“懂会计就会指导 AI 编程”

python 编程逻辑、特性和案例

—以会计的思维

四川省注册会计师协会信息化委员会

四川省注册会计师协会信息化委员会

2026年6月8日

目录

前言 1
第一部分 编程逻辑 2
一、开发工具（IDE）——您的“会计工作环境” 2
1.1 什么是 IDE？ 2
1.2 环境的核心：Python 解释器——您的“会计基础概念框架” 2
1.3 项目的管理：Project —— 您的“会计账套” 3
1.4 为什么推荐 PyCharm？ 3
1.5 【拓展知识】计算机如何工作：从物理到逻辑 4
二、对象：从“记账凭证”到“会计准则” 5
2.1 类与对象 5
2.2 属性与方法 5
2.3 进阶理解：继承与多态——以“会计科目”与“辅助核算”为例 8
2.4 进阶类比：会计准则作为一个类 12
2.5 总结：对象思维的核心 12
三、结构：程序的骨架 vs 流程的控制 13
3.1 程序的组成结构（三层架构） 13
3.2 程序的控制结构（顺序、分支、循环） 14
3.3 函数与模块：可复用的“标准作业程序(SOP)”与“功能子系统” 15
3.4、数据：编程的血液 vs 会计的凭证 16
四、算法：把会计规则写成代码 19
4.1 案例一：借款费用资本化 —— 顺序 + 分支 19
4.2 案例二：摊余成本（实际利率法） —— 循环 + 迭代 21
4.3 合并报表的“递归”本质：递归 VS 规则复用 22
第一部分小结 25
第二部分 特性 —— 会计人员的“工具箱” 26
一、Flask —— “自定义的财务数据接口” 26
二、pandas / polars —— “财务数据工作台”/存储 30
三、numpy / scipy —— “财务计算器与统计工具箱” 33
四、 matplotlib / seaborn / pyvis 38
—— “财务图表与关系可视化” 38
五、networkx —— “会计科目网络与勾稽关系” 48
六、scikit-learn —— “财务预测与风险评分” 50
七、beautifulsoup —— “财务数据自动采集器” 72
八、all-minilm-l6-v2 —— “财务文本语义理解器” 73
第二部分小结 75
第三部分 案例演示—从代码到应用 76
1. 大规模数据分析 76
1.1 项目文件结构 76
1.2 项目功能概览 77
1.3 代码运行逻辑简述 77
2. 拖拽映射 79

四川省注册会计师协会信息化委员会

2.1 项目文件结构 79
2.2 项目功能概览 79
2.3 代码运行逻辑简述 80
3. Excel 公式链接图谱 82
3.1 项目文件结构 82
3.2 项目功能概览 82
3.3 代码运行逻辑简述 83
4. XML 解析（标准会计信息系统数据） 85
4.1 项目文件结构 85
4.2 项目功能概览 85
4.3 代码运行逻辑简述 87
5、 语义对齐（基于 nlp） 89
5.1 项目文件结构 89
5.2 项目功能总览 90
5.3 代码运行逻辑简述 91
附录：数据安全、伦理与职业准则 93
一、核心原则 93
二、具体场景下的准则 93
三、总结：做一个负责任的技术赋能者 94

四川省注册会计师协会信息化委员会

前言

本书基于会计内禀的可计算性，用会计概念类比编程概念，以此阐述计算思维。会计的可计算性可从静态、动态两个维度展开：静态维度为概念+数据，动态维度为概念+计算。其中，会计概念，指会计基本准则、具体准则，以及围绕会计要素、会计方法、会计原则所形成的相关定义，同时包含凭证、账簿、报表的规范结构；会计数据，是以符合会计概念定义的形式呈现的原始业务数据；会计计算，则指会计政策与会计估计的执行。

会计的可计算性，是构建可解释、可信赖的会计智能化系统的基础性条件。凡是需稳健地引入人工智能的领域，首先需具备可计算性；但可计算性不等同于智能化，也绝非智能化的全部。会计领域从可计算性通往智能化的核心难点，在于会计概念与明确的计算规则之间，须建立完整、可追溯、可重复的推理链条。而在这一过程中我们会发现，能够以计算规则（形式化规则）方式精准表达的会计概念极为有限，大量内容都属于模糊化表达。诸如“很可能”“实质上”这类措辞，本身就带有模糊属性，是基于经验判断与归纳总结形成的专业表述。会计计算，正是在“可形式化规则”与“需判断输入”的交织中展开的——当我们尝试将“很可能”等职业判断设定为阈值或触发条件时，计算模型本身便已确立并可自动执行，但阈值的触发判断、条件语义边界的划定与校准，仍属于需要专业经验输入的主观领域。

会计概念形式化的困境，并非今日才被察觉。早在20世纪60年代，理查德·马特西奇(Richard Mattessich)在《会计与分析方法》（Accounting and Analytical Methods 1964/1977）中，提出将会计作为一门管理科学、运用形式化与公理化方法构建会计一般理论的宏大构想。马特西奇的探索揭示了会计形式化的根本难题——即便在最基本的会计要素层面，完全的形式化表达也面临巨大挑战。半个多世纪过去，这一困境依然横亘在我们面前。完成概念到计算规则之推理链条的过程，恰恰需要人工智能大模型的介入与支撑，帮助我们梳理概念层级、量度模糊地带、辅助设定规则边界。这与“结论-证据”的追溯式推理不同，后者是审计与合规验证的工作重心，而前者，是会计智能化所需倾力攻克的核心命题之一。

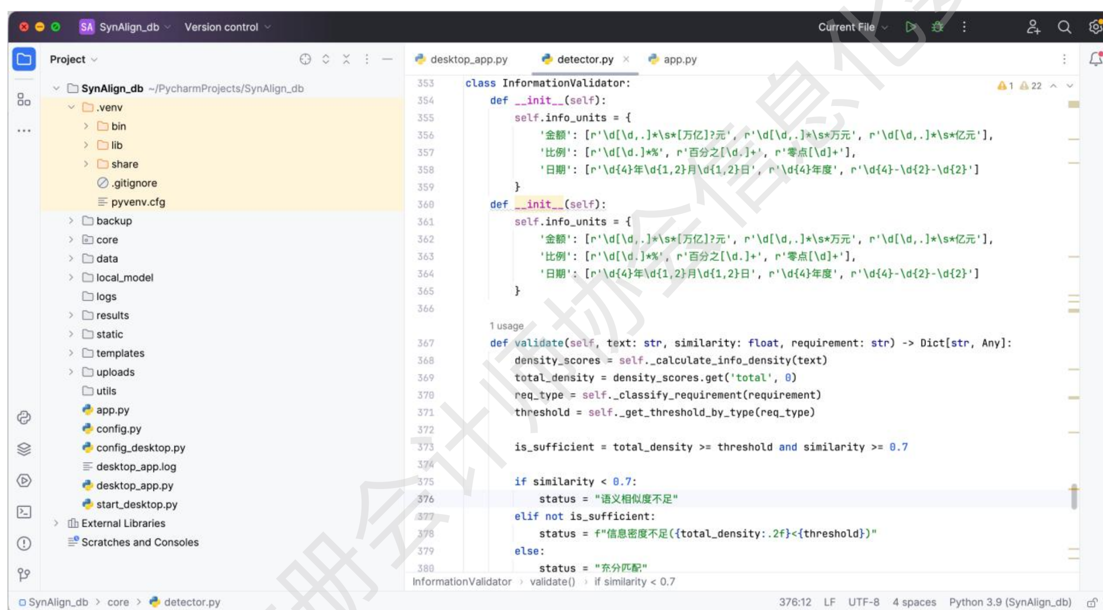
本书尝试在从概念、数据、计算三位一体的会计体系中，将”计算”这一维度予以突出与强化，重点凸显会计内禀可计算性中的计算本质。本书的写作初衷，是希望会计从业人员，能够将部分精力从单纯的会计概念理解与应用中分流出来，主动关注“会计概念-计算规则”推理链条的构建工作，将这一内容纳入自身的专业视野与执业思考范畴之中。

本书所有文字和案例都经 DeepSeek 和元宝润色，所有代码都由 DeepSeek 和元宝创作。

第一部分 编程逻辑

> 目标：让会计人员理解 Python 编程的核心逻辑，并用熟悉的会计场景对应起来，消除对代码的陌生感。

一、开发工具（IDE）——您的“会计工作环境”



1.1 什么是 IDE?

IDE 是集成开发环境，是您编写、运行和调试代码的软件平台。它好比财务人员的一体化工作台：这个工作台上既有画好的记账凭证模板（代码编辑器），也有计算器（运行环境），还有帮助你检查分录平衡的审核工具（调试器）。

1.2 环境的核心：Python 解释器——您的“会计基础概念框架”

• IDE 提供了编程的界面，但真正理解和执行您代码的，是 Python 解释器。您必须确保在 IDE 中配置了正确的解释器。

四川省注册会计师协会信息化委员会

• 会计类比：

◦ Python 解释器的作用，相当于整个会计工作的最底层基础概念框架，即会计恒等式（资产=负债+所有者权益、借贷必相等）和会计要素（资产、负债等）以及由此衍生的会计科目表、凭证结构（借贷、科目、金额）。

◦ 正如没有这个框架，任何凭证和报表都无从谈起；没有 Python 解释器，您写的任何代码都无法运行。解释器定义了什么是“资产”（整数、字符串等数据类型），什么是“借贷规则”（语法规则），以及如何“过账”（执行运算）。

◦ 您在 IDE 中写的每一行代码，最终都由解释器根据这套底层“会计基本准则”来理解和执行。

1.3 项目的管理：Project —— 您的“会计账套”

• 在 IDE 中，代码通常以“项目（Project）”为单位进行组织和管理。

• 会计类比：

◦ 一个 Project，就相当于财务软件中的一个“账套”。

◦ 一个独立的会计主体（如一家公司）就需要一个独立的账套，以确保数据清晰、规则独立。同样，一个独立的业务应用或分析任务，也应该创建一个独立的 Python Project。

◦ 在同一个 IDE 中，您可以像切换账套一样，轻松地在多个项目之间切换，每个项目都有自己独立的代码文件、数据资源和运行环境，互不干扰。

1.4 为什么推荐 PyCharm？

常见 Python IDE：
- 文本编辑器（记事本）→ 手工记账，不推荐
- IDLE（Python 自带）→ 简易计算器，适合练习
- Anaconda（集大成者）→ 一个装好各种工具的“财务工具包”
- Jupyter Notebook → 类似会计工作底稿，可以一边写说明、一边算数、一边看结果
- PyCharm → 专业的“财务自动核算模板设计平台”，推荐使用

- 自动补全代码（像财务软件中输入科目代码自动带出科目名称）
- 调试功能（像追查一笔错账，一步步看数字变化）
- 项目管理（一个会计主体对应一个 Python 工程）

四川省注册会计师协会信息化委员会

会计类比：
> IDE 好比你在设计一套自动化的会计核算模板（工作环境）——不需要每次重新写公式，只需定义好规则，以后每个会计期间都可以复用。

1.5 【拓展知识】计算机如何工作：从物理到逻辑

更深入理解计算机（包括 Python 解释器）执行规则的底层逻辑，可以了解这个最基础的原理链条：
1. 物理基础：计算机硬件基于电路，用电压的高低来代表 0（低于 0.7V）和 1（高于 3.5V）。这是所有数据的最终形态。
2. 逻辑实现：通过晶体管可以组成“与非门”等基本逻辑电路。这些门电路像一个个简单的“逻辑开关”，通过组合可以实现“与”、“或”、“非”等所有基本逻辑动作。
3. 数学理论：这些逻辑动作的规则，其数学基础是 “布尔代数” （Boolean Algebra）。布尔代数处理的就是“真”（True，可对应高电压/1）和“假”（False，可对应低电压/0）的逻辑运算。

• 终极类比：您所写的会计规则（如“if 费用 > 预算 then 标记为异常”）和 Python 代码，最终都会被转化为基于布尔代数的逻辑操作，通过数以亿计的微小“逻辑开关”（门电路）的协同工作来完成。这就像会计的记账规则，无论多复杂，最终都建立在“借贷必相等”这个最基础、最底层的恒等式逻辑之上。理解这一点，有助于您明白计算机执行规则的严谨性和本质。

会计类比总结：
IDE（如 PyCharm）是您设计自动化核算模板（会计信息系统）的工作台。在这个工作台上，您需要基于 Python 解释器（会计基础框架） 的规则来编写代码，并将一个完整的自动化任务放在一个独立的 Project（账套） 中进行管理。而这一切自动化执行的物理和逻辑根源，在于计算机严谨的二进制和布尔代数世界，这与会计工作建立在严谨的恒等式和逻辑之上异曲同工。

四川省注册会计师协会信息化委员会

二、对象：从“记账凭证”到“会计准则”

2.1 类与对象

- 类：一张空白的记账凭证模板
　定义了所有凭证共有的属性（日期、摘要、借方科目、贷方科目、金额）和方法（审核、过账、冲销）。
- 对象：一张具体的记账凭证
　例如 “2025－01－10 支付办公费 500 元”，它有具体的属性值，也能执行“审核”这个动作。

2.2 属性与方法

- 属性：描述对象是什么（有哪些数据）。
　凭证的属性：`日期 = “2025－01－10”`，`金额 = 500`，`借方科目 = “管理费用”`。
- 方法：对象能做什么（有哪些操作）。
　凭证的方法：`凭证.审核()`，`凭证.过账()`，`凭证.冲销()`。

举例：

```
from datetime import date


class Voucher:
    """记账凭证类（模板）"""
    def __init__(self, date: date, description: str, debit_account: str, credit_account: str, amount: float):
        self.date = date
        self.description = description
        self.debit_account = debit_account
        self.credit_account = credit_account
        self.amount = amount
        self._is_audited = False          是否已审核
        self._is_posted = False           是否已过账
        self._is_reversed = False         是否已冲销


    def audit(self):
        """审核凭证"""
        if self._is_audited:
            print("凭证已经审核过了，无需重复审核。")
        else:
            self._is_audited = True
            print(f"凭证 {self.date} {self.description} 审核通过。")
```


四川省注册会计师协会信息化委员会

```
    def post(self):
        """过账"""
        if not self._is_audited:
            print("凭证尚未审核，不能过账。")
        elif self._is_posted:
            print("凭证已经过账，不能重复过账。")
        elif self._is_reversed:
            print("凭证已被冲销，不能过账。")
        else:
            self._is_posted = True
            print(f"凭证 {self.date} {self.description} 已过账至总账。")

    def reverse(self):
        """冲销"""
        if self._is_reversed:
            print("凭证已经冲销，不能再次冲销。")
        elif not self._is_posted:
            print("凭证尚未过账，不能冲销。")
        else:
            self._is_reversed = True
            print(f"凭证 {self.date} {self.description} 已冲销。")

    def __str__(self):
        """打印凭证信息"""
        status = []
        if self._is_audited:
            status.append("已审核")
        if self._is_posted:
            status.append("已过账")
        if self._is_reversed:
            status.append("已冲销")
        status_str = "、".join(status) if status else "未审核"
        return (f"记账凭证\n 日期：{self.date}\n 摘要：{self.description}\n"
                f"借方：{self.debit_account}   {self.amount:.2f}\n"
                f"贷方：{self.credit_account}   {self.amount:.2f}\n"
                f"状态：{status_str}")


 示例：创建一张具体的凭证对象
voucher1 = Voucher(
    date=date(2025, 1, 10),
    description="支付办公费",
    debit_account="管理费用",
```

四川省注册会计师协会信息化委员会

```
    credit_account="银行存款",
    amount=500.00
)

print(voucher1)
print("-"    30)

voucher1.audit()      审核
voucher1.post()        过账
voucher1.reverse()   冲销（前提是已过账）
```

四川省注册会计师协会信息化委员会

2.3 进阶理解：继承与多态——以“会计科目”与“辅助核算”为例

在掌握了基本的“类”和“对象”后，面向对象编程还有两大强大特性：“继承”和“多态”。它们能帮你像设计一个灵活的财务核算系统一样，来组织你的代码。

财务软件中的会计科目体系和辅助核算功能：
• 情景：你需要核算“应收账款”和“主营业务成本”。两者都是会计科目，但“应收账款”需要按客户核算，而“主营业务成本”可能需要按部门和项目核算。

1. 继承：建立科目的“家族树”

• 父类（基类）：定义一个通用的 会计科目类，包含所有科目都有的共同属性（科目代码、科目名称、余额）和方法（计算余额()）。

• 子类：通过继承，我们可以创建更具体的科目类，它们自动拥有父类的全部特征，并可以增加自己特有的特征。

  ◦ 往来类科目 继承自 会计科目，并新增一个辅助核算_客户属性。

  ◦ 成本类科目 继承自 会计科目，并新增辅助核算_部门和辅助核算_项目属性。

会计类比：这就像在科目表中，你为“应收账款”这个一级科目挂接了“客户辅助核算”，为“主营业务成本”挂接了“部门”和“项目”辅助核算。子类继承了父类（科目）的全部基础信息（代码、名称），并扩展了专属的核算维度。

举例

```
class AccountingSubject:
    """父类：会计科目（所有科目的通用模板）"""

    def __init__(self, code: str, name: str, balance: float = 0.0):
        self.code = code             科目代码
        self.name = name              科目名称
        self.balance = balance      余额

    def calculate_balance(self):
        """计算余额（基础方法，子类可重写）"""
        return self.balance
```

四川省注册会计师协会信息化委员会

```
    def generate_ledger(self):
        """生成明细账（多态方法，子类将重写）"""
        print(f"[{self.code} {self.name}] 生成通用明细账，余额：{self.balance:.2f}")


class CurrentAccountSubject(AccountingSubject):
    """子类：往来类科目 —— 继承自会计科目，新增客户辅助核算"""

    def __init__(self, code: str, name: str, balance: float, aux_customer: str):
        调用父类构造方法，继承通用属性
        super().__init__(code, name, balance)
        self.aux_customer = aux_customer      新增属性：辅助核算_客户

    def generate_ledger(self):
        """重写父类方法：按客户生成往来明细账"""
        print(f"[{self.code} {self.name}] 按客户「{self.aux_customer}」生成往来明细账，余额：{self.balance:.2f}")


class CostSubject(AccountingSubject):
    """子类：成本类科目 —— 继承自会计科目，新增部门和项目辅助核算"""

    def __init__(self, code: str, name: str, balance: float, aux_department: str, aux_project: str):
        super().__init__(code, name, balance)
        self.aux_department = aux_department      新增属性：辅助核算_部门
        self.aux_project = aux_project            新增属性：辅助核算_项目

    def generate_ledger(self):
        """重写父类方法：按部门和项目生成多维度成本明细账"""
        print(f"[{self.code} {self.name}] 按部门「{self.aux_department}」、项目「{self.aux_project}」生成成本
明细账，余额：{self.balance:.2f}")


 - 测试继承效果 -
if __name__ == "__main__":
    创建子类对象（具体科目实例）
    receivable = CurrentAccountSubject("1122", "应收账款", 15000.00, "客户 A")
    main_cost = CostSubject("6401", "主营业务成本", 8000.00, "销售部", "产品 X")

    print("=== 继承验证 ===")
    print(f"科目：{receivable.code} {receivable.name}")
    print(f"  余额：{receivable.calculate_balance():.2f}")
    print(f"  辅助核算_客户：{receivable.aux_customer}")

    print(f"\n 科目：{main_cost.code} {main_cost.name}")
```

四川省注册会计师协会信息化委员会

```
print(f"    余额：{main_cost.calculate_balance():.2f}")
print(f"    辅助核算_部门：{main_cost.aux_department}")
print(f"    辅助核算_项目：{main_cost.aux_project}")
```

2. 多态：同一指令，不同结果

“多态”意味着“多种形态”。它允许你使用同一个指令，但不同的对象会根据其自身类型做出不同的响应。多态的实现依赖于在继承体系中，子类对父类方法进行“重写”，从而提供各自特定的实现。

