信息工程系

教

案

课程名称： 专业技能实训 IV

教　　师： 黄梅佳、黄苗苗

总 学 时 ： 36+36

理论学时： 0

实训学时： 36+36

上课班级： 计算机应用技术 241 1 组

授课学期： 25-26（2）

专业技能实训 IV （Word+Python 办公自动化）

授课课程：专业技能实训 IV

授课内容：Word 2016 固定版式长文档编排 + Python 办公自动化

总学时：36 学时（每周前 2 学时，共 18 周）

授课对象：计算机应用技术专业

授课形式：机房实操授课

思政融入理念：结合实训课特点，将工匠精神、职业规范、责任意识、数据安全意识融入每节课，培养学生严谨细致、精益求精的职业素养，树立正确的职场价值观。

一、课程整体教学目标

（一）知识目标

- 掌握 Word 2016 固定版式长文档（毕业论文/公文）的排版规范与核心操作；
- 熟练掌握 Python 办公自动化相关库（openpyxl、smtplib、re 等）的使用方法；
- 掌握 Excel 数据读取、写入、批量处理及图表生成的核心逻辑；
- 理解 SMTP 协议原理，掌握 Python 批量发送邮件的方法；
- 掌握正则表达式的基本语法与文本批量匹配、数据清洗技巧。

（二）能力目标

- 能独立完成 Word 长文档的标准化排版（封面、目录、页眉页脚、三线表等）；
- 能运用 Python 编写脚本，实现 Excel 数据的批量读取、编辑、格式设置及图表生成；
- 能使用 Python 实现纯文本、HTML 格式及带附件的批量邮件发送；
- 能运用正则表达式完成文本批量匹配、数据清洗等办公场景需求；
- 能独立调试、优化 Python 办公自动化脚本，解决实操中的常见问题。

（三）素养目标（含思政元素）

- 养成规范排版、规范编码的职业习惯，注重细节与格式统一性，渗透工匠精神；
- 建立自动化、高效化的办公思维，提升办公效率意识，培养务实肯干的工作态度；
- 培养独立思考、主动解决问题的能力，增强实操自信心，树立终身学习理念；
- 养成代码注释清晰、文档整理规范的良好习惯，契合职场需求，强化责任意识；
- 树立数据安全意识，规范使用办公数据，尊重他人劳动成果，杜绝抄袭行为。

二、教学重难点汇总

（一）教学重点

- Word 2016 长文档排版：分节符使用、多级列表设置、交叉引用与自动目录生成；
- Python-Excel 自动化：openpyxl 库的基础操作、数据读取与写入、公式与图表生成；
- Python 批量邮件发送：SMTP 协议配置、带附件邮件编写与批量发送逻辑；
- 正则表达式：字符组、量词、分组匹配及文本批量处理技巧。

（二）教学难点

- Word 长文档分节排版：页眉页脚分节设置、页码连续与断开控制；
- Python-Excel 自动化：单元格格式批量设置、工作簿/工作表拆分与合并；
- Python 批量邮件发送：SMTP 服务器配置、附件路径设置与异常处理；
- 正则表达式：贪婪/懒惰匹配的区别、复杂文本匹配逻辑的编写与调试。

三、教学准备

（一）硬件准备

计算机机房（一人一机），配置：Windows 10 及以上操作系统、Office 2016、Python 3.8+、PyCharm。

（二）软件准备

- Python 第三方库：openpyxl、smtplib、email、re、pandas（提前安装）；
- 教学软件：PPT 课件（融入思政案例）、实操演示视频、Python 代码模板（规范注释示例）。

（三）素材准备

- Word 素材：论文封面模板、空白长文档、三线表示例、交叉引用素材、规范与不规范排版对比案例；
- Excel 素材：空白工作簿、测试数据表格、数据透视表示例、包含敏感数据的警示案例；
- Python 素材：代码模板（带规范注释）、测试脚本、邮件发送测试账号、正则匹配测试文本；
- 任务单：每周实操任务清单、重难点提示、考核标准、思政素养评价要点；
- 思政素材：职场规范案例、工匠精神典型案例、数据安全违规警示案例。

四、分周课时教案（共 18 周，每周 2 学时）

第 1 周 教案（2 学时）

教学课题	Word 2016 固定版式：模板创建、论文封面/任务书制作
教学学时	2 学时
教学目标	• 知识目标：掌握 Word 模板创建的方法，了解论文封面/任务书的排版规范； • 能力目标：能独立创建 Word 模板，完成论文封面、任务书的标准化制作； • 素养目标（思政）：培养规范排版意识，注重封面元素的对齐与格式统一，渗透“细节决定成败”的职业理念，树立严谨细致的学习态度。
教学重难点	重点：Word 模板创建步骤、论文封面元素（标题、姓名、学号等）的排版； 难点：模板格式的固定与保存，封面元素的对齐与美观度控制。
教学过程（融入思政）	1. 导入：展示规范与不规范的论文封面案例，结合职场实际说明“规范排版是职业素养的基础”，强调固定版式对文档专业性的影响，明确本次课任务，渗透“工匠精神”的初步认知。 2. 理论讲解：简要讲解 Word 模板的作用、创建流程，论文封面的排版规范；结合思政案例，说明“规范排版不仅是技术要求，

	更是责任意识的体现”，比如毕业论文封面的规范直接影响论文的严肃性。 3. 实操演示：分步演示模板创建（新建模板、设置页面大小、保存为.dotx 格式），论文封面、任务书的制作（插入文本框、设置字体、对齐方式）；演示过程中强调“细节把控”，比如字体大小、间距的统一，渗透严谨细致的工作态度。 4. 学生实操：学生根据素材，独立完成模板创建、封面及任务书制作，教师巡回指导，解决学生操作难题；重点关注学生的排版细节，提醒学生“每一个格式的规范，都是职业素养的体现”，杜绝敷衍了事，对认真排版的学生及时给予肯定。 5. 小结与布置任务：总结本次课重难点，点评学生实操情况，表扬规范排版、态度认真的学生，引导学生重视细节；布置课后任务（完善模板，提交封面作品），要求学生在作品中体现规范意识，同时思考“规范排版在未来职场中的重要性”。
教学后记	学生对 Word 模板创建的兴趣较高，多数学生能按照要求完成封面制作，部分学生能主动关注排版细节，体现出良好的学习态度；思政元素融入自然，通过案例对比，让学生初步理解了规范排版与职业素养的关联。

第 2 周 教案（2 学时）

教学课题	Word 2016 固定版式：页面设置、分页/分节符、样式与多级列表
教学学时	2 学时
教学目标	• 知识目标：掌握 Word 页面设置、分页/分节符使用方法，理解样式与多级列表的作用； • 能力目标：能完成页面设置（页边距、纸张大小），正确使用分节符，创建规范的多级列表； • 素养目标（思政）：培养严谨的排版习惯，理解分节排版对长文档的重要性，渗透“条理清晰、权责明确”的职业思维，培养耐心细致的工作作风。
教学重难点	重点：分节符的使用（分页符、分节符的区别）、多级列表的创

	建与修改； 难点：分节符的插入位置，多级列表样式的统一与修改。
教学过程（融入思政）	1. 导入（5 分钟）：回顾上节课模板创建内容，展示长文档排版问题（页码混乱、标题层级不清晰），结合职场公文案例，说明“条理清晰的排版能提升工作效率，体现职业素养”，引出本次课核心内容，渗透“条理化、规范化”的职业理念。 2. 理论讲解（5 分钟）：简要说明页面设置参数、分节符的作用、样式与多级列表的关联；结合思政点，强调“分节符的合理使用，如同职场中的分工协作，能让文档结构更清晰、责任更明确”，培养学生的条理思维。 3. 实操演示（25 分钟）：演示页面设置（页边距、纸张大小、行距），分页/分节符插入与删除，样式创建（标题 1、标题 2），多级列表关联样式；演示过程中，强调“每一个操作都要严谨，避免因操作失误导致文档混乱”，渗透耐心细致的工作作风。 4. 学生实操（45 分钟）：学生打开上节课创建的模板，完成页面设置、分节符插入、多级列表创建，教师巡回指导；针对学生出现的分节符插入错误、多级列表样式不统一等问题，耐心讲解，引导学生反思“操作不严谨可能带来的后果”，强化责任意识；鼓励学生相互交流，培养协作意识。 5. 小结与布置任务（20 分钟）：总结分节符与多级列表的易错点，点评学生实操情况，表扬条理清晰、操作规范的学生；布置课后任务（完善长文档的页面与标题层级），要求学生在实操中注重条理，同时思考“如何通过规范排版体现自身的职业素养”。
教学后记	1. 课堂亮点：学生对分节符和多级列表的学习积极性较高，多数学生能正确区分分页符与分节符，掌握多级列表的创建方法；思政元素与教学内容结合紧密，学生能初步理解“条理化”对职业工作的重要性，部分学生能主动帮助操作有困难的同学，体现了协作精神。 2. 存在不足：少数学生对分节符的插入位置掌握不精准，导致文档结构混乱；部分学生创建多级列表时，不注重样式统一，敷衍了事；思政引导的针对性不足，对不同层次学生的思政教育未能做到因材施教。 3. 改进措施：下次课课前，针对分节符插入位置的易错点，制作简易操作口诀，帮助学生记忆；实操过程中，对操作不规范的学

	生进行个别辅导，强化规范意识；结合学生的实操表现，分层进行思政引导，对基础薄弱的学生重点培养耐心，对基础较好的学生引导其追求精益求精。

第 3 周 教案（2 学时）

教学课题	Word 2016 固定版式：三线表、题注、交叉引用、自动目录
教学学时	2 学时
教学目标	• 知识目标：掌握三线表制作、题注添加、交叉引用设置及自动目录生成的方法； • 能力目标：能制作规范的三线表，添加题注与交叉引用，生成并更新自动目录； • 素养目标（思政）：注重文档的规范性与实用性，理解交叉引用与自动目录对长文档的便捷性，渗透“高效办公、精益求精”的职业理念，培养责任意识与敬业精神。
教学重难点	重点：三线表的制作（边框设置）、交叉引用的添加与修改、自动目录生成； 难点：交叉引用的更新、自动目录与标题层级的关联及更新。
教学过程（融入思政）	1. 导入（5 分钟）：展示规范的三线表、交叉引用及自动目录案例，结合毕业论文、职场报告案例，说明“规范的表格与目录能提升文档的专业性和可读性，体现敬业精神”，明确本次课任务，渗透“精益求精”的工匠精神。 2. 理论讲解（5 分钟）：简要说明三线表的排版规范、题注与交叉引用的作用、自动目录的生成原理；结合思政点，强调“交叉引用与自动目录的使用，不仅能提升办公效率，更能体现对读者的尊重，是责任意识的体现”，引导学生树立“以用户为中心”的办公理念。 3. 实操演示（30 分钟）：分步演示三线表制作（插入表格、设置边框），题注添加（表格/图片题注），交叉引用设置（引用题注/标题），自动目录生成与更新；演示过程中，强调“每一个细节的规范，都是敬业精神的体现”，比如三线表的线条粗细、题注的编号规范，避免出现细节错误。

	4. 学生实操（40 分钟）：学生在之前的文档中，添加三线表、题注、交叉引用，生成自动目录，教师巡回指导，解决更新难题；重点关注学生的操作规范性，对三线表制作不规范、交叉引用错误的学生，耐心指导，引导学生反思“细节失误可能带来的误解”，强化责任意识；鼓励学生主动检查作品，培养自我纠错能力。 5. 小结与布置任务（20 分钟）：总结本次课重难点，点评学生实操情况，表扬操作规范、注重细节的学生；布置课后任务（完善文档的表格、题注与目录），要求学生在实操中追求精益求精，同时思考“如何通过规范操作提升文档的专业性，体现自身的敬业精神”。
教学后记	1. 课堂亮点：学生对三线表、交叉引用和自动目录的掌握情况较好，多数学生能独立完成操作，部分学生能主动优化表格和目录的格式，体现出精益求精的态度；思政元素融入自然，学生能理解规范操作与敬业精神的关联，自我纠错能力有所提升。 2. 存在不足：少数学生对交叉引用的更新方法掌握不熟练，导致引用内容与实际不符；部分学生制作三线表时，线条设置不规范，细节把控不到位；部分学生缺乏主动检查作品的习惯，责任心有待加强。 3. 改进措施：下次课课前，简要演示交叉引用更新的关键步骤，针对常见错误进行集中讲解；实操过程中，引导学生养成“操作完成后主动检查”的习惯，强化责任意识；结合职场案例，进一步深化“精益求精”的工匠精神，让学生明白细节决定文档的专业性。

第 4 周 教案（2 学时）

教学课题	Word 2016 固定版式：页眉页脚、页码、文档审阅、综合实操
教学学时	2 学时
教学目标	• 知识目标：掌握页眉页脚设置、页码插入与格式修改方法，了解文档审阅功能； • 能力目标：能设置分节页眉页脚、插入规范页码，使用审阅功能修改文档，完成长文档综合排版； • 素养目标（思政）：培养全面的排版思维，能独立完成长文

	档的全流程标准化排版，渗透“综合素养、责任担当”的职业理念，培养严谨细致、认真负责的工作态度，杜绝敷衍了事。
教学重难点	重点：分节页眉页脚的设置（不同节页眉页脚不同）、页码格式修改、文档审阅； 难点：分节页眉页脚的断开与连续设置，页码的连续与重新编号。
教学过程（融入思政）	1. 导入（5 分钟）：回顾前 3 周内容，展示长文档排版的完整案例，结合职场公文、毕业论文的实际要求，说明“全流程规范排版是职业综合素养的体现”，强调页眉页脚、页码的规范性直接影响文档的专业性，明确本次课综合实操任务，渗透“责任担当”的思政理念。 2. 理论讲解（5 分钟）：简要说明分节页眉页脚的设置逻辑、页码编号规则、文档审阅功能（批注、修订）；结合思政点，强调“文档审阅是对自己、对他人负责的体现，能避免错误传播，提升工作质量”，引导学生树立“认真负责、精益求精”的工作态度。 3. 实操演示（25 分钟）：演示分节页眉页脚设置（断开链接到前一节），页码插入与格式修改，文档审阅（批注、修订、接受/拒绝修订）；演示过程中，强调“综合实操需要全面考虑每一个细节，如同职场工作，需要统筹兼顾、认真负责”，渗透全面思维与责任意识。 4. 学生综合实操（45 分钟）：学生独立完成长文档全流程排版（结合前 3 周内容），教师巡回指导，解决综合排版难题；重点关注学生的综合操作能力，对排版不规范、敷衍了事的学生，及时提醒，引导学生反思“不认真排版可能带来的不良影响”；鼓励学生相互审阅作品，培养互助协作与自我完善的能力，渗透“互助共赢”的职业理念。 5. 小结与布置任务（20 分钟）：总结 Word 长文档排版的核心要点，点评学生综合实操作品，表扬操作规范、认真负责的学生，批评敷衍了事的行为；布置课后任务（提交完整的长文档排版作品），要求学生在作品中体现综合素养与责任意识，同时反思“自己在排版过程中存在的不足，如何改进”。
教学后记	1. 课堂亮点：多数学生能独立完成长文档综合排版，体现出较强的综合操作能力；部分学生能主动帮助操作有困难的同学，相互审阅作品，体现了互助协作精神；思政元素与综合实操结合紧

	密，学生能理解“综合素养与责任担当”在职业工作中的重要性，认真负责的态度有所提升。 2. 存在不足：少数学生对分节页眉页脚的断开与连续设置掌握不熟练，导致页码混乱；部分学生综合实操时缺乏统筹思维，排版顺序混乱，效率较低；个别学生存在敷衍了事的情况，作品排版不规范，责任意识有待加强。 3. 改进措施：下次课课前，针对分节页眉页脚设置的难点，制作分步演示视频，方便学生回顾；后续 Python 教学中，继续强化统筹思维与责任意识的培养；对敷衍了事的学生进行个别谈心，引导其树立正确的学习态度，明确“认真负责是职业发展的基础”。

第 5 周 教案（2 学时）

教学课题	Python-Excel：openpyxl 安装、工作簿/工作表基础操作
教学学时	2 学时
教学目标	• 知识目标：了解 openpyxl 库的作用，掌握 openpyxl 安装方法及工作簿、工作表的基础操作； • 能力目标：能安装 openpyxl 库，创建、打开、保存工作簿，创建、删除、重命名工作表； • 素养目标（思政）：培养规范编码习惯，了解 Python 自动化办公的便捷性，激发学习兴趣，渗透“创新思维、高效办公”的职业理念，培养耐心细致、主动解决问题的能力。
教学重难点	重点：openpyxl 库的安装，工作簿的创建与保存，工作表的基础操作； 难点：openpyxl 库安装失败的排查，工作簿路径的正确设置。
教学过程（融入思政）	导入（5 分钟）：对比 Excel 手动操作与 Python 自动化操作的效率，展示 Python 处理 Excel 的案例（如批量创建工作表），结合职场实际说明“自动化办公是提升工作效率的关键，创新思维能助力职业发展”，引出本次课核心内容，渗透“创新、高效”的职业理念。 理论讲解（5 分钟）：简要说明 openpyxl 库的功能、安装命令，

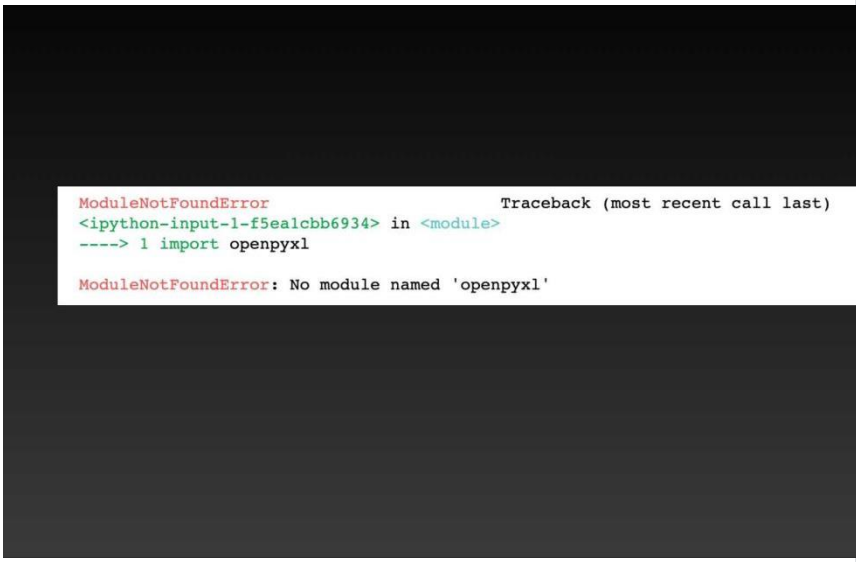
工作簿与工作表的关系；结合思政点，强调“规范编码是避免错误、提升效率的基础，如同职场工作，规范操作能减少失误”，引导学生养成规范编码的习惯；同时鼓励学生主动探索，培养创新思维。

实操演示（25 分钟）

1.安装第三方库

为了能够使用 Python 对 Excel 文件进行操作，我们需要安装第三方库 openpyxl。
1.1 检查有没有安装第三方库

首先，检查有没有安装 openpyxl 库。在 Python 的终端或 IDE 里输入下面的命令：import openpyxl 如果没有报错，说明已经安装了 openpyxl。如果报如下错误：



则说明没有安装 openpyxl。这时候，我们可以使用如下命令来安装第三方库 openpyxl。pip install openpyxl
或者在以上命令后边加上：-i https://pypi.tuna.tsinghua.edu.cn/simple 加快下载速度
2.Excel 的基本概念

相信不少同学在生活工作中都使用过 Excel。接下来我们就来了解一下 Excel 的几个基本概念：
2.1 工作簿

一个 Excel 电子表格文档称为一个工作簿，一个工作簿保存在扩展名为 .xlsx 的文件中。
2.2 工作表

每个工作簿可以包含多个表（也称为工作表），用户当前查看的表（或关闭 Excel 前最后查看的表），称为活动表。

2.3 单元格

每个表都有一些列（地址是从 A 开始的字母）和一些行（地址是从 1 开始的数字）。在特定行和列的方格称为单元格。每个单元格都包含一个数字或文本值。单元格形成的网格和数据构成了表。

3.使用 Python 操作 Excel

在了解了 Excel 的基本概念后，我们来看如何使用 Python 进行 Excel 文件的操作。

3.1 工作簿的创建

```
# 工作簿的创建
from openpyxl import Workbook

wb = Workbook()
wb.save('test.xlsx')
```

在上面的代码中，我们首先导入 openpyxl 库，接着创建一个 Workbook 对象并取名保存即可。这样在当前目录下便生成了一个 Excel 文件。如下图所示：

3.2 工作表的创建

讲完工作簿的创建后，我们来看下工作表的创建。

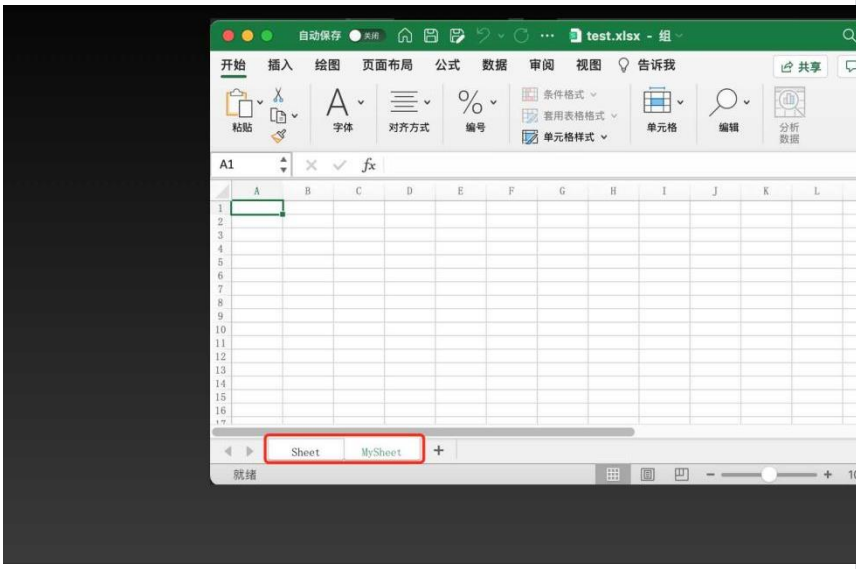
```
# 工作表的创建
from openpyxl import Workbook

wb = Workbook()

wb.create_sheet('MySheet')
wb.save('test.xlsx')
```

在上面的代码中：

1.导入 `openpyxl` 库并创建一个 `Workbook` 对象 `wb`。
2.使用 `create_sheet` 方法创建工作表，在调用 `create_sheet` 方法时

只需要传入工作表的名称即可。

由于在创建 Workbook 对象 wb 时已经默认创建了一个工作表 Sheet，所以在调用 create_sheet 创建 MySheet 工作表之后，工作簿中包含了两个工作表 Sheet 和 MySheet。如下图所示：



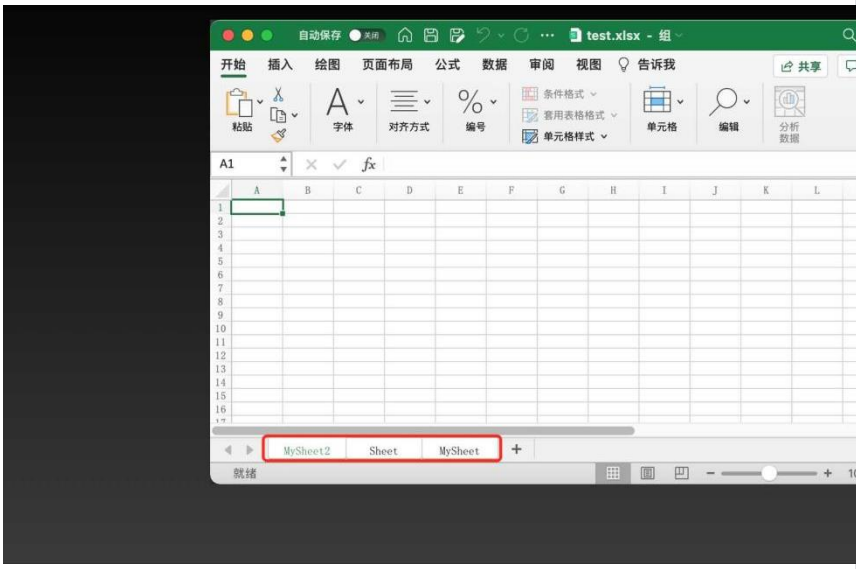
上面在调用 create_sheet 方法创建工作表时，只传入了工作表的名称。还有另外一种调用 create_sheet 的方式，就是不但传入工作表的名称，而且传入工作表在工作簿中的位置。接着上面的代码，添加如下代码：

```
wb.create_sheet('MySheet2', 0)
wb.save('test.xlsx')
```

上面的代码中，在调用 create_sheet 方法时，传入了工作表的名称和工作表在工作簿的位置。上述代码添加后的完整代码为：

```
from openpyxl import Workbook

wb = Workbook()

wb.create_sheet('MySheet')
wb.save('test.xlsx')

wb.create_sheet('MySheet2', 0)
wb.save('test.xlsx')
```

上述的代码执行之后，工作簿中的工作表的布局如下图所示：



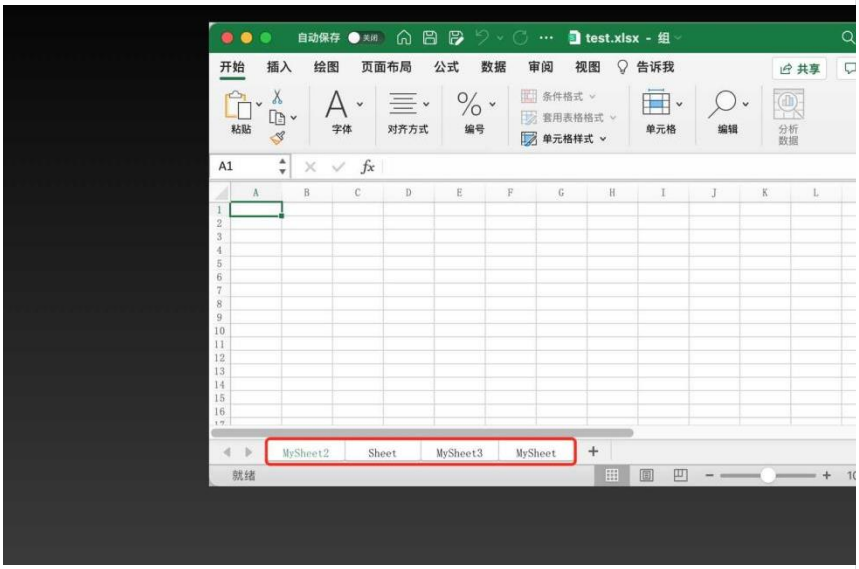
由于指定 MySheet2 工作表的位置为 0，所以 MySheet2 工作表是工作簿的第一个工作表。我们使用上面的方法再来添加一个工作表，接着上面的代码，添加如下代码：

```
wb.create_sheet('MySheet3', 2)
wb.save('test.xlsx')
```

上述代码添加后的完整代码为：

```
from openpyxl import Workbook

wb = Workbook()

wb.create_sheet('MySheet')
wb.save('test.xlsx')

wb.create_sheet('MySheet2', 0)
wb.save('test.xlsx')

wb.create_sheet('MySheet3', 2)
wb.save('test.xlsx')
```

上述的代码执行之后，工作簿中的工作表的布局如下图所示：



由于指定 MySheet3 的位置为 2，所以 MySheet3 为工作簿中的第三个工作表。

3.3 查看工作表的名字

工作表创建完成后，我们可以查看工作簿中的工作表，接上面的代码，添加如下代码：wb.sheetnames 上述代码添加后的完整代码为：

```
from openpyxl import Workbook

wb = Workbook()

wb.create_sheet('MySheet')
wb.save('test.xlsx')

wb.create_sheet('MySheet2', 0)
wb.save('test.xlsx')

wb.create_sheet('MySheet3', 2)
wb.save('test.xlsx')

wb.sheetnames
```

3.4 改变工作表的名字

通过代码我们同样可以改变工作表的名字，接上面的代码，添加如下代码：

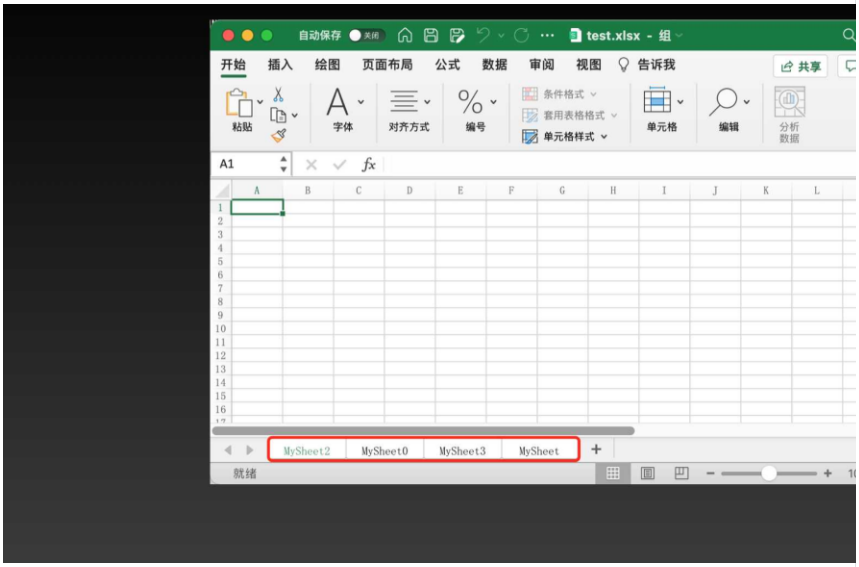
```
ws = wb['Sheet']
ws.title = 'MySheet0'
wb.save('test.xlsx')
wb.sheetnames
```

上述代码添加后的完整代码为：

```
from openpyxl import Workbook

wb = Workbook()
```

```
wb.create_sheet('MySheet')
wb.save('test.xlsx')

wb.create_sheet('MySheet2', 0)
wb.save('test.xlsx')

wb.create_sheet('MySheet3', 2)
wb.save('test.xlsx')

wb.sheetnames

ws = wb['Sheet']
ws.title = 'MySheet0'
wb.save('test.xlsx')
wb.sheetnames
```

在上面的代码中，首先获取工作表 Sheet，然后将其名字改为 MySheet0，最后进行工作簿的保存。执行完上述代码后，工作簿中工作表的情况如下图所示：



3.5 获取活跃表

用户当前查看的表（或关闭 Excel 前最后查看的表），称为活动表。通过代码我们可以获取活跃表。接上面的代码，添加如下代码：

```
ws = wb.active
ws.title
```

上述代码添加后的完整代码为：

```
from openpyxl import Workbook

wb = Workbook()

wb.create_sheet('MySheet')
wb.save('test.xlsx')
```

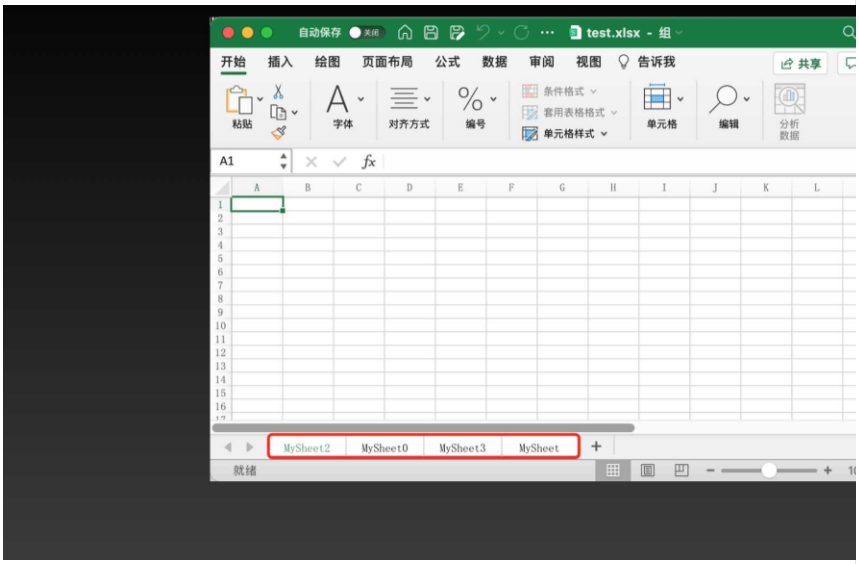
```
wb.create_sheet('MySheet2', 0)
wb.save('test.xlsx')

wb.create_sheet('MySheet3', 2)
wb.save('test.xlsx')

wb.sheetnames

ws = wb['Sheet']
ws.title = 'MySheet0'
wb.save('test.xlsx')
wb.sheetnames

ws = wb.active
ws.title
```

3.6 删除工作表

我们不但可以添加、修改工作表，还可以删除工作表。接上面的代码，添加如下代码：wb.sheetnames 上述代码添加后的完整代码为：

```
from openpyxl import Workbook

wb = Workbook()

wb.create_sheet('MySheet')
wb.save('test.xlsx')

wb.create_sheet('MySheet2', 0)
wb.save('test.xlsx')

wb.create_sheet('MySheet3', 2)
wb.save('test.xlsx')

wb.sheetnames

ws = wb['Sheet']
ws.title = 'MySheet0'
wb.save('test.xlsx')
wb.sheetnames

ws = wb.active
ws.title

wb.sheetnames
```

