

学	10212819108
---	-------------

武汉华夏理工学院

# 课 程 设 计

课程名称：C 语言程序设计课程设计

题 目 学生成绩排名  
 专 业 软件工程  
 班 级 软件 1191z  
 姓 名 胡文卓  
 成 绩 \_\_\_\_\_  
 指导老师 王绪梅

2020 年 5 月 11 日至 2020 年 5 月 22 日

# 武汉华夏理工学院信息工程学院

## 课程设计任务书

课程名称: C 语言程序设计课程设计 指导教师: 王绪梅  
班级名称: 软件 1191z 开课院系: 计算机与网络工程系

### 一、课程设计目的与任务

C语言程序设计课程是计算机类(包括3个专业: 计算机科学与技术、软件工程、物联网工程)的集中实践教学模块。该课程设计的目的是进一步培养学生结构化程序设计思想, 加深学生对高级语言基本语言要素和控制结构的理解, 针对C语言中的重点和难点内容进行训练, 使学生能独立完成有一定工作量的程序设计任务, 为后续课程, 如《C++程序设计》、《数据结构》等奠定坚实基础。一方面通过实践训练, 培养学生的编程能力及实践动手能力, 体现科学技术在社会发展中的作用; 另一方面, 提高学生综合运用所学知识解决实际问题的能力及自主创新能力。

### 二、课程设计的内容与基本要求

#### 1. 设计内容

到了期末考试了, 老师都会对班上所有的学生的成绩进行排序, 一般情况下是从高分到低分排序, 现在你就是老师, 要求你利用我们课堂上所学C语言的知识编写一个对你班学生的成绩进行排序的程序, 要求采用冒泡排序法。

提示: 熟悉变量、数组定义、使用、输入、输出等基本操作, 以及进行选择、循环结构程序设计练习和掌握冒泡法排序的算法等。

#### 2. 要求完成的任务

- (1) 完成规定任务的设计及调试, 且一定要画出程序流程图, 最后得出正确结果, 并经教师检查及答辩;
- (2) 写出规范的课程设计说明书;
- (3) 课程设计结束后提交课程设计报告及电子版资料。

### 三、学时分配进度安排

序号	设计内容	所用时间
1	选题及查阅相关资料	1 天
2	软件结构设计	1 天
3	软件编程软件	1 天
4	调试及撰写报告	1 天
5	答辩	1 天
合 计		1 周

### 四、课程设计考核及评分标准

#### 1. 设计报告要求

课程设计报告要求逻辑清晰、层次分明、书写整洁。格式完全按照附录：课程设计规范模板书写，包括封面、任务书、目录、正文、参考文献、总结、成绩评定等。课程设计报告须每人一份，独立完成，不可雷同。

## 2. 评分标准

评 分 依 据	分 值	评分成绩
1. 选题完成程度	10 分	
2. 态度认真、学习刻苦、遵守纪律	10 分	
3. 设计方案正确，具有可行性	20 分	
4. 独立性与创新性	5 分	
5. 系统调试与结果	20 分	
6. 参考文献充分（不少于 5 篇）	5 分	
7. 设计报告撰写规范	10 分	
8. 答辩	20 分	
总 分	10 分	

注：成绩等级：优（90分—100分）、良（80分—89分）、中（70分—79分）、及格（60分—69分）、60分以下为不及格。

## 五、指导时间

**班级：**计算机 1191、软件 1191

周次	星期一	星期二	星期三	星期四	星期五
13周	3-4 节 腾讯会议号： 503846630		3-4 节 腾讯会议号： 503846630		
14周	3-4 节 腾讯会议号： 503846630		3-4 节 腾讯会议号： 503846630	3-4 节 腾讯会议号： 503846630	

# 目 录

1 设计题目 .....	1
2 开发环境 .....	1
3 开发工具 .....	1
4 完成时间 .....	1
5 设计思想 .....	1
6 设计过程及设计步骤 .....	5
7 测试运行 .....	11
8 评价与修订 .....	15
9 设计体会 .....	17
附 录 .....	18
致 谢 .....	22

# 1 设计题目

学生成绩排名

# 2 开发环境

硬件环境：ThinkPad T470，内存 16G，软件环境：Microsoft Windows 10

# 3 开发工具

PyCharm，Python 3.7，Miniconda，JupyterNotebook

# 4 完成时间

2020-5-11----2020-5-22

# 5 设计思想

该项目需要实现的功能较多，故此分多节描述。

## 5.1 需求分析

1. 学生信息管理
  - a) 学生信息的插入
  - b) 学生信息的删除
  - c) 学生信息的修改
  - d) 学生信息的查找
  - e) 学生信息的保存
  - f) 学生信息的展示
2. 数据管理
  - a) 求平均值
  - b) 求最大值

- c) 求最小值
- d) 求总和
- 3. 数据可视化
  - a) 总分表的绘制
  - b) 平均分表的绘制
  - c) 最高/低分表的绘制

## 5.2 界面设置

为了给用户一个良好的体验环境，这里对界面的设置是选项、文本居中，输入输出左对齐。

页面的布局参考图 1.1

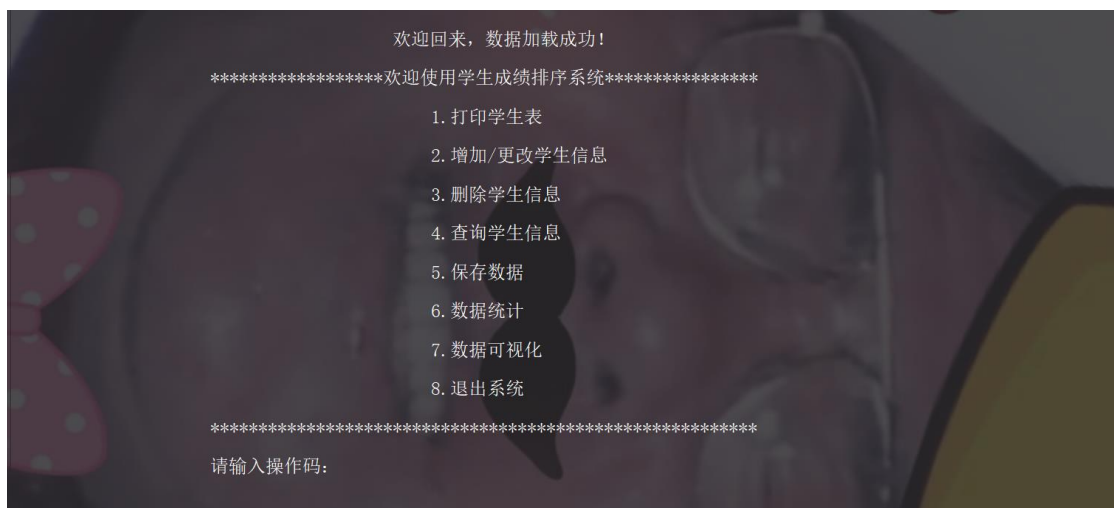


图 1.1 页面的布局参考图

代码参考图 1.2（这里用到了对齐文本的方法，即补齐空格）

```
# 打印菜单
def printMenu():
    print("".rjust(20) + "欢迎使用学生成绩排序系统".center(45, "*"))
    # 对齐文本
    for i in range(len(menu)): print("".rjust(19) + (str(i + 1) + "." + menu[i]).rjust(25 + len(menu[i])))
    print("".ljust(19) + "*" * 57)
```

图 1.2 代码参考图

## 5.3 选项设置

根据对应的功能提供对应的选项，用户输入选项代码即可执行对应的代码。部分选项有：打印学生表、增加/查找/删除/更新学生信息等..

