华南農業大學

数据结构与算法程序设计报告

姓 名：邱森

班 级：20 人工智能 2

学 号：20202034610216

指导老师：徐兴，胡洁，陆建强

日期：2021.12.10～2021.1.18

华南农业大学电子工程/人工智能学院

摘　　要

本次课程设计我的选题是选题 1：学生成绩管理系统，首先可以完成基本要求的所有功能（如成绩的录入、按班级统计学生成绩、求总分以及平均分、平均分排序，查询成绩、输出成绩单、按学号删除、存储信息……）。且经过大量测试、改进、优化后程序 bug 出现概率几乎为 0。创新性功能也十分多样，如班级排名，年级排名，单科排名，分数段统计……在查询上，学号和姓名任意输入查询，而且非常快速，排序方法上也十分多样，体现在不同科目或总分排序运用了不同的排序方法，包括希尔排序、快速排序、选择排序、折半排序、插入排序以及冒泡排序。在数据结构上采用独特的链表式存储，添加操作有前插和尾插两种，对内存管理十分优秀，解析外部文件、链表信息的程序也十分稳定，程序的最开始就能迅速地将 json 文件中的数据进行解析，存入临时链表中。其中代码结构有条理，逻辑性，整篇代码 0 警告，运用 2 大类，多达 28 个函数 746 行代码，有的函数调用其他函数，代码规范，同时运用 python 第三方包 collections 和 json，在存储上的不论是观赏性或是解析难度都很优秀，同时学生成绩中，不同科目或其学生属性值的合法性也可以得到保证，所以作者程序稳定性相较于其他管理系统有充足的信心，函数间有相应的注释，可读性强，且程序在进行中有相应输出指导用户使用，信息量足。链表看似引用复杂，但作者结合列表，类列表，字典，类字典对排序、查询、存储操作便可以顺利实现。

目　　录

目录

1. 课程设计目的及内容

目的：设计一个学生成绩管理系统，能用于学生成绩信息的录入跟统计。

内容：（1）能按学期、按班级完成对学生成绩的录入。

（2）能按班级统计学生的成绩，求学生的总分及平均分，并能根据学生的平均成绩进行排序。

（3）能查询学生成绩，不及格科目及学生名单。

（4）能按班级输出学生的成绩单。

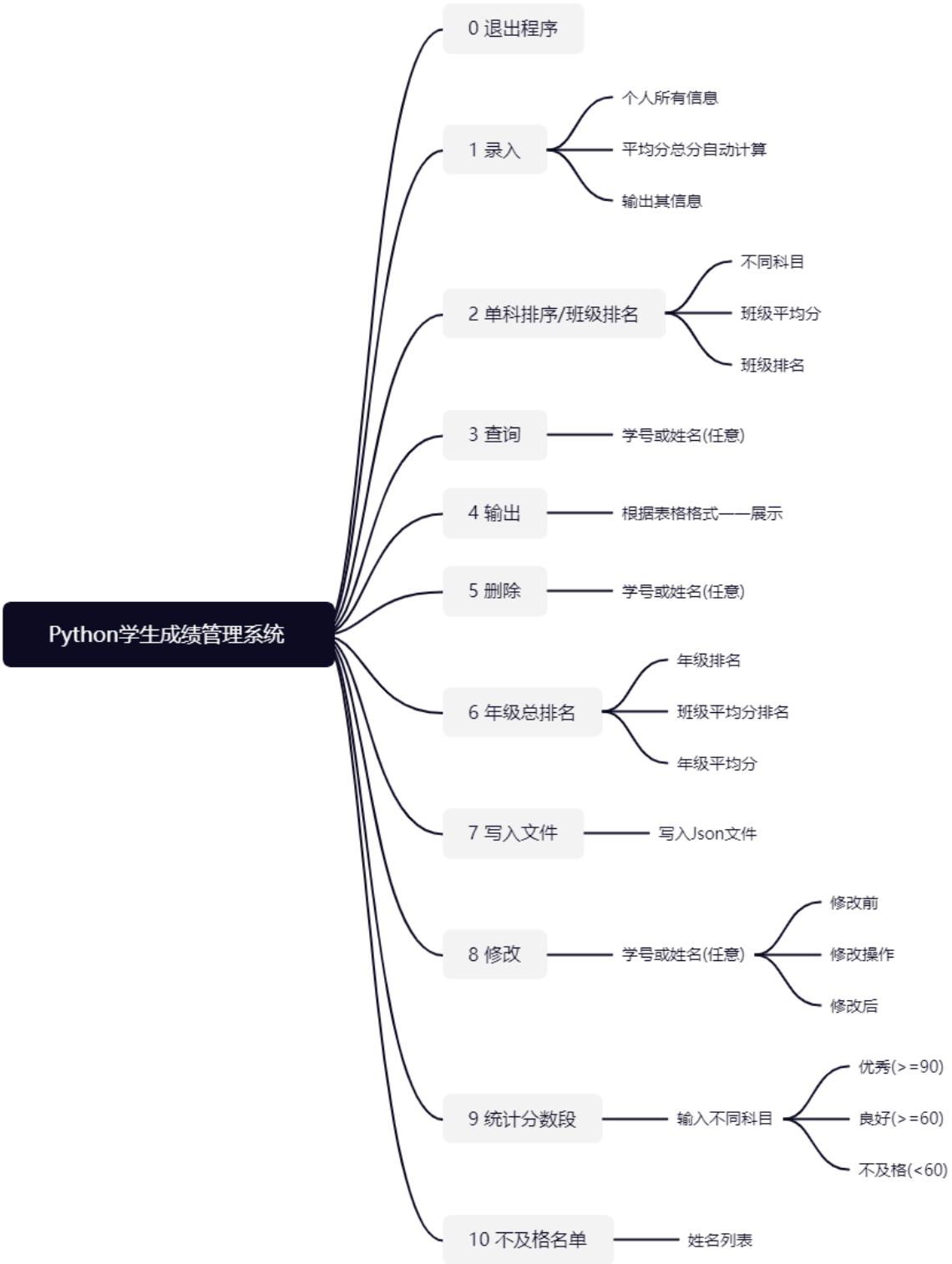
（5）能按学号删除学生记录，成绩等。

（6）将学生的学号、姓名及各门课程的成绩等信息保存于外部存储器的文件中。

增加新的功能比如班级总排名，单科成绩排名，各分数段学生统计等，使用多种方法排序，读取外部文件，解析学生信息。

2.程序设计

2.1 程序流程图



2.2 分模块程序及算法设计

一、导入第三方包

1. json：写入以及读取学生信息，存储方面优良，可对文件进行加密操作，解析性好。
2. collections.defaultdict ：写入以及读取时解析格式，类字典方式，便于存取多个多属性数据。

```
from collections import defaultdict
import json
```

二、指针元素类定义

类定义

```
# 指针元素类
class itemNode(object):
```

1. 初始化 学期, 班级, 学号信息

```
    # 初始化 学期, 班级, 学号信息
    def __init__(self, Term, Class, Number):
        self.Term = Term
        self.Class = Class
        self.Number = Number
        self.Name = None
        self.A = 0
        self.B = 0
        self.C = 0
        self.Total = 0
        self.Aver = 0
        self.Rank = 0
        self.RankA = 0
        self.RankB = 0
        self.RankC = 0
        self.next = None
```

2. 输入操作输入姓名、A、B、C 成绩，自动计算总分、平均分

```
    # 输入操作
    def TypeIn(self):
        Name = input("输入姓名: ")
        self.Name = Name
        try:
            Ag = int(input("输入 A 成绩: "))
        except ValueError:
            Ag = int(input("请输入正确格式的 A 成绩: "))
        try:
            Bg = int(input("输入 B 成绩: "))
        except ValueError:
            Bg = int(input("请输入正确格式的 B 成绩: "))
        try:
            Cg = int(input("输入 C 成绩: "))
        except ValueError:
            Cg = int(input("请输入正确格式的 C 成绩: "))
        self.A = Ag
        self.B = Bg
        self.C = Cg
        self.Total = Ag + Bg + Cg
        self.Aver = int((Ag + Bg + Cg) / 3)
```

3. 修改 输入对应操作进行相应修改

```
    # 修改
    def Amend(self):
        print("操作\t\t 项目\nname\t 姓名\nnum\t\t 学号\nclass\t 班级\nterm\t
学期\nA\t\tA\nB\t\tB\nC\t\tC")
        which = input("您想要修改什么？: ")
```

```
        if which.isalpha():
            if which == 'name':
                Namenew = input("姓名输入: ")
                self.Name = Namenew
            elif which == 'num':
                Numnew = input("学号输入: ")
                self.Number = Numnew
            elif which == 'class':
                Classnew = input("班级输入: ")
                self.Class = Classnew
            elif which == 'term':
                Termnew = input("学期输入: ")
                self.Term = Termnew
            elif which == 'A':
                try:
                    Anew = int(input("成绩输入: "))
                except ValueError:
                    Anew = int(input("请输入合法成绩: "))
                self.A = Anew
            elif which == 'B':
                try:
                    Bnew = int(input("成绩输入: "))
                except ValueError:
                    Bnew = int(input("请输入合法成绩: "))
                self.B = Bnew
            elif which == 'C':
                try:
                    Cnew = int(input("成绩输入: "))
                except ValueError:
                    Cnew = int(input("请输入合法成绩: "))
```

```
                self.C = Cnew
            else:
                print("您输入的操作科目不存在")
        else:
            print("您输入的操作科目不存在")
        self.Total = self.A + self.B + self.C
        self.Aver = int(self.Total / 3)
```

