中山大学计算机学院人工智能本科生实验报告

课程：Artificial Intelligence

姓名： 学号：

一、实验题目

编写 StuData 类。

二、实验内容

1.算法原理

按照指示完成即可。

2.伪代码

```
类 StuData:
    构造函数(输入：文件名):
        打开文件(文件名):
            foreach 文件每一行:
                将此行信息填入list型成员变量

    函数AddData(输入：学生信息):
        将学生信息填入list型成员变量

    函数SortData(输入：学生属性):
        按照给定学生属性，作为key内部排序

    函数ExportFile(输入：文件名):
        打开文件(文件名):
            foreach list型成员变量中每一学生信息:
                向文件中写入信息
```

3.关键代码展示

```
class StuData:
    def __init__(self, filename: str) -> None:
        self.data = [] #成员变量，首先初始化为空列表
        with open(filename) as file_obj:
            for line in file_obj.readlines():
                self.data.append(line.split()) #因为读入的是字符串str，所以先按空格split为几个str的
                self.data[-1][3] = int(self.data[-1][3]) #将最后一个元素转化为int型

    def AddData(self, **single_stu) -> None:
        self.data.append([ single_stu['name'], single_stu['stu_num'], single_stu['gender'], sing

    def SortData(self, status: str) -> None:
        map = {'name' : 0, 'stu_num' : 1, 'gender' : 2, 'age' : 3} #python中没有switch，就用字典
        self.data.sort(key = lambda statuss : statuss[map[status]]) #调用sort + lambda表达，最大

    def ExportFile(self, filename: str) -> None:
        with open(filename, 'w') as file_obj:
            for single_stu in self.data:
                file_obj.write(" ".join([str(status) for status in single_stu]) + '\n') #由于最后
```

三、实验结果分析

1.实验结果展示

现在给出一个测试用例：

```
students = StuData("student_data.txt")
print(students.data)
students.AddData(name = "Bob", stu_num = "003", gender = "M", age = 20)
print(students.data)
students.SortData('stu_num')
print(students.data)
students.ExportFile("output.txt")
```

其中， `student_data.txt` 是课程给出的用例，其中的内容是：

```
Aaron 243 M 18
Eric 249 M 19
Alex 812 M 19
Leo 092 M 17
Sam 230 F 18
Ruth 942 M 19
Beryl 091 F 20
Cynthia 920 F 19
```

运行之后，终端输出为：

```
[['Aaron', '243', 'M', 18], ['Eric', '249', 'M', 19], ['Alex', '812', 'M', 19], ['Leo', '092', 'M', 17], ['Sam', '230', 'F', 18], ['Ruth', '94
2', 'M', 19], ['Beryl', '091', 'F', 20], ['Cynthia', '920', 'F', 19]]
[['Aaron', '243', 'M', 18], ['Eric', '249', 'M', 19], ['Alex', '812', 'M', 19], ['Leo', '092', 'M', 17], ['Sam', '230', 'F', 18], ['Ruth', '94
2', 'M', 19], ['Beryl', '091', 'F', 20], ['Cynthia', '920', 'F', 19], ['Bob', '003', 'M', 20]]
[['Bob', '003', 'M', 20], ['Beryl', '091', 'F', 20], ['Leo', '092', 'M', 17], ['Sam', '230', 'F', 18], ['Aaron', '243', 'M', 18], ['Eric', '24
9', 'M', 19], ['Alex', '812', 'M', 19], ['Cynthia', '920', 'F', 19], ['Ruth', '942', 'M', 19]]
```

可以看到，第一第二行的输出，对应的就是原文件中的内容。第三第四行的输出，在成员变量的末尾添加了信息 `name = "Bob", stu_num = "003", gender = "M", age = 20` 。第五第六的输出，是按照学号的大小，顺序输出的。当然，最后还有一个输出的文件，如图所示：

都如预料中的运行了。

2.评测指标展示分析

构造函数和输出函数都必须遍历已有内容，因此没有可再优化区间，时间复杂度都是$O(n)$。

添加单个学生的信息，能够以$O(1)$复杂度完成。

学生信息排序函数，仅针对排序其复杂度为$O(nlogn)$。而针对排序对象，当其为字符串时，需要比较字典序，那么这部分复杂度为$O(len(str))$。

四、思考题

1.可哈希数据结构

如果用列表作为字典的键，会发生什么现象？用元组呢？

使用列表作为键时，会报错提示 `TypeError: unhashable type: 'list'` ，如图所示：

```
5   diction = {[1,2,3] : 1, [2,3,4] : 2, [3,4,5] : 3}
6   print(diction[[1,2,3]])
Traceback (most recent call last):
  File "f:\资料及作业\作业\人工智能\Python Project\第二次实验.py", line 5, in <module>
    diction = {[1,2,3] : 1, [2,3,4] : 2, [3,4,5] : 3}
TypeError: unhashable type: 'list'
```

使用元组时：

```
5   diction = {(1,2,3) : 1, (2,3,4) : 2, (3,4,5) : 3}
6   print(diction[(1,2,3)])
```

顺利输出 `1` 。

这是因为 `list` 并不支持 `hash` 函数。而 `tuple` 是不可变的，其性质允许其匹配 `hash` 函数。

2.可变数据类型

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。

可变/不可变数据类型：变量值发生改变时，变量的内存地址不变/改变。

提示：① 你可能会用到id()函数。② Python的赋值运算符（=）是引用传递。

不可变数据类型：基本数据类型 `int,float,bool` ， 字符串 `str` ， 元组 `tuple` 。

可变数据类型：列表 `list` ， 字典 `dict` ， 集合 `set` 。

如图所示，仅有 list,dict,set 数据类型的 id 没有改变：

```
data_bool, data_int, data_float, data_str = True, 1, 3.14, 'JDK9'
data_tuple, data_list = (1,2,3), [10,11,12]
data_dict, data_set = {'C++' : 20, 'JAVA' : 9}, {100,200,300}
print(id(data_bool), id(data_int), id(data_float), id(data_str), id(data_tuple), id(data_list), id(data_d
data_bool, data_int, data_float, data_str = False, 5, 2.79, "Python 3.9.7"
data_tuple = (3,2,1)
data_list.append(13)
data_dict['Python'] = 3
data_set.add(400)
print(id(data_bool), id(data_int), id(data_float), id(data_str), id(data_tuple), id(data_list), id(data_d
140711078193256 2380937521456 2380938167696 2380943303024 2380942992768 2380942957952 2380942919360 2380943178432
140711078193288 2380937521584 2380943031280 2380943305712 2380943004288 2380942957952 2380942919360 2380943178432
```

五、参考资料

无。