选项参考图 1.1

代码参考图 1.3（此处不仅提供输入操作码的功能，也提供了判断操作符是否合法的功能）

```
def printPrompt():
    print("\n.ljust(19), end="")
    code = -1
    try:
        code = int(input("请输入操作码: "))
        # 非法数据
        if code < 1 or code > len(menu):
            code = -1
            print("\n.ljust(19) + "操作码非法!")
    except ValueError:
        print("\n.ljust(19) + "请输入整形!") # 非法输入
    return code
```

图 1.3 代码参考图

## 5.4 代码执行设置

考虑到该项目需要设计的功能较多，且每个功能的选定都是由对应的操作码执行的，因此如果写很多个选择语句对应操作码，必然会造成代码的冗余。

如果将每个要执行的方法看作是一串字符，便可发现大部分的方法执行都是差不多的。于是可以规定每个方法的方法名只有对应的功能字符串不相同，其它部分的字符串全相同，便可利用 `eval` 函数直接通过“操作码->字符串”的映射执行对应的方法。

操作码对应字符串参考图 1.4（用户输入操作码，根据操作码到对应字典里取字符串）

```
# 选项文本
menu = ["打印学生表", "增加/更改学生信息", "删除学生信息", "查询学生信息", "保存数据", "数据统计", "数据可视化", "退出系统"]
# 操作码对应的方法的字符串
orderMap = {1: "print", 2: "insert", 3: "delete", 4: "select", 5: "updateData", 6: "alaData", 7: "showData"}
```

图 1.4 操作码对应字符串参考图

方法的执行参考图 1.5（`student` 是一个实例，`Student` 类提供了对应方法，命名规则是：对应功能字符串+`Student()`）

```
eval("student." + orderMap[code] + "Student()")
```

图 1.5 方法的执行参考图

## 5.5 函数功能简要介绍

1. printMenu 函数，提供了打印菜单的功能
2. printPrompt 函数，提供了获得操作码的功能
3. runCode 函数，提供了调用 Student 类的功能，该函数会被循环调用多次，同时该函数也会调用
4. printPrompt 函数，提供了获得用户输入的操作码的功能

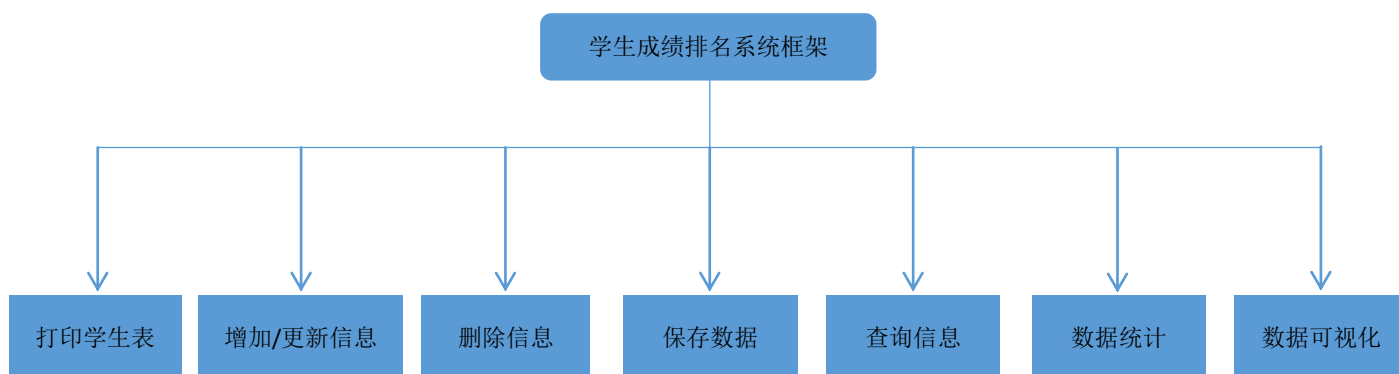
主函数代码参考图 1.6

```
def main():  
    # 因为在循环里无限实例化Student类导致全部操作都有问题  
    student = Student()  
    while True:  
        printMenu()  
        runCode(printPrompt(), student=student)
```

图 1.6 主函数代码参考图

## 5.6 框架图

该项目一共提供 7 个功能，可以根据用户输入执行相应的代码。





## 6 设计过程及设计步骤

本地数据格式请参考附录：本地数据

### 6.1 Student 类-学生信息管理设计

构造一个**学生类**，使该对象整合学生信息的增加/删除/修改/查询，并提供简单的学生成绩分析，学生成绩可视化功能。

Student 类是本程序的核心类，**所有选项**都是通过调用 Student 的方法进行执行的。

**数据本地化**的实现：利用 json 格式化字典，然后将 json 数据保存在程序运行目录下的 data.json 里。

#### Student 类提供的方法

1. insertStudent 方法，提供了学生信息的增加与修改
2. deleteStudent 方法，提供了学生信息的删除
3. selectStudent 方法，提供了学生信息的查找
4. printStudentInfoLine 方法，提供了按学号打印学生信息功能，多次调用该方法可以打印多个学生信息，比如打印学生表功能，该方法主要通过调用 prettytable 库中的 PrettyTable 类打印文字表格
5. printStudentScore 方法，提供了打印某一类数据信息的功能，比如最大值，最小值等，多次调用可以打印多个数据信息，比如数据统计功能
6. updateDataStudent 方法，提供了将学生信息本地化的功能
7. 构造方法，提供了加载数据，初始化字典的功能

备注： 2.以及 6.的执行会调用 Marina 类的 udk 实例更新数据统计的表

**insertStudent 方法**参考图 2.1（该方法是 Student 类中的重要方法，这里只截取了部分代码）

```

if flag:
    # 此处不可使用一个list对象，不然共用一个内存地址
    # 改成dict了，因为list不能做映射，那么修改数据的时候就要判断一堆下标之类的东西
    self.data["info"][suid] = {}
    self.data["score"][suid] = {}
totScore = 0
for i in range(1, len(lt) - 1):
    # 根据学号存入学生信息
    if i <= 3: tKey = "info"
    else: tKey = "score"
    temp = input("".ljust(19) + "请输入" + lt[i] + ":")
    if not temp and flag: temp = "空" # 如果该学生不存在且输入为空设置默认值
    elif not temp and not flag: continue # 如果输入为空但学生存在则不作更改
    self.data[tKey][suid][lt[i]] = temp
for i in self.data["score"][suid].values(): totScore += float(i) # 计算总分
self.data["tot_score"][suid] = totScore
self.u = udk(self.data, lt[4:7]) # 学生成绩是会改变的，因此需要多次实例化
input("".ljust(19) + "按任意键继续...")