4. 输出操作 输出所有学生信息 包括平均分、总分

```
    # 输出操作
    def Check(self):
        print("姓名： {}".format(self.Name))
        print("学号：{}".format(self.Number))
        print("班级：{}".format(self.Class))
        print("学期：{}".format(self.Term))
        print("A 成绩：{}".format(self.A))
        print("B 成绩：{}".format(self.B))
        print("C 成绩：{}".format(self.C))
        print("总成绩：{}".format(self.Total))
        print("平均分：{}".format(self.Aver))
```

三、链表类及其操作

链表类定义

1. 初始化

```
# 链表
    def __init__(self):
        # 初始化链表大小, 而不是创建
        self.head = None  # 首地址指针为 None
```

2. 判断空链表

```
    # 判断空链表
    def Empty(self):
        return self.head is None
```

3. 解析先前的数据 加入链表

```
    # 解析先前的数据 加入链表
    def Analysis(self):
        try:
            with open('record.json', 'r', encoding='utf-8') as ExJsonFile:
                ExAll = json.load(ExJsonFile)
                ExJsonFile.close()
            for ExData in ExAll.values():
                # ExAll.values() : dict_values([[{}]])
                # ExData : [{}]
                for ExDict in ExData:
                    # ExDict : {}
                    ExElemDataList = []
                    for ExElemData in ExDict.values():
                        # ExElemData : 所有属性数据 value
                        ExElemDataList.append(ExElemData)
                    TempTerm = ExElemDataList[0]
                    TempClass = ExElemDataList[1]
                    TempName = ExElemDataList[2]
                    TempNum = ExElemDataList[3]
                    TempA = ExElemDataList[4]
                    TempB = ExElemDataList[5]
                    TempC = ExElemDataList[6]
                    TempNode = itemNode(TempTerm, TempClass, TempNum)
                    TempNode.Name = TempName
                    TempNode.A = TempA
```

```
                    TempNode.B = TempB
                    TempNode.C = TempC
                    TempNode.Total = TempA + TempB + TempC
                    TempNode.Aver = int(TempNode.Total / 3)
                    TempNode.next = self.head
                    self.head = TempNode
        except json.decoder.JSONDecodeError:
            return
```

4. 遍历

```
    # 遍历
    def Item(self):
        cursor = self.head
        ElemDict = defaultdict(list)
        while cursor is not None:
            Elem = {
                '学期': cursor.Term,
                '班级': cursor.Class,
                '姓名': cursor.Name,
                '学号': cursor.Number,
                'A': cursor.A,
                'B': cursor.B,
                'C': cursor.C,
                '平均分': cursor.Aver,
                '总分': cursor.Total
            }
            ElemDict["成绩"].append(Elem)
            cursor = cursor.next
        return ElemDict
```

5. 表格形式显示

```
    # 表格形式显示
    # noinspection PyMethodMayBeStatic
    def PrintItem(self, ElemDict):
        Header = ['学期', '班级', '姓名', '学号', 'A', 'B', 'C', '平均', '总分']
        print('序号', end='\t' * 2)
        for HeaderElem in Header:
            print(HeaderElem, end='\t' * 3)
        print()
        # ElemDict : defaultdict(<class 'list'>, {})
        for i in ElemDict.values():
            # ElemDict.values() : dict_values([[{}]])
            Rank = 1
            for j in i:
                print(Rank, end='\t' * 2)
                Rank += 1
                # i : [{}]    j : {}
                for k in j.values():
                    # k : values()
                    print(k, end='\t' * 3)
                print()
```

6. 头部添加元素

```
    # 头部添加元素
    def Add(self, ATerm, AClass, ANumber):
        cursor = self.search(ANumber)
        if cursor == 0:
            NewNode = itemNode(ATerm, AClass, ANumber)
            NewNode.TypeIn()
```

```
            # 前插法
            NewNode.next = self.head
            self.head = NewNode
            NewNode.Check()
        else:
            cursor.Check()
            print("该学号学生已经存在")
```

7. 尾部添加元素

```
    # 尾部添加元素
    def Append(self, ATerm, AClass, ANumber):
        NewNode = itemNode(ATerm, AClass, ANumber)
        NewNode.TypeIn()
        if self.Empty():
            self.head = NewNode
        else:
            cursor = self.head
            while cursor.next is not None:
                cursor = cursor.next
            cursor.next = NewNode
```

8. 删除元素

```
    # 删除元素
    def Del(self, DIndex):
        cursor = self.head
        tem = None
        # 寻找
        while cursor is not None:
            if cursor.Name == DIndex or str(cursor.Number) == str(DIndex):
                # 如果第一个就是删除的节点
```

```
                if not tem:
                    # 头指针指向头指针的后一个节点
                    self.head = cursor.next
                else:
                    # 删除位置前一个指向其删除节点的 next
                    tem.next = cursor.next
                cursor.Check()
                return "OK"
            else:
                # 后移节点
                tem = cursor
                cursor = cursor.next
        return "未找到对应学生"
```

9. 提取数据 并快速排序班级内学生总分 计算班级平均分

```
    # 提取数据 并快速排序班级内学生总分 计算班级平均分
    def Extract(self, EClass):
        cursor = self.head
        NumList = []
        TotalList = []
        while cursor is not None:
            if EClass == cursor.Class:
                Number = cursor.Number
                Total = cursor.Total
                NumList.append(Number)
                TotalList.append(Total)
            cursor = cursor.next
        if not TotalList:
            return 0, 0
        Dict = dict(zip(NumList, TotalList))
```

```
        key = [value for value in Dict.values()]
        try:
            ClassAver = int(sum(key) / (len(key)))
        except ZeroDivisionError:
            return [], 0
        Quicksort(key, 0, len(key) - 1)
        key.reverse()
        ElemDict = defaultdict(list)
        for i in key:
            RankElem = [key for key, value in Dict.items() if value == i]
            RankNum = RankElem[0]
            RankPeople = self.search(RankNum)
            Elem = {
                '学期': RankPeople.Term,
                '班级': RankPeople.Class,
                '姓名': RankPeople.Name,
                '学号': RankPeople.Number,
                'A': RankPeople.A,
                'B': RankPeople.B,
                'C': RankPeople.C,
                '平均分': RankPeople.Aver,
                '总分': RankPeople.Total
            }
            ElemDict["成绩"].append(Elem)
        return ClassAver, ElemDict
```

10. 提取数据 并希尔排序 计算平均分 (所有人)

```
    # 提取数据 并希尔排序 计算平均分 (所有人)
    def ExtractAll(self):
        cursor = self.head
```

```
        if cursor is None:
            return [], 0
        NumList = []
        TotalList = []
        while cursor is not None:
            Num = cursor.Number
            Total = cursor.Total
            NumList.append(Num)
            TotalList.append(Total)
            cursor = cursor.next
        Dict = dict(zip(NumList, TotalList))
        key = [value for value in Dict.values()]
        ClassAver = int(sum(key) / (len(key)))
        ShellSort(key)
        key.reverse()
        ElemDict = defaultdict(list)
        for i in key:
            RankElem = [key for key, value in Dict.items() if value == i]
            RankNum = RankElem[0]
            RankPeople = self.search(RankNum)
            Elem = {
                '学期': RankPeople.Term,
                '班级': RankPeople.Class,
                '姓名': RankPeople.Name,
                '学号': RankPeople.Number,
                'A': RankPeople.A,
                'B': RankPeople.B,
                'C': RankPeople.C,
                '平均分': RankPeople.Aver,
                '总分': RankPeople.Total
```

```
            }
            ElemDict["成绩"].append(Elem)
        return ClassAver, ElemDict
```

11. 提取数据 快速排序进行班级平均分排序

```
    # 提取数据 快速排序进行班级平均分排序
    def ExtractClass(self):
        ClassList = []
        GradeAver = {}
        cursor = self.head
        while cursor is not None:
            ClassElem = cursor.Class
            if ClassElem not in ClassList:
                ClassList.append(ClassElem)
            cursor = cursor.next
        for i in ClassList:
            AverElem, _ = self.Extract(i)
            iElem = {i: AverElem}
            GradeAver.update(iElem)
        key = [value for value in GradeAver.values()]
        Quicksort(key, 0, len(key) - 1)
        key.reverse()
        res = []
        for j in key:
            RankElem = [key for key, value in GradeAver.items() if value ==
j]
            RankElem.append(j)
            res.append(RankElem)
        for Rank in range(len(res)):
            res[Rank].append(Rank + 1)
```

```
            res[Rank].reverse()
        return res
```

12. 提取数据 并冒泡排序(年级单科 A)

```
    # 提取数据 并冒泡排序(年级单科 A)
    def ASort(self):
        cursor = self.head
        NameList = []
        TotalList = []
        while cursor is not None:
            Name = cursor.Name
            ATem = cursor.A
            NameList.append(Name)
            TotalList.append(ATem)
            cursor = cursor.next
        Dict = dict(zip(NameList, TotalList))
        key = [value for value in Dict.values()]
        key = Maopao(key)
        key.reverse()
        res = []
        for i in key:
            RankElem = [key for key, value in Dict.items() if value == i]
            RankElem.append(i)
            res.append(RankElem)
        for Rank in range(len(res)):
            res[Rank].append(Rank + 1)
            res[Rank].reverse()
        return res
```

