# Python 技术管理规范文档

Python 技术管理规范文档：本规范遵循 Google、PEP8 等权威风格指南，并结合实际开发经验，按三个级别组织：a. 强制 (必须执行)；b. 推荐 (建议执行)；c. 允许 (在某些场景允许)

## a. 强制规则 (Must)

1. 代码必须遵循 PEP8 样式指南
2. 每个 Python 文件必须有 UTF-8 头部声明 (Python3 默认允许省略)
3. 使用 4 个空格缩进
4. 禁止使用混合缩进 (空格 + tab)
5. 不得直接把密码写入代码
6. 每个函数必须有 docstring 文档说明
7. 不得使用 from module import \*
8. 文件名必须下划线分隔（snake\_case）
9. 程序运行输入要通过 main() 函数
10. 代码中不得启用关联无效或未定义的变量
11. 必须按照 isort 样式分类和排序 import
12. 推行前必须通过 lint (pylint/flake8) 检查
13. 运行必须同时通过单元测试 (pytest)
14. 无效代码必须删除，禁止别名 dead code
15. 不得在代码中使用容易造成不清的简写。例如：l 和 O

## b. 推荐规则 (Should)

1. 必要时用 type hint 标示变量类型
2. 使用统一的字段命名样式（snake\_case for 变量/函数名，CamelCase for class）
3. 使用 f-string 考虑性能和阅读性
4. 可重用逻辑应尽量抽象成函数/类
5. 推荐在代码中实现 logging，以便调试
6. 使用统一的问题抓取方式：except要指定类型
7. 不建议在代码中写 TODO，应使用方案跟踪系统
8. 不建议使用过长的函数 (超过 50 行)
9. 不建议一个文件包含过多 class 或功能混合
10. 使用清晰的评论文本，而非未描述的 #
11. 推荐在 CI 流程中实现自动核查、打包、部署
12. 推荐对关键功能实施自动化测试
13. 给公用库或 API 提供文档生成 (e.g., sphinx)
14. 推荐使用虚拟环境管理依赖 (如: venv/pipenv/poetry)
15. 推荐尽量使用 pathlib 和新路径 API
16. 推荐在大型项目中加入 pre-commit 钩子（如检查格式、测试等）
17. 推荐所有异常类以 Error 结尾，并继承自 Exception
18. 推荐团队使用统一的代码格式化工具（如 black）
19. 推荐使用 .editorconfig 统一 IDE 编辑器设置
20. 推荐使用 pyproject.toml 统一项目配置入口

## c. 允许规则 (Can)

1. 允许使用表达式生成器 (如 list comprehension)
2. 允许使用内置函数，如 lambda/被封装函数
3. 允许使用多个 requirements.txt 用于分环境
4. 允许对不同模块或封装使用特殊命名
5. 允许将旧代码用 legacy/ 或 \_old.py 格式隔离
6. 允许在测试文件中使用 print，但不应出现在产品代码
7. 允许使用自定义 Exception 类，前提是统一命名和文档
8. 允许使用多级目录结构管理应用
9. 允许使用 .env 文件进行环境变量配置
10. 允许使用路径相对化工具如 os.path.join 或 pathlib