大作业项目总结报告

题目：D. Task Schedule

小组成员：程浩原（22373313）、孔嘉辉（21373456）、

柴振业（22373214）、籍泽宇（22373234）

2024年7月

一、功能简介（简述项目拥有的功能）

1. 用户登录与个人信息系统：填写用户基本信息，并使用数据库存储，支持注册与登录功能
2. 主页显示：逐条列出该用户的任务，允许用户通过勾选设置任务的完成状态，同时使用不同的颜色对不同任务状态进行可视化区分
3. 日历系统：可通过浏览日历查询某日进行中的任务
4. 任务提醒：支持查看今日到所选日期开始/结束的任务，并对今日要开始/结束根据不同紧急程度进行可视化区分
5. 任务调度：通过设置任务列表并提供空闲时间，系统将根据调度方法自动安排任务计划并给出日程表【TODO】

二、已完成任务

必做任务完成情况（7/7）

1. 用户登录与个人信息系统：填写用户基本信息，并使用数据库存储，支持注册与登录功能。
2. CRUD：（创建、读取、更新、删除）任务的操作，包括标题、类型、优先级、是否为每日任务、开始时间、截止时间、预计所需时间、内容描述等，支持任务修改和删除，使用数据库存储。
3. 任务状态区分：未开始、进行中、已完成和过期任务的任务状态以不同颜色可视化形式清晰地区分。
4. 任务完成确认：用户可通过在主页勾选代表任务已完成。
5. 任务提醒：根据所选日期筛选和显示时间段内开始/结束任务，对于今天要开始/完成的任务，根据任务紧急程度使用不同颜色进行可视化区分。
6. 日历系统：可通过浏览日历来查看某日进行中的任务。
7. 任务调度：通过设置任务列表并提供空闲时间，系统将根据调度方法自动安排任务计划并给出日程表。【TODO】

选做任务完成情况（3/4）

1. 利用权重安排任务：通过设置任务列表、设置任务的优先级权重以及给予空闲时间，可以自动安排任务计划。
2. 支持每日任务：对于每日任务，在设置后将在指定的时间内每天自动出现在任务列表中。
3. 任务类别划分：设置任务类别，如运动、学习等，同时支持自定义。
4. 总体设计方案（请详述具体功能的实现逻辑、核心代码及具体实现图）

