**Python技术管理规范**

**执行要求**

强制(a)：CI/CD流水线必须拦截违规（如测试失败、类型错误）

推荐(b)：Code Review重点检查（性能、安全、可维护性）

允许(c)：需团队共识并记录例外原因

**1. 代码风格**

a) 使用4个空格缩进（禁用Tab）

a) 每行不超过80字符（复杂表达式可放宽至100）

a) 导入按顺序分组：标准库 → 第三方库 → 本地模块（每组空一行）

b) 使用snake\_case命名变量/函数，CapWords命名类

c) 允许单行简单语句（如if x: return）

**2. 语法规范**

a) 必须用is None/is not None判断空值（禁用== None）

a) 异常捕获需指定类型（禁用裸except:）

b) 优先用f-string格式化字符串（Python 3.6+）

b) 列表推导式避免嵌套超两层

c) 允许在简单逻辑中使用三元表达式x = a if cond else b

**3. 类型与注解**

a) 公共API必须添加类型注解（函数参数、返回值）

b) 复杂类型使用typing模块（如List[Tuple[str, int]]）

c) 允许对局部变量省略类型注解

**4. 工程结构**

a) 项目必须包含requirements.txt或Pipfile

a) 测试目录命名为tests/（与源码同级）

b) 模块按功能拆分（避免>500行的单文件）

b) 使用\_\_init\_\_.py显式定义包导出

c) 允许在小型工具脚本中使用全局变量

**5. 测试与质量**

a) 核心逻辑必须覆盖单元测试（覆盖率≥80%）

a) 使用pytest或unittest框架（禁用nose）

b) 测试用例按test\_<模块>\_<行为>.py格式命名

b) 集成静态检查（flake8 + mypy）

c) 允许# pragma: no cover标记无需覆盖代码

**6. 性能与安全**

a) SQL查询必须使用参数化（禁止字符串拼接）

a) 密码处理用hashlib或bcrypt（禁用MD5/SHA1）

b) 大文件处理使用生成器（yield）

b) 避免eval()/exec()（确需使用时需安全审查）

c) 允许用locals()简化模板渲染（限内部工具）

**7. 面向对象**

a) 类方法按顺序排列：\_\_init\_\_ → @classmethod → 普通方法 → 私有方法

b) 优先组合而非继承（避免超3层继承）

b) 使用@property封装属性校验

c) 允许\_\_slots\_\_优化内存密集型对象

**8. 工具与流程**

a) 必须配置.editorconfig统一编辑器设置

a) 提交前运行black自动格式化

b) 用pre-commit管理Git钩子

b) 文档字符串遵循Google风格（参数/返回值说明）

c) 允许Jupyter Notebook用于探索性分析

**9. 并发处理**

a) I/O密集型优先用asyncio（Python 3.7+）

b) CPU密集型用multiprocessing（避免threading）

b) 共享资源必须加锁（with threading.Lock():）

c) 允许concurrent.futures简化线程池

**10. 兼容性**

a) 新项目必须支持Python 3.9+

b) 弃用特性需添加@deprecated装饰器

b) 跨版本兼容代码用sys.version\_info分支

c) 允许为遗留系统维护Python 2.7代码（需专项审批）