```

图 2.1 insertStudent 方法参考图

构造方法参考图 2.2（该方法用来加载本地数据，如果本地文件不存在或者格式不合法，将会自动创建新文件并初始化空数据）

```

# 创建类的时候加载本地数据
def __init__(self):
    try:
        # 读入学生信息
        with open("data.json", "r") as f: self.data = json.load(f)
        print("".ljust(38) + "欢迎回来，数据加载成功！")
    except (json.decoder.JSONDecodeError, FileNotFoundError):
        # 可能文本内容不是json格式，第一次运行文件可能不存在
        open("data.json", "w").close()
        self.data = {"info": {}, "score": {}, "tot_score": {}}
    self.u = udk(self.data, lt[4:7]) # 实例化数据统计类

```

图 2.2 构造方法参考图

printStudentInfoLine 方法参考图 2.3（具体效果参考“测试运行”）

```

# 打印一行学生信息
def printStudentInfoLine(self, suid, flag=False):
    table = pt(lt)
    table.align[lt[1]] = "l"
    row = [suid] + list(self.data["info"][suid].values()) + list(self.data["score"][suid].values()) + \
        [self.data["tot_score"][suid]]
    table.add_row(row)
    # 去掉边框，因为边框存在布局错误，再转换为字符串逐行对齐输出
    table.border = False
    # 通过多次调用逐行打印学生成绩达到打印整个表的效果，但是只能输出一次表头
    for s in table.get_string().splitlines():
        if flag: # 如果执行的是打印整个表，则不会打印表头
            flag = not flag
            continue
        print("".ljust(19) + s)

```

图 2.3 printStudentInfoLine 方法参考图

## 6.2 Marina 类-学生成绩的统计

考虑到仅仅对学生信息进行增删改查是完全不能满足该项目要求的，于是这里构造了一个 Marina 类用来提供学生信息统计功能。

Marina 类不会进行冗余计算，即该类的实例只有在数据发生改变时才会调用方法更新数据。

该类是一个独立的模块，由 main.py 导入并由 Student 类进行实例化与调用。

### Marina 类提供的方法

1. 构造方法，该方法提供了数据格式化以及调用统计的功能，可以保证在数据不发生改变的情况下不进行重复计算
2. \_\_cmpSumScore 方法，提供了计算学生成绩之和的功能
3. \_\_cmpTopScore 方法，提供了计算学生成绩最大值，最小值的功能
4. getSumScore 方法，提供了返回学生成绩之和的功能
5. getArrScore 方法，提供了返回学生成绩平均值的功能
6. getMaxScore/getMinScore 方法，提供了返回学生成绩最大值最小值的功能

构造方法参考图 3.1（该方法初始化了数据表，以及调用两个计算方法）

```
def __init__(self, data, course_list):
    self.data = data
    self.sum_dict = {}
    self.max_dict = {}
    self.min_dict = {}
    for i in course_list:
        self.sum_dict[i] = .0
        self.max_dict[i] = (.0, 0)
        self.min_dict[i] = (3777.0, 0)
    self.__cmpSumScore()
    self.__cmpTopScore()
```

图 3.1 构造方法参考图

计算方法参考图 3.2（两个私有方法，是该类的核心方法）

```
# 计算每门成绩的总和，同时可以通过该字典获取平均分
def __cmpSumScore(self):
    for val in self.data["score"].values():
        for c in val:
            self.sum_dict[c] += float(val[c])

# 获取最高分最低分学生学号以及成绩
def __cmpTopScore(self):
    for tup in self.data["score"].items(): # 拿到成绩元组
        for cor in tup[1]: # 遍历元组里的values, cor是key, 也就是课程名称
            # 依次对比成绩，先放成绩，后放学号
            if float(self.max_dict[cor][0]) < float(tup[1][cor]): self.max_dict[cor] = (float(tup[1][cor]), tup[0])
            if float(self.min_dict[cor][0]) > float(tup[1][cor]): self.min_dict[cor] = (float(tup[1][cor]), tup[0])
```

图 3.2 计算方法参考图

get 方法参考图 3.3（参考了 Java 类的 getter 方法，通过两个私有方法计算出数据，然后直接调用便可得到结果）

```

# 获取每门成绩的总和
def getSumScore(self):
    return self.sum_dict

# 获取每门成绩平均分，均分四舍五入保留两位小数
def getArrScore(self):
    return {i[0]: round(i[1] / len(self.data["score"]), 2) for i in self.sum_dict.items()}

def getMaxScore(self):
    return self.max_dict

def getMinScore(self):
    return self.min_dict

```

图 3.3 get 方法参考图

## showData 函数-数据的可视化

考虑到仅仅在 Console 打印数据表并不直观，所以这里提供了数据可视化功能。该函数主要绘制三个表，总分表、均分表、最高/最低分表。

其中最高/最低分表为了更直观的对比，直接绘制在了一个子图里。

showData 函数参考图 4.1（此处给出部分代码，具体运行效果参考“测试运行”）

```

max_vl = [i[0] for i in max_score.values()]
min_vl = [i[0] for i in min_score.values()]
pos = [i for i in range(0, 3)]
wid = 0.25
max_bt = plt.bar(pos, max_vl, width=wid, color="deepskyblue")
for i in range(3): pos[i] += wid
min_bt = plt.bar(pos, min_vl, width=wid, color="red")
pos = [i + 0.1 for i in range(3)]
plt.xticks(pos, course_list)
for min_t in min_bt:
    height = min_t.get_height()
    plt.text(min_t.get_x() + wid / 2 - 0.01, height + 13, height, color="crimson")
for max_t in max_bt:
    height = max_t.get_height()
    plt.text(max_t.get_x() + wid / 2 - 0.01, height + 13, height, color="c")
plt.savefig("./table.png") # 得先保存，不然plt对象刷新了就只能保存空白图片了
plt.show()
input("\n.ljust(19) + "表格绘制完成！已保存到当前目录下的table.png，请按任意键继续..."")