13. 提取数据 并选择排序(年级单科 B)

```
    # 提取数据 并选择排序(年级单科 B)
    def BSort(self):
        cursor = self.head
        NameList = []
        TotalList = []
        while cursor is not None:
            Name = cursor.Name
            BTem = cursor.B
            NameList.append(Name)
            TotalList.append(BTem)
            cursor = cursor.next
        Dict = dict(zip(NameList, TotalList))
        key = [value for value in Dict.values()]
        key = SelectSort(key)
        key.reverse()
        res = []
        for i in key:
            RankElem = [key for key, value in Dict.items() if value == i]
            RankElem.append(i)
            res.append(RankElem)
        for Rank in range(len(res)):
            res[Rank].append(Rank + 1)
            res[Rank].reverse()
        return res
```

14. 提取数据 并插入排序(年级单科 C)

```
    # 提取数据 并插入排序(年级单科 C)
    def CSort(self):
        cursor = self.head
        NameList = []
```

```
        TotalList = []
        while cursor is not None:
            Name = cursor.Name
            CTem = cursor.C
            NameList.append(Name)
            TotalList.append(CTem)
            cursor = cursor.next
        Dict = dict(zip(NameList, TotalList))
        key = [value for value in Dict.values()]
        key = InsertSort(key)
        key.reverse()
        res = []
        for i in key:
            RankElem = [key for key, value in Dict.items() if value == i]
            RankElem.append(i)
            res.append(RankElem)
        for Rank in range(len(res)):
            res[Rank].append(Rank + 1)
            res[Rank].reverse()
        return res
```

15. 写入操作

```
    # 写入操作
    def WriteIn(self):
        resDict = self.Item()
        with open('record.json', 'w', encoding='utf-8', newline='\n') as
JsonFile:
            JsonStr = json.dumps(resDict, indent=4, ensure_ascii=False)
            print(JsonStr)
            JsonFile.write(JsonStr)
```

```
            JsonFile.close()
        return "成绩写入成功"
```

16. 不及格名单

```
    # 不及格名单
    def Fail(self):
        cursor = self.head
        FailList = []
        FailA = []
        FailB = []
        FailC = []
        while cursor is not None:
            if cursor.A < 60:
                FailList.append(cursor.Name)
                FailA.append(cursor.Name)
            elif cursor.B < 60:
                FailB.append(cursor.Name)
                FailList.append(cursor.Name)
            elif cursor.C < 60:
                FailList.append(cursor.Name)
                FailC.append(cursor.Name)
            cursor = cursor.next
        return FailList, FailA, FailB, FailC
```

17. 统计分数段

```
    # 统计分数段
    def Sat(self, subject):
        cursor = self.head
        Great = 0
        Good = 0
```

```
        Failed = 0
        if subject == 'A':
            while cursor is not None:
                if cursor.A >= 90:
                    Great += 1
                elif cursor.A >= 60:
                    Good += 1
                else:
                    Failed += 1
                cursor = cursor.next
        elif subject == 'B':
            while cursor is not None:
                if cursor.B >= 90:
                    Great += 1
                elif cursor.B >= 60:
                    Good += 1
                else:
                    Failed += 1
                cursor = cursor.next
        elif subject == 'C':
            while cursor is not None:
                if cursor.C >= 90:
                    Great += 1
                elif cursor.C >= 60:
                    Good += 1
                else:
                    Failed += 1
                cursor = cursor.next
        else:
            return "ERROR SUBJECT"
```

```
        res = {
            '优秀': Great,
            '良好': Good,
            '不及格': Failed
        }
        return res
```

18. 查找学生

```
    # 查找学生
    def search(self, sIndex):
        cursor = self.head
        while cursor is not None:
            if str(cursor.Name) == str(sIndex) or str(cursor.Number) ==
str(sIndex):
                return cursor
            cursor = cursor.next
        # print("ERROR" * 100)
        return 0
```

四、排序或查询算法

1. 快速排序函数

```
# 快速排序函数
def Quicksort(qList, start, end):
    if start >= end:  # 递归的退出条件
        return
    mid = qList[start]  # 设定起始的基准元素
    low = start  # low 为序列左边在开始位置的由左向右移动的游标
    high = end  # high 为序列右边末尾位置的由右向左移动的游标
    while low < high:
        # 如果 low 与 high 未重合，high(右边)指向的元素大于等于基准元素，则 high
```

向左移动

```
        while low < high and qList[high] >= mid:
            high -= 1
        qList[low] = qList[high]  # 走到此位置时 high 指向一个比基准元素小的元素,将 high 指向的元素放到 low 的位置上,此时 high 指向的位置空着,接下来移动 low 找到符合条件的元素放在此处
        # 如果 low 与 high 未重合，low 指向的元素比基准元素小，则 low 向右移动
        while low < high and qList[low] < mid:
            low += 1
        qList[high] = qList[low]  # 此时 low 指向一个比基准元素大的元素,将 low 指向的元素放到 high 空着的位置上,此时 low 指向的位置空着,之后进行下一次循环,将 high 找到符合条件的元素填到此处


    # 退出循环后，low 与 high 重合，此时所指位置为基准元素的正确位置,左边的元素都比基准元素小,右边的元素都比基准元素大
    qList[low] = mid  # 将基准元素放到该位置,
    # 对基准元素左边的子序列进行快速排序
    Quicksort(qList, start, low - 1)  # start :0  low -1 原基准元素靠左边一位
    # 对基准元素右边的子序列进行快速排序
    Quicksort(qList, low + 1, end)  # low+1 : 原基准元素靠右一位  end: 最后
```

2. 折半查找函数

```
# 折半查找函数
def Zheban(arr, start, end, data):
    while start <= end:
        half = int((start + end) / 2)
        if arr[half] == data:
            return half
        elif arr[half] > data:
            end = half - 1
```

```
        else:
            start = half + 1
    # 为了防止找不到
    return -1
```

3. 冒泡排序函数

```
# 冒泡排序函数
def Maopao(MPList):
    while True:
        flag = 0
        for i in range(len(MPList) - 1):
            if MPList[i] > MPList[i + 1]:
                MPList[i], MPList[i + 1] = MPList[i + 1], MPList[i]
                flag = 1
        if not flag:
            return MPList
```

4. 插入排序函数

```
# 插入排序函数
def InsertSort(InList):
    resList = [InList[0]]
    for i in InList[1:]:
        flag = 1
        for j in range(len(resList) - 1, -1, -1):
            if i >= resList[j]:
                resList.insert(j + 1, i)
                flag = 0
                break
        if flag:
            resList.insert(0, i)
```

```
    return resList
```

5. 希尔排序函数

```
# 希尔排序
def ShellSort(ShList):  # d 为乱序数组,l 为初始增量,其中 l<len(d),取为 len(d)/2 比较好操作。最后还是直接省略 length 输入
    if len(ShList) == (1 or 0):
        return
    length = int(len(ShList) / 2)  # 10
    num = int(len(ShList) / length)  # 2
    while 1:
        for i in range(length):
            ShList_mid = []
            for j in range(num):
                ShList_mid.append(ShList[i + j * length])
            ShList_mid = InsertSort(ShList_mid)
            for j in range(num):
                ShList[i + j * length] = ShList_mid[j]
        length = int(length / 2)
        if length == 0:
            return ShList
        num = int(len(ShList) / length)
```

6. 选择排序

```
# 选择排序
def SelectSort(SeList):
    res = []
    while len(SeList):
        minT = [0, SeList[0]]
        for i in range(len(SeList)):
```

```
            if minT[1] > SeList[i]:
                minT = [i, SeList[i]]
        del SeList[minT[0]]  # 找到剩余部分的最小值，并且从原数组中删除
        res.append(minT[1])  # 在新数组中添加
    return res
```

五、主函数程序

先解析先前数据并生成链表，再根据操作来执行相应算法

```
if __name__ == '__main__':
    StuLink = LinkList()
    StuLink.Analysis()
    while True:
        print("*" * 100)
        print("输入\t 操作")
        print("0\tEXIT\n1\t 录入\n2\t 单科排序/班级排名\n3\t 查询\n4\t 输出\n5\t 删除\n6\t 年级总排名/各班级平均分\n7\t 写入文件\n8\t 修改\n9\t 统计分数段\n10\t 不及格名单")
        try:
            oper = int(input("请输入您想进行的操作: "))
            print("*" * 43 + "PROGRAM START" + "*" * 44)
            if oper == 1:
                flag1 = 'y'
                while flag1 == 'y':
                    print('*' * 48 + '录入' + '*' * 49)
                    T = input("学期: ")
                    C = input("班级: ")
                    try:
                        N = int(input("学号: "))
                    except ValueError:
                        N = int(input("请输入合法学号: "))
                    StuLink.Add(T, C, N)
```

```
                print("*" * 43 + "PROGRAM FINISH" + "*" * 43)
                flag1 = input("是否继续录入(y/n): ")
        if oper == 2:
            flag2 = 'y'
            while flag2 == 'y':
                print("A 科目 - A\nB 科目 - B\nC 科目 - C\n 班级平均分 班级
排名 - S")
                whichData = input("您想查看什么排序? ")
                if whichData == 'A':
                    print("排名\t\t 成绩\t\t 姓名")
                    for StuElem2 in StuLink.ASort():
                        for value2 in StuElem2:
                            print(value2, end='\t\t')
                        print()
                    print("*" * 43 + "PROGRAM FINISH" + "*" * 43)
                elif whichData == 'B':
                    print("排名\t\t 成绩\t\t 姓名")
                    for StuElem2 in StuLink.BSort():
                        for value2 in StuElem2:
                            print(value2, end='\t\t')
                        print()
                    print("*" * 43 + "PROGRAM FINISH" + "*" * 43)
                elif whichData == 'C':
                    print("排名\t\t 成绩\t\t 姓名")
                    for StuElem2 in StuLink.CSort():
                        for value2 in StuElem2:
                            print(value2, end='\t\t')
                        print()
                    print("*" * 43 + "PROGRAM FINISH" + "*" * 43)
                elif whichData == 'S':
```

```
                Class_2 = input("查询哪一个班级？ ")
                aver2, Dict2 = StuLink.Extract(Class_2)
                if aver2 == 0:
                    print("不存在该班级")
                    continue
                print("班级排名:")
                StuLink.PrintItem(Dict2)
                print("班级平均分: {}".format(aver2))
                print("*" * 43 + "PROGRAM FINISH" + "*" * 43)
            else:
                print("输入功能违法!")
                print("*" * 43 + "PROGRAM ERROR" + "*" * 44)
            flag2 = input("是否继续?(y/n) ")
    if oper == 3:
        flag3 = 'y'
        while flag3 == 'y':
            Cha3 = input("请输入学号或姓名: ")
            res3 = StuLink.search(Cha3)
            if res3 != 0:
                res3.Check()
            print("*" * 43 + "PROGRAM FINISH" + "*" * 43)
            flag3 = input("是否继续?(y/n) ")
    if oper == 4:
        StuLink.PrintItem(StuLink.Item())
        print("*" * 43 + "PROGRAM FINISH" + "*" * 43)
        continue
    if oper == 5:
        flag5 = 'y'
        while flag5 == 'y':
            Cha3 = input("请输入学号或姓名: ")
```