• 我们为会计科目类定义一个叫做生成明细账()的方法。

• 当这个方法在不同的子类对象上被调用时，会产生不同的行为：

◦ 对 往来类科目对象（如应收账款-客户 A）调用.生成明细账()，它自动按客户生成往来明细账。

◦ 对 成本类科目对象（如主营业务成本-销售部-产品 X）调用.生成明细账()，它自动按部门和项目生成多维度成本明细账。

会计类比：在财务软件中，你点击不同科目的“明细账”按钮，系统会自动根据该科目挂接的辅助核算类型，呈现出不同格式和维度的明细账页。你发出的指令是相同的（“查看明细账”），但系统根据对象的不同类型，智能地展示了不同的结果。 这就是“多态”在业务中的体现。

举例

沿用上面定义的三个类（AccountingSubject、CurrentAccountSubject、CostSubject）

```
def demo_polymorphism():
     创建一个混合科目列表（包含不同子类的对象）
    subjects = [
        AccountingSubject("1001", "库存现金", 5000.00),                     父类对象
        CurrentAccountSubject("1122", "应收账款", 15000.00, "客户 A"),        往来类
        CostSubject("6401", "主营业务成本", 8000.00, "销售部", "产品 X"),      成本类
        CurrentAccountSubject("2202", "应付账款", 20000.00, "供应商 B"),      往来类
        CostSubject("6402", "其他业务成本", 1200.00, "研发部", "项目 Y")       成本类
    ]

    print("=== 多态演示：统一调用 generate_ledger()，不同对象产生不同行为 ===\n")
    for subj in subjects:
        subj.generate_ledger()     同一个方法名，不同的执行结果

if __name__ == "__main__":
```

四川省注册会计师协会信息化委员会

demo_polymorphism()

编程价值：通过“继承”，我们可以避免重复定义通用属性，使系统结构清晰；通过“多态”，我们可以写出更通用、更灵活的代码。例如，在遍历一堆不同科目生成报表时，你只需要统一命令它们.生成明细账()，而不需要写一堆 if...else 来判断科目类型。这极大地增强了代码的扩展性和维护性，当新增一种辅助核算类型时，只需新增一个子类，原有代码几乎无需改动。

四川省注册会计师协会信息化委员会

2.4 进阶类比：会计准则作为一个类

以《企业会计准则第 1 号——存货》举例：
- 类：准则文本（定义所有存货相关规则）
- 属性：准则编号、名称、生效日期、适用范围描述
- 方法：`确认()`（判断是否符合存货确认条件）、`计量()`（初始及后续计量规则）、`披露()`（报表附注要求）
- 对象：某企业根据该准则制定的存货核算办法实例（具体应用）

拓展思考：一旦，42 个具体会计准则被合理完整地用类来实现，企业会计工作即是实现可计算。但并不意味着会计工作可以智能化且无人化了，会计工作智能化的难点在于：会计准则仍属于概念体系；业务对象属于数据体系。一项会计处理流程必须“概念驱动　数据驱动”实现完全“耦合”，才能实现。

2.5 总结：对象思维的核心

> 编程中的对象，就是把现实世界的“事物”和“规则”封装在一起，方便调用。编程中的对象，就是把现实世界的“事物”（如凭证、科目）和“规则”（如会计准则、辅助核算逻辑）封装在一起。通过“类”来定义模板，通过“对象”来具体操作，再通过“继承”和“多态”来构建灵活、可扩展的系统。这和你用财务软件管理一个日益复杂的核算体系，思路是相通的。

四川省注册会计师协会信息化委员会

三、结构：程序的骨架 vs 流程的控制

在编程中，“结构”有两个层次，我们需要分开理解。

3.1 程序的组成结构（三层架构）

任何一个程序都由三大部分构成，就像一份会计工作底稿也有数据来源、计算规则和最终结果。

程序组成部分	会计类比	说明
输入/界面	原始凭证、业务单据（发票、合同、银行回单）	数据从哪里来
处理/逻辑	会计准则、核算办法、科目体系	怎么算、怎么判断
输出/存储	会计账簿、会计报表、会计档案	结果记录在哪里

> “你写一个 Python 程序，就是告诉计算机：从哪儿拿数据（输入），按什么规则算（处理），最后把结果放哪儿（输出）。这和会计人员先收集原始凭证，再按准则记账，最后生成报表完全一样。”

四川省注册会计师协会信息化委员会

3.2 程序的控制结构（顺序、分支、循环）

这就是我们常说的“算法”的基础。
- 顺序：一步一步执行（先确认收入，再结转成本）
- 分支：根据条件判断走哪条路（if - else，如资本化条件是否满足）
- 循环：重复做某件事（for / while，如逐期计算摊余成本）

> 这两类“结构”并不冲突：组成结构是程序的“骨架”，控制结构是骨架里的“动作逻辑”。

> 程序的控制结构就是业务处理的“动作逻辑”，是算法的基础。

- 顺序：一步一步顺序执行，如录入凭证 -> 审核凭证 -> 过账。
- 分支 (if/elif/else)：根据条件判断走哪条路，如同会计决策树。
  - 场景：if 费用金额 > 5000: 送总监审批 else: 送经理审批
  - 场景：if 存货计价方法 == "先进先出": 用 FIFO 法计算
- 循环 (for/while)：重复执行某件事，实现批量作业。
  - for 循环 (确定次数)：for 资产 in 固定资产列表: 计提折旧(资产)
  - while 循环 (满足条件则持续)：while 试算不平衡: 查找并调整分录
- 异常处理 (try/except/finally)：程序的“差错更正与审计追踪”机制。

```
try:
    执行过账()    核心业务
except 科目余额不足错误:
    记录审计线索("余额异常，自动冲销")
    执行冲销()
finally:
    记录操作日志()    无论成功失败，必须留痕
```

四川省注册会计师协会信息化委员会

3.3 函数与模块：可复用的“标准作业程序(SOP)”与“功能子系统”

• 函数：封装特定功能的代码块，可重复调用。

  • 会计类比：财务部制定的《增值税计算 SOP》、《月末结账流程》。就像您之前写的资本化利息()函数，定义一次，每月调用。

  • 价值：一次定义，多处使用。修改规则只需改一处，保证政策一致性，提升代码可维护性。

• 模块：将相关函数、变量组织在一个文件中，形成独立功能包。

  • 会计类比：财务软件中的总账模块、固定资产模块、应收模块。

  • 编程体现：import pandas as pd 就像启用了软件的“超级报表分析模块”，立刻获得强大工具。

  • 核心思想：分而治之，高内聚低耦合。是构建复杂系统（如 ERP）的基石。

小结：编程结构就是用顺序、分支、循环这些控制流来描述业务规则，用函数和模块来像搭建积木一样组织代码，最终实现输入->处理->输出的自动化流程。这与设计一套权责清晰、运转高效的财务内控流程，思维完全一致。

四川省注册会计师协会信息化委员会

3.4、数据：编程的血液 vs 会计的凭证

数据是程序处理的对象，如同凭证是会计核算的起点。理解数据的组织形式和交换标准至关重要。

3.4.1 数据结构：数据的〝档案柜〞与〝表单〞

数据结构决定了数据如何存储和组织，以便高效访问和处理。常用数据结构都有其会计对应物：

• 变量：单个数据项，如 本期收入 = 1000000。如同一张记账凭证，记录一项经济业务。

• 列表/数组 (List/Array)：有序的元素集合，可增减。如同一册记账凭证，或一行日记账，按时间顺序记录多笔分录。

• 字典 (Dictionary)：键-值对的集合，通过唯一“键”快速查找“值”。如同科目余额表，通过科目代码（键）快速查到当前余额（值）。

• 数据框 (DataFrame, pandas 库)：二维表格，有行索引和列标签。这是会计人员最应掌握的结构，它直接对应 Excel 工作表、科目余额表、序时账、报表。可以方便地进行分组、透视、筛选，是财务数据分析的核心。

当我们用 pandas 定义 DataFrame 的列（columns）和数据类型（dtypes）时，我们其实就在隐式地定义一个简单的 schema。但在真正的数据库系统中，schema 的定义更加严格和明确。

会计类比：一家公司的会计科目表及其辅助核算设置，就是一套核心的财务数据 Schema。它预先定义了：

表结构：有哪些‘表’（科目表、凭证表、余额表）。

字段规范：每张表里有哪些‘字段’（科目代码、科目名称、借贷方向、金额），以及字段的类型（文本、数字、日期）。

约束关系：字段间的逻辑（如凭证的借贷方金额必须相等、明细科目余额汇总至总账科目）。

例如，在凭证表中，Schema 会强制规定每笔分录必须有‘会计期间’、‘凭证号’、‘科目代码’、‘借贷方向’、‘金额’等字段，且‘科目代码’必须存在于科目表中。这就像记账凭证必须有日期、编号、摘要、科目、金额，并且借贷必相等一样，是数据录入前就定好的、不可随意更改的规范格式。”

那么，谁来定义这个 Schema？

> - 在单个系统或部门内部，可以由开发者或分析师定义。

- 但是，当需要在不同系统、不同组织之间交换数据时（如集团合并报表、事务所审计、企

四川省注册会计师协会信息化委员会

业更换财务软件），问题就来了： 如果 A 公司用一套 Schema，B 软件用另一套 Schema，数据就无法直接“对话”。

> 这就如同，如果每家公司的会计科目表和报表格式都完全不同，注册会计师将无法进行审计，投资者也无法比较不同公司的财务状况。

> 因此，我们必须从“内部定义的 Schema”，上升到“各方公认、统一的数据标准”。

3.4.2 数据标准与数据交换：财务的〝会计准则〞与〝系统接口〞

• 数据标准 (格式与类型)：如同会计准则统一了会计要素的确认、计量和报告。

• 数据类型：编程中必须明确数据是整数(int)、小数(float)、文本(str)还是日期(datetime)。这就像会计中必须区分“金额”、“数量”、“科目名称”、“日期”字段，不能将文本当数字相加。

• 数据格式：常见如 CSV(逗号分隔值)、JSON(轻量级数据交换)、XML(可扩展标记语言，常见于财务软件导出)。选择合适格式，如同选择是用凭证、账簿还是报表来承载信息。

• 数据交换 (输入与输出)：如同财务部门与业务部门、银行、税务的系统对接。

• 读取 (输入)：用 pandas.read_csv()读取 CSV 格式的银行流水，用 xml.etree 解析从财务软件导出的 XML 凭证数据。这相当于从外部获取原始凭证。

• 写入 (输出)：用 DataFrame.to_excel()将分析结果写入 Excel 报表，用 json.dump()将数据转换为 API 接口要求的格式。这相当于生成并报送账簿报表。

• 核心价值：统一、标准的数据交换格式，是实现自动化对账、合并报表、银企直连等技术的基础。

GB/T 24589《财经信息技术 会计核算软件数据接口》是由中华人民共和国审计署提出，全国审计信息化标准化技术委员会组织制定，并由国家质量监督检验检疫总局、中国国家标准化管理委员会发布的国家标准系列。该标准的核心目的是统一和规范各类会计核算软件（包括 ERP 系统中的财务模块）的数据输出格式，从根本上解决不同软件之间数据接口不统一导致的交换困难、信息孤岛等问题，为审计、财政、税务等经济监督管理部门高效、准确地采集和分析会计数据提供了技术基础。

标准结构与最新进展

该标准系列按适用对象分为四个部分：

1. 第 1 部分：企业（最新版为 GB/T 24589.1-2024，于 2025 年 7 月 1 日实施，替代 2010 版）。

四川省注册会计师协会信息化委员会

2. 第 2 部分：行政事业单位（GB/T 24589.2-2010）。
3. 第 3 部分：总预算会计（GB/T 24589.3-2011）。
4. 第 4 部分：商业银行（GB/T 24589.4-2011）。

技术要点与影响

- 规范内容：标准详细规定了会计核算数据元素（如电子账簿、会计科目、科目余额、记账凭证、报表等）以及数据接口输出文件的内容和格式要求。

- 输出格式：主要采用 XML 等结构化格式进行数据输出，确保了数据的可读性、可解析性和跨平台交换能力。

- 行业意义：自 2010 年实施以来，它已成为我国财经信息技术领域的基础性标准。用友、金蝶等主流财务软件厂商首批通过了该标准的认证。它不仅保护了软件用户的权益，更极大地提升了审计信息化、监管智能化的水平，是连接会计实务与信息化审计的关键桥梁。

四川省注册会计师协会信息化委员会

四、算法：把会计规则写成代码

算法不是高深的数学，而是解决某一类问题的明确步骤。用合适的数据结构（如列表存储支出明细）承载数据，通过严谨的控制结构（顺序、分支、循环）和清晰的函数，按照既定的业务规则（数据标准），将输入数据转化为输出结果。

在会计工作中，你每天都在用算法：折旧计提、坏账准备、借款费用资本化、摊余成本计算、往来款核销……编程只是把这些步骤写成计算机能执行的代码。

4.1 案例一：借款费用资本化 —— 顺序 + 分支

业务场景：企业为购建固定资产借入专门借款，每季度需要计算有多少利息可以计入资产成本（资本化），多少应计入财务费用。

算法步骤：
1. 确定本季度实际发生的借款利息
2. 计算累计资产支出加权平均数（每笔支出 × 支出时间占季度比例）
3. 资本化利息 = 加权平均支出 × 资本化率
4. 如果资本化利息 > 实际利息，则只能按实际利息资本化（限制条件）

Python 代码（已简化为教学用）：

```python
def 资本化利息(借款本金，年利率，支出明细):
    """
    支出明细: list of (支出金额，本季度内经过的月数)
    """
    季度利率 = 年利率 / 4
    加权支出总额 = 0
    for 金额，月数 in 支出明细:
        加权支出总额 += 金额 (月数 / 3)

    资本化利息 = 加权支出总额 季度利率
    实际利息 = 借款本金 季度利率

     分支判断：不能超过实际利息
    if 资本化利息 > 实际利息:
        资本化利息 = 实际利息

    return 资本化利息
```

四川省注册会计师协会信息化委员会

示例：本金 500 万，年利率 6%，一季度内支出 200 万(满 3 个月)和 100 万(1.5 个月)

利息 = 资本化利息(500, 0.06, [(200, 3), (100, 1.5)])

print(f"应资本化利息：{利息:.2f} 万元")

控制结构：

- `for` 循环遍历每一笔支出（顺序累加）
- `if` 分支实现“资本化利息不能超过实际利息”的限制

四川省注册会计师协会信息化委员会

4.2 案例二：摊余成本（实际利率法） —— 循环 + 迭代

业务场景：企业折价发行债券，需要按实际利率法逐年计算利息收入和期末摊余成本。

算法步骤：
1. 期初摊余成本 = 上一期期末数（第一期就是发行价）
2. 利息收入 = 期初摊余成本 × 实际利率
3. 现金流入 = 面值 × 票面利率
4. 期末摊余成本 = 期初 + 利息收入 − 现金流入
5. 重复 1 - 4 步直到债券到期

Python 代码：

```python
def 摊余成本表(发行价，面值，票面利率，实际利率，总期数):
    摊余成本 = 发行价
    for 期次 in range(1，总期数 + 1):
        利息收入 = 摊余成本   实际利率
        现金流入 = 面值   票面利率
        期末成本 = 摊余成本 + 利息收入 - 现金流入
        print(f"第{期次}年：期初{摊余成本:.2f}，利息{利息收入:.2f}，"
              f"收回{现金流入:.2f}，期末{期末成本:.2f}")
        摊余成本 = 期末成本      迭代：期末变成下一期的期初

  参数：发行价 900 万，面值 1000 万，票面 4%，实际 6%，3 年期
摊余成本表(900, 1000, 0.04, 0.06, 3)
```

控制结构：
- `for` 循环固定执行 3 次（对应 3 年）
- 每次循环都使用上一轮计算出的“期末成本”作为新的“期初成本”——这就是迭代思想，和 Excel 中向下拖拽公式本质一样。

4.3 总结：算法不是什么魔法
> 你每天在会计工作中写的 “=IF(…)” 、 “=ROUND(…)” 、拖拽填充柄，都是算法。Python 只是让你把这些规则写得更加清晰、可复用，并且能处理成百上千行的财务数据。

四川省注册会计师协会信息化委员会

4.3 合并报表的“递归”本质：递归 VS 规则复用

从定义和操作流程上看，标准的合并报表编制过程，本身就是一次完美的递归算法演示。

- **递归算法的三要素：**

1. 基准情形：当一家公司（节点）没有子公司时，其个别报表就是合并报表。这是递归的“叶子节点”，计算终止。
2. 递归调用：要得到母公司 A 的合并报表，必须首先获得其每一个子公司 B、C……的合并报表。而对子公司 B 的合并，又需要先获得 B 的子公司（即 A 的孙公司）的合并报表。如此层层下探。
3. 合并（或称为“归结”）操作：在获得了下一层所有实体的合并报表后，在当前层执行合并操作（如权益的抵消、内部交易的抵消、数据的汇总）。这个操作在每个层级都一样，但应用的对象是下一层递归返回的结果。

- **对应到合并报表流程：**

1. 基准情形：一个没有任何投资、纯粹的经营实体，它的报表不需要合并，就是其自身。这对应递归的“底”。
2. 递归调用：集团合并会计人员的工作，在逻辑上必须从合并结构的最底层（那些只有经营业务，没有长期股权投资的实体）开始，逐级向上编制合并报表。你不能直接合并母公司，因为它的报表数字里包含了对于公司的投资，而子公司的报表里可能又包含了孙公司的权益。你必须先“解决”更底层的问题。
3. 合并操作：每一层级的合并，所遵循的准则（如《企业会计准则第 33 号——合并财务报表》）是同一套、固定的规则集。无论是母公司合并子公司，还是子公司合并孙公司，执行的抵消分录逻辑（长期股权投资与所有者权益抵消、内部往来抵消、内部交易未实现损益抵消）在性质上是完全相同的，只是应用的数据对象不同。

所以，整个合并报表的编制过程，就是一个以后台股权结构图为“树”，以合并抵销规则为“递归函数”，自底向上遍历计算的过程。

举例：

```
定义 函数 阶乘(n):
    如果 n 等于 0 或 1:          # 1. 基线条件
        返回 1
    否则:                        # 2. 递归调用 (并向基线靠近)
        返回 n * 阶乘(n-1)      # 3. 利用返回值组合出当前结果
```

四川省注册会计师协会信息化委员会

- **递归的实现：**

1. “树枝的展开” —— 递归的“递去”过程

- 动作：您从树干（主问题）开始，沿着一个分支（子问题）向前探索。每遇到一个分叉点（函数调用），您就选择一条分支继续向前，而这条分支的结构和您走过的路本质上相同（调用自己）。
- 对应递归：这就是函数不断自我调用，向更深处、更小规模的问题推进的过程。每一次调用，都像展开一个新的、更小的树枝。
- 在合并报表中：从母公司（树干）出发，为了合并它，您必须“展开”到它的子公司（第一层树枝）。为了合并子公司，您又必须“展开”到孙公司（第二层树枝）……直到抵达没有子公司的“叶子公司”。

2. “树枝、树叶的回卷” —— 递归的“归来/回溯”过程

- 动作：当您到达一片树叶（最小问题，基线条件）时，您无法再继续展开。此时，您开始沿着原路返回。在返回的路上，您会把沿途收集到的树叶信息（子问题的解）逐层整合起来。
- 对应递归：这就是函数到达基线条件后，开始逐层返回结果的过程。每一层函数在收到内层返回的结果后，结合自己这一层的信息，形成一个更大的结果，再返回给它的外层。

3、递归的核心机制：

执行（运行时）：是时间性的、顺序的过程。必然是 “先递后归”：先一路向下展开到基线，再带着结果一路向上回卷。这是程序在物理时间中的运行轨迹。

设计（规则/定义）：是逻辑性的、一体的。在递归函数的定义中，推动展开的 “调用自身” 和实现回卷的 “返回与组合” 是同时被规定、共存于一个整体规则之内的。它们是一个硬币不可分割的两面。

就是“在同一个静态代码框架下，动态地、层层嵌套地配置独立的运行环境”。当递归函数调用自身时：

（1）配置新环境：系统并不会复制函数代码，但会立即创建一个新的栈帧，为这次新调用配置一个全新的、独立的运行环境。新环境中的参数，通常代表了更小、更深入的问题（如子公司 vs 母公司）。