在删除之前，我们首先查看当前工作簿中的工作表。如下图所示：



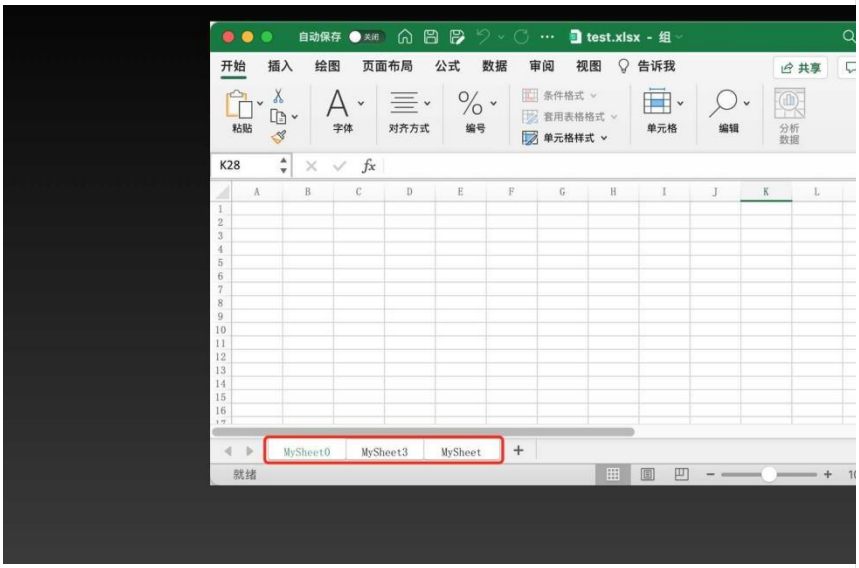
继续添加如下代码：

```
del wb['MySheet2']
wb.save('test.xlsx')
```

上述代码添加后的完整代码为：

```
from openpyxl import Workbook

wb = Workbook()

wb.create_sheet('MySheet')
wb.save('test.xlsx')

wb.create_sheet('MySheet2', 0)
wb.save('test.xlsx')

wb.create_sheet('MySheet3', 2)
wb.save('test.xlsx')

wb.sheetnames

ws = wb['Sheet']
ws.title = 'MySheet0'
wb.save('test.xlsx')
wb.sheetnames

ws = wb.active
ws.title

wb.sheetnames

del wb['MySheet2']
wb.save('test.xlsx')
```

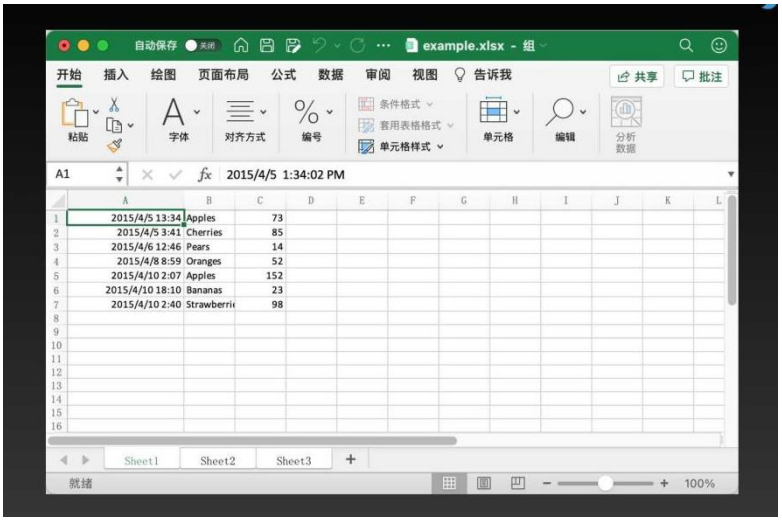
在上面的代码中，使用 del 关键字来删除工作表，删除工作表

	MySheet2 并保存之后，工作簿中的工作表如下图所示：  可以看到工作表 MySheet2 已经被删除。 小结与布置任务（20 分钟）：总结本次课重难点，点评学生实操情况，表扬编码规范、主动解决问题的学生；布置课后任务（编写脚本，完成工作簿与工作表的基础操作），要求学生规范编写代码、添加注释，同时思考“Python 自动化办公能解决哪些职场痛点，如何通过创新思维提升办公效率”。
教学后记	1. 课堂亮点：学生对 Python-Excel 自动化的学习兴趣较高，多数学生能顺利安装 openpyxl 库，完成基础操作；部分学生能主动尝试不同的操作方法，体现出创新思维；学生的规范编码意识有所提升，多数学生能为代码添加注释；遇到问题时，部分学生能主动排查、相互交流，解决问题的能力有所增强。 2. 存在不足：少数学生安装 openpyxl 库时遇到困难，缺乏主动排查问题的意识，过度依赖教师；部分学生编写代码时不规范，无注释、变量命名混乱；个别学生对工作簿路径的设置掌握不熟练，导致保存失败。 3. 改进措施：下次课课前，整理 openpyxl 库安装失败的常见原因及排查方法，分享给学生；实操过程中，加强对编码规范的指导，明确注释与变量命名的要求；引导学生养成“遇到问题先独立排查，再寻求帮助”的习惯，培养自主学习能力；结合职场案例，进一步强化“创新思维、高效办公”的理念。

第 6 周 教案（2 学时）

教学课题	Python-Excel：Excel 文档读取、单元格遍历、数据提取
教学学时	2 学时
教学目标	• 知识目标：掌握 openpyxl 库读取 Excel 文档的方法，理解单元格遍历的逻辑，掌握数据提取的核心技巧； • 能力目标：能读取 Excel 数据，实现单元格遍历与指定数据提取，处理简单的数据读取异常； • 素养目标（思政）：培养严谨的编码思维，注重数据读取的准确性，渗透“数据真实、责任至上”的职业理念，培养耐心细致、认真负责的工作态度，树立数据安全意识。
教学重难点	重点：Excel 数据读取方法，单元格遍历逻辑，指定数据提取技巧； 难点：单元格遍历的效率优化，数据读取异常的简单处理。
教学过程（融入思政）	导入（5 分钟）：结合职场数据处理案例（如员工信息读取、成绩数据提取），说明“数据读取的准确性直接影响工作决策，是责任意识的体现”，强调数据真实、严谨的重要性，引出本次课核心内容，渗透“数据安全、责任至上”的思政理念。 理论讲解（5 分钟）：简要说明 openpyxl 读取 Excel 数据的核心方法，单元格遍历的两种方式（行遍历、列遍历），数据提取的核心逻辑；结合思政点，强调“数据是职场决策的基础，读取数据时必须严谨，避免因数据错误导致决策失误”，同时提醒学生注重数据安全，不随意读取、传播敏感数据。 实操演示： 1.打开 Excel 文档 Excel 文档创建完成后，为了读取 Excel 文档。首先需要打开 Excel 文档。代码如下： import openpyxl wb = openpyxl.load_workbook("example.xlsx") type(wb) 在上面的代码中，首先导入第三方库 openpyxl。接着使用 load_workbook 方法来读取 Excel 文档，在使用 load_workbook 时，

只需要向 load_workbook 方法传入要打开的文档的名称即可。最后，可以通过 type 函数查看打开的文档的类型。文档 example.xlsx 的内容如下所示：



2.从工作簿取得工作表

从工作簿中取得工作表之前，首先看下工作簿中包含哪些工作表，接上面的代码，添加如下代码：wb.sheetnames 上述代码添加后的完整代码为：

```
import openpyxl

wb = openpyxl.load_workbook("example.xlsx")
type(wb)

wb.sheetnames
```

从输出结果中看到，工作簿总共包含了三个工作表，分别是：Sheet1、Sheet2、Sheet3。接着上面的代码，添加如下代码：

```
ws = wb['Sheet3']
ws.title
```

上述代码添加后的完整代码为：

```
import openpyxl

wb = openpyxl.load_workbook("example.xlsx")
type(wb)

wb.sheetnames

ws = wb['Sheet3']
ws.title
```

上面代码中，通过 object[name]的方式来获取工作表，想要获取哪个工作表，只需要传入工作表的名称即可。另外，我们还可以通过对象的 active 属性来获取当前活跃的工作表。接着上面的代码，添加

如下代码：

```
ws = wb.active
ws.title
```

上述代码添加后的完整代码为：

```
import openpyxl

wb = openpyxl.load_workbook("example.xlsx")
type(wb)

wb.sheetnames

ws = wb['Sheet3']
ws.title

ws = wb.active
ws.title
```

这样便通过对象的 active 属性获取到当前活跃的工作表。
3.从工作表中取得单元格

上面讲到了从工作簿中获取工作表，下面来讲从工作表中获取单元格。接着上面的代码，添加如下代码：

```
ws = wb['Sheet1']
wc = ws['A1']
wc.value
```

上述代码添加后的完整代码为：

```
import openpyxl

wb = openpyxl.load_workbook("example.xlsx")
type(wb)

wb.sheetnames

ws = wb['Sheet3']
ws.title

ws = wb.active
ws.title

ws = wb['Sheet1']
wc = ws['A1']
wc.value
```

上面添加的代码首先获取工作表 Sheet1，然后通过 object[name]的方式工作表中的单元格。这样便获取到了单元格 A1，并通过单元格的属性 value 获取到了单元格中所包含的值。

当列数较小时，使用字母来指定某个列并没有太多不方便，但是当列数逐渐增加，特别是在 Z 列之后，列开始使用两个字母 AA、AB、AC 等，这时候再通过字母来指定列时，便有诸多不便。这时候可以使用 cell() 方法来获取工作表中的单元格，在调用 cell() 方法时，传入参数 row 和 column。接着上面的代码，添加如下代码：

```
ws = wb['Sheet1']
wc = ws.cell(row=1, column=2)
wc.value
```

上述代码添加后的完整代码为：

```
import openpyxl

wb = openpyxl.load_workbook("example.xlsx")
type(wb)

wb.sheetnames

ws = wb['Sheet3']
ws.title

ws = wb.active
ws.title

ws = wb['Sheet1']
wc = ws['A1']
wc.value

ws = wb['Sheet1']
wc = ws.cell(row=1, column=2)
wc.value
```

上面添加的代码获取了 B1 单元格。在使用方法 cell() 方法获取单元格时，列和行的序号都是从 1 开始的。有了 cell() 方法，可以很方便地通过循环一次获取多个单元格。接着上面的代码，添加如下代码：

```
for i in range(1, 8, 2):
print(ws.cell(row=i, column=2).value)
```

上述代码添加后的完整代码为：

```
import openpyxl

wb = openpyxl.load_workbook("example.xlsx")
type(wb)

wb.sheetnames

ws = wb['Sheet3']
ws.title
```

```
ws = wb.active
ws.title

ws = wb['Sheet1']
wc = ws['A1']
wc.value

ws = wb['Sheet1']
wc = ws.cell(row=1, column=2)
wc.value

for i in range(1, 8):
   print(ws.cell(row=i, column=2).value)
```

添加的代码获取到了第二列从第一行到第七行的单元格的值。
4.列字母和数字之间的转换

由于 Excel 文档中的列名是使用字母来表示的，当别数较大时，通过字母不能很方便的得出是第几行。不过，openpyxl 模块提供了 column_index_from_string() 方法，使用此方法可以很方便的得出字母列所对应的列数。举例代码如下：

```
from openpyxl.utils import column_index_from_string

column_index_from_string('A')
column_index_from_string('AA')
column_index_from_string('AHP')
```

上面的代码得到了列 A、AA、AHP 所对应的列数分别为 1、27、900。
5.遍历行和列
5.1 遍历所有行的单元格

```
import openpyxl

wb = openpyxl.load_workbook("example.xlsx")
ws = wb['Sheet1']

for row in ws.rows:
   for item in row:
      print(item.value)
   print('--- END OF ROW ---')
```

上面代码我们以行的方向遍历了所有的单元格。使用了二层循环，第一次循环获取所有的行，第二层循环获取每行中的每个单元格。
5.2 遍历所有列的单元格

```
import openpyxl

wb = openpyxl.load_workbook("example.xlsx")
ws = wb['Sheet1']

for col in ws.columns:
```

```
    for item in col:
        print(item.value)
    print('--- END OF COL ---')
```

上面代码我们以列的方向遍历了所有的单元格。使用了二层循环，第一次循环获取所有的列，第二层循环获取每列中的每个单元格。

5.3 遍历单行的单元格

```
import openpyxl

wb = openpyxl.load_workbook("example.xlsx")
ws = wb['Sheet1']

for cell in ws[2]:
    print(cell.value)
```

上面代码获取了第二行的所有单元格。

5.4 遍历单列的单元格

```
import openpyxl

wb = openpyxl.load_workbook("example.xlsx")
ws = wb['Sheet1']

for cell in ws['B']:
    print(cell.value)
```

上述代码获取了第二列的所有单元格。

5.5 遍历多行的单元格

```
import openpyxl

wb = openpyxl.load_workbook("example.xlsx")
ws = wb['Sheet1']

for row in ws[1:3]:
    for cell in row:
        print(cell.value)
    print('--- END OF ROW ---')
```

上述代码获取了第一行到第三行的所有单元格。

5.6 遍历多列的单元格

```
import openpyxl

wb = openpyxl.load_workbook("example.xlsx")
ws = wb['Sheet1']

for col in ws['A':'C']:
    for cell in col:
        print(cell.value)
    print('--- END OF COL ---')
```

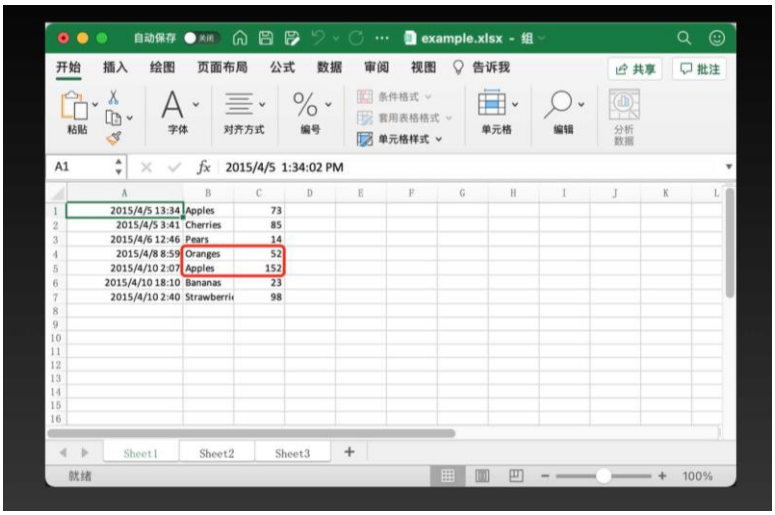
上述代码获取了第一列到第三列的所有单元格。

5.7 遍历一个区域内的单元格

5.7.1 行优先

```
import openpyxl

wb = openpyxl.load_workbook("example.xlsx")
ws = wb['Sheet1']

for row in ws.iter_rows(min_row=4,max_row=5,min_col=2,max_col=3):
    for cell in row:
        print(cell.value)
```

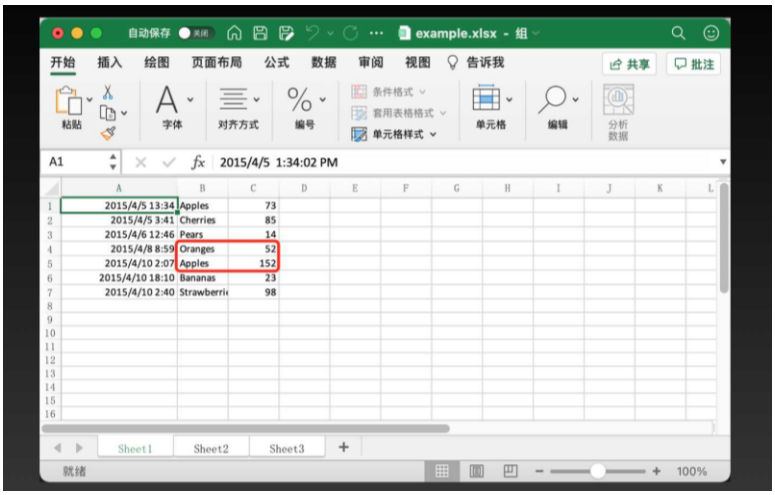
上面代码遍历的单元格如下图所示：



5.7.2 列优先

```
import openpyxl

wb = openpyxl.load_workbook("example.xlsx")
ws = wb['Sheet1']

for col in ws.iter_cols(min_row=4,max_row=5,min_col=2,max_col=3):
    for cell in col:
        print(cell.value)
```

上面代码遍历的单元格如下图所示：

	 小结与布置任务（20 分钟）：总结本次课重难点，点评学生实操情况，表扬数据读取准确、编码规范、有异常处理的学生；布置课后任务（编写脚本，读取指定 Excel 数据并提取关键信息），要求学生注重数据准确性与编码规范性，同时思考“如何在数据处理中体现责任意识与数据安全意识”。
教学后记	1. 课堂亮点：多数学生能掌握 Excel 数据读取与单元格遍历的方法，顺利完成指定数据提取；部分学生能主动添加异常处理语句，体现出较强的责任意识；学生的编码规范度有所提升，数据安全意识初步建立；遇到问题时，能主动思考、相互交流，解决问题的能力有所增强。 2. 存在不足：少数学生对单元格遍历的逻辑理解不透彻，遍历效率较低；部分学生数据读取时出现错误，缺乏认真检查的习惯；个别学生未添加异常处理语句，责任意识有待加强；少数学生对数据安全的重视程度不足，随意传播测试数据。 3. 改进措施：下次课课前，简要回顾单元格遍历的逻辑与效率优化方法，针对常见错误进行集中讲解；实操过程中，引导学生养成“数据读取后主动检查”的习惯，强化责任意识；加强数据安全教育，结合敏感数据泄露的警示案例，让学生重视数据安全；对未添加异常处理的学生，进行个别指导，明确异常处理的重要性。

第 7 周 教案（2 学时）

教学课题	Python-Excel：Excel 写入、批量编辑、行/列插入删除

教学学时	2 学时
教学目标	• 知识目标：掌握 openpyxl 库写入 Excel 数据的方法，理解批量编辑的逻辑，掌握行/列插入与删除的操作技巧； • 能力目标：能向 Excel 写入数据，实现批量编辑与行/列操作，确保数据写入的准确性； • 素养目标（思政）：培养严谨细致的编码习惯，注重数据写入的准确性与规范性，渗透“精益求精、责任担当”的职业理念，培养高效办公、主动优化的思维。
教学重难点	重点：Excel 数据写入方法，批量编辑逻辑，行/列插入与删除操作； 难点：批量编辑的效率优化，行/列插入删除后的数据对齐。

教学过程（融入思政）	导入（5 分钟）：结合职场批量数据处理案例（如批量录入员工信息、批量修改成绩数据），说明“批量编辑能大幅提升办公效率，而数据写入的准确性是工作质量的核心”，强调“精益求精”的重要性，引出本次课核心内容，渗透“责任担当、高效办公”的思政理念。
	理论讲解（5 分钟）：简要说明 openpyxl 写入 Excel 数据的核心方法，批量编辑的逻辑（循环写入、批量修改），行/列插入与删除的操作原理；结合思政点，强调“数据写入的每一个步骤都要严谨，一旦出错可能导致整个表格失效，体现责任担当”，同时引导学生思考如何优化批量编辑效率，培养创新思维。
	实操演示：
	1 向单元格写入数据
	1.1 一个单元格的写入
	向工作表的一个单元格写入内容的代码如下：
	`import openpyxl`
	`wb = openpyxl.Workbook()`
	`ws = wb['Sheet']`
	`ws['A1'].value = 'Hello World!'`
	`wb.save('write_excel.xlsx')`
	上面代码中，首先导入第三方库 openpyxl。其次，获取工作簿中的工作表。最后，向单元格 A1 写入 Hello World! 并保存文档为 write_excel.xlsx。这样，我们便向文档 write_excel.xlsx 的 Sheet 工

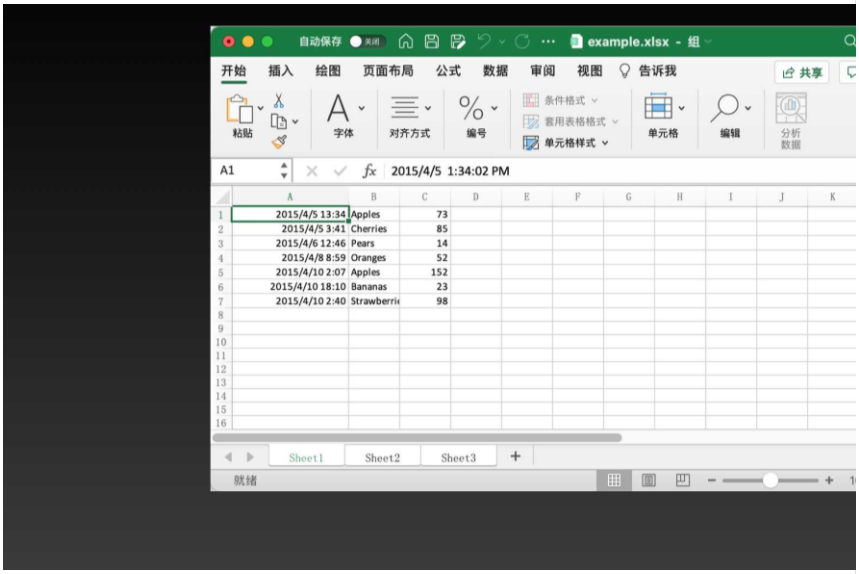
作表的 A1 单元格写入了 Hello World!。
1.2 多个单元格的写入

上面一次写入一个单元格，我们可以使用循环一次写入多个单元格。代码如下：

```
import openpyxl

wb = openpyxl.load_workbook("write_excel.xlsx")
ws = wb['Sheet']

for i in range(1, 10):
   ws.cell(row=2, column=i).value = i

wb.save('write_excel.xlsx')
```

上面的代码向工作表第二行的前九个单元格分别写入了 1，2，3，4，5，6，7，8，9。
2 插入、删除行和列

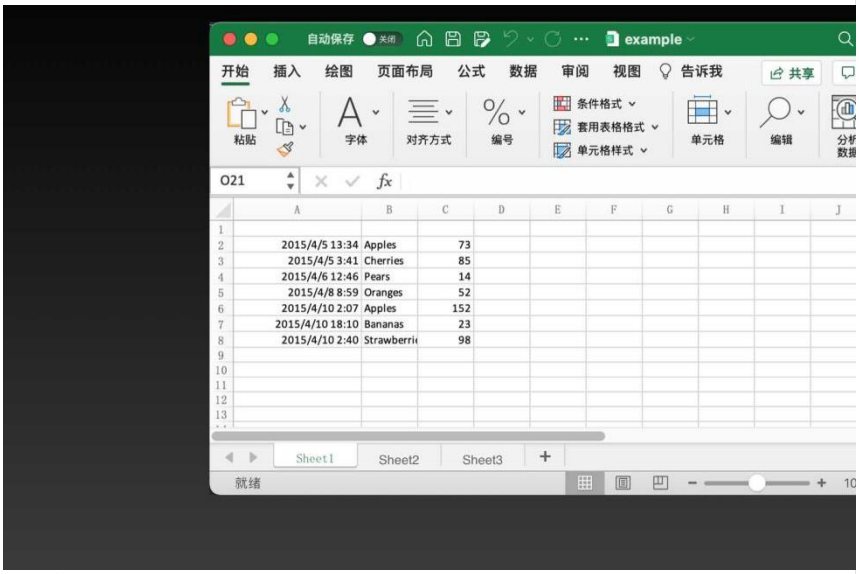
文档 example.xlsx 的内容如下图所示：



2.1 在指定位置插入一行

```
import openpyxl

wb = openpyxl.load_workbook("example.xlsx")
ws = wb.active

ws.insert_rows(idx=1)
wb.save('example.xlsx')
```

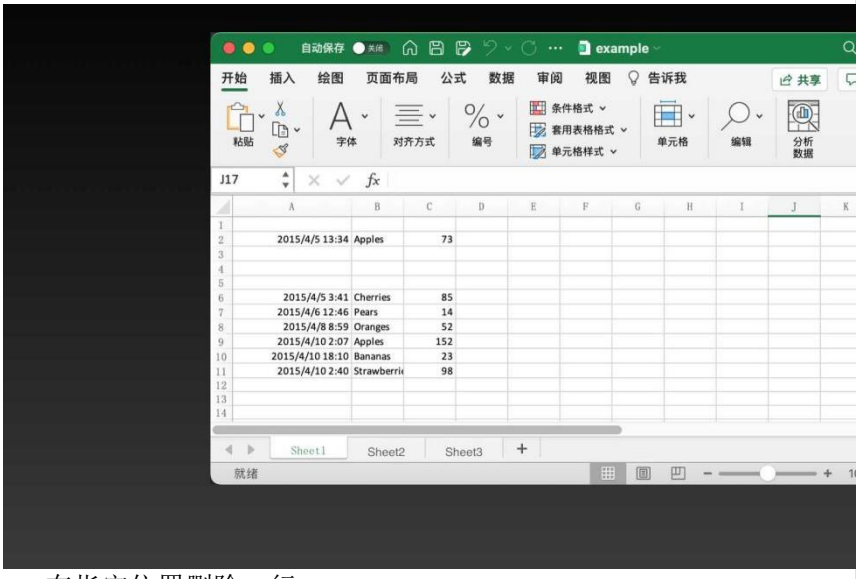
上面代码中，首先读取文档 example.xlsx，然后获取活跃的工作表。接下来，在第一行的位置插入一行后进行保存。插入之后的文档如下图所示：



2.2 在指定位置插入多行

```
import openpyxl

wb = openpyxl.load_workbook("example.xlsx")
ws = wb.active

ws.insert_rows(idx=3, amount=3)
wb.save('example.xlsx')
```

上面代码中，首先读取文档 example.xlsx，然后获取活跃的工作表。接下来，在第三行的位置插入三行后进行保存。插入之后的文档如下图所示：

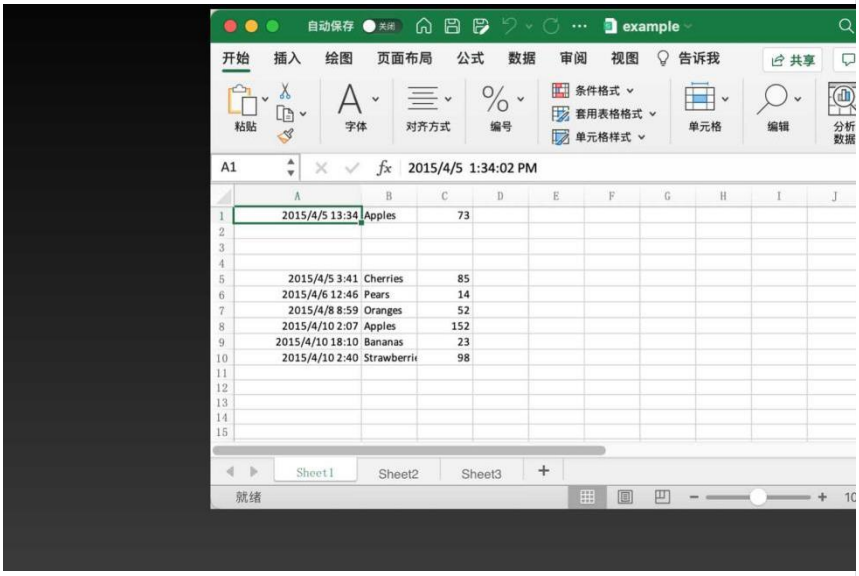


2.3 在指定位置删除一行

```
import openpyxl

wb = openpyxl.load_workbook("example.xlsx")
ws = wb.active
```

```
ws.delete_rows(idx=1)
wb.save('example.xlsx')
```

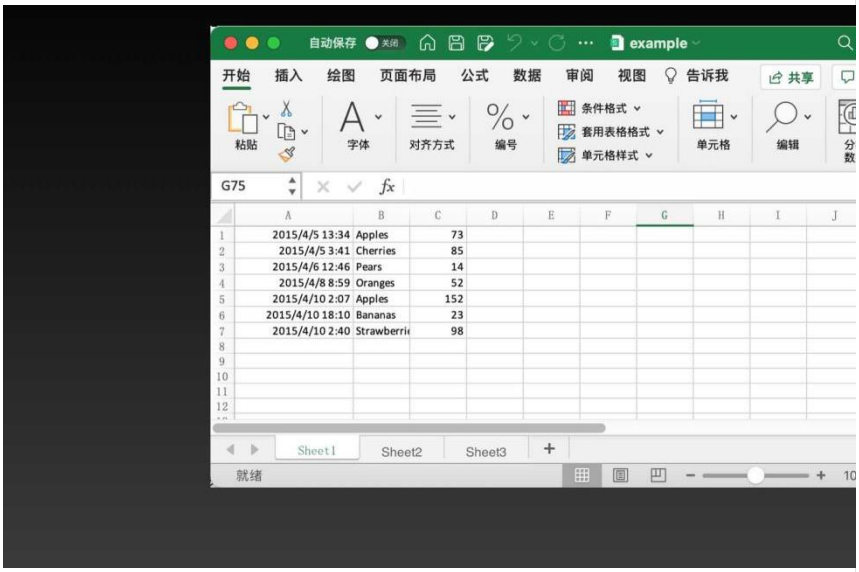
上面代码中，首先读取文档 example.xlsx，然后获取活跃的工作表。接下来，在第一行的位置删除一行后进行保存。删除之后的文档如下图所示：



2.4 在指定位置删除多行

```
import openpyxl

wb = openpyxl.load_workbook("example.xlsx")
ws = wb.active

ws.delete_rows(idx=2, amount=3)
wb.save('example.xlsx')
```

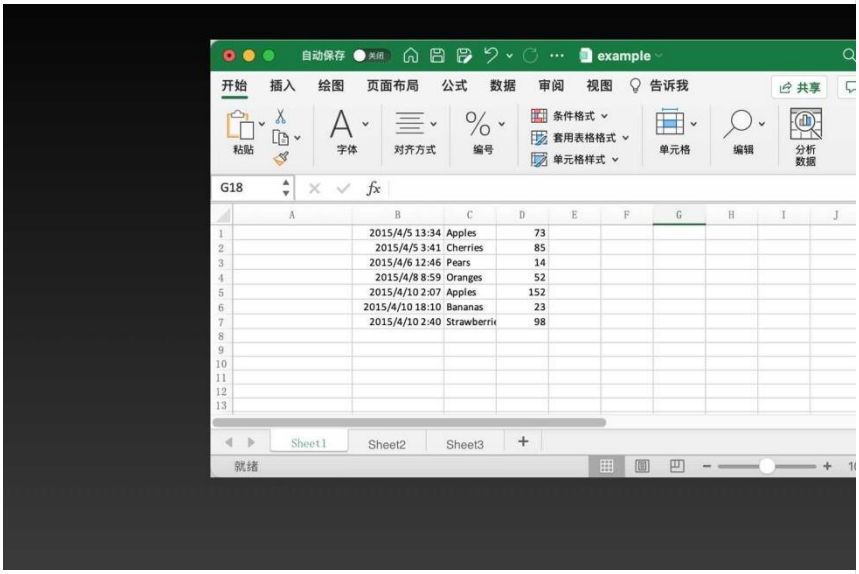
上面代码中，首先读取文档 example.xlsx，然后获取活跃的工作表。接下来，在第二行的位置删除三行后进行保存。删除之后的文档如下图所示：



2.5 在指定位置插入一列

```
import openpyxl

wb = openpyxl.load_workbook("example.xlsx")
ws = wb.active

ws.insert_cols(idx=1)
wb.save('example.xlsx')
```

上面代码中，首先读取文档 example.xlsx，然后获取活跃的工作表。接下来在第一列插入一列并进行保存。删除之后的文档如下图所示：

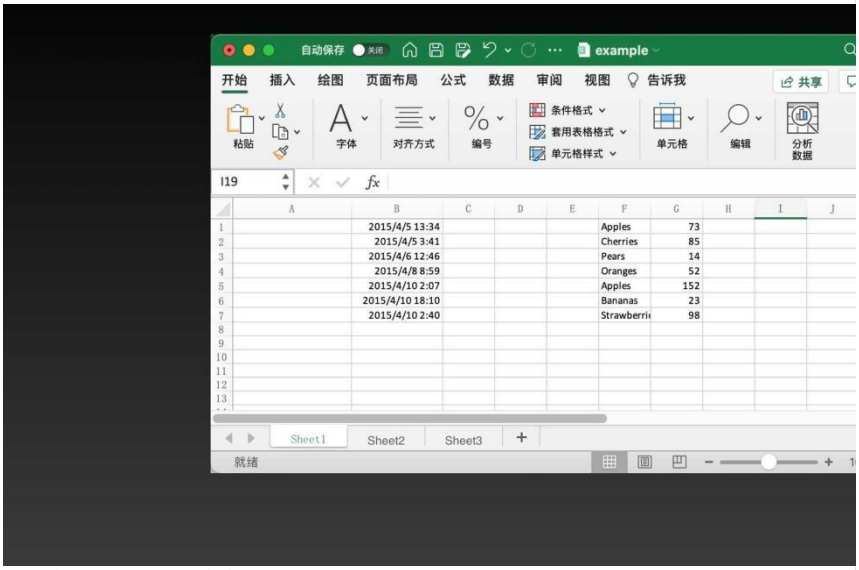


2.6 在指定位置插入多列

```
import openpyxl

wb = openpyxl.load_workbook("example.xlsx")
ws = wb.active
```

```
ws.insert_cols(idx=3, amount=3)
wb.save('example.xlsx')
```

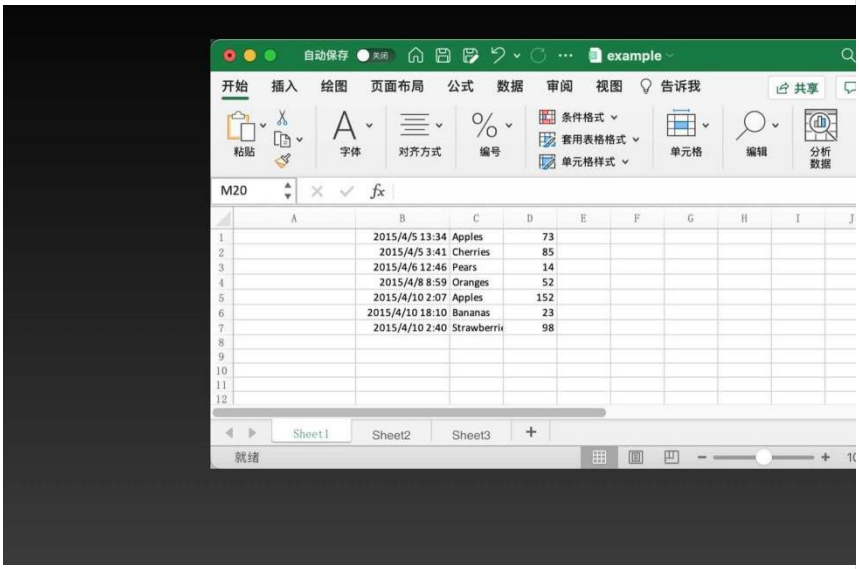
上面代码中，首先读取文档 example.xlsx，然后获取活跃的工作表。接下来，在第三列的位置插入三列后进行保存。插入之后的文档如下图所示：