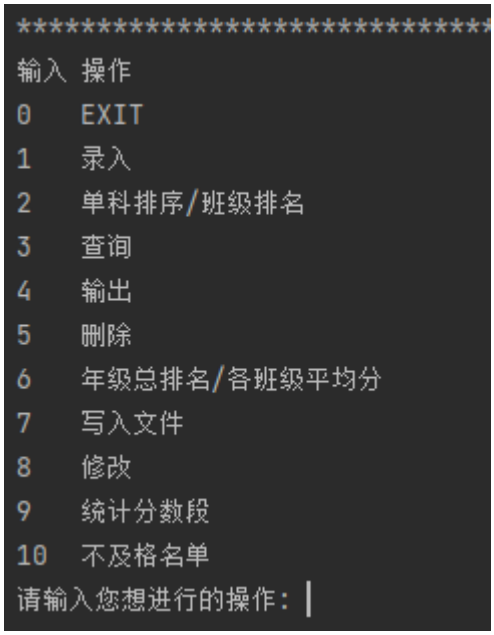
```
                    print(StuLink.Del(Cha3))
                    print("*" * 43 + "PROGRAM FINISH" + "*" * 43)
                    flag5 = input("是否继续删除操作?(y/n) ")
            if oper == 6:
                aver6, Dict6 = StuLink.ExtractAll()
                print("年级排名:")
                StuLink.PrintItem(Dict6)
                print("班级平均分排名: {}\n年级平均分: {}".format(StuLink.ExtractClass(), aver6))
                print("*" * 43 + "PROGRAM FINISH" + "*" * 43)
                continue
            if oper == 7:
                print(StuLink.WriteIn())
                continue
            if oper == 8:
                flag8 = 'y'
                while flag8 == 'y':
                    oper8 = input("请输入学号或姓名: ")
                    cur8 = StuLink.search(oper8)
                    if cur8 != 0:
                        print("修改前: ")
                        cur8.Check()
                        print("修改:")
                        cur8.Amend()
                        print("修改后:")
                        cur8.Check()
                        print("*" * 43 + "PROGRAM FINISH" + "*" * 43)
                    flag8 = input("是否继续?(y/n) ")
            if oper == 9:
                flag9 = 'y'
```

```
            while flag9 == 'y':
                oper9 = input("请输入科目: ")
                res9 = StuLink.Sat(oper9)
                print(res9)
                print("*" * 43 + "PROGRAM FINISH" + "*" * 43)
                flag9 = input("是否继续?(y/n) ")
        if oper == 10:
            FailAll, FailA10, FailB10, FailC10 = StuLink.Fail()
            print("所有不及格名单： ")
            for All in FailAll:
                print(All, end='\t')
            print()
            print("A 不及格名单： ")
            for All in FailA10:
                print(All, end='\t')
            print()
            print("B 不及格名单： ")
            for All in FailB10:
                print(All, end='\t')
            print()
            print("C 不及格名单： ")
            for All in FailC10:
                print(All, end='\t')
            print()
            continue
        if oper == 0:
            print("EXITING......")
            print('*' * 42 + 'EXIT THE PROGRAM' + '*' * 42)
            break
        else:
```

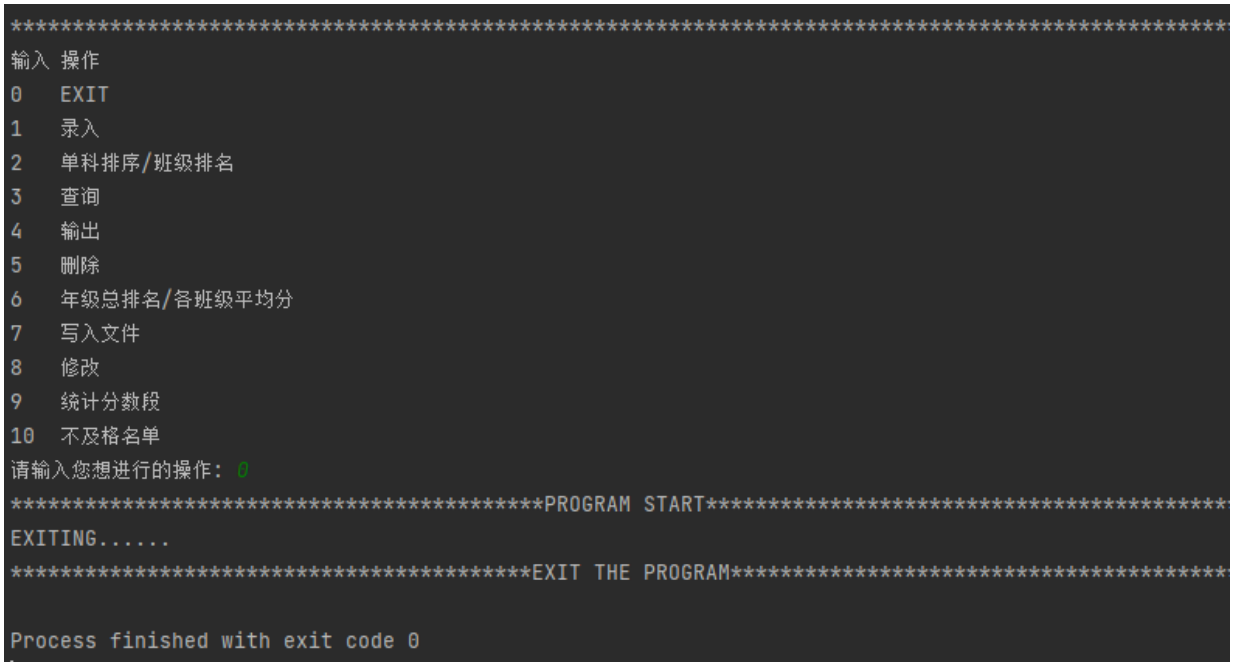
```
                print("*" * 43 + "ERROR OPERATION" + '*' * 42)
        except ValueError:
            print("*" * 43 + "ERROR OPERATION" + '*' * 42)
            continue
```