（2） 暂停当前环境：当前的运行环境（栈帧）会被完整保留在栈中，但执行被暂停，等待内层调用的结果。

（3.）重复此过程：这个“调用-创建新环境-暂停”的过程会一直重复，直到抵达基线条件。

- **合并报表过程的递归属性：**

静态剧本：合并算法（抵消、汇总等规则）。

动态舞台：每个公司实体（母公司、子公司、孙公司…）都是一个独立的“舞台”。

配置过程：为了合并母公司（舞台 A），算法运行到需要子公司数据时，就为子公司搭建一个新舞台 B，并暂停 A。舞台 B 同样运行这个算法，如果它还有下属，又会为孙公司搭建舞台 C……

结果传递：舞台 C（孙公司，叶子节点）演完，得出自己的报表，结果交给舞台 B。舞台 B 结合 C 的结果和自己本地的数据，完成合并，将结果交给舞台 A。最终舞台 A 完成整个集团的合并。

四川省注册会计师协会信息化委员会

当您抵达“叶子公司”（基线），它的个别报表就是最终结果。您带着这个结果“回卷”到其母公司（上一层）。母公司利用所有叶子公司返回的报表，完成自己这一层的抵消合并，生成新的、更大的合并结果，再继续向更上一层“回卷”。最终回卷到集团母公司，形成完整的合并报表。

“每一层或微观的展开，都在调用自己” —— 这正是递归调用，将问题推向基线。

“从全局或宏观来看，由于是调用自己，就像是回卷” —— 这是因为所有调用都发生在同一个函数名下，当最内层调用返回时，控制权和数据会沿着调用链原路逐层返回，宏观上看起来就像是从深处“卷”回来。

四川省注册会计师协会信息化委员会

第一部分小结

概念	会计类比	关键点
IDE	财务模板设计台与账套	解释器是底层框架，Project 是独立账套
对象	记账凭证/会计准则实例/辅助核算科目	属性+方法，继承与多态实现灵活系统
程序结构	流程控制(SOP)与功能模块(子系统)	顺序/分支/循环/异常处理；函数与模块化
数据	凭证、账簿、报表与交换标准	结构如档案柜，标准如准则，交换如接口
算法	借款费用资本化/摊余成本计算/合并过程	结构、数据与规则的代码实现/递归
IDE	财务模板设计台与账套	解释器是底层框架，Project 是独立账套

> 一句话记住第一部分：
> “编程就是用会计的思维（对象与结构），按照会计的规则（算法），去处理会计的数据，并在一个高效的环境（IDE 与库）中实现自动化。”

>会计的可计算性

静态维度：会计 = 概念 + 数据

概念：科目、准则、凭证结构、余额方向、辅助核算维度。
数据：具体金额、日期、业务摘要、客商名称。

动态维度：会计 = 概念 + 计算

计算：确认、计量、过账、折旧、摊销、合并抵消、成本分摊。

>会计的智能化

可计算是智能化的基础，但不能平替“智能化”。

概念本身有很多模糊定义，比如“很可能”、“实质上转移了风险”等。导致：概念与数据无法耦合（静态维度的断裂）、概念与计算无法标准化（动态维度的断裂）。**智能化，可能会以规则连续统的形式逐步出现、实现。也就是说，智能化不是一步到位的，而是在模糊概念和可计算规则之间，逐步引入判断标准，让灰色地带越来越窄的过程。**在一个模糊概念和可计算规则之间建立连续统，往往需要引入外部锚点——比如监管案例、审计实践惯例、行业共识、企业内部标准。这些外部锚点本身不是会计准则的一部分，却是把“很可能”翻译成可操作定义的关键。

接下来的第二部分和第三部分，就是在可计算边界之内，用 Python 的工具生态去做那些‘已经可以被计算’的财务工作。

四川省注册会计师协会信息化委员会

第二部分 特性 —— 会计人员的〝工具箱〞

> 目标：了解 Python 生态中常用的第三方库能帮你做什么，每个库解决一类特定的财务问题，就像财务软件中的不同模块。

注意与结构部分的联系。程序结构的实现都需要充分利用第三方库。

一、Flask —— “自定义的财务数据接口”

是什么：一个轻量级的 Web 框架，可以快速搭建一个小型网页应用。

会计类比：
- 好比你在 Excel 上做了一个复杂的预算模型，但希望同事通过浏览器输入几个数字就能运行，而不需要安装 Excel 或看懂公式。
- Flask 就是那个“把财务模型变成网页工具”的转换器。

管理会计场景：
- 搭建一个内部预算填报系统：各部门通过网页填预算数，后端自动汇总并生成差异分析报表。
- 制作一个成本测算小工具：销售人员输入产品参数，网页返回预估成本。

> 不需要会前端，用 Flask 几十行代码就能把 Python 计算能力“挂”到网页上。

举例

```
from flask import Flask, request, render_template_string

app = Flask(__name__)

 模拟已设定的年度总预算上限（例如公司总预算池）
TOTAL_BUDGET = 50000

 HTML 模板（内嵌，无需额外文件）
FORM_TEMPLATE = """
<!DOCTYPE html>
<html>
<head>
    <title>部门预算填报</title>
    <style>
        body { font-family: Arial; margin: 40px; }
        table { border-collapse: collapse; width: 60%; margin: 20px 0; }
        th, td { border: 1px solid ccc; padding: 8px; text-align: left; }
```

四川省注册会计师协会信息化委员会

```
        th { background-color: f2f2f2; }
        .summary { background-color: e6f7ff; font-weight: bold; }
        .warning { color: red; }
    </style>
</head>
<body>
    <h2> 部门预算填报系统</h2>
    <p>公司年度总预算额度：<strong>{{ total_budget }}</strong> 元</p>

    <form method="POST">
        <table>
            <tr>
                <th>部门</th>
                <th>预算金额（元）</th>
            </tr>
            <tr>
                <td>销售部</td>
                <td><input type="number" name="sales" step="0.01" min="0" required></td>
            </tr>
            <tr>
                <td>市场部</td>
                <td><input type="number" name="marketing" step="0.01" min="0" required></td>
            </tr>
            <tr>
                <td>研发部</td>
                <td><input type="number" name="rd" step="0.01" min="0" required></td>
            </tr>
            <tr>
                <td>行政部</td>
                <td><input type="number" name="admin" step="0.01" min="0" required></td>
            </tr>
        </table>
        <button type="submit">提交并汇总</button>
    </form>

    {% if result %}
    <hr>
    <h3> 预算汇总与差异分析</h3>
    <table>
        <tr>
            <th>部门</th>
            <th>填报预算（元）</th>
        </tr>
        {% for dept, amount in result.departments.items() %}
```

四川省注册会计师协会信息化委员会

```
            <tr>
                <td>{{ dept }}</td>
                <td>{{ "%.2f"|format(amount) }}</td>
            </tr>
            {% endfor %}
            <tr class="summary">
                <td><strong>合计</strong></td>
                <td><strong>{{ "%.2f"|format(result.total) }}</strong></td>
            </tr>
            <tr class="summary">
                <td>总预算额度</td>
                <td>{{ "%.2f"|format(result.budget_limit) }}</td>
            </tr>
            <tr class="summary">
                <td>差异（额度 - 合计）</td>
                <td>
                    {{ "%.2f"|format(result.diff) }}
                    {% if result.diff < 0 %}
                    <span class="warning">ˆ 超出预算！</span>
                    {% elif result.diff == 0 %}
                    <span>      刚好用完</span>
                    {% else %}
                    <span>      预算内结余</span>
                    {% endif %}
                </td>
            </tr>
        </table>
        {% endif %}
</body>
</html>
"""

@app.route('/', methods=['GET', 'POST'])
def budget_tool():
    """预算填报与汇总视图"""
    result = None
    if request.method == 'POST':
         从表单获取各部门预算（默认为 0）
        try:
            sales = float(request.form.get('sales', 0))
            marketing = float(request.form.get('marketing', 0))
            rd = float(request.form.get('rd', 0))
            admin = float(request.form.get('admin', 0))
        except ValueError:
```

四川省注册会计师协会信息化委员会

```python
            输入非法字符时返回提示（实际由前端 number 控件保证基本有效性）
            return "请输入有效的数字金额。", 400

        total = sales + marketing + rd + admin
        diff = TOTAL_BUDGET - total

        result = {
            'departments': {
                '销售部': sales,
                '市场部': marketing,
                '研发部': rd,
                '行政部': admin
            },
            'total': total,
            'budget_limit': TOTAL_BUDGET,
            'diff': diff
        }

    return render_template_string(FORM_TEMPLATE, total_budget=TOTAL_BUDGET, result=result)

if __name__ == '__main__':
app.run(debug=True)
```

四川省注册会计师协会信息化委员会

二、pandas / polars —— “财务数据工作台”/存储

是什么：处理表格数据的核心库，能读、写、清洗、合并、透视、计算财务数据。

会计类比：
- pandas 是 Excel 的超级加强版，但可以处理几十万行不卡顿，而且每一步操作都可以记录成代码，实现自动化。
- polars 是 pandas 的“速度升级版”，适合更大数据量。

管理会计/财务分析场景：
- 从 ERP 导出全年科目余额表，用 pandas 快速生成部门利润表（按成本中心拆分）。
- 合并多家子公司的报表，自动抵消内部往来。
- 对销售明细数据做多维分析（按产品、地区、月份汇总收入和成本）。

```python
import pandas as pd
df = pd.read_excel("2024年凭证.xlsx")
部门利润 = df.groupby(["部门", "科目类别"])["金额"].sum()
```

> 会计人员学 pandas，就像当年从手工账过渡到 Excel——掌握了，效率提升十倍。

举例

```
import pandas as pd

 ==================== 1. 模拟从 ERP 导出的科目余额表 ====================
 假设导出的 Excel 包含：日期、凭证号、部门、科目代码、科目名称、借方金额、贷方金额
data = {
    '日期': ['2024-01-15', '2024-02-20', '2024-03-10', '2024-03-15', '2024-04-05',
             '2024-05-12', '2024-06-08', '2024-07-22', '2024-08-30', '2024-09-18',
             '2024-10-25', '2024-11-10', '2024-12-05'],
    '部门': ['销售部', '销售部', '市场部', '研发部', '销售部',
             '市场部', '研发部', '销售部', '市场部', '研发部',
             '销售部', '市场部', '行政部'],
    '科目名称': ['主营业务收入', '管理费用', '主营业务收入', '研发费用', '销售费用',
               '主营业务收入', '研发费用', '主营业务收入', '管理费用', '研发费用',
               '销售费用', '主营业务收入', '管理费用'],
    '借方金额': [0, 1200, 0, 3500, 800,
               0, 4200, 0, 1500, 2800,
               950, 0, 600],
    '贷方金额': [15000, 0, 8000, 0, 0,
               12000, 0, 20000, 0, 0,
               0, 9000, 0]
```


四川省注册会计师协会信息化委员会

```
}

df = pd.DataFrame(data)
df['日期'] = pd.to_datetime(df['日期'])    转为日期类型

print("===== 原始科目余额表（前 5 行）=====")
print(df.head())
print("\n")

 =2. 计算各科目的净发生额（收入类：贷方-借方；费用类：借方-贷方）====================
 为简化，我们定义科目类型字典
科目类型 = {
    '主营业务收入': '收入',
    '管理费用': '费用',
    '研发费用': '费用',
    '销售费用': '费用'
}
df['科目类型'] = df['科目名称'].map(科目类型)

 定义函数计算净额（收入为正，费用为负）
def 计算净额(row):
    if row['科目类型'] == '收入':
        return row['贷方金额'] - row['借方金额']
    else:    费用类
        return row['借方金额'] - row['贷方金额']

df['净额'] = df.apply(计算净额, axis=1)

 ==================== 3. 按部门汇总生成部门利润表（贡献式） ====================
 以部门为分组，分别汇总收入和各项费用
部门汇总 = df.groupby(['部门', '科目名称'])['净额'].sum().unstack(fill_value=0)

 计算部门利润（收入 - 总费用）
部门汇总['费用合计'] = 部门汇总[['管理费用', '研发费用', '销售费用']].sum(axis=1)
部门汇总['部门利润'] = 部门汇总['主营业务收入'] + 部门汇总['费用合计']     费用合计为负数，直接相加即收入-
费用

print("===== 部门利润表（按成本中心拆分）=====")
print(部门汇总[['主营业务收入', '管理费用', '研发费用', '销售费用', '费用合计', '部门利润']])
print("\n")

 ==================== 4. 多维度分析：按月份和科目汇总 ====================
df['月份'] = df['日期'].dt.month
月度收入费用 = df.pivot_table(index='月份', columns='科目类型', values='净额', aggfunc='sum', fill_value=0)
```

四川省注册会计师协会信息化委员会

```
月度收入费用['净利润'] = 月度收入费用['收入'] + 月度收入费用['费用']    费用为负数

print("===== 月度损益汇总 =====")
print(月度收入费用)
print("\n")

 ==================== 5. 筛选并导出特定分析结果（模拟写入 Excel） ====================
 例如：只导出销售部的明细，用于进一步分析
销售部明细 = df[df['部门'] == '销售部'].copy()
销售部明细 = 销售部明细[['日期', '科目名称', '借方金额', '贷方金额', '净额']]

print("===== 销售部明细（可用于进一步追溯）=====")
print(销售部明细)

 如果希望保存为 Excel 文件（取消注释即可）
 with pd.ExcelWriter('部门利润分析报告.xlsx') as writer:
     df.to_excel(writer, sheet_name='原始数据', index=False)
     部门汇总.to_excel(writer, sheet_name='部门利润表')
     月度收入费用.to_excel(writer, sheet_name='月度损益')
     销售部明细.to_excel(writer, sheet_name='销售部明细', index=False)
 print("\n    分析报告已保存至 '部门利润分析报告.xlsx'")
```

四川省注册会计师协会信息化委员会

三、numpy / scipy —— “财务计算器与统计工具箱”

是什么：
- numpy 提供高效的数组运算（比 Python 原生快几十倍）。
- scipy 在 numpy 基础上增加了科学计算函数（如统计检验、最优化、插值）。

会计类比：
- numpy 好比 财务专用计算器（能快速算年金、现值、数组运算）。
- scipy 好比 财务建模中的高级函数库（如回归分析、概率分布）。

管理会计场景：
- 用 numpy 计算固定资产折旧矩阵（多项资产、多期间一次性算出）。
- 用 scipy 的优化功能求解最优生产组合（线性规划，用于资源分配决策）。
- 做销售预测：用 scipy 的统计函数检验历史数据的分布，拟合趋势。

```python
import numpy as np
 计算多项资产的年折旧额（直线法）
资产原值 = np.array([10000, 20000, 15000])
残值 = np.array([1000, 2000, 1500])
使用年限 = 5
年折旧 = (资产原值 - 残值) / 使用年限
举例
import numpy as np
from scipy import stats, optimize
import matplotlib.pyplot as plt
import warnings
warnings.filterwarnings('ignore')    忽略部分警告

 场景 1: 固定资产折旧计算（使用 numpy）
print("="  50)
print("场景 1: 固定资产折旧计算")
print("="  50)

 多项资产的年折旧额（直线法）
资产原值 = np.array([10000, 20000, 15000, 30000])
残值 = np.array([1000, 2000, 1500, 3000])
使用年限 = np.array([5, 8, 6, 10])    不同资产有不同使用年限

 计算各项资产的年折旧额
年折旧 = (资产原值 - 残值) / 使用年限
print(f"各项资产年折旧额: {年折旧}")

 创建折旧表（多项资产，多期间）
```

四川省注册会计师协会信息化委员会

```
折旧年限 = 10
资产数量 = len(资产原值)
折旧表 = np.zeros((折旧年限, 资产数量))

for i in range(资产数量):
    for j in range(min(使用年限[i], 折旧年限)):
        折旧表[j, i] = 年折旧[i]

print("\n 折旧表（行: 年份, 列: 资产）:")
print(折旧表)

 计算每年的总折旧费用
每年总折旧 = 折旧表.sum(axis=1)
print(f"\n 每年总折旧费用: {每年总折旧}")

 场景 2: 线性规划求解最优生产组合（使用 scipy.optimize）
print("\n" + "="    50)
print("场景 2: 最优生产组合（线性规划）")
print("="    50)

 问题：公司生产两种产品 A 和 B，需要最大化利润
 产品 A: 利润 50 元/单位，需要 2 小时机器时间，1 公斤原料
 产品 B: 利润 60 元/单位，需要 1 小时机器时间，3 公斤原料
 约束: 机器时间 ≤ 100 小时，原料 ≤ 90 公斤

 目标函数系数（最大化利润 = 最小化负利润）
c = np.array([-50, -60])     负号因为 linprog 默认求最小值

 不等式约束矩阵 A_ub    x <= b_ub
A_ub = np.array([
    [2, 1],      机器时间约束
    [1, 3]       原料约束
])
b_ub = np.array([100, 90])

 变量边界（生产数量不能为负）
bounds = [(0, None), (0, None)]

 求解线性规划问题
result = optimize.linprog(c, A_ub=A_ub, b_ub=b_ub, bounds=bounds, method='highs')

if result.success:
    x1, x2 = result.x
    max_profit = -result.fun
```

四川省注册会计师协会信息化委员会

```
    print(f"最优生产计划:")
    print(f"    产品 A 产量: {x1:.2f}  单位")
    print(f"    产品 B 产量: {x2:.2f}  单位")
    print(f"    最大利润: {max_profit:.2f}  元")

     检查约束
    machine_hours = 2x1 + 1x2
    material_used = 1x1 + 3x2
    print(f"\n 资源使用情况:")
    print(f"    机器时间使用: {machine_hours:.2f}  小时  (上限: 100 小时)")
    print(f"    原料使用: {material_used:.2f}  公斤  (上限: 90 公斤)")
else:
    print("未能找到最优解")

 场景 3:  销售预测和统计分析（使用 scipy.stats）
print("\n" + "="    50)
print("场景 3:  销售预测与统计分析")
print("="    50)

 模拟 12 个月的销售额数据（单位：万元）
months = np.arange(1, 13)
sales_data = np.array([12.5, 13.2, 13.8, 14.5, 15.1, 15.8,
                       16.5, 17.3, 18.1, 18.9, 19.7, 20.6])

print(f"销售额数据（12 个月）: {sales_data}")

 描述性统计分析
sales_mean = np.mean(sales_data)
sales_std = np.std(sales_data)
sales_min = np.min(sales_data)
sales_max = np.max(sales_data)
sales_median = np.median(sales_data)

print(f"\n 描述性统计:")
print(f"    平均值: {sales_mean:.2f}  万元")
print(f"    标准差: {sales_std:.2f}  万元")
print(f"    最小值: {sales_min:.2f}  万元")
print(f"    最大值: {sales_max:.2f}  万元")
print(f"    中位数: {sales_median:.2f}  万元")

 线性回归拟合趋势
slope, intercept, r_value, p_value, std_err = stats.linregress(months, sales_data)
print(f"\n 线性回归结果:")
print(f"    斜率: {slope:.3f} (每月增长)")
```

四川省注册会计师协会信息化委员会

```
print(f"    截距: {intercept:.3f}")
print(f"    R²: {r_value2:.3f} (拟合优度)")

 预测未来 3 个月的销售额
future_months = np.arange(13, 16)
predicted_sales = intercept + slope    future_months
print(f"\n 销售预测:")
for month, sales in zip(future_months, predicted_sales):
    print(f"    第{month}个月预测销售额: {sales:.2f} 万元")

 检验数据是否服从正态分布（Shapiro-Wilk 检验）
 注意：小样本量可能影响检验结果
if len(sales_data) >= 3 and len(sales_data) <= 5000:
    shapiro_stat, shapiro_p = stats.shapiro(sales_data)
    print(f"\n 正态性检验 (Shapiro-Wilk):")
    print(f"    统计量: {shapiro_stat:.3f}")
    print(f"    P 值: {shapiro_p:.3f}")
    if shapiro_p > 0.05:
        print("    结论：不能拒绝正态性假设（α=0.05)")
    else:
        print("    结论：拒绝正态性假设（α=0.05)")

 可视化
fig, axes = plt.subplots(1, 3, figsize=(15, 5))

 子图 1：折旧表
axes[0].bar(range(1，资产数量+1)，资产原值, alpha=0.6, label='资产原值')
axes[0].bar(range(1，资产数量+1)，残值, alpha=0.6, label='残值')
axes[0].set_xlabel('资产编号')
axes[0].set_ylabel('金额 (元)')
axes[0].set_title('固定资产折旧计算')
axes[0].legend()

 子图 2：线性规划结果
labels = ['产品 A', '产品 B']
x_pos = np.arange(len(labels))
axes[1].bar(x_pos, result.x, alpha=0.7, color=['blue', 'orange'])
axes[1].set_xlabel('产品')
axes[1].set_ylabel('最优产量 (单位)')
axes[1].set_title('最优生产组合')
axes[1].set_xticks(x_pos)
axes[1].set_xticklabels(labels)

 子图 3：销售预测
```

