，执行 `self.AddData(name="Bob", stu_num = "003", gender="M", age=20)`后，`data` 变为

```
[["Aaron", "243", "M", 18], ["Eric", "249", "M", 19], ["Bob", "003", "M", 20]]
```

3. `SortData` 方法，以学生某个属性 (`str` 类型，是 `'name'`, `'stu_num'`, `'gender'`, `'age'` 的其中之一) 作为输入，将 `data` 按该属性从小到大排序。可以假定不会输入非学生属性的字符串。

例如，执行 `self.Sort('stu_num')`后，`data` 的学生信息按学号从小到大排序，变为

```
[["Bob", "003", "M", 20], ["Aaron", "243", "M", 18], ["Eric", "249", "M", 19]]
```

4. `ExportFile` 方法，以导出的文件名 (`str` 类型，带 `.txt` 后缀) 为输入，新建一个 `txt` 文件，将 `data` 中的数据按当前列表顺序导出到该文件内，格式同原 `student_data.txt` 文本文件，即 "姓名学号性别年龄"，并存储到目标文件夹。例如，调用 `self.ExportFile('new_stu_data.txt')`，则将 `data` 中数据导出到 `new_stu_data.txt` 文件。

## 二、实验内容

### 1. 算法原理

这段代码实现了一个学生信息管理系统的基本功能，包括从文件中读取学生信息、添加新的学生信息、根据指定属性对学生信息进行排序以及将学生信息导出到文件中。

1. 读取文件并解析数据：首先，程序会打开指定的文件，逐行读取文件内容。对于每一行数据，使用空格分割，然后将姓名、学号、性别和年龄分别存储到临时列表 `line_data` 中，并将其添加到 `self.data` 中，形成一个二维列表，存储了所有学生的信息。

2. 添加新的学生信息：通过 `AddData` 方法，可以向学生信息列表 `self.data` 中添加新的学生信息。这个方法接受姓名、学号、性别和年龄作为参数，将这些信息组成一个列表并添加到 `self.data` 中。

3. 根据指定属性排序：`SortData` 方法允许根据学号或年龄对学生信息进行排序。如果指定排序属性为学号 (`stu_num`)，则调用 `take_stu_num` 函数进行排序；如果指定为年龄 (`age`)，则调用 `take_stu_age` 函数进行排序。排序完成后，学生信息列表 `self.data` 会按照指定属性的顺序重新排列。

4. 导出学生信息到文件：通过 `ExportFile` 方法，可以将学生信息导出到指定的文件中。该方法会打开指定文件，遍历学生信息列表 `self.data`，将每个学生的姓名、学号、性别和年龄以指定格式写入文件中，每个学生信息占据一行，以换行符分隔。

整体算法的实现思路主要包括文件读取与解析、添加数据、排序以及文件导出。通过这些功能，可以方便地管理学生信息，并对学生信息进行排序和导出操作。

### 2. 伪代码

---

**Algorithm 1:** Pseudocode for Student Data Management

---

```
Data: a list of student data
Result: sorted and exported student data
1 initialization;
2 foreach line in the file do
3   | parse the line data into name, student number, gender, and age;
4   | append the parsed data to the list of student data;
5 end
6 Function Main(name, stu_num, gender, age):
7   | append a new entry with the provided data to the list of student
   | data;
8 Function SortData(attr):
9   | if attr is 'stu_num' then
10    | | sort the student data list by student number;
11    | end
12   | else if attr is 'age' then
13    | | sort the student data list by age;
14    | end
15 Function ExportFile(filename):
16   | foreach line in the sorted student data list do
17    | | write the name, student number, gender, and age to the file;
18   | end
```