2.7 在指定位置删除多列

```
import openpyxl

wb = openpyxl.load_workbook("example.xlsx")
ws = wb.active

ws.delete_cols(idx=3, amount=3)
wb.save('example.xlsx')
```

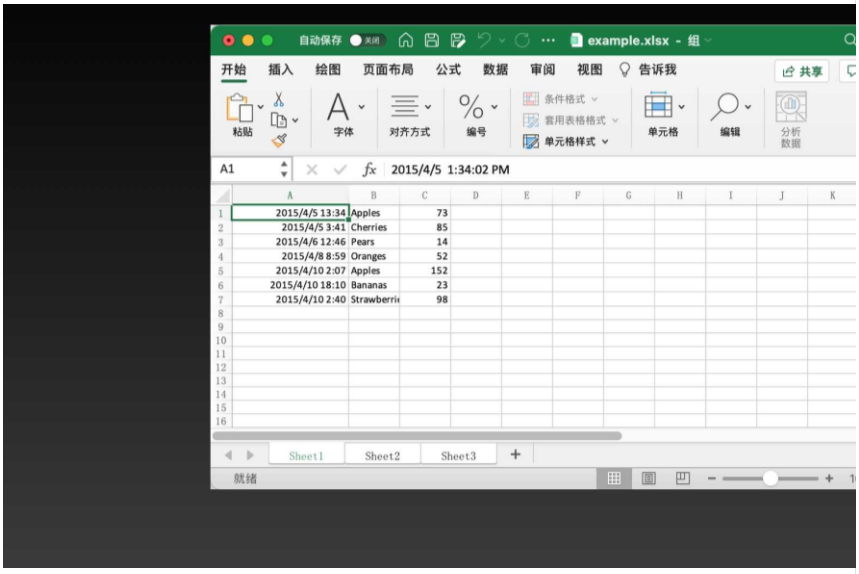
上面代码中，首先读取文档 example.xlsx，然后获取活跃的工作表。接下来，在第三列的位置删除三列后进行保存。删除之后的文档如下图所示：



2.8 在指定位置删除一列

```
import openpyxl

wb = openpyxl.load_workbook("example.xlsx")
ws = wb.active

ws.delete_cols(idx=1)
wb.save('example.xlsx')
```

上面代码中，首先读取文档 example.xlsx，然后获取活跃的工作表。接下来，在第一列的位置删除一列后进行保存。删除之后的文档如下图所示：



小结与布置任务：总结本次课重难点，点评学生实操情况，表扬

	数据准确、编码规范、主动优化脚本效率的学生；布置课后任务（编写脚本，完成批量数据写入、行/列插入删除及数据对齐），要求学生注重数据准确性与编码规范性，添加清晰注释，同时思考“如何在批量数据处理中平衡效率与准确性，体现精益求精的职业态度”。

第 8 周 教案（2 学时）

教学课题	Python-Excel：单元格格式设置、公式与图表生成
教学学时	2 学时
教学目标	• 知识目标：掌握 openpyxl 库设置单元格格式（字体、对齐、边框）的方法，理解 Excel 公式的嵌入逻辑，掌握图表生成的核心操作； • 能力目标：能批量设置单元格格式，嵌入 Excel 公式进行数据计算，生成规范的 Excel 图表（柱状图、折线图）； • 素养目标（思政）：培养严谨细致的编码与数据处理习惯，注重数据呈现的规范性与美观度，渗透“精益求精、注重细节”的职业理念，培养数据可视化思维与责任担当意识。
教学重难点	重点：单元格格式批量设置，Excel 公式嵌入，基础图表（柱状图、折线图）生成； 难点：公式嵌入的语法规范与错误排查，图表样式优化与数据关联。
教学过程（融入思政）	导入（5 分钟）：展示格式规范与不规范的 Excel 表格、有无图表的数据对比案例，结合职场数据汇报场景，说明“规范的格式与清晰的图表能提升数据可读性，体现职业专业性与细节把控能力”，引出本次课核心内容，渗透“精益求精、注重细节”的工匠精神。 理论讲解（5 分钟）：简要说明单元格格式设置的核心参数、Excel 公式嵌入的语法规则，图表生成的基本流程；结合思政点，强调“单元格格式的规范设置，如同职场文档的排版，是细节素养的体现；公式计算的准确性，直接关系到数据汇报的可信度，是责任担当的体现”，引导学生重视细节与数据真实。

4. 实操演示（25 分钟）：

1.公式

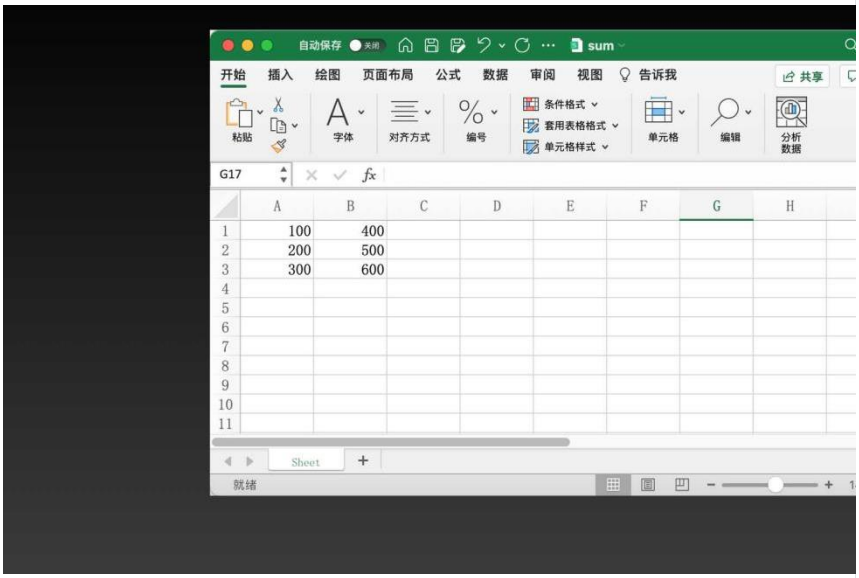
1.1 SUM

SUM 函数将值相加，可以将单个值、单元格引用或是区域相加，或者将三者的组合相加。例如: =SUM(A1:A3)将单元格 A1：A3 中的值加在一起，=SUM(A1:A3,B1:B3)将单元格 A1：A3 以及单元格 B1：B3 中的值加在一起。

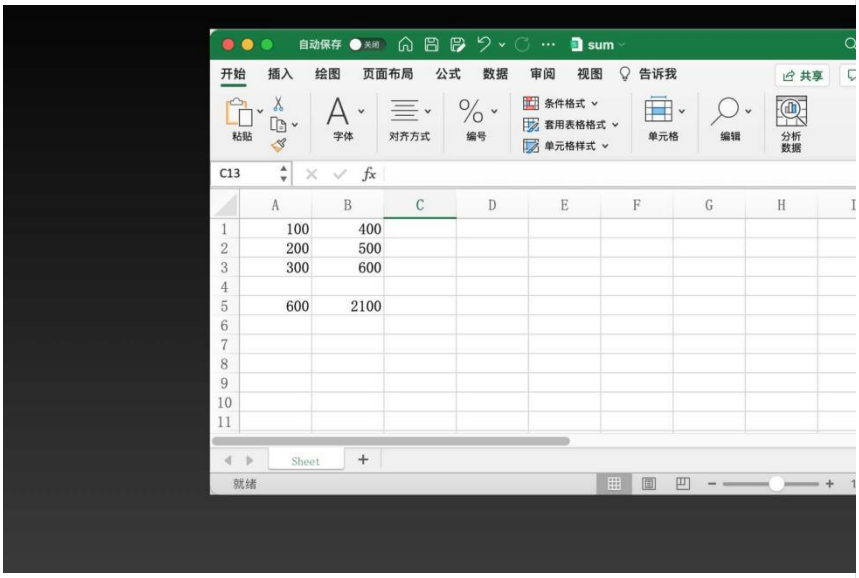
语法：SUM(number1,[number2],…)，number1（必需）：要相加的第一个数字。该数字可以是 4 之类的数字，A1 之类的单元格引用或 A1：A3 之类的单元格范围。number2（可选）：要相加的第二个数字。可以按照这种方式最多指定 255 个数字。下面我们来看怎么通过 Python 使用 SUM 函数。代码如下：

```
import openpyxl

wb = openpyxl.load_workbook('sum.xlsx')
ws = wb.active

ws['A5'] = '=SUM(A1:A3)'
ws['B5'] = '=SUM(A1:A3,B1:B3)'

wb.save('sum.xlsx')
```

上面代码中，首先读取 Excel 文档 sum.xlsx 并获取活跃工作表；其次在单元格 A5 以及 B5 中分别写入公式 =SUM(A1:A3) 和 =SUM(A1:A3,B1:B3)；最后对 Excel 文档进行保存。运行上述代码之前的文档如下图所示：



上述代码运行之后的文档如下图所示：



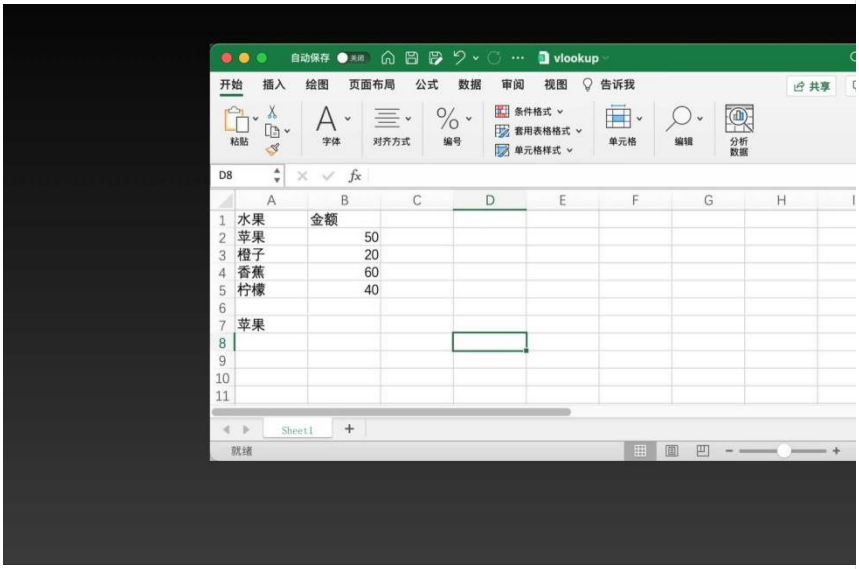
A5 单元格的值为 A1，A2，A3 单元格的值相加。B5 单元格的值为 A1，A2，A3，B1，B2，B3 单元格的值相加。

1.2 VLOOKUP

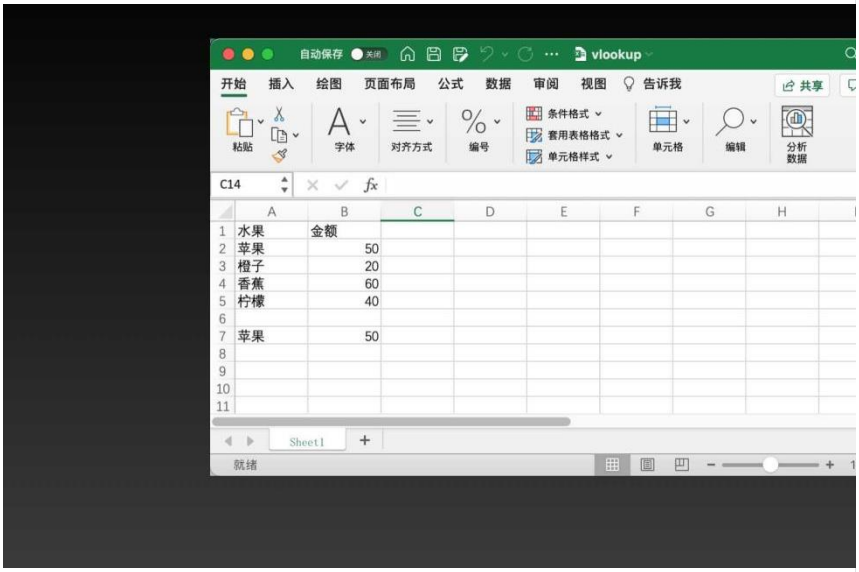
使用 VLOOKUP 可以在表格或区域中按行查找内容。下面我们来看怎么通过 Python 使用 VLOOKUP 函数。代码如下：

```
import openpyxl

wb = openpyxl.load_workbook('vlookup.xlsx')
ws = wb.active

ws['B7'] = '=VLOOKUP(A7,A1:B5,2,0)'
wb.save('vlookup.xlsx')
```

上面代码中，首先读取 Excel 文档 vlookup.xlsx 并获取活跃工作表；其次在单元格 B7 中写入公式 =VLOOKUP(A7,A1:B5,2,0)；最后对 Excel 文档进行保存。运行上述代码之前的文档如下图所示：



上述代码运行之后的文档如下图所示：



根据 A7 中的苹果查找到了金额 50。在公式 VLOOKUP(A7,A1:B5,2,0)中，第一个参数 A7 是要查找的值；第二个参数 A1:B5 为要在其中查找值的区域；第三个参数 2 为查找值所在列的列号；第四个参数 0 为查找匹配项，近似匹配指定 TRUE（1），精确匹配指定 FALSE（0）。

2.图表

我们可以使用 openpyxl 提供的方法为 Excel 中的数据作图表，下面以柱状图举例说明：

2.1 步骤

1.创建数据的 Reference 对象以及类别的 Reference 对象。
2.创建一个 BarChart 对象，并设置对象的属性，如：标题、x 轴名称、y 轴名称等。
3.将数据的 Reference 对象以及类别的 Reference 对象添加到 BarChart 对象。
4.将 BarChart 对象添加到工作表并保存工作表。

2.2 代码

```
import openpyxl
from openpyxl.chart import BarChart, Reference

wb = openpyxl.load_workbook('sampleChart.xlsx')
ws = wb.active

data = Reference(ws, min_row=1, max_row=9, min_col=3, max_col=4)
cats = Reference(ws, min_col=2, max_col=2, min_row=2, max_row=9)

chart = BarChart()
chart.title = "销售数量&销售金额"
chart.y_axis.title = '数量&金额'
chart.x_axis.title = '商品名称'
```

```
chart.add_data(data, titles_from_data=True)
chart.set_categories(cats)

ws.add_chart(chart,"F2")
wb.save("sampleChart.xlsx")
```

上面的代码中,步骤 1 对应的代码为：

```
data = Reference(ws, min_row=1, max_row=9, min_col=3, max_col=4)
cats = Reference(ws, min_col=2, max_col=2, min_row=2, max_row=9)
```

步骤 2 对应的代码为：

```
chart = BarChart()
chart.title = "销售数量&销售金额"
chart.y_axis.title = '数量&金额'
chart.x_axis.title = '商品名称'
```

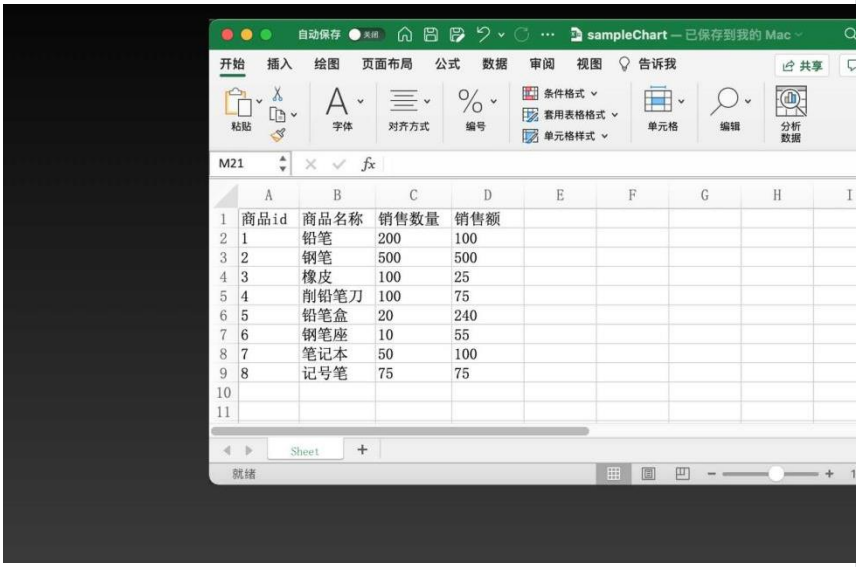
步骤 3 对应的代码为：

```
chart.add_data(data, titles_from_data=True)
chart.set_categories(cats)
```

步骤 4 对应的代码为：

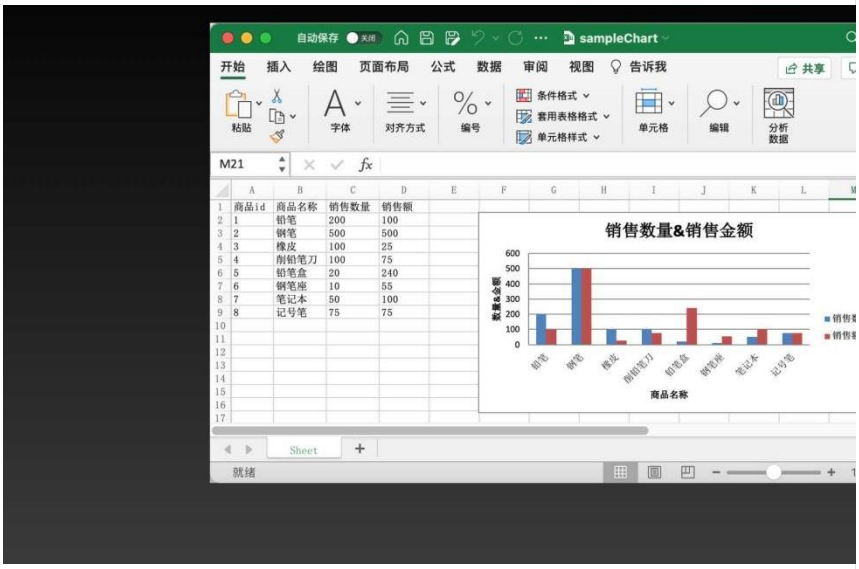
```
ws.add_chart(chart,"F2")
wb.save("sampleChart.xlsx")
```

运行上述代码之前的文档如下图所示：



	A	B	C	D
1	商品id	商品名称	销售数量	销售额
2	1	铅笔	200	100
3	2	钢笔	500	500
4	3	橡皮	100	25
5	4	削铅笔刀	100	75
6	5	铅笔盒	20	240
7	6	钢笔座	10	55
8	7	笔记本	50	100
9	8	记号笔	75	75
10				
11				

上述代码运行之后的文档如下图所示：



3.单元格格式的设置

我们还可以使用 openpyxl 提供的方法对 Excel 文档的单元格格式进行设置。

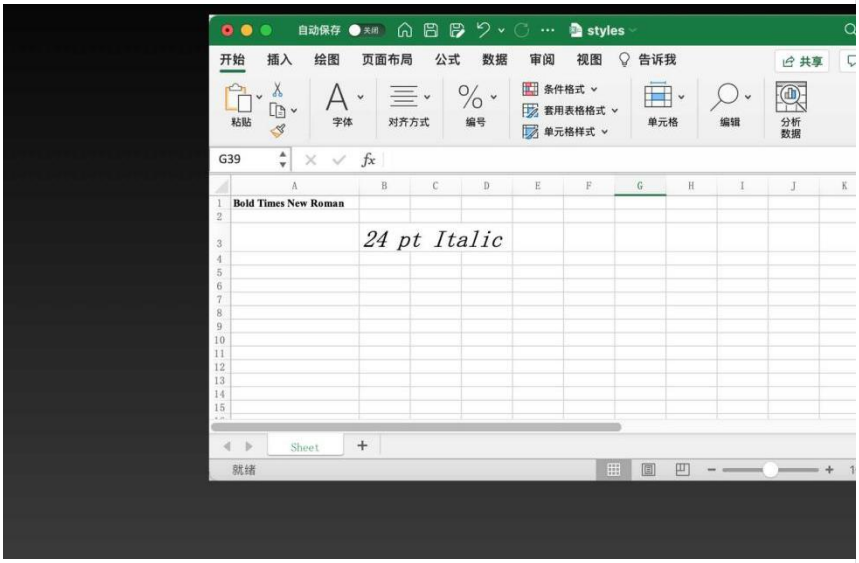
3.1 单元格字体的设置

示例代码如下：

```
from openpyxl.styles import Font

wb = openpyxl.Workbook()
ws = wb['Sheet']

fontObj1 = Font(name='Times New Roman', bold=True)
ws['A1'] = 'Bold Times New Roman'
ws['A1'].font = fontObj1

fontObj2 = Font(size=24, italic=True)
ws['B3'] = '24 pt Italic'
ws['B3'].font = fontObj2

wb.save('styles.xlsx')
```

在上面的代码中，创建了两个字体对象 fontObj1 和 fontObj2。并把 A1 单元格的字体设置成 fontObj1，B3 单元格的字体设置成 fontObj2。

上面代码执行完成后的 Excel 文档如下图所示：

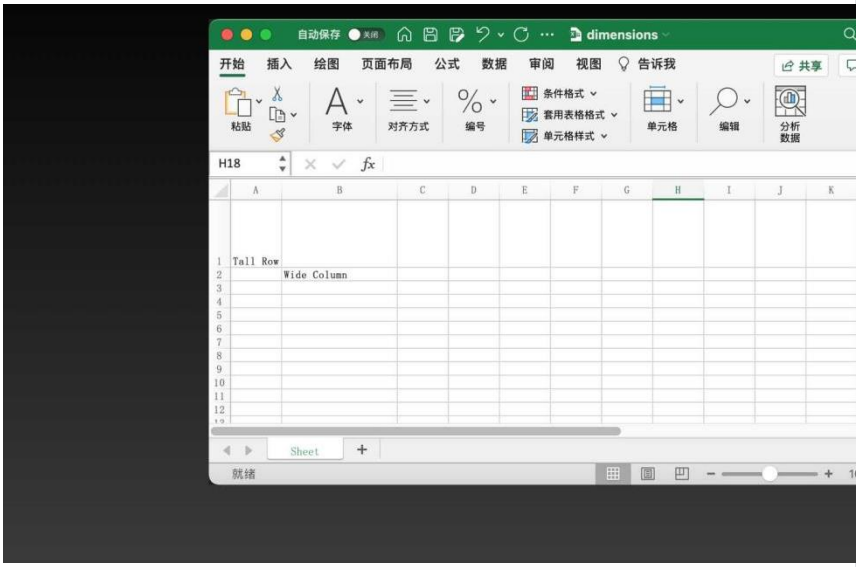


3.2 调整行高和列宽

示例代码如下：

```
import openpyxl

wb = openpyxl.Workbook()
ws = wb.active

ws['A1'] = 'Tall Row'
ws['B2'] = 'Wide Column'

ws.row_dimensions[1].height = 70
ws.column_dimensions['B'].width = 20

wb.save('dimensions.xlsx')
```

在上面的代码中，将第一行的行高设置成 70，将第二列的列宽设置成 20。上面代码执行完成后的 Excel 文档如下图所示：

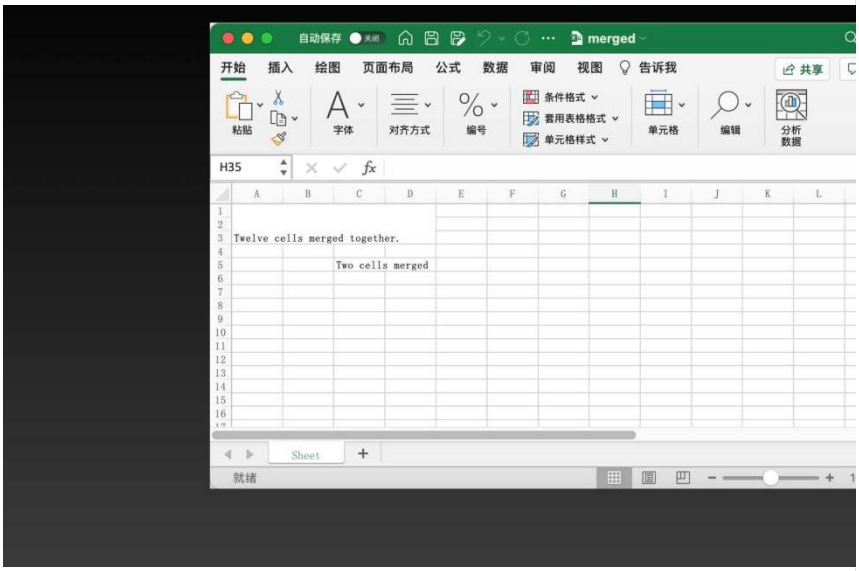


3.3 合并和拆分单元格

合并单元格的示例代码如下：

```
import openpyxl

wb = openpyxl.Workbook()
ws = wb.active

ws.merge_cells('A1:D3')
ws['A1'] = 'Twelve cells merged together.'

ws.merge_cells('C5:D5')
ws['C5'] = 'Two cells merged together.'

wb.save('merged.xlsx')
```

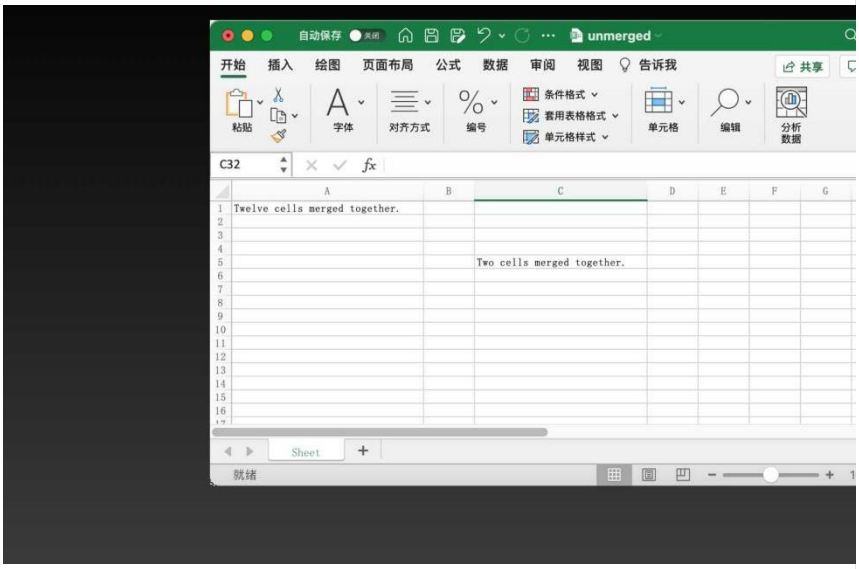
在上面的代码中，首先将 A1:D3 矩形区域内的单元格进行合并，其次将 C5 和 D5 单元格进行合并。上面代码执行完成后的 Excel 文档如下图所示：



拆分单元格的示例代码如下：

```
import openpyxl

wb = openpyxl.load_workbook('merged.xlsx')
ws = wb.active

ws.unmerge_cells('A1:D3')
ws.unmerge_cells('C5:D5')
wb.save('unmerged.xlsx')
```

在上面的代码中，将合并后的单元格进行拆分。上面代码执行完成后的 Excel 文档如下图所示：

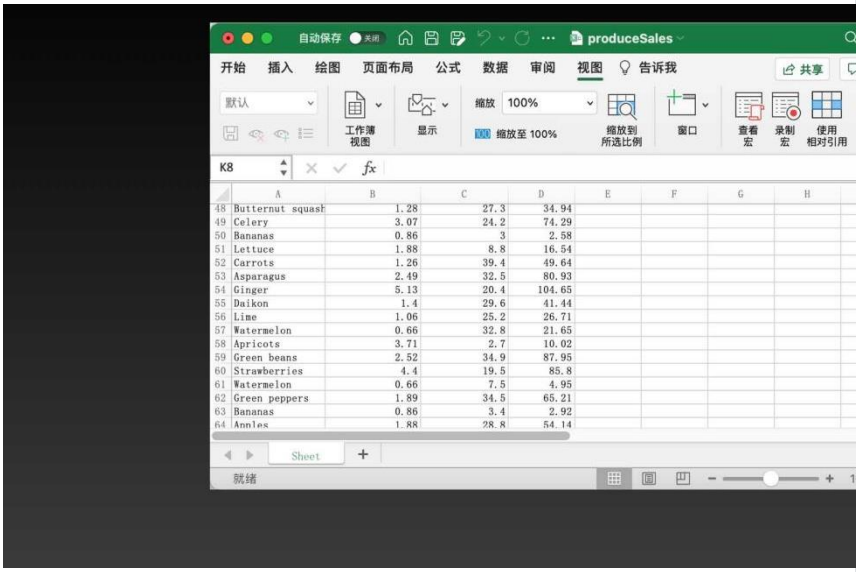


3.4 冻结单元格

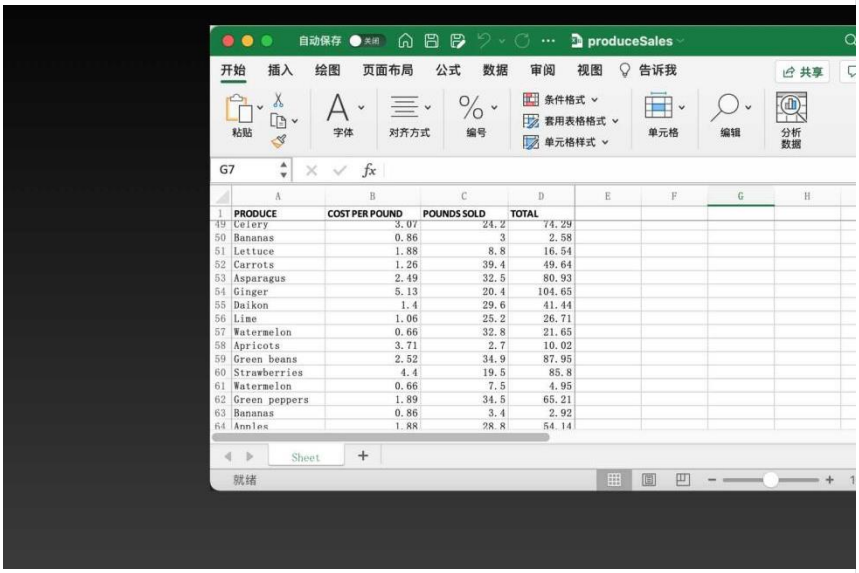
当 Excel 文档中的行数较多时，我们下滑鼠标查看行内容时，行的标题也会上滑消失，这时候想知道没列代表的含义就不是很方便。为了查看的方便，我们可以冻结标题。冻结单元格的示例代码如下：

```
import openpyxl

wb = openpyxl.load_workbook("produceSales.xlsx")
ws = wb.active
ws.freeze_panes = 'E2'

wb.save('produceSales.xlsx')
```

冻结前查看后边行内容的 Excel 文档如下图所示：



可以看到下滑到后面行的时候，标题看不到了。上面代码执行完成后的 Excel 文档如下图所示：

	 当冻结首行后，当下滑到后面行的时候，标题依然是可以看到的。 小结与布置任务（20 分钟）：总结本次课重难点，点评学生实操情况，表扬格式规范、公式准确、图表美观的学生；布置课后任务（编写脚本，完成单元格格式设置、公式计算与图表生成），要求学生注重细节与规范性，同时思考“如何通过规范的数据呈现，体现自身的职业素养与责任担当”。
教学后记	1. 课堂亮点：多数学生能熟练掌握单元格格式设置、公式嵌入与图表生成方法，实操完成度较高；部分学生能主动优化图表样式，注重数据呈现的美观度与规范性，体现出精益求精的态度；学生对公式错误的排查能力有所提升，能主动发现并解决简单的语法错误，责任意识进一步增强。 2. 存在不足：少数学生对 Excel 公式的语法规范掌握不熟练，出现公式嵌入错误且无法自主排查；部分学生批量设置单元格格式时，存在格式不统一的问题，细节把控不到位；个别学生生成图表后，未核对图表与数据源的关联性，导致图表数据错误；少数学生缺乏数据可视化思维，图表样式简单，未能体现数据的核心信息。 3. 改进措施：下次课课前，整理常见 Excel 公式的语法规范与错误排查方法，制作简易手册分享给学生；实操过程中，加强对格

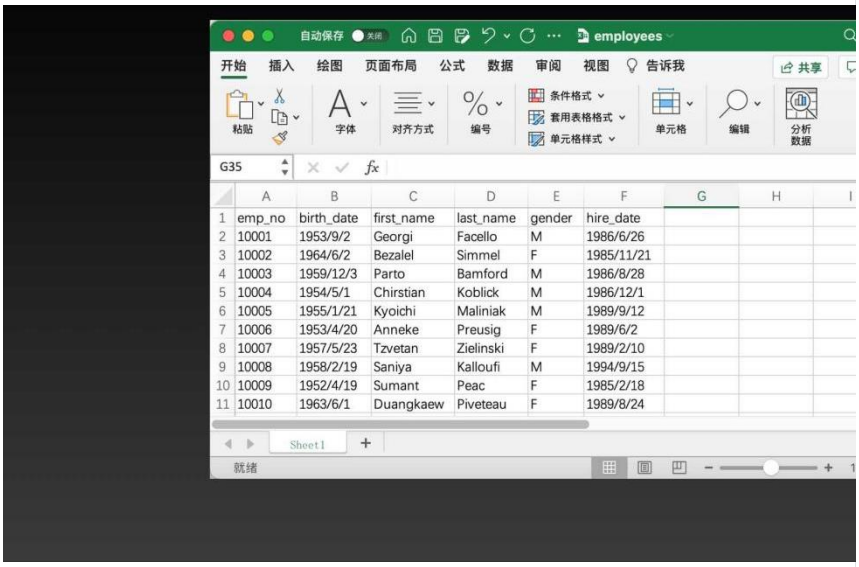
	式统一性与公式准确性的指导，引导学生养成“逐行核对”的习惯；结合职场数据汇报案例，讲解数据可视化的技巧，培养学生的可视化思维；对图表数据错误的学生，进行个别辅导，强化“数据关联”意识，明确数据呈现的责任。

