Python 技术管理规范

1. 代码风格

**a. 强制**

1. 必须遵循 PEP 8 代码风格指南，包括缩进、命名约定等
2. 必须使用 4 个空格作为缩进，禁止使用制表符
3. 必须使用 UTF-8 编码
4. 每行代码不得超过 100 个字符
5. 导入模块必须分组并按以下顺序：标准库、第三方库、本地应用/库
6. 禁止使用 **from module import \*** 形式的导入
7. 类名必须使用 **CapWords** 命名法
8. 函数名、变量名必须使用 **snake\_case** 命名法
9. 常量必须使用 **ALL\_CAPS** 命名法

**b. 推荐**

1. 推荐使用类型注解（Python 3.5+）
2. 推荐使用 f-string 进行字符串格式化（Python 3.6+）
3. 推荐在复杂表达式周围使用括号以提高可读性
4. 推荐在运算符前后和逗号后添加一个空格
5. 推荐在顶级函数和类定义之间留两个空行
6. 推荐在类方法定义之间留一个空行

**c. 允许**

1. 允许在字符串连接时使用 **+** 运算符（但推荐 f-string 或 .format()）
2. 允许使用 **\_** 前缀表示非公共方法和变量
3. 允许在特殊情况下适当放宽 100 字符的行长限制
4. 代码质量

**a. 强制**

1. 必须为所有公共模块、函数、类和方法编写文档字符串
2. 必须处理可能引发异常的情况，禁止捕获所有异常（**except:**）
3. 必须确保资源（如文件、数据库连接）在使用后正确关闭
4. 必须为单元测试提供有意义的断言消息
5. 禁止在生产代码中使用 **print()** 进行调试，必须使用日志系统

**b. 推荐**

1. 推荐使用 **with** 语句管理资源
2. 推荐使用 **isinstance()** 进行类型检查而非 **type()**
3. 推荐使用列表推导式而非 **map()** 和 **filter()**（简单情况下）
4. 推荐使用 **collections.namedtuple** 或 **dataclasses** 创建简单数据结构
5. 推荐为复杂函数编写类型注解

**c. 允许**

1. 允许在性能关键路径使用 **map()** 和 **filter()**
2. 允许在适当情况下使用 **@staticmethod** 和 **@classmethod**
3. 允许在测试代码中使用 **mock** 模块
4. 工程实践

**a. 强制**

1. 必须使用虚拟环境管理项目依赖
2. 必须使用 **requirements.txt** 或 **Pipfile** 明确记录所有依赖
3. 必须为项目提供清晰的 **README.md** 文件
4. 必须为所有提交提供有意义的提交信息
5. 必须为所有新功能或重大变更编写单元测试

**b. 推荐**

1. 推荐使用 **pre-commit** 工具管理代码质量钩子
2. 推荐使用 **pylint** 或 **flake8** 进行静态代码分析
3. 推荐使用 **black** 自动格式化代码
4. 推荐使用 **isort** 自动排序导入语句
5. 推荐使用语义化版本控制（SemVer）
6. 推荐为 CI/CD 流程编写清晰的脚本

**c. 允许**

1. 允许在适当情况下使用 Jupyter Notebook 进行探索性分析
2. 允许在项目初期使用 **setup.py**，但推荐 **pyproject.toml**（PEP 517/518）
3. 允许在测试代码中使用 **pytest** 的高级特性（如 fixture）
4. 性能与安全

**a. 强制**

1. 必须对用户输入进行验证和清理
2. 必须避免在代码中硬编码敏感信息（如密码、API 密钥）
3. 必须使用参数化查询防止 SQL 注入

**b. 推荐**

1. 推荐使用 **hashlib** 而非自实现哈希算法
2. 推荐在性能敏感代码中使用内置函数和库函数

**c. 允许**

1. 允许在特殊情况下使用 **eval()**，但必须有严格的安全控制
2. 允许在明确需要时使用全局变量
3. 异常处理

**a. 强制**

1. 必须使用特定异常而非裸露的 **except:** 语句
2. 必须在异常处理中包含足够的上下文信息

**b. 推荐**

1. 推荐创建自定义异常类而非使用通用异常
2. 推荐使用 **logging.exception()** 记录异常详细信息

**c. 允许**

1. 允许在适当情况下使用 **try-except-else** 结构
2. 允许在清理操作中使用 **try-finally**
3. 测试

**a. 强制**

1. 必须为所有关键路径编写单元测试
2. 必须确保测试覆盖率不低于 80%（关键模块不低于 90%）

**b. 推荐**

1. 推荐使用 **pytest** 作为测试框架
2. 推荐为测试使用有意义的描述性名称

**c. 允许**

1. 允许在集成测试中使用真实数据库（但推荐 mock）
2. 允许在性能测试中使用时间敏感断言