## Python 技术管理规范文档

**1. 代码风格**

a. 强制

1. 必须遵循 PEP 8 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. 每行代码不得超过 79 个字符（注释和文档字符串不超过 72 个字符）

4. 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，每组之间用空行分隔

5. 禁止使用 `from module import \*`（除了在 `\_\_init\_\_.py` 中）

b. 推荐

6. 推荐使用 Black 或 autopep8 自动格式化代码

7. 推荐使用 isort 对导入语句进行排序和格式化

8. 推荐在运算符前后和逗号后添加空格，但不要有多余的空格

9. 推荐使用下划线命名法（snake\_case）命名变量和函数

10. 推荐为包和模块使用简短的全小写名称

c. 允许

11. 允许在特别长的上下文管理器中使用反斜杠换行

12. 允许在特别长的条件判断中使用括号换行

13. 允许在特别情况下突破行长度限制，但需添加注释说明原因

**2. 类型注解**

a. 强制

14. 所有新代码必须使用类型注解（Python 3.6+）

15. 公共 API 必须包含完整的类型注解

16. 禁止在类型注解中使用 `Any`，除非有充分理由

b. 推荐

17. 推荐使用 `typing` 模块提供的更精确类型（如 `Optional`, `Union` 等）

18. 推荐为复杂类型定义类型别名

19. 推荐使用 `mypy` 进行静态类型检查

20. 推荐为返回 `None` 的函数显式标注返回类型

c. 允许

21. 允许在遗留代码或原型代码中暂时省略类型注解

22. 允许在动态性特别强的代码中使用 `# type: ignore`

**3. 测试**

a. 强制

23. 所有新功能必须包含单元测试

24. 测试覆盖率不得低于 80%（关键模块不低于 90%）

25. 测试代码和生产代码必须遵循相同的代码风格标准

b. 推荐

26. 推荐使用 pytest 作为测试框架

27. 推荐为每个测试用例添加清晰的描述性名称

28. 推荐使用工厂模式（如 factory\_boy）创建测试数据

29. 推荐测试遵循 Arrange-Act-Assert 模式

30. 推荐定期运行性能测试和负载测试

c. 允许

31. 允许在原型阶段暂时省略测试

32. 允许对特别简单的代码（如纯数据类）减少测试覆盖

**4. 异常处理**

a. 强制

33. 禁止捕获所有异常（`except:` 或 `except Exception:`），除非在最外层

34. 必须为自定义异常添加文档字符串

35. 必须在日志中记录异常详细信息

b. 推荐

36. 推荐使用特定的异常类型

37. 推荐在异常消息中包含足够上下文信息

38. 推荐使用 `contextlib` 管理资源

39. 推荐在适当情况下使用 `try/except/else` 结构

c. 允许

40. 允许在特定情况下抑制异常，但必须明确记录原因

41. 允许在清理操作中捕获并记录宽泛的异常

**5. 性能与安全**

a. 强制

42. 禁止在代码中硬编码敏感信息（密码、API 密钥等）

43. 必须对用户输入进行验证和清理

44. 必须使用参数化查询防止 SQL 注入

b. 推荐

45. 推荐使用生成器处理大数据集

46. 推荐使用 `functools.lru\_cache` 缓存昂贵函数调用

47. 推荐使用连接池管理数据库连接

48. 推荐定期进行安全审计和代码审查

c. 允许

49. 允许在性能关键路径使用 C 扩展或 Cython

50. 允许在特定情况下使用全局变量，但必须明确记录原因

**6. 文档**

a. 强制

51. 所有公共 API 必须包含文档字符串（Google 或 NumPy 格式）

52. 必须为每个模块添加模块级文档字符串

53. 必须维护最新的 `CHANGELOG.md` 文件

b. 推荐

54. 推荐使用 Sphinx 生成文档

55. 推荐为复杂算法添加注释说明

56. 推荐维护架构决策记录（ADR）

57. 推荐为类型别名和常量添加文档

c. 允许

58. 允许内部工具和脚本省略详细文档

59. 允许使用 Markdown 编写补充文档

**7. 工具与流程**

a. 强制

60. 必须使用版本控制（Git）管理代码

61. 必须使用 CI/CD 流水线

62. 必须进行代码审查才能合并到主分支

b. 推荐

63. 推荐使用 pre-commit 钩子自动检查代码风格

64. 推荐使用依赖管理工具（poetry 或 pipenv）

65. 推荐使用虚拟环境隔离项目依赖

66. 推荐定期更新依赖项

c. 允许

67. 允许在特定情况下绕过 CI，但必须手动验证

68. 允许在紧急修复时简化代码审查流程

**8. 面向对象设计**

a. 强制

69. 必须遵循单一职责原则

70. 必须为抽象基类使用 `abc` 模块

b. 推荐

71. 推荐优先使用组合而非继承

72. 推荐使用数据类（`@dataclass`）替代简单类

73. 推荐使用属性（`@property`）封装内部状态

c. 允许

74. 允许在特定情况下使用多重继承

75. 允许在性能关键代码中使用 `\_\_slots\_\_`

**9. 并发与异步**

a. 强制

76. 禁止在多线程中共享可变状态而不加锁

77. 必须为异步函数使用明确的 `async/await` 语法

b. 推荐

78. 推荐使用 `concurrent.futures` 处理 CPU 密集型任务

79. 推荐使用 `asyncio` 处理 I/O 密集型任务

80. 推荐使用线程/进程池管理并发

c. 允许

81. 允许在特定情况下使用全局解释器锁（GIL）感知的优化

82. 允许在遗留代码中使用传统的多线程/多进程