四川省注册会计师协会信息化委员会

```
axes[2].scatter(months, sales_data, label='实际销售额', color='blue')
axes[2].plot(months, intercept + slope   months, 'r-', label='趋势线')
axes[2].scatter(future_months, predicted_sales, label='预测', color='green', s=100, marker='s')
axes[2].set_xlabel('月份')
axes[2].set_ylabel('销售额 (万元)')
axes[2].set_title('销售趋势与预测')
axes[2].legend()
axes[2].grid(True, alpha=0.3)

plt.tight_layout()
plt.show()

 输出总结
print("\n" + "="   50)
print("总结")
print("="   50)
print("1. numpy 提供了高效的数组运算，特别适合处理财务中的批量计算")
print("2. scipy.optimize 可以解决资源分配、成本最小化、利润最大化等优化问题")
print("3. scipy.stats 提供了丰富的统计函数，可用于数据分析、预测和假设检验")
```

四川省注册会计师协会信息化委员会

四、matplotlib / seaborn / pyvis

—— “财务图表与关系可视化”

是什么：
- matplotlib 是基础绘图库（画折线图、柱状图、饼图等）。
- seaborn 是统计图表库，更美观、适合分析报告。
- pyvis 是交互式网络图库，用于展示节点和连线关系。

会计类比：
- 好比 Excel 中的图表功能，但自动生成、样式统一、可嵌入报告。
- pyvis 则像资金流向图或关联关系图。

管理会计/财务分析场景：
- 用 matplotlib 绘制月度成本趋势图，一眼看出异常波动。
- 用 seaborn 制作热力图展示各产品在各地区的毛利率分布。
- 用 pyvis 展示母子公司股权结构图或资金拆借网络（点击可拖动）。

```python
import matplotlib.pyplot as plt
months = ["1 月","2 月","3 月"]
actual = [120,135,128]
budget = [125,130,130]
plt.plot(months, actual, label="实际")
plt.plot(months, budget, label="预算")
plt.legend(); plt.show()
```

举例

```
"""
财务可视化分析工具
使用 matplotlib、seaborn 和 pyvis 实现财务图表与关系可视化
"""


import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from pyvis.network import Network
import warnings
warnings.filterwarnings('ignore')


 设置中文字体（如果系统支持）
plt.rcParams['font.sans-serif'] = ['SimHei', 'Arial Unicode MS', 'DejaVu Sans']
```


四川省注册会计师协会信息化委员会

```
plt.rcParams['axes.unicode_minus'] = False

 设置 seaborn 样式
sns.set_style("whitegrid")
sns.set_palette("husl")
```

1. 月度成本趋势分析 (使用 matplotlib)

```
def monthly_cost_analysis():
    """月度成本趋势分析"""
    print("="   50)
    print("1. 月度成本趋势分析")
    print("="   50)

     创建示例数据
    months = ["1 月", "2 月", "3 月", "4 月", "5 月", "6 月",
              "7 月", "8 月", "9 月", "10 月", "11 月", "12 月"]

     实际成本数据
    actual_costs = {
        "原材料成本": np.array([120, 135, 128, 140, 155, 160, 165, 170, 168, 175, 180, 185]),
        "制造费用": np.array([80, 85, 82, 90, 95, 100, 105, 110, 108, 115, 120, 125]),
        "销售费用": np.array([50, 55, 52, 60, 65, 70, 75, 80, 78, 85, 90, 95]),
        "管理费用": np.array([40, 42, 41, 45, 48, 50, 52, 55, 54, 58, 60, 62])
    }

     预算数据
    budget_costs = {
        "原材料成本": np.array([125, 130, 130, 135, 150, 155, 160, 165, 165, 170, 175, 180]),
        "制造费用": np.array([85, 85, 85, 90, 95, 100, 105, 110, 110, 115, 120, 125]),
        "销售费用": np.array([55, 55, 55, 60, 65, 70, 75, 80, 80, 85, 90, 95]),
        "管理费用": np.array([45, 45, 45, 45, 50, 50, 55, 55, 55, 60, 60, 65])
    }

     计算总成本
    total_actual = sum(actual_costs.values())
    total_budget = sum(budget_costs.values())

     创建图形
    fig, axes = plt.subplots(2, 2, figsize=(14, 10))
    fig.suptitle('月度成本趋势分析', fontsize=16, fontweight='bold')

     子图 1: 各成本项目对比
    x = np.arange(len(months))
    width = 0.35
```

四川省注册会计师协会信息化委员会

```
for idx, (cost_type, costs) in enumerate(actual_costs.items()):
    ax = axes[idx // 2, idx % 2]

     绘制柱状图
    ax.bar(x - width/2, costs, width, label='实际', alpha=0.8, color='steelblue')
    ax.bar(x + width/2, budget_costs[cost_type], width, label='预算', alpha=0.8, color='lightcoral')

    ax.set_xlabel('月份')
    ax.set_ylabel('成本 (万元)')
    ax.set_title(f'{cost_type}趋势')
    ax.set_xticks(x)
    ax.set_xticklabels(months, rotation=45)
    ax.legend()
    ax.grid(True, alpha=0.3)

plt.tight_layout()
plt.show()

 创建第二个图形：总成本趋势
plt.figure(figsize=(12, 6))

 绘制总成本趋势线
plt.plot(months, total_actual, 'o-', linewidth=2, markersize=8,
         label='实际总成本', color='2E86AB')
plt.plot(months, total_budget, 's--', linewidth=2, markersize=8,
         label='预算总成本', color='A23B72')

 标记异常点（实际成本超过预算 10%以上）
threshold = 1.1
anomalies = np.where(total_actual > total_budget    threshold)[0]

for anomaly in anomalies:
    plt.scatter(months[anomaly], total_actual[anomaly],
                s=200, color='red', alpha=0.6,
                label='异常点' if anomaly == anomalies[0] else "")

plt.xlabel('月份', fontsize=12)
plt.ylabel('总成本 (万元)', fontsize=12)
plt.title('月度总成本趋势（实际 vs 预算）', fontsize=14, fontweight='bold')
plt.legend()
plt.grid(True, alpha=0.3)
plt.xticks(rotation=45)
plt.tight_layout()
```

四川省注册会计师协会信息化委员会

```
plt.show()

 计算成本偏差
cost_variance = total_actual - total_budget
variance_percentage = (cost_variance / total_budget)   100

print("月度成本分析报告:")
print("-"   40)
for i, month in enumerate(months):
    variance_sign = "+" if cost_variance[i] > 0 else ""
    print(f"{month}: 实际={total_actual[i]:.1f}万, "
          f"预算={total_budget[i]:.1f}万, "
          f"偏差={variance_sign}{cost_variance[i]:.1f}万 "
          f"({variance_sign}{variance_percentage[i]:.1f}%)")

return months, actual_costs, budget_costs
```

2. 产品地区毛利率热力图 (使用 seaborn)

```
def product_region_profitability():
    """产品地区毛利率分析"""
    print("\n" + "="   50)
    print("2. 产品地区毛利率热力图分析")
    print("="   50)

     创建示例数据
    products = ["产品 A", "产品 B", "产品 C", "产品 D", "产品 E"]
    regions = ["华东", "华南", "华北", "华中", "西南", "西北", "东北"]

     生成随机的毛利率数据（-10% 到 50%）
    np.random.seed(42)
    profit_margin = np.random.uniform(-10, 50, size=(len(products), len(regions)))

     创建 DataFrame
    profit_df = pd.DataFrame(profit_margin, index=products, columns=regions)

    print("各产品在各地区的毛利率 (%):")
    print(profit_df.round(1))

     创建热力图
    plt.figure(figsize=(12, 8))

     使用 seaborn 绘制热力图
    ax = sns.heatmap(profit_df,
                     annot=True,
```

四川省注册会计师协会信息化委员会

```
                    fmt=".1f",
                    cmap="RdYlGn",    红-黄-绿色系，红色表示低，绿色表示高
                    center=20,    中心点为 20%
                    vmin=-10,     最小值-10%
                    vmax=50,      最大值 50%
                    linewidths=0.5,
                    linecolor='gray',
                    cbar_kws={'label': '毛利率 (%)'})

    plt.title('各产品在各地区的毛利率分布', fontsize=16, fontweight='bold', pad=20)
    plt.xlabel('地区', fontsize=12)
    plt.ylabel('产品', fontsize=12)
    plt.tight_layout()
    plt.show()

     添加统计摘要
    print("\n 毛利率统计分析:")
    print("-"   40)
    print(f"整体平均毛利率: {profit_df.values.mean():.1f}%")
    print(f" 毛 利 率 最 大 值 ： {profit_df.values.max():.1f}% （ 产 品 ： {profit_df.stack().idxmax()[0]}， 地 区 ：
{profit_df.stack().idxmax()[1]})")
    print(f" 毛 利 率 最 小 值 ： {profit_df.values.min():.1f}% （ 产 品 ： {profit_df.stack().idxmin()[0]}， 地 区 ：
{profit_df.stack().idxmin()[1]})")

     按产品分析
    print("\n 按产品平均毛利率排序:")
    product_avg = profit_df.mean(axis=1).sort_values(ascending=False)
    for product, margin in product_avg.items():
        print(f"   {product}: {margin:.1f}%")

     按地区分析
    print("\n 按地区平均毛利率排序:")
    region_avg = profit_df.mean(axis=0).sort_values(ascending=False)
    for region, margin in region_avg.items():
        print(f"   {region}: {margin:.1f}%")

    return profit_df
```

3. 股权结构与资金流向图 (使用 pyvis)

```
def corporate_structure_network():
    """公司股权结构与资金流向网络图"""
    print("\n" + "="   50)
    print("3. 公司股权结构与资金流向网络图")
    print("="   50)
```


四川省注册会计师协会信息化委员会

```
 创建网络图
net = Network(height="700px", width="100%",
              bgcolor="222222", font_color="white",
              notebook=False, directed=True)

 设置物理布局
net.barnes_hut(gravity=-80000, central_gravity=0.3,
               spring_length=250, spring_strength=0.001,
               damping=0.09, overlap=0)

 定义节点 (公司)
companies = [
    {"id": 1, "label": "母公司\n(控股集团)", "group": "parent", "value": 100, "title": "总资产: 1000 亿"},
    {"id": 2, "label": "子公司 A\n(制造业)", "group": "subsidiary", "value": 40, "title": "总资产: 200 亿"},
    {"id": 3, "label": "子公司 B\n(金融)", "group": "subsidiary", "value": 60, "title": "总资产: 300 亿"},
    {"id": 4, "label": "子公司 C\n(房地产)", "group": "subsidiary", "value": 30, "title": "总资产: 150 亿"},
    {"id": 5, "label": "孙公司 A1\n(零部件)", "group": "grandchild", "value": 20, "title": "总资产: 50 亿"},
    {"id": 6, "label": "孙公司 A2\n(装配)", "group": "grandchild", "value": 20, "title": "总资产: 50 亿"},
    {"id": 7, "label": "关联公司 X", "group": "related", "value": 15, "title": "总资产: 80 亿"},
    {"id": 8, "label": "关联公司 Y", "group": "related", "value": 10, "title": "总资产: 60 亿"},
]

 定义边 (股权关系和资金流向)
edges = [
     股权关系 (母公司->子公司)
    {"from": 1, "to": 2, "label": "控股 60%", "value": 60, "color": "2E86AB", "title": "股权投资: 120 亿"},
    {"from": 1, "to": 3, "label": "控股 80%", "value": 80, "color": "2E86AB", "title": "股权投资: 240 亿"},
    {"from": 1, "to": 4, "label": "控股 70%", "value": 70, "color": "2E86AB", "title": "股权投资: 105 亿"},

     子公司->孙公司
    {"from": 2, "to": 5, "label": "控股 100%", "value": 100, "color": "A23B72", "title": "股权投资: 50 亿"},
    {"from": 2, "to": 6, "label": "控股 100%", "value": 100, "color": "A23B72", "title": "股权投资: 50 亿"},

     关联关系
    {"from": 1, "to": 7, "label": "参股 20%", "value": 20, "color": "F18F01", "title": "股权投资: 16 亿"},
    {"from": 3, "to": 8, "label": "参股 15%", "value": 15, "color": "F18F01", "title": "股权投资: 9 亿"},

     资金拆借关系
    {"from": 3, "to": 2, "label": "拆借 5 亿", "value": 5, "color": "73AB84", "title": "资金拆借: 5 亿元",
"dashes": True},
    {"from": 1, "to": 4, "label": "拆借 8 亿", "value": 8, "color": "73AB84", "title": "资金拆借: 8 亿元",
"dashes": True},
    {"from": 2, "to": 5, "label": "借款 2 亿", "value": 2, "color": "73AB84", "title": "内部借款: 2 亿元",
```

四川省注册会计师协会信息化委员会

```
"dashes": True},
                    ]

                    添加节点
                    for company in companies:
                        根据分组设置不同颜色
                        color_map = {
                            "parent": "2E86AB",        母公司 - 深蓝色
                            "subsidiary": "A23B72",    子公司 - 紫色
                            "grandchild": "C73E1D",    孙公司 - 红色
                            "related": "F18F01"        关联公司 - 橙色
                        }

                        net.add_node(company["id"],
                                     label=company["label"],
                                     value=company["value"],
                                     title=company["title"],
                                     color=color_map[company["group"]],
                                     shape="dot")

                    添加边
                    for edge in edges:
                        net.add_edge(edge["from"], edge["to"],
                                     label=edge["label"],
                                     value=edge["value"],
                                     title=edge.get("title", ""),
                                     color=edge["color"],
                                     width=edge["value"]/20,    线宽与值成正比
                                     arrows="to",
                                     dashes=edge.get("dashes", False))

                    添加图例节点
                    legend_nodes = [
                        {"id": 100, "label": "图例", "shape": "box", "color": "ffffff", "font": {"color": "000000"}},
                        {"id": 101, "label": "母公司", "shape": "dot", "color": "2E86AB", "size": 20},
                        {"id": 102, "label": "子公司", "shape": "dot", "color": "A23B72", "size": 20},
                        {"id": 103, "label": "孙公司", "shape": "dot", "color": "C73E1D", "size": 20},
                        {"id": 104, "label": "关联公司", "shape": "dot", "color": "F18F01", "size": 20},
                        {"id": 105, "label": "股权关系", "shape": "dot", "color": "2E86AB", "size": 20},
                        {"id": 106, "label": "资金拆借", "shape": "dot", "color": "73AB84", "size": 20},
                    ]

                    for node in legend_nodes:
                        net.add_node(node["id"],
```

四川省注册会计师协会信息化委员会

```
                label=node["label"],
                color=node["color"],
                shape=node["shape"],
                size=node.get("size", 10),
                font=node.get("font", {"color": "white"}),
                fixed=True,
                x=100 if node["id"] == 100 else 100,
                y=100 + (node["id"] - 100)   50)

 设置图选项
net.set_options("""
var options = {
  "nodes": {
    "font": {
      "size": 14
    }
  },
  "edges": {
    "font": {
      "size": 12,
      "align": "middle"
    },
    "smooth": {
      "type": "continuous"
    }
  },
  "physics": {
    "enabled": true
  },
  "interaction": {
    "dragNodes": true,
    "hideEdgesOnDrag": false,
    "tooltipDelay": 200
  }
}
""")

 保存为 HTML 文件
output_file = "corporate_structure_network.html"
net.save_graph(output_file)

print(f"股权结构网络图已保存为: {output_file}")
print("请用浏览器打开该文件查看交互式图表")
print("\n 图例说明:")
```

四川省注册会计师协会信息化委员会

```
    print("  • 节点大小表示公司规模")
    print("  • 实线: 股权关系 (线宽表示持股比例)")
    print("  • 虚线: 资金拆借关系")
    print("  • 颜色分类:")
    print("     - 深蓝色: 母公司")
    print("     - 紫色: 子公司")
    print("     - 红色: 孙公司")
    print("     - 橙色: 关联公司")
    print("     - 绿色: 资金拆借")

    return output_file

 主程序
def main():
    """主函数"""
    print("财务可视化分析工具")
    print("="   50)
    print("本工具包含以下功能:")
    print("1. 月度成本趋势分析 (matplotlib)")
    print("2. 产品地区毛利率热力图 (seaborn)")
    print("3. 股权结构与资金流向图 (pyvis)")
    print("="   50)

    try:
         运行月度成本分析
        months, actual_costs, budget_costs = monthly_cost_analysis()

         运行产品地区毛利率分析
        profit_df = product_region_profitability()

         运行股权结构网络图
        network_file = corporate_structure_network()

        print("\n" + "="   50)
        print("分析完成!")
        print("="   50)
        print("输出结果:")
        print(f"1. 月度成本趋势图: 已显示")
        print(f"2. 产品地区毛利率热力图: 已显示")
        print(f"3. 股权结构网络图: 已保存为 '{network_file}'")
        print("\n 注意事项:")
        print("- 确保已安装所需库: pip install numpy pandas matplotlib seaborn pyvis")
        print("- 股权结构网络图是交互式的, 可用鼠标拖拽节点")
        print("- 将鼠标悬停在节点或连线上可查看详细信息")
```

四川省注册会计师协会信息化委员会

```
    except ImportError as e:
        print(f"导入错误: {e}")
        print("请确保已安装所有必要的库:")
        print("    pip install numpy pandas matplotlib seaborn pyvis")
    except Exception as e:
        print(f"运行过程中发生错误: {e}")

if __name__ == "__main__":
    main()
```

四川省注册会计师协会信息化委员会

五、networkx —— “会计科目网络与勾稽关系”