```

图 4.1 showData 函数参考图



## 7 测试运行

该项目所有功能均对空数据的情况进行了处理，以打印提示的方式提醒用户数据为空，故此，此处的演示默认数据不为空。

程序界面参考图 1.1

打印学生表功能演示参考图 5.1（该表按“总分”关键字（即学生三门课程的成绩和）进行了降序排序，且可以保证表中数据的实时更新）



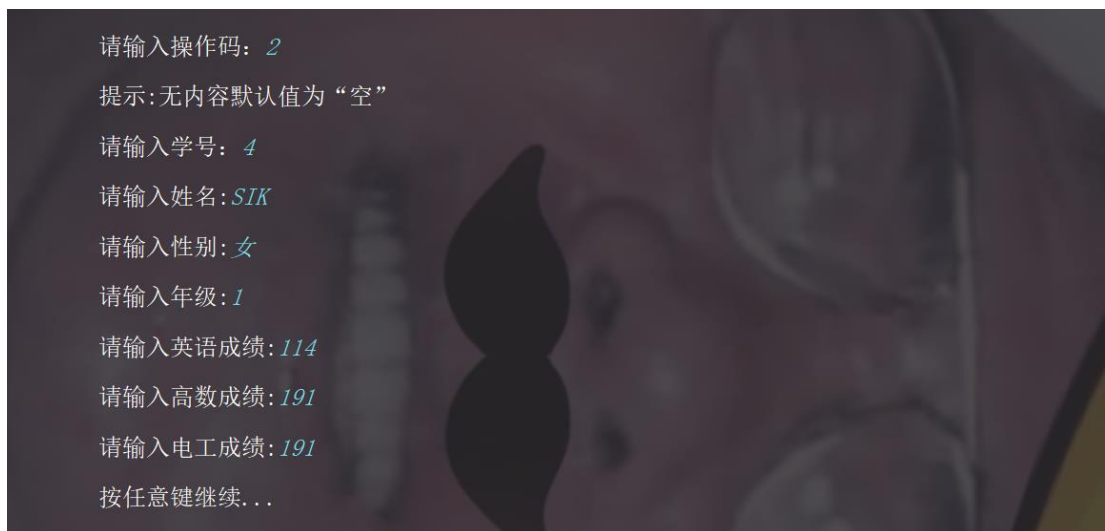
```
请输入操作码: 1
```

学号	姓名	性别	年级	英语成绩	高数成绩	电工成绩	总分
3	HNS	女	1	999	114	514	1627.0
1	UDK	女	145	124	956	145	1225.0
2	RU	14	25	114	514	90	718.0

按任意键继续...

图 5.1 打印学生表功能演示参考图

增加学生信息功能演示参考图 5.2（该功能可以通过关键字“学号”自动判别当前执行的操作，如果学号存在，即进行增加学生信息操作，否则进行修改学生信息操作）



```
请输入操作码: 2
提示:无内容默认值为“空”
请输入学号: 4
请输入姓名: SIK
请输入性别: 女
请输入年级: 1
请输入英语成绩: 114
请输入高数成绩: 191
请输入电工成绩: 191
按任意键继续...
```