第 9、10 周 教案（4 学时）

教学课题	Python-Excel：工作簿/工作表拆分与合并、批量处理综合实操
教学学时	4 学时
教学目标	• 知识目标：掌握工作簿、工作表拆分与合并的方法，理解 Excel 批量处理的综合逻辑； • 能力目标：能实现工作簿拆分（按工作表拆分）、工作表合并（多工作表合并为一个），完成 Excel 批量处理综合任务； • 素养目标（思政）：培养统筹规划的思维能力，注重批量处理的效率与准确性，渗透“高效办公、责任担当”的职业理念，培养团队协作与问题解决能力。
教学重难点	重点：工作簿拆分与合并的操作方法，Excel 批量处理综合任务的执行逻辑； 难点：多工作表合并时的数据去重与对齐，批量处理过程中的异常处理。
教学过程（融入思政）	导入（5 分钟）：结合职场批量数据处理案例（如多部门数据合并、员工信息按部门拆分），说明“工作簿/工作表的拆分与合并能大幅提升办公效率，统筹规划是高效完成任务的关键”，引出本次课核心内容，渗透“统筹思维、高效办公”的职业理念。 理论讲解（5 分钟）：简要说明工作簿拆分与合并的核心逻辑，多工作表合并时数据去重与对齐的方法，批量处理过程中的常见异常及处理思路；结合思政点，强调“批量处理不仅要追求效率，更要注重数据的准确性与完整性，体现责任担当；统筹规划能让任务更有序，如同职场工作中的分工协作，能提升整体效率”，引导学生培养统筹思维与协作意识。 实操演示（25 分钟）：

1.将一个工作表的内容拆分到多个工作簿中

要拆分的工作表的内容如下图所示：



	A	B	C	D	E	F
1	emp_no	birth_date	first_name	last_name	gender	hire_date
2	10001	1953/9/2	Georgi	Facello	M	1986/6/26
3	10002	1964/6/2	Bezalel	Simmel	F	1985/11/21
4	10003	1959/12/3	Parto	Bamford	M	1986/8/28
5	10004	1954/5/1	Chirstian	Koblick	M	1986/12/1
6	10005	1955/1/21	Kyoichi	Maliniak	M	1989/9/12
7	10006	1953/4/20	Anneke	Preusig	F	1989/6/2
8	10007	1957/5/23	Tzvetan	Zielinski	F	1989/2/10
9	10008	1958/2/19	Saniya	Kalloufi	M	1994/9/15
10	10009	1952/4/19	Sumant	Peac	F	1985/2/18
11	10010	1963/6/1	Duangkaew	Piveteau	F	1989/8/24

工作表总共有 11 行，其中包括标题 1 行，数据 10 行。现在我们要做的是将这 10 行数据拆分到 10 个工作表中，拆分后的工作表包含标题 1 行和数据 1 行。完成上述拆分的代码如下：

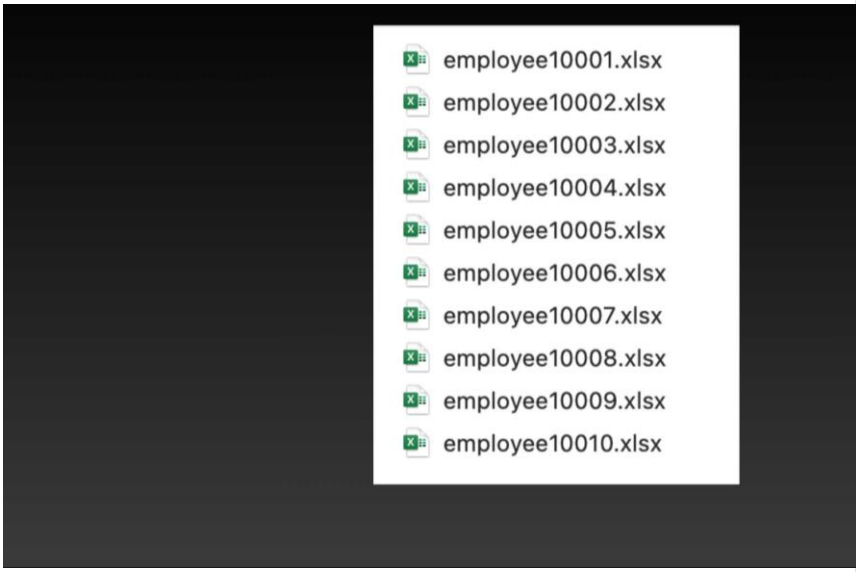
```
import openpyxl
import os


wb = openpyxl.load_workbook('employees.xlsx')
ws = wb.active

# 获取工作表的数据标题
title = []

for row in ws.iter_rows(min_row=1, max_row=1):
    title = [cell.value for cell in row]

i = 1
# 从第二行开始遍历工作表的每一行
for row in ws.iter_rows(min_row=2):
    wb_new = openpyxl.Workbook()
    ws_new = wb_new.active
    ws_new.append(title)

    # 获取每一行的内容并添加到工作簿
    values = [cell.value for cell in row]
    ws_new.append(values)
    ws_new.title = 'employee' + str(values[0])

    wb_new.save(os.getcwd() + '/employee/employee' + str(values[0]) +
'.xlsx')
```

```
    i += 1

wb_new.close()
wb.close()
```

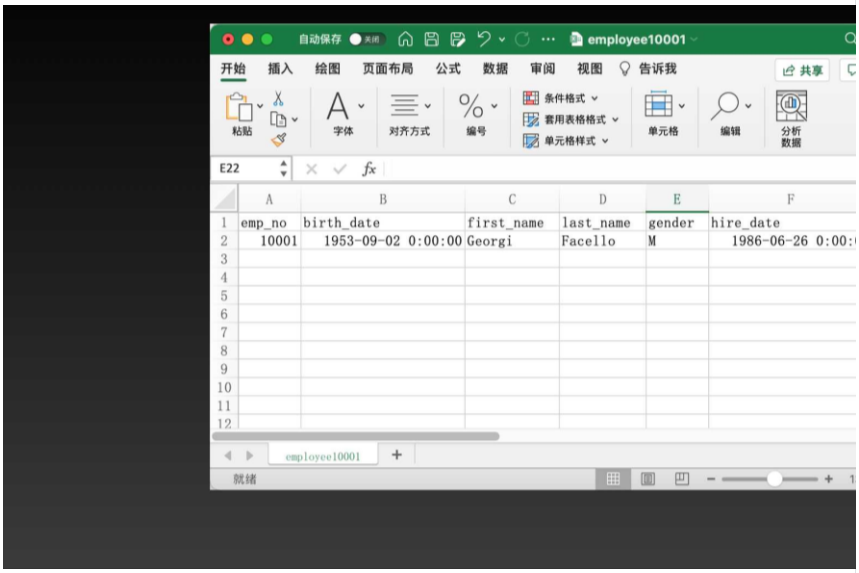
在上面的代码中：

1.打开文档并获取当前活跃的工作表。
2.获取工作表的数据标题。
3.从数据开始的第二行开始遍历，遍历时首先新建一个工作簿并获取当前活跃的工作表，将数据内容添加到新的工作表中并对新工作簿进行保存。

执行完上述代码之后，一个工作簿被拆分成了 10 工作簿，如下图所示：

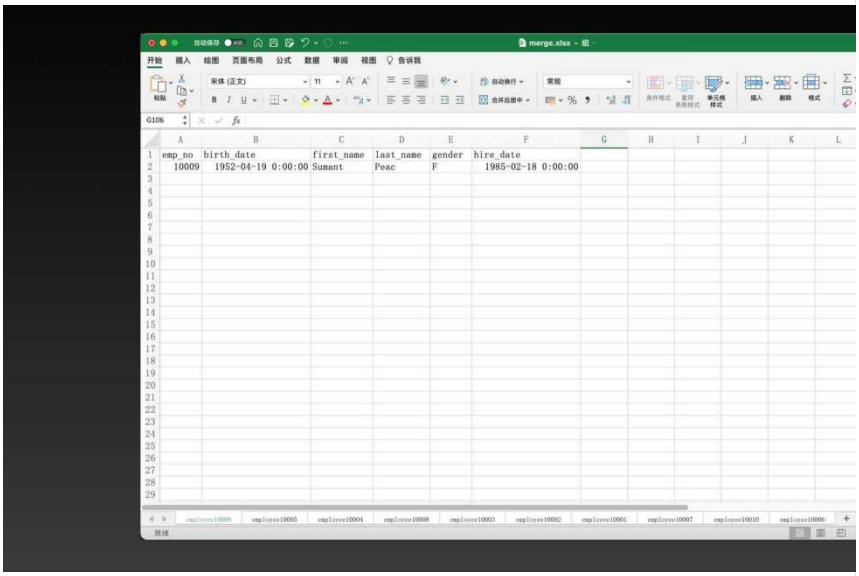


每个工作簿的内容（只是内容不同，形式是一样的）如下所示：



2.将一个工作簿的多个工作表拆分到多个工作簿中

上面讲的是将一个工作表的内容拆分到多个工作簿中，下面我们来讲将一个工作簿的多个工作表拆分到多个工作簿中。要拆分的工作簿的内容如下图所示：



上面工作簿中总共有 10 张工作表，每个工作表包含一行数据标题和一行数据。下面我们要做的是将这 10 张工作表拆分到 10 个工作簿中。完成上述拆分的代码如下：

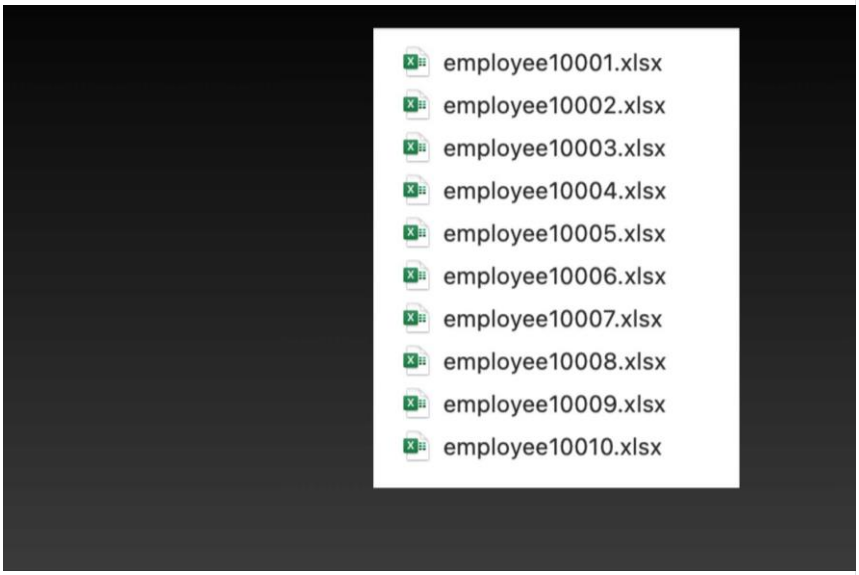
```
import openpyxl
import os


wb = openpyxl.load_workbook('merge.xlsx')

for ws in wb.worksheets:
    wb_new = openpyxl.Workbook()
    ws_new = wb_new.active
    for row_data in ws.iter_rows():
        for row_cell in row_data:
            ws_new[row_cell.coordinate].value = row_cell.value
    ws_new.title = ws.title

    wb_new.save(os.getcwd() + "/employee/" + ws.title + ".xlsx")

wb_new.close()
wb.close()
```

在上面的代码中：

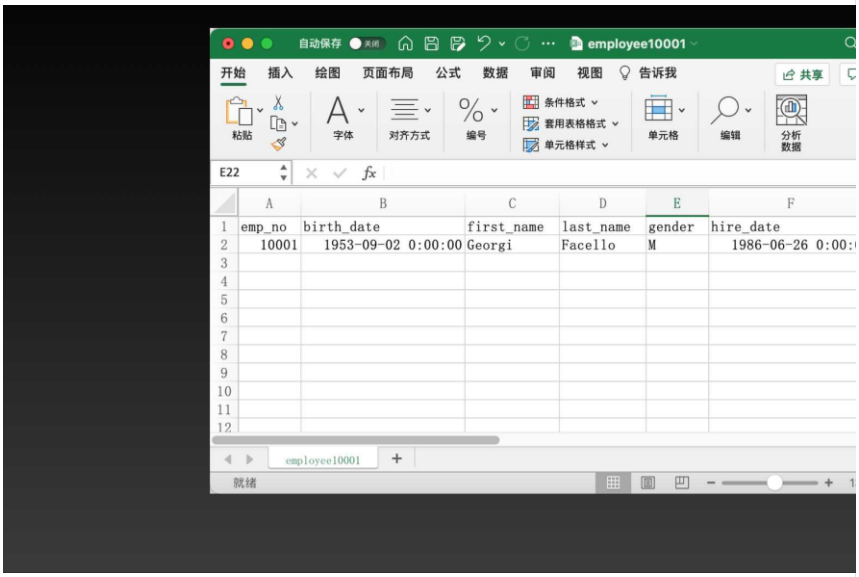
1.打开工作簿。
2.遍历工作簿中的每一个工作表，并把工作表的内容写到一个新的工作簿中。

执行完上述代码之后，工作簿中的工作表被拆分到了 10 工作簿，如

下图所示：

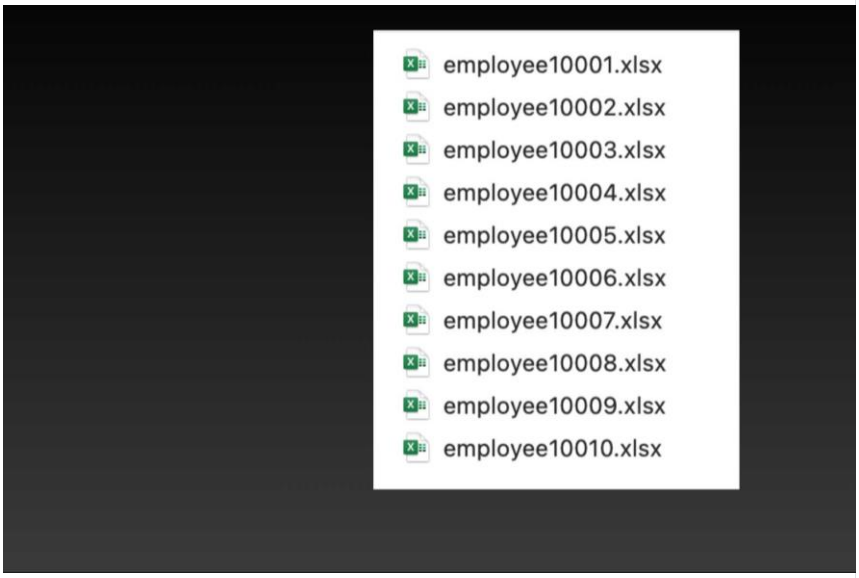


每个工作簿的内容（只是内容不同，形式是一样的）如下所示：

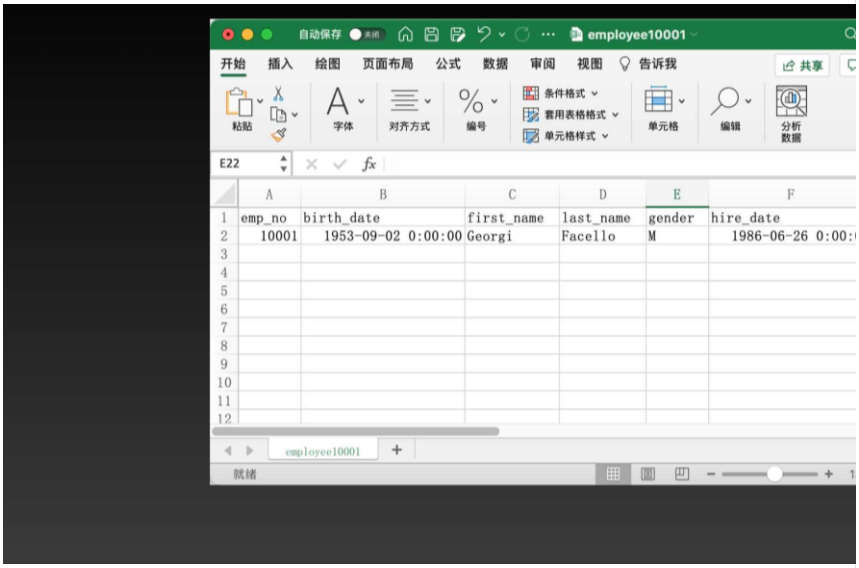


3.将多个工作簿内容合并到一个工作表中

上面两个例子讲的是工作簿的拆分，下面讲下工作簿的合并。要合并的工作簿如下图所示：



我们要将上面 10 个工作簿的内容合并到一个工作簿的工作表中。每个工作簿的内容（只是内容不同，形式是一样的）如下所示：



完成上述合并的代码如下：

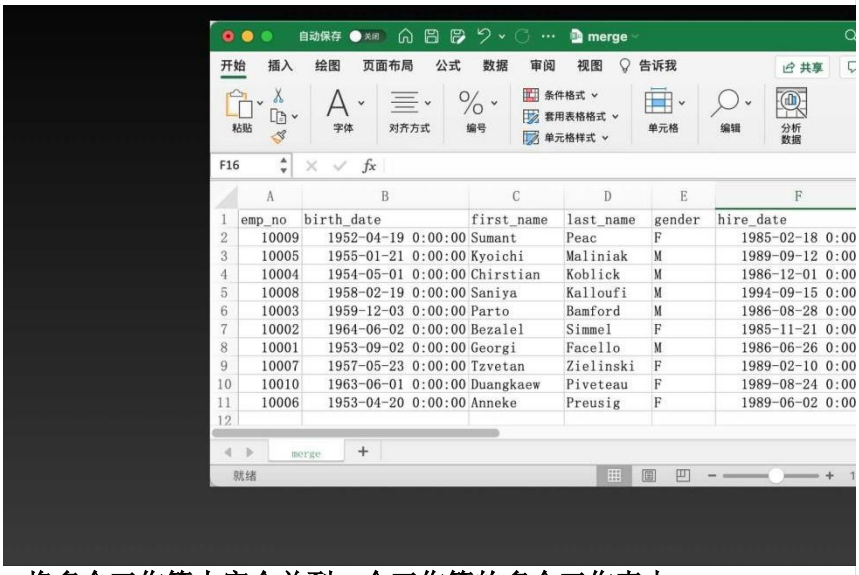
```
import os
import glob
import openpyxl


# 获取指定目录下所有的 xlsx 文件
xlsx_files = glob.glob(os.path.join(os.getcwd()+'/employee', '*.xlsx'))

# 创建一个新的工作簿
wb_new = openpyxl.Workbook()
ws_new = wb_new.active
ws_new.title = 'merge'
is_first_file = True
```

```
# 遍历所有的 Excel 文件
for filename in xlsx_files:
    print(filename)
    wb = openpyxl.load_workbook(filename)
    # 获取每个 Excel 文件中活跃的工作表
    ws = wb.active
    # 按行获取所有单元格
    if is_first_file:
        for row in ws.iter_rows(min_row=1):
            values = []
            for cell in row:
                values.append(cell.value)
            # values = [cell.value for cell in row]
            # 向表格末尾添加数据
            ws_new.append(values)
            is_first_file = False
    else:
        for row in ws.iter_rows(min_row=2):
            values = []
            for cell in row:
                values.append(cell.value)
            # values = [cell.value for cell in row]
            # 向表格末尾添加数据
            ws_new.append(values)

wb_new.save('merge.xlsx')
wb_new.close()
```

在上面的代码中：

1.获取指定目录下所有的 xlsx 文件，这里便是我们要合并的 10 个工作簿。
2.创建一个新的工作簿，用来容纳要合并的内容。
3.遍历所有的工作簿，获取工作簿的内容并添加到新的工作簿中并对新的工作簿进行保存。

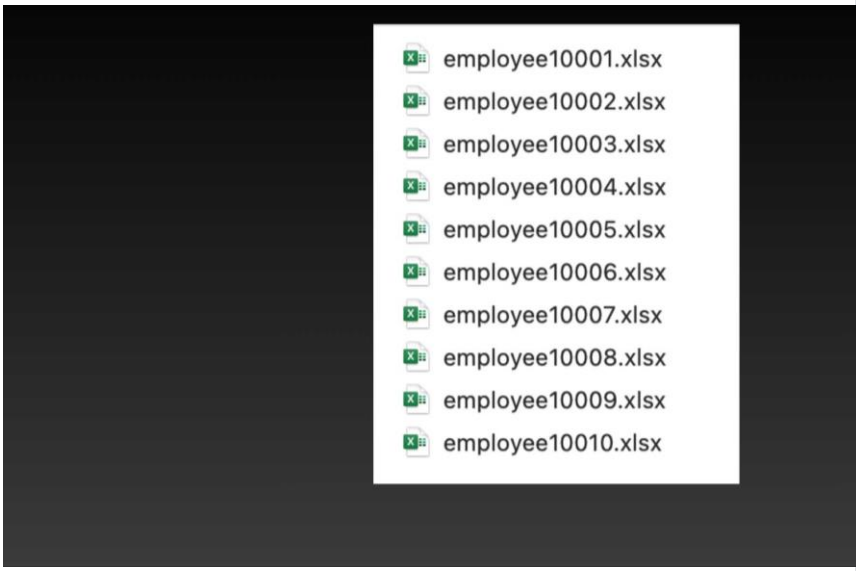
合并后的工作簿的内容如下图所示：



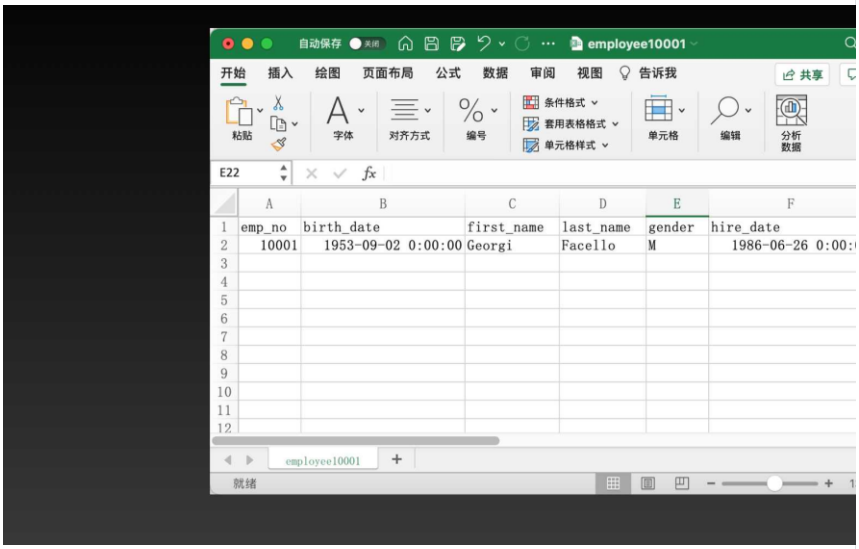
	A	B	C	D	E	F
1	emp_no	birth_date	first_name	last_name	gender	hire_date
2	10009	1952-04-19 0:00:00	Sumant	Peac	F	1985-02-18 0:00
3	10005	1955-01-21 0:00:00	Kyoichi	Maliniak	M	1989-09-12 0:00
4	10004	1954-05-01 0:00:00	Chirstian	Koblick	M	1986-12-01 0:00
5	10008	1958-02-19 0:00:00	Saniya	Kalloufi	M	1994-09-15 0:00
6	10003	1959-12-03 0:00:00	Parto	Bamford	M	1986-08-28 0:00
7	10002	1964-06-02 0:00:00	Bezalel	Simmel	F	1985-11-21 0:00
8	10001	1953-09-02 0:00:00	Georgi	Facello	M	1986-06-26 0:00
9	10007	1957-05-23 0:00:00	Tzvetan	Zielinski	F	1989-02-10 0:00
10	10010	1963-06-01 0:00:00	Duangkaew	Piveteau	F	1989-08-24 0:00
11	10006	1953-04-20 0:00:00	Anneke	Preusig	F	1989-06-02 0:00
12						

4.将多个工作簿内容合并到一个工作簿的多个工作表中

上面讲到的是将多个工作簿内容合并到一个工作表中，下面来讲下将多个工作簿内容合并到一个工作簿的多个工作表中。要合并的工作簿如下图所示：



我们要将上面 10 个工作簿的内容合并到一个工作簿的 10 个工作表中。每个工作簿的内容（只是内容不同，形式是一样的）如下所示：

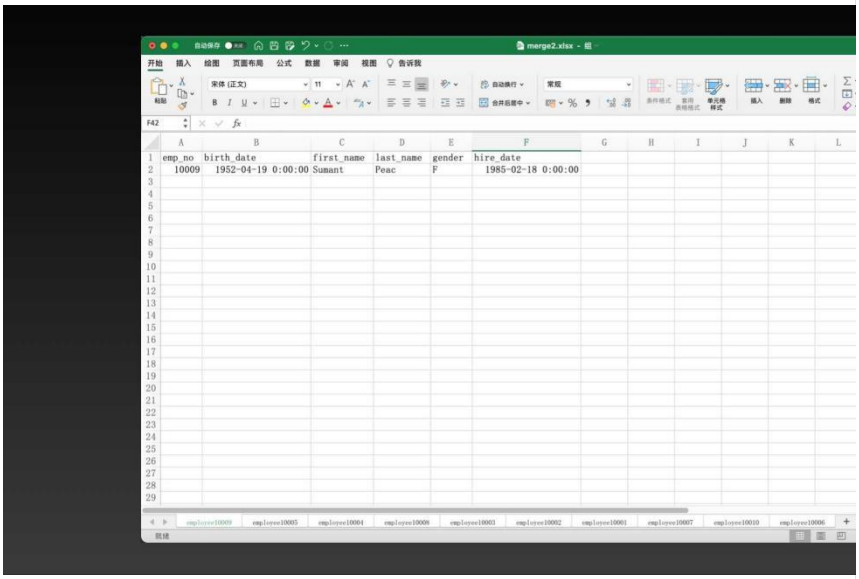


完成上述合并的代码如下：

```
import os
import glob
import openpyxl


# 创建一个新的工作簿
wb_new = openpyxl.Workbook()

xlsx_files = glob.glob(os.path.join(os.getcwd()+'/employee', '*.xlsx'))

# 遍历所有的 Excel 文件
for filename in xlsx_files:
    # 新建一个工作表
    ws_new = wb_new.create_sheet()
    wb = openpyxl.load_workbook(filename)
    # 获取每个 Excel 文件中活跃的工作表
    ws = wb.active
    # 按行获取所有单元格
    for row in ws.iter_rows(min_row=1):
        values = [cell.value for cell in row]
        # 向表格末尾添加数据
        ws_new.append(values)
        ws_new.title = ws.title
        print(ws.title)

wb_new.remove(wb_new['Sheet'])
wb_new.save('merge2.xlsx')
wb_new.close()
```

在上面的代码中：

1.获取指定目录下所有的 xlsx 文件，这里便是我们要合并的 10 个工作簿。
2.遍历所有的 xlsx 文件，每遍历一个文件，新建一个工作表，将文件

	的内容写入到新的工作表中。 3.保存合并后的工作簿。 合并后的工作簿的内容如下图所示：  学生综合实操（45 分钟）：学生根据综合任务要求，完成工作簿/工作表拆分与合并、批量数据处理，编写脚本并添加异常处理，教师巡回指导；重点关注学生的统筹思维与操作规范性，对拆分合并错误、数据去重不到位的学生，及时指导，引导学生反思“统筹规划对任务完成的重要性”；鼓励学生相互交流操作思路，培养团队协作能力；提醒学生操作完成后，全面检查数据的准确性与完整性，强化责任意识。 小结与布置任务（20 分钟）：总结工作簿/工作表拆分与合并的易错点，点评学生综合实操情况，表扬统筹规划能力强、操作规范、数据准确的学生；布置课后任务（完成 Excel 批量处理综合任务，含拆分、合并、数据编辑），要求学生注重统筹规划与数据准确性，同时思考“如何通过统筹规划提升办公效率，体现自身的职业素养”。
教学后记	1. 课堂亮点：多数学生能掌握工作簿/工作表拆分与合并的方法，顺利完成 Excel 批量处理综合任务，体现出较强的统筹规划能力；部分学生能主动添加异常处理语句，注重数据的准确性与完整性，责任意识突出；学生之间能相互交流操作思路，协作能力有所提升；对批量处理的逻辑理解更加深入，高效办公意识进一

	步增强。 2. 存在不足：少数学生缺乏统筹规划思维，操作步骤混乱，导致任务完成效率较低；部分学生多工作表合并时，未能正确进行数据去重与对齐，出现数据冗余或错位；个别学生批量处理过程中，未添加异常处理语句，遇到问题无法自主解决；少数学生编写脚本时，代码逻辑混乱，缺乏注释，不利于后续查看与修改。 3. 改进措施：下次课课前，通过案例讲解统筹规划的方法，引导学生先梳理任务逻辑再动手操作；实操过程中，加强对数据去重与对齐的指导，针对常见错误进行集中讲解；引导学生养成“添加异常处理、规范编写代码”的习惯，强化编码规范意识；结合职场案例，进一步深化“统筹思维、高效办公”的理念，提升学生的综合任务处理能力。

第 11 周 教案（2 学时）

教学课题	Python-Excel 自动化综合实操（报表批量生成与优化）
教学学时	2 学时
教学目标	• 知识目标：整合 Python-Excel 自动化的核心知识点，掌握报表批量生成与优化的完整流程； • 能力目标：能独立编写脚本，完成报表批量生成、格式设置、图表添加、数据汇总的全流程操作，优化脚本效率； • 素养目标（思政）：培养综合实操能力与统筹规划思维，注重报表的规范性与实用性，渗透“精益求精、责任担当”的职业理念，培养高效务实、主动优化的工作态度。
教学重难点	重点：报表批量生成的完整流程，脚本的优化方法； 难点：多步骤操作的统筹衔接，脚本效率优化与异常处理。
教学过程（融入思政）	2. 导入（5 分钟）：回顾前 4 周 Python-Excel 自动化的核心内容，展示职场报表批量生成案例，说明“综合实操能力是职场核心竞争力，统筹衔接与主动优化能大幅提升工作质量与效率”，明确本次课综合实操任务，渗透“精益求精、责任担当”的思政理念。

	3. 理论讲解（5 分钟）：简要梳理报表批量生成的核心流程（数据读取、写入、格式设置、图表添加、汇总分析），脚本优化的基本方法；结合思政点，强调“综合实操需要统筹规划，每一个步骤都要严谨，脚本优化体现创新思维与务实态度，是职业素养的重要体现”，引导学生树立“全流程把控”的工作思维。 4. 实操演示（20 分钟）：演示报表批量生成全流程（读取原始数据、批量创建工作表、写入数据、设置格式、添加图表、数据汇总），脚本优化（简化代码、批量赋值、异常处理）；演示过程中，强调“全流程的统筹衔接，避免步骤混乱，脚本注释要清晰，方便后续修改与维护”，渗透规范编码与统筹规划意识；针对常见的实操难题，演示排查与解决方法，培养学生的问题解决能力。 5. 学生综合实操（50 分钟）：学生独立编写脚本，完成报表批量生成与优化的全流程操作，教师巡回指导，解决综合实操难题；重点关注学生的流程衔接与脚本规范性，对步骤混乱、脚本冗余的学生，及时指导，引导学生优化流程与脚本；鼓励学生主动排查异常问题，培养独立解决问题的能力；对完成较快的学生，引导其进一步优化脚本效率，培养创新思维。 6. 小结与布置任务（20 分钟）：总结 Python-Excel 自动化的核心要点与综合实操注意事项，点评学生实操作品，表扬流程规范、脚本优化、态度认真的学生；布置课后任务（完善脚本，提交批量生成的报表与优化说明），要求学生注重报表规范性与脚本效率，同时反思“自己在综合实操中的不足，如何提升统筹规划与问题解决能力”。
教学后记	1. 课堂亮点：多数学生能独立完成报表批量生成与优化的全流程操作，体现出较强的综合实操能力；部分学生能主动优化脚本，简化代码、添加异常处理，体现出创新思维与务实态度；学生的统筹规划意识有所提升，能合理安排操作步骤，提高实操效率；遇到问题时，能独立思考、主动排查，解决问题的能力大幅增强。 2. 存在不足：少数学生流程衔接不顺畅，操作步骤混乱，导致报表生成失败；部分学生脚本冗余，未进行优化，效率较低；个别学生未添加异常处理，脚本稳定性不足；少数学生对报表的规范性重视不足，格式混乱、数据不准确。 3. 改进措施：下次课课前，简要回顾综合实操流程，针对常见的流程衔接问题进行集中讲解；实操过程中，引导学生先梳理操作

	流程，再编写脚本，强化统筹规划意识；加强对脚本优化与异常处理的指导，提升脚本稳定性与效率；对报表规范度不足的学生，进行个别辅导，强化规范意识；结合职场报表案例，进一步深化“精益求精、责任担当”的职业理念，提升学生的综合职业素养。