**后端：（信息存储在本地的MySQL数据库）**

1. **数据库自动初始化（init\_database.py）**

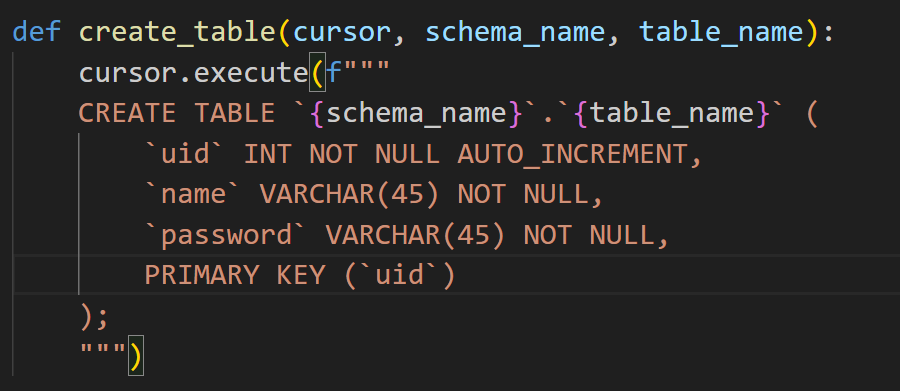
通过三个函数实现，其作用分别为：

检查数据库中具有某个名称的Schema是否存在、删掉具有某个名称的Schema、创建Schema。具体通过使用pymysql库以及sql语句实现。

****

1. **用户登录系统（login.py用户登录部分）**

使用login Table来存储相关信息，该表初始化信息如下：

****

使用自增且不重复的uid作为用户的唯一标识、name为用户学号，不可为空，最长可使用45个字符，密码同理。

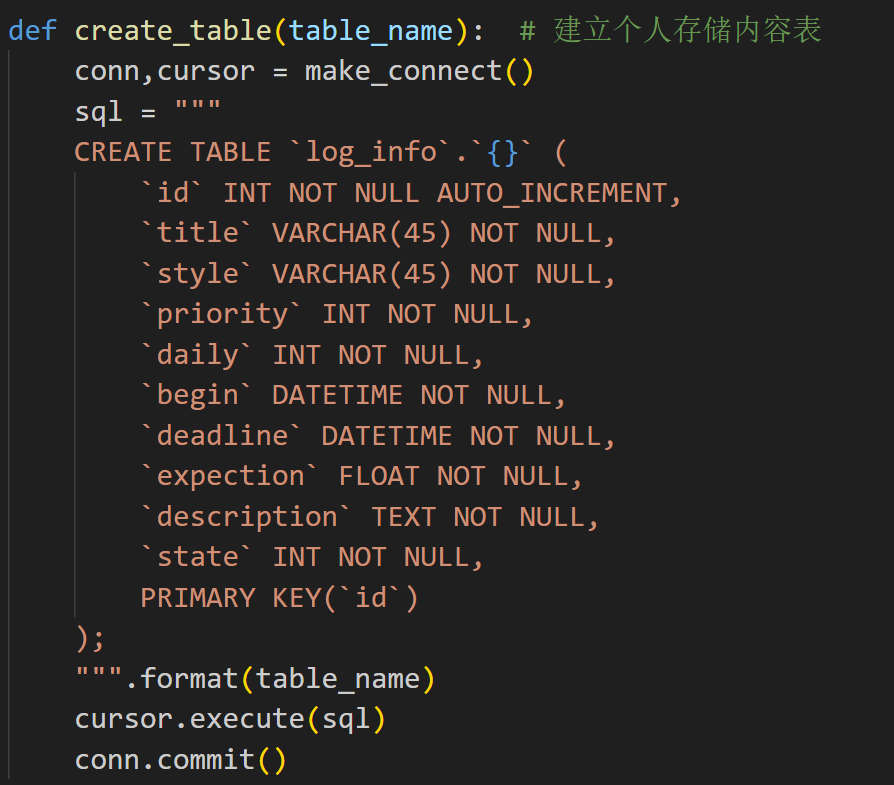
用户登录系统的后端实现了四个函数：

1. 注册模块add\_person(name, password)
2. 身份验证模块judge\_person(name, password)
3. 注销模块delete\_person(name, password)
4. 注册表总览scan\_login\_table()
5. **用户任务的增删改查模块（login.py用户任务部分）**

对每个用户，以用户名的小写作为其唯一数据库的名称建立Table。其初始化方式如下：

每个表都有十个属性，与下图对应依次为：

任务序号、标题、任务类型、任务优先级、是否为每日任务、开始日期、截止日期、预期完成时长、任务描述、任务状态。



对数据库的操作分为增删改查四类，分别为：

* 1. add\_schedule(name,task)
  2. delete\_schedule(name,task)
  3. edit\_schedule\_系列
  4. search\_schedule\_by\_date(name,datetime)和search\_schedule\_by\_title(name,title)

1. **任务排序模块（login.py用户任务部分）**

提供三种排序依据：

* 1. 按照任务的剩余时间排序sort\_schedules\_by\_deadline(tasks)
  2. 按照任务优先级排序sort\_schedules\_by\_priority(tasks)
  3. 综合排序（以任务的剩余时间为首要依据、以任务优先级为次要依据）sort\_schedules(tasks)