图 5.2 增加学生信息功能演示参考图

**更新学生信息功能演示**参考图 5.3（该方法可以根据输入自动判别是否更改学生信息，当输入为空时，不会进行更改，否则进行更改，效果见“查询学生信息”功能演示）

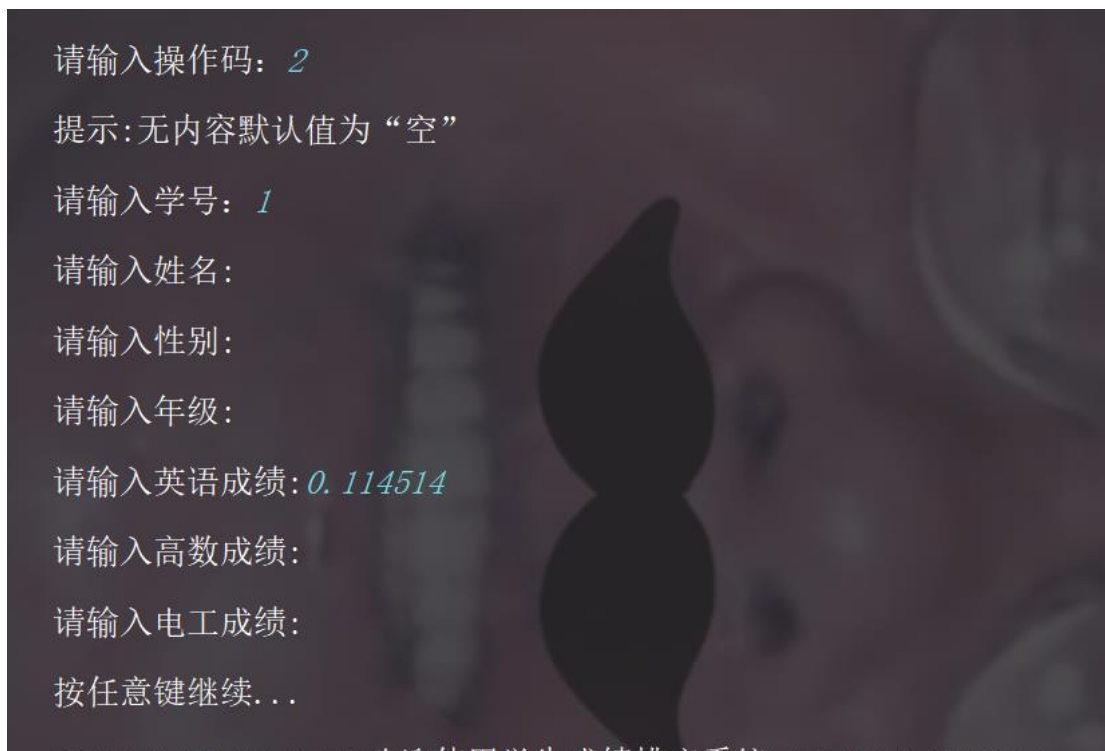


图 5.3 更新学生信息功能演示参考图

**删除学生信息功能演示**参考图 5.4（该方法可以在内存中删除学生信息，但不会在本地删除学生信息，即用户可以选择是否真正执行删除操作）

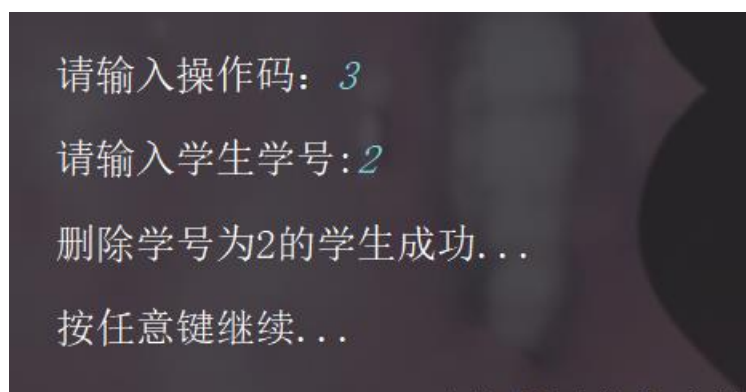


图 5.4 删除学生信息功能演示参考图

**查询学生信息功能演示**参考图 5.5（该方法可以根据学号查询该学生的详细信息，“打印学生表”功能即多次调用该方法以做到打印全部学生的效果）



图 5.5 查询学生信息功能演示参考图

保存数据功能演示参考图 5.6（该方法可以将内存数据写入外存进而实现持久化的效果，同样也可以选择是否真正修改本地数据）

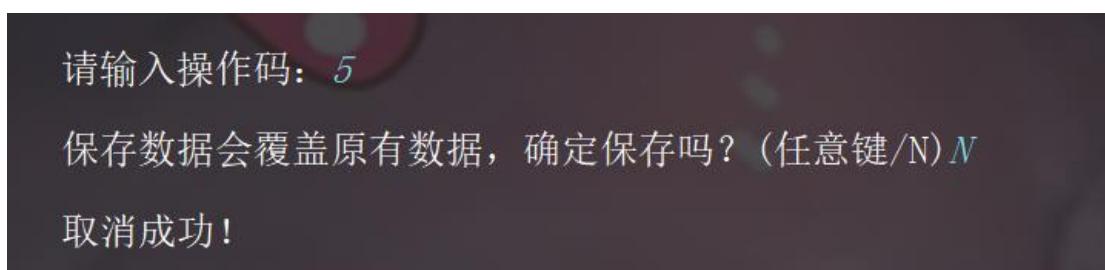


图 5.6 保存数据功能演示参考图

数据统计功能演示参考图 5.7、图 5.8（该方法提供打印总分、平均分、最高分、最低分表的功能，该表只有在执行“增加/修改，删除”功能时进行更新，且该表支持数据可视化）



图 5.7 数据统计功能演示参考图



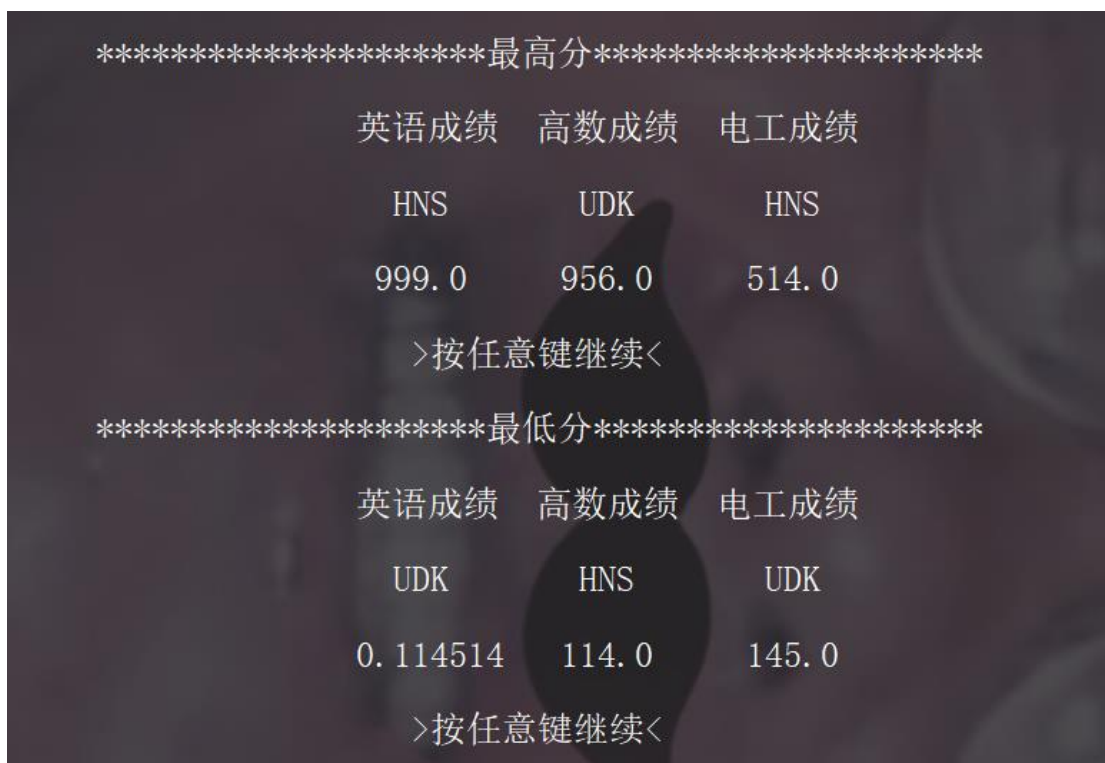


图 5.8 数据统计功能演示参考图

数据可视化功能演示参考图 5.9、图 5.10（该方法通过调用 `showData` 函数将数据可视化成表格并将图保存在程序运行目录下的 `table.png`，分别有折线、条形、柱状图）