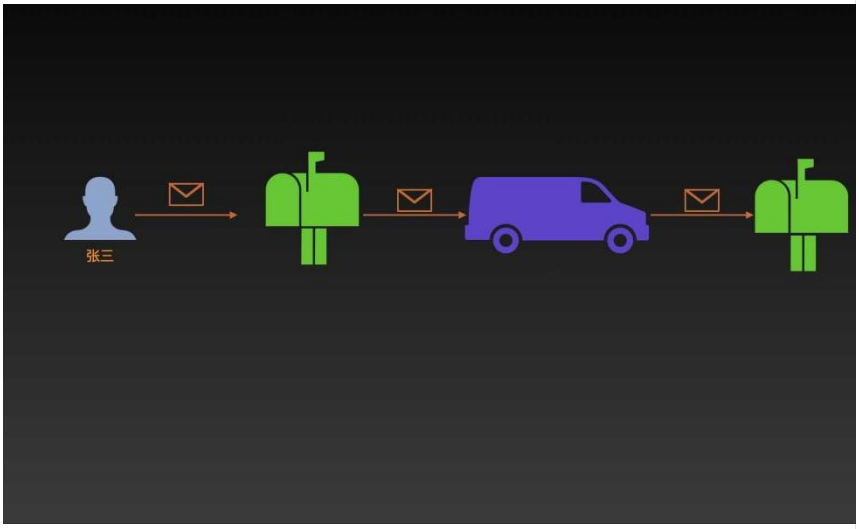
第 12 周 教案（2 学时）

教学课题	Python 批量邮件发送：SMTP 协议、纯文本/HTML 格式邮件
教学学时	2 学时
教学目标	• 知识目标：理解 SMTP 协议原理，掌握 Python smtplib、email 库的基础使用方法，了解纯文本与 HTML 格式邮件的区别； • 能力目标：能编写脚本，实现纯文本、HTML 格式邮件的单发与批量发送，配置 SMTP 服务器； • 素养目标（思政）：培养规范编码与信息传递意识，注重邮件内容的准确性与规范性，渗透“责任意识、高效办公”的职业理念，培养严谨细致、尊重他人的工作态度。
教学重难点	重点：SMTP 服务器配置，纯文本与 HTML 格式邮件的编写，邮件单发与批量发送逻辑； 难点：SMTP 服务器授权码获取，HTML 格式邮件的样式设置，批量发送的异常处理。
教学过程（融入思政）	导入（5 分钟）：结合职场邮件沟通案例（如批量通知、报表发送），说明“批量邮件发送能大幅提升办公效率，邮件内容的准确性与规范性直接影响沟通效果，体现责任意识与职业素养”，引出本次课核心内容，渗透“高效办公、责任意识”的职业理念。 理论讲解（5 分钟）：简要说明 SMTP 协议的作用与工作原理，smtplib、email 库的核心功能，纯文本与 HTML 格式邮件的适用场景；结合思政点，强调“邮件是职场重要的沟通工具，内容要准确、规范，避免因内容错误导致误解，体现对他人的尊重与自身的责任意识”，同时提醒学生规范使用邮件功能，不发送无关、违规内容。

实操演示（25 分钟）：

1.传统信件的发送流程

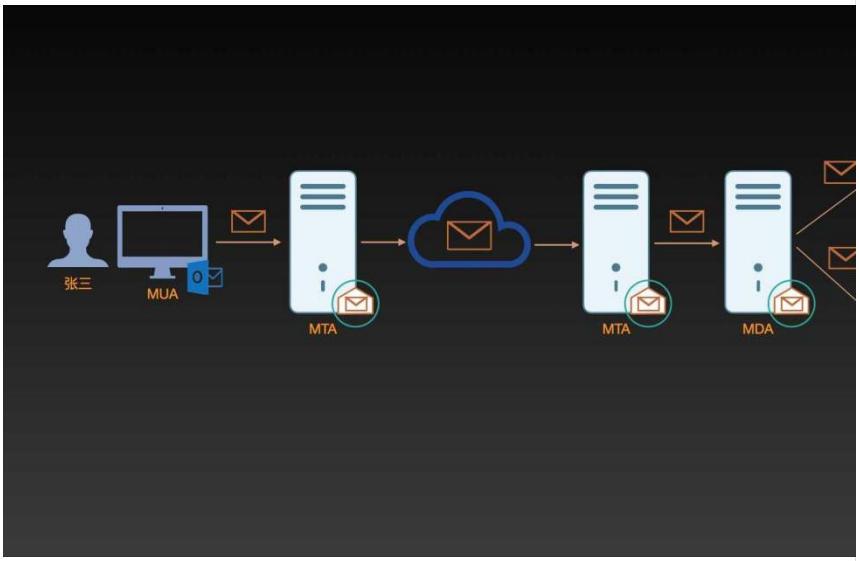
假如在安徽合肥的张三要给在山东济南的朋友李四寄一封信。如下图所示：



首先张三把信写好，装进信封，在信封上写上寄件人和收件人的地址，贴上邮票，并把信件投递到附近的邮局。信件便由小邮局到大邮局，并经过若干中间环节的运输，到达李四家附近的邮局，再经过邮递员的投递，最终到达李四的手中。

2.电子邮件的发送流程

上边便是传统信件的发送流程，电子邮件的流程基本上也是按上面的方式运作的。假如张三要将一封邮件同时发送给李四和王五。如下图所示：



假设张三的电子邮箱地址为 zhangsan@sina.com，李四和王五的电子邮箱地址分别为 lisi@163.com 和 wangwu@163.com。首先张三通过电脑上的 outlook 软件写好一封电子邮件，在收件人栏中填上李四和

王五的邮件地址，并点击发送按钮，这样一封邮件便发送出去了，这里的 outlook 称为 MUA（Mail User Agent）———— 邮件用户代理。电子邮件首先会发送到 MTA（Mail Transfer Agent）———— 邮件传输代理，由于张三的电子邮箱地址为 zhangsan@sina.com，所以会到达新浪的 MTA，邮件到达新浪的 MTA 之后再经过若干中间环节的传输，到达网易的 MTA（因为李四和王五的邮箱地址分别为 lisi@163.com 和 wangwu@163.com）。到达网易的 MTA 之后，再由网易的 MTA 发送到最终的目的地 MDA（Mail Delivery Agent）———— 邮件投递代理。然后，李四和王五通过 outlook 把邮件从 MDA 取到本地。这便是整个电子邮件的发送流程。

发送邮件时，MUA 和 MTA 使用的协议是 SMTP（Simple Mail Transfer Protocol），SMTP 是一种提供可靠且有效的电子邮件传输的协议。SMTP 是建立在 FTP 文件传输服务上的一种邮件服务，主要用于系统之间的邮件信息传递，并提供有关来信的通知。

收邮件时，MUA 和 MDA 使用的协议有两种：POP（Post Office Protocol）目前版本是 3，俗称 POP3；IMAP（Internet Message Access Protocol），目前版本是 4。

3.发送邮件

上面便是电子邮件的发送流程，下面我们来讲下如何用代码来实现电子邮件的发送。用代码来实现电子邮件的发送其实就是实现 MUA 的功能。

3.1 纯文本邮件的发送

纯文本邮件发送的代码如下：

```
import smtplib
from email.message import EmailMessage

msg = EmailMessage()

email_sender = '******@qq.com'
email_receiver = '******@gmail.com'


# 发件人
msg['From'] = email_sender
# 收件人
msg['to'] = email_receiver
# 邮件主题
msg['Subject'] = 'Hi there'
# 邮件内容
msg.set_content('This message is sent from Python.')

server = 'smtp.qq.com'
port = 465
passwd = '******'


with smtplib.SMTP_SSL(server, port) as server:
```

```
    server.login(email_sender, passwd)
    server.send_message(msg)
    server.quit()
```

在上面的代码中：

1.创建一个 `EmailMessage` 对象 msg，并设置发件人、收件人、邮件主题以及邮件内容。
2.连接 SMTP 服务器并发送邮件。
3.2 html 邮件内容的发送

html 邮件内容发送的代码如下：

```
import smtplib
from email.message import EmailMessage

msg = EmailMessage()

email_sender = '******@qq.com'
email_receiver = '******@gmail.com'

# 发件人
msg['From'] = email_sender
# 收件人
msg['to'] = email_receiver
# 邮件主题
msg['Subject'] = 'Hi there'
# 邮件内容
html = """\
<html>
  <body>
    <p>Hi,<br>
       How are you?<br>
       <a href="http://www.aidaxue.com">aidaxue</a>
       has many great tutorials.
    </p>
  </body>
</html>
"""

msg.add_alternative(html, subtype="html")

server = 'smtp.qq.com'
port = 465
passwd = '******'

with smtplib.SMTP_SSL(server, port) as server:
    server.login(email_sender, passwd)
    server.send_message(msg)
    server.quit()
```

上面的代码和纯文本邮件发送的不同点在于邮件的内容由纯文本变成了 html。还有就是往 msg 添加邮件内容的时候使用的是

add_alternative 方法。
3.3 带附件邮件的发送

带附件邮件的发送代码如下：

```
import smtplib
from email.message import EmailMessage
import imghdr

# 构造邮件内容。
msg = EmailMessage()

email_sender = '******@qq.com'
email_receiver = '******@gmail.com'

# 发件人
msg['From'] = email_sender
# 收件人
msg['to'] = email_receiver
# 邮件主题
msg['Subject'] = 'Hi there'
# 邮件内容
text = 'dinosaur'

msg.add_attachment(text)

with open('dinosaur.png', 'rb') as f:
   img = f.read()
   msg.add_attachment(img,                                    maintype='image',
subtype=imghdr.what(None, img))

server = 'smtp.qq.com'
port = 465
passwd = '******'

with smtplib.SMTP_SSL(server, port) as server:
   server.login(email_sender, passwd)
   server.send_message(msg)
   server.quit()
```

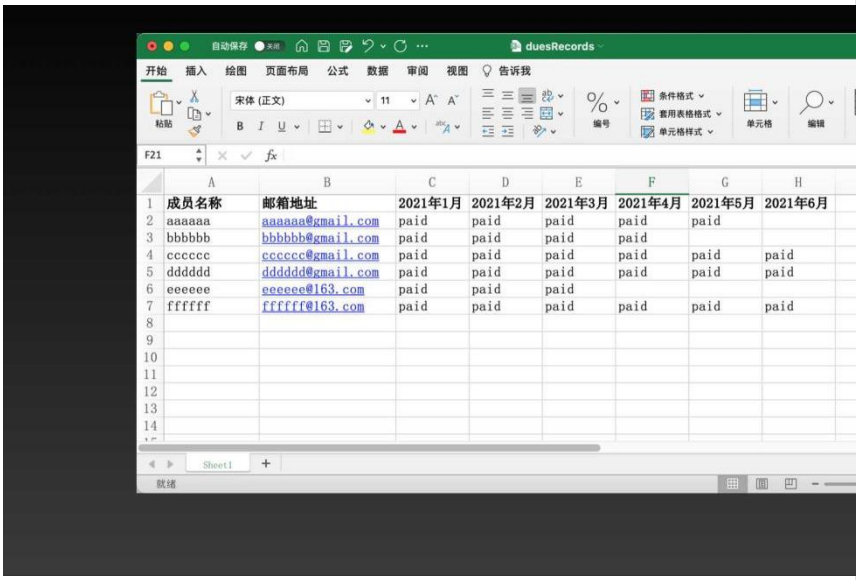
上面的代码和前面两种不同的是，邮件内容是文本和附件。添加附件的方法为 add_attachment。

学生实操（45 分钟）：学生配置 SMTP 服务器，编写脚本，完成纯文本、HTML 格式邮件的单发与批量发送，教师巡回指导；重点关注学生的 SMTP 配置与邮件内容准确性，对配置失败、邮件内容错误的学生，及时指导；提醒学生保管好授权码，树立数据安全意识；鼓励学生优化邮件样式，提升沟通效果；对遇到异

	常问题的学生，引导其独立排查、解决问题，增强实操自信心。 小结与布置任务（20 分钟）：总结本次课重难点，点评学生实操情况，表扬配置规范、邮件内容准确、操作熟练的学生；布置课后任务（编写脚本，批量发送 HTML 格式通知邮件），要求学生注重邮件内容规范性与数据安全，同时思考“批量邮件发送在职场中的应用场景，如何通过规范发送提升沟通效率”。
教学后记	1. 课堂亮点：学生对 Python 批量邮件发送的学习兴趣较高，多数学生能顺利配置 SMTP 服务器，完成纯文本与 HTML 格式邮件的发送；部分学生能主动优化邮件样式，提升沟通效果；学生的数据安全意识有所提升，能妥善保管授权码；遇到问题时，能主动思考、相互交流，解决问题的能力有所增强。 2. 存在不足：少数学生 SMTP 服务器配置失败，授权码获取方法掌握不熟练；部分学生 HTML 格式邮件样式设置混乱，可读性较差；个别学生批量发送时，未添加异常处理，导致发送失败；少数学生邮件内容不够规范，存在错别字、格式混乱等问题，责任意识有待加强。 3. 改进措施：下次课课前，整理 SMTP 配置与授权码获取的步骤手册，分享给学生；实操过程中，加强对 HTML 邮件样式设置与异常处理的指导，提升邮件规范性与脚本稳定性；引导学生养成“邮件发送前核对内容”的习惯，强化责任意识；结合职场邮件规范案例，深化“高效办公、尊重他人”的职业理念，提升学生的邮件沟通素养。

第 13 周 教案（2 学时）

教学课题	Python 批量邮件发送：带附件邮件、邮件发送异常处理
教学学时	2 学时
教学目标	• 知识目标：掌握 Python 发送带附件邮件的方法，理解邮件发送异常的常见类型与处理逻辑； • 能力目标：能编写脚本，实现带附件（Excel、Word）的批量邮件发送，添加异常处理语句，解决常见发送问题； • 素养目标（思政）：培养严谨的编码习惯与问题解决能力，注重邮件附件的准确性与安全性，渗透“责任意识、数据安全”的

	职业理念，培养高效务实、主动担当的工作态度。
教学重难点	重点：带附件邮件的编写与批量发送，邮件发送异常处理语句的添加； 难点：附件路径的正确设置，多种异常类型（文件不存在、发送失败）的处理逻辑。
教学过程（融入思政）	导入（5 分钟）：回顾上周纯文本/HTML 邮件发送内容，展示带附件邮件的职场案例（如批量发送报表、通知附件），说明“带附件邮件是职场常用的沟通方式，附件的准确性与安全性、异常处理能力，直接体现职业素养与责任意识”，引出本次课核心内容，渗透“责任意识、数据安全”的思政理念。 理论讲解（5 分钟）：简要说明带附件邮件的发送逻辑，附件路径的设置方法，邮件发送异常的常见类型（文件不存在、SMTP 连接失败、收件人地址错误）与处理逻辑；结合思政点，强调“附件是数据传递的重要载体，要确保附件准确、安全，避免发送错误或敏感附件，体现责任意识与数据安全意识”，同时引导学生树立“主动处理异常”的工作思维，提升问题解决能力。 实操演示（25 分钟）： **1.批量电子邮件的发送** 在例子中，我们有一个 Excel 文档，文档内容如下图所示：  在上面的文档中，第一列为成员名称，第二列为成员的邮箱地址，后面各列为每个成员在每个月的付款情况。我们现在要做的是针对每个成员最后一个月未付款的发邮件提醒付款。如果成员数较多，

手动发送这些邮件不但耗时而又容易出错。像这种每封邮件内容的格式固定的场景是自动化的用武之处。下面我们来看下自动化发送这些邮件的代码：

```
import openpyxl
import smtplib
import sys
from email.message import EmailMessage

wb = openpyxl.load_workbook('duesRecords.xlsx')
ws = wb['Sheet1']
lastCol = ws.max_column
latestMonth = ws.cell(row=1, column=lastCol).value

unpaidMembers = {}
for r in range(2, ws.max_row + 1):
    payment = ws.cell(row=r, column=lastCol).value
    if payment != 'paid':
        name = ws.cell(row=r, column=1).value
        email = ws.cell(row=r, column=2).value
        unpaidMembers[name] = email


msg = EmailMessage()

server = 'smtp.qq.com'
port = 465
email_sender = '******@qq.com'
passwd = '******'

with smtplib.SMTP_SSL(server, port) as server:
    server.login(email_sender, passwd)
    for name, email in unpaidMembers.items():
        body = "Dear %s,\nRecords show that you have not paid dues Ior %s. Please make this payment as soon as possible. Thank you!'" % (name, latestMonth)
        print('Sending email to %s...' % email)
        msg['From'] = email_sender
        msg['to'] = email
        msg['Subject'] = f'SubMect: {latestMonth} dues unpaid.'
        msg.set_content(body)
        status = server.send_message(msg)
        if status != {}:
            print('There was a problem sending email to %s: %s' % (email, status))
        msg.clear()
    print('Done!')
    server.quit()
```

在上面的代码中：

1.获取最后一个月未付款的用户的名称和邮箱地址。
2.循环遍历未付款的用户，为每个用户发邮件，提醒其进行付款。

	这便是自动化发送批量邮件的流程。在上面的代码中有一点要注意的是，在发送完一封邮件后进行邮件内容的清空。 学生实操（45 分钟）：学生编写脚本，完成带附件邮件的批量发送，添加异常处理语句，排查常见问题，教师巡回指导；重点关注学生的附件路径设置与异常处理能力，对路径错误、未添加异常处理的学生，及时指导；提醒学生检查附件的准确性与安全性，树立数据安全意识；鼓励学生尝试处理不同类型的异常，培养问题解决能力；对基础薄弱的学生，耐心讲解异常处理逻辑，帮助建立实操信心。 小结与布置任务（20 分钟）：总结本次课重难点，点评学生实操情况，表扬附件准确、异常处理完善、操作规范的学生；布置课后任务（编写脚本，批量发送带 Excel 附件的通知邮件，包含异常处理），要求学生注重附件安全与操作规范，同时思考“如何通过完善的异常处理提升脚本的稳定性，体现自身的责任意识”。
教学后记	1. 课堂亮点：多数学生能掌握带附件邮件的发送方法，顺利添加异常处理语句，解决常见的发送问题；部分学生能主动检查附件的准确性与安全性，体现出较强的责任意识与数据安全意识；学生的问题解决能力有所提升，能独立排查并处理简单的发送异常；编码规范度持续提升，多数学生能为代码添加清晰注释。 2. 存在不足：少数学生附件路径设置错误，导致附件无法正常发送；部分学生异常处理语句不完善，无法覆盖常见的异常类型；个别学生对异常处理逻辑理解不透彻，无法独立排查复杂异常；少数学生发送附件前未检查附件内容，存在附件错误的情况，责任意识有待加强。 3. 改进措施：下次课课前，简要回顾附件路径设置与异常处理的关键要点，结合案例进行集中讲解；实操过程中，引导学生养成“发送前检查附件”的习惯，强化责任意识；加强对异常处理逻辑的指导，帮助学生理解不同异常类型的处理方法；对基础薄弱的学生进行个别辅导，提升其问题解决能力；结合数据安全案例，深化“责任意识、数据安全”的职业理念，提升学生的职业素养。

第 14 周 教案（2 学时）

教学课题	Python 批量邮件发送综合实操（场景化应用）
教学学时	2 学时
教学目标	• 知识目标：整合 Python 批量邮件发送核心知识点，掌握场景化邮件发送的完整流程，理解不同职场场景下邮件发送的规范与要求； • 能力目标：能结合职场场景（批量通知、报表推送、文件分发），编写脚本完成带附件、多格式、多收件人的批量邮件发送，优化脚本稳定性与可读性； • 素养目标（思政）：培养场景化实操能力与职业适配性，注重邮件发送的规范性、安全性与高效性，渗透“责任意识、服务意识”的职业理念，培养严谨细致、灵活应变的工作态度。
教学重难点	重点：场景化批量邮件发送的完整流程，脚本的优化与适配，邮件发送的规范性； 难点：不同场景下邮件内容、附件的灵活适配，复杂异常的排查与处理，脚本的通用性优化。
教学过程（融入思政）	1. 导入（5 分钟）：回顾前 2 周批量邮件发送内容，展示 3 类职场场景案例（批量通知员工、推送月度报表、分发培训资料），说明“场景化实操是职场核心需求，邮件发送的规范性、适配性直接体现职业素养与服务意识”，明确本次课综合实操任务，渗透“责任意识、服务意识”的思政理念。 2. 理论讲解（5 分钟）：梳理场景化邮件发送的核心适配点（内容适配、附件适配、收件人适配），脚本优化的核心方向（通用性、稳定性、可读性）；结合思政点，强调“不同场景下，邮件内容要贴合需求、简洁规范，附件要准确安全，体现对收件人的尊重与自身的责任担当”，引导学生树立“场景适配、高效服务”的办公思维。 3. 实操演示（20 分钟）：选取“批量推送月度报表”场景，演示脚本编写全流程（读取收件人列表、编写 HTML 格式邮件正文、添加 Excel 附件、设置异常处理、优化脚本注释），演示不同场景下的脚本适配方法（修改邮件内容、更换附件类型）；演示过程中，强调“脚本的通用性设计，方便后续适配其他场景，代码注

	释清晰，便于团队协作维护”，渗透规范编码与团队协作意识；针对复杂异常（多个附件发送、部分收件人发送失败），演示排查与解决方法，培养学生的灵活应变能力。 4. 学生综合实操（50 分钟）：学生分组选取不同职场场景（通知、报表、资料分发），编写脚本完成批量邮件发送，优化脚本并排查异常，教师巡回指导；重点关注学生的场景适配性、脚本规范性与邮件安全性，对适配不当、脚本冗余的学生，及时指导；引导学生注重邮件内容的规范性，避免错别字与格式混乱，强化责任意识；鼓励学生相互交流脚本优化方法，培养团队协作能力；对完成较快的学生，引导其拓展脚本功能，提升通用性。 5. 小结与布置任务（20 分钟）：总结批量邮件发送的核心要点与场景化适配技巧，点评学生实操作品，表扬场景适配好、脚本规范、安全意识强的学生；布置课后任务（选取 2 个不同场景，编写通用批量邮件发送脚本，提交脚本与场景适配说明），要求学生注重脚本通用性与安全性，同时思考“如何通过场景化实操提升自身的职场适配能力”。
教学后记	1. 课堂亮点：多数学生能快速适配所选职场场景，完成批量邮件发送脚本编写，脚本规范性与稳定性有所提升；部分学生能主动优化脚本，实现通用性设计，体现出较强的灵活应变能力；学生的场景化思维与服务意识有所提升，能注重邮件内容与附件的适配性；团队协作氛围良好，学生能相互交流解决实操难题。 2. 存在不足：少数学生对不同场景的适配掌握不熟练，邮件内容与场景需求不符；部分学生脚本优化不足，通用性较差，无法快速适配其他场景；个别学生对复杂异常（多个附件发送失败）的排查能力不足，难以独立解决；少数学生邮件内容不够规范，缺乏场景适配意识，责任意识有待进一步强化。 3. 改进措施：下次课课前，展示不同场景的优秀脚本案例，集中讲解场景适配技巧；实操过程中，加强对脚本通用性优化与复杂异常排查的指导，提升学生的灵活应变能力；引导学生深入分析职场场景需求，强化场景适配与责任意识；对基础薄弱的学生进行个别辅导，帮助其掌握复杂异常的排查方法；结合职场场景案例，深化“责任意识、服务意识”的职业理念，提升学生的职场适配能力。

第 15 周 教案（2 学时）

教学课题	正则表达式基础：语法规则、字符匹配与量词使用
教学学时	2 学时
教学目标	• 知识目标：掌握正则表达式的基本语法规则，理解字符组、量词、转义字符的核心用法，了解正则表达式的应用场景； • 能力目标：能运用正则表达式完成简单的字符匹配、文本筛选，掌握 re 库的基础使用方法（match、search、findall）； • 素养目标（思政）：培养严谨的逻辑思维与细节把控能力，注重正则表达式的准确性与规范性，渗透“精益求精、务实高效”的职业理念，培养耐心细致、主动探索的学习态度。
教学重难点	重点：正则表达式的基本语法（字符组、量词、转义字符），re 库的基础方法使用； 难点：量词的正确使用（贪婪/懒惰匹配的初步认知），转义字符的应用场景，字符匹配的准确性。

教学过程（融入思政）	1. 导入（5 分钟）：展示办公场景案例（提取文本中的手机号、筛选无效数据、清洗杂乱文本），说明“正则表达式能高效解决文本批量处理问题，是办公自动化的重要工具，其准确性直接影响数据处理质量，体现严谨细致的职业素养”，引出本次课核心内容，渗透“精益求精、务实高效”的思政理念。
	2. 理论讲解（10 分钟）：讲解正则表达式的核心概念与应用场景，重点讲解字符组（匹配特定范围字符）、量词（匹配次数）、转义字符（匹配特殊字符）的基本语法；结合思政点，强调“正则表达式的语法严谨，每一个字符、每一个量词的使用都要准确，稍有疏忽就会导致匹配错误，体现细节把控的重要性”，引导学生树立严谨细致的学习与工作态度。
	实操演示（25 分钟）：
	
	1.不使用正则表达式
	
	不使用正则表达式的代码如下：
	
	def is_phone_number(text):
	  if len(text) != 12:
	  return False
	  for i in range(0, 3):
	  if not text[i].isdecimal():

```
        return False
    if text[3] != '-':
        return False
    for i in range(4, 7):
        if not text[i].isdecimal():
            return False
    if text[7] != '-':
        return False
    for i in range(8, 12):
        if not text[i].isdecimal():
            return False
    return True

message = 'My number is 415-555-4242.'

for i in range(len(message)):
    chunk = message[i:i+12]
    if is_phone_number(chunk):
        print('Phone number found: ' + chunk)
```

上面的代码中，首先定义一个 is_phone_number 函数，此函数描述了电话号码的组成规则。接着我们对要查找的字符串遍历，遍历时从前往后每次取 12 字符，调用 is_phone_number 函数对这 12 字符进行判断。一次判断结束后，往后推移一个字符，接着进行判断。直到余下的字符串中的字符个数不足 12 个。这种通过代码来查找符合某种组成规则的字符串的方法有很大的局限性，当字符串的组成规则稍微变动时，整个代码都需要推到重来。例如，如果要查找的电话号码的形式变为：(123)-456-7890，这时候我们便要重写代码。显然这种查找方式不是很灵活。下面我们来看下使用正则表达式如何来查找符合某种组成规则的字符串。

2.使用正则表达式

正则表达式出现的目的就是为了更加灵活方便地描述字符串的组成规则。

2.1 步骤

使用正则表达式查找字符串的步骤如下：

1.使用 import re 导入正则模块。
2.调用 rearch 方法来匹配字符串，第一个参数为规则，第二个参数为要匹配的字符串。
3.调用 group() 方法来获取匹配的字符串。

2.2 使用普通字符组

下面使用正则表达式来查找上面例子中的电话号码，代码如下：

```
import re

mo = re.search(r"[0123456789][0123456789][0123456789]-[0123456789][0123456789][0123456789]-[0123456789][0123456789][0123456789][0123456789]", "My number is 415-555-4242")
```

	print('Phone number found: ' + mo.group()) 在代码中，[0123456789] 为字符组，字符组就是在某个位置可能出现的所有字符的集合。[0123456789] 表示在这个位置上可能出现 0 ~ 9 这 10 个数字中的任何一个。 我们可以看到使用正则表达式可以更简洁地描述要查找字符串的组成规则。 5. 学生实操（40 分钟）：学生根据测试文本，运用正则表达式完成字符匹配、文本筛选任务，使用 re 库实现批量提取与筛选，教师巡回指导；重点关注学生的语法规范性与匹配准确性，对语法错误、匹配偏差的学生，及时指导；鼓励学生主动尝试不同的匹配规则，探索更高效的匹配方法，培养主动探索的学习态度；引导学生校验匹配结果，培养自我纠错能力，强化责任意识。 5. 小结与布置任务（20 分钟）：总结正则表达式的基本语法与 re 库基础方法，点评学生实操情况，表扬语法规范、匹配准确、主动探索的学生；布置课后任务（编写脚本，运用正则表达式提取文本中的邮箱地址与手机号，提交脚本与匹配结果），要求学生注重匹配准确性，同时思考“正则表达式在办公自动化中的更多应用场景”。
教学后记	1. 课堂亮点：多数学生能掌握正则表达式的基本语法，熟练使用 re 库基础方法完成字符匹配任务；部分学生能主动探索不同的匹配规则，优化匹配效率，体现出较强的主动探索意识；学生的细节把控能力有所提升，能主动校验匹配结果，避免匹配错误；对正则表达式的学习兴趣较高，能积极提问解决疑问。 2. 存在不足：少数学生对量词的使用掌握不熟练，无法准确控制匹配次数；部分学生对转义字符的应用场景理解不透彻，遗漏转义字符导致匹配失败；个别学生匹配规则设计不严谨，出现漏匹配、错匹配的情况；少数学生缺乏耐心，遇到匹配错误时不愿主动排查，依赖教师指导。 3. 改进措施：下次课课前，整理正则表达式基本语法口诀与常见量词、转义字符对照表，分享给学生；实操过程中，加强对量词与转义字符的指导，结合案例讲解不同场景下的使用方法；引导学生养成“耐心排查、主动校验”的习惯，培养独立解决问题的能

	力；对基础薄弱的学生进行个别辅导，帮助其理解语法规则；结合文本处理案例，深化“精益求精、务实高效”的职业理念，提升学生的细节把控能力。

第 16 周 教案（2 学时）

教学课题	正则表达式进阶：分组匹配、贪婪与懒惰匹配、文本替换
教学学时	2 学时
教学目标	• 知识目标：掌握正则表达式分组匹配、贪婪与懒惰匹配的核心用法，理解 re 库 sub 方法的使用逻辑，掌握文本批量替换技巧； • 能力目标：能运用分组匹配提取复杂文本信息，区分并使用贪婪/懒惰匹配，运用 sub 方法完成文本批量替换与数据清洗； • 素养目标（思政）：培养严谨的逻辑思维与问题解决能力，注重文本处理的准确性与高效性，渗透“精益求精、务实创新”的职业理念，培养耐心细致、主动优化的工作态度。
教学重难点	重点：分组匹配的使用方法，贪婪与懒惰匹配的区别，re 库 sub 方法实现文本批量替换； 难点：分组匹配的逻辑设计，贪婪与懒惰匹配的场景适配，复杂文本替换的规则设计。
教学过程（融入思政）	导入（5 分钟）：回顾上周正则表达式基础内容，展示复杂文本处理案例（提取姓名与联系方式、清洗杂乱格式文本、批量修改文本内容），说明“进阶正则表达式能解决更复杂的文本处理问题，逻辑设计的严谨性与场景适配性，直接影响处理效率与质量，体现务实创新的职业素养”，引出本次课核心内容，渗透“精益求精、务实创新”的思政理念。 理论讲解（10 分钟）：讲解分组匹配的语法与逻辑（用括号分组、提取分组内容），贪婪与懒惰匹配的区别及适用场景，re 库 sub 方法的使用语法（替换规则、替换次数）；结合思政点，强调“分组匹配的逻辑要严谨，贪婪与懒惰匹配的选择要贴合场景，每一次替换都要确保准确性，避免因规则设计不当导致文本错乱”，引导学生树立“精准适配、主动优化”的工作思维。

实操演示（25 分钟）：

1.字符组范围表示法

上次课我们讲到使用正则表达式来查找字符串，代码如下：

```
import re

mo = re.search(r"[0123456789][0123456789][0123456789]-[0123456789][0123456789][0123456789]-[0123456789][0123456789][0123456789][0123456789]", "My number is 415-555-4242")

print("Phone number found: " + mo.group())
```

上面的代码中，字符组 [0123456789] 的表示方法是很繁琐的，那有没有简单的方法呢？答案是使用字符组的范围表示法，[0123456789] 可以用范围表示法表示为 [0-9]。于是，上述代码使用范围表示法可以改写成：

```
import re

mo = re.search(r"[0-9][0-9][0-9]-[0-9][0-9][0-9]-[0-9][0-9][0-9][0-9]", "My number is 415-555-4242")

print('Phone number found: ' + mo.group())
```

2.字符组简记法

针对数字、字母以及空白字符等常见字符，正则表达式提供了更简洁的表示法。常见的字符组简记法有 \d、 \w、 \s ，其中 \d 等价于 [0-9] ，其中 d 代表数字；\w 等价于 [0-9a-zA-Z] ，其中 w 代表单词字符；\s 等价于 [\t\r\n\v\f]（第一个字符是空格），s 表示空白字符。使用字符组简记法，上述代码可以改写成：

```
import re

mo = re.search(r"\d\d\d-\d\d\d-\d\d\d\d", "My number is 415-555-4242")

print('Phone number found: ' + mo.group())
```

3.排除性字符组

字符组是匹配在一个位置可能出现的所有字符，有时候使用字符组匹配某个位置上可能出现的所有字符不是很方便，反而表示这个位置上不会出现的字符很方便。如果要表示某个位置上不会出现的字符，便用到了排除性字符组。例如当我们要匹配一个字符串中的所有非元音字符时，代码如下：

```
import re

mo = re.findall(r"[^aeiouAEIOU]", "RoboCop eats baby food. BABY FOOD")
for match in mo:
```

```
print(match)
```

上面的代码中我们使用排除性字符组 [^aeiouAEIOU] 来表示所有的非元音字符。排除型字符组非常类似普通字符组，只是在开方括号 [之后紧跟一个脱字符 ^，表示在当前位置匹配一个没有列出的字符。

相对于 \d、\w 和 \s 这三个普通字符组简记法，正则表达式也提供了对应排除型字符组的简记法：\D、\W 和 \S，字母完全一样，只是改为大写。这些简记法匹配的字符互补：\s 能匹配的字符，\S 一定不能匹配；\w 能匹配的字符，\W 一定不能匹配；\d 能匹配的字符，\D 一定不能匹配。

4.匹配开头和结尾

前面讲的字符串匹配都是用正则表达式去匹配字符串的一部分。除了部分匹配外，我们还有完全匹配、只匹配开头、只匹配结尾等需求。

4.1 匹配开头和结尾

```
import re

mo = re.search(r"^\d\d\d-\d\d\d-\d\d\d\d$", "My number is 415-555-4242")

print(mo == None)
```

为了进行完全匹配，只要在正则表达式的开头加上 ^ 字符，在结尾加上 $ 字符。由于进行的是完全匹配，所以上面的代码输出 True，也就是没有匹配到。下面将上述代码更改为如下形式：

```
import re

mo = re.search(r"^\d\d\d-\d\d\d-\d\d\d\d$", "415-555-4242")

print('Phone number found: ' + mo.group())
```

这样便可以匹配到电话号码了。

4.2 匹配结尾

为了进行尾部匹配，只要在正则表达式的尾部加上 $ 字符。代码如下：

```
import re

mo = re.search(r"\d\d\d-\d\d\d-\d\d\d\d$", "My number is 415-555-4242")

print("Phone number found: " + mo.group())
```

