第四周学习笔记​

一、计算机与编程基础​

（一）硬件与数据存储​

计算机依靠硬件与软件协同工作。磁盘、磁带可在不通电时长期存储数据，CPU 和内存则需通电运行。在金融领域，稳定的长期存储对交易数据保存意义重大，为后续分析和审计筑牢基础。​

（二）操作系统与进程管理​

操作系统负责管理 CPU 和内存，进程是其执行任务的基本单元，各进程相互隔离。以微软 Word 和 Python 解释器为例，它们启动后均作为进程占用系统资源。在金融编程里，合理管理多个数据处理进程，能够提升系统运行效率，防止资源冲突。​

（三）Python 解释器工作原理​

Python 解释器将用户代码翻译为二进制指令，调用操作系统或其他依赖项执行操作。在金融编程中，我们聚焦代码逻辑，利用 Python 实现数据获取、分析等自动化工作流程。​

二、Python 编程实操​

（一）Python 基础概念​

变量用于存储数据，函数封装特定功能，对象则集成数据与相关操作。在金融编程场景下，变量可存储金融数据，函数可实现金融计算，助力构建复杂金融模型。​

（二）VSCode 环境配置与文件操作​

掌握在 VSCode 中新建项目、目录和文件，以及在命令行进行文件操作的方法，对整理和分析金融数据文件极为关键。​

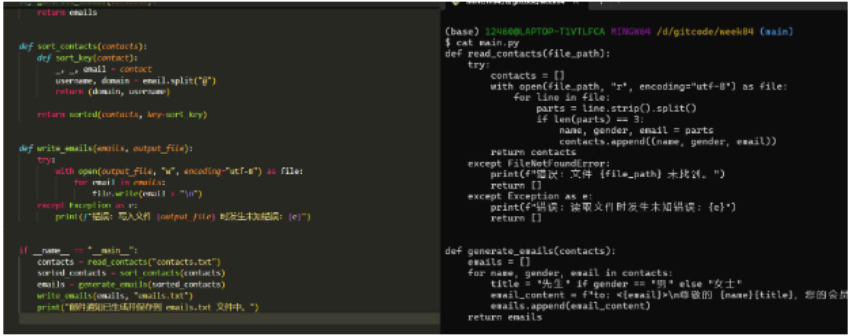
三、数据处理与 AI 编程​

（一）数据处理项目实践​

通过编写 Python 程序读取、处理文件内容，并按规则排序输出，理解编程在数据读取、计算和写入方面的原理。在金融场景中，对交易数据进行排序分析，有助于投资者发掘交易规律。​

（二）AI 编程与代码调试​

AI 编程借助概率模型生成代码，虽具有随机性且可能出错，但可通过逐行调试检查运行情况。在金融编程中，AI 可辅助生成复杂代码，提升编程效率，但开发者仍需具备扎实知识，通过调试确保代码准确。​



四、Python 扩展与调试工具​

（一）AI 生成代码与 Python 扩展实操​

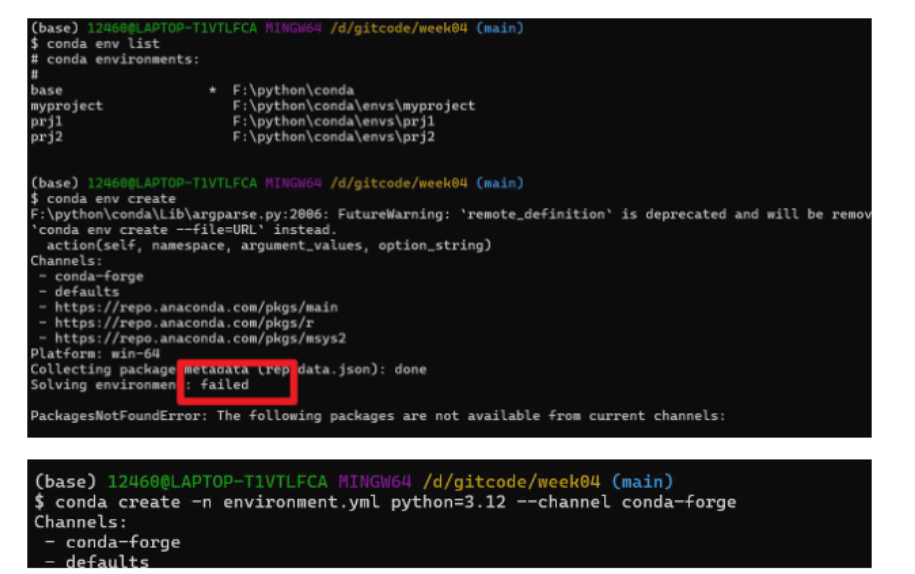
学会保存、修改 AI 生成的代码，掌握微软官方 Python 扩展及第三方 Rough 软件的安装、使用和配置。技术工具的多样性和可配置性，能够提高编程效率，打造个性化编程环境。​

