Python 技术管理规范

分级说明

- **a. 强制**：必须遵守。违反即视为不合规。
- **b. 推荐**：强烈建议遵守，若不遵守需有充分理由。
- **c. 允许**：可以使用，需在合适场景下。

1. 代码风格与格式化

a. 强制

1. **a.1** 代码必须符合 PEP 8，且使用 *black* 自动格式化。
2. **a.2** 行长度不得超过 100 字符（docstring 可到 120）。
3. **a.3** 文件必须使用 UTF-8 编码并显式声明 `# -*- coding: utf-8 -*-`（Python 2 兼容项目除外）。
4. **a.4** 文件末尾必须保留单个换行符。
5. **a.5** 禁止在生产代码中出现 print 调试语句，使用 logging。

b. 推荐

6. **b.1** 使用 *isort* 保持 import 分组顺序：标准库→第三方→本地。
7. **b.2** 使用 4 空格缩进，不使用制表符。
8. **b.3** 对长列表/字典/参数列表使用多行尾随逗号格式。
9. **b.4** 命名采用 Google Python Style：类名 CamelCase，函数/变量 snake_case。
10. **b.5** 使用 f-string 代替 % 或 format() 进行字符串插值。

c. 允许

11. **c.1** 在脚本/REPL 原型阶段可使用 print，但提交前需移除。
12. **c.2** 允许在过渡期内使用 yapf 或公司内部工具，只要生成的结果兼容 black。

2. 结构与组织

a. 强制

13. **a.6** 每个 Python 包必须包含空的 __init__.py。
14. **a.7** 顶层包名不得与标准库冲突。

15. **a.8** 单个模块文件长度不得超过 800 行，函数不得超过 80 行。

b. 推荐

16. **b.6** 使用“src layout”组织源代码，测试放在 tests/。
17. **b.7** 同一层级目录下避免超过 10 个模块文件。
18. **b.8** 对于脚本入口使用 if __name__ == "__main__": 防止意外导入执行。
19. **b.9** 将业务逻辑与 CLI/UI 层分离，便于复用与测试。

c. 允许

20. **c.3** 可在 __init__.py 中暴露公共 API，但禁止执行耗时逻辑。

3. 类型注解

a. 强制

21. **a.9** 新增代码必须全部使用 PEP 484 类型注解并通过 *mypy*（或 pyright）零错误。
22. **a.10** 禁止使用 Any 作为参数或返回类型，除非有 TODO 说明并限期修复。

b. 推荐

23. **b.10** 对已有未注解代码的修改块补齐类型注解。
24. **b.11** 使用 typing.NamedTuple 或 dataclasses 明确数据结构。
25. **b.12** 对外部 API 返回值使用 typing.Protocol 描述鸭子类型接口。

c. 允许

26. **c.4** 在性能敏感热路径可使用 typing.cast 以减少检查误报。

4. 文档与注释

a. 强制

27. **a.11** 所有 public 函数、类、模块必须拥有 Google style docstring。
28. **a.12** 若算法复杂，需在函数体首行写逻辑概要注释。

b. 推荐

29. **b.13** 对复杂正则或位运算提供示例输入输出。
30. **b.14** 使用 pydocstyle CI 检查文档规范。

c. 允许

31. **c.5** 可在临时代码块前使用 # TODO(username): 注释，需附预计完成时间。

5. 测试与质量保证

a. 强制

32. **a.13** 所有业务逻辑需覆盖≥80% 行测试覆盖率并在 CI 中强检查。

33. **a.14** 单元测试使用 pytest 并保持无全局状态。

34. **a.15** CI 必须运行 black - —check, isort - —check, mypy, pytest。

b. 推荐

35. **b.15** 采用 hypothesis 进行属性测试覆盖边界。

36. **b.16** 关键路径性能测试基线随 PR 更新。

c. 允许

37. **c.6** 使用 unittest 兼容旧测试，但逐步迁移到 pytest。

6. 依赖与环境管理

a. 强制

38. **a.16** 生产依赖与开发依赖分离，使用 requirements.txt / requirements-dev.txt。

39. **a.17** 项目必须提供 pip install -e . 方式安装的 pyproject.toml 或 setup.cfg。

40. **a.18** 依赖版本必须锁定精确版本（==）并使用 Dependabot/ Renovate 定期升级。

b. 推荐

41. **b.17** 开发环境使用 pyenv+virtualenv 或 conda 隔离。

42. **b.18** 使用 pip-compile 生成锁文件。

c. 允许

43. **c.7** 对科研/探索性 notebook 可使用未锁定的上游版本，但需在环境说明中注明。

7. 性能与可扩展性

a. 强制

44. **a.19** 禁止在协程内阻塞性 I/O，不得使用 time.sleep 模拟阻塞。

45. **a.20** 生产代码禁止使用 *args, **kwargs 捕获未显式声明的 API。

b. 推荐

46. **b.19** 大数据循环优先使用生成器表达式和惰性迭代。

47. **b.20** 并发场景优先 asyncio，其次 concurrent.futures。

c. 允许

48. **c.8** 允许在极端性能场景使用 Cython 或 Rust 扩展，需文档说明。

8. 安全与可靠性

a. 强制

49. **a.21** 禁止使用 eval, exec 处理外部输入。

50. **a.22** Web 项目必须通过安全静态分析（Bandit 等）0 高危漏洞。

51. **a.23** 所有外部输入必须显式校验并使用类型安全的序列化库。

b. 推荐

52. **b.21** 在敏感数据处理函数使用 typing.Final 和只读属性防止覆写。

53. **b.22** 使用 secrets 代替 random 生成凭据。

c. 允许

54. **c.9** 对内部工具可接受环境变量中的明文配置，但禁止记录到日志。

9. 工具与自动化

a. 强制

55. **a.24** 新项目必须集成 GitHub Actions / GitLab CI 基线工作流。

56. **a.25** 所有 PR 需通过自动化检查后才可合并。

b. 推荐

57. **b.23** 使用 pre-commit 钩子本地运行格式化及静态检查。

58. **b.24** 使用 taskfile 或 Makefile 统一开发脚本命令。

10. 版本控制

a. 强制

59. **a.26** 提交信息遵循 Conventional Commits，并在 CI 中验证。

60. **a.27** 禁止提交大于 10 MiB 的二进制文件至仓库。

b. 推荐

61. **b.25** 对主干采用 trunk-based 开发，feature flag 控制未完成功能。

62. **b.26** 发布版本使用 Git tag 并自动生成 changelog。

c. 允许

63. **c.10** 对实验分支可采用临时命名规则，但合并前需按规范 squash。