是什么：专门处理图（网络）结构的库，可以分析节点之间的连接关系。

会计类比：
- 会计科目本身可以看作一个树状结构（一级科目→明细科目），但 networkx 还可以分析分录之间的勾稽关系或科目之间的影响传导。

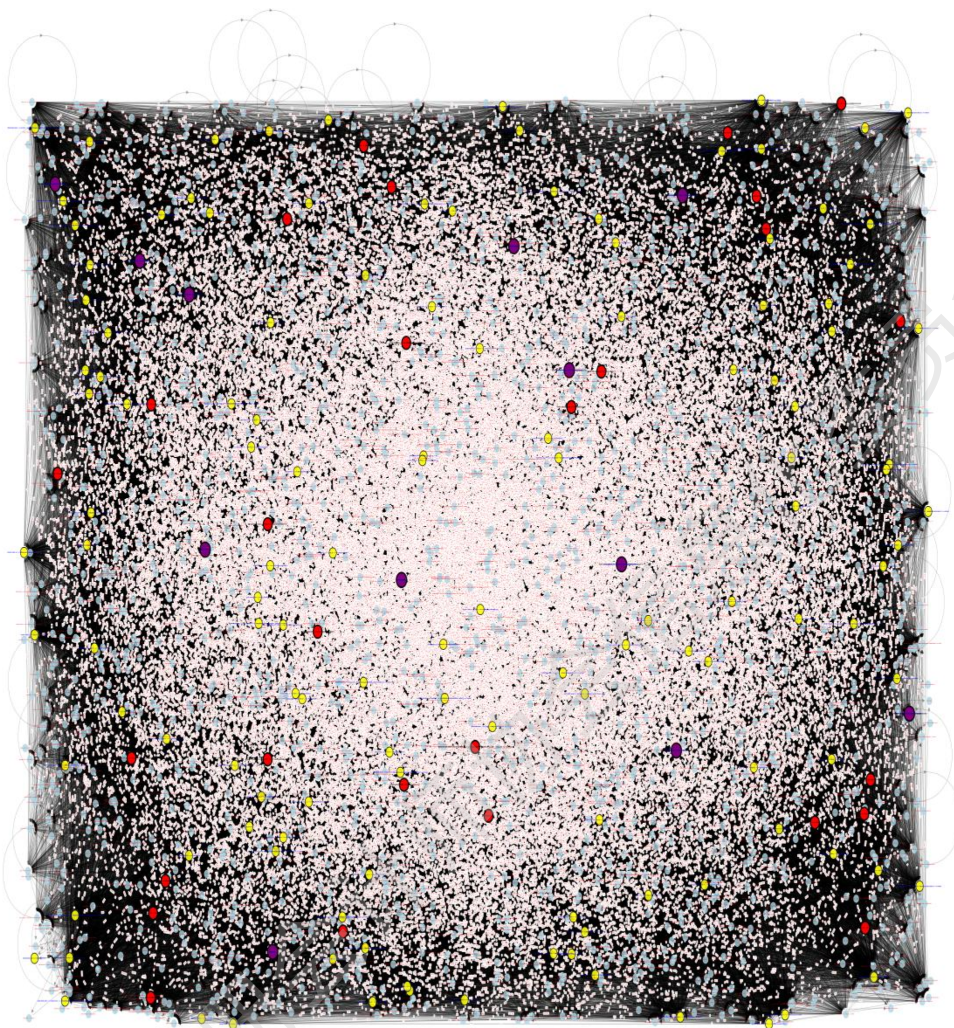
财务分析场景：
- 构建会计科目关联图谱：比如“营业收入”变动会影响“应收账款”、“税金”、“净利润”等。
- 分析现金流量路径：从一笔销售到最终回款，经过哪些中间科目。
- 做合并报表时的抵消关系网：哪些公司之间有关联交易。

```python
import networkx as nx
G = nx.DiGraph()
G.add_edges_from([("营业收入","应收账款"), ("营业收入","应交税费"), ("应收账款","经营活动现金流")])
nx.draw(G, with_labels=True)
```

> 管理会计中也可以用于成本分配路径（辅助生产车间之间的交互分配）。

四川省注册会计师协会信息化委员会

试验2-1-12月--第1分图
关键失效顶点占比:0.0197、图密度:0.9716、幂律分布斜率绝对值:0.319
红色——割点 黄色——中心点 紫色——割点+中心点 蓝色——一般点



六、scikit-learn —— “财务预测与风险评分”

是什么：最流行的机器学习库，包含分类、回归、聚类等算法。

会计类比：
- 相当于一个自动建模工具，可以从历史数据中学习规律，然后对新数据进行预测或分类。

管理会计/财务分析场景：
- 客户信用评分：根据历史回款数据，训练模型预测新客户的逾期风险。
- 成本动因分析：用回归模型找出哪些因素（产量、订单数、运输距离）对成本影响最大。
- 预算异常检测：自动标记与历史模式差异过大的费用报销单。
- 销售预测：用时间序列特征预测下季度收入。

> 不需要成为数据科学家，调用几行代码就能完成经典算法（如线性回归、决策树）。

```python
from sklearn.linear_model import LinearRegression
 X: 广告投入, Y: 销售收入
model = LinearRegression().fit(X, Y)
预测收入  = model.predict([[新广告投入]])
```
举例

```
"""
财务预测与风险评分系统
使用 scikit-learn 进行财务预测、风险评分、异常检测和成本分析
"""

import numpy as np
import pandas as pd
from datetime import datetime, timedelta
import warnings
warnings.filterwarnings('ignore')

 机器学习相关库
from sklearn.model_selection import train_test_split, cross_val_score, GridSearchCV
from sklearn.preprocessing import StandardScaler, LabelEncoder, OneHotEncoder
from sklearn.impute import SimpleImputer
from sklearn.pipeline import Pipeline
from sklearn.compose import ColumnTransformer
from sklearn.metrics import (accuracy_score, precision_score, recall_score, f1_score,
                             roc_auc_score, confusion_matrix, classification_report,
                             mean_absolute_error, mean_squared_error, r2_score)
```

四川省注册会计师协会信息化委员会

```
 模型
from sklearn.linear_model import LinearRegression, LogisticRegression, Ridge, Lasso
from sklearn.ensemble import RandomForestClassifier, RandomForestRegressor, GradientBoostingRegressor
from sklearn.svm import SVC, SVR
from sklearn.tree import DecisionTreeClassifier, DecisionTreeRegressor
from sklearn.neighbors import KNeighborsClassifier
from sklearn.cluster import KMeans
from sklearn.ensemble import IsolationForest
from sklearn.decomposition import PCA

 可视化
import matplotlib.pyplot as plt
import seaborn as sns
from matplotlib import rcParams

 设置中文字体
rcParams['font.sans-serif'] = ['Arial Unicode MS', 'DejaVu Sans', 'sans-serif']
rcParams['axes.unicode_minus'] = False

 设置 seaborn 样式
sns.set_style("whitegrid")
sns.set_palette("husl")

class FinancialMLSystem:
    """财务机器学习系统"""

    def __init__(self, random_state=42):
        """初始化"""
        self.random_state = random_state
        np.random.seed(random_state)

    def generate_credit_data(self, n_samples=1000):
        """
        生成客户信用评分模拟数据

        Args:
            n_samples: 样本数量

        Returns:
            DataFrame: 包含特征和标签的数据
        """
        print("生成客户信用评分数据...")

         基本特征
```

四川省注册会计师协会信息化委员会

```
        data = pd.DataFrame({
            '年龄': np.random.normal(45, 15, n_samples).astype(int).clip(20, 70),
            '年收入(万)': np.random.lognormal(3.5, 0.8, n_samples).clip(5, 200),
            '工作年限': np.random.exponential(10, n_samples).astype(int).clip(0, 40),
            '信用卡数量': np.random.poisson(2.5, n_samples).clip(0, 10),
            '负债收入比': np.random.beta(2, 5, n_samples)  0.8 + 0.1,   0.1-0.9
            '历史逾期次数': np.random.poisson(0.5, n_samples).clip(0, 5),
            '房产情况': np.random.choice(['无房产', '有房产', '有房产有贷款'], n_samples, p=[0.3, 0.5, 0.2]),
            '婚姻状况': np.random.choice(['未婚', '已婚', '离异', '丧偶'], n_samples, p=[0.3, 0.5, 0.15, 0.05]),
            '教育程度': np.random.choice(['高中及以下', '大专', '本科', '硕士', '博士'], n_samples, p=[0.2, 0.3,
0.3, 0.15, 0.05]),
        })

         计算信用分数（逻辑函数）
         高风险因素：高负债收入比、高逾期次数、无房产
        risk_score = (
            -0.1  (data['年龄'] - 45) / 10 +    年龄适中风险低
            0.5  (data['负债收入比'] - 0.5) +    高负债风险高
            0.8  data['历史逾期次数'] +    逾期记录风险高
            0.3  (data['信用卡数量'] > 5).astype(int) -    过多信用卡风险高
            0.4  (data['房产情况'] == '有房产').astype(int) +    有房产风险低
            0.2  (data['教育程度'].isin(['硕士', '博士'])).astype(int)    高学历风险低
        )

         将风险分数转换为违约概率
        default_prob = 1 / (1 + np.exp(-risk_score))

         生成标签（是否违约）
        data['是否违约'] = (default_prob > 0.5).astype(int)

         添加噪声
        noise = np.random.rand(n_samples) < 0.1
        data.loc[noise, '是否违约'] = 1 - data.loc[noise, '是否违约']

        print(f"生成完成: {n_samples} 条记录")
        print(f"违约客户: {data['是否违约'].sum()} ({data['是否违约'].mean()100:.1f}%)")
        print(f"正常客户: {(data['是否违约']==0).sum()} ({(data['是否违约']==0).mean()100:.1f}%)")

        return data

    def credit_scoring_model(self, data):
        """
        客户信用评分模型
```

四川省注册会计师协会信息化委员会

```
Args:
    data: 客户数据

Returns:
    训练好的模型和评估结果
"""
print("\n" + "="60)
print("场景 1: 客户信用评分模型")
print("="60)

 准备特征和标签
X = data.drop('是否违约', axis=1)
y = data['是否违约']

 划分训练集和测试集
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=self.random_state, stratify=y
)

print(f"训练集大小: {X_train.shape[0]} 条记录")
print(f"测试集大小: {X_test.shape[0]} 条记录")

 数据预处理管道
numeric_features = ['年龄', '年收入(万)', '工作年限', '信用卡数量', '负债收入比', '历史逾期次数']
categorical_features = ['房产情况', '婚姻状况', '教育程度']

numeric_transformer = Pipeline(steps=[
    ('imputer', SimpleImputer(strategy='median')),
    ('scaler', StandardScaler())
])

categorical_transformer = Pipeline(steps=[
    ('imputer', SimpleImputer(strategy='most_frequent')),
    ('onehot', OneHotEncoder(handle_unknown='ignore', sparse_output=False))
])

preprocessor = ColumnTransformer(
    transformers=[
        ('num', numeric_transformer, numeric_features),
        ('cat', categorical_transformer, categorical_features)
    ]
)

 创建和训练模型
```

四川省注册会计师协会信息化委员会

```
model = Pipeline(steps=[
    ('preprocessor', preprocessor),
    ('classifier', RandomForestClassifier(
        n_estimators=100,
        max_depth=10,
        min_samples_split=5,
        min_samples_leaf=2,
        random_state=self.random_state,
        class_weight='balanced'
    ))
])

print("\n 训练模型...")
model.fit(X_train, y_train)

 预测
y_pred = model.predict(X_test)
y_pred_proba = model.predict_proba(X_test)[:, 1]

 评估模型
print("\n 模型评估结果:")
print("-"40)
print(f"准确率: {accuracy_score(y_test, y_pred):.4f}")
print(f"精确率: {precision_score(y_test, y_pred):.4f}")
print(f"召回率: {recall_score(y_test, y_pred):.4f}")
print(f"F1 分数: {f1_score(y_test, y_pred):.4f}")
print(f"AUC 得分: {roc_auc_score(y_test, y_pred_proba):.4f}")

 混淆矩阵
cm = confusion_matrix(y_test, y_pred)
print("\n 混淆矩阵:")
print(f"              预测正常    预测违约")
print(f"实际正常    {cm[0,0]:>8}    {cm[0,1]:>8}")
print(f"实际违约    {cm[1,0]:>8}    {cm[1,1]:>8}")

 特征重要性
if hasattr(model.named_steps['classifier'], 'feature_importances_'):
     获取特征名称
    feature_names = numeric_features.copy()

     添加分类特征的 one-hot 编码后的名称
    categorical_encoder = model.named_steps['preprocessor'].named_transformers_['cat'].named_steps['onehot']
    cat_feature_names = categorical_encoder.get_feature_names_out(categorical_features)
```

四川省注册会计师协会信息化委员会

```
    all_feature_names = np.concatenate([feature_names, cat_feature_names])

    importances = model.named_steps['classifier'].feature_importances_
    indices = np.argsort(importances)[::-1]

    print("\n 特征重要性 (前 10 个):")
    print("-"40)
    for i in range(min(10, len(indices))):
        print(f"{i+1:2d}. {all_feature_names[indices[i]]:30s} {importances[indices[i]]:.4f}")

 对新客户进行预测
print("\n 新客户信用评分示例:")
print("-"40)
new_customers = pd.DataFrame([{
    '年龄': 35,
    '年收入(万)': 50,
    '工作年限': 8,
    '信用卡数量': 2,
    '负债收入比': 0.3,
    '历史逾期次数': 0,
    '房产情况': '有房产',
    '婚姻状况': '已婚',
    '教育程度': '本科'
}, {
    '年龄': 28,
    '年收入(万)': 20,
    '工作年限': 3,
    '信用卡数量': 5,
    '负债收入比': 0.7,
    '历史逾期次数': 2,
    '房产情况': '无房产',
    '婚姻状况': '未婚',
    '教育程度': '大专'
}])

new_predictions = model.predict(new_customers)
new_probabilities = model.predict_proba(new_customers)[:, 1]

for i, (_, customer) in enumerate(new_customers.iterrows()):
    risk_level = "高风险" if new_predictions[i] == 1 else "低风险"
    print(f"客户{i+1}: 违约概率={new_probabilities[i]:.2%}, 风险等级={risk_level}")

return model, X_test, y_test, y_pred, y_pred_proba
```

四川省注册会计师协会信息化委员会

```
def generate_cost_data(self, n_samples=500):
    """
    生成成本动因分析模拟数据

    Args:
        n_samples: 样本数量

    Returns:
        DataFrame: 包含成本动因和总成本的数据
    """
    print("\n 生成成本动因分析数据...")

    data = pd.DataFrame({
        '产量(万件)': np.random.uniform(1, 100, n_samples),
        '订单数量': np.random.poisson(200, n_samples).clip(10, 500),
        '运输距离(km)': np.random.exponential(300, n_samples).clip(10, 1000),
        '原材料价格指数': np.random.normal(100, 20, n_samples).clip(60, 140),
        '直接人工工时': np.random.exponential(1000, n_samples).clip(100, 3000),
        '机器运行时间(小时)': np.random.exponential(500, n_samples).clip(50, 1500),
        '能耗(千瓦时)': np.random.exponential(10000, n_samples).clip(1000, 30000),
    })

     生成总成本（多元线性关系+噪声）
    data['总成本(万元)'] = (
        5.0 +      固定成本
        0.8   data['产量(万件)'] +      变动成本
        0.05   data['订单数量'] +      订单处理成本
        0.002   data['运输距离(km)'] +      运输成本
        0.1   data['原材料价格指数'] +      原材料成本
        0.006   data['直接人工工时'] +      人工成本
        0.003   data['机器运行时间(小时)'] +      机器折旧
        0.0001   data['能耗(千瓦时)'] +      能源成本
        np.random.normal(0, 2, n_samples)      随机噪声
    ).clip(10, 150)

    print(f"生成完成: {n_samples} 条记录")
    print(f"总成本统计: 均值={data['总成本(万元)'].mean():.2f}万, "
          f"标准差={data['总成本(万元)'].std():.2f}万")

    return data

def cost_driver_analysis(self, data):
    """
    成本动因分析（多元线性回归）
```

四川省注册会计师协会信息化委员会

```
Args:
    data: 成本数据

Returns:
    回归模型和评估结果
"""
print("\n" + "="60)
print("场景 2: 成本动因分析（多元线性回归）")
print("="60)

 准备特征和标签
X = data.drop('总成本(万元)', axis=1)
y = data['总成本(万元)']

 划分训练集和测试集
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=self.random_state
)

print(f"训练集大小: {X_train.shape[0]} 条记录")
print(f"测试集大小: {X_test.shape[0]} 条记录")

 数据标准化
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)

 训练多元线性回归模型
model = LinearRegression()
model.fit(X_train_scaled, y_train)

 预测
y_pred = model.predict(X_test_scaled)

 评估模型
print("\n 模型评估结果:")
print("-"40)
print(f"R²分数: {r2_score(y_test, y_pred):.4f}")
print(f"均方误差(MSE): {mean_squared_error(y_test, y_pred):.4f}")
print(f"平均绝对误差(MAE): {mean_absolute_error(y_test, y_pred):.4f}")
print(f"均方根误差(RMSE): {np.sqrt(mean_squared_error(y_test, y_pred)):.4f}")

 系数分析
```


四川省注册会计师协会信息化委员会

```
print("\n 成本动因系数分析:")
print("-"40)
coefficients = pd.DataFrame({
    '特征': X.columns,
    '系数': model.coef_,
    '绝对值': np.abs(model.coef_)
}).sort_values('绝对值', ascending=False)

for idx, row in coefficients.iterrows():
    direction = "正向" if row['系数'] > 0 else "负向"
    print(f"{row['特征']:20s} {row['系数']:8.4f} ({direction})")

 可视化特征重要性
plt.figure(figsize=(10, 6))
colors = ['green' if c > 0 else 'red' for c in coefficients['系数']]
plt.barh(coefficients['特征'], coefficients['系数'], color=colors)
plt.xlabel('系数大小')
plt.title('成本动因系数分析')
plt.axvline(x=0, color='black', linestyle='-', linewidth=0.5)
plt.tight_layout()
plt.show()

 实际 vs 预测
plt.figure(figsize=(10, 6))
plt.scatter(y_test, y_pred, alpha=0.6, edgecolors='k')
plt.plot([y_test.min(), y_test.max()], [y_test.min(), y_test.max()], 'r--', lw=2)
plt.xlabel('实际成本 (万元)')
plt.ylabel('预测成本 (万元)')
plt.title('实际成本 vs 预测成本')
plt.grid(True, alpha=0.3)
plt.tight_layout()
plt.show()

 预测示例
print("\n 成本预测示例:")
print("-"40)
new_data = pd.DataFrame([{
    '产量(万件)': 50,
    '订单数量': 150,
    '运输距离(km)': 200,
    '原材料价格指数': 110,
    '直接人工工时': 1200,
    '机器运行时间(小时)': 600,
    '能耗(千瓦时)': 15000
```

四川省注册会计师协会信息化委员会

```
    }])

    new_data_scaled = scaler.transform(new_data)
    predicted_cost = model.predict(new_data_scaled)[0]
    print(f"预测总成本: {predicted_cost:.2f} 万元")

    return model, X_test, y_test, y_pred, coefficients

def generate_budget_data(self, n_samples=300):
    """
    生成预算报销数据

    Args:
        n_samples: 样本数量

    Returns:
        DataFrame: 报销数据
    """
    print("\n 生成预算报销数据...")

     生成正常数据
    normal_data = pd.DataFrame({
        '报销金额(元)': np.random.lognormal(6, 1.2, n_samples).clip(100, 10000),
        '报销类型': np.random.choice(['差旅费', '办公用品', '业务招待', '交通费', '通讯费'],
                                     n_samples, p=[0.3, 0.2, 0.2, 0.2, 0.1]),
        '月份': np.random.randint(1, 13, n_samples),
        '部门': np.random.choice(['销售部', '技术部', '行政部', '财务部', '市场部'],
                                 n_samples, p=[0.3, 0.25, 0.2, 0.15, 0.1]),
        '员工职级': np.random.choice(['P1', 'P2', 'P3', 'P4', 'P5'],
                                     n_samples, p=[0.2, 0.3, 0.3, 0.15, 0.05]),
        '是否为异常': 0    正常
    })

     生成异常数据
    n_anomalies = int(n_samples    0.1)    10%异常
    anomalies = pd.DataFrame({
        '报销金额(元)': np.random.lognormal(8, 1.5, n_anomalies).clip(5000, 50000),    金额异常
        '报销类型': np.random.choice(['差旅费', '业务招待', '其他'], n_anomalies, p=[0.5, 0.3, 0.2]),
        '月份': np.random.choice([2, 7, 12], n_anomalies),    集中在特定月份
        '部门': np.random.choice(['销售部', '市场部'], n_anomalies, p=[0.7, 0.3]),    集中在特定部门
        '员工职级': np.random.choice(['P1', 'P2'], n_anomalies, p=[0.4, 0.6]),    低职级高报销
        '是否为异常': 1    异常
    })
```

四川省注册会计师协会信息化委员会

```
    合并数据
    data = pd.concat([normal_data, anomalies], ignore_index=True)
    data = data.sample(frac=1, random_state=self.random_state).reset_index(drop=True)

    print(f"生成完成: {data.shape[0]} 条记录")
    print(f"正常报销: {(data['是否为异常']==0).sum()} 条")
    print(f"异常报销: {data['是否为异常'].sum()} 条 ({data['是否为异常'].mean()100:.1f}%)")

    return data

def budget_anomaly_detection(self, data):
    """
    预算异常检测

    Args:
        data: 报销数据

    Returns:
        异常检测模型和结果
    """
    print("\n" + "="60)
    print("场景 3: 预算异常检测")
    print("="60)

    准备特征
    X = data.drop('是否为异常', axis=1)
    y = data['是否为异常']

    编码分类特征
    categorical_cols = ['报销类型', '部门', '员工职级']
    numeric_cols = ['报销金额(元)', '月份']

    创建预处理管道
    preprocessor = ColumnTransformer(
        transformers=[
            ('num', StandardScaler(), numeric_cols),
            ('cat', OneHotEncoder(handle_unknown='ignore', sparse_output=False), categorical_cols)
        ]
    )

    X_processed = preprocessor.fit_transform(X)

    使用 Isolation Forest 进行异常检测
    print("\n 使用 Isolation Forest 进行异常检测...")
```