图 5.9 数据可视化功能演示参考图

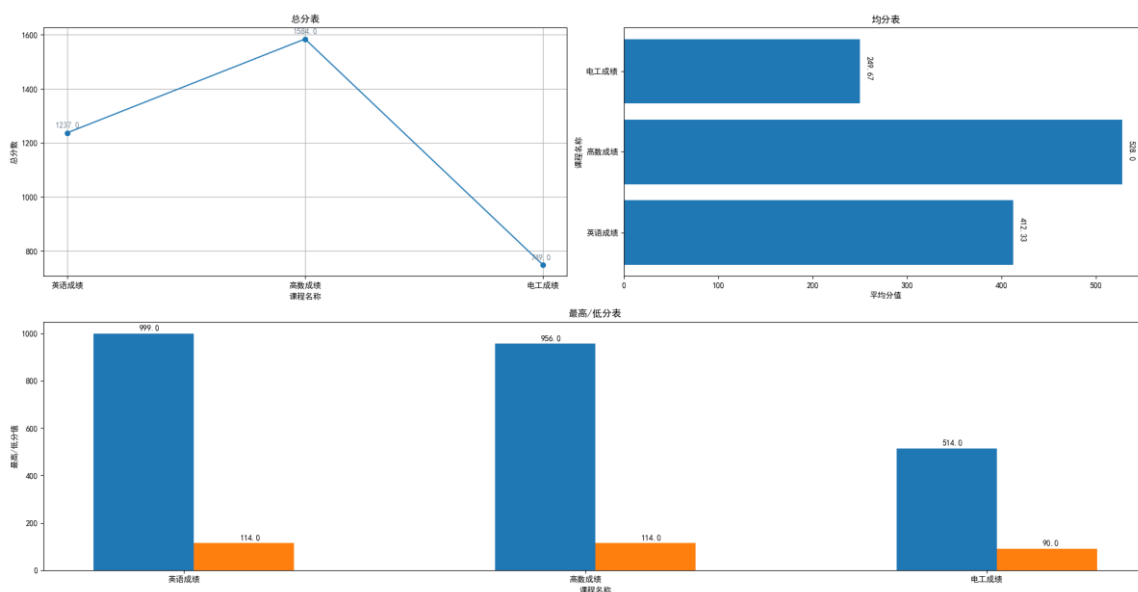


图 5.10 数据可视化功能演示参考图

退出系统方法演示参考图 5.11（该方法可以终止程序的运行，并且提示用户是否保存运行数据）

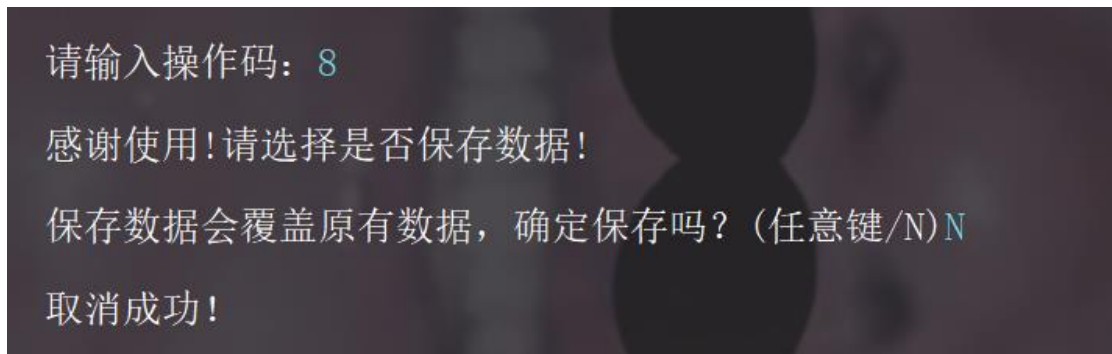


图 5.11 退出系统方法演示参考图

## 8 评价与修订

这次项目总体来说比较一般，虽然代码写了很多，但基本都没什么很特别的。大部分功能也不是一开始就设计好了的，只有增删改查等基本功能是一开始就打算写的，其它功能基本是写着写着想起来的。

这次项目里个人感觉最有创造性的是“利用 eval 函数执行方法”以及“数据可视化”了，虽然后者并没什么新意，但是作为一个一闪而过的灵感而言也导致了后面编码中或多或少在为这个功能服务。而利用 eval 函数是因为 Python 不提供类似 switch 语句的功能，如果写很多个 if..else..elif..无疑是对 Python “简洁”原则的违背，虽然其它部分的代码也是各种冗余，但是这里的简化操作的价值还是很大的。

整体来说我对这个项目还是比较满意的，用 Python 写大概也比较特别，但是也有很多不足，比如这个项目一个手写的算法/数据结构也没用到，还有功能设计的不完善等。对此，我认为改进方案是在一开始就写好需求分析，并在后续编码中完善、增加功能。

## 9 设计体会

“时间就像海绵里的水只要愿挤总还是有的”，虽然这个项目写的时间不是很长，但是能用来写的时间却很少，为此我用了大部分无聊的时间写这个项目。

在设计功能时也发现了能力上的不足，大部分代码都是“即兴”写出来，这也导致了有些功能的冲突进而只能重构代码。另外因为太久没有写 Python 所以只能靠着印象中的语法、设计模式来编码。不过这个项目也让我重新找回了编程的快乐，每一次调试、对功能的完善，无不都是对自己能力的肯定。

BUG 虽然很可恶，但是在思考问题、解决问题的过程中，我收获的却不仅仅只有对程序为什么出了 BUG 的疑惑，还有一步一个脚印，踏踏实实的进步的成就感。

编写程序的过程中，我也意识到了利用搜索引擎是多么重要。许多问题都是只有碰到了才会发现自己不会，这种时候只是暴力调试往往不能获得理想的效果。而通过谷歌搜索引擎，往往可以很方便的找到问题的症结，进而帮助我们解决问题。

回首大约三年前的一天，我正式的接触到了 C 语言这门编程语言，然而当时连”hello world”都写了半天。即是如此，编程的魅力依然深深的吸引着我。后续的学习中，我也尝试过自己写一些小程序，应用之类的东西，但因为急于求成大部分知识都是囫圇吞枣，草草写完程序就忘记了。这个习惯和我对编程的热爱一起被保留到了现在，这次项目也不例外，随意的写完功能分析后就直接开始编码了，但好在积累了一些经验所以并没有像三年前那样吃瘪。

展望未来，我并不敢断言自己能够坚持下去，“三分钟热度”是我改不了的坏习惯。但学习编程无疑是我目前为止做的最对的选择，它不仅锻炼了我的耐力，更教会了我自学的重要性。

以上便是我的设计体验，感谢阅读！

## 参 考 文 献

- <sup>[1]</sup>Aditya Bhargava. Grokking Algorithms: An illustrated guide for programmers and other curious people. 北京:人民邮电出版社. 2017-3
- <sup>[2]</sup>米切尔. Web Scraping with Python: Collecting Data from the Modern Web. 北京:人民邮电出版社. 2016-3-1
- <sup>[3]</sup>Luciano Ramalho. Fluent Python. 北京:人民邮电出版社. 2017-5-15
- <sup>[4]</sup>Wes McKinney. Python for Data Analysis. 北京:机械工业出版社. 2013-11-18
- <sup>[5]</sup>啊哈磊. 啊哈!算法. 北京:人民邮电出版社. 2014-6-1

# 附录

```
▼ {info: {1: {姓名: "UDK", 性别: "女", 年级: "145"}, 2: {姓名: "RU", 性别: "14", 年级: "25"},...},...}
  ▼ info: {1: {姓名: "UDK", 性别: "女", 年级: "145"}, 2: {姓名: "RU", 性别: "14", 年级: "25"},...}
    ▼ 1: {姓名: "UDK", 性别: "女", 年级: "145"}
      姓名: "UDK"
      年级: "145"
      性别: "女"
    ▶ 2: {姓名: "RU", 性别: "14", 年级: "25"}
    ▶ 3: {姓名: "HNS", 性别: "女", 年级: "1"}
  ▼ score: {1: {英语成绩: "124", 高数成绩: "956", 电工成绩: "145"}, 2: {英语成绩: "114", 高数成绩: "514", 电工成绩: "90"},...}
    ▼ 1: {英语成绩: "124", 高数成绩: "956", 电工成绩: "145"}
      电工成绩: "145"
      英语成绩: "124"
      高数成绩: "956"
    ▶ 2: {英语成绩: "114", 高数成绩: "514", 电工成绩: "90"}
    ▶ 3: {英语成绩: "999", 高数成绩: "114", 电工成绩: "514"}
  ▼ tot_score: {1: 1225, 2: 718, 3: 1627}
    1: 1225
    2: 718
    3: 1627
```