由于字符串的结尾刚好是电话号码，所以用尾部匹配是可以匹配到电话号码的。

4.3 匹配开头

为了进行头部匹配，只要在正则表达式的头部加上 ^ 字符。代码如下：

```
import re

mo = re.search(r"^\d\d\d-\d\d\d-\d\d\d\d", "My number is 415-555-4242")

print(mo == None)
```

由于进行的是头部匹配，所以上面的代码输出 True，也就是没有匹配到。下面将上述代码更改为如下形式：

```
import re

mo = re.search(r"^\d\d\d-\d\d\d-\d\d\d\d", "415-555-4242 is a number")

print("Phone number found: " + mo.group())
```

由于现在字符串的头部刚好是电话号码，所以用头部匹配是可以匹配到电话号码的。

5.匹配特殊字符

在正则表达式中，以下字符具有特殊含义：.^$*+?{}[]\|() 如果需要从字符串中匹配出以上具有特殊含义的字符，需要对以上字符进行转义：.^$*+?{}[]\|()。我们举个匹配括号的例子，代码如下：

```
import re

mo = re.search(r"\(\d\d\d\)-\d\d\d-\d\d\d\d", "My number is (415)-555-4242")

print("Phone number found: " + mo.group())
```

上面代码在进行匹配时，对括号进行了转义。

学生实操（40 分钟）：学生根据复杂测试文本，运用分组匹配提取信息，使用贪婪/懒惰匹配完成精准匹配，运用 sub 方法完成文本批量替换与数据清洗，教师巡回指导；重点关注学生的规则设计与匹配准确性，对逻辑混乱、替换错误的学生，及时指导；鼓励学生优化匹配规则，提升处理效率，培养务实创新的态度；引导学生分步调试规则，培养严谨的逻辑思维，强化责任意识。

5. 小结与布置任务（20 分钟）：总结正则表达式进阶知识点与

	实操技巧，点评学生实操情况，表扬规则设计严谨、匹配准确、善于优化的学生；布置课后任务（编写脚本，运用分组匹配与文本替换，完成杂乱文本的数据清洗，提交脚本与清洗前后对比结果），要求学生注重处理准确性与效率，同时思考“如何通过优化正则规则提升文本处理的高效性”。
教学后记	1. 课堂亮点：多数学生能掌握分组匹配、贪婪与懒惰匹配的用法，熟练使用 sub 方法完成文本批量替换；部分学生能主动优化匹配规则，提升处理效率，体现出较强的务实创新意识；学生的逻辑思维与问题解决能力有所提升，能分步调试规则，排查匹配与替换错误；对复杂文本处理的兴趣较高，能主动尝试解决难题。 2. 存在不足：少数学生对分组匹配的逻辑设计掌握不熟练，无法准确提取复杂信息；部分学生无法准确区分贪婪与懒惰匹配的适用场景，导致匹配偏差；个别学生文本替换规则设计不严谨，出现替换错误、文本错乱的情况；少数学生缺乏耐心，遇到复杂规则调试时容易放弃，独立解决问题的能力有待提升。 3. 改进措施：下次课课前，展示分组匹配与贪婪/懒惰匹配的典型案例，集中讲解规则设计技巧；实操过程中，引导学生分步设计规则、分步调试，培养严谨的逻辑思维；加强对文本替换规则设计的指导，帮助学生避免替换错误；对基础薄弱的学生进行个别辅导，帮助其理解进阶知识点；结合数据清洗案例，深化“精益求精、务实创新”的职业理念，提升学生的文本处理能力。

第 17 周 教案（2 学时）

教学课题	正则表达式综合实操：办公场景数据清洗与文本批量处理
教学学时	2 学时
教学目标	• 知识目标：整合正则表达式核心知识点，掌握办公场景下数据清洗与文本批量处理的完整流程，理解正则表达式与 Python 办公自动化的结合逻辑； • 能力目标：能独立编写脚本，运用正则表达式完成办公场景下的复杂数据清洗、文本批量提取、格式批量修改，解决实际办公文本处理需求； • 素养目标（思政）：培养综合实操能力与问题解决能力，注

	重数据清洗的准确性与高效性，渗透“精益求精、责任担当”的职业理念，培养务实高效、主动作为的工作态度。
教学重难点	重点：正则表达式在办公场景中的综合应用，数据清洗与文本批量处理的完整流程，脚本的编写与优化； 难点：复杂文本处理规则的设计与调试，正则表达式与 Python 办公自动化的结合应用，异常数据的处理。
教学过程（融入思政）	导入（5 分钟）：回顾前 2 周正则表达式知识点，展示 3 类办公场景综合案例（员工信息数据清洗、公文文本格式批量修改、报表文本信息提取），说明“正则表达式的综合应用能大幅提升办公效率，解决实际办公难题，数据清洗的准确性直接影响后续工作质量，体现责任担当”，明确本次课综合实操任务，渗透“精益求精、责任担当”的思政理念。 理论讲解（5 分钟）：梳理正则表达式综合应用的核心流程（需求分析、规则设计、脚本编写、结果校验），强调正则表达式与 Python 办公自动化的结合要点（读取文本/数据、运用正则处理、保存结果）；结合思政点，强调“综合实操需要统筹规划，每一步规则设计与脚本编写都要严谨，数据清洗要确保准确，避免因处理失误导致后续工作出错，体现责任担当”，引导学生树立“全流程把控”的工作思维。 实操演示： 1. 准备测试数据（employee_raw.txt） Plain Text 【员工信息】张三|男|1990-05-12|部门：技术部|工号：JS001|联 13800138001 入职时间：2020/01/15 【员工信息】李四|女|1988-08-25|部门：人事部|工号：RS002|联 13900139002 入职时间：2018/03/20 备注：无 【员工信息】王五|男|1995-11-08|部门：财务部|工号：CW003|联 13700137003 入职时间：2022/05/10 （试用期） 【员工信息】赵六|女|1992-03-18|部门：市场部|工号：SC004|联 13600136004 入职时间：2021/07/08 备注：驻外 【员工信息】钱七|男|1985-09-30|部门：技术部|工号：JS005|联 13500135005 入职时间：2019/09/01 （离职待办） 【异常数据】孙八|女|1998-XX-20|部门：行政部|工号：XZ006|联 13400134006 入职时间：2023/02/15 （日期格式错误）

【员工信息】周九|男|1991-07-05|部门：技术部|工号：JS007|联系电话：13300133007 入职时间：2020/11/30

2. 完整实操代码

```
Python
import re
import csv
import os

def clean_employee_data(raw_file_path, cleaned_file_path):
    """
    清洗员工信息数据的核心函数
    :param raw_file_path: 原始杂乱数据文件路径
    :param cleaned_file_path: 清洗后数据保存路径
    :return: 清洗成功的条数、异常数据列表
    """
    # 步骤 1：定义正则规则（严谨设计规则，确保匹配准确，体现
    # 分组匹配：姓名、性别、出生日期、部门、工号、电话、入职
    # 规则说明：兼顾格式多样性，同时校验日期/电话等关键信息的
    pattern = r'【员工信息】([^|]+)\|([^|]+)\|(\d{4}-\d{2}-\d{2})\|部门：
([^|]+)\|联系电话：(\d{11})\s+入职时间：(\d{4}/\d{2}/\d{2})'
    # 异常数据匹配规则（标记不符合规范的数据）
    error_pattern = r'【(.*?)】([\s\S]*?)（(.*?)）'

    # 步骤 2：读取原始文件（确保文件读取容错，避免因文件问题
    try:
        with open(raw_file_path, 'r', encoding='utf-8') as f:
            lines = f.readlines()
    except FileNotFoundError:
        print(f"错误：未找到文件 {raw_file_path}，请检查文件路径！
        return 0, ["文件不存在"]
    except Exception as e:
        print(f"读取文件时发生异常：{str(e)}")
        return 0, [f"读取异常：{str(e)}"]

    # 步骤 3：遍历处理每一行数据，清洗+筛选异常
    cleaned_data = []  # 存储清洗后的有效数据
    error_data = []    # 存储异常数据（便于后续核对处理，体现全
    success_count = 0
```

```
    for line_num, line in enumerate(lines, 1):
        line = line.strip()  # 去除首尾空格/换行
        if not line:
            continue  # 跳过空行

        # 匹配有效员工信息
        match = re.match(pattern, line)
        if match:
            # 提取分组数据，标准化格式
            name = match.group(1).strip()
            gender = match.group(2).strip()
            birth_date = match.group(3).strip()
            department = match.group(4).strip()
            employee_id = match.group(5).strip()
            phone = match.group(6).strip()
            hire_date = match.group(7).strip()

            # 将入职时间格式统一为 yyyy-mm-dd（格式标准化，提升数据可用性）
            hire_date = re.sub(r'/', '-', hire_date)

            # 组装清洗后的数据
            cleaned_row = [name, gender, birth_date, department, employee_id,
phone, hire_date]
            cleaned_data.append(cleaned_row)
            success_count += 1
        else:
            # 识别异常数据并记录（行号+内容+异常说明）
            error_match = re.match(error_pattern, line)
            if error_match:
                error_type = error_match.group(1)
                error_note = error_match.group(3)
                error_info = f"第{line_num}行 - 类型：{error_type} - 内容：{line} - 异常说
明：{error_note}"
            else:
                error_info = f"第{line_num}行 - 内容：{line} - 异常说明：格式不匹配，无
法提取有效信息"
            error_data.append(error_info)

    # 步骤 4：保存清洗后的数据（确保数据持久化，格式规范便于后续使用）
    try:
        with open(cleaned_file_path, 'w', encoding='utf-8', newline='') as csvfile:
            # 定义表头（标准化表头，符合办公数据规范）
```

```
            fieldnames = ['姓名', '性别', '出生日期', '部门', '工号', '联系电话', '入职时间']
            writer = csv.DictWriter(csvfile, fieldnames=fieldnames)
            writer.writeheader()  # 写入表头
            # 写入清洗后的数据
            for row in cleaned_data:
                writer.writerow({
                    '姓名': row[0],
                    '性别': row[1],
                    '出生日期': row[2],
                    '部门': row[3],
                    '工号': row[4],
                    '联系电话': row[5],
                    '入职时间': row[6]
                })
        print(f"清洗后的数据已保存至：{cleaned_file_path}")
    except Exception as e:
        print(f"保存文件时发生异常：{str(e)}")
        return success_count, [f"保存异常：{str(e)}"] + error_data

    # 步骤 5：输出校验信息（主动校验结果，体现严谨性）
    print(f"\n=== 数据清洗结果校验 ===")
    print(f"原始数据总行数：{len(lines)}")
    print(f"清洗成功条数：{success_count}")
    print(f"异常数据条数：{len(error_data)}")
    if error_data:
        print(f"\n 异常数据详情：")
        for error in error_data:
            print(f"- {error}")

    return success_count, error_data

# 主程序执行（统筹规划执行流程，全流程把控）
if __name__ == "__main__":
    # 定义文件路径（确保路径可配置，提升脚本灵活性）
    raw_file = "employee_raw.txt"
    cleaned_file = "employee_cleaned.csv"

    # 检查原始文件是否存在
    if not os.path.exists(raw_file):
        print(f"请先创建原始数据文件：{raw_file}，并填入测试数据！")
    else:
```

```
    # 执行数据清洗
    success_num, error_list = clean_employee_data(raw_file, cleaned_file)

    # 思政引导：强调数据准确性的重要性
    if error_list and "文件" not in error_list[0]:
        print(f"\n【重要提醒】本次清洗发现{len(error_list)}条异常数据，需逐一核对修正！")
        print("数据清洗的准确性直接影响后续员工档案管理、薪资核算等工作，务必严谨处理每一条数据，体现责任担当！")
```

3. 分步讲解

步骤 1：需求分析与规则设计

需求：从杂乱文本中提取姓名、性别、出生日期等核心信息，标准化格式，筛选异常数据，保存为 CSV（便于 Excel 打开）。

规则设计：

核心匹配规则 pattern：通过分组 () 精准提取关键信息，限定电话为 11 位数字、出生日期为 yyyy-mm-dd 格式，确保数据合法性；

异常匹配规则 error_pattern：识别标记异常数据，便于后续核对，避免错误数据流入后续工作。

步骤 2：读取原始文件

增加 try-except 异常处理，避免因文件不存在、编码错误等导致程序崩溃，体现代码的健壮性；

读取后按行处理，跳过空行，提升数据处理效率。

步骤 3：数据清洗与异常筛选

遍历每一行，用 re.match()匹配有效数据，提取分组信息后统一入职时间格式（/替换为-），实现格式标准化；

未匹配到的行标记为异常数据，记录行号、内容和异常说明，便于后续人工核对修正，体现“全流程把控”的工作思维。

步骤 4：保存清洗后数据

保存为 CSV 格式（办公常用格式），定义标准化表头，便于 Excel 直接打开使用；

增加保存异常处理，确保数据保存环节不丢失成果。

步骤 5：结果校验

输出原始行数、成功条数、异常条数，直观展示处理结果；

打印异常数据详情，引导学生重视异常数据处理，强化责任意识。

三、演示执行与结果展示

1. 执行代码

运行脚本后，控制台输出：

```
Plain Text
清洗后的数据已保存至：employee_cleaned.csv

=== 数据清洗结果校验 ===
原始数据总行数：7
清洗成功条数：6
异常数据条数：1

异常数据详情：
- 第 6 行 - 类型：异常数据 - 内容：【员工信息】孙八|女|1998-XX
|工号：XZ006|联系电话：13400134006 入职时间：2023/02/15
- 异常说明：日期格式错误

【重要提醒】本次清洗发现 1 条异常数据，需逐一核对修正！
数据清洗的准确性直接影响后续员工档案管理、薪资核算等工作，
一条数据，体现责任担当！
```

2. 查看清洗后文件（employee_cleaned.csv）

姓名	性别	出生日期	部门	工号	联系
张三	男	1990-05-12	技术部	JS001	138800
李四	女	1988-08-25	人事部	RS002	139900
王五	男	1995-11-08	财务部	CW003	137700
赵六	女	1992-03-18	市场部	SC004	136600

钱七	男	1985-09-30	技术部	JS005	13500135005	2019-09-01
周九	男	1991-07-05	技术部	JS007	13300133007	2020-11-30

学生综合实操（50 分钟）：学生独立选取办公场景（数据清洗、文本提取、格式修改），编写脚本完成正则表达式综合应用任务，处理异常数据并校验结果，教师巡回指导；重点关注学生的规则设计、脚本规范性与处理准确性，对规则混乱、处理错误的学生，及时指导；鼓励学生优化脚本，提升处理效率，培养务实高效的态度；引导学生妥善处理异常数据，强化责任意识；对完成较快的学生，引导其拓展脚本功能，结合 Excel/Word 自动化，实现全流程办公处理。

小结与布置任务（20 分钟）：总结正则表达式综合应用的核心要点与实操技巧，点评学生实操作品，表扬规则设计严谨、处理准确、善于优化的学生；布置课后任务（完善综合实操脚本，结合 Python 办公自动化，完成“文本处理-数据保存”全流程操作，提交脚本与处理报告），要求学生注重处理准确性与流程完整性，同时反思“自己在综合实操中的不足，如何提升问题解决能力与职业素养”。

教学后记

1. 课堂亮点：多数学生能独立完成正则表达式综合实操任务，顺利解决办公场景下的文本处理与数据清洗需求；部分学生能主动优化脚本，结合 Python 办公自动化实现全流程处理，体现出较强的综合实操能力与创新意识；学生的问题解决能力与责任意识有所提升，能妥善处理异常数据，主动校验处理结果；编码规范度持续提升，多数学生能为脚本添加清晰注释。

2. 存在不足：少数学生对复杂场景的需求分析不透彻，规则设计不合理，导致处理效率低下；部分学生脚本编写不规范，缺乏注释，难以维护；个别学生对异常数据的处理能力不足，无法独立排查复杂规则错误；少数学生缺乏统筹规划意识，操作步骤混乱，影响处理效率。

3. 改进措施：下次课课前，简要回顾正则表达式综合应用流程，针对常见的规则设计与异常处理问题进行集中讲解；实操过程

	中，引导学生先分析需求、设计规则，再编写脚本，强化统筹规划意识；加强对脚本规范性与异常数据处理的指导，提升脚本可维护性与处理准确性；对基础薄弱的学生进行个别辅导，帮助其提升问题解决能力；结合职场实际案例，深化“精益求精、责任担当”的职业理念，提升学生的综合职业素养。

第 18 周 教案（2 学时）

教学课题	课程综合实操与实训总结（Word+Python 办公自动化全流程）
教学学时	2 学时
教学目标	• 知识目标：整合本课程所有核心知识点（Word 长文档排版、Python-Excel 自动化、Python 批量邮件发送、正则表达式），掌握办公自动化全流程的核心逻辑； • 能力目标：能独立完成办公自动化全流程实操任务（文档排版-数据处理-邮件发送-文本清洗），能调试优化脚本，解决全流程中的常见问题； • 素养目标（思政）：培养综合应用能力与职业适配能力，总结实训收获与不足，渗透“终身学习、精益求精”的职业理念，培养务实肯干、勇于担当的职业素养，树立正确的职场价值观。
教学重难点	重点：课程核心知识点的整合应用，办公自动化全流程的实操，实训总结与反思； 难点：全流程操作的统筹衔接，跨模块问题的排查与解决，实训收获的深度总结。
教学过程（融入思政）	1. 导入（5 分钟）：回顾本课程 18 周核心内容，梳理办公自动化全流程（Word 排版-Excel 数据处理-正则文本清洗-批量邮件发送），说明“综合应用能力是职场核心竞争力，实训的目的是提升职业适配性，总结反思能帮助自身持续进步”，明确本次课核心任务（综合实操+实训总结），渗透“终身学习、精益求精”的思政理念。 2. 理论回顾与总结（10 分钟）：简要回顾课程核心知识点与重难点，梳理各模块的衔接逻辑，强调“办公自动化的核心是高效、规范、准确，每一个模块的操作都要贴合职场需求，体现职业素养”；结合思政点，引导学生回顾实训过程中的收获与不足，强调

	“反思是成长的关键，终身学习能不断提升自身竞争力”，引导学生树立终身学习理念。 3. 综合实操任务布置（5 分钟）：布置全流程综合实操任务（编写 Word 规范文档-用 Python 处理 Excel 数据-用正则清洗文本数据-批量发送带附件邮件），明确任务要求、考核标准与思政素养评价要点，强调“全流程的统筹衔接，操作规范，数据准确，脚本优化”，渗透责任意识与工匠精神。 4. 学生综合实操（40 分钟）：学生独立完成全流程综合实操任务，调试优化脚本，排查跨模块问题，教师巡回指导，重点解决全流程衔接中的难点问题；关注学生的综合应用能力与操作规范性，对流程混乱、问题较多的学生，及时指导；鼓励学生相互交流，分享实操技巧，培养团队协作能力；引导学生注重每一个细节，确保操作准确、规范，强化责任意识。 5. 实训总结与点评（20 分钟）：邀请学生分享实训收获、实操心得与不足，教师进行集中点评，肯定学生的进步，指出普遍存在的问题与改进方向；总结本课程的核心收获，强调“办公自动化技能的实用性与重要性，严谨细致、精益求精的职业素养是职场立足的关键”；引导学生制定后续学习计划，树立终身学习理念，鼓励学生将所学技能应用到实际工作中，培养务实肯干、勇于担当的职业态度。
教学后记	1. 课堂亮点：多数学生能顺利完成办公自动化全流程综合实操任务，体现出较强的综合应用能力；学生能主动分享实训收获与反思，对自身不足有清晰的认知，树立了终身学习理念；部分学生能优化全流程脚本，提升操作效率，体现出较强的创新意识与务实态度；团队协作氛围良好，学生能相互交流解决跨模块难题，责任意识与职业素养得到进一步提升。 2. 存在不足：少数学生全流程衔接不顺畅，跨模块问题的排查能力不足，无法独立解决复杂难题；部分学生操作不够规范，细节把控不到位，存在格式混乱、数据错误等问题；个别学生实训总结不够深入，未能结合自身实操情况进行反思；少数学生对所学技能的职场应用认知不足，缺乏主动应用的意识。 3. 改进措施：后续可提供拓展实训案例，供学生课后练习，提升综合应用能力；整理课程重难点与常见问题手册，方便学生后续复习；引导学生关注职场办公自动化最新动态，主动学习新技能，践行终身学习理念；结合职场实际应用场景，开展拓展教学，提升学生的职业适配性；持续强化工匠精神与责任意识，帮

	助学生树立正确的职场价值观，为未来职场发展奠定基础。

五、课程考核方式

本课程以实操考核为主，理论考核为辅，综合评价学生的知识掌握、能力提升与素养养成情况，考核占比如下：

- 平时实操表现（40%）：包括每周实操任务完成情况、操作规范性、思政素养表现（责任意识、严谨性等）；
- 课程终期综合考核（60%）：完成办公自动化全流程实操任务，提交实操作品与实训总结，考核综合应用能力与职业素养。

六、课程总结

本课程通过 18 周的实操教学，围绕 Word 2016 长文档排版与 Python 办公自动化核心技能，结合职场实际场景，系统讲解了长文档排版、Excel 数据处理、批量邮件发送、正则表达式文本处理的核心知识点与实操技巧。教学过程中，始终将工匠精神、责任意识、数据安全意识等思政元素融入每节课，注重培养学生的规范操作习惯、综合实操能力与职业素养。

通过本课程学习，学生能熟练掌握办公自动化核心技能，能独立解决职场常见的办公难题，建立高效、规范的办公思维，为未来从事计算机应用、办公自动化相关工作奠定坚实基础。同时，引导学生树立终身学习理念，培养务实肯干、精益求精的职业态度，助力学生成长为符合职场需求的技能型人才。

信息工程系

教

案

课程名称：专业技能实训 IV

教　　师：黄苗苗

总 学 时 ：36（总 72）

理论学时：0

实训学时：36

上课班级：计算机 241 1 组

授课学期：25-26（2）

课 程 性 质

本课程属于实践性教学环节，是高等教育或职业教育中培养学生专业能力、职业素养和实际操作技能的重要课程类型

课程要求侧重掌握为将来从事开发等提供必要的基础知识，打下良好的基础。

本课程负责其中一部分——常见计算机应用工具 git 的使用。

课程要求：

1. 熟悉基础命令行操作（Windows/Linux 终端或 Mac Terminal）
2. 熟悉 office 高级应用软件
3. 熟练使用 Access 数据库
4. 应用 Access 数据库
5. 安装 Git 客户端（推荐 Git 官方版本）。
6. 注册 GitHub/GitLab/Gitee 账号（需科学上网访问 GitHub 时提前准备）。
7. 配置 SSH Key（实训完成）

课程性质：

计算机综合实训专业课程，本课程是对其中一部分——Git——做导入的介绍使用，需要学生在此前具备相应专业知识，同时了解专业动态，对此投入适当的的学习量，并将其应用到实际操作中才能切实掌握该技术。

课程教学手段

依据课程性质及周课时安排：

实验课（2 节/周）：

针对本课程特性，实操性实验课是检验学生掌握程度和加深学生技能的重要过程。

实际结合网络知识以及日常工作可能布置网络应用设计情况设计。需要从浅入深，有基础语法的小实验一步一步扎实练习，最后才能汇总为大的设计完成。

实验课的难度系数属于中等难度系数，需学生课后配合多加练习。

第 0 篇 概述

课时安排： 4 学时（理论讲解）

前置知识：office 高级应用、基础命令行操作、软件开发生命周期概念

教学目标：

- 知识目标：了解 ACCESS 数据库；了解 Git 的发展历史和核心设计理念。掌握版本控制系统的基本概念（集中式 vs 分布式）。
- 能力目标：能区分 Git 与其他版本控制系统（如 SVN、CVS）的差异。能解释分布式版本控制的优势及适用场景。
- 素养目标：培养对版本控制的规范意识（如提交日志、分支管理）。理解开源协作的文化背景（如 GitHub 社区）。

教学重点与难点：

- 重点：

  Access 的窗体

  Access 的宏应用

  VBA 浅涉

  Git 的分布式特性与工作流程。

  四大工作区的功能与交互逻辑。
- 难点：

  Access 在本机的高级综合应用

  分布式版本控制与集中式的本质区别。

  暂存区（Stage）的设计意义（为什么需要 git add？）。

教学内容与过程

1. 课程导入

   作为计算机应用专业，我们可能需要应用到那些方面的专业技能？

   是否有对应的成熟应用或软件为我们服务？

2.掌握 Access 的启动和界面操作

导入（5 分钟）

- 提问：大家平时怎么管理数据？Excel 的优缺点？
- 引出：当数据量大、关系复杂时，需要数据库

  - 创建空数据库（15 分钟）

    1. 启动 Access
    2. 选择“空数据库”
    3. 命名规范
       - 见名知意：学生管理.accdb、图书借阅.accdb
       - 不包含特殊符号
    4. 保存位置选择
       - 建议放在 D 盘或 E 盘自己的文件夹

- 创建空表
  1. 表的组成
     - 字段（列）：数据的类别
     - 记录（行）：一条完整的数据
  2. 数据类型讲解
     - 文本：姓名、地址
     - 数字：年龄、价格
     - 日期/时间：出生日期、借书日期
     - 是否：性别（男/女）、是否毕业
  3. 创建第一个表：学生表
     - 字段设计：
       - 学号（文本，主键）
       - 姓名（文本）
       - 性别（是否，或文本）
       - 出生日期（日期/时间）
       - 联系电话（文本）
  4. 设置主键
     - 什么是主键？唯一标识一条记录
     - 如何设置：选中字段 → 主键按钮

3.Git 核心概念讲解

提问互动：

“如果多人同时修改同一份代码，如何避免冲突？”

“如果没有版本控制，如何回退到昨天的代码状态？”

案例展示：

展示 GitHub 上开源项目（如 Linux 内核）的协作历史，说明 Git 的实际价值。

（1）Git 的历史与特点

关键点：

由 Linus Torvalds 开发，初衷是管理 Linux 内核代码。

分布式设计：每个开发者拥有完整的仓库副本，支持离线操作。

对比表格：

特性	Git（分布式）	SVN/CVS（集中式）
服务器依赖	无需实时连接服务器	必须连接服务器提交/更新
版本号机制	哈希 ID（唯一标识）	线性递增数字版本号
容灾能力	本地仓库即备份	服务器损坏则历史丢失

（2）Git 工作流程

四大区域：

- Workspace（工作区）：本地文件目录（未跟踪的修改）。
- Index（暂存区）：通过 git add 暂存变更，准备提交。
- Repository（本地仓库）：通过 git commit 生成版本快照。
- Remote（远程仓库）：如 GitHub，用于团队协作（git push/pull）。

流程图：

工作区 → `git add` → 暂存区 → `git commit` → 本地仓库 → `git push` →

远程仓库

（3）Git 安装与配置

演示步骤：

下载 Git for Windows（官网链接）。

安装后运行 Git Bash，配置全局用户信息：

bash

git config --global user.name "YourName"

git config --global user.email "email@example.com"

3. 课堂练习

任务 1：在本地创建一个 Git 仓库并提交初始版本。

bash

mkdir my-project

cd my-project

git init

echo "# Hello Git" > README.md

git add README.md

git commit -m "Initial commit"

任务 2：对比 git status、git log、git diff 的输出差异。

4. 总结与答疑

- 核心总结：

Git 的核心价值：分布式、高效协作、完整历史追溯。

工作流程的本质：通过暂存区控制提交粒度。

- 常见问题：

Q：为什么 Git 需要暂存区？

A：允许选择性提交部分修改，提高提交的灵活性。

Q：GitHub 和 Git 的关系？

A：Git 是工具，GitHub 是基于 Git 的代码托管平台。

- 课后作业

实践作业：

在 GitHub 上创建个人仓库，完成首次 push 操作。

思考题：

集中式版本控制系统（如 SVN）是否仍有适用场景？举例说明。

- 教学资源

推荐阅读：

Git 官方文档

Git 可视化学习工具

课件下载：学习通平台（2025 年版本）。

- 备注

本教案结合 PPT 内容提炼关键知识点，后续课程将深入分支管理、冲突解决等实操内容。

强调实践：建议在后续课时中安排团队协作项目（如模拟 GitHub 的 PR 流程）。

第 1 篇 Access 数据库应用

课时安排：18 学时（理论+实操）

教学目标

- 知识目标

掌握本地数据库 Access 的创建与管理流程。

学会使用基础高级窗体和报表。

尽可能使用宏设计、VBA 应用

- 能力目标

能独立完成本地数据库的使用。

- 素养目标

为综合应用设计系统。

理解本地数据库的高级灵活度。

教学重点与难点

- 重点：
  - 创建空数据库
  - 创建空表
  - 导入外部数据
  - 数据库表间关系
  - 查询（SQL）
  - 利用查询完成的其他更新表的操作
  - 创建窗体：导入&输出
  - 报表

- 难点：

窗体、报表与 VBA

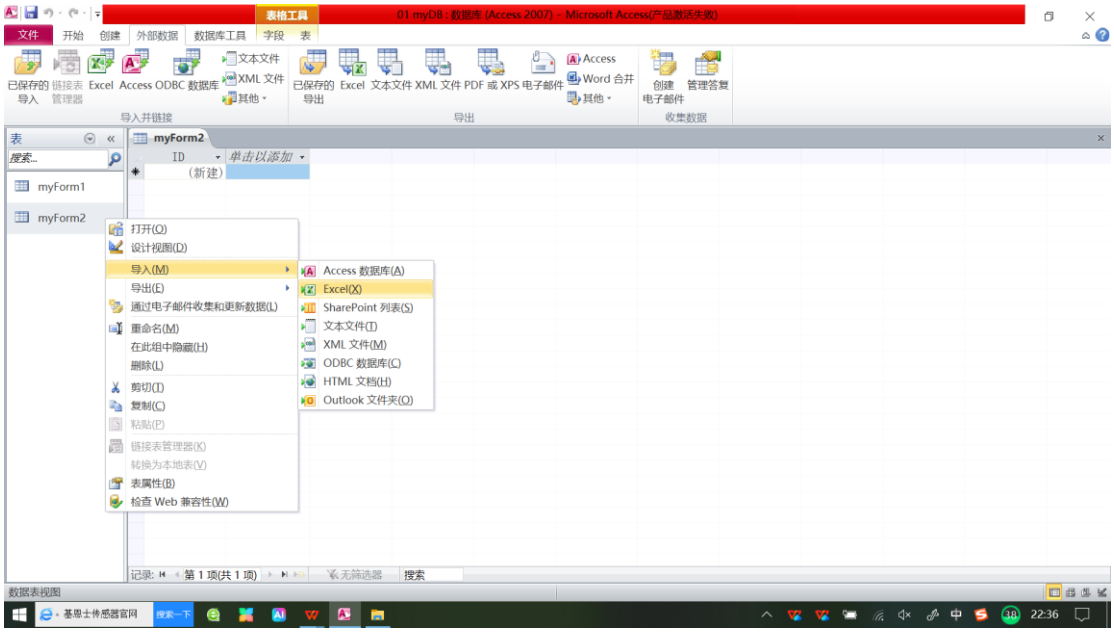
教学内容与过程

1. 课程导入

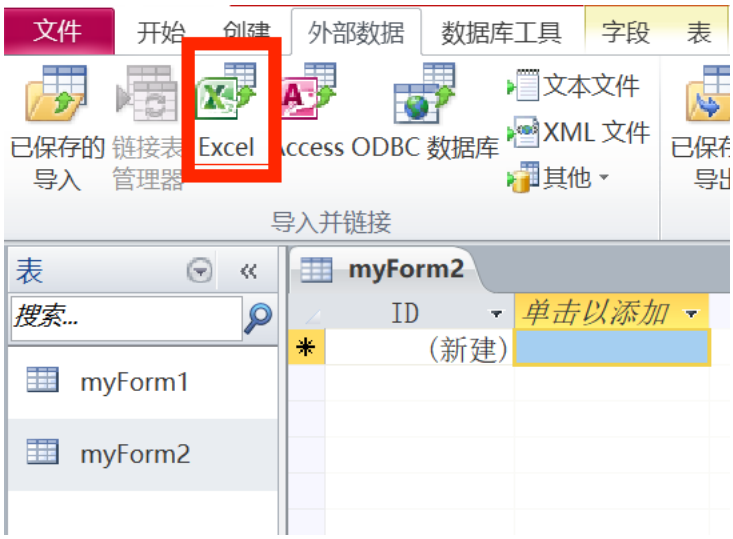
(1)