1. **其他模块**

前后端的链接如使用User类和Task类等实现操作接口化。

**前端：**

1.**任务删除**

具体实现逻辑：

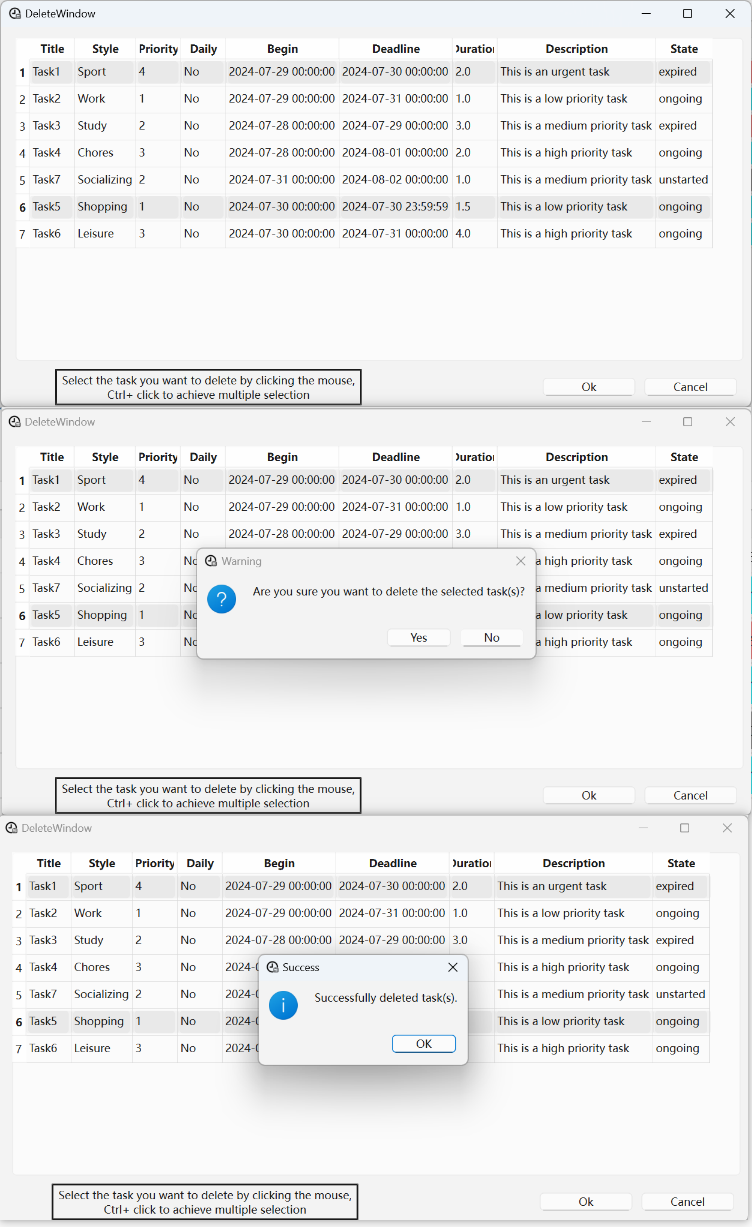
主页面点击删除按钮，数据库中获取用户任务列表，弹出子页面展示任务列表，通过点击选择要删除的任务，并在删除前进行确认，删除后将任务列表同步到主页面与数据库中。