---



### 3. 关键代码展示（带注释）

```
def take_stu_num(elem):  
    return int(elem[1]) # 获取学生学号  
  
def take_stu_age(elem):  
    return elem[3] # 获取学生年龄  
  
class StuData():  
    def __init__(self, filename):  
        self.data = [] # 初始化存储数据的列表  
        # 读取文件  
        with open(filename) as file_object:  
            for line in file_object.readlines():  
                line_data = [] # 临时存储解析的每行数据  
                # 分割每行数据并转换类型，最后加入到self.data中  
                line_data.append(line.split()[0]) # 姓名  
                line_data.append(line.split()[1]) # 学号  
                line_data.append(line.split()[2]) # 性别  
                line_data.append(int(line.split()[3])) # 年龄  
                self.data.append(line_data)  
  
    def AddData(self, name, stu_num, gender, age):  
        self.data.append([name, stu_num, gender, age]) # 添加新的学生信息  
  
    def SortData(self, attr):  
        # 根据指定属性对数据进行排序  
        if attr == 'stu_num':  
            self.data.sort(key=take_stu_num)  
        elif attr == 'age':  
            self.data.sort(key=take_stu_age)  
  
    def ExportFile(self, filename):  
        with open(filename, 'w') as file_object:  
            for line in self.data:  
                file_object.write(line[0])  
                file_object.write(' ')  
                file_object.write(line[1])  
                file_object.write(' ')  
                file_object.write(line[2])  
                file_object.write(' ')  
                file_object.write(str(line[3]))  
                file_object.write('\n')  
        file_object.close()
```



### 三、实验结果及分析

#### 1. 实验结果展示示例（可图可表可文字，尽量可视化）

```
43
44 # 以下是测试代码，用于测试类的各个方法是否能正确执行
45
46 test_filename = "student_data.txt"
47
48 # 实例化StuData类并测试方法
49 def test_class_methods():
50     # 初始化StuData类
51     student_data = StuData(test_filename)
52     print(student_data.data)
53
54     # 添加新的学生信息
55     student_data.AddData("John", "1004", "Male", 23)
56     print(student_data.data)
57
58     # 根据学号排序
59     student_data.SortData('stu_num')
60     print(student_data.data)
61
62     # 根据年龄排序
63     student_data.SortData('age')
64     print(student_data.data)
65
66     # 导出数据到文件
67     student_data.ExportFile('exported_data.txt')
68
69 # 运行测试
70 if __name__ == "__main__":
71     test_class_methods()
```

#### 测试程序输入样例

```
问题 输出 调试控制台 终端 端口 1 注释
@JasonW41k3r →/workspaces/AI_2024/Lab1/code (main) $ /home/codespace/.python/current/bin/python3 /workspaces/AI_2024/
[[['Aaron', '243', 'M', 18], ['Alex', '812', 'M', 24], ['Eric', '249', 'M', 19]]
[[['Aaron', '243', 'M', 18], ['Alex', '812', 'M', 24], ['Eric', '249', 'M', 19], ['John', '1004', 'Male', 23]]
[[['Aaron', '243', 'M', 18], ['Eric', '249', 'M', 19], ['Alex', '812', 'M', 24], ['John', '1004', 'Male', 23]]
[[['Aaron', '243', 'M', 18], ['Eric', '249', 'M', 19], ['John', '1004', 'Male', 23], ['Alex', '812', 'M', 24]]
@JasonW41k3r →/workspaces/AI_2024/Lab1/code (main) $
```

#### 测试程序输出样例

```
file_operation.py exported_data.txt U X
Lab1 > code > exported_data.txt
1 Aaron 243 M 18
2 Eric 249 M 19
3 John 1004 Male 23
4 Alex 812 M 24
```

#### 测试程序输出文件内容样例

如图所示，程序正确初始化数据，可以以指定字段对学生数据进行排序，并且可以正确输出数据到指定文件当中。

## 四、 思考题

### 1. 如果用列表作为字典的键，会发生什么现象？用元组呢？

答：

列表不可以作为字典的键，如果强行将列表作为字典的键，python 解释器会报错"`TypeError: unhashable type: 'list'`"，原因是因为 python 的字典类型要求其键必须是不可变类型，类似 C/C++ 语言当中的右值，而在 python 当中列表是一种可变类型，相当于 C/C++ 当中的左值，可变类型在更改其内容的时候其哈希值会变，导致 python 无法将定义时的键与值链接，所以会报错。

如果用元组的话不会报错，程序可以正常运行，因为元组是一种不可变类型，拥有固定的哈希值，因此可以作为 python 的键。

### 2. 在本课件第 2 章和第 4 章提到的数据类型中，哪些是可变数据类型，哪些是不可变数据类型？试结合代码分析。

答：

不可变类型：数字，字符串，空值，元组（均可以作为右值）

可变类型：列表，字典，集合（均可以作为左值）