（二）图形化界面与网络问题处理​

修改配置文件可指定 Python 解释器，Rough 插件能规范代码。实际操作时需留意网络问题，确保工具稳定运行。​

（三）代码规范化与运行效果验证​

Rough 扩展可检查和修复语法错误，自动规范代码。掌握在命令行和 VSCode 中切换工作目录的方法，对运行和测试代码十分重要 。规范的代码能提升可读性与运行效率，减少潜在错误。​



五、AI 时代的编程思考​

（一）AI 时代人类的编程定位​

AI 降低了编程门槛，但人类学习编程的必要性并未降低，创新在编程中的重要性愈发凸显。解决复杂问题、赋予代码个性，是人类编程的独特价值。​

（二）人类与 AI 的协作和掌控​

人类在工作中追求原创性，不会完全依赖 AI。在 AI 辅助编程时，需保持清醒，明确自身主导地位，掌控工具发展，防范潜在风险。​

（三）AI 对编程教育的冲击​

AI 普及可能催生 “文盲程序员”，编程教育应注重培养学生对编程原理的理解，避免过度依赖 AI。​

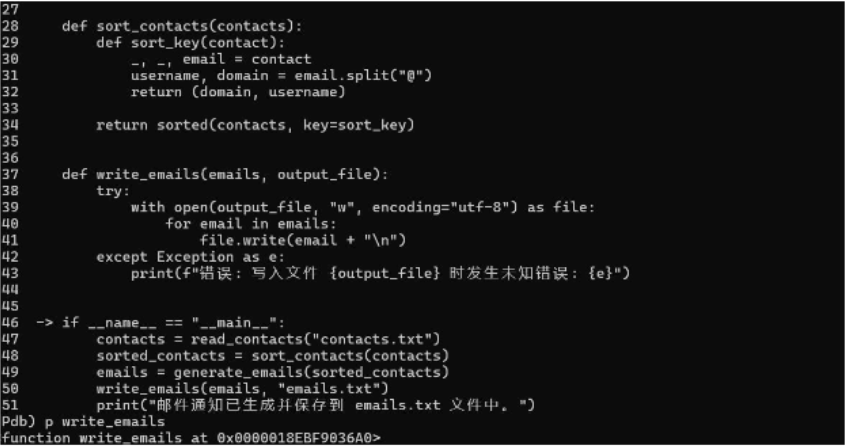
六、Python 调试进阶​

（一）调试器基础命令与代码理解​

调试器是理解和优化代码的关键工具。L、N、P 等常用命令可查看代码运行状态，帮助预判执行流程、获取关键数据，深入理解代码执行逻辑。​

（二）Python 调试工具的运用​

借助 Python 调试工具 LL，可全方位检查代码和变量。P 命令显示变量内容，S 命令进入函数内部操作，通过实例演示加深对编程概念的理解。​



（三）Python 函数运行机制及文件操作​

文件操作和函数调用时，close 函数和清理函数保障资源正确释放和程序安全。step in 等操作可深入代码内部，观察运行过程，strip、split 等函数丰富了字符串操作方法。​

（四）PDB 调试工具的实践应用​

PDB 是强大的程序调试工具，借助其进行数据处理、文件操作、函数调用和排序等操作，能够完整完成程序调试流程。​

本次学习让我认识到，Python 编程学习不仅涵盖语法知识，还需掌握工具使用，理解各类编程概念与机制。只有理论结合实践，才能提升编程能力，解决实际问题。​