四川省注册会计师协会信息化委员会

```
 模型训练
iso_forest = IsolationForest(
    n_estimators=100,
    contamination=0.1,    预期异常比例
    random_state=self.random_state
)

 预测异常（-1 表示异常，1 表示正常）
anomaly_labels = iso_forest.fit_predict(X_processed)

 转换为 0/1 标签（0=正常，1=异常）
predictions = (anomaly_labels == -1).astype(int)

 评估（与真实标签比较）
print("\n 异常检测结果评估:")
print("-"40)
print(f"准确率: {accuracy_score(y, predictions):.4f}")
print(f"精确率: {precision_score(y, predictions):.4f}")
print(f"召回率: {recall_score(y, predictions):.4f}")
print(f"F1 分数: {f1_score(y, predictions):.4f}")

 混淆矩阵
cm = confusion_matrix(y, predictions)
print("\n 混淆矩阵:")
print(f"                预测正常    预测异常")
print(f"实际正常    {cm[0,0]:>8}    {cm[0,1]:>8}")
print(f"实际异常    {cm[1,0]:>8}    {cm[1,1]:>8}")

 添加预测结果到数据
data['预测异常'] = predictions
data['异常分数'] = iso_forest.score_samples(X_processed)  -1    负分数表示异常程度高

 可视化异常检测结果
fig, axes = plt.subplots(2, 2, figsize=(14, 10))

 1. 报销金额分布
axes[0, 0].hist(data[data['是否为异常']==0]['报销金额(元)'],
               bins=30, alpha=0.7, label='正常', color='blue')
axes[0, 0].hist(data[data['是否为异常']==1]['报销金额(元)'],
               bins=30, alpha=0.7, label='异常', color='red')
axes[0, 0].set_xlabel('报销金额(元)')
axes[0, 0].set_ylabel('频次')
axes[0, 0].set_title('正常 vs 异常报销金额分布')
```


四川省注册会计师协会信息化委员会

```
axes[0, 0].legend()
axes[0, 0].grid(True, alpha=0.3)

 2. 按部门统计
dept_anomaly_rate = data.groupby('部门')['是否为异常'].mean()
axes[0, 1].bar(dept_anomaly_rate.index, dept_anomaly_rate.values, color='orange')
axes[0, 1].set_xlabel('部门')
axes[0, 1].set_ylabel('异常比例')
axes[0, 1].set_title('各部门异常报销比例')
axes[0, 1].tick_params(axis='x', rotation=45)
axes[0, 1].grid(True, alpha=0.3)

 3. 异常分数分布
axes[1, 0].hist(data[data['是否为异常']==0]['异常分数'],
                bins=30, alpha=0.7, label='正常', color='blue')
axes[1, 0].hist(data[data['是否为异常']==1]['异常分数'],
                bins=30, alpha=0.7, label='异常', color='red')
axes[1, 0].set_xlabel('异常分数（负值表示更异常）')
axes[1, 0].set_ylabel('频次')
axes[1, 0].set_title('异常分数分布')
axes[1, 0].legend()
axes[1, 0].grid(True, alpha=0.3)

 4. 异常检测结果
correct = ((data['是否为异常']==0) & (data['预测异常']==0)).sum() + \
          ((data['是否为异常']==1) & (data['预测异常']==1)).sum()
incorrect = len(data) - correct

labels = ['正确检测', '错误检测']
sizes = [correct, incorrect]
colors = ['lightgreen', 'lightcoral']

axes[1, 1].pie(sizes, labels=labels, colors=colors, autopct='%1.1f%%', startangle=90)
axes[1, 1].axis('equal')
axes[1, 1].set_title(f'异常检测准确率: {accuracy_score(y, predictions):.2%}')

plt.tight_layout()
plt.show()

 展示检测到的异常
print("\n 检测到的前 10 个异常报销:")
print("-"80)
anomalies = data[data['预测异常']==1].sort_values('异常分数').head(10)
for idx, row in anomalies.iterrows():
```

四川省注册会计师协会信息化委员会

```
            print(f"金额: {row['报销金额(元)']:8.2f}元 | "
                  f"类型: {row['报销类型']:8s} | "
                  f"部门: {row['部门']:8s} | "
                  f"职级: {row['员工职级']:5s} | "
                  f"月份: {row['月份']:2d} | "
                  f"异常分数: {row['异常分数']:.4f}")

        return iso_forest, data

    def generate_sales_data(self, n_periods=60):
        """
        生成销售预测时间序列数据

        Args:
            n_periods: 时间期数（月）

        Returns:
            DataFrame: 时间序列数据
        """
        print("\n 生成销售时间序列数据...")

         基础趋势
        time = np.arange(n_periods)
        trend = 0.5   time    线性趋势

         季节性（月度）
        seasonal = 20   np.sin(2   np.pi   time / 12)    年度季节性
        monthly_seasonal = 10   np.sin(2   np.pi   time)    月度波动

         周期性
        cycle = 15   np.sin(2   np.pi   time / 24)    2 年周期

         随机波动
        random = np.random.normal(0, 10, n_periods)

         节假日效应
        holiday_effect = np.zeros(n_periods)
        holiday_months = [1, 5, 10]    1 月（春节）、5 月（五一）、10 月（国庆）
        for month in holiday_months:
            indices = [i for i, t in enumerate(time) if t % 12 == month-1]
            for idx in indices:
                if idx < n_periods:
                    holiday_effect[idx] = 30
```

四川省注册会计师协会信息化委员会

```
        # 促销活动
        promotion_effect = np.zeros(n_periods)
        for i in range(0, n_periods, 6):  # 每 6 个月一次促销
            if i < n_periods:
                promotion_effect[i] = 40
                if i+1 < n_periods:
                    promotion_effect[i+1] = 20

        # 组合所有成分
        base_sales = 100
        sales = base_sales + trend + seasonal + monthly_seasonal + cycle + random + holiday_effect +
promotion_effect
        sales = sales.clip(50, 300)  # 确保在合理范围内

        # 创建特征
        data = pd.DataFrame({
            '时间': time + 1,
            '月份': (time % 12) + 1,
            '季度': ((time % 12) // 3) + 1,
            '年份': (time // 12) + 2020,
            '销售额(万元)': sales,
            '趋势成分': trend,
            '季节成分': seasonal + monthly_seasonal,
            '周期成分': cycle,
            '促销活动': promotion_effect > 0,
            '是否节假日': np.isin((time % 12) + 1, holiday_months)
        })

        print(f"生成完成: {n_periods} 个月的数据")
        print(f"销售额统计: 均值={data['销售额(万元)'].mean():.2f}万, "
              f"标准差={data['销售额(万元)'].std():.2f}万")

        return data

    def sales_forecasting(self, data, forecast_periods=12):
        """
        销售预测

        Args:
            data: 历史销售数据
            forecast_periods: 预测期数

        Returns:
            预测结果
```


四川省注册会计师协会信息化委员会

```
"""
print("\n" + "="60)
print("场景 4: 销售预测")
print("="60)

 创建时间序列特征
df = data.copy()

 滞后特征
for lag in [1, 2, 3, 12]:    1 个月、2 个月、3 个月、12 个月（年同比）滞后
    df[f'销售额_滞后{lag}'] = df['销售额(万元)'].shift(lag)

 移动平均特征
for window in [3, 6, 12]:    3 个月、6 个月、12 个月移动平均
    df[f'销售额_MA{window}'] = df['销售额(万元)'].rolling(window=window).mean().shift(1)

 季节性编码
df['月份_sin'] = np.sin(2   np.pi   df['月份'] / 12)
df['月份_cos'] = np.cos(2   np.pi   df['月份'] / 12)

 删除有 NaN 的行
df = df.dropna()

 准备特征和标签
features = ['月份', '季度', '月份_sin', '月份_cos', '促销活动', '是否节假日',
            '销售额_滞后 1', '销售额_滞后 2', '销售额_滞后 3', '销售额_滞后 12',
            '销售额_MA3', '销售额_MA6', '销售额_MA12']

X = df[features]
y = df['销售额(万元)']

 划分训练集和测试集
split_point = int(len(df)   0.8)
X_train, X_test = X.iloc[:split_point], X.iloc[split_point:]
y_train, y_test = y.iloc[:split_point], y.iloc[split_point:]

print(f"训练集: {len(X_train)} 个月 (前{len(X_train)/12:.1f}年)")
print(f"测试集: {len(X_test)} 个月 (后{len(X_test)/12:.1f}年)")

 标准化特征
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)
```

四川省注册会计师协会信息化委员会

```
 训练多个模型进行比较
models = {
    '线性回归': LinearRegression(),
    '岭回归': Ridge(alpha=1.0),
    '随机森林': RandomForestRegressor(n_estimators=100, random_state=self.random_state),
    '梯度提升': GradientBoostingRegressor(n_estimators=100, random_state=self.random_state)
}

results = {}

print("\n 模型性能比较:")
print("-"60)
print(f"{'模型':<15} {'R²':<10} {'MSE':<10} {'MAE':<10} {'RMSE':<10}")
print("-"60)

for name, model in models.items():
     训练模型
    model.fit(X_train_scaled, y_train)

     预测
    y_pred = model.predict(X_test_scaled)

     评估
    r2 = r2_score(y_test, y_pred)
    mse = mean_squared_error(y_test, y_pred)
    mae = mean_absolute_error(y_test, y_pred)
    rmse = np.sqrt(mse)

    results[name] = {
        'model': model,
        'predictions': y_pred,
        'r2': r2,
        'mse': mse,
        'mae': mae,
        'rmse': rmse
    }

    print(f"{name:<15} {r2:<10.4f} {mse:<10.2f} {mae:<10.2f} {rmse:<10.2f}")

 选择最佳模型
best_model_name = max(results, key=lambda x: results[x]['r2'])
best_model = results[best_model_name]['model']
best_predictions = results[best_model_name]['predictions']
```

四川省注册会计师协会信息化委员会

```
print(f"\n 最佳模型: {best_model_name} (R²  = {results[best_model_name]['r2']:.4f})")

 可视化预测结果
plt.figure(figsize=(14, 8))

 历史数据
plt.plot(range(len(y_train)), y_train.values, 'b-', label='训练数据', linewidth=2)
plt.plot(range(len(y_train), len(y_train) + len(y_test)), y_test.values,
        'g-', label='实际测试数据', linewidth=2)

 预测数据
for name, result in results.items():
    plt.plot(range(len(y_train), len(y_train) + len(y_test)), result['predictions'],
            '--', label=f'{name}预测', linewidth=1.5, alpha=0.8)

plt.axvline(x=len(y_train), color='red', linestyle='--', alpha=0.7, label='预测起始点')
plt.xlabel('时间 (月)')
plt.ylabel('销售额 (万元)')
plt.title('销售预测: 实际 vs 预测')
plt.legend()
plt.grid(True, alpha=0.3)
plt.tight_layout()
plt.show()

 特征重要性（如果模型支持）
if hasattr(best_model, 'feature_importances_'):
    importances = best_model.feature_importances_
    indices = np.argsort(importances)[::-1]

    plt.figure(figsize=(10, 6))
    plt.title('特征重要性')
    plt.bar(range(len(indices)), importances[indices])
    plt.xticks(range(len(indices)), [features[i] for i in indices], rotation=45, ha='right')
    plt.tight_layout()
    plt.show()

 未来预测
print(f"\n 未来{forecast_periods}个月销售预测:")
print("-"40)

 创建未来时间点
last_data = df.iloc[-1:].copy()

future_predictions = []
```

四川省注册会计师协会信息化委员会

```
for i in range(1, forecast_periods + 1):
    创建未来时间特征
    future_month = (last_data['月份'].values[0] + i - 1) % 12 + 1
    future_quarter = ((future_month - 1) // 3) + 1

    这里简化处理，实际应用中需要更复杂的特征工程
    future_features = pd.DataFrame({
        '月份': [future_month],
        '季度': [future_quarter],
        '月份_sin': [np.sin(2  np.pi  future_month / 12)],
        '月份_cos': [np.cos(2  np.pi  future_month / 12)],
        '促销活动': [i % 6 == 0],   假设每 6 个月有促销
        '是否节假日': [future_month in [1, 5, 10]],
        '销售额_滞后 1': [last_data['销售额(万元)'].values[0] if i == 1 else future_predictions[-1]],
        '销售额_滞后 2': [last_data['销售额_滞后 1'].values[0] if i == 1 else (
            last_data['销售额(万元)'].values[0] if i == 2 else future_predictions[-2])],
        '销售额_滞后 3': [last_data['销售额_滞后 2'].values[0] if i == 1 else (
            last_data['销售额_滞后 1'].values[0] if i == 2 else (
            last_data['销售额(万元)'].values[0] if i == 3 else future_predictions[-3]))],
        '销售额_滞后 12': [last_data['销售额_滞后 11'].values[0] if '销售额_滞后 11' in
last_data.columns else last_data['销售额(万元)'].values[0]],
        '销售额_MA3': [last_data['销售额_MA3'].values[0]],
        '销售额_MA6': [last_data['销售额_MA6'].values[0]],
        '销售额_MA12': [last_data['销售额_MA12'].values[0]]
    })

    确保特征顺序一致
    future_features = future_features[features]

    标准化
    future_features_scaled = scaler.transform(future_features)

    预测
    pred = best_model.predict(future_features_scaled)[0]
    future_predictions.append(pred)

    print(f"第{i:2d}个月：预测销售额 = {pred:7.2f} 万元")

可视化未来预测
plt.figure(figsize=(12, 6))

历史数据
historical_months = np.arange(1, len(data) + 1)
plt.plot(historical_months, data['销售额(万元)'], 'b-', label='历史数据', linewidth=2)
```


四川省注册会计师协会信息化委员会

```
        未来预测
        future_months = np.arange(len(data) + 1, len(data) + forecast_periods + 1)
        plt.plot(future_months, future_predictions, 'r--', label='未来预测', linewidth=2, marker='o')

        plt.axvline(x=len(data), color='red', linestyle='--', alpha=0.7, label='预测起始点')
        plt.xlabel('月份')
        plt.ylabel('销售额 (万元)')
        plt.title(f'销售预测：历史与未来{forecast_periods}个月')
        plt.legend()
        plt.grid(True, alpha=0.3)
        plt.tight_layout()
        plt.show()

        return best_model, results, future_predictions

    def run_all_analyses(self):
        """运行所有分析"""
        print("="70)
        print("财务预测与风险评分系统")
        print("="70)

        1. 客户信用评分
        credit_data = self.generate_credit_data(1000)
        credit_model, _, _, _, _ = self.credit_scoring_model(credit_data)

        2. 成本动因分析
        cost_data = self.generate_cost_data(500)
        cost_model, _, _, _, _ = self.cost_driver_analysis(cost_data)

        3. 预算异常检测
        budget_data = self.generate_budget_data(300)
        anomaly_model, _ = self.budget_anomaly_detection(budget_data)

        4. 销售预测
        sales_data = self.generate_sales_data(60)
        sales_model, _, _ = self.sales_forecasting(sales_data, 12)

        print("\n" + "="70)
        print("分析完成！")
        print("="70)
        print("\n 总结:")
        print("-"70)
        print("1. 客户信用评分模型")
```

四川省注册会计师协会信息化委员会

```
        print("    √ 使用随机森林分类器")
        print("    √ 可预测新客户违约风险")
        print("    √ 提供特征重要性分析")
        print()
        print("2. 成本动因分析")
        print("    √ 使用多元线性回归")
        print("    √ 识别关键成本驱动因素")
        print("    √ 可预测不同场景下的成本")
        print()
        print("3. 预算异常检测")
        print("    √ 使用 Isolation Forest 异常检测算法")
        print("    √ 自动识别异常报销单")
        print("    √ 可视化异常分布")
        print()
        print("4. 销售预测")
        print("    √ 使用时间序列特征工程")
        print("    √ 比较多种回归模型性能")
        print("    √ 提供未来 12 个月销售预测")
        print()
        print("所有模型已训练完成，可用于实际财务分析和预测任务。")


def main():
    """主函数"""
     创建财务机器学习系统
    financial_ml = FinancialMLSystem(random_state=42)

     运行所有分析
    financial_ml.run_all_analyses()



if __name__ == "__main__":
     检查必要的库
    try:
        import sklearn
        import pandas as pd
        import numpy as np
        import matplotlib.pyplot as plt
    except ImportError as e:
        print(f"错误：缺少必要的库: {e}")
        print("请安装以下库:")
        print("    pip install scikit-learn pandas numpy matplotlib seaborn")
        exit(1)
```

四川省注册会计师协会信息化委员会

运行主程序

```
main()
```

四川省注册会计师协会信息化委员会

七、beautifulsoup —— “财务数据自动采集器”

是什么：解析 HTML/XML 网页，提取其中的数据。

会计类比：
- 好比一个 自动抓取机器人，能从外部网站（如央行汇率、上交所公告）把数据摘下来，填入你的财务模型。

财务场景：
- 自动抓取每日人民币中间价，用于外币折算。
- 爬取上市公司年报 PDF 中的关键财务指标（如果网页展示）。
- 获取同行企业披露的销售数据，用于标杆分析。

> 注意遵守网站的 robots 协议，仅用于内部非营利分析。

```python
from bs4 import BeautifulSoup
import requests
r = requests.get("http://example.com/汇率")
soup = BeautifulSoup(r.text, "html.parser")
rate = soup.find("span", id="usd_cny").text
```

四川省注册会计师协会信息化委员会

八、all-minilm-l6-v2 —— “财务文本语义理解器”

是什么：一个轻量级的句子嵌入模型（属于自然语言处理），可以将文字转换成向量，用于判断两段文本是否语义相似。

会计类比：
- 类似于一个 智能会计科目匹配引擎：它能理解“购办公用品”和“买打印纸”其实是同一类费用，而不仅仅是关键词匹配。

管理会计/财务场景：
- 费用摘要自动归类：将报销单中的摘要（如“付××公司咨询费”）自动映射到“咨询费”科目，处理模糊描述。
- 合同关键条款比对：找出两份合同中的付款条件是否相似。
- 审计发现辅助（虽然您要求避免审计，但这里可改用于内控自评）：将员工填报的“问题描述”与标准风险库进行相似度匹配。

```python
from sentence_transformers import SentenceTransformer
model = SentenceTransformer('all-MiniLM-L6-v2')
emb1 = model.encode("购买办公用纸")
emb2 = model.encode("采购 A4 打印纸")
相似度 = cosine_similarity(emb1, emb2)    接近 0.9，语义相似
```

> 对于大量非结构化的财务文本（发票备注、摘要、合同条款），这个库能帮你实现“半自动”分类。

四川省注册会计师协会信息化委员会

财报语义覆盖检测系统 2026/4/23 14:21:04

文件上传

准则及规定文件

Choose Files no files selected 上传

支持格式: .txt, .pdf, .docx, .doc

财报文件

Choose Files no files selected 上传

支持格式: .txt, .pdf, .docx, .doc

行业类型

通用

参数设置

灵敏度参数 (k值) 1.8

宽松 (1.0) 适中 (2.0) 严格 (3.0)

文本窗口大小 3

精细 (1句) 适中 (3-5句) 宏观 (10句)

最低相似度阈值 0.65

宽松 (0.3) 适中 (0.6) 严格 (0.9)