① 导 入 外 部 数 据

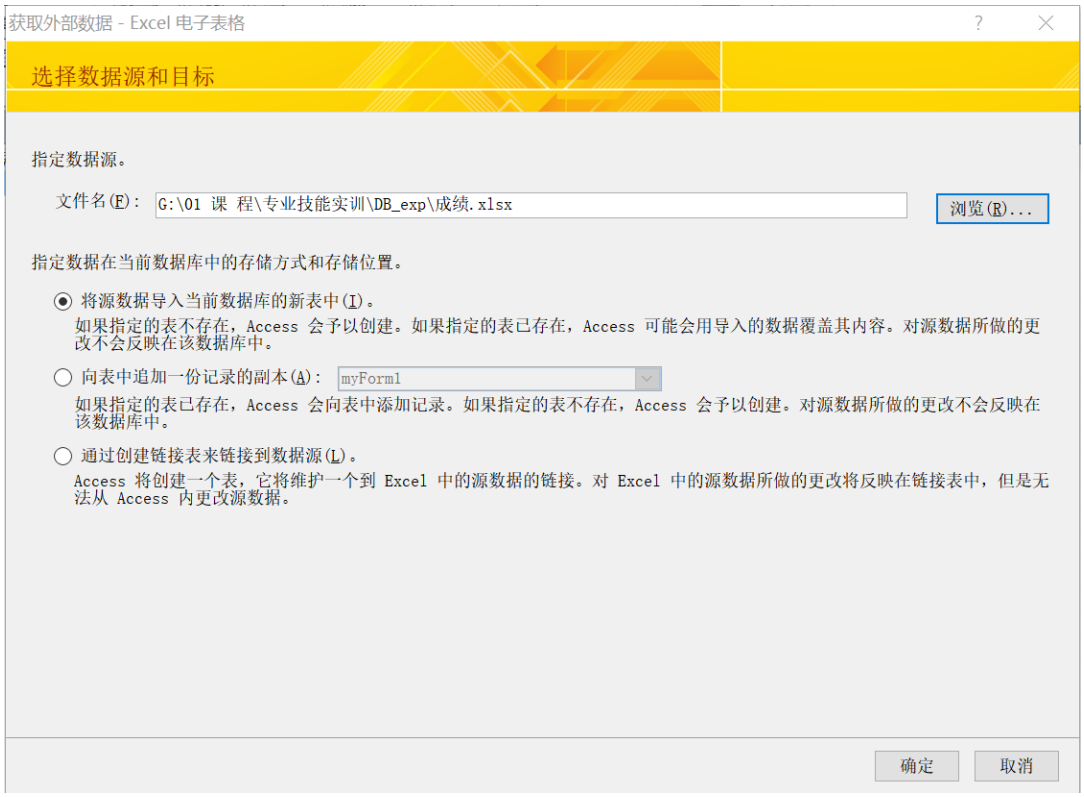
- 可以从外部已经存在的表，导入当前数据库中

- 

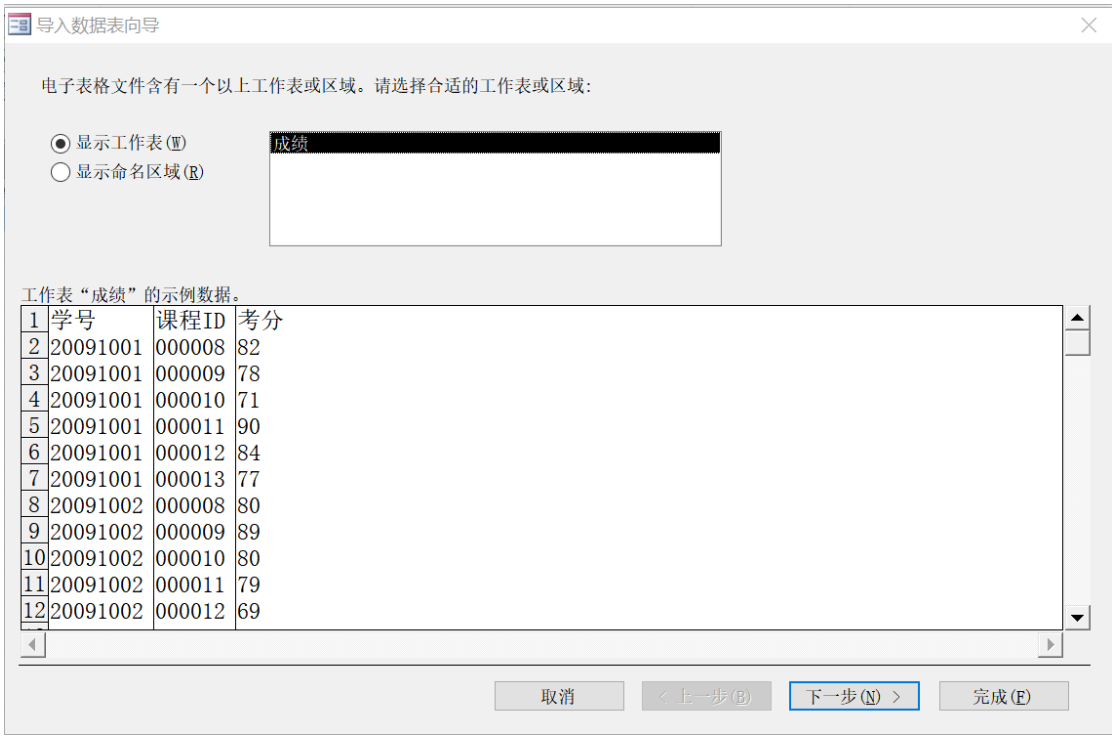
- 也可以选择另一种方式：

- 

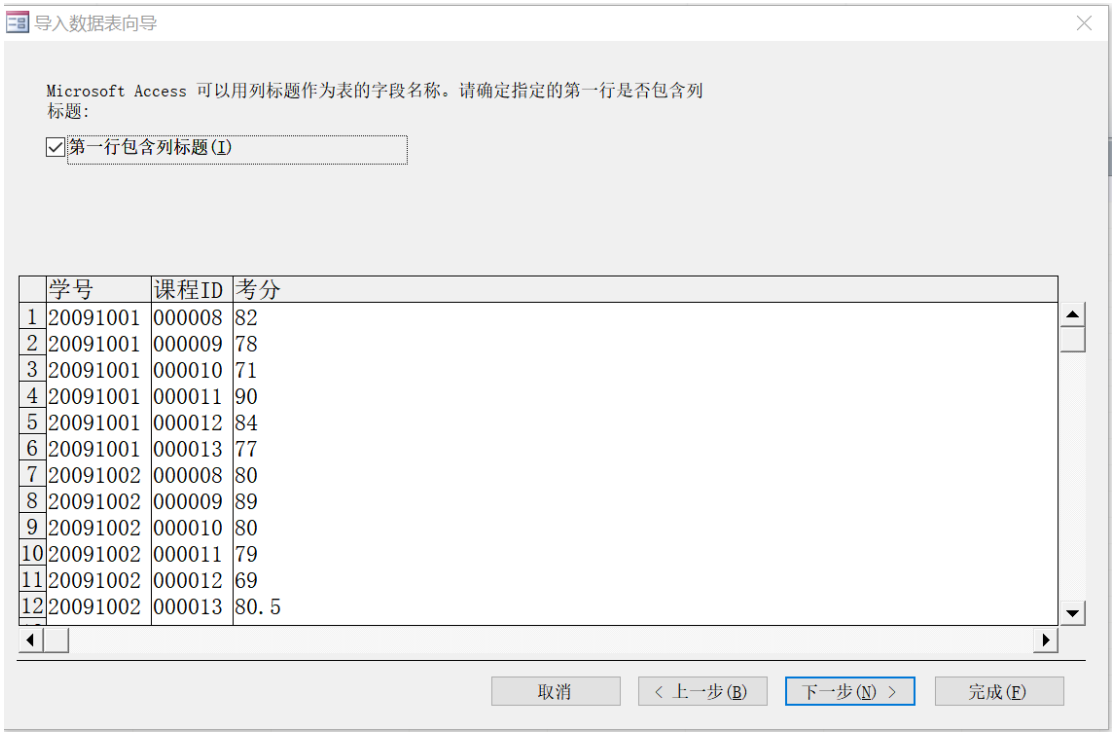
- 导入比如已经准备好的数据表“成绩.xls”


获取外部数据 - Excel 电子表格
选择数据源和目标
指定数据源。
文件名(F)：
G:\01 课 程\专业技能实训\DB_exp\成绩.xlsx
浏览(R)...
指定数据在当前数据库中的存储方式和存储位置。
将源数据导入当前数据库的新表中(I)。
如果指定的表不存在，Access 会予以创建。如果指定的表已存在，Access 可能会用导入的数据覆盖其内容。对源数据所做的更改不会反映在该数据库中。
向表中追加一份记录的副本(A)：
myForm1
如果指定的表已存在，Access 会向表中添加记录。如果指定的表不存在，Access 会予以创建。对源数据所做的更改不会反映在该数据库中。
通过创建链接表来链接到数据源(L)。
Access 将创建一个表，它将维护一个到 Excel 中的源数据的链接。对 Excel 中的源数据所做的更改将反映在链接表中，但是无法从 Access 内更改源数据。
确定
取消

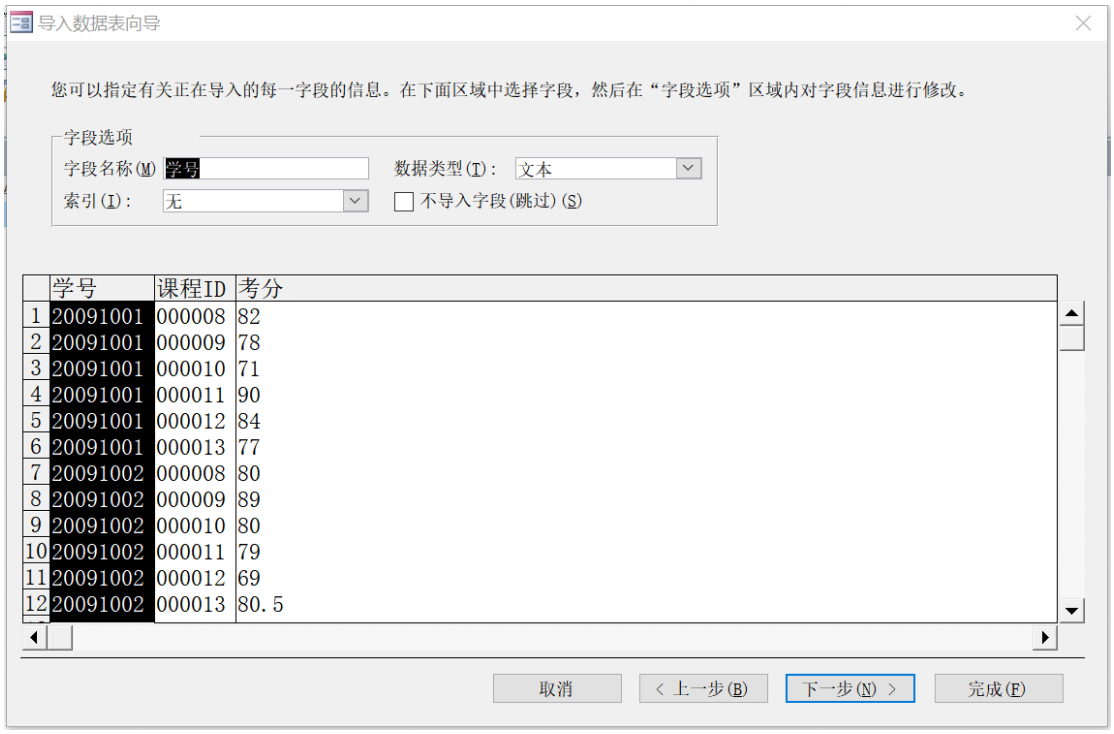
✧


导入数据表向导
电子表格文件含有一个以上工作表或区域。请选择合适的工作表或区域：
显示工作表(W)
显示命名区域(R)
成绩
工作表“成绩”的示例数据。
学号 课程ID 考分
20091001 000008 82
20091001 000009 78
20091001 000010 71
20091001 000011 90
20091001 000012 84
20091001 000013 77
20091002 000008 80
20091002 000009 89
20091002 000010 80
20091002 000011 79
20091002 000012 69
取消
< 上一步(B)
下一步(N) >
完成(F)

✧

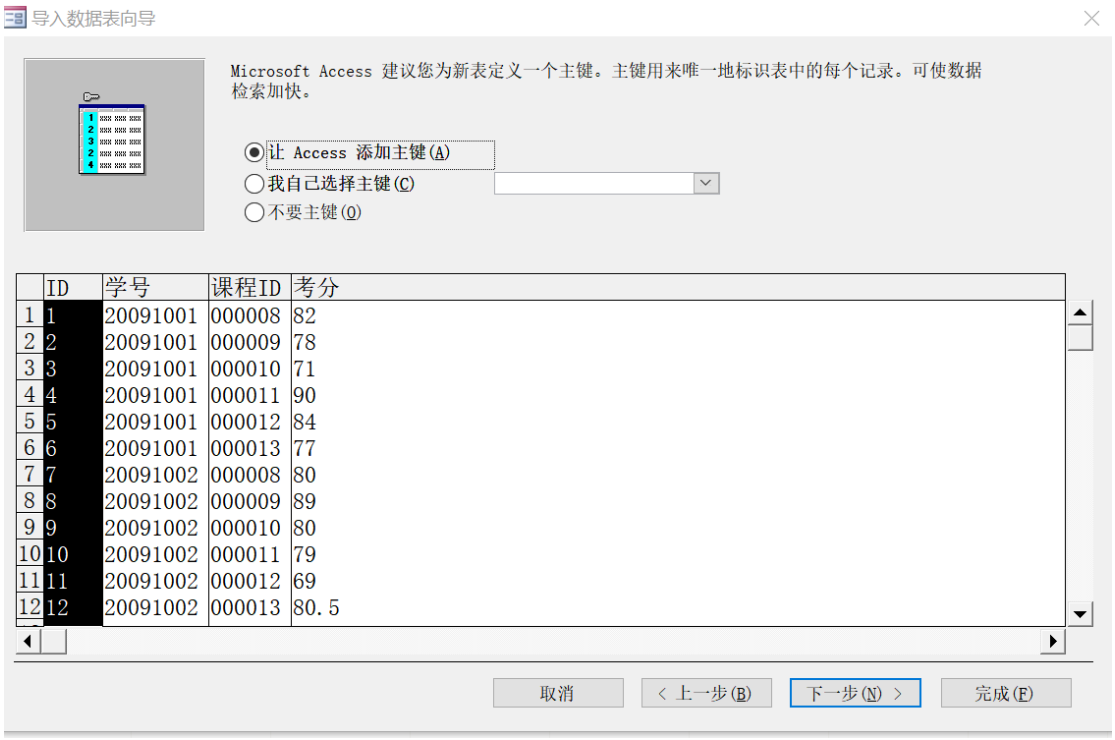


✧

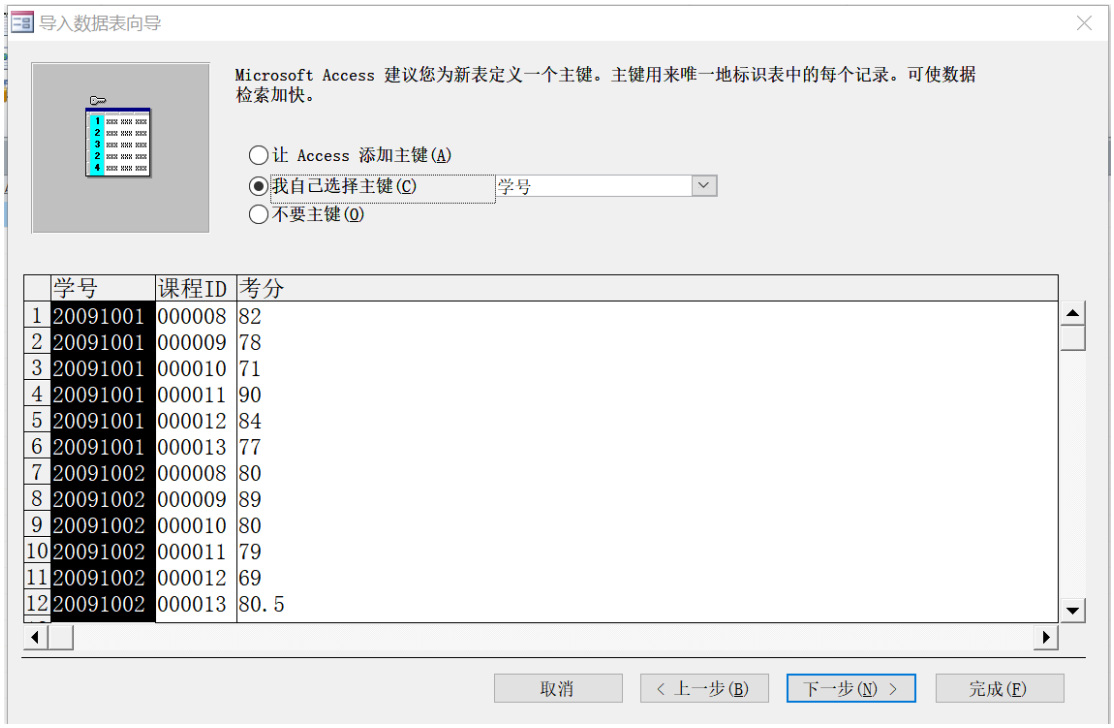


✧

✧ （这里，你选中哪一个字段，便显示在上方进行设置）

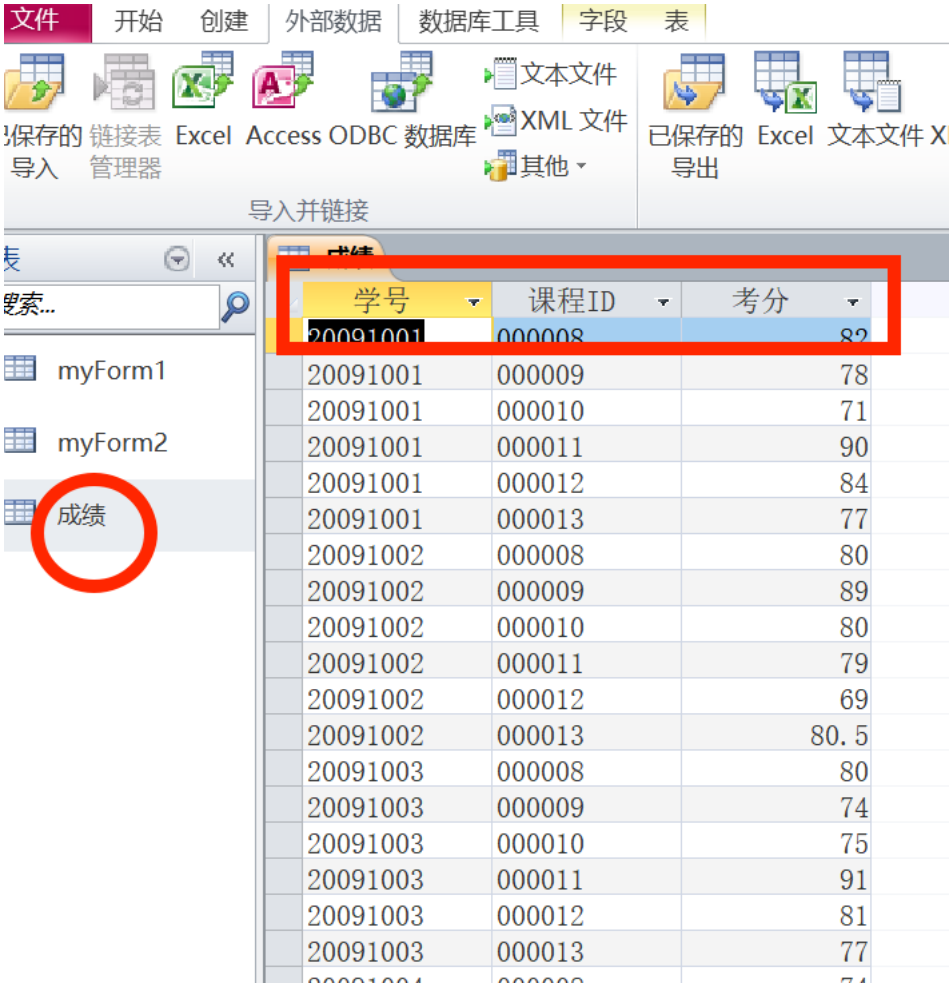


- ✧
- ✧ （
  - ■ **让 Access 添加**是生成新的主键
  - ■ **自己选择**则不会
- ✧ ）
- ✧ 我们这里选择了“学号”



- ✧

✧ 完成之后，可见如下一些表格和截图



✧

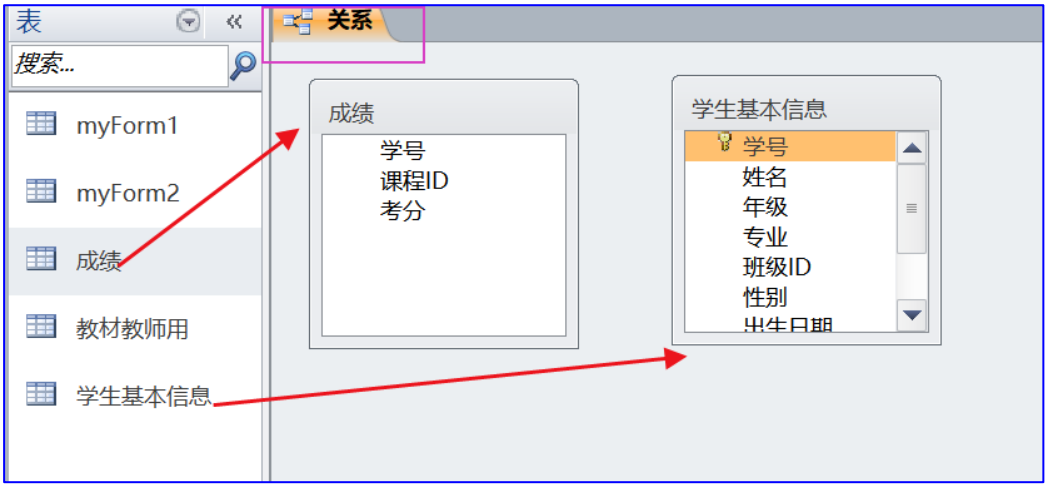
✧ 记得保存

- 使用相同的方法导入“学生基本信息”
-
-
-

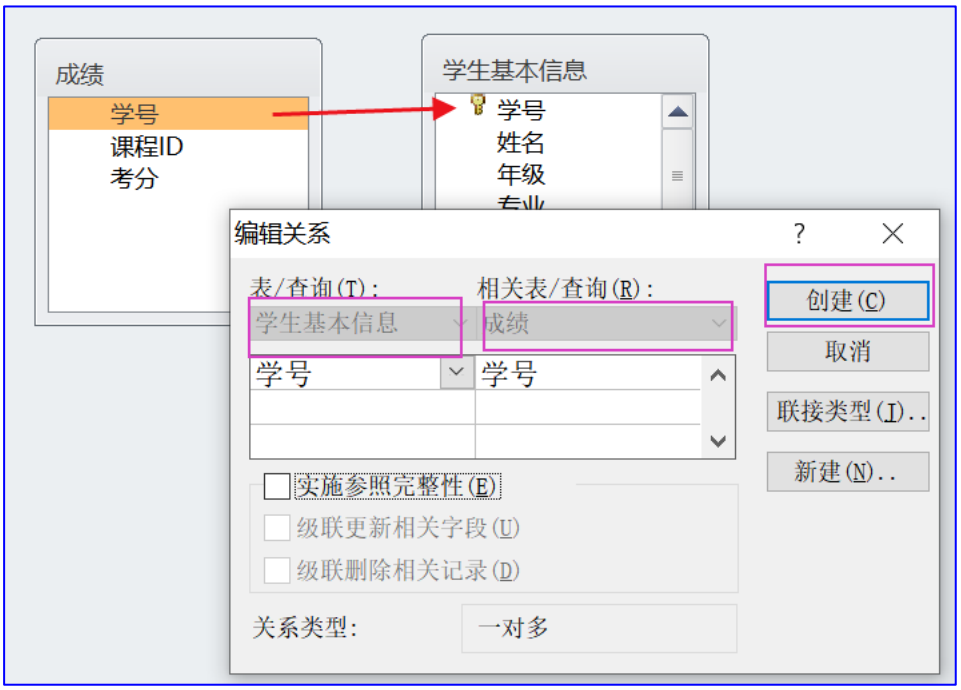
② 数 据 库 表 间 关 系

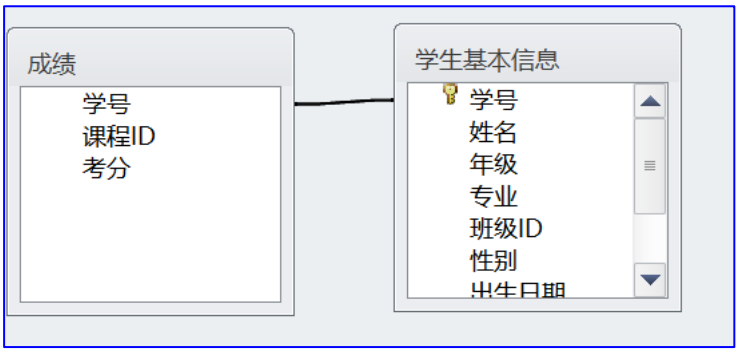
- 可以直接视图建立表之间的关联
- 表“成绩”和“学生基本信息”是可以关联的两个表，可以如下操作：

- 拉放量表到工作区域



- 选中一个表中的任一标签拖动到另一个表即可出现编辑关系页面，选择两个表中相同的标签，即可成功创建它们之间的关系，如下图,选择结束之后单击确定即可





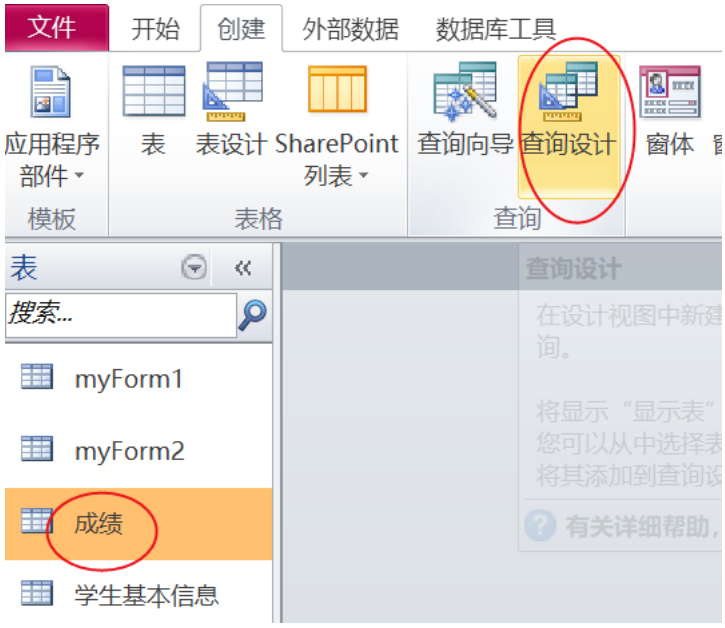
③ 查 询 （ SQL ）

- 要点说明：

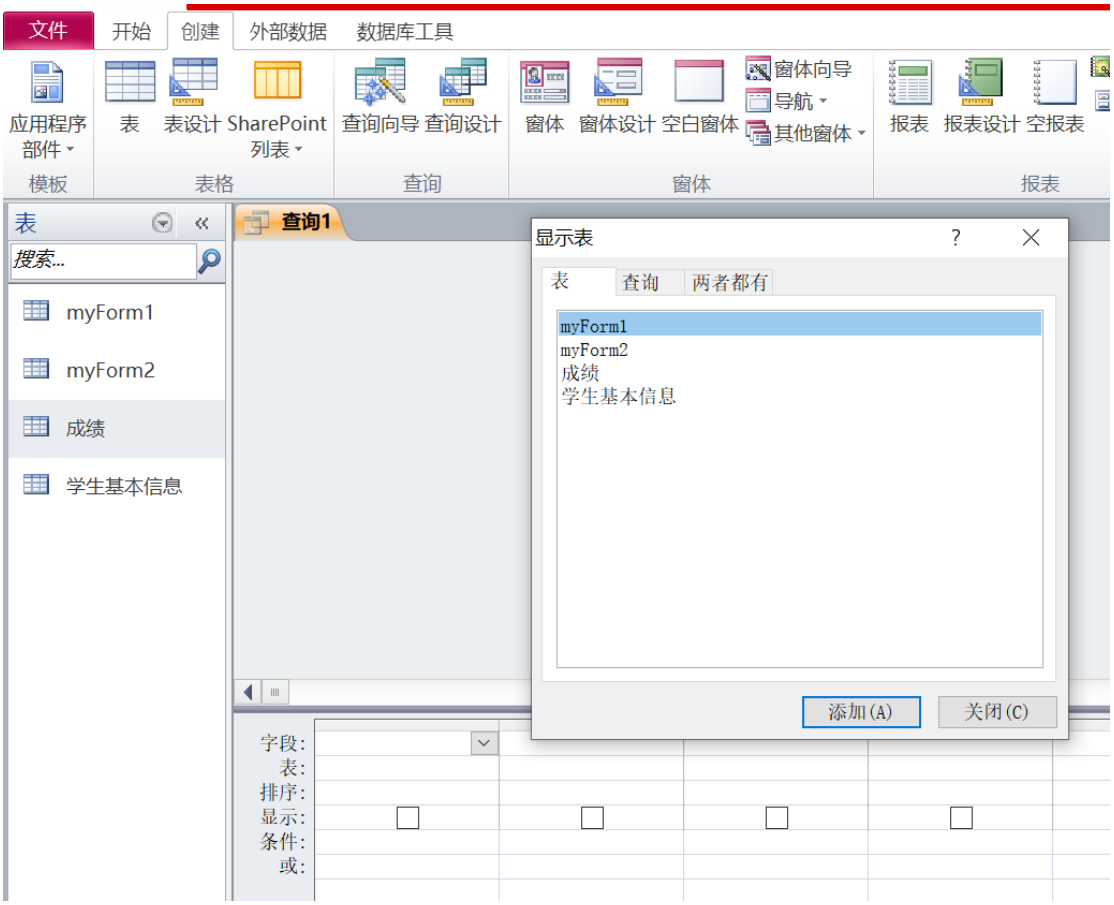
查询是 Access 的主要计算手段。单机数据库的核心，其实就是对表的查询。查询的操作，在 Access 中是可视化的，这也是 Access 数据库的优势之一。

查询是以表或查询为数据源的再生表。查询的运行结果是一个动态数据集合

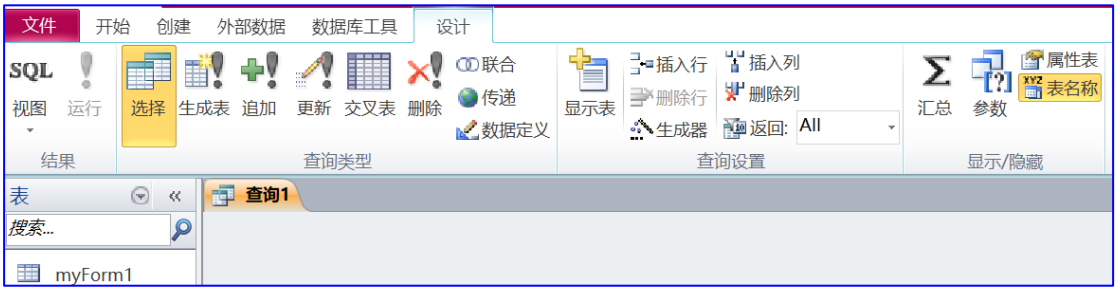
- 可视化查询：
- 以我们上述例子为基础执行查询操作：


文件
开始
创建
外部数据
数据库工具
应用程序部件
表
表设计
SharePoint列表
查询向导
查询设计
窗体
模板
表格
查询
表
搜索...
myForm1
myForm2
成绩
学生基本信息
查询设计

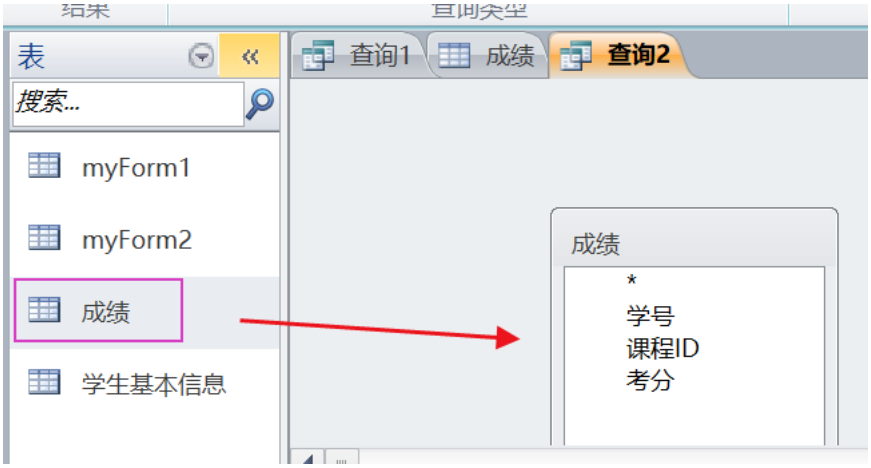
- 打开可见：


文件
开始
创建
外部数据
数据库工具
应用程序部件
表
表设计
SharePoint列表
查询向导
查询设计
窗体
窗体设计
空白窗体
窗体向导
导航
其他窗体
报表
报表设计
空报表
模板
表格
查询
窗体
报表
表
搜索...
myForm1
myForm2
成绩
学生基本信息
查询1
显示表
表
查询
两者都有
myForm1
myForm2
成绩
学生基本信息
添加(A)
关闭(C)
字段:
表:
排序:
显示:
条件:
或:

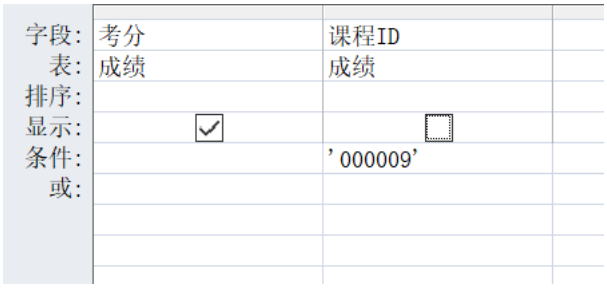
- 关闭之后可见：



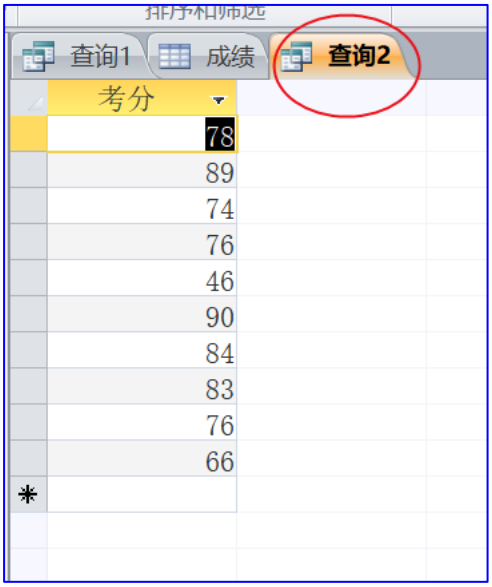
- 例如查询上述表中课程 ID 是“000009”的所有同学的成绩：
- 拉放你需要用到的表，如这里是“成绩”