3 程序调试

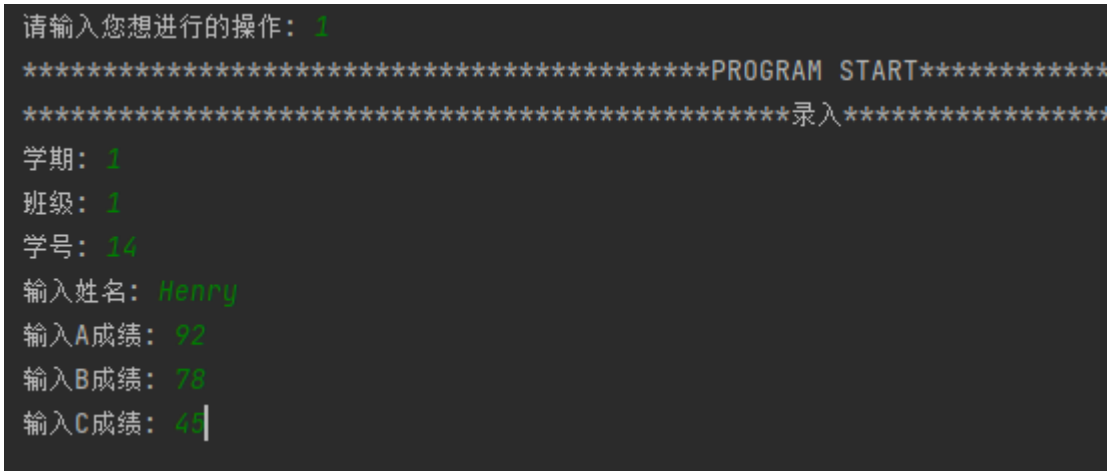
1. 最开始程序运行界面 菜单栏

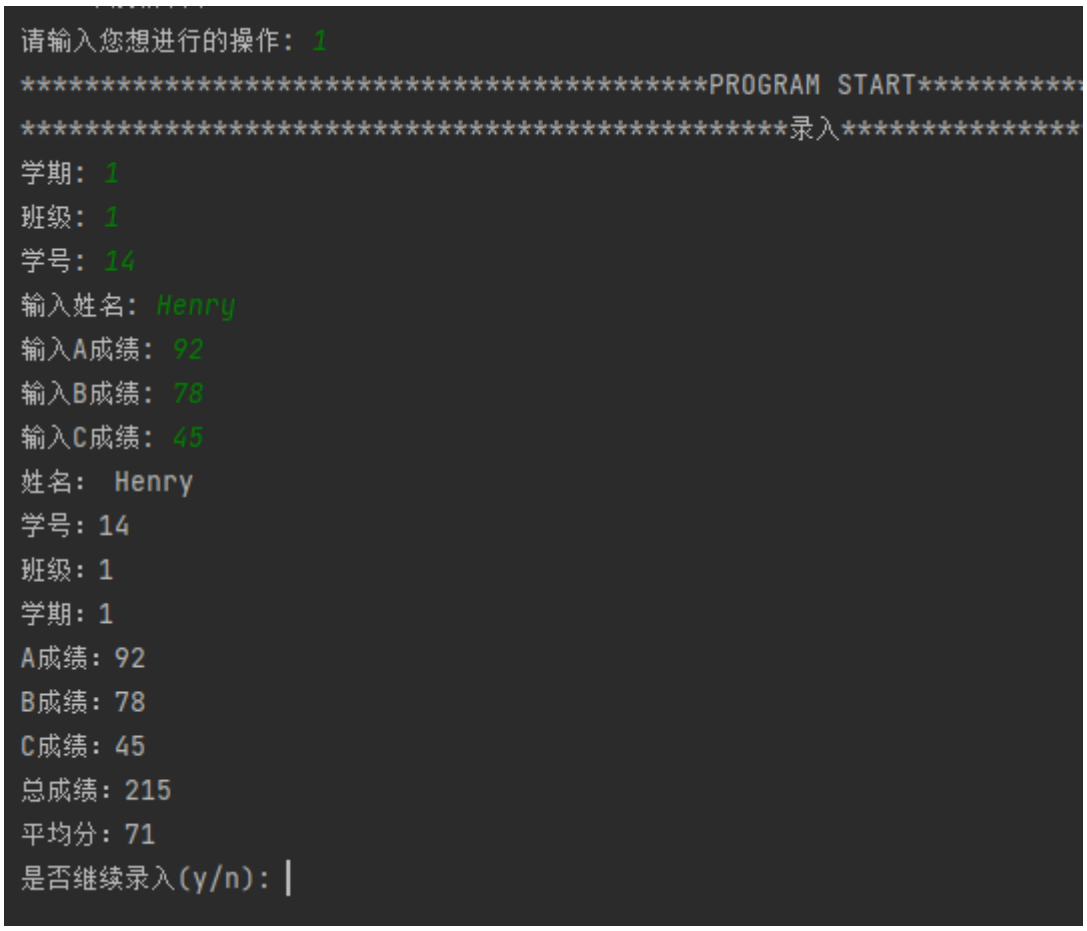


2. 输入 0 退出程序

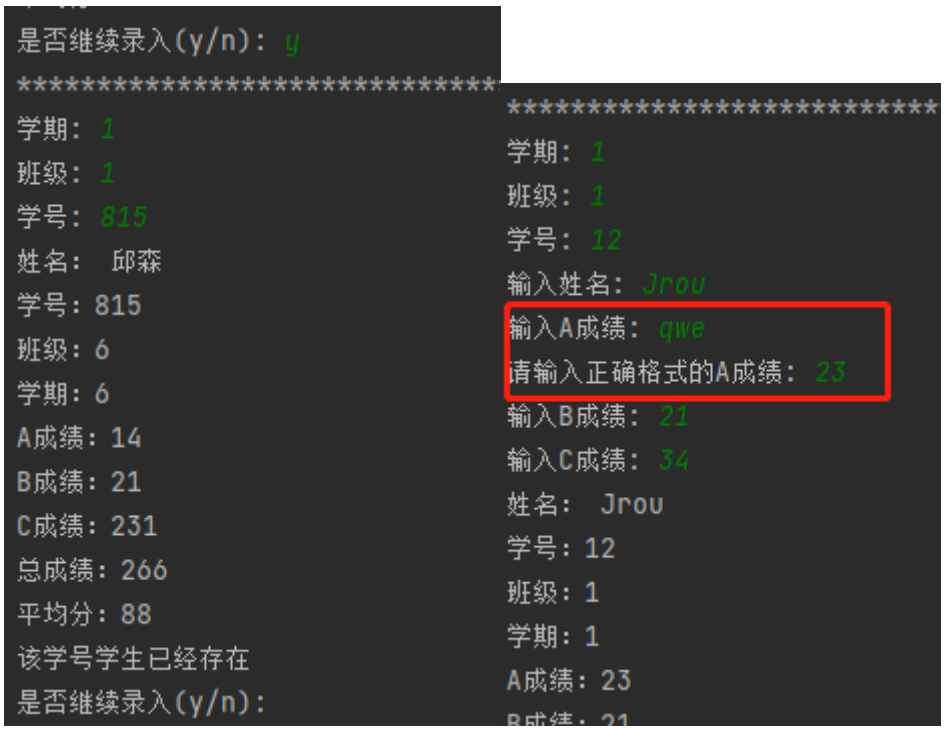


3. 输入 1 学生信息录入操作 （绿色字符为输入字符）

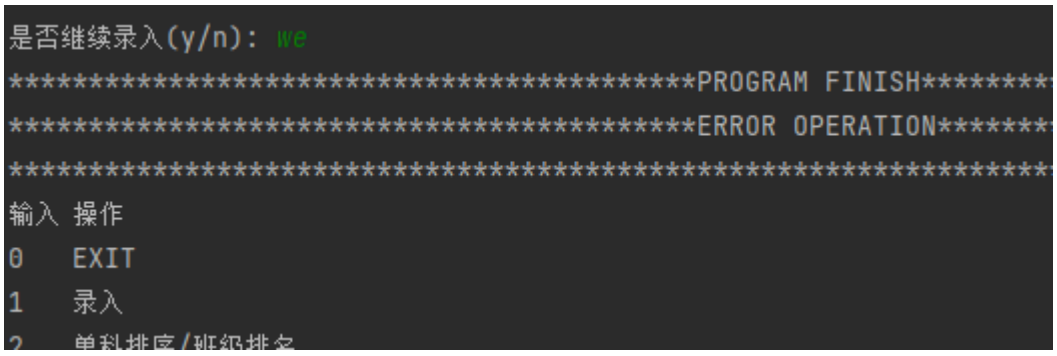




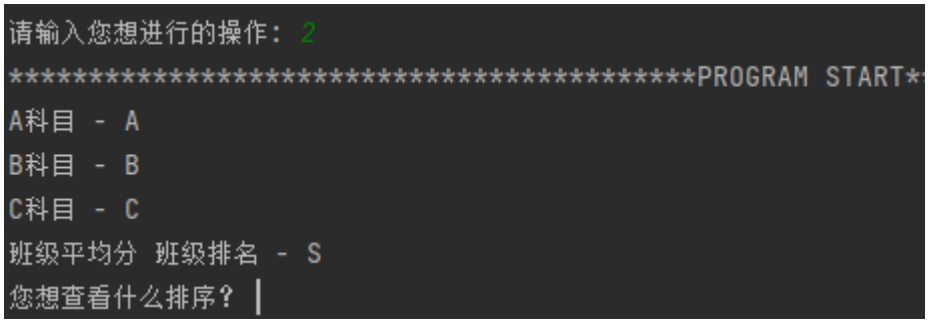
学号中输入已经存在的学号 成绩中输入非法字符：



退出该操作程序：



4. 输入 2 查看单科排序/班级排名



一一科目展示： （同分放在同行）

```
**********************************************PROGRAM START*****
A科目 - A
B科目 - B
C科目 - C
班级平均分 班级排名 - S
您想查看什么排序？ A
排名      成绩      姓名
1         92        Henry
2         76        梅东
3         23        Jrou
4         14        邱森
5         1         C罗
**********************************************PROGRAM FINISH****
```

```
**********************************************PROGRAM FINISH**
是否继续?(y/n) y
A科目 - A
B科目 - B
C科目 - C
班级平均分 班级排名 - S
您想查看什么排序？ B
排名      成绩      姓名
1         98        梅东
2         87        Henry
3         21        邱森      Jrou
4         21        邱森      Jrou
5         1         C罗
**********************************************PROGRAM FINISH**
```

```
**********************************************PROGRAM FINISH*
是否继续?(y/n) y
A科目 - A
B科目 - B
C科目 - C
班级平均分 班级排名 - S
您想查看什么排序？ C
排名      成绩      姓名
1         231       邱森
2         97        梅东
3         45        Henry
4         34        Jrou
5         1         C罗
**********************************************PROGRAM FINISH*
```

```
****************************************PROGRAM FINISH****************************************
是否继续?(y/n) y
A科目 - A
B科目 - B
C科目 - C
班级平均分 班级排名 - S
您想查看什么排序? S
查询哪一个班级? 6
班级排名:
序号    学期    班级    姓名    学号    A    B    C    平均    总分
1    1    6    梅东    30    76    98    97    90    271
2    6    6    邱森    815    14    21    231    88    266
班级平均分: 268
****************************************PROGRAM FINISH****************************************
是否继续?(y/n)
```

输入错误操作 以及 不存在班级查询情况：

```
*********************************************PROGRAM FINISH*
是否继续?(y/n) y
A科目 - A
B科目 - B
C科目 - C
班级平均分 班级排名 - S
您想查看什么排序? F
输入功能违法!
*********************************************PROGRAM ERROR**
是否继续?(y/n)
```

```
*********************************************PROGRAM ERROR****
是否继续?(y/n) y
A科目 - A
B科目 - B
C科目 - C
班级平均分 班级排名 - S
您想查看什么排序? S
查询哪一个班级? 2
不存在该班级
A科目 - A
B科目 - B
C科目 - C
班级平均分 班级排名 - S
您想查看什么排序?
```