已上传文件

准则文件	财报文件

控制面板

准备就绪 0%

开始处理 停止 导出报告

结果预览

处理完成后将显示结果预览

© 2026 财报语义覆盖检测系统 | 基于语义对齐与动态噪声门限技术

四川省注册会计师协会信息化委员会

第二部分小结

库	一句话会计类比	典型管理会计/财务分析场景
Flask	把财务模型变成网页工具	内部预算填报系统、成本测算器
pandas/polars	Excel 的超级加强版	部门利润表、合并报表、多维分析
numpy/scipy	财务计算器+统计工具箱	批量折旧、最优生产组合、销售预测
matplotlib/seaborn/pyvis	自动化财务图表与关系图	成本趋势、毛利率热力图、股权结构图
networkx	会计科目网络与勾稽关系	成本分配路径、关联交易网络
scikit-learn	财务预测与风险评分	信用评分、成本动因、异常检测
beautifulsoup	财务数据自动采集器	抓取汇率、同行数据
all-minilm-l6-v2	财务文本语义理解器	费用摘要自动归类、合同比对

> 一句话记住第二部分:
> “这些库就像财务软件的不同模块——有的管数据处理，有的管画图，有的管预测，组合起来就能搭建属于你自己的智能财务分析系统。”

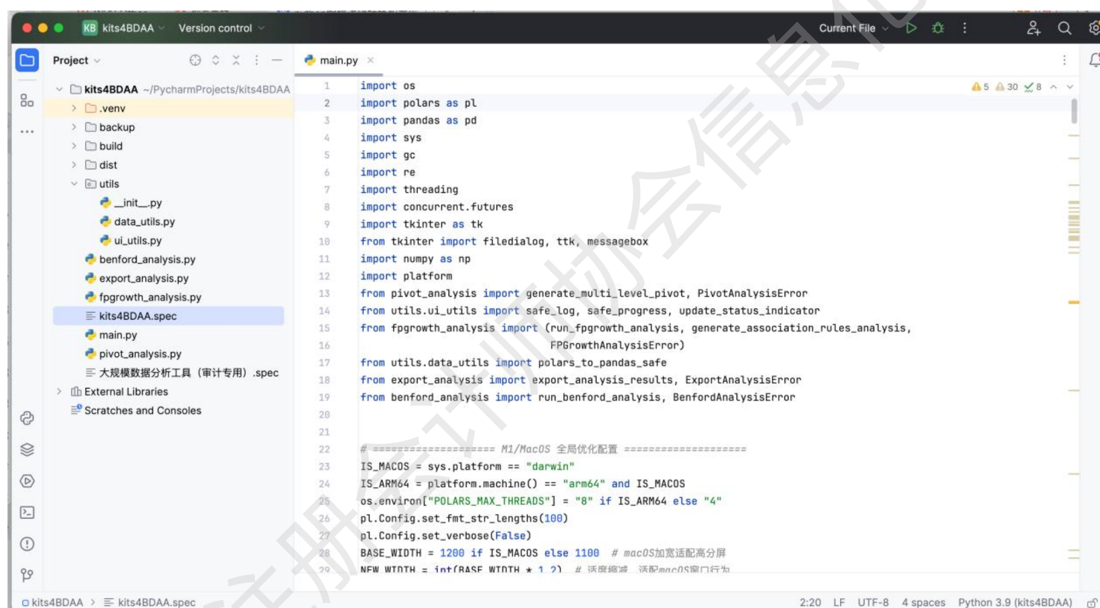
四川省注册会计师协会信息化委员会

第三部分 案例演示—从代码到应用

1. 大规模数据分析

场景：百万级凭证行快速分组汇总、透视分析。
涉及库：pandas / polars

1.1 项目文件结构




```
project_root/
├── main.py                    主入口（图形界面 + 任务调度）
├── utils/
│   ├── data_utils.py          数据清洗、内存计算、数值格式化、Polars→Pandas 安全转换
│   └── ui_utils.py            线程安全的日志、进度条与状态指示器更新
├── pivot_analysis.py          多级透视表生成与综合分析报告
├── fpgrowth_analysis.py       频繁项集挖掘（FP-Growth）与关联规则生成
├── benford_analysis.py        Benford 定律分析（首位数分布 + 自然分布特征检测）
└── export_analysis.py         将各类分析结果导出为 Excel 文件
```

四川省注册会计师协会信息化委员会

1.2 项目功能概览

这是一个面向财务审计数据的智能化分析工具，具有以下核心功能：

1. 数据加载与预览
    - 支持 CSV 和 Excel（多 Sheet）文件读取，采用 Polars 加速大数据集处理。
    - 自动将日期时间列拆分为“年/月”派生列。
    - 提供 Schema 查看（字段类型、非空率、唯一值统计）和前 20 行数据预览。

2. 多级透视表
    - 按用户指定的分组列（支持多级）和聚合列（支持多个数值列）生成透视表。
    - 自动清洗聚合列中的非法字符（如货币符号），转为浮点数进行求和。
    - 可生成综合分析报告，包含各级别统计、Top N 排名、数据质量检查与审计建议。

3. FP - Growth 关联分析
    - 对选定的分类列进行事务化处理, 调用 mlxtend 的 FP - Growth 算法发现频繁项集。
    - 动态调整支持度，适应不同数据量（采样限制 10 万行）。
    - 基于频繁项集进一步生成关联规则，输出“前因 → 后果”及支持度、置信度、提升度。

4. Benford 定律分析
    - 对数值列计算首位数（1~9）的实际频率，与 Benford 理论分布做 χ^2 检验。
    - 结合数据的自然分布特征（对数偏度、峰度、箱分布、首位数字自相关性、唯一值比例等）综合评估数据的真实性。
    - 给出“异常分数”“自然度分数”以及“风险等级”和详细结论。

5. 结果导出
    - 将透视表、频繁项集、关联规则、Benford 分析结果合并写入一个 Excel 文件（多个 Sheet），支持行数截断控制。

1.3 代码运行逻辑简述

1. 启动
   `main.py` 创建 `DataAnalysisTool` 类实例，初始化线程池、GUI 布局、取消标志等。界面左侧为文件操作与状态区，右侧为多标签页配置区（透视表、FP - Growth、关联规则、Benford）、进度条、数据展示表格和日志窗口。

2. 数据加载
    - 用户选择文件后，后台线程用 Polars（CSV）或 Pandas（Excel）读取数据，同时将

四川省注册会计师协会信息化委员会

日期时间列拆分为“年份/月份”派生列。
  - 读取成功后更新界面状态，支持再次选择 Excel 的 Sheet 进行加载。

3. 触发分析任务
  - 用户在对应标签页配置参数后点击按钮，主线程将该任务提交给线程池执行。
  - 每个任务内部通过回调函数（进度回调、日志回调、取消回调）与 GUI 通信，实现进度更新、日志记录和任务取消。
  - 分析完成后的结果（DataFrame）通过 `root.after(0, ...)` 安全地传回主线程并更新表格显示。

4. 任务取消与清理
  - 点击“取消当前任务”仅设置 `self.cancel_task = True`，各模块在计算过程中周期检查该标志实现安全退出。
  - 窗口关闭时执行线程池关闭、内存回收。

5. 模块间协作
  - 各分析模块（pivot、fpgrowth、benford）均遵循统一的接口模式：接收 `df`、必要参数和回调函数，返回结构化结果。
  - `export_analysis.py` 则接收各模块的产出结果，统一写入 Excel。

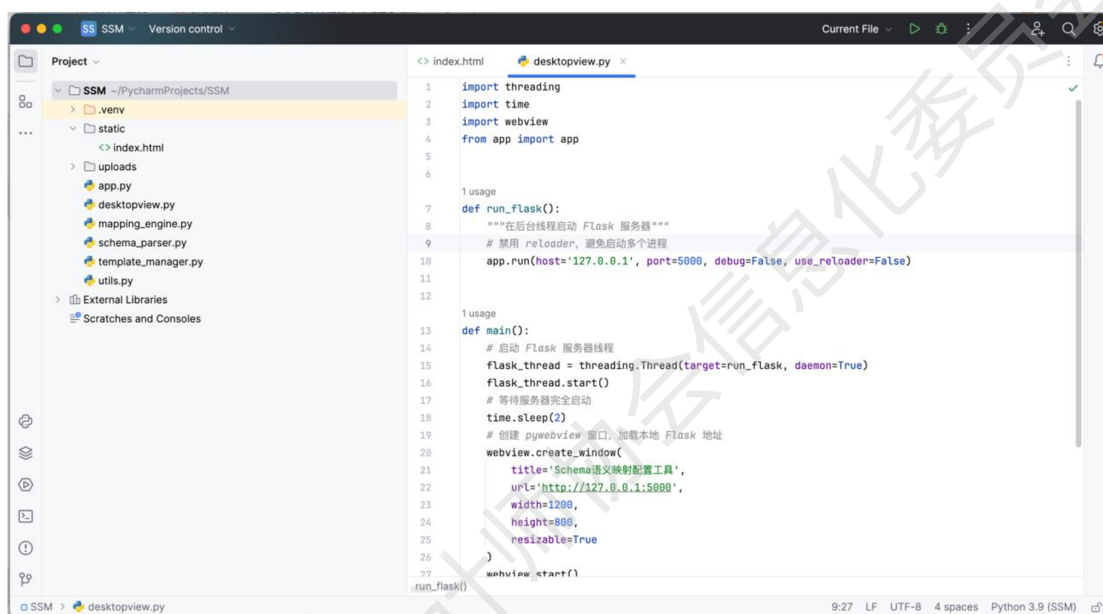
整体上，该项目采用多线程 + 回调解耦的设计，将耗时的数据分析逻辑放在后台执行，保持了 GUI 的响应性，同时通过 Polars 和惰性计算优化了大数据场景下的内存与性能。

四川省注册会计师协会信息化委员会

2. 拖拽映射

场景：通过图形界面拖拽，将源表字段映射到目标科目，自动生成转换脚本。
涉及库：flask

2.1 项目文件结构



根据提供的文件，该项目的文件结构如下：

project_root/
├── app.py　　Flask 应用入口（API 路由）
├── desktopview.py　　桌面版启动器（pywebview 封装）
├── index.html　　前端界面（拖拽式配置工具）
├── mapping_engine.py　　映射执行引擎（核心逻辑）
├── schema_parser.py　　源文件解析器（读取 Excel/CSV）
├── template_manager.py　　标准模板管理器（加载与管理模板）
└── utils.py　　工具函数（文件读写、编码检测）

2.2 项目功能概览

这是一个 Schema 语义映射配置工具，用于将源数据文件的字段按照标准模板进行重新组织，主要功能包括：

四川省注册会计师协会信息化委员会

1. 双层映射配置
   - Sheet 层映射：指定源 Excel 的哪个 Sheet 对应目标模板的哪个 Sheet。
   - 字段层映射：分两种模式：
     - 拼接映射：将多个源字段拼接成一个目标字段（空格连接）。
     - 独立映射：每个源字段作为独立列输出到目标 Sheet（列名格式：`目标字段__源字段`）。

2. 可视化拖拽操作
   - 前端以树形结构展示源 Schema 和目标 Schema。
   - 支持拖拽源字段到目标字段（拼接映射），或先选中目标字段再拖拽到“非拼接区”（独立映射）。
   - 实时显示映射状态（Badge 标注已映射的源字段，列表展示独立映射规则）。

3. 模板与源数据加载
   - 上传标准模板（Excel），自动解析每个 Sheet 的字段列表。
   - 上传源文件（Excel/CSV），自动解析 Schema 并保存完整数据供后续映射使用。

4. 执行映射与导出
   - 根据配置的所有规则，由后端引擎生成一个新的 Excel 文件，按目标 Sheet 输出，直接提供下载。

5. 桌面应用支持
   - `desktopview.py` 使用 `pywebview` 将 Web 工具打包成独立桌面窗口，无需手动打开浏览器。

2.3 代码运行逻辑简述

1. 启动方式
   - 直接运行 `app.py` 启动 Flask 开发服务器，访问 `http://127.0.0.1:5000` 使用。
   - 或运行 `desktopview.py`，后台启动 Flask 线程，并弹出桌面窗口加载该地址。

2. 加载模板
   - 用户在前端选择模板文件，`POST /api/upload_template`。
   - `TemplateManager` 解析 Excel，返回 `{Sheet 名: [字段列表]}`，前端渲染目标树。

3. 加载源文件
   - 用户选择源文件，`POST /api/upload_source`。
   - `SchemaParser` 解析文件（Excel 多 Sheet 或 CSV 单 Sheet），返回 schema 并保存源数据 DataFrame 到全局 `current_source_data`。

4. 配置 Sheet 映射
   - 点击“配置 Sheet 映射”打开模态框，为每个源 Sheet 选择对应的目标 Sheet，结

四川省注册会计师协会信息化委员会

果保存在前端全局变量 `sheetMapping` 中，并显示在状态栏。

5. 配置字段映射（拖拽）
- 拼接映射：从源树拖拽字段到目标树的某个字段上。前端检查 Sheet 映射是否匹配，若匹配则将映射关系存入 `mappingRules`（可多个源字段对应同一目标字段）。
- 独立映射：先单击选中目标字段，再将源字段拖入“非拼接标记区”。映射存入 `nonJoinFields`。
- 两种映射互斥：同一源字段只能出现在一种映射中，拖拽时会自动移除另一种。

6. 执行映射
- 点击“执行映射”，前端将 `sheet_mapping`、`mapping_rules`、`non_join_mapping` 等发送到 `POST /api/execute_mapping`。
- `MappingEngine` 核心逻辑：
  - 标准化规则（确保每个字段的规则为列表）。
  - 根据 `sheet_mapping` 构建目标 Sheet → 源 Sheet 列表的映射。
  - 遍历每个目标 Sheet：
    - 对每个关联的源 Sheet，分别处理该 Sheet 下的拼接规则（调用 `_build_join_column` 将多个源字段拼接成单列）和独立规则（调用 `_build_nonjoin_columns` 为每个源字段生成独立列，列名带前缀）。
    - 收集所有列组成 DataFrame，最后将所有源 Sheet 的 DataFrame 纵向拼接（`concat`），得到该目标 Sheet 的最终数据。
  - 将所有目标 DataFrame 写入一个新的 Excel 文件（`result.xlsx`），并以附件形式返回给前端下载。

7. 异常处理与重置
- 若 Sheet 映射未配置或字段映射不匹配，前端会弹出提示阻止操作。
- 可清空字段映射规则（保留 Sheet 映射）或重置所有规则。

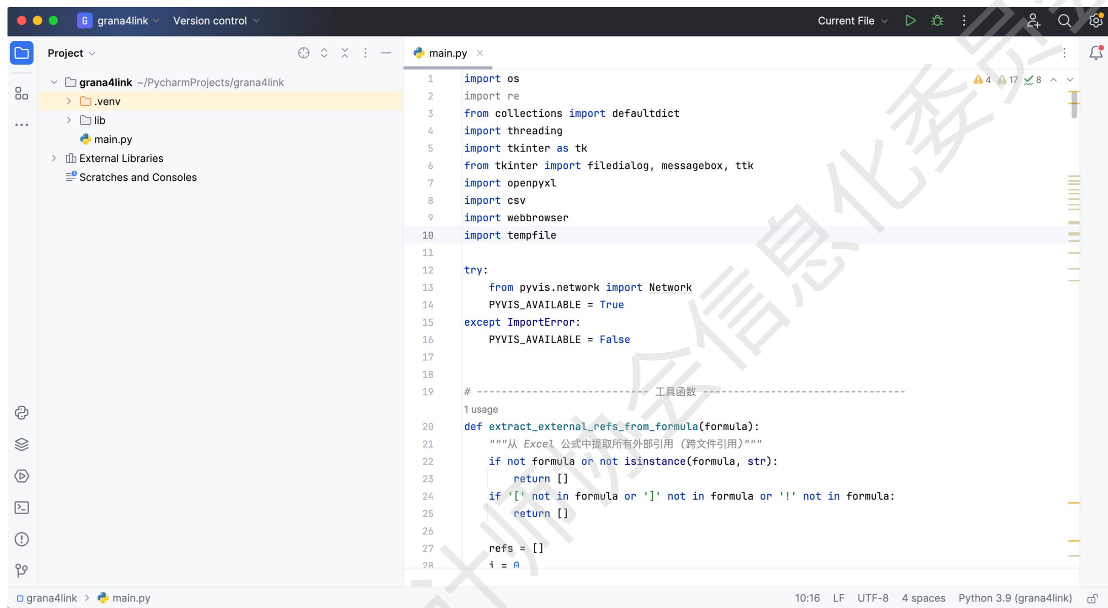
整体设计遵循前后端分离模式：前端负责交互配置，后端负责解析文件与执行映射，通过 REST API 传递 JSON 规则和文件数据，最终产出符合标准模板结构的结果文件。

四川省注册会计师协会信息化委员会

3. Excel 公式链接图谱

场景：解析 Excel 工作表中的公式依赖关系，生成跨表引用的网络图，帮助检查公式错误和循环引用。
涉及库：openpyxl / networkx / pyvis

3.1 项目文件结构




```
项目文件夹/
├── main.py            唯一源码文件，包含所有逻辑
└── (可选依赖)        openpyxl, pyvis (可选), tkinter (内置)
```

代码结构上分为几个部分：
- 工具函数：`extract_external_refs_from_formula()` 从公式字符串解析外部引用。
- 辅助判断：`is_excel_file()` 判断是否为 Excel 文件，`IGNORE_PREFIXES` 定义忽略的临时文件前缀。
- 主界面类 `LinkAnalyzerGUI`：包含界面构建、分析逻辑、结果展示、图谱生成、导出等全部功能。
- 入口 `main()`：创建 Tkinter 根窗口并启动 GUI。

3.2 项目功能概览

四川省注册会计师协会信息化委员会

1. 文件夹扫描
　　递归遍历用户指定的底稿文件夹，收集所有 Excel 文件（`.xlsx/.xlsm/.xlsb`）并记录路径（处理重名警告）。

2. 外部引用解析
　　利用 `openpyxl` 以只读公式模式打开每个工作簿，遍历所有单元格，提取公式中引用的外部文件和工作表（形如 `[其他文件.xlsx]Sheet1!A1`）。

3. 断链检测
　　将外部引用中的目标文件名与已知文件列表比对：若目标文件不存在于当前文件夹，则标记为断链，并记录源文件、目标文件、单元格位置和公式片段。

4. 影响分析
　　统计每个文件被哪些其他文件引用（`impacted_by`），按被引用次数排序显示，直观呈现关键依赖文件。

5. 详细依赖清单
　　列出所有外部引用关系（无论是否断链），包含源文件、源工作表、目标文件、目标工作表、单元格、公式片段及目标是否存在。

6. 交互式图谱生成（需 pyvis）
　　- Workbook 层图谱：以文件为节点，聚合所有跨文件引用，边权重为引用次数，生成有向网络图并在浏览器中打开。
　　- Sheet 层图谱：以“文件::工作表”为节点，展示更细粒度的引用关系。

7. 结果导出
　　将所有引用关系（含目标是否存在）导出为 UTF－8 编码的 CSV 文件。

8. 进度与日志
　　界面底栏显示当前分析的文件名和进度条，日志标签页实时输出分析过程，即使处理大量文件也能了解当前状态。

四川省注册会计师协会信息化委员会

3.3 代码运行逻辑简述

1. 启动：运行 `main.py`，创建 `Tk` 窗口并初始化 `LinkAnalyzerGUI` 实例，构建界面（文件夹选择、按钮、多标签页结果区、日志区、进度显示）。

2. 开始分析：用户选择文件夹并点击“开始分析”→ 启动后台线程执行 `scan_worker()`，避免阻塞界面。

3. 扫描与分析（`analyze` 方法）：

  - `os.walk` 遍历目录，过滤出 Excel 文件并建立 `{文件名: 绝对路径}` 字典，同时记录存在性集合。

  - 遍历每个文件，调用 `analyze_workbook`：

    - `openpyxl.load_workbook(file, data_only=False)` 保留公式。

    - 逐单元格检查公式，用 `extract_external_refs_from_formula` 提取外部引用，收集为 `(源文件，源工作表，目标文件，目标工作表，单元格位置，公式片段)`。