## 本地数据参考图

### main.py 代码

```
from prettytable import PrettyTable as pt
from marina import Marina as udk
from show import showData
import json

# dump/load: 文件操作  dumps/loads: 对象转字符串

# 选项文本
menu = ["打印学生表", "增加/更改学生信息", "删除学生信息", "查询学生信息", "保存数据", "数据统计", "数据可视化", "退出系统"]

# 操作码对应的方法的字符串
orderMap = {1: "print", 2: "insert", 3: "delete", 4: "select", 5: "updateData", 6: "alaData", 7: "showData"}

# 学生信息列表
lt = ["学号", "姓名", "性别", "年级", "英语成绩", "高数成绩", "电工成绩", "总分"]

class Student(object):
    # 创建类的时候加载本地数据
    def __init__(self):
        try:
            # 读入学生信息
            with open("data.json", "r") as f: self.data = json.load(f)
            print("\n.ljust(38) + "欢迎回来，数据加载成功！")
        except (json.decoder.JSONDecodeError, FileNotFoundError):
            # 可能文本内容不是 json 格式,第一次运行文件可能不存在
            open("data.json", "w").close()
            self.data = {"info": {}, "score": {}, "tot_score": {}}
            self.u = udk(self.data, lt[4:7]) # 实例化数据统计类
```

```

# 检查数据是否为空
def isEmpty(self):
    return not len(self.data["info"]) or not len(self.data["score"])

# 更新本地数据
def updateDataStudent(self):
    flag = input("\nJust(19) + "保存数据会覆盖原有数据，确定保存吗？(任意键/N)")
    if flag == "N":
        print("\nJust(19) + "取消成功！")
        return 0
    if self.isEmpty():
        print("\nJust(19) + "数据为空，请先输入数据！")
        return 0
    try:
        with open("data.json", "w") as f: json.dump(self.data, f)
    except IOError:
        print("\nJust(19) + "数据保存失败！")
        return 0
    input("\nJust(19) + "数据保存成功！")
    input("\nJust(19) + "按任意键继续...")

#打印学生表
def printStudent(self):
    if self.isEmpty():
        print("\nJust(19) + "表格为空，请先添加数据！")
        return 0
    # 降序输出成绩表
    sortedList = sorted(self.data["tot_score"].items(), key=lambda item: item[1], reverse=True)
    # 打印一次表头
    table = pt(lt)
    table.align[lc[1]] = "l"
    table.border = False
    table.add_row(lt)
    # 小 BUG，这里不能单独输出表头，因此得多加一行单独输出
    print("\nJust(19) + table.get_string().splitlines()[1])
    for k in sortedList: self.printStudentInfoLine(k[0], True)
    input("\nJust(19) + "按任意键继续...")

# 增加学生信息
def insertStudent(self):
    print("\nJust(19) + "提示:无内容默认值为“空”")
    try:
        suid = input("\nJust(19) + "请输入学号：")
        if int(suid) < 1:

```

东西

```
        print("".ljust(19) + "学号必须大于等于 1! ")
        return 0
except ValueError:
    print("".ljust(19) + "格式错误!")
    return 0
# 如果该学生不存在，则创建新的信息列表，否则在原有信息列表进行修改，即更新
flag = suid not in self.data["info"].keys() or suid not in self.data["score"].keys()
if flag:
    # 此处不可使用一个 list 对象，不然共用一个内存地址
    # 改成 dict 了，因为 list 不能做映射，那么修改数据的时候就要判断一堆下标之类的

    self.data["info"][suid] = {}
    self.data["score"][suid] = {}
totScore = 0
for i in range(1, len(lt) - 1):
    # 根据学号存入学生信息
    if i <= 3: tKey = "info"
    else: tKey = "score"
    temp = input("".ljust(19) + "请输入" + lt[i] + ":")
    if not temp and flag: temp = "空" # 如果该学生不存在且输入为空设置默认值
    elif not temp and not flag: continue # 如果输入为空但学生存在则不作更改
    self.data[tKey][suid][lt[i]] = temp
for i in self.data["score"][suid].values(): totScore += float(i) # 计算总分
self.data["tot_score"][suid] = totScore
self.u = udk(self.data, lt[4:7]) # 学生成绩是会改变的，因此需要多次实例化
input("".ljust(19) + "按任意键继续...")

# 删除学生信息
def deleteStudent(self):
    # 学生不存在
    suid = self.selectStudent(isprint=False)
    if suid is None: return 0
    self.data["info"].pop(suid)
    self.data["score"].pop(suid)
    self.u = udk(self.data, lt[4:7])
    input("".ljust(19) + "删除学号为" + str(suid) + "的学生成功...")
    input("".ljust(19) + "按任意键继续...")

# 打印学生成绩情况
def printStudentScore(self, info, tb_name):
    print("".ljust(24) + tb_name.center(45, "*"))
    tb = pt(list(info.keys()))
    tb.border = False
    if tb_name in "总分" or tb_name in "平均分":
```

```

        tb.add_row(list(info.values()))
    else:
        tb.add_row([self.data["info"][str(val[1])]["姓名"] for val in info.values()]) # 先输出姓名,
再输出成绩
        tb.add_row([val[0] for val in info.values()])
    for s in tb.get_string().splitlines(): print("".ljust(37) + s)
    input("".ljust(23) + ">按任意键继续<".center(43))

# 学生成绩统计
def alaDataStudent(self):
    if self.isDataEmpty():
        print("".ljust(19) + "数据为空! ")
        return 0
    # 依次打印总分, 平均分, 最高分, 最低分
    self.printStudentScore(self.u.getSumScore(), "总分")
    self.printStudentScore(self.u.getArrScore(), "平均分")
    self.printStudentScore(self.u.getMaxScore(), "最高分")
    self.printStudentScore(self.u.getMinScore(), "最低分")

# 打印一行学生信息
def printStudentInfoLine(self, suid, flag=False):
    table = pt(lt)
    table.align[lc[1]] = "l"
    row = [suid] + list(self.data["info"][suid].values()) + list(self.data["score"][suid].values()) + \
        [self.data["tot_score"][suid]]
    table.add_row(row)
    # 去掉边框, 因为边框存在布局错误, 再转换为字符串逐行对齐输出
    table.border = False
    # 通过多次调用逐行打印学生成绩达到打印整个表的效果, 但是只能输出一次表头
    for s in table.get_string().splitlines():
        if flag: # 如果执行的是打印整个表, 则不会打印表头
            flag = not flag
            continue
        print("".ljust(19) + s)

# 选择学生信息
def selectStudent(self, isprint=True):
    suid = input("".ljust(19) + "请输入学生学号:")
    # 提供两种查询模式, 一种是打印学生信息, 一种是用来判断当前学生是否存在
    if suid not in self.data["info"].keys() or suid not in self.data["score"].keys():
        print("".ljust(19) + "该学生不存在! ")
        return None
    if not isprint: return suid # 如果是不打印模式这个方法做的就是查询下学生是否存在
    self.printStudentInfoLine(suid)