5. 输入 3 查询学生信息并展示

```
请输入您想进行的操作: 3
*****************************************PROGRAM START**
请输入学号或姓名:
```

```
请输入您想进行的操作: 3
*****************************************PROGRAM START***
请输入学号或姓名: 815
姓名:  邱森
学号: 815
班级: 6
学期: 6
A成绩: 14
B成绩: 21
C成绩: 231
总成绩: 266
平均分: 88
*****************************************PROGRAM FINISH**
是否继续?(y/n)
```

```
*****************************************PROGRAM FINISH**
是否继续?(y/n) y
请输入学号或姓名: Henry
姓名:  Henry
学号: 14
班级: 1
学期: 1
A成绩: 92
B成绩: 87
C成绩: 45
总成绩: 224
平均分: 74
*****************************************PROGRAM FINISH**
```

输入不存在学号或姓名：

```
*********************************
是否继续?(y/n) y
请输入学号或姓名: 3214
*********************************
是否继续?(y/n) y
请输入学号或姓名: rqsds
*********************************
是否继续?(y/n)
```

6. 输入 4 输出所有人的成绩单 以表格形式展示

```
请输入您想进行的操作: 4
****************************************PROGRAM START****************************************
序号    学期    班级    姓名    学号    A    B    C    平均    总分
1    1    1    C罗    7    1    1    1    1    3
2    6    6    邱森    815    14    21    231    88    266
3    1    6    梅东    30    76    98    97    90    271
4    1    1    Henry    14    92    87    45    74    224
5    1    1    Jrou    12    23    21    34    26    78
****************************************PROGRAM FINISH****************************************
**********************************************************************************************
输入 操作
0   EXIT
```

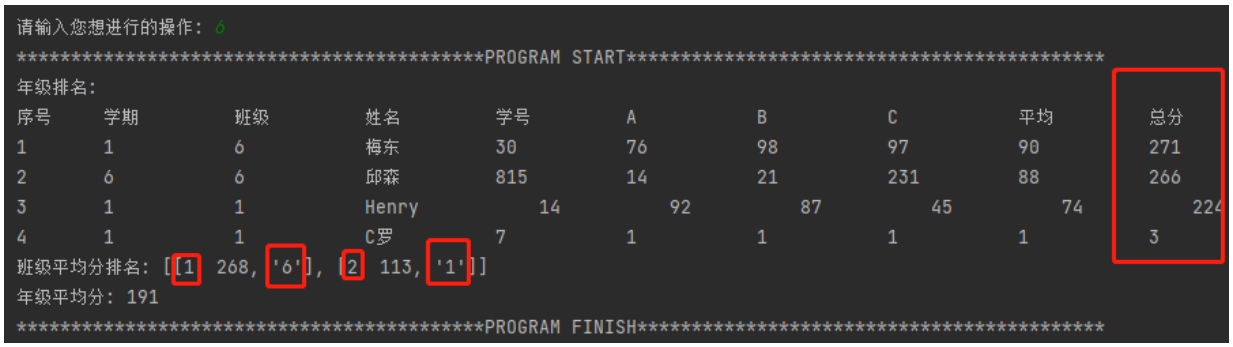
7. 输入 5 删除学生信息 先根据学号或姓名查询后展示、删除

```
请输入您想进行的操作: 5
*********************************************PROGRAM START***
请输入学号或姓名: Jrou
姓名:  Jrou
学号: 12
班级: 1
学期: 1
A成绩: 23
B成绩: 21
C成绩: 34
总成绩: 78
平均分: 26
OK
*********************************************PROGRAM FINISH**
是否继续删除操作?(y/n)
```

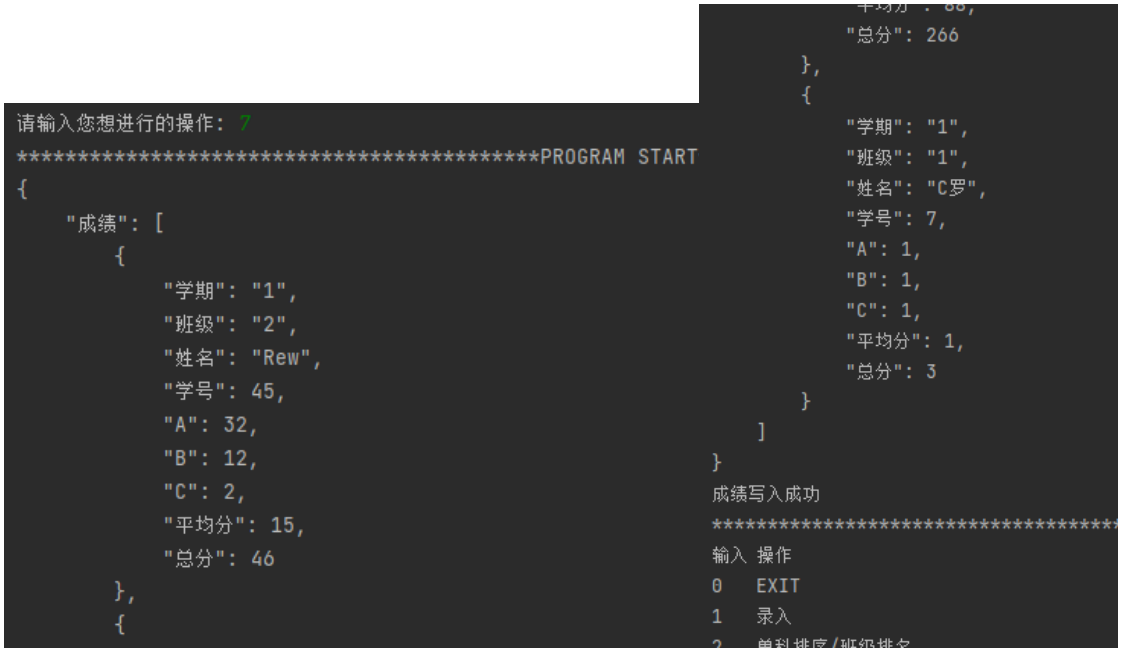
输入不存在的学生信息：

```
*********************************************PROGRAM FINISH*
是否继续删除操作?(y/n) y
请输入学号或姓名: 412432
未找到对应学生
*********************************************PROGRAM FINISH*
是否继续删除操作?(y/n)
```

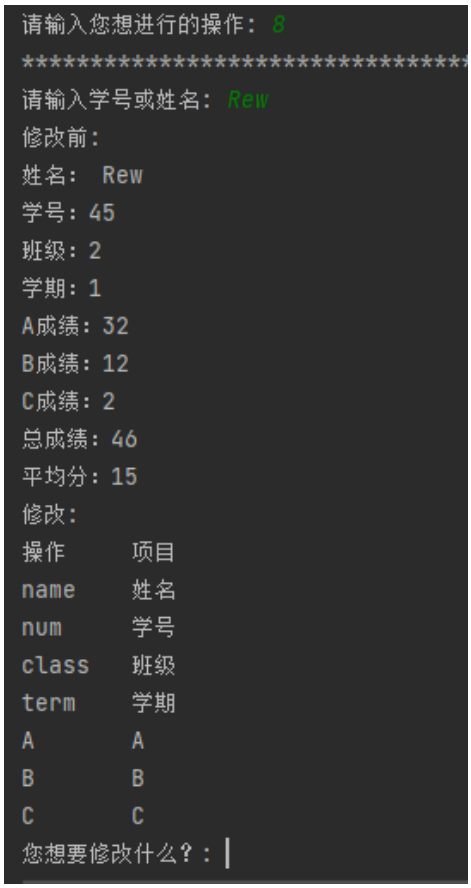
8. 输入 6 输出年级总排名/各班级平均分排名

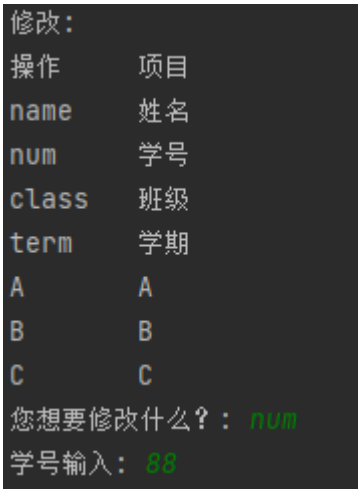


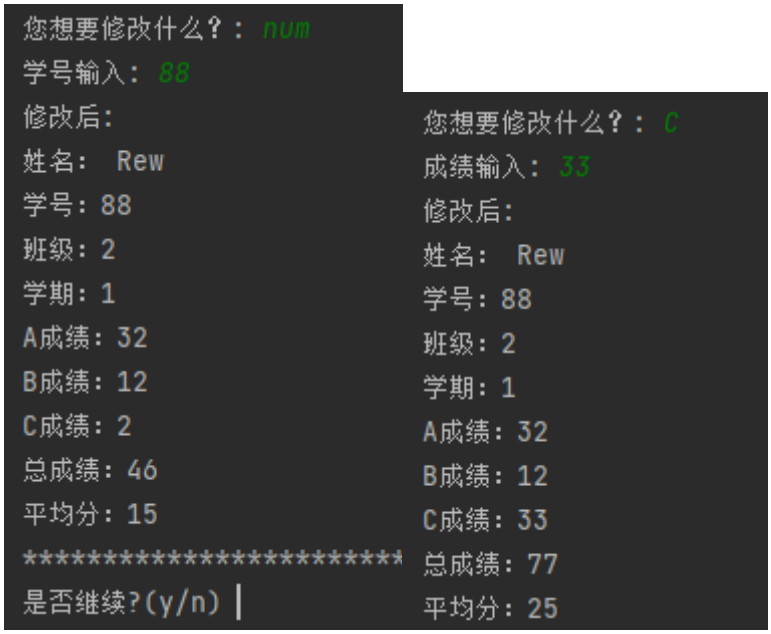
9. 输入 7 进行写入 json 文件操作 并输出写入内容



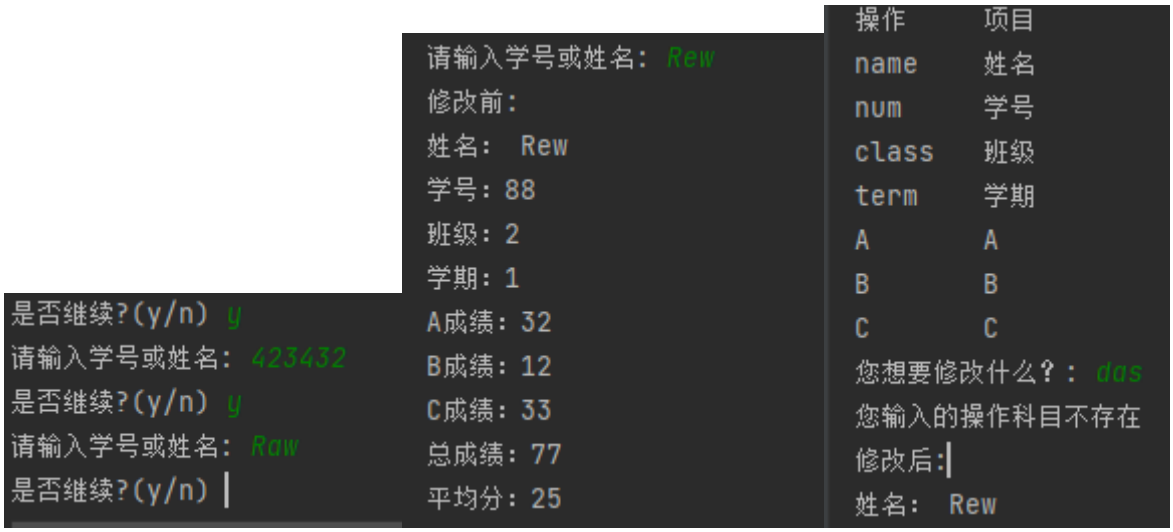
10. 输入 8 进行修改学生成绩/信息操作

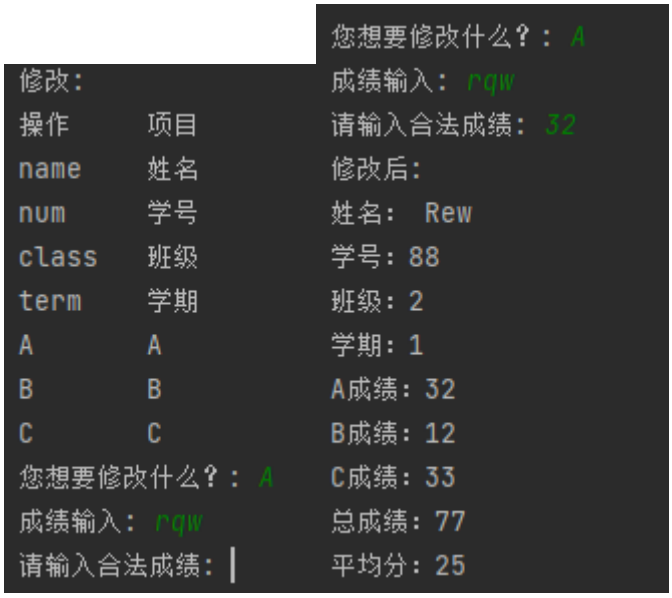




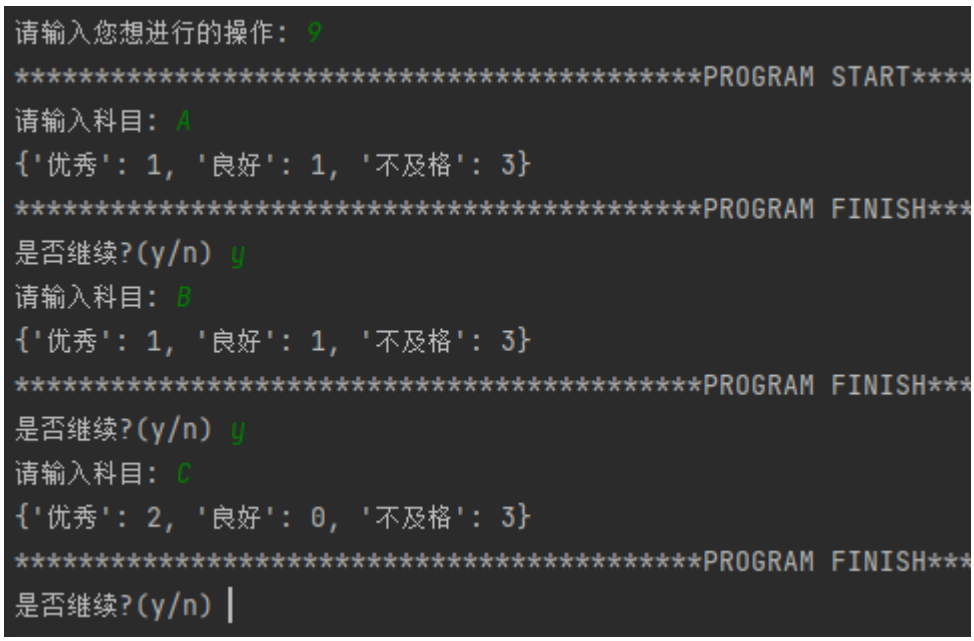


在学生查询时输入不存在的信息 输入修改项目时胡乱输入 成绩非法输入情况：

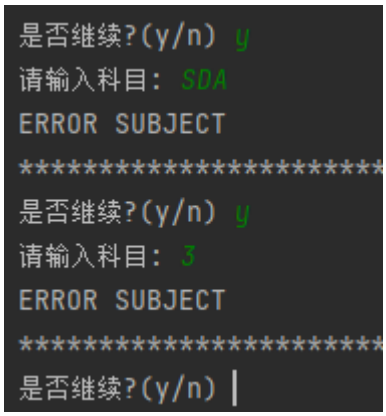




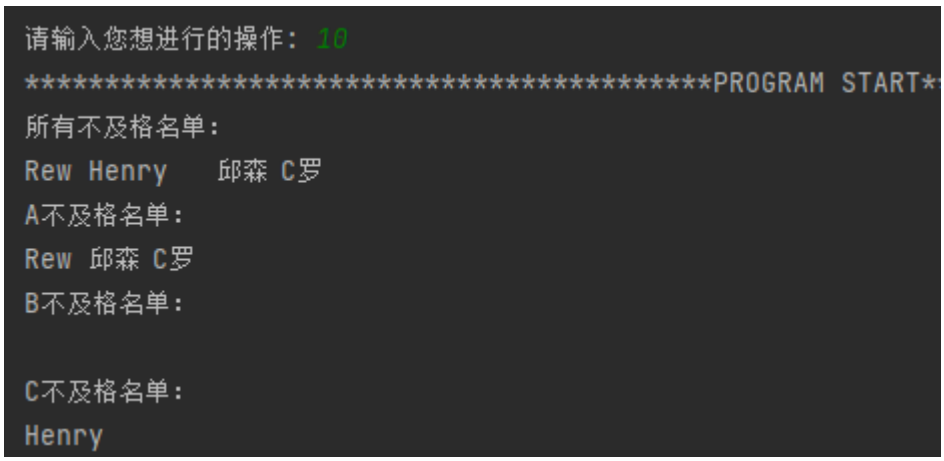
11. 输入 9 统计分数段的学生 根据科目



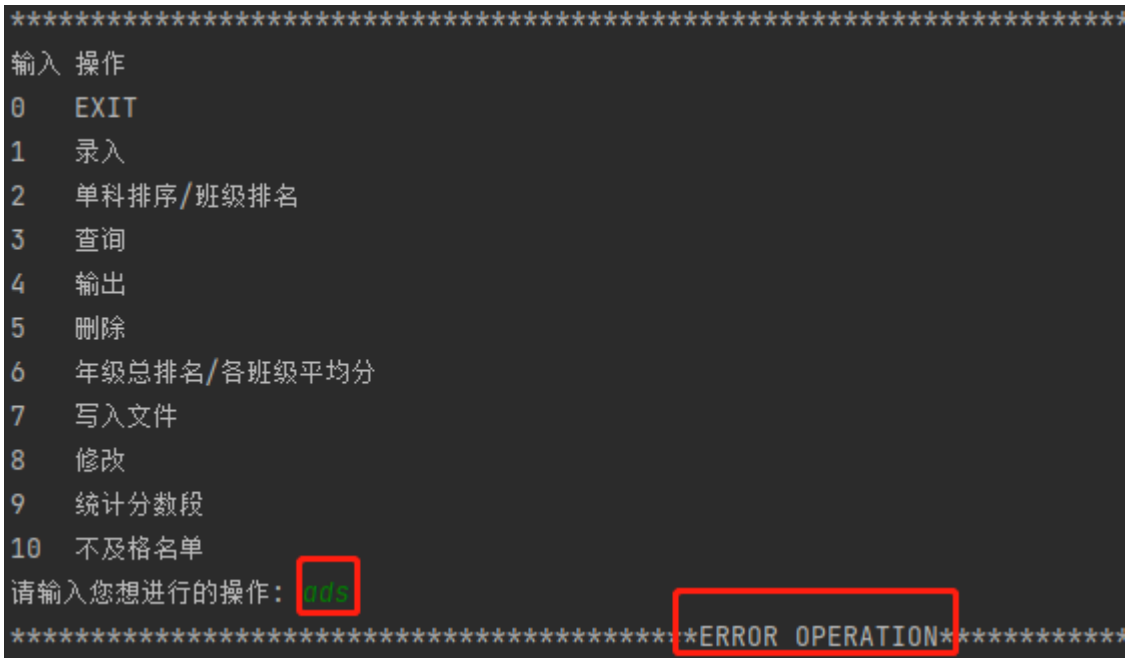
输入非法科目：

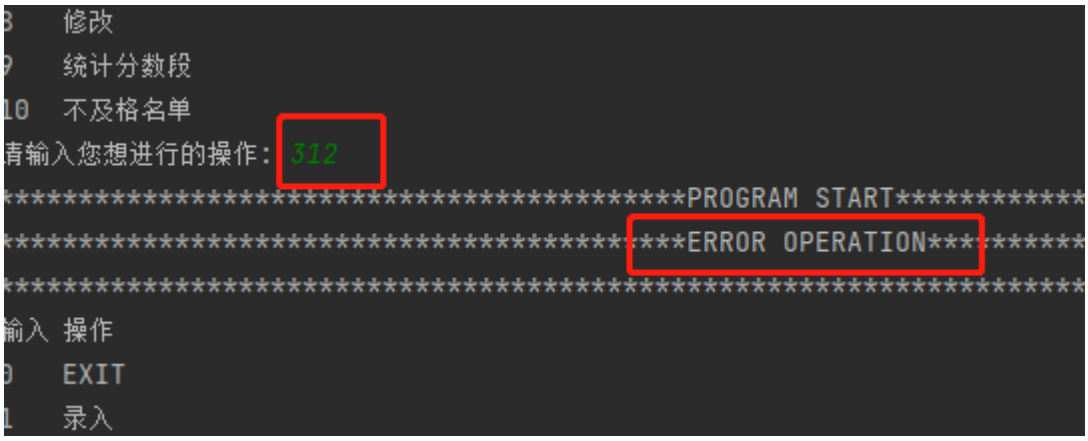


12. 输入 10 查看不及格人员名单 科目分开 同时有汇总



13. 在菜单栏输入非法操作





4. 讨论及进一步研究建议

本设计创新点：

在能准确完成其基本要求的情况下，结合基本要求中已经存在的算法，相互引用，代入不同的参数得到新的功能，例如排序的功能，对班级参数、单科参数进行规范可以得到班级排名、单科排名，还得到了有对比性的年级排名。同时在读取参数的同时统计分数，进行归类就可以得到一个各分数段统计的算法……

其原因在于写好的函数算法可用性、可读性强，数据结构统一，解析方便，同时运用了链表结构的优越性，对于参数的运用更加简单、高效。

本系统用的排序方法上丰富多样，希尔、插入、冒泡、选择、折半排序体现在不同科目、班级、年级的一些参数排序中，且十分准确，无失误。

在程序与外部函数间数据解析、存储上，用了易于读写、结构简单，格式压缩，占用带宽小的 json 文件，支持多种语言，且可以加密。

运用第三方包来支持函数算法进行，python 的社区环境十分庞大，且数据类型十分多样，运用多样的数据类型来对学生数据进行整合、处理、输出等操作十分有利，且解析起来十分方便，一一对应出错率低。

在新增节点的功能上，有主要的前插法和尾插法，这两种方法主要用前者的原因在于其优秀的时间性能，两者同样稳定的情况下使用性能优越的前者，但后者并没有删除，作为备用算法使用。

查询功能上，在日常使用的过程中我们通常会根据学生的学号或姓名来查询，二者选一，但此程序利用 python 语法的优越可以让二者合一，即在查询过程中，任意输入学号或姓名都可以进行查询操作。

代码层面上，对类与方法的统一归并，调用，数据传递一一对应。对可能出现错误的地方进行测试收集数据，然后通过错误处理操作来相应输出信息，告诉用户，且错误信息不会传递到核心的链表区域，做到了整篇程序几乎 0bug。

新见解：

从主函数出发看整篇代码,可以将主函数另写一个函数,将其优化到一个 GUI 程序中，但对于已经写好的主函数再进行拆分归并应用到一个个 GUI 程序函数中，工作量十分大。变成事后诸葛亮，同时除 GUI 之外，可以运用到 HTML 中，在一个网页上运行，输出。

在输出函数中，由于没有运用到真正的表格(excel)文件，且是链表结构，解析在一个临时的类列表或类字典中输出出来，格式在统一的情况下会显得工整，但如果学生的数据长短不统一，会在输出上不工整，所以在输出函数上可以进行一定优化。为何不优化呢？我认为 json 的文件格式经过规范化，直接对 json 文件进行查看，而且在程序中临时输出无伤大雅，没有明显的错误，数据都是一一对应的。

在 2 个大类，链表元素（对应学生）与链表类之间相互引用，可以单独制作成一个包，随用随拿。还有很多个排序函数，也可以制作成一个单独的包，然后在主函数中只需要引用即可。

程序进行中，可以进行更好的界面展示给用户，输出一个感观良好的画面给用户。

最后也是对应最开始，程序的开始，一个用户登入系统是可以加入到程序的，但一个完整可以真正使用的用户登入程序也需要不比此本程序代码量少的工作量投入，用简单的用户登入程序反而会出现许许多多的 bug，再进行一一测试优化，时间上已经不允许，好的解决方法就是直接引用其他人做好的包，直接引用即可，但这不符合课程设计的目的。

5.课程设计心得

程序设计过程中，对于学生的链表元素处理起来，即在外部文件和内存中转换的思路思考了很久，对如何选择解析方式，解析到哪，什么地步，都要结合性能进行比较，python拥有庞大的社区也代表了有多种多样的数据类型、数据结构，所以在如此自由的语言当作进行课程设计也十分“自由”，这次课程设计还算简单，数据类型上处理起来不算复杂，都是在字符串和数字中引用，字典和列表中相互提取转化排序等操作，学习到了一些高级的语法，运用到了许久没用的异常处理，和自己之前写的程序有区别，此课程设计更考究基础功力，运用能力的结合性强，但拓展性因为市面上有十分多的例子且十分成熟而略显不足。所以我在设计的时候更加注重其性能与稳定性，做到没有 bug 出现，且符合数据结构课程设计要求。

在外部文件上，第一次用 json 去存储数据，先前用此来读取数据集，而在写入的过程又有些不同，对于数据类型也有一定要求，在写的时候疏忽过，探索过。但做完后，再写读取的时候就游刃有余了。

写此程序主体不难，但后面的调试，自己将错误信息或者操作输入进去的时候很容易出错，所以进行了十分多优化和调试，但是在调试的过程中又添加了许许多多的新代码，新功能，对函数引用更加充分，优化输出参数的结构，分离一些功能又补足到另一些地方……反反复复调试了 2 天，而写程序主体则只需要几小时。就好像我班主任俞龙老师说的一样，写代码 3 分时间在写，7 分时间在调试、优化。3 分时间在真正内容，7 分时间在加密程序。这次做出来的是一个完整的系统，而不是平时要实现某个复杂包的功能实现，所以对调试有十分大的体会。

这次写的系统对我来说最好的一点就是写的过程中代码规范性十分好，可读性高，复杂数据传递的地方有注释，函数有注释，函数名清晰。

虽然函数功能实现的很好，但是在主函数的书写上没有做到模块化，大多是函数间的参数传递，性能浪费现象没有得到很好的解决，以后的实践中是完全可以得到改善的。

Student Achievement Management System

Abstract

My topic of this course design is topic 1: student achievement management system, First of all, it can complete all the functions of the basic requirements (such as inputting scores, counting student scores by class, calculating total scores, sorting average scores, querying scores, outputting transcripts, deleting by student number, storing information...). After a lot of testing, improvement and optimization, the probability of program bugs is almost zero. Innovative functions are also very diverse, such as class ranking, grade ranking, single subject ranking, score segment Statistics ... in the query, the student number and name can be entered arbitrarily, which is very fast, and the sorting methods are also very diverse, which is reflected in the different sorting methods used in the sorting of different subjects or total scores, including shell sorting, quick sorting, selection sorting, half sorting, insertion sorting and bubble sorting. The unique link list storage is adopted in the data structure. There are two kinds of adding operations: pre insertion and tail insertion. It has excellent memory management. The program for parsing external files and linked list information is also very stable. At the beginning of the program, the data in JSON files can be quickly parsed and stored in the temporary linked list. The code structure is organized and logical. The whole code is 0 warning. It uses 2 categories, up to 28 functions, and 746 lines of code. Some functions call other functions. The code is standardized. At the same time, it uses Python third-party packages collections and JSON. It is excellent in both viewing and analysis difficulty in storage. At the same time, the students' grades are good, The legitimacy of different subjects or their student attribute values can also be guaranteed, so the author has sufficient confidence in the stability of the program compared with other management systems, there are corresponding notes between functions, which is highly readable, and the program has corresponding output to guide users in use, with sufficient information. The linked list seems to be complex, but the author can realize the sorting, query and storage operations by combining the list, class list, dictionary and class dictionary.