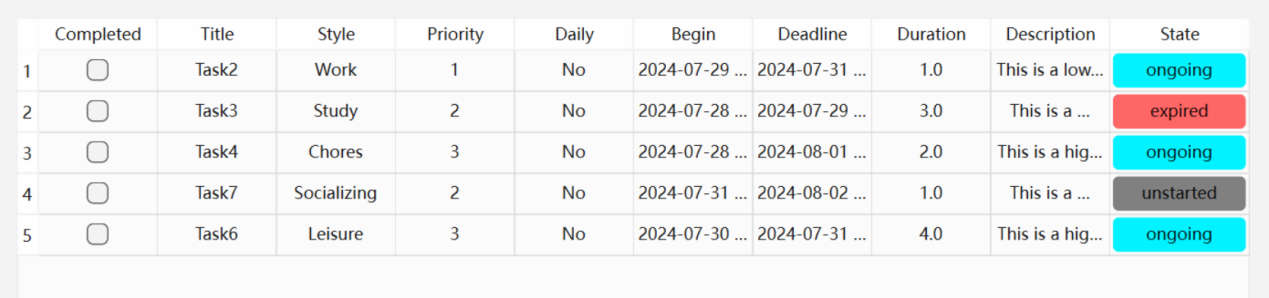
核心代码：

1. # ui\_delete.py
2. **def** setupUi(self, Delete\_Window):
3. # ... 省略其他代码 ...
5. # 逐行显示
6. **for** row\_idx, task **in** enumerate(globals.tasks):
7. print\_state = get\_and\_update\_state(globals.login\_user.get\_username(), task, datetime.now())
8. globals.tasks = scan\_schedule(globals.login\_user.get\_username())
9. print\_daily = "Yes" **if** task.daily != 0 **else** "No"
10. self.task\_table.setItem(row\_idx, 0, QTableWidgetItem(task.title))
11. self.task\_table.setItem(row\_idx, 1, QTableWidgetItem(task.style))
12. self.task\_table.setItem(row\_idx, 2, QTableWidgetItem(str(task.priority)))
13. self.task\_table.setItem(row\_idx, 3, QTableWidgetItem(print\_daily))
14. self.task\_table.setItem(row\_idx, 4, QTableWidgetItem(str(task.begin)))
15. self.task\_table.setItem(row\_idx, 5, QTableWidgetItem(str(task.deadline)))
16. self.task\_table.setItem(row\_idx, 6, QTableWidgetItem(str(task.expection)))
17. self.task\_table.setItem(row\_idx, 7, QTableWidgetItem(task.description))
18. self.task\_table.setItem(row\_idx, 8, QTableWidgetItem(print\_state))
19. # ... 省略其他代码 ...
21. **def** delete\_selected\_tasks(self):
22. """
23. 删除选中的任务项。
25. 从任务表中获取所有选中项的索引，然后根据这些索引获取相应的任务标题。向用户显示一个确认对话框，
26. 如果用户确认删除，那么将选中的任务从数据库中删除。如果删除操作失败，显示错误信息，并关闭当前窗口。
27. 如果删除成功，显示删除成功的提示信息，并触发任务删除信号，然后关闭当前窗口。
28. """
30. # 获取任务表的选择模型
31. selection\_model = self.task\_table.selectionModel()
33. # 获取所有选中行的索引
34. selected\_indexes = [index.row() **for** index **in** selection\_model.selectedRows()]
36. # 根据索引获取选中任务的标题
37. selected\_titles = []
38. **for** index **in** selected\_indexes:
39. item = self.task\_table.item(index, 0)
40. **if** item:
41. selected\_titles.append(item.text())
43. # 显示确认对话框，询问用户是否确认删除选中的任务
44. reply = QMessageBox.question(self, "Warning", "Are you sure you want to delete the selected task(s)?",
45. QMessageBox.Yes | QMessageBox.No, QMessageBox.No)
46. **if** reply == QMessageBox.No:
47. # 如果用户取消删除，则返回不做任何操作
48. **return**
50. # 遍历选中的任务标题，尝试从数据库中删除每个任务
51. **for** title **in** selected\_titles:
52. # 创建一个临时任务对象，用于删除操作
53. tmp\_task = Task(title, "", 0, 0, 0, 0, 0, "", 0)
54. # 调用删除任务的函数，传入登录用户的用户名和待删除的任务
55. result, msg = delete\_schedule(globals.login\_user.get\_username(), tmp\_task)
56. **if** **not** result:
57. # 如果删除失败，显示错误信息，并关闭窗口
58. QMessageBox.information(self, "Failed",
59. msg + " Please make sure every task to be deleted already exists.", )
60. self.close()
62. # 如果所有任务都成功删除，显示删除成功的提示信息
63. QMessageBox.information(self, "Success", "Successfully deleted task(s).")
64. # 触发任务删除信号
65. self.taskDeleted.emit()
66. # 关闭当前窗口
67. self.close()

70. # homepage.py
71. **def** clickDeleteButton(self):
72. """
73. 点击删除按钮的响应函数。
75. 弹出删除确认窗口，并连接删除操作完成后更新主窗口任务列表的信号。
76. """
77. # 初始化删除确认窗口
78. self.deleteWindow = Ui\_Delete()
79. # 连接删除任务完成后更新主窗口任务列表的信号
80. self.deleteWindow.taskDeleted.connect(self.updateHomeTasks)
81. # 显示删除确认窗口
82. self.deleteWindow.show()