    - 同时更新 `impacted_by[目标文件]` 添加源文件。

  - 检测断链：如果目标文件名不在 `file_exists` 集合中，则记录为断链。

  - 通过回调 `progress_callback` 实时更新界面进度。

4. 结果呈现（`update_gui`）：

  - 将断链数据填入“断链清单”表格；

  - 将被引用文件及其源文件列表填入“影响分析”表格；

  - 将所有引用关系填入“详细依赖”表格，并标注目标是否存在；

  - 启用导出按钮和图谱按钮，更新状态栏。

5. 图谱生成：点击按钮后，使用 `pyvis` 构建 `Network` 对象，添加节点和边，保存为临时 `.html` 文件并用默认浏览器打开。

6. 导出：将全部引用关系写出 CSV，带 BOM 头以兼容 Excel 打开。

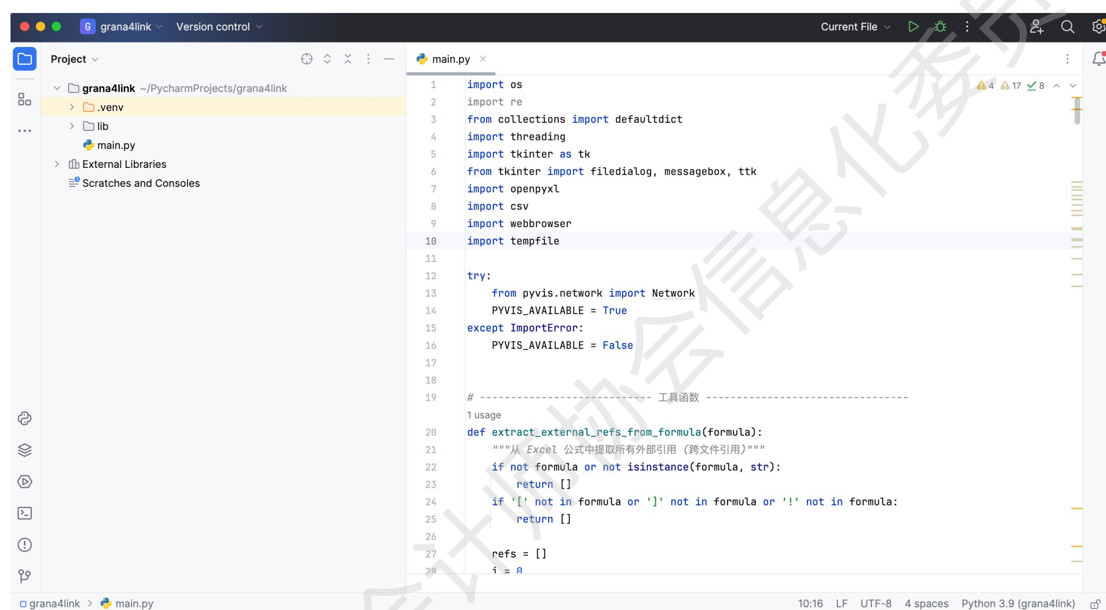
整个工具基于 `tkinter` 构建 GUI，将繁重的文件解析放在后台线程，利用 `openpyxl` 解析公式，实现了审计场景下对 Excel 底稿间链接依赖的快速梳理与断链定位。

四川省注册会计师协会信息化委员会

4. XML 解析（标准会计信息系统数据）

场景：解析金蝶、用友等财务软件导出的 XML 格式凭证数据，提取科目、金额、辅助核算等信息。
涉及库：xml.etree.ElementTree / pandas

4.1 项目文件结构



项目文件夹/
└─── xml2xlsz.py　　全部功能：GUI、XML 解析、数据转换、Excel 导出

依赖库：`tkinter`（内置）、`pandas`、`openpyxl`、`xml.etree.ElementTree`、`chardet`、`codecs`。

四川省注册会计师协会信息化委员会

4.2 项目功能概览

1. 批量读取
   自动扫描指定文件夹下所有 XML 文件（按“公共档案类”文件优先处理），支持自动编码检测（chardet）或手动指定编码，并带有错误忽略模式应对损坏文件。

2. XML 数据提取

按照标准解析 XML 中的主要模块：

- 会计科目
- 科目余额及发生额
- 科目辅助核算
- 现金流量项目
- 记账凭证
- 现金流量凭证项目数据
- 公共档案类（辅助核算档案值、名称等）

3. 科目编码拆分

从 XML 总账基础信息中读取或使用默认的科目编号规则（如 `4-2-2-2-2-2`），将科目余额中的科目编号拆分为多级编码，并标记末级科目。

4. 余额方向标准化

根据会计科目档案中的余额方向，统一将借方/贷方方向的余额转化为带符号数值（如资产类借贷方向反转），重新计算期初、期末余额的“方向”与绝对值，确保借贷方金额可以直接代数求和。

5. 多维度聚合与透视

- 对科目余额先按辅助项（辅助项 1~7 编号名称）进行分组，再按会计期间聚合，生成包含期初、借方、贷方、期末余额的汇总行。
- 对纯科目（无辅助项或辅助项全空）直接按科目编号和会计期间聚合。
- 使用 `pivot_month` 将各科目每个月的借方/贷方本币金额展开为 1~12 月的列，方便横向对比。

6. 辅助核算关联填充

通过公共档案类文件提取的“档案值编码→档案值名称”映射，为科目余额汇总表和记账凭证中的辅助项编号列自动添加对应的名称列，并在记账凭证中新增各辅助核算类型列并填充实际值。

7. 凭证组合与追溯

- 为记账凭证生成“组合凭证号”（会计期间+凭证类型+凭证号）。
- 对每个 4 位一级科目，生成以该科目为核心的“凭证整理”工作表，展示该科目相关的所有凭证分录，便于科目审计追溯。

8. Excel 输出

每个 XML 文件生成一个 Excel 工作簿，包含多个 Sheet：

-`会计科目`、`科目余额`、`科目余额汇总`、`辅助核算`、`现金流量项目`、`现金流量凭证信息`、`记账凭证`、`数据统计`，以及各一级科目的 `{科目名称}凭证整理`。

所有列宽自动调整，数值列格式保留，日期等字段按原样输出。

9. 日志与进度监控

界面底部日志窗口实时显示处理步骤、提取记录数、匹配统计信息，进度条反映文件级

四川省注册会计师协会信息化委员会

处理进度。

4.3 代码运行逻辑简述

1. 启动 GUI
   `main()` 创建 `tk.Tk` 窗口并实例化 `LedgerXMLConverter`，构建包含文件夹选择、编码设置、转换按钮、进度条和日志区的界面。

2. 开始转换
   用户选择输入文件夹（含 XML）和输出文件夹后，点击“开始转换”。
   - 程序获取所有 `.xml` 文件，按文件名排序，并确保 `公共档案类.xml` 最先处理。
   - 首先处理公共档案类文件，调用 `extract_archival_data` 提取所有辅助核算的档案值编码、名称和描述，存入 `self.files_df`，供后续查找映射使用。

3. 单个文件处理（`convert_ledger_xml`）
   - 编码解析：根据用户选择或自动检测编码，多种回退策略（BOM 移除、多编码尝试、错误忽略模式）读取 XML 内容并解析为 ElementTree。
   - 科目编号规则提取：查找 XML 中的 `总账基础信息/会计科目编号规则` 获得拆分规则，默认 `4-2-2-2-2-2`。
   - 数据提取：遍历并收集所有 `会计科目`、`科目余额及发生额`、`科目辅助核算`、`现金流量项目`、`记账凭证`、`现金流量凭证项目数据` 子元素，构建为 pandas DataFrame。
   - 科目余额处理：
     调用 `split_subject_codes` 拆分科目编码，标记级次和末级科目。
     调用 `aggregate_balances` 进行聚合：
      - 对末级科目按辅助项分组，再用 `period_sort` 调整余额方向并聚合到月份，然后用 `pivot_month` 展开为 12 个月列，最后按科目编号聚合；
      - 对非末级科目直接按期间排序、聚合并展开月份列。
     所有聚合过程中，若存在辅助项编号，则从 `files_df` 查找对应的辅助值名称。
   - 记账凭证处理：
     为凭证添加科目名称、辅助项名称列（通过 `add_subject_name` 和 `add_auxiliary_columns_and_fill`）。
     添加组合凭证号列。
     对每个一级科目（4 位编码）调用 `gj_fx` 提取相关凭证，生成单独的整理表。
   - 现金流量处理：为 `现金流量凭证项目数据` 添加现金流量项目名称和报表项目名称。

4. 写入 Excel
   使用 `pd.ExcelWriter` 将所有 DataFrame 写入不同 Sheet，并调用 `auto_adjust_columns` 调整列宽。最后添加一个“数据统计” Sheet，记录各模块记录数。

5. 进度更新

四川省注册会计师协会信息化委员会

每处理完一个文件更新进度条，日志窗口实时输出提取数量、匹配情况等；全部文件处理完毕后弹出完成对话框。

整个工具通过 分层解析 → 多维度聚合 → 关联映射填充 → 透视与追溯 的链条，将复杂的总账 XML 自动转化为审计人员可直接使用的财务分析 Excel 报表。

四川省注册会计师协会信息化委员会

5、语义对齐（基于 nlp）

场景：在财务审计、数据分析或系统迁移中，面对来自不同子公司、业务系统或历史账套的杂乱交易描述、费用摘要或会计科目名称，需要将其智能、准确地映射到统一的标准科目体系。例如，将“付 xx 办公用品款”、“购买打印纸及墨盒”自动归类到“管理费用-办公费”，解决因描述不一带来的手工对账耗时长、合并报表编制难、审计抽样效率低等问题。

涉及库：

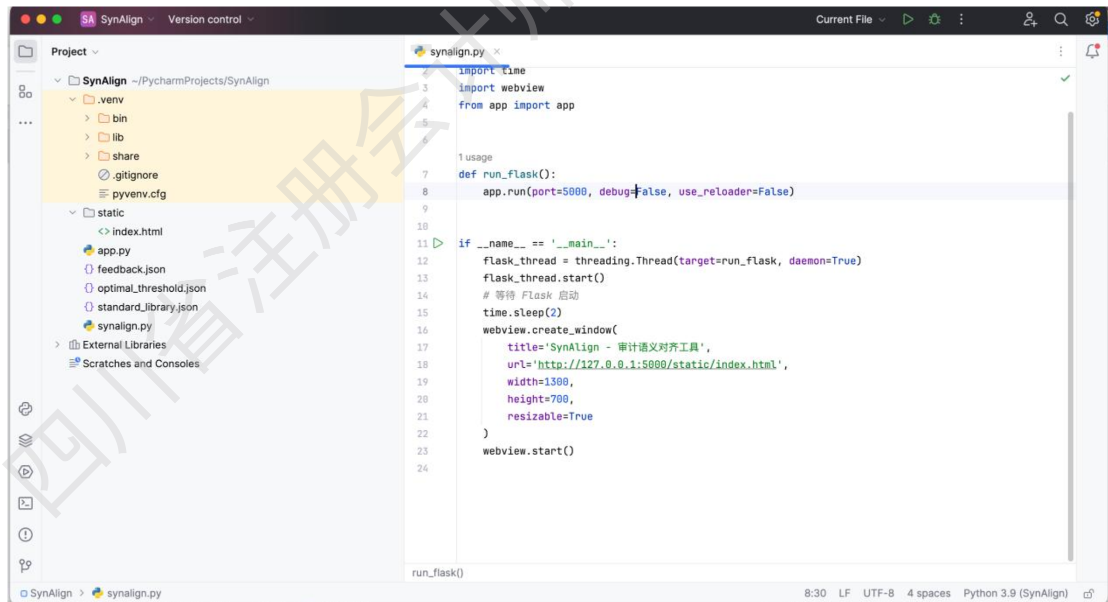
• 核心 NLP 模型库：sentence-transformers （特别是轻量级模型 all-MiniLM-L6-v2），用于将文本转换为蕴含语义的向量。

• 数据处理与计算：pandas（组织标准科目与待匹配数据）、numpy（向量计算）。

• 相似度计算：scikit-learn 中的 cosine_similarity 函数，或直接使用 numpy 计算余弦相似度。

• 中文分词：jieba，用于对中文描述进行分词预处理，可能提升语义理解精度。

5.1 项目文件结构



该项目的文件结构如下：

SynAlign/
├── app.py　　　　Flask 后端服务（语义匹配、标准库管理、反馈与阈值校准）
├── index.html　　　前端交互界面

四川省注册会计师协会信息化委员会

```
├── synalign.py              桌面应用启动器（pywebview 包装）
├── standard_library.json       （自动创建）标准语句库，JSON 文件
├── feedback.json                （自动创建）用户反馈记录
└── optimal_threshold.json      （自动创建）当前最优阈值
```

请注意 `index.html` 实际需放置在 `static/` 子目录下，才可通过 `/static/index.html` 访问。

5.2 项目功能总览

SynAlign — 审计语义对齐工具，是一款基于句子嵌入的智能辅助系统，用于将审计人员输入的自由文本（如审计底稿中的措辞、检查结论等）与预设的“标准语义库”进行相似度匹配，并依据匹配阈值筛选出最相近的标准语句。

主要功能包括：

1. 标准语义库管理
   - 左侧栏可添加、编辑、删除标准语句。
   - 预加载了示例审计常用表述（如“经盘点，账实相符”等），并持久化保存为 JSON 文件。

2. 文本输入与文件上传
   - 直接在文本区输入待对齐的审计措辞。
   - 支持上传 `.txt`、`.csv`、`.xlsx` 文件，后端自动提取文本内容（有长度限制，提升速度）。

3. 语义相似度匹配
   - 使用多语言句子向量模型 `paraphrase-multilingual-MiniLM-L12-v2` 计算用户输入与标准库中所有语句的余弦相似度。
   - 返回相似度最高的前 10 条，并按当前阈值过滤后展示。

4. 动态阈值控制
   - 提供滑块手动调整阈值（0~1），可即时应用。
   - 根据用户的“正确/错误”反馈，自动校准阈值（寻找使反馈分类准确率最高的相似度分界点）。
   - 每次反馈后自动更新阈值并持久化。

5. 用户反馈闭环
   - 每条匹配结果旁有“正确”和“错误”按钮。
   - 点击后记录反馈（用户文本片段、匹配标准语句、相似度、标签），存入 `feedback.json`。
   - 统计反馈总数，支持一键导出反馈记录为 CSV。

四川省注册会计师协会信息化委员会

6. 桌面应用模式

- `synalign.py` 利用 `pywebview` 创建原生窗口，内嵌 Flask 后端页面，无需单独打开浏览器。

5.3 代码运行逻辑简述

1. 启动

- 直接运行 `app.py` 启动 Flask 服务（端口 5000），访问 `http://127.0.0.1:5000/static/index.html` 使用。
- 或运行 `synalign.py`，后台线程启动 Flask，延迟 2 秒后用 `pywebview` 打开同地址，体验如桌面软件。

2. 后端初始化

- 加载 SentenceTransformer 模型（首次下载到本地并缓存）。
- 从 `standard_library.json` 加载标准语句表，计算所有语句的向量，并存储为 Tensor 缓存，加速后续匹配。
- 从 `feedback.json` 和 `optimal_threshold.json` 加载历史反馈与当前阈值。

3. 前端交互

- 页面加载后，`refreshStdLibrary()` 请求 `/api/std_library` 获取标准库，渲染到左侧。
- 阈值滑块显示当前值，反馈计数也通过 `/api/feedback/count` 实时更新。

4. 语义对齐

- 用户点击“开始语义对齐” → 发送 `POST /api/match`，携带输入文本。
- 后端编码用户输入为向量，与预缓存的标准向量计算余弦相似度，取 Top-10，应用当前阈值筛选，返回 `matches`。
- 前端渲染表格，每行显示相似度百分比、标准语句和反馈按钮。

5. 反馈与阈值更新

- 用户点击“正确/错误” → 前端 `submitFeedback` 调用 `POST /api/feedback`。
- 后端追加反馈记录，然后调用 `update_threshold()` 遍历所有反馈的相似度值，找到分类准确率最高的阈值，写入 `optimal_threshold.json`，同时返回新阈值给前端。
- 手动调整滑块并点击“应用”，调用 `PUT /api/threshold`，直接覆盖阈值文件。
- “自动校准”按钮触发 `POST /api/threshold/recalibrate`，基于已有反馈重新计算最优阈值。

6. 文件处理

- 上传文件后，后端根据扩展名读取内容：txt 直接解码；csv 和 xlsx 用 pandas 读取并拼接成字符串，限制总长 2000 字符。
- 返回拼接后的文本填入输入框，继续语义对齐。

7. 数据持久化
   - 标准库、反馈、阈值均以 JSON 文件保存在项目根目录，重启后数据不丢失。
   - 反馈导出为 CSV（`/api/feedback/export`），带 BOM 头，方便用 Excel 打开。

整个工具将句子嵌入与反馈驱动的阈值优化相结合，实现对审计措辞的快速标准化映射，帮助审计人员发现措辞是否符合准则或常见规范，并随着使用逐渐改善匹配准确率。

四川省注册会计师协会信息化委员会

附录：数据安全、伦理与职业准则

在用 Python 工具解决财务问题时，请始终将本附录中的准则置于首位。技术是放大器，但职业操守和法律责任是基石。

一、核心原则

1. 保密性原则：您通过编程处理的财务数据，与您手工处理的账簿、报表具有同等的保密级别。保护数据安全就是保护商业机密和客户隐私。
2. 最小化原则：仅访问和处理完成特定分析任务所必需的最小数据集。不必要的访问就是风险。
3. 合法性原则：确保数据获取途径、处理方式及用途完全符合法律法规、公司制度和职业道德规范。
4. 复核与留痕原则：自动化程序的输出不能替代您的职业判断。任何由代码生成的结果，在用于重要决策或审计证据前，都必须经过适当的人工复核与验证。关键的数据处理步骤应留有日志。

二、具体场景下的准则

1. 数据获取环节

- 内部数据：仅从授权渠道（如公司数据库、经批准的备份文件）获取数据。严禁未经授权访问、复制或导出敏感财务信息。

- 外部数据：

  ◦ 使用 BeautifulSoup 等工具进行网络数据采集时，必须严格遵守目标网站的 robots.txt 协议和服务条款。禁止对网站进行恶意爬取或造成负荷攻击。

  ◦ 用于分析的公开数据（如市场数据），应确认其许可协议允许用于您的分析目的。

- 个人数据：处理包含个人信息（如员工薪酬、客户信息）的数据时，必须进行脱敏处理，并严格遵守《个人信息保护法》等相关法规。

2. 数据处理与分析环节

四川省注册会计师协会信息化委员会

• 环境安全：确保您编写和运行代码的计算机环境安全（如设置密码、安装防病毒软件）。避免在公共电脑或未加密的 U 盘中处理敏感数据。

• 代码安全：不要在代码中硬编码数据库密码、API 密钥等敏感信息。应使用环境变量或配置文件进行管理，并确保这些配置文件不被上传至公开的代码仓库（如 GitHub）。

• 模型与算法的审慎性：

◦ 使用 scikit-learn 等库构建的预测模型，其输出是概率性参考，而非确定性结论。尤其在信用评分、风险预警等场景，模型结果必须结合业务上下文进行解读，并建立人工复核机制。

◦ 理解模型的局限性（如数据偏见可能导致结果偏差），避免盲目信任"黑箱"输出。

3. 结果输出与保存环节
• 输出物管控：自动化生成的报告、分析结果等输出物，其传播范围应受到与手工报告同等的内部控制。

• 数据清理：在个人设备或临时环境中完成分析任务后，应及时、安全地删除涉及的原始敏感数据。

• 归档与审计轨迹：重要的分析项目，应保留代码、所用数据的版本、以及关键输出，形成完整的、可追溯的"电子审计轨迹"。

三、总结：做一个负责任的技术赋能者

学习本课程的目标，是让您成为一名能驾驭技术的专业财务人士，而非脱离业务的程序员。请时刻用以下问题审视您的技术实践：
• 我的数据来源是否合法、合规？

• 我是否在保护我所处理数据的机密性与完整性？

• 我是否理解并能够解释我所用工具的基本原理与局限？

• 我的自动化结果是否经过适当的控制与复核？

遵循这些准则，Python 技能将成为您职业道路上强大、可靠且负责任的加速器，助您在智能财务时代脱颖而出。

四川省注册会计师协会信息化委员会