Python 技术管理文档

王鸿恺 2022141461163

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **序号** | **级别** | **类别** | **规范项** | **示例 / 说明** |
| 1 | A. 强制 | 代码规范 | 遵守 PEP 8，使用自动化工具进行校验和格式化 | flake8 + black |
| 2 | A. 强制 | 命名规范 | 模块/包小写；类 CapWords；函数/变量 snake\_case | my\_module.py、MyClass、my\_function |
| 3 | A. 强制 | 代码规范 | 缩进统一 4 个空格，不允许使用 Tab | 编辑器配置 .editorconfig |
| 4 | A. 强制 | 代码规范 | 单行长度 ≤ 88 字符 | 长表达式使用反斜杠或括号换行 |
| 5 | A. 强制 | 编码声明 | 文件头部声明 UTF‑8（仅 Python 2）或使用 Unicode 字符串 | # -\*- coding: utf-8 -\*- |
| 6 | A. 强制 | 版本管理 | 必须使用 Git，提交关联 JIRA／Issue | 提交信息格式：[JIRA-123] 修复登录 BUG |
| 7 | A. 强制 | 文档 | 模块/类/函数必须写 Google Docstring | 使用三引号格式，包含 Args/Returns |
| 8 | A. 强制 | 类型检查 | 公开 API 添加类型注解，使用 mypy 校验 | def foo(x: int) -> str: |
| 9 | A. 强制 | 依赖管理 | 所有依赖声明在 requirements.txt 或 pyproject.toml | 无全局环境依赖 |
| 10 | A. 强制 | 安全 | 集成 Bandit 进行安全扫描，高危须修复 | CI 中集成 bandit --exit-zero |
| 11 | A. 强制 | 测试覆盖 | 覆盖率 ≥ 80%，关键模块 ≥ 90% | CI 报告覆盖率 badge |
| 12 | A. 强制 | 测试框架 | 统一使用 pytest，测试代码放在 tests/ 目录 | tests/test\_utils.py |
| 13 | A. 强制 | 持续集成 | 分支推送前通过风格、测试、安全检查 | GitHub Actions, GitLab CI |
| 14 | A. 强制 | 异常处理 | 禁止裸 except，捕获具体异常并记录 | except ValueError as e: |
| 15 | A. 强制 | 日志 | 使用 logging，统一级别/格式，不用 print | 配置 logging.config.dictConfig |
| 16 | B. 推荐 | 文件结构 | 按功能划分包/模块，每文件 ≤ 300 行 | user/、order/ 模块 |
| 17 | B. 推荐 | 环境隔离 | 使用 venv 或 conda 虚拟环境 | .venv/ 目录 |
| 18 | B. 推荐 | 架构分层 | 遵循 MVC 或 Hexagonal，将业务逻辑与框架解耦 | service 层、repository 层 |
| 19 | B. 推荐 | 配置管理 | 使用环境变量或 dotenv，敏感信息不硬编码 | .env 文件 |
| 20 | B. 推荐 | 依赖锁定 | 维护锁文件（Pipfile.lock / poetry.lock） | pipenv lock |
| 21 | B. 推荐 | 文档生成 | 使用 Sphinx 或 MkDocs 自动生成 API 文档 | make docs |
| 22 | B. 推荐 | 异常链传递 | 捕获后使用 raise from 保留原始异常 | raise CustomError from err |
| 23 | B. 推荐 | 性能分析 | 关键路径使用 cProfile、line\_profiler | python -m cProfile app.py |
| 24 | B. 推荐 | 静态类型 | 大项目使用 TypedDict、Protocol 提升可维护性 | class UserDict(TypedDict): |
| 25 | B. 推荐 | 安全加固 | 外部接口加速率限制，使用 OAuth2/JWT | FastAPI 插件 fastapi-limiter |
| 26 | B. 推荐 | 容器化 | 部署使用 Docker，镜像大小 ≤ 200MB | 多阶段构建 Dockerfile |
| 27 | B. 推荐 | 并发模型 | I/O 密集用 asyncio，CPU 密集用多进程 | async def / multiprocessing.Pool |
| 28 | B. 推荐 | 静态资源 | 静态文件走 CDN，并版本控制 | cdn.example.com/script.v1.js |
| 29 | B. 推荐 | 数据迁移 | 使用 Alembic 或 Django Migrations 管理数据库变更 | alembic revision --autogenerate |
| 30 | B. 推荐 | 国际化 | UI 使用 gettext 或第三方库支持多语言 | gettext('Welcome') |
| 31 | C. 允许 | 框架选择 | 允许 Flask、Django、FastAPI 等，根据需求选用 | app = FastAPI() |
| 32 | C. 允许 | ORM 选择 | 可用 SQLAlchemy、Django ORM、Peewee 等 | session.query(User) |
| 33 | C. 允许 | 缓存方案 | 可选 Redis、Memcached | redis = StrictRedis() |
| 34 | C. 允许 | 消息队列 | 允许 RabbitMQ、Kafka、Celery | celery -A tasks worker |
| 35 | C. 允许 | 接口风格 | 支持 RESTful、GraphQL、gRPC | @app.get('/items') |
| 36 | C. 允许 | 文档工具 | 可用 Markdown、reST、Asciidoc | README.md |
| 37 | C. 允许 | 代码生成 | 允许使用 cookiecutter、yeoman | cookiecutter gh:audreyr/cookiecutter-pypackage |
| 38 | C. 允许 | 自动化部署 | 可用 Ansible、Terraform、Kubernetes | ansible-playbook deploy.yml |
| 39 | C. 允许 | