武汉软件工程职业学院

课程设计报告

2022-2023学年度第 一 学期

题目： Python学生信息管理系统

姓名： 杨开发

班级： 网络2004

系部： 计算机学院

指导教师： 王子龙

日期：2022年11月 17日

1. 项目规划：

1.设计目的

利用Python编程语言编写一个学生信息管理系统，每个学生应该包含如下信息：学号、姓名、性别、年龄、班级以及班级排名等。可以任意添加学生，删除学生，编辑学生信息，根据学号或年龄对学生信息进行排序，并且具有保存和读入学生信息的功能以及根据年龄段统计学生人数情况。

如今学校的学生越来越多，成绩管理的工作量越来越大，手工管理成绩的弊端也越来越明显。随着科学技术的不断提高,计算机科学日渐成熟,它已进入人类社会的各个领域并发挥着越来越重要的作用。作为计算机应用的一部分，使用计算机对学生档案信息进行管理,具有手工管理所无法比拟的优点。所以我想借本次课程设计之际，设计一个学生成绩管理系统

1. 课程设计的实验环境：

* 操作系统：Windows7、Windows10、Windows11。
* Python版本：Python 3.11；
* 开发工具：Pycharm；
* Python模块：tkinter，time，csv，os，pickle

1. 概要设计：

学生信息管理系统分为7大功能模块，主要包括录入学生信息模块、查找学生信息模块、删除学生信息模块、修改学生信息模块、学生成绩排名模块、统计学生总人数模块及显示全部学生信息模块。

1. 详细设计：
2. 主函数设计

运行学生信息管理系统，首先将进入主功能菜单的选择界面，在这里列出了程序中的所有功能，用户可以根据需要输入要执行功能对应的数字编号或者按下键盘上的方向键，进入对应的子功能中去。其次在menu（）函数中主要使用print（）函数自爱控制台输出文字和特殊字符组成的功能菜单。当用户输入功能编号或者选择相应的功能后，程序会根据用户选择的功能编号（如果是通过方向键选择的功能，程序会自动提取出对应的数字）调用不同的函数。

1. 学生信息维护模块设计

### **2.1实现录入学生信息功能**

1. 编写一个向文件中写入指定内容的函数，将其命名为save（），该函数有一个列表类型的参数，用于指定要写入的内容。
2. 编写主函数的调用的录入学生信息的函数insert（）。在该函数中，先定义一个保存学生信息的空列表，然后设置一个while循环，在该循环中通过input（）函数要求用户输入学生信息（包括学生ID、名字、英语成绩、Python成绩和C语言成绩）。如果这些内容都符合要求，则将它们保存到字典中，再将该字典添加到列表中并且询问是否继续录入；如果不再录入则结束while循环，并调用save（）函数，将录入的学生信息保存到文件中。

**2.2实现删除学生信息功能**

编写主函数中调用的删除学生信息的函数delete（）。在该函数中，设置一个while循环，在该循环中，首先通过input（）函数要求用户输入要删除学生ID；然后以只读模式打开保存学生信息的文件，并且读取其内容保存到一个列表中；再以写模式打开保存学生信息的文件，并且遍历保存学生信息的列表，将每个元素转换为字典，从而方便根据输入的学生ID判断是否为要删除的信息。如果不是要删除的信息，则将其重新写入到文件中。

2.3实现修改学生信息功能

编写主函数中调用的修改学生信息的函数modify()。在该函数中，调用show（）函数显示全部学生信息，之后再判断保存学生信息的文件是否存在。如果存在，以只读模式打开文件，并且读取全部学生信息保存到列表中，否则返回。接下来提示用户输入要修改的学生ID，并且以只写模式打开文件。打开文件后，遍历保存学生信息的列表，将每个元素转换为字典，再根据输入的学生ID判断是否为要修改的信息。如果是要修改的信息，则提示用户输入新的信息，并保存到文件，否则直接将其写入文件中。

1. 查询/统计模块设计

### 3.1**实现查找学生信息功能**

编写主函数中调用的查找学生信息的函数search（），设置一个循环并在该循环中先判断保存学生信息的文件是否存在，如果不存在则给出提示并返回，否则提示用户选择查询方式；然后根据选择的方式到保存学生信息的文件中查找对应的学生信息，并且调用show\_student（）函数将查询结果进行显示。

### 3.2**实现统计学生总人数功能**

编写主函数中调用的统计学生总人数的函数show（）。在该函数中，添加一个if语句，用于判断保存到学生信息的文件是否存在，如果存在则以只读模式打开该文件，读取该文件的全部内容后保存到一个列表中。然后再遍历该列表，并将其元素转换为字典，再添加到一个新列表中，最后再调用show\_student（）函数将新列表中的信息显示。show（）函数的具体代码如下：

### 3.3**实现显示所有学生信息功能**

编写主函数中调用的统计学生总人数的函数show（）。在该函数中，添加一个if语句，用于判断保存学生信息的文件是否存在，如果存在则以只读模式打开该文件，读取该文件的全部内容后保存到一个列表中。然后再遍历该列表，并将其元素转换为字典，再添加到一个新列表中，最后再调用show\_student()函数将新列表中的信息显示。

### 排序模块设计

### 4.1**实现按学生成绩排序**

编写主函数中调用的排序的函数sort（）。在该函数中，首先判断保存学生信息的文件是否存在，如果存在，则打开该文件读取全部学生信息，并将每一名学生信息转换为字典保存到一个新的列表中，然后获取用户输入的排序方式，再根据选择结果进行相应的排序，最后调用show\_student（）函数显示排序结果。

1. 核心代码

# \_\*\_ coding:utf-8 \_\*\_

"""

@ 功能 : 本模块用于对窗口界面的定义

@ author : 杨开发

@ create : 2022/11

"""

from tkinter import \*

from tkinter import messagebox

from tkinter import ttk

import DataProcess as Dp

import time

class FirstFrame(Frame):

"""程序首界面，用于选择执行的系统"""

global Student # 存放学生信息的列表

def \_\_init\_\_(self, master=None):

super().\_\_init\_\_(master) # super()代表了父类的定义

self.master = master

self.pack(anchor='n') # 创建frame时将其放置在master父模块上

'''建造时创建组件'''

# 初始化组件的属性

self.label\_tit = Label(self, text='学生成绩系统', width=24, height=1, fg='black', font=('黑体', 25))

self.btn\_stu = Button(self, text='学生成绩查询系统', width=18, height=1, bg='gray', font=('黑体', 15))

self.btn\_adm = Button(self, text='学生成绩管理系统', width=18, height=1, bg='gray', font=('黑体', 15))

self.btnQuit = Button(self, text='退出', width=18, height=1, font=('黑体', 15), command=Exit)

# 组件的放置位置

self.label\_tit.grid(row=0, column=2, columnspan=3, pady=30)

self.btn\_stu.grid(row=1, column=0, columnspan=3, pady=30, padx=50)

self.btn\_adm.grid(row=1, column=4, columnspan=3, pady=30, padx=50)

self.btnQuit.grid(row=2, column=2, columnspan=3, pady=30)

# 组件和事件的绑定

self.btn\_stu.bind('<Button-1>', self.btn\_stu\_ev)

self.btn\_adm.bind('<Button-1>', self.btn\_adm\_ev)

def btn\_stu\_ev(self, event):

"""学生成绩查询系统 的操作"""

global root

RegisterFrame(master=root, TypeChoice='Student')

self.destroy()

def btn\_adm\_ev(self, event):

"""学生成绩管理系统 的操作"""

global root

RegisterFrame(master=root, TypeChoice='Admin')

self.destroy()

class RegisterFrame(Frame):

"""登录界面"""

def \_\_init\_\_(self, master=None, TypeChoice=None): # TypeChoice 用于判断学生查询界面或管理员登录界面

super().\_\_init\_\_(master) # super()代表了父类的定义

self.master = master

self.type = TypeChoice

self.Save = 0

self.pack(anchor='center') # 创建时将Frame放置在master中

if self.type == 'Student':

self.type = '学生成绩查询系统登录界面'

elif self.type == 'Admin':

self.type = '学生成绩管理系统登录界面'

else:

self.type = '登陆界面函数参数错误' # 通过判断 self.type 的值决定界面的标题抬头

"""创建组件"""

# 初始化组件的属性

self.label\_tit = Label(self, text=self.type, width=24, height=1, fg='black', font=('黑体', 20))

self.label\_adm = Label(self, text='用户名(姓名):', width=12, height=1, fg='black', font=('黑体', 15))

self.label\_pwd = Label(self, text='密 码(学号):', width=12, height=1, fg='black', font=('黑体', 15))

self.btn\_cfm = Button(self, text='确定', width=8, height=1, bg='gray', font=('黑体', 15))

self.btn\_cnc = Button(self, text='清空', width=8, height=1, bg='gray', font=('黑体', 15))

self.btn\_rtn = Button(self, text='返回', width=8, height=1, bg='gray', font=('黑体', 15))

self.btnQuit = Button(self, text='退出系统', width=8, height=1, font=('黑体', 15), command=Exit)

self.adm = StringVar() # 将Entry框的内容与变量进行绑定

self.pwd = StringVar()

self.adm.set('admin') # 设定Entry框显示的内容

self.pwd.set('')

self.entry\_adm = Entry(self, textvariable=self.adm)

self.entry\_pwd = Entry(self, textvariable=self.pwd, show='\*')

# 设定组件在父Frame 中的位置

self.label\_tit.grid(row=0, column=0, columnspan=48, pady=30)

self.label\_adm.grid(row=1, column=0, pady=20, columnspan=12, sticky=W)

self.label\_pwd.grid(row=2, column=0, pady=20, columnspan=12, sticky=W)

self.btn\_cfm.grid(row=3, column=0, pady=20, padx=20, columnspan=8)

self.btn\_cnc.grid(row=3, column=10, pady=20, padx=20, columnspan=8)

self.btn\_rtn.grid(row=3, column=20, pady=20, padx=20, columnspan=8)

self.btnQuit.grid(row=4, column=10, pady=20, columnspan=8)

self.entry\_adm.grid(row=1, column=20, columnspan=12, )

self.entry\_pwd.grid(row=2, column=20, columnspan=12, )

# 设定组件绑定的事件

self.btn\_cfm.bind('<Button-1>', self.btn\_cfm\_ev)

self.btn\_cnc.bind('<Button-1>', self.btn\_cnc\_ev)

self.btn\_rtn.bind('<Button-1>', self.btn\_rtn\_ev)

self.entry\_adm.bind('<KeyPress-Return>', self.btn\_cfm\_ev)

self.entry\_pwd.bind('<KeyPress-Return>', self.btn\_cfm\_ev)

def btn\_cfm\_ev(self, event):

"""确认按钮对应的事件"""

global Admin, School, root, Empty, Student, adMenuFrame, showFrame, FindResult, findStudent

if self.type == '学生成绩管理系统登录界面' and [self.adm.get(), self.pwd.get()] in Admin: # 当在管理员登陆界面并且账号密码在管理员列表中

if School.get('name') is None: # 如果学校信息元组为空时，对学校信息进行初始化

messagebox.showinfo(title='提示信息', message='请初始化学校信息！')

self.destroy()

StudentSet(master=root)

else: # 转入管理员界面

self.destroy()

Empty.destroy()

adMenuFrame = AdMenuFrame(master=root)

showFrame = ShowFrame(master=root)

elif self.adm.get() == '': # 账号Entry框为空

messagebox.showinfo(title='提示信息', message='账号(姓名)不能为空！')

elif self.type == '学生成绩查询系统登录界面' and self.pwd.get() == '':

messagebox.showinfo(title='提示信息', message='密码(学号)不能为空！')

elif self.type == '学生成绩查询系统登录界面' and self.pwd.get() != '':

for i in range(1, len(Student)): # 学生成绩查询系统界面 判断学生姓名和学号是否存在并对应同一个人

if self.pwd.get() == Student[i][0] and self.adm.get() == Student[i][1]:

self.Save = i

if self.Save != 0:

FindResult = Student[self.Save]

self.destroy()

Empty.destroy()

findStudent = FindStudent(master=root)

else:

messagebox.showinfo(title='提示信息', message='姓名或学号输入错误,不存在该学生信息！')

else:

messagebox.showinfo(title='提示信息', message='账号或密码输入错误！')

def btn\_cnc\_ev(self, event):

"""清空按钮对应的事件"""

self.adm.set('')

self.pwd.set('')

def btn\_rtn\_ev(self, event):

"""返回按钮对应的事件"""

global root

self.destroy()

FirstFrame(master=root)

class StudentSet(Frame):

"""学校信息初始化界面"""

def \_\_init\_\_(self, master=None):

super().\_\_init\_\_(master) # super()代表了父类的定义

self.master = master

self.pack(anchor='n')

"""创建组件"""

# 组件内容的初始化

self.label\_tit = Label(self, text='学校信息初始化', width=24, height=1, fg='black', font=('黑体', 25))

self.label\_scn = Label(self, text='学 校名称:', width=12, height=1, fg='black', font=('黑体', 15))

self.label\_C\_grade = Label(self, text='C 语言学分:', width=12, height=1, fg='black', font=('黑体', 15))

self.label\_M\_grade = Label(self, text='数 学学分:', width=12, height=1, fg='black', font=('黑体', 15))

self.label\_E\_grade = Label(self, text='英 语学分:', width=12, height=1, fg='black', font=('黑体', 15))

self.btn\_cfm = Button(self, text='确定', width=8, height=1, bg='gray', font=('黑体', 15))

self.btn\_rtn = Button(self, text='返回上一级', width=10, height=1, bg='gray', font=('黑体', 15))

self.btn\_cnc = Button(self, text='清空', width=8, height=1, bg='gray', font=('黑体', 15))

self.btnQuit = Button(self, text='退出系统', width=8, height=1, font=('黑体', 15), command=Exit)

self.v\_Name = StringVar()

self.v\_C\_grade = StringVar()

self.v\_M\_grade = StringVar()

self.v\_E\_grade = StringVar()

self.entry\_scn = Entry(self, textvariable=self.v\_Name, width=24)

self.entry\_C\_grade = Entry(self, textvariable=self.v\_C\_grade, width=24)

self.entry\_M\_grade = Entry(self, textvariable=self.v\_M\_grade, width=24)

self.entry\_E\_grade = Entry(self, textvariable=self.v\_E\_grade, width=24)

# 组件的放置位置

self.label\_tit.grid(row=0, column=0, columnspan=24, pady=30)

self.label\_scn.grid(row=1, column=0, columnspan=12)

self.label\_C\_grade.grid(row=2, column=0, columnspan=12)

self.label\_M\_grade.grid(row=3, column=0, columnspan=12)

self.label\_E\_grade.grid(row=4, column=0, columnspan=12)

self.btn\_cfm.grid(row=5, column=0, pady=20, padx=20, columnspan=8)

self.btn\_cnc.grid(row=5, column=10, pady=20, padx=20, columnspan=8)

self.btn\_rtn.grid(row=5, column=20, pady=20, padx=20, columnspan=8)

self.btnQuit.grid(row=6, column=10, pady=20, columnspan=8)

self.entry\_scn.grid(row=1, column=15, pady=10, columnspan=24)

self.entry\_C\_grade.grid(row=2, column=15, pady=10, columnspan=24)

self.entry\_M\_grade.grid(row=3, column=15, pady=10, columnspan=24)

self.entry\_E\_grade.grid(row=4, column=15, pady=10, columnspan=24)

# 组件对应的事件

self.btn\_cfm.bind('<Button-1>', self.btn\_cfm\_ev)

self.btn\_cnc.bind('<Button-1>', self.btn\_cnc\_ev)

self.btn\_rtn.bind('<Button-1>', self.btn\_rtn\_ev)

def btn\_cfm\_ev(self, event):

"""确定按钮"""

global School, Empty, adMenuFrame, showFrame

Name = self.v\_Name.get()

E\_grade = self.v\_E\_grade.get()

C\_grade = self.v\_C\_grade.get()

M\_grade = self.v\_M\_grade.get()

List = [Name, E\_grade, C\_grade, M\_grade]

try:

if Name == '' or E\_grade == '' or C\_grade == '' or M\_grade == '':

messagebox.showinfo(title='提示信息', message='数据不能为空！')

elif 0 < float(E\_grade) < 10 and 0 < float(C\_grade) < 10 and 0 < float(M\_grade) < 10:

School['name'] = Name

School['E\_grade'] = float(E\_grade)

School['C\_grade'] = float(C\_grade)

School['M\_grade'] = float(M\_grade)

Dp.Student.setting(School)

messagebox.showinfo(title='提示信息', message='初始化学校信息成功！')

self.destroy()

Empty.destroy()

adMenuFrame = AdMenuFrame(master=root)

showFrame = ShowFrame(master=root)

else:

messagebox.showinfo(title='提示信息', message='学分必须在 0-10 之间!')

except ValueError:

messagebox.showinfo(title='提示信息', message='学分必须为数字!')

def btn\_cnc\_ev(self, event):

self.v\_Name.set('')

self.v\_E\_grade.set('')

self.v\_M\_grade.set('')

self.v\_C\_grade.set('')

def btn\_rtn\_ev(self, event):

global root

self.destroy()

FirstFrame(master=root)

class AdMenuFrame(Frame):

"""管理员操作界面，用于对学生信息的操作"""

def \_\_init\_\_(self, master=None):

super().\_\_init\_\_(master) # super()代表了父类的定义

self.master = master

self.pack(anchor='n')

"""创建组件"""

self.label\_tit = Label(self, text='学生成绩管理系统', width=24, height=1,

fg='black', font=('黑体', 25))

self.btn\_add = Button(self, text='新增学生信息', width=12, height=1,

bg='gray', font=('黑体', 15))

self.btn\_del = Button(self, text='删除学生信息', width=12, height=1,

bg='gray', font=('黑体', 15))

self.btn\_adm = Button(self, text='登录账户管理', width=12, height=1,

bg='gray', font=('黑体', 15))

self.btn\_rtn = Button(self, text='返回主界面', width=12, height=1,

bg='gray', font=('黑体', 15))

self.label\_tit.grid(row=0, column=8, columnspan=24)

self.btn\_add.grid(row=1, column=0, pady=10, columnspan=12)

self.btn\_del.grid(row=1, column=12, pady=10, columnspan=12)

self.btn\_adm.grid(row=1, column=24, pady=10, columnspan=12)

self.btn\_rtn.grid(row=2, column=12, pady=10, columnspan=12)

self.btn\_add.bind('<Button-1>', self.btn\_add\_ev)

self.btn\_del.bind('<Button-1>', self.btn\_del\_ev)

self.btn\_adm.bind('<Button-1>', self.btn\_adm\_ev)

self.btn\_rtn.bind('<Button-1>', self.btn\_rtn\_ev)

def btn\_add\_ev(self, event):

global root, showFrame, Empty

self.destroy()

showFrame.destroy()

Empty = Frame(root, height=100)

Empty.pack()

AddStudent(master=root)

def btn\_del\_ev(self, event):

global root, showFrame

self.destroy()

showFrame.destroy()

DelStudent(master=root)

def btn\_adm\_ev(self, event):

global root, adminSetFrame, Empty

self.destroy()

showFrame.destroy()

Empty = Frame(root, height=100)

Empty.pack()

adminSetFrame = AdminSetFrame(master=root)

def btn\_rtn\_ev(self, event):

global showFrame, Empty

self.destroy()

showFrame.destroy()

Empty = Frame(root, height=100)

Empty.pack()

FirstFrame(master=root)

class ShowFrame(Frame):

"""将学生信息以表格形式显示"""

def \_\_init\_\_(self, master=None):

super().\_\_init\_\_(master) # super()代表了父类的定义

self.master = master

self.place(relx=0.1, y=150, relwidth=0.8, height=330)

"""创建组件"""

global Student, count

count = len(Student)

columns = Student[0]

self.widths = [100, 60, 60, 60, 60, 60, 60, 60, 90]

self.canvas = Canvas(self, bg='blue') # 创建canvas

self.canvas.place(relx=0, y=30, relwidth=1, relheight=1)

self.treeview = ttk.Treeview(self.canvas, show="headings", columns=columns) # 表格

self.treeview.place(relx=0, rely=0, relwidth=0.97, relheight=1)

self.VScroll1 = Scrollbar(self.canvas, orient='vertical', command=self.treeview.yview)

self.VScroll1.place(relx=0.97, rely=0, relwidth=0.03, height=300)

self.treeview.configure(yscrollcommand=self.VScroll1.set) # 给treeview添加配置

for i in range(len(self.widths)):

self.treeview.column(columns[i], width=self.widths[i], anchor='center') # 表示列,不显示

self.treeview.heading(columns[i], text=columns[i]) # 显示表头

for i in range(len(Student) - 1): # 写入数据

self.treeview.insert('', i, values=Student[i + 1])

self.entry = 1

self.treeview.bind('<Double-1>', self.set\_cell\_value) # 双击左键进入编辑

self.btn\_new = ttk.Button(self, text='新建学生信息', width=20, command=self.NewRow)

self.btn\_new.place(x=50, y=0)

for col in columns: # 绑定函数，使表头可排序

self.treeview.heading(col, text=col,

command=lambda \_col=col: self.treeview\_sort\_column(self.treeview, \_col, False))

def treeview\_sort\_column(self, treeview, col, reverse): # 'Treeview' 、列名、排列方式

sort = [(treeview.set(k, col), k) for k in treeview.get\_children('')]

sort.sort(reverse=reverse) # 排序方式

# rearrange items in sorted positions

for index, (val, k) in enumerate(sort): # 根据排序后索引移动

treeview.move(k, '', index)

treeview.heading(col, command=lambda: self.treeview\_sort\_column(treeview, col, not reverse)) # 重写标题，使之成为再点倒序的标题

def set\_cell\_value(self, event): # 双击进入编辑状态

if self.entry == 0:

self.entryedit.destroy()

self.okb.destroy()

for self.item in self.treeview.selection():

# item = I001

self.item\_text = self.treeview.item(self.item, "values")

self.items = self.item

# print(self.item\_text[0:]) # 输出所选行的值

self.column = self.treeview.identify\_column(event.x) # 列

self.row = self.treeview.identify\_row(event.y) # 行

self.cn = int(str(self.column).replace('#', ''))

self.rn = int(eval(str(self.row).replace('I', '0X').lower()))

self.VScroll1Place = [round(list(self.VScroll1.get())[0], 2), round(list(self.VScroll1.get())[1], 2)]

if self.cn <= 6:

self.v\_entryedit = StringVar()

self.entryedit = Entry(self.treeview, textvariable=self.v\_entryedit, width=12)

self.okb = ttk.Button(self.treeview, text='OK', width=4)

if self.rn < len(Student):

self.v\_entryedit.set(str(Student[self.rn][self.cn - 1]))

else:

self.v\_entryedit.set(str(Student[0][self.cn - 1]))

self.entry = x = 0

for i in range(self.cn - 1):

x += self.widths[i]

if self.VScroll1Place[0] != 0:

self.entryedit.place(x=x, y=self.rn \* 20 + 5 - count \* 20 \* self.VScroll1Place[0])

self.okb.place(x=x + 85, y=self.rn \* 20 + 5 - count \* 20 \* self.VScroll1Place[0])

else:

self.entryedit.place(x=x, y=self.rn \* 20 + 5)

self.okb.place(x=x + 85, y=self.rn \* 20 + 5)

self.entryedit.bind('<KeyPress-Return>', self.save\_edit)

self.okb.bind('<Button-1>', self.save\_edit)

else:

messagebox.showinfo(title='提示信息', message='该数据不能直接修改!')

def save\_edit(self, event):

global Student, showFrame

if self.rn < len(Student) and self.judge():

Student[self.rn][self.cn - 1] = self.entryedit.get()

if [i for i in range(3, 6) if Student[self.rn][i] != Student[0][i]] == [3, 4, 5]:

Student[self.rn] = Dp.Student.append(Student[self.rn][0:6])

elif self.rn >= len(Student) and self.judge():

Student.append(list(Student[0]))

Student[-1][self.cn - 1] = self.entryedit.get()

if [i for i in range(3, 6) if Student[self.rn][i] != Student[0][i]] == [3, 4, 5]:

Student[self.rn] = Dp.Student.append(Student[self.rn][0:6])

self.destroy()

showFrame = ShowFrame(master=root)

self.entryedit.destroy()

self.okb.destroy()

def judge(self):

try:

if self.entryedit.get() == '':

messagebox.showinfo(title='提示信息', message='数据不能为空！')

elif self.cn == 1 and len(self.entryedit.get()) != 10:

messagebox.showinfo(title='提示信息', message='学号必须为 10 位！')

elif self.cn == 1 and self.entryedit.get() in [Student[i][0] for i in range(1, len(Student)) if

i != self.rn]:

messagebox.showinfo(title='提示信息', message='该学号已存在！')

elif self.cn == 2 and self.entryedit.get() in ['男', '女']:

messagebox.showinfo(title='提示信息', message='性别必须为‘男’or‘女’！')

elif 3 < self.cn < 7 and 0 < float(self.entryedit.get()) < 100:

return True

elif 3 < self.cn < 7:

messagebox.showinfo(title='提示信息', message='成绩必须在 0-100 之间!')

else:

return True

except ValueError:

messagebox.showinfo(title='提示信息', message='成绩必须为数字!')

def NewRow(self):

global count, Student

self.treeview.insert('', count, values=Student[0])

count += 1

self.treeview.update()

self.btn\_new.update()

class AddStudent(Frame):

def \_\_init\_\_(self, master=None):

super().\_\_init\_\_(master) # super()代表了父类的定义

self.master = master

self.pack(anchor='n')

"""创建组件"""

self.label\_tit = Label(self, text='新增学生信息', width=12, height=1, fg='black', font=('黑体', 25))

self.label\_Num = Label(self, text='学 号:', width=12, height=1, fg='black', font=('黑体', 15))

self.label\_Name = Label(self, text='姓 名:', width=12, height=1, fg='black', font=('黑体', 15))

self.label\_Sex = Label(self, text='性 别:', width=12, height=1, fg='black', font=('黑体', 15))

self.label\_Cgrade = Label(self, text='C语言成绩:', width=12, height=1, fg='black', font=('黑体', 15))

self.label\_Mgrade = Label(self, text='数 学成绩:', width=12, height=1, fg='black', font=('黑体', 15))

self.label\_Egrade = Label(self, text='英 语成绩:', width=12, height=1, fg='black', font=('黑体', 15))

self.btn\_cfm = Button(self, text='确定', width=12, height=1, bg='gray', font=('黑体', 15))

self.btn\_del = Button(self, text='清空', width=12, height=1, bg='gray', font=('黑体', 15))

self.btn\_rtn = Button(self, text='返回上一级', width=12, height=1, bg='gray', font=('黑体', 15))

self.v\_num = StringVar()

self.v\_name = StringVar()

self.v\_Cgrade = StringVar()

self.v\_Mgrade = StringVar()

self.v\_Egrade = StringVar()

self.entry\_Num = Entry(self, textvariable=self.v\_num)

self.entry\_Name = Entry(self, textvariable=self.v\_name)

self.sexvalue = StringVar()

self.sexvalue.set('男')

self.sexman = Radiobutton(self, text="男", value="男", variable=self.sexvalue)

self.sexwoman = Radiobutton(self, text="女", value="女", variable=self.sexvalue)

self.entry\_Cgrade = Entry(self, textvariable=self.v\_Cgrade)

self.entry\_Mgrade = Entry(self, textvariable=self.v\_Mgrade)

self.entry\_Egrade = Entry(self, textvariable=self.v\_Egrade)

self.label\_tit.grid(row=0, column=6, columnspan=24, pady=40)

self.label\_Num.grid(row=1, column=0, columnspan=12)

self.label\_Name.grid(row=2, column=0, columnspan=12)

self.label\_Sex.grid(row=3, column=0, columnspan=12)

self.label\_Cgrade.grid(row=4, column=0, columnspan=12)

self.label\_Mgrade.grid(row=5, column=0, columnspan=12)

self.label\_Egrade.grid(row=6, column=0, columnspan=12)

self.entry\_Num.grid(row=1, column=18, columnspan=18)

self.entry\_Name.grid(row=2, column=18, columnspan=18)

self.sexman.grid(row=3, column=18, columnspan=9)

self.sexwoman.grid(row=3, column=27, columnspan=9)

self.entry\_Cgrade.grid(row=4, column=18, columnspan=18)

self.entry\_Mgrade.grid(row=5, column=18, columnspan=18)

self.entry\_Egrade.grid(row=6, column=18, columnspan=18)

self.btn\_cfm.grid(row=7, column=0, pady=20, padx=5, columnspan=12)

self.btn\_del.grid(row=7, column=12, pady=20, padx=5, columnspan=12)

self.btn\_rtn.grid(row=7, column=24, pady=20, padx=5, columnspan=12)

self.btn\_cfm.bind('<Button-1>', self.btn\_cfm\_ev)

self.btn\_del.bind('<Button-1>', self.btn\_del\_ev)

self.btn\_rtn.bind('<Button-1>', self.btn\_rtn\_ev)

def btn\_cfm\_ev(self, event):

global Student

AppendSet = list(

[self.v\_num.get(), self.v\_name.get(), self.sexvalue.get(), self.v\_Cgrade.get(), self.v\_Mgrade.get(),

self.v\_Egrade.get()])

try:

if [i for i in range(6) if AppendSet[i] != ''] != [0, 1, 2, 3, 4, 5]:

messagebox.showinfo(title='提示信息', message='数据不能为空！')

elif len(AppendSet[0]) != 10:

messagebox.showinfo(title='提示信息', message='学号必须为 10 位！')

elif AppendSet[0] in [Student[i][0] for i in range(1, len(Student))]:

messagebox.showinfo(title='提示信息', message='该学号已存在！')

elif int(AppendSet[0]) > 0 and 0 < float(AppendSet[3]) < 100 and 0 < float(

AppendSet[4]) < 100 and 0 < float(AppendSet[5]) < 100:

Student.append(Dp.Student.append(AppendSet))

self.v\_num.set('')

self.v\_name.set('')

self.sexvalue.set('男')

self.v\_Cgrade.set('')

self.v\_Mgrade.set('')

self.v\_Egrade.set('')

else:

messagebox.showinfo(title='提示信息', message='成绩必须在 0-100 之间!')

except BaseException:

messagebox.showinfo(title='提示信息', message='学号和成绩必须为数字!')

def btn\_del\_ev(self, event):

self.v\_num.set('')

self.v\_name.set('')

self.sexvalue.set('男')

self.v\_Cgrade.set('')

self.v\_Mgrade.set('')

self.v\_Egrade.set('')

def btn\_rtn\_ev(self, event):

global Empty, adMenuFrame, showFrame

self.destroy()

Empty.destroy()

adMenuFrame = AdMenuFrame(master=root)

showFrame = ShowFrame(master=root)

class DelStudent(Frame):

"""用于删除学生信息"""

def \_\_init\_\_(self, master=None):

super().\_\_init\_\_(master) # super()代表了父类的定义

self.master = master

self.place(relx=0.1, relwidth=0.8, relheight=1)

"""创建组件"""

self.label\_tit = Label(self, text='删除学生信息', width=12, height=1, fg='black', font=('黑体', 25))

self.label\_Num = Label(self, text='学 号:', width=12, height=1, fg='black', font=('黑体', 15))

self.label\_Name = Label(self, text='姓 名:', width=12, height=1, fg='black', font=('黑体', 15))

self.label\_Rmd = Label(self, text='支持模糊搜索,双击需要删除的行并确定', width=40, height=1, fg='black', font=('黑体', 10))

self.btn\_cfm = Button(self, text='确定', width=12, height=1, bg='gray', font=('黑体', 15))

self.btn\_del = Button(self, text='清空', width=12, height=1, bg='gray', font=('黑体', 15))

self.btn\_rtn = Button(self, text='返回上一级', width=12, height=1, bg='gray', font=('黑体', 15))

self.v\_num = StringVar()

self.v\_name = StringVar()

self.entry\_Num = Entry(self, textvariable=self.v\_num)

self.entry\_Name = Entry(self, textvariable=self.v\_name)

self.label\_tit.grid(row=0, column=18, columnspan=24, pady=40)

self.label\_Num.grid(row=1, column=12, columnspan=12)

self.label\_Name.grid(row=2, column=12, columnspan=12)

self.label\_Rmd.grid(row=8, column=12, columnspan=40)

self.entry\_Num.grid(row=1, column=30, columnspan=18)

self.entry\_Name.grid(row=2, column=30, columnspan=18)

self.btn\_cfm.grid(row=7, column=12, pady=20, padx=5, columnspan=12)

self.btn\_del.grid(row=7, column=24, pady=20, padx=5, columnspan=12)

self.btn\_rtn.grid(row=7, column=36, pady=20, padx=5, columnspan=12)

self.btn\_cfm.bind('<Button-1>', self.btn\_cfm\_ev)

self.btn\_del.bind('<Button-1>', self.btn\_del\_ev)

self.btn\_rtn.bind('<Button-1>', self.btn\_rtn\_ev)

def FindShowFrame(self):

"""用于显示查找到的学生信息"""

global Student

columns = Student[0]

widths = [100, 60, 60, 60, 60, 60, 60, 60, 90]

self.canvas = Canvas(self) # 创建canvas

self.canvas.place(relx=0, y=280, relwidth=1, height=250)

self.treeview = ttk.Treeview(self.canvas, show="headings", columns=columns) # 表格

self.treeview.place(relx=0, rely=0, relwidth=0.97, relheight=1)

self.treeview.bind('<Double-1>', self.del\_value) # 双击左键进入删除确定

self.VScroll1 = Scrollbar(self.canvas, orient='vertical', command=self.treeview.yview)

self.VScroll1.place(relx=0.97, rely=0, relwidth=0.03, relheight=1)

for i in range(len(widths)):

self.treeview.column(columns[i], width=widths[i], anchor='center') # 表示列,不显示

self.treeview.heading(columns[i], text=columns[i]) # 显示表头

for i in range(len(self.Find)): # 写入数据

self.treeview.insert('', i, values=self.Find[i])

def btn\_cfm\_ev(self, event):

global Student

self.Find = []

num = name = Save = True

for i in range(1, len(Student)):

if self.v\_num.get() != '' and self.v\_num.get() in Student[i][0]:

num = Save = False

if self.v\_name.get() != '' and self.v\_name.get() in Student[i][1]:

name = Save = False

if not Save:

self.Find.append(Student[i])

Save = True

self.FindShowFrame()

if len(self.Find) > 1:

messagebox.showinfo(title='提示信息', message='找到多个学生信息！')

elif self.v\_num.get() == self.v\_name.get() == '':

messagebox.showinfo(title='提示信息', message='请输入学生信息！')

if self.v\_num.get() != '' and num:

messagebox.showinfo(title='提示信息', message='按“学号”不存在该生信息！')

elif self.v\_name.get() != '' and name:

messagebox.showinfo(title='提示信息', message='按“姓名”不存在该生信息！')

def btn\_del\_ev(self, event):

self.v\_num.set('')

self.v\_name.set('')

def btn\_rtn\_ev(self, event):

global Empty, adMenuFrame, showFrame

self.destroy()

adMenuFrame = AdMenuFrame(master=root)

showFrame = ShowFrame(master=root)

def del\_value(self, event): # 双击进行删除

global Student

for self.item in self.treeview.selection():

# item = I001

self.item\_text = self.treeview.item(self.item, "values")

self.items = self.item

a = '是否删除学号为:{0},姓名为{1}的学生？'.format(self.item\_text[0], self.item\_text[1])

res = messagebox.askokcancel(title='提示消息', message=a)

if res == True:

for i in range(1, len(Student)):

if list(self.item\_text[0:]) == Student[i]:

del Student[i]

for i in range(len(self.Find)):

if list(self.item\_text[0:]) == self.Find[i]:

del self.Find[i]

self.canvas.destroy()

self.FindShowFrame()

class FindStudent(Frame):

"""用于显示学生查询系统的查询结果"""

def \_\_init\_\_(self, master=None):

super().\_\_init\_\_(master) # super()代表了父类的定义

self.master = master

self.place(relx=0.1, relwidth=0.8, relheight=1)

'''创建组件'''

global Student, FindResult

columns = Student[0]

widths = [100, 60, 60, 60, 60, 60, 60, 60, 90]

self.label\_tit = Label(self, text='查询结果', width=12, height=1, fg='black', font=('黑体', 25))

self.btn\_rtn = Button(self, text='返回上一级', width=12, height=1, bg='gray', font=('黑体', 15))

self.btnQuit = Button(self, text='退出系统', width=12, height=1, font=('黑体', 15), command=Exit)

self.canvas = Canvas(self) # 创建canvas

self.treeview = ttk.Treeview(self.canvas, show="headings", columns=columns) # 表格

for i in range(len(widths)):

self.treeview.column(columns[i], width=widths[i], anchor='center') # 表示列,不显示

self.treeview.heading(columns[i], text=columns[i]) # 显示表头

self.treeview.insert('', 1, values=FindResult)

self.treeview.place(relx=0, rely=0, relwidth=1, relheight=1)

self.canvas.place(relx=0, rely=0.2, relwidth=1, height=200)

self.label\_tit.place(relx=0.25, rely=0.1, relwidth=0.5, height=40)

self.btnQuit.place(relx=0.2, rely=0.6, width=150, height=40)

self.btn\_rtn.place(relx=0.55, rely=0.6, width=150, height=40)

self.btn\_rtn.bind('<Button-1>', self.btn\_rtn\_ev)

def btn\_rtn\_ev(self, event):

global Empty, registerFrame

self.destroy()

Empty = Frame(root, height=100)

Empty.pack()

registerFrame = RegisterFrame(master=root, TypeChoice='Student')

class AdminSetFrame(Frame):

"""学校信息初始化界面"""

def \_\_init\_\_(self, master=None):

super().\_\_init\_\_(master) # super()代表了父类的定义

self.master = master

self.pack(anchor='n')

"""创建组件"""

self.label\_tit = Label(self, text='管理员账户设置', width=24, height=1,

fg='black', font=('黑体', 25))

self.label\_adm = Label(self, text='请输入账号: ', width=18, height=1,

fg='black', font=('黑体', 15))

self.label\_rmd = Label(self, text='(可新增账号或修改已有账号密码)', width=18, height=1,

fg='black', font=('黑体', 8))

self.label\_pwd = Label(self, text='请输入密码: ', width=18, height=1,

fg='black', font=('黑体', 15))

self.label\_pwdagn = Label(self, text='再次输入密码密码:', width=18, height=1,

fg='black', font=('黑体', 15))

self.btn\_cfm = Button(self, text='确定', width=8, height=1, bg='gray', font=('黑体', 15))

self.btn\_rtn = Button(self, text='返回上一级', width=10, height=1, bg='gray', font=('黑体', 15))

self.btn\_cnc = Button(self, text='清空', width=8, height=1, bg='gray', font=('黑体', 15))

self.btnQuit = Button(self, text='退出系统', width=8, height=1, font=('黑体', 15), command=Exit)

self.v\_adm = StringVar()

self.v\_pwd = StringVar()

self.v\_pwdagn = StringVar()

self.entry\_adm = Entry(self, textvariable=self.v\_adm, width=24)

self.entry\_pwd = Entry(self, textvariable=self.v\_pwd, width=24)

self.entry\_pwdagn = Entry(self, textvariable=self.v\_pwdagn, width=24)

self.label\_tit.grid(row=0, column=0, columnspan=24, pady=30)

self.label\_adm.grid(row=1, column=0, columnspan=12)

self.label\_rmd.place(x=110, rely=0.2, width=200, height=20)

self.label\_pwd.grid(row=2, column=0, columnspan=12)

self.label\_pwdagn.grid(row=3, column=0, columnspan=12)

self.btn\_cfm.grid(row=5, column=0, pady=20, padx=20, columnspan=8)

self.btn\_cnc.grid(row=5, column=10, pady=20, padx=20, columnspan=8)

self.btn\_rtn.grid(row=5, column=20, pady=20, padx=20, columnspan=8)

self.btnQuit.grid(row=6, column=10, pady=20, columnspan=8)

self.entry\_adm.grid(row=1, column=15, pady=10, columnspan=24)

self.entry\_pwd.grid(row=2, column=15, pady=10, columnspan=24)

self.entry\_pwdagn.grid(row=3, column=15, pady=10, columnspan=24)

self.btn\_cfm.bind('<Button-1>', self.btn\_cfm\_ev)

self.btn\_cnc.bind('<Button-1>', self.btn\_cnc\_ev)

self.btn\_rtn.bind('<Button-1>', self.btn\_rtn\_ev)

def btn\_cfm\_ev(self, event):

global Admin

adm = pwd = Save = False

if self.v\_adm.get() == '':

messagebox.showinfo(title='提示信息', message='请输入账号！')

else:

adm = True

if self.v\_pwd.get() != self.v\_pwdagn.get():

messagebox.showinfo(title='提示信息', message='两次输入密码不一致！')

else:

pwd = True

if adm and pwd:

for x in range(len(Admin)):

if self.v\_adm.get() == Admin[x][0]:

Save = x

if Save == 0 and messagebox.askokcancel(title='提示消息', message='该账号已存在，是否修改密码？'):

Admin.append(['admin', self.v\_pwd.get()])

elif not Save and messagebox.askokcancel(title='提示消息', message='是否新建账号？'):

Admin.append([self.v\_adm.get(), self.v\_pwd.get()])

elif messagebox.askokcancel(title='提示消息', message='该账号已存在，是否修改密码？'):

Admin[Save] = [self.v\_adm.get(), self.v\_pwd.get()]

def btn\_cnc\_ev(self, event):

self.v\_adm.set('')

self.v\_pwd.set('')

self.v\_pwdagn.set('')

def btn\_rtn\_ev(self, event):

global root, Empty, adMenuFrame, showFrame

self.destroy()

Empty.destroy()

adMenuFrame = AdMenuFrame(master=root)

showFrame = ShowFrame(master=root)

def Windows(width, height, app='FirstFrame(master=root)', empty=True):

global root, Student, Empty, School

x = root.winfo\_screenwidth()

y = root.winfo\_screenheight()

align\_str = '%dx%d+%d+%d' % (width, height, (x - width) / 2, (y - height) / 2)

root.geometry(align\_str)

if Dp.Student.School is None:

root.title('学生成绩系统')

else:

root.title('学生成绩系统 BY {0}'.format(Dp.Student.School))

if empty:

Empty = Frame(root, height=100)

Empty.pack()

eval(app)

get\_time()

root.protocol('WM\_DELETE\_WINDOW', Exit)

root.mainloop()

def Save():

if Dp.FileProcess('Save', Students=Student) and Dp.AdScProcess('Save', Admins=Admin, Schools=School):

messagebox.showinfo(title='提示消息', message='数据保存成功,欢迎下次使用！')

def Exit():

# True or 'False'

res = messagebox.askokcancel(title='提示消息', message='是否退出系统？')

if res:

Save()

root.destroy()

def get\_time(): # 屏幕刷新时间

global root, TimeNow

time\_str = time.strftime("%H:%M:%S", time.localtime()) # 获得系统现在时间

TimeNow = Label(root, text=time\_str, width=12, height=1, fg='black', font=('黑体', 25))

TimeNow.place(relx=0.7, rely=0.9)

time\_str = time.strftime("%H:%M:%S", time.localtime()) # 获得系统现在时间

TimeNow.configure(text=time\_str) # 重新设置文本标签

root.after(1000, get\_time)

if \_\_name\_\_ == "\_\_main\_\_":

'''当在该界面运行时的操作'''

root = Tk()

Student = Dp.FileProcess('Read')

Admin, School = Dp.AdScProcess('Read')

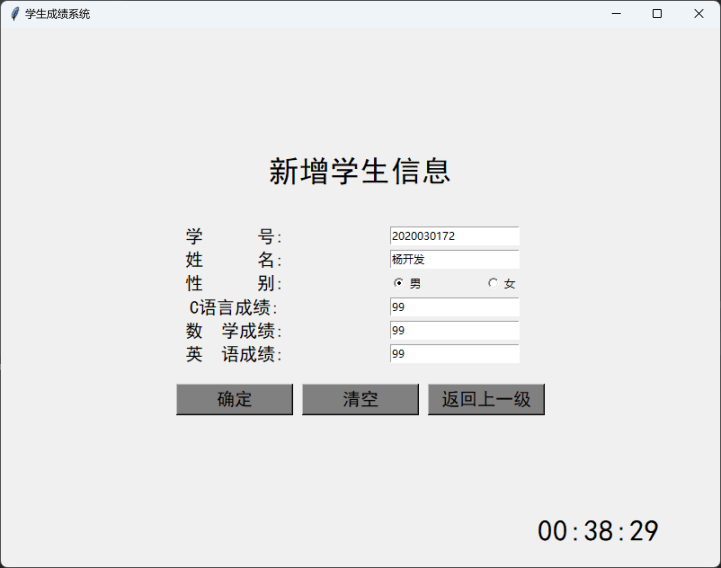
Windows(800, 600)

六、结果测试和验证

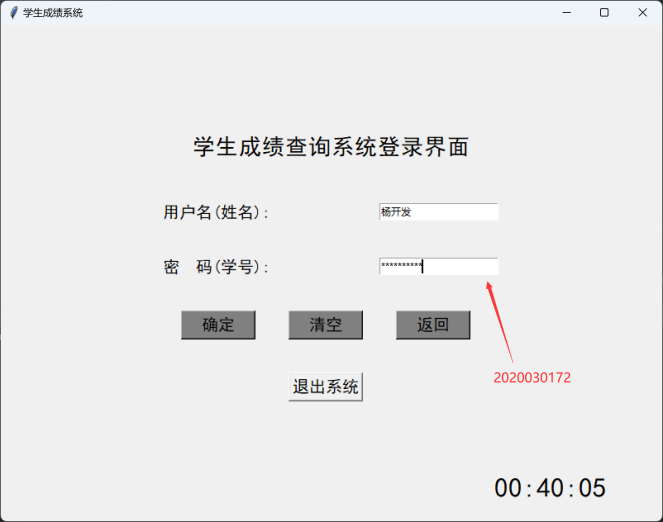


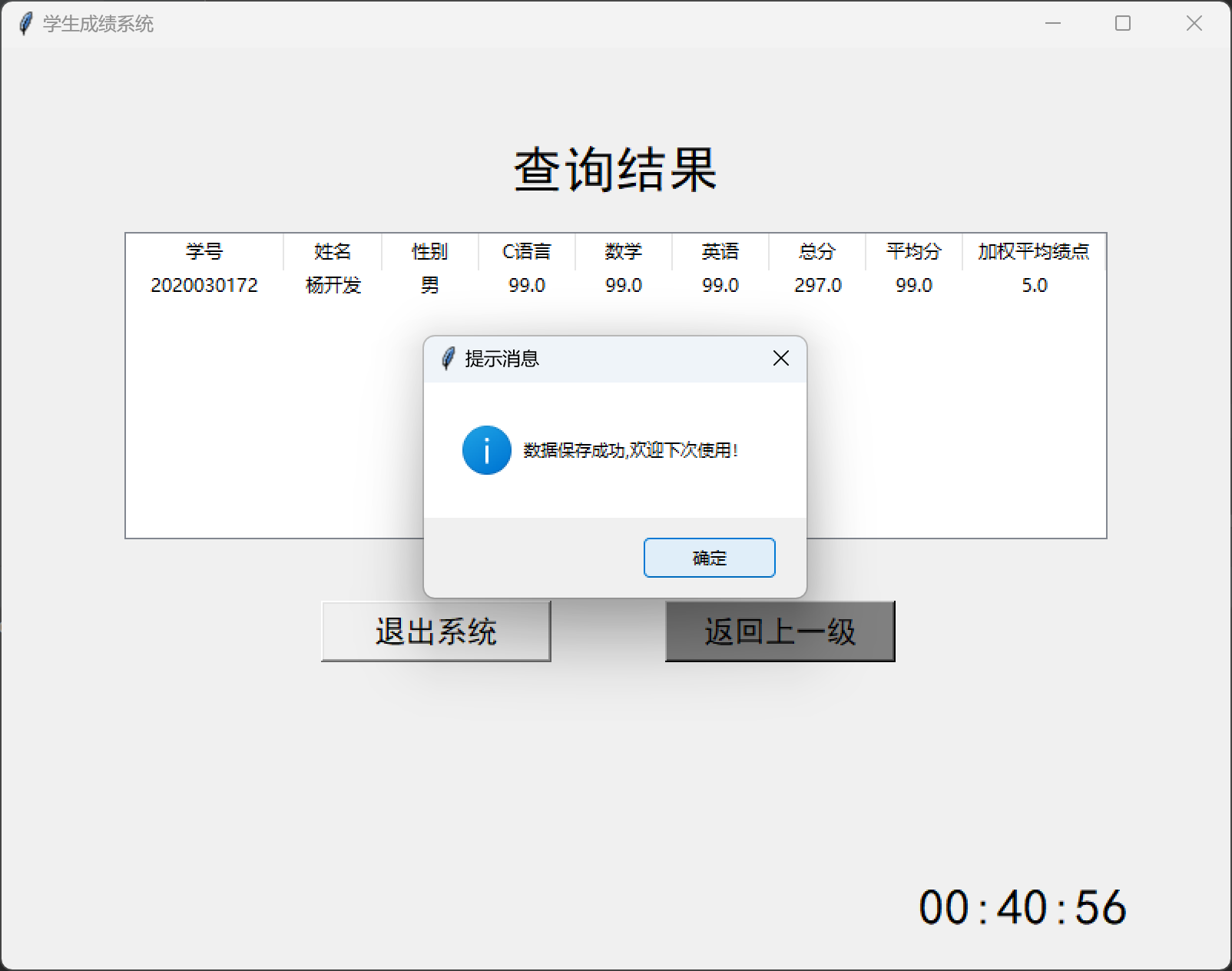


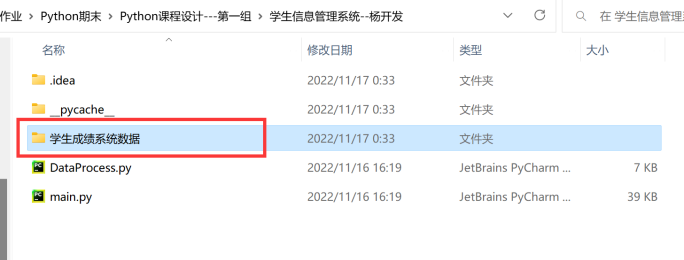


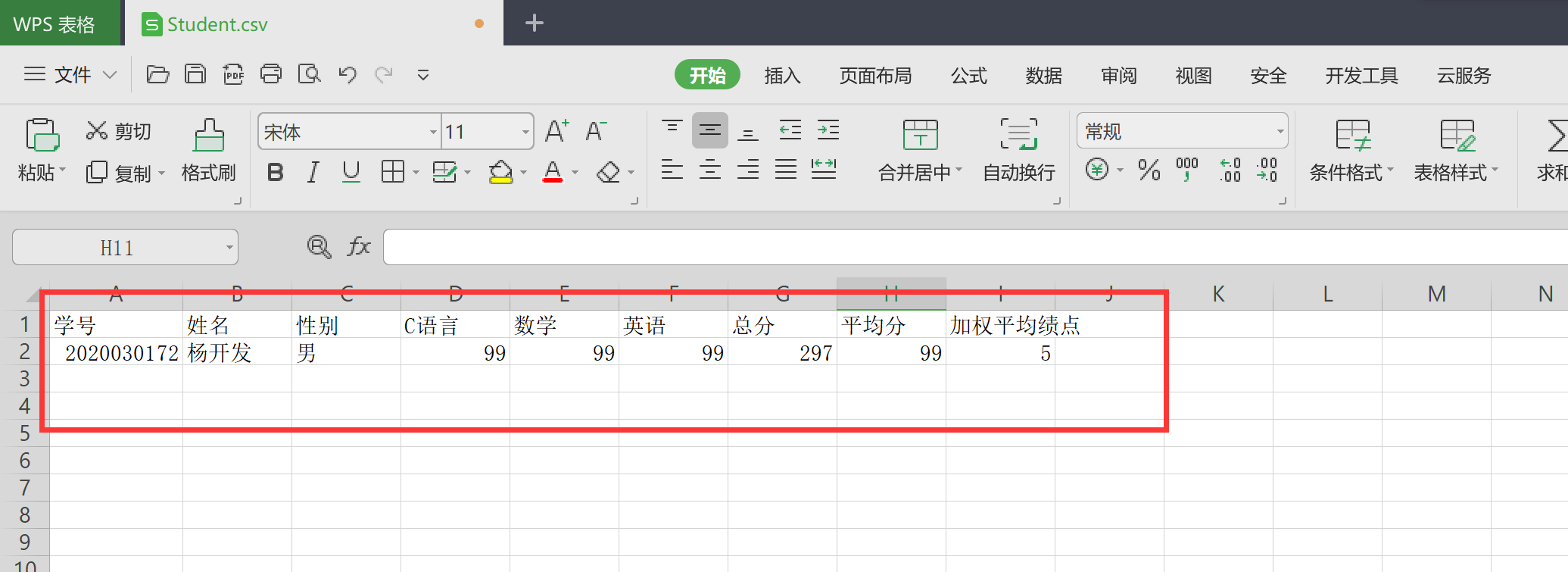




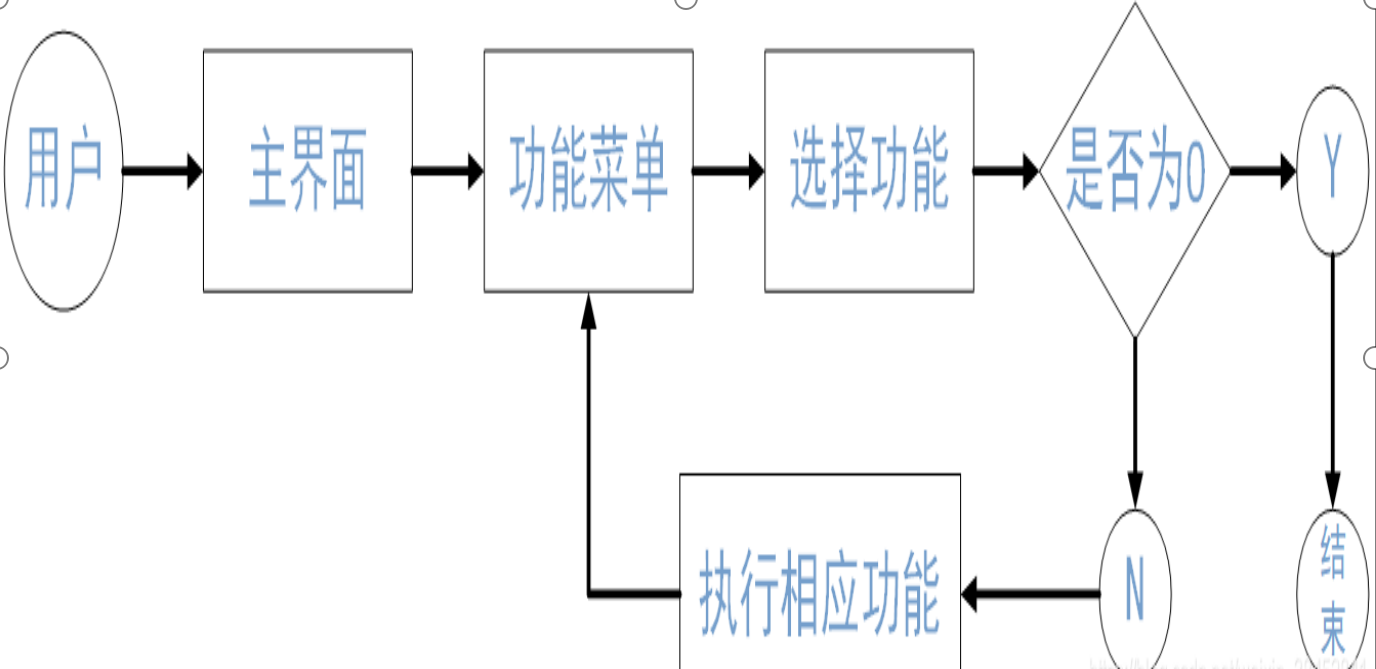








七、解题思路



八、过程总结

通过此次课程设计，使我们更加扎实的掌握了有关链表和结构体方面的知识，在设计过程中，中虽然遇到了一些问题，但经过一次又一次的思考，一遍又一遍的检查终于找出了原因。所在，也暴露出了前期我们在这方面的知识欠缺和经验不足。实践出真知，通过亲自动手制作，使我们掌握的知识不再是纸上谈兵。这次课程设计终于顺利完成了，在设计中遇到了很多问题，最后在老师的指导下，终于游逆而解。在今后社会的发展和学习实践过程中，一定要不懈努力，不能遇到问题就想到要退缩，一定要不厌其烦的发现问题所在，然后一一进行解决，只有这样，才能成功的做成想做的事，才能在今后的道路上劈荆斩棘，而不是知难而退，那样永远不可能收课程设计诚然是一门专业课，给我们很多专业知识以及专业技能上的提升，同时又是一门讲道课，一门辩思课，给了我们许多通往成功的道路，给了我们很多思考，给了我们莫大的空间。同时，设计让我们感触很深。使我们对抽象的理论有了具体的认识。通过这次课程设计，我们对建立一个工程有了一个新的认识，首先是想清楚这个系统的流程，将框架搭建起来后，再起写一些细节方面的东西，千万不能稀里糊涂的写，到最后没有用处，白白浪费时间。

我们认为此次课设不仅培养了独立思考、动手操作的能力，在各种其它能力上也都有了提高。同时也希望今后能够多花时间去思考，去动手写一些代码。