```



```

        input("".ljust(19) + "按任意键继续...")

# 数据可视化
def showDataStudent(self):
    if self.isDataEmpty():
        print("".ljust(19) + "数据为空! ")
        return 0
    showData(self.u, lt[4:7])

# 打印菜单
def printMenu():
    print("".rjust(20) + "欢迎使用学生成绩排序系统".center(45, "*"))
    # 对齐文本
    for i in range(len(menu)): print("".rjust(19) + (str(i + 1) + ". " + menu[i]).rjust(25 + len(menu[i])))
    print("".ljust(19) + "*" * 57)

def printPrompt():
    print("".ljust(19), end="")
    code = -1
    try:
        code = int(input("请输入操作码: "))
        # 非法数据
        if code < 1 or code > len(menu):
            code = -1
            print("".ljust(19) + "操作码非法! ")
    except ValueError:
        print("".ljust(19) + "请输入整形! ") # 非法输入
    return code

# 根据操作码执行对应操作
def runCode(code, student):
    if code == -1: return
    if code == len(menu):
        print("".ljust(18), "感谢使用!请选择是否保存数据!")
        student.updateDataStudent()
        exit(0)
    # student.*Student(code), 省去了很多个判断
    eval("student." + orderMap[code] + "Student()")

def main():
    # 因为在循环里无限实例化 Student 类导致全部操作都有问题
    student = Student()
    while True:
        printMenu()

```

```
runCode(printPrompt(), student=student)
```

```
if __name__ == '__main__':  
    main()
```

marina.py 代码

"""

作者: hwz

时间: 2020/5/12 13:01

功能: 简单汇总分析学生数据

"""

```
class Marina(object):
```

```
    def __init__(self, data, course_list):
```

```
        self.data = data
```

```
        self.sum_dict = {}
```

```
        self.max_dict = {}
```

```
        self.min_dict = {}
```

```
        for i in course_list:
```

```
            self.sum_dict[i] = .0
```

```
            self.max_dict[i] = (.0, 0)
```

```
            self.min_dict[i] = (3777.0, 0)
```

```
        self.__cmpSumScore()
```

```
        self.__cmpTopScore()
```

```
# 计算每门成绩的总和，同时可以通过该字典获取平均分
```

```
def __cmpSumScore(self):
```

```
    for val in self.data["score"].values():
```

```
        for c in val:
```

```
            self.sum_dict[c] += float(val[c])
```

```
# 获取最高分最低分学生学号以及成绩
```

```
def __cmpTopScore(self):
```

```
    for tup in self.data["score"].items(): # 拿到成绩元组
```

```
        for cor in tup[1]: # 遍历元组里的 values, cor 是 key, 也就是课程名称
```

```
            # 依次对比成绩, 先放成绩, 后放学号
```

```
            if float(self.max_dict[cor][0]) < float(tup[1][cor]): self.max_dict[cor] =  
(float(tup[1][cor]), tup[0])
```

```
            if float(self.min_dict[cor][0]) > float(tup[1][cor]): self.min_dict[cor] =  
(float(tup[1][cor]), tup[0])
```

```
# 获取每门成绩的总和
```

```
def getSumScore(self):
```

```
    return self.sum_dict
```

```

# 获取每门成绩平均分，均分四舍五入保留两位小数
def getArrScore(self):
    return {i[0]: round(i[1] / len(self.data["score"]), 2) for i in self.sum_dict.items()}

def getMaxScore(self):
    return self.max_dict

def getMinScore(self):
    return self.min_dict

```

### show.py 代码

"""

作者: hwz

时间: 2020/5/13 19:33

功能: 数据可视化

"""

```

from matplotlib import pyplot as plt
def showData(data, course_list):
    print("\n.ljust(19) + "绘制表格中，请稍等...")
    plt.rcParams['font.sans-serif'] = ['SimHei'] # 用黑体显示中文
    tot_score = data.getSumScore()
    arr_score = data.getArrScore()
    max_score = data.getMaxScore()
    min_score = data.getMinScore()
    plt.figure(figsize=(19.80, 10.24))
    # 绘制总分表
    plt.subplot(2, 2, 1)
    plt.title("总分表")
    plt.xlabel("课程名称")
    plt.ylabel("总分数")
    plt.grid()
    vl = list(tot_score.values())
    plt.plot(course_list, vl, color="cornflowerblue", marker="o")
    for p in range(3): plt.text(p - 0.05, vl[p] + 19, vl[p], color="slategrey")
    # 绘制均分表
    plt.subplot(2, 2, 2)
    plt.title("均分表")
    plt.ylabel("课程名称")
    plt.xlabel("平均分")
    vl = list(arr_score.values())
    plt.barh(course_list, vl, color="cyan")
    for p in range(3): plt.text(vl[p] + 6, p - 0.1, vl[p], color="springgreen", rotation=270)
    # 绘制最高/低分表

```

```

plt.subplot(2, 1, 2)
plt.title("最高/低分表")
plt.xlabel("课程名称")
plt.ylabel("最高/低分值")
max_vl = [i[0] for i in max_score.values()]
min_vl = [i[0] for i in min_score.values()]
pos = [i for i in range(0, 3)]
wid = 0.25
max_bt = plt.bar(pos, max_vl, width=wid, color="deepskyblue")
for i in range(3): pos[i] += wid
min_bt = plt.bar(pos, min_vl, width=wid, color="red")
pos = [i + 0.1 for i in range(3)]
plt.xticks(pos, course_list)
for min_t in min_bt:
    height = min_t.get_height()
    plt.text(min_t.get_x() + wid / 2 - 0.01, height + 13, height, color="crimson")
for max_t in max_bt:
    height = max_t.get_height()
    plt.text(max_t.get_x() + wid / 2 - 0.01, height + 13, height, color="c")
plt.savefig("./table.png") # 得先保存，不然 plt 对象刷新了就只能保存空白图片了
plt.show()
input("\n.ljust(19) + "表格绘制完成！已保存到当前目录下的 table.png，请按任意键继续...")

```

# 致 谢

感谢王绪梅老师的指导，Google 搜索引擎的干净准确的搜索结果，以及那些将知识与经验汇聚成博文的博主们。

## 课程设计成绩评定表

课程设计 题目	学生成绩排名	
<p>课程设计学生答辩或质疑记录：</p> <p>问：如何判断学生是否存在。 答：根据数据表字典的键判断。</p> <p>问：学习过程中遇到了什么困难。 答：在完成数据可视化功能时遇到了类似不知道方法作用、参数列表等的困难。</p> <p>问：Python 与 Java 的优缺点。 答：Python 语言较为简洁，适合做爬虫、数据分析，但无法对源文件加密，对格式要求过高。Java 语言较前者更严谨，适合用来做后台开发，但代码较为繁琐。</p>		
评 分 依 据	分 值	评分成绩
1. 选题完成程度	10 分	
2. 态度认真、学习刻苦、遵守纪律	10 分	
3. 设计方案正确，具有可行性	20 分	
4. 独立性与创新性	5 分	
5. 系统调试与结果	20 分	
6. 参考文献充分（不少于 5 篇）	5 分	
7. 设计报告撰写规范	10 分	
8. 答辩	20 分	
总 分	100 分	
<p>最终评定等级为：</p> <p style="text-align: right;">指导老师签字：</p> <p style="text-align: right;">年    月    日</p>		

注明：1 答辩记录至少应为 3 个；2.最终评定成绩（以优、良、中、及格、不及格评定）