- 对具体需要呈现的结果执行设置：



- 保存好查询，执行运行，结果如下：

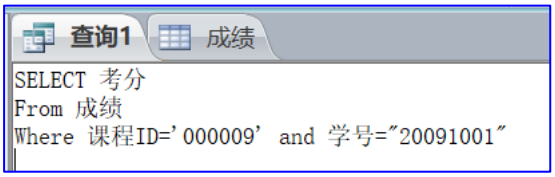


- SQL 语句查询：

- 例如查询上述表中某个圈中同学的成绩：

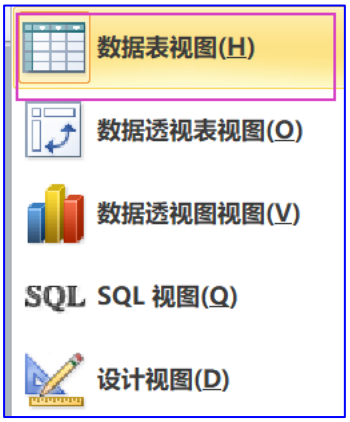
- 首选，需要转换成语句查询视图：

- 在该视图中输入你的 SQL 语句——请翻阅自己以前学习的数据库语句：



- 保存查询并执行，请结果如图：

（这个是我们查询所要的结果！）（这也是“↓”）



（用户可以依据需求，选择其他的显示形式）

-
- 高级查询（关联表的查询），子句需要参考你们以前学过的 SQL 语句
-

④ 利用查询完成的其他更新表的操作

- **要点说明：**

  - 修改记录是数据库操作中的一项基础任务。Access 提供了多种方法来实现这一点，但无论是手动修改还是通过查询批量修改，都需要遵循一定的步骤，并注意一些关键点。
  - 步骤：

    (1) **打开表**：要修改记录，首先打开需要编辑的表。在表视图中，可以浏览和修改数据

    (2) **定位到需要修改的记录**：找到你想要修改的记录，可以使用导航按钮移动记录，或使用搜索功能快速定位

    (3) **修改数据**：在需要修改的字段上双击，输入新的数据即可完成修改

(4) 注意事项：

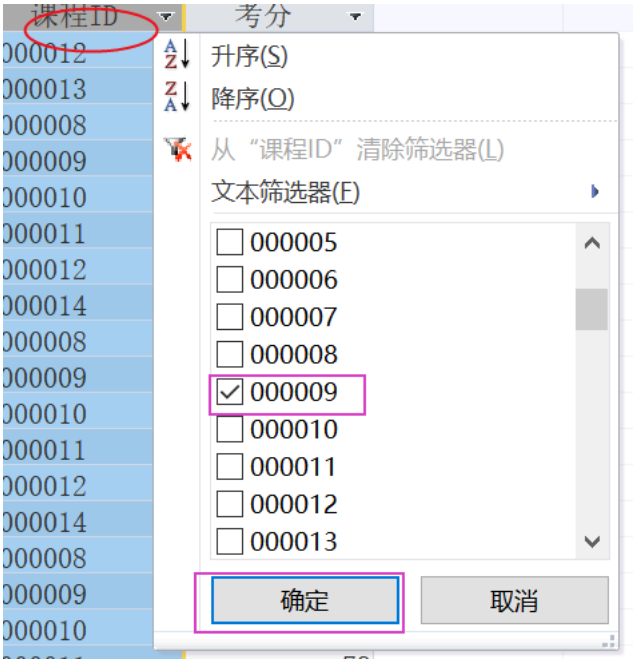
✓ - **修改前最好备份原始数据**，以防万一出现错误。

✓ - 确保所有修改都符合数据完整性规则，比如字段数据类型、约束等。

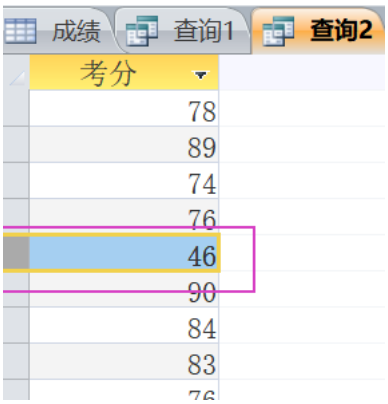
✓ - 如果修改影响到其他表的关联数据，需谨慎处理。

删除操作

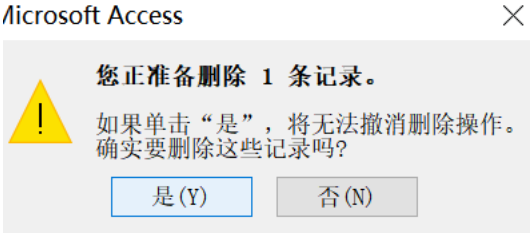
- 紧接上述“查询 2”的结果，以“数据表视图”形式打开
- 删除其中成绩为“46”和“66”两条记录
- 删除前，先简单看其中一条记录“46”如图



- 记住“46”这条记录 我们来执行删除——直接在查询视图里就可以：



选中 右击 删除



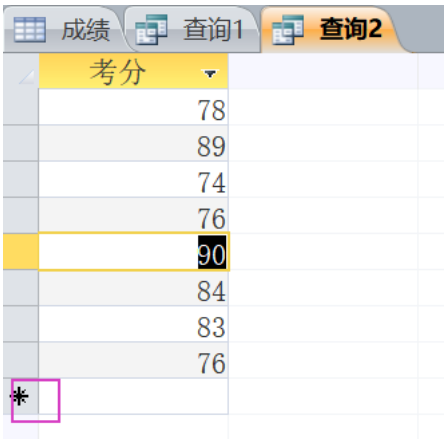
选择是

回顾成绩表

学号	课程ID	考分
20091001	000009	78
20091002	000009	89
20091003	000009	74
20091004	000009	76
#已删除的	#已删除的	#已删除的
20091006	000009	90
20091007	000009	84
20091008	000009	83

- 类似执行“66”删除后，你发现已修改数据库和查询数据结果的表了

学号	课程ID	考分
20091001	000009	78
20091002	000009	89
20091003	000009	74
20091004	000009	76
20091006	000009	90
20091007	000009	84
20091008	000009	83
20091009	000009	76



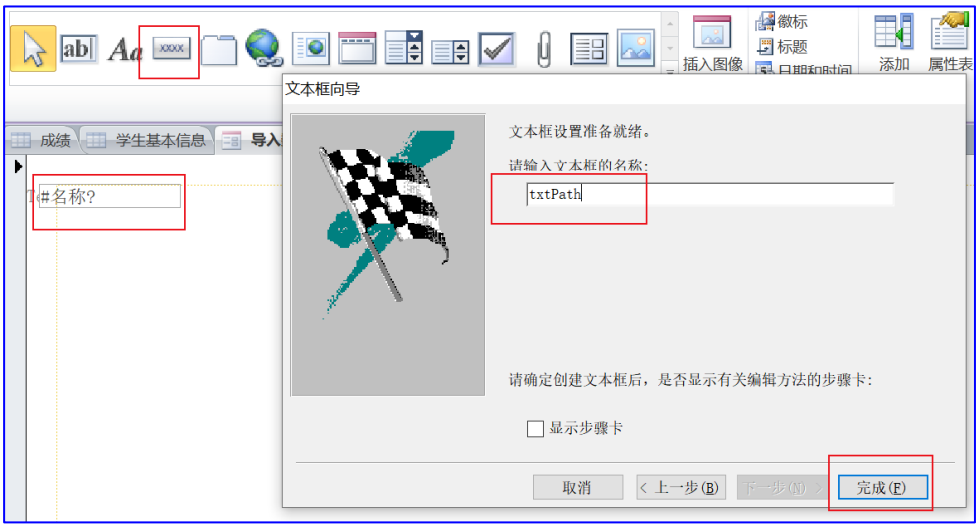
- **增加操作**
- 上图中“*”为增加，类似删除操作执行即可
-

- **修改操作**
- 类似删除操作执行即可

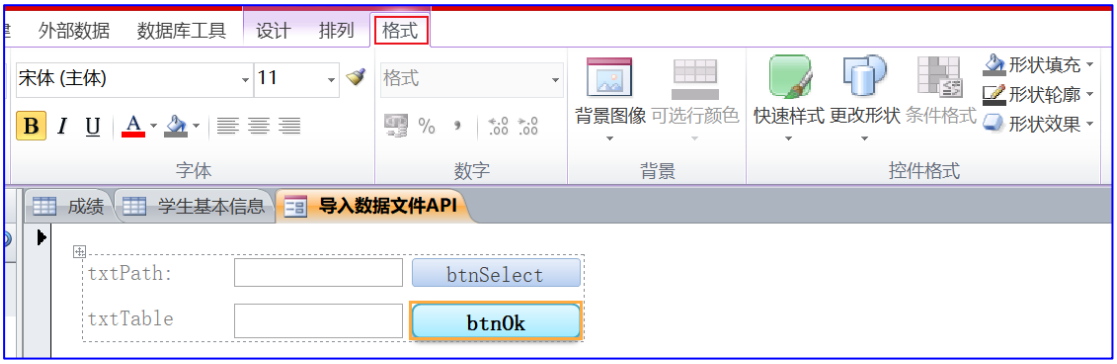
① 创 建 窗 体 ：导入&输出

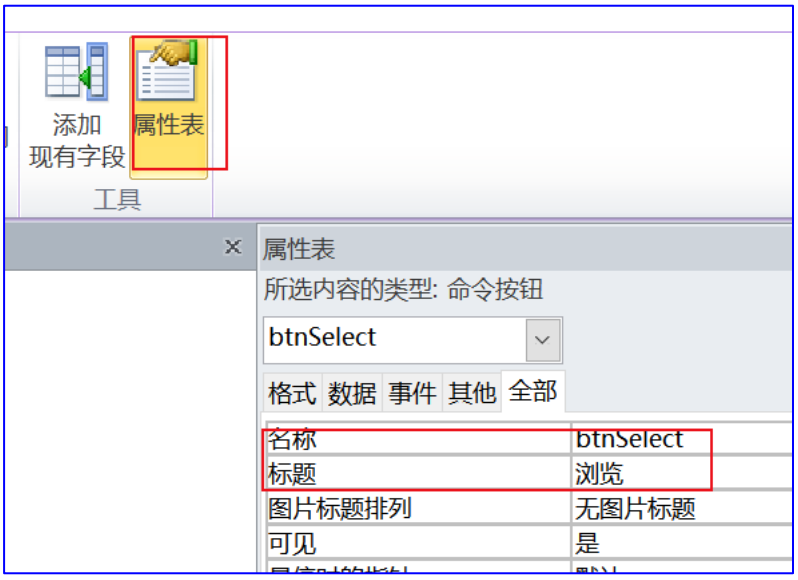
- 依据需求设计各种窗体
- 例如：添加引入文件的窗体
- 创建空白窗体：

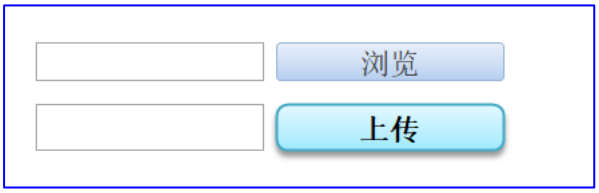
- 在窗体上放：
  两个文本框：名称分别为 txtPath、txtTable，
  再添加两个按钮，名称分别为：btnSelect、btnOK



可在对应的格式设置中设置控件的样式：

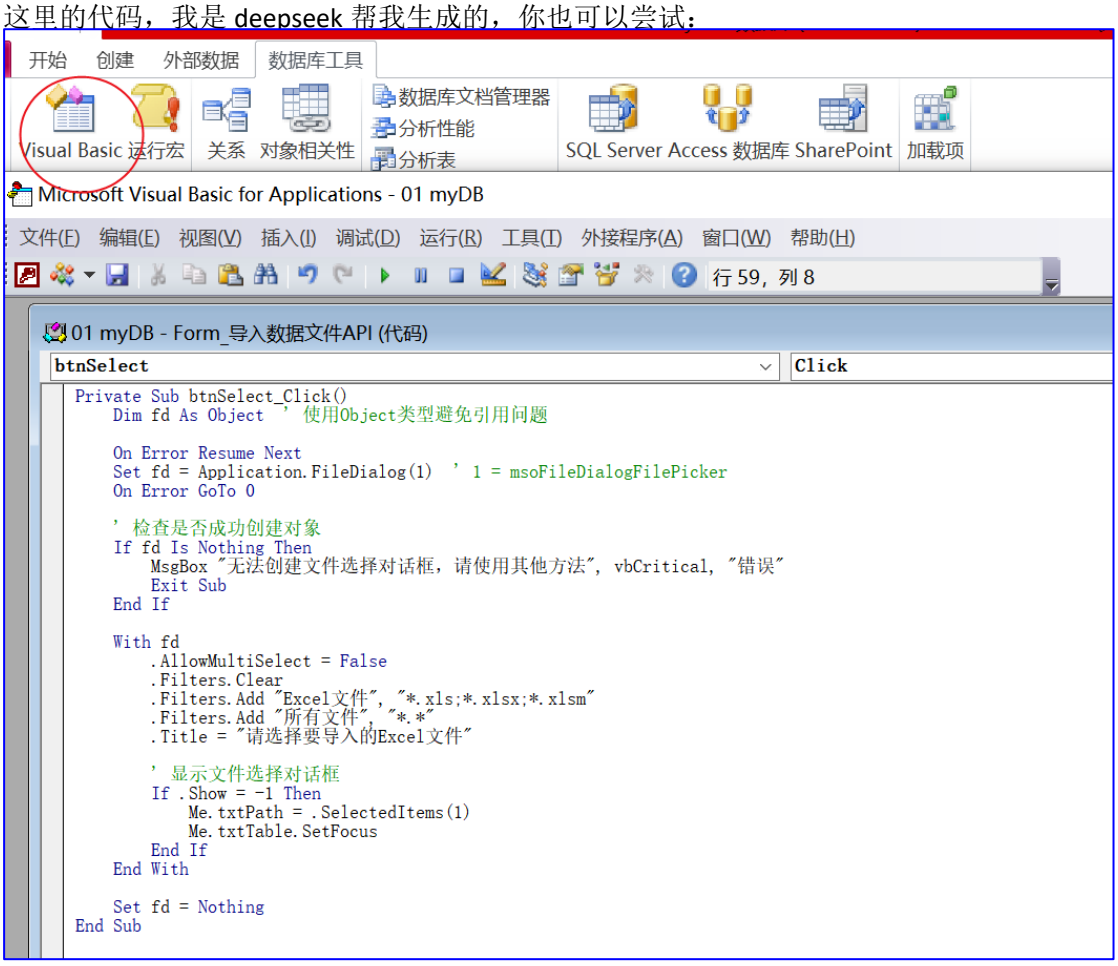


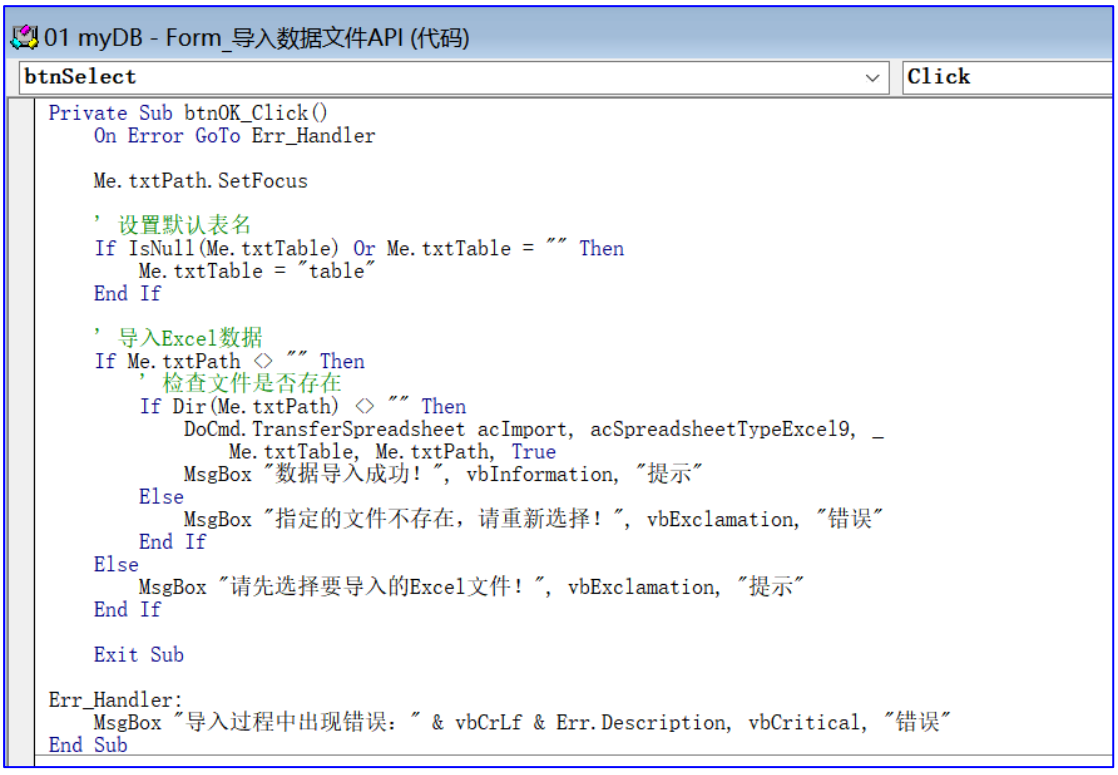




（我们要的效果）

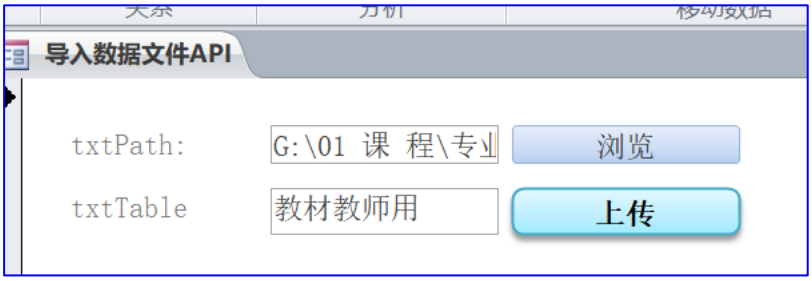
- 接下来，我们为这两个按钮编写 VBA 代码：
  这里的代码，我是 deepseek 帮我生成的，你也可以尝试：



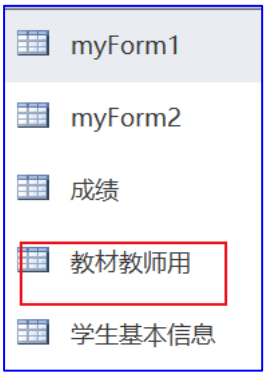


共连个按钮的自动配置 VBA 代码

- 成功运行之后，



可见，在表里多出了我们需要导入的新表，且以上述表名命名：

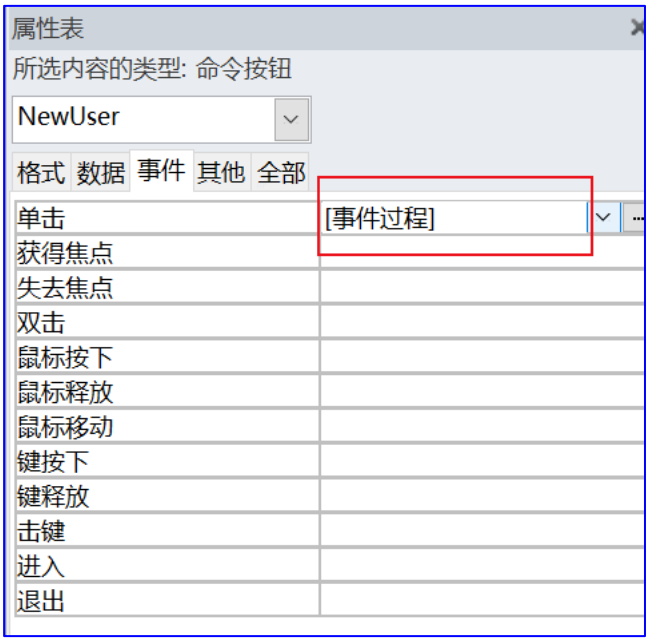


-

使用窗体为“myForm1”逐一增加记录

- 类似上述操作，增加相对应的文本框，并为其命名对应的名称“username””account”
- 新增按钮，尽管我们选择了“新添加一条记录”的宏，最后还是使用了 VBA 处理
- 大致界面如下：

- 选择按钮，增加 VBA，可以多样化处理：



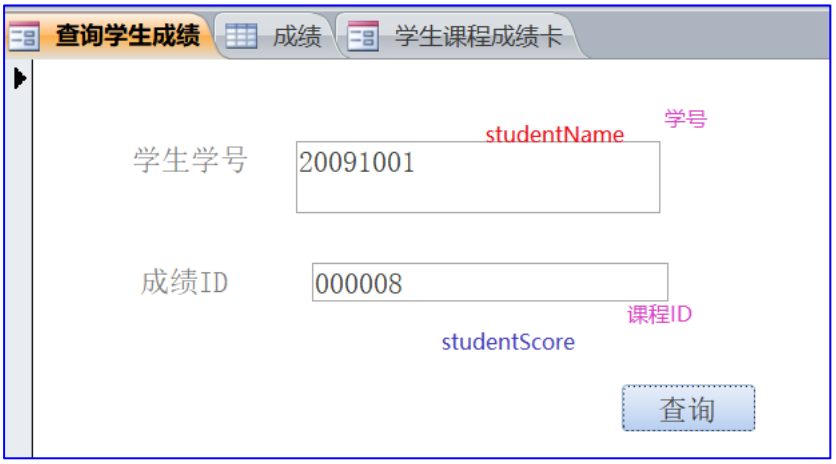
NewUser

```
Private Sub NewUser_Click()



    On Error GoTo Err_Handler

    Dim db As DAO.Database
    Dim rs As DAO.Recordset

    ' 检查输入
    If IsNull(Me.username) Or IsNull(Me.account) Then
        MsgBox "请输入用户名和账号！", vbExclamation, "提示"
        Exit Sub
    End If

    Set db = CurrentDb
    Set rs = db.OpenRecordset("myForm1", dbOpenDynaset)

    ' 添加新记录
    rs.AddNew

    ' 直接赋值，Access会自动处理数据类型转换
    rs.Fields("username").Value = Trim(Me.username)
    rs.Fields("account").Value = Trim(Me.account)

    rs.Update
    rs.Close

    MsgBox "添加成功！", vbInformation, "成功"

    ' 清空输入框
    Me.username = Null
    Me.account = Null
    Me.username.SetFocus

Exit_Here:
    Set rs = Nothing
    Set db = Nothing
    Exit Sub

Err_Handler:
    MsgBox "错误 " & Err.Number & ": " & Err.Description, vbCritical, "错误"
    Resume Exit_Here
```

- 可前往 PPT 学习什么是 DAO 哦!
-

利用窗体，查看查询或者要看见的效果

- **我在这里踩了很大的坑，总以为宏是简易操作，现在踩发现，一点不，你得对所有的宏操作及其之间的细微差别很在行，比如我直接选择 selectForRecord，左右思想不到竟在这里 bug,应该选择 applyFilter，简直了啊。。。。**
- 前面窗体基础操作参照上述步骤进行
- 界面设计大致如下：



- 各文本框对应了成绩表中的字段
- 重点是查询按钮的处理，事实证明，还是 VBA 简单！
- 效果：
  - 当输入对应学号和课程 ID 之后，我们希望看见对应查询的记录
  - 由于这个效果需要窗体显示，因此，我们必须先建立一个单独要看见我们记录的“学生课程成绩卡”
  - 在查询按钮之后弹开并显示正确的记录
  -
- 先建立“学生课程成绩卡”

  - 由于目前没有指定，所有的记录均可在此显示

主要作用是为我们一会弹开的窗体做准备

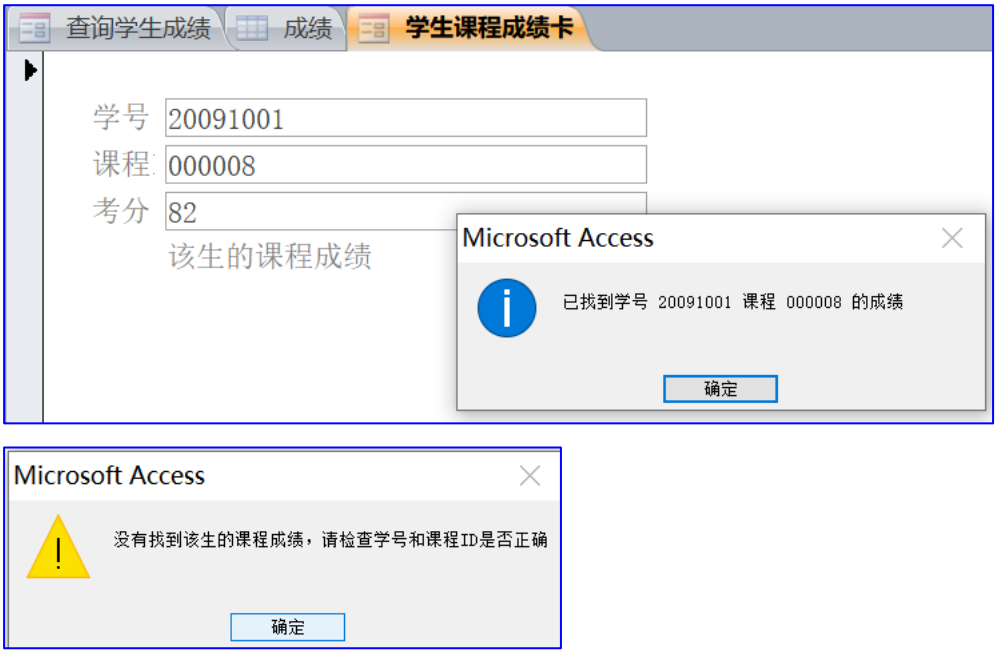
- 打开查询设置宏：

```
OpenForm
        窗体名称  学生课程成绩卡
            视图  窗体
        筛选名称
            当条件
        数据模式  只读
        窗口模式  普通
SelectObject
        对象类型  窗体
        对象名称  学生课程成绩卡
    在数据库窗口中  否
ApplyFilter
        筛选名称
            当条件  =[学号]=[Forms]![查询学生成绩]![studentName] And [课程ID]=[Forms]![查询学生成绩]![studentScore]
        控件名称
添加新操作
```

- 我们还可以利用 messagebox 和 if 来进行更细致的逻辑检查
- 具体如下：

```
If  DCount("*","成绩","[学号]='" & [Forms]![查询学生成绩]![studentName] & "' AND [课程ID]='" & [Forms]![查询学生成绩]![studentScore] & "'")=0  Then
    MessageBox
            消息  没有找到该生的课程成绩，请检查学号和课程ID是否正确
        发嘟嘟声  是
            类型  警告!
            标题
End If
If  DCount("*","成绩","[学号]='" & [Forms]![查询学生成绩]![studentName] & "' AND [课程ID]='" & [Forms]![查询学生成绩]![studentScore] & "'")>0  Then
    MessageBox
            消息  ="已找到学号 " & [Forms]![查询学生成绩]![studentName] & " 课程 " & [Forms]![查询学生成绩]![studentScore] & " 的成绩"
        发嘟嘟声  是
            类型  信息
            标题
End If
```

运行效果：


查询学生成绩
成绩
学生课程成绩卡
学号 20091001
课程 000008
考分 82
该生的课程成绩
Microsoft Access
已找到学号 20091001 课程 000008 的成绩
确定

Microsoft Access
没有找到该生的课程成绩，请检查学号和课程ID是否正确
确定

第 2 篇 Git 基础操作

课时安排：12 学时（理论+实操）

教学目标

- 知识目标

掌握 Git 本地仓库的创建与管理流程。

理解文件在 Git 工作区、暂存区、版本库的状态转换。

学会使用基础 Git 命令（init、add、commit、status、log）。

- 能力目标

能独立完成本地仓库的初始化与版本提交。

能通过 git status 和 git log 分析版本状态与历史记录。

- 素养目标

培养规范的版本提交习惯（如清晰的 commit message）。

理解版本控制对协作开发的重要性。

教学重点与难点

- 重点：

Git 三大区域（工作区、暂存区、版本库）的交互逻辑。

文件状态转换（未跟踪、已修改、已暂存、已提交）。

- 难点：

暂存区（Stage）的作用与实际意义。

git reset 与 git checkout 的区别与应用场景。

教学内容与过程

1. 课程导入

- 提问互动：

“如果误删了文件，如何通过 Git 恢复？”

“为什么 Git 需要 git add 后再 commit？”

- 案例演示：

展示一个未提交的修改如何通过 git checkout -- file 撤销。

2. 核心内容讲解

（1）Git Bash 基础操作（复习）

常用命令速览：

```
bash
cd ~            # 切换到用户根目录
mkdir repo # 创建文件夹
touch file.txt # 创建文件
ls -al          # 查看所有文件（含隐藏文件）
```

（2）Git 本地仓库管理

- 初始化仓库：

```
bash
git init                # 当前目录转为 Git 仓库
```

```
ls -a .git          # 查看生成的 Git 元数据目录
```

- 文件状态与操作：

  未跟踪 → 已暂存：git add file.txt

  已暂存 → 已提交：git commit -m "描述"

  状态查询：git status（关键输出解析）

- 版本回退：

  查看历史：git log --oneline（简化输出）

  回退到指定版本：

```
bash
git reset --hard HEAD^      # 回退到上一个提交
git reset --hard <commit-id> # 回退到特定提交
```

（3）撤销与删除文件

- 撤销工作区修改：

```
bash
git checkout -- file.txt   # 丢弃未暂存的修改
```

- 删除文件并提交：

```
bash
git rm file.txt             # 删除文件并暂存变更
git commit -m "删除 file.txt"
```

3. 课堂实操任务

- 任务 1：本地仓库全流程练习

创建目录 git-lab 并初始化为 Git 仓库。

新建 README.md 文件，写入内容并提交。

修改文件后使用 git diff 查看变更，再提交新版本。

- 任务 2：版本回退实战

提交 3 次不同版本的 README.md（内容随意）。

使用 git log 查看提交历史，回退到第二个版本。

通过 git reflog 找回最新的提交记录。

4. 易错点与注意事项

警告：

git reset --hard 会永久丢弃未提交的修改！

删除文件后需 git commit 才能生效。

- 常见问题：

Q：git add .和 git add -A 有什么区别？

A：git add .仅添加当前目录变更，-A 添加所有目录变更。

课后作业

基础练习：

在本地创建仓库，完成 5 次提交并回退到第 3 次提交。

思考题：

为什么 Git 设计暂存区？直接提交到版本库是否可行？

- 教学资源

推荐工具：

Learn Git Branching（可视化练习分支管理）

■ 参考文档：

Git 官方文档

菜鸟教程-Git 基础

第 3 篇 多人协作管理 GIT

课时安排：12 学时（理论+实操）

教学目标（这部分属于多于扩展，可选，可不选）

- 知识目标

  掌握 Git 分支的创建、切换、合并与冲突解决。

  理解远程仓库（GitHub）的协作流程与 Pull Request 机制。

  熟悉 SSH Key 配置与远程仓库操作（clone、push、pull）。

- 能力目标

  能独立完成分支管理及多人协作开发任务。

  能通过 GitHub 发起和审核 Pull Request。

- 素养目标

  培养团队协作意识与代码评审习惯。

  理解开源协作的文化与规范。

教学重点与难点

- 重点：

  分支管理流程（创建、合并、冲突解决）。

  远程仓库协作（PR 流程、代码同步）。

- 难点：

  合并冲突的手动解决逻辑。

  SSH 密钥配置与权限管理。

教学内容与过程

1. 课程导入

“如果多人同时修改同一文件的同一行代码，会发生什么？如何解决？”

“为什么 GitHub 要求使用 SSH 密钥而非密码推送代码？”

案例演示：

展示 GitHub 上开源项目的 PR 页面（如 Vue.js 的合并请求）。

2. 核心内容讲解

（1）Git 分支管理

- 分支原理：

  分支是指向提交对象的可变指针，默认分支为 main/master。

  HEAD 指针指向当前所在分支。

- 常用命令：

```
bash
git branch feature-1          # 创建分支
git checkout feature-1        # 切换分支
git merge feature-1           # 合并分支到当前分支
git branch -d feature-1       # 删除分支
```

- 冲突解决：

冲突场景：多人修改同一文件的同一区域。

- 解决步骤：

git pull origin main（拉取最新代码）。

手动编辑冲突文件（保留需合并的内容）。

git add → git commit → git push。

（2）远程仓库协作（GitHub）

- SSH Key 配置：

bash

ssh-keygen -t rsa -C "email@example.com" # 生成密钥

cat ~/.ssh/id_rsa.pub # 复制公钥到 GitHub 账户

- 协作流程：

克隆仓库：git clone git@github.com:user/repo.git。

创建分支：git checkout -b feature-2。

推送分支：git push origin feature-2。

发起 PR：在 GitHub 页面操作，请求合并到 main 分支。

代码评审：团队成员评论或批准 PR。

合并分支：通过 GitHub 完成合并。

（3）多人协作冲突模拟

实操演示：

两名学生同时修改 README.md 的同一行并推送。

通过 git pull 触发冲突，演示手动解决过程。

3. 课堂实操任务

- 任务 1：分支管理实战

在本地创建 dev 分支并修改文件。

切换回 main 分支，合并 dev 分支。

故意制造冲突并解决。

- 任务 2：GitHub 协作练习

分小组克隆教师提供的 GitHub 仓库。

每组创建分支并提交修改，发起 PR 请求合并。

其他小组审核 PR 并评论。

4. 易错点与注意事项

- 提示：

未提交的更改会导致分支切换失败（需 git stash 暂存）。

强制推送（git push -f）可能覆盖他人代码，慎用！

- 常见问题：

Q：PR 合并后如何同步本地仓库？

A：git pull origin main 更新本地 main 分支。

- 课后作业

基础练习：

在 GitHub 上 Fork 一个开源项目（如 https://github.com/ freeCodeCamp/ freeCodeCamp ），创建分支并提交修改。

- 思考题：

为什么企业开发中常用 feature 分支而非直接修改 main 分支？