具体实现图：





2.**任务提醒**

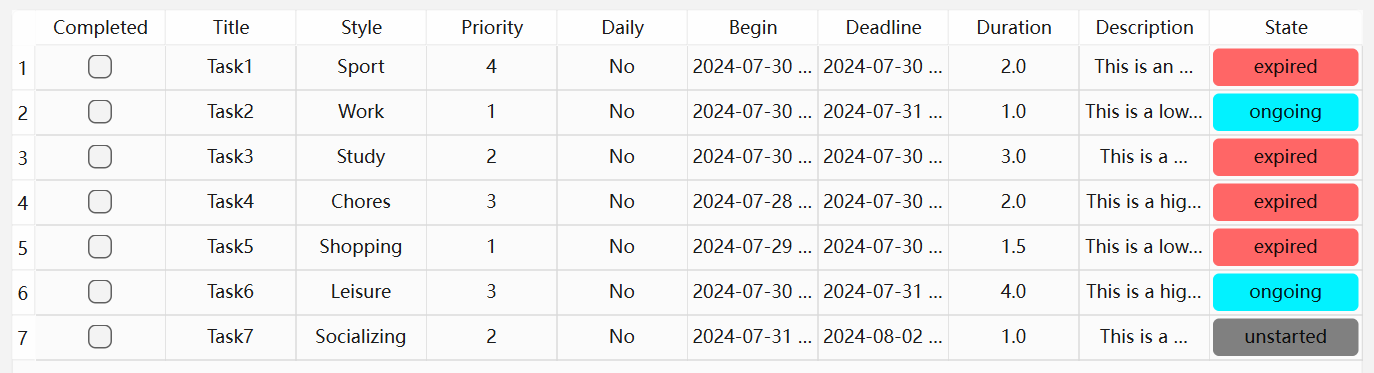
具体实现逻辑：

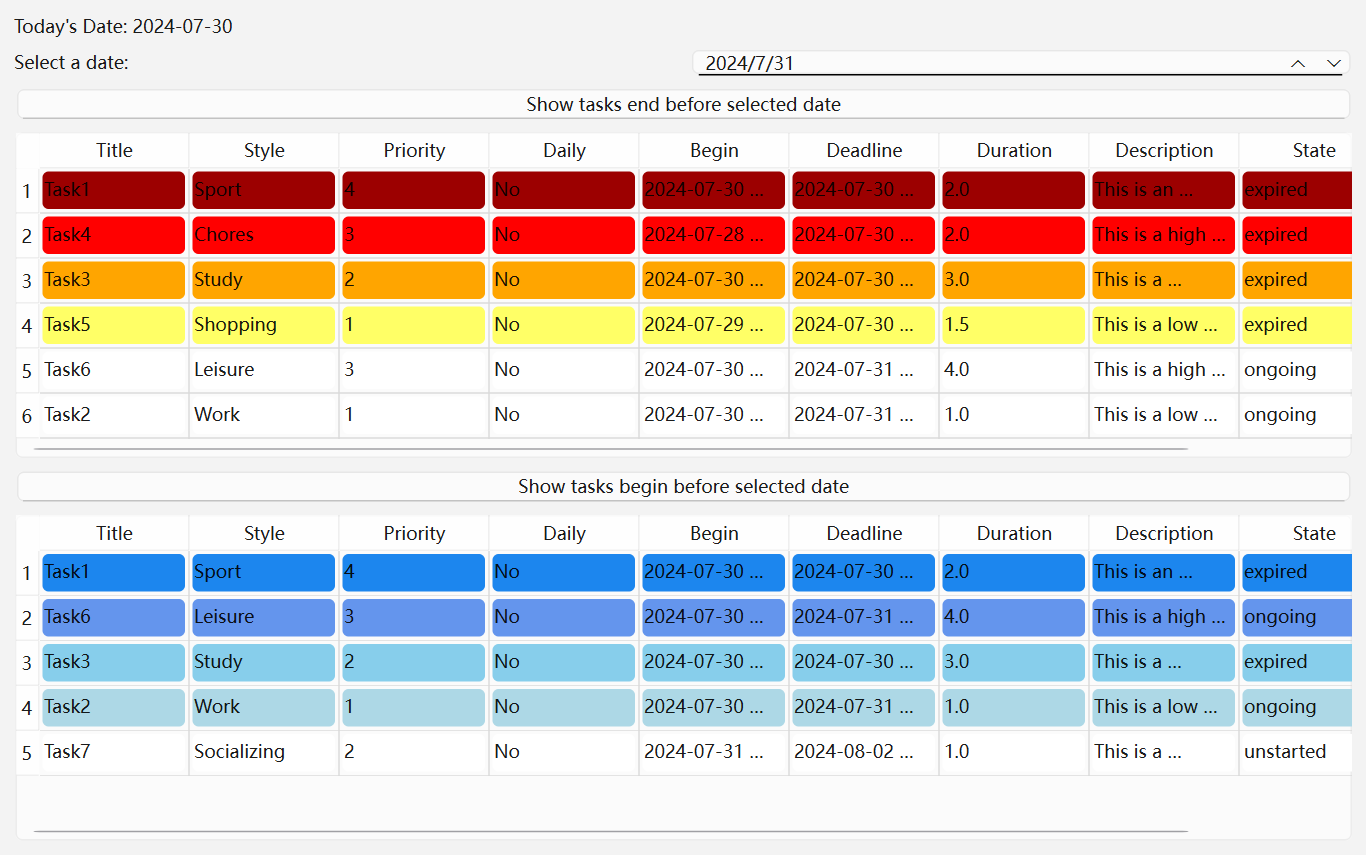
通过点击主页reminder按钮，跳转至任务提醒页面，用户可选择日期查看时间段内开始/结束的任务，同时对今天开始/结束的任务根据不同紧急程度通过颜色梯度来可视化区分。

核心代码：

1. # ui\_reminder.py
2. **def** update\_end\_table(self):
3. """
4. 更新结束表，显示所有在指定日期范围内且未完成的任务。
6. 该方法首先获取当前日期编辑框中的日期，并清空表格中的所有行。然后，它遍历全局任务列表，
7. 选择那些截止日期在今天和所选日期之间且状态不是已完成的任务。选中的任务按截止日期和优先级排序后，
8. 被添加到表格中。每个任务的各个属性（如标题、样式、优先级等）被添加到表格的相应单元格中。
9. 如果任务的截止日期是今天，单元格的背景色将根据任务的优先级被设置为相应的颜色。
10. """
11. # 获取选择的日期
12. chosen\_date = self.date\_edit.date().toPython()
13. # 清空表格行
14. self.table\_widget.setRowCount(0)  # 清空表格
16. # 初始化选中任务的列表
17. selected\_tasks = []
18. # 筛选任务（今日及今日后，指定日期前）并添加到selected\_tasks列表
19. **for** task **in** globals.tasks:
20. **if** date.today() <= task.deadline.date() <= chosen\_date **and** task.state != 2:  # task.state != 'completed':
21. selected\_tasks.append(task)
22. # 对选中的任务按截止日期和优先级进行排序
23. # 按deadline升序，priority降序
24. selected\_tasks.sort(reverse=False, key=**lambda** task: (task.deadline, -task.priority))
26. # 遍历选中的任务，并将它们添加到表格中
27. **for** task **in** selected\_tasks:
28. # 获取当前表格的行数，然后在末尾插入一行
29. row\_position = self.table\_widget.rowCount()
30. self.table\_widget.insertRow(row\_position)
31. # 根据任务状态更新任务状态，并重新扫描计划
32. print\_state = get\_and\_update\_state(globals.login\_user.get\_username(), task, datetime.now())
33. globals.tasks = scan\_schedule(globals.login\_user.get\_username())
34. # 判断任务是否为每日任务
35. print\_daily = "Yes" **if** task.daily != 0 **else** "No"
36. # 遍历任务属性，创建单元格并设置内容
37. **for** column, content **in** enumerate(
38. [task.title, task.style, task.priority, print\_daily, task.begin,
39. task.deadline, task.expection, task.description, print\_state]):
40. item = QTableWidgetItem(str(content))
41. # 根据任务的截止日期设置单元格的背景色
42. **if** task.deadline.date() == date.today():
43. # 设置当日任务颜色
44. item.setBackground(self.get\_priority\_color(task.priority))
45. **else**:
46. # 设置非当日任务的背景颜色为白色
47. item.setBackground(QColor(255, 255, 255))
48. # 在表格中插入单元格
49. self.table\_widget.setItem(row\_position, column, item)

具体实现图：





四、项目运行过程

请用尽可能简单的语言描述我们应该如何运行你们提交的代码，比如需要安装的库，如何注入数据等。

1. 需要安装的库：pymysql、PySide6…
2. 本地需完成mysql配置：下载MySQL后运行init\_database.py文件即可自动化生成log\_info Schema及login Table，以此作为以下程序运行的基础。
3. 在login. make\_connect()中修改password为本地MySQL用户root@localhost的连接密码。
4. 完成各项配置后，运行main.py即可。

五、项目总结

项目总体完成情况概述，是否符合自己预期。

六、课程学习总结

**1、课程收获和难点分析（小组成员是否有Python或大作业要求的基础，做完这个大作业自我感觉是否有提高等其他收获，本次项目感觉最困难的地方在哪里）**

**2、教师授课评价（老师上课过程的一些建议，以及希望老师之后能够介绍一些什么东西）**

**3、助教评价**

**4、当前课程教授内容评价与课程进一步改进建议**

七、主要参考资料

**请将此项目报告命名为项目总结报告-2024Python大作业-题目编号-小组编号，如项目总结报告-2024Python大作业-A题-01组，项目视频命名为项目视频-2024Python大作业-题目编号-小组编号，项目源代码压缩包命名为项目源代码-2024Python大作业-题目编号-小组编号。**

**并将项目报告+项目展示视频 + 项目源代码打包后命名为2024Python大作业-题目编号-小组编号，于2024年7月31号23：59前上传至课程群中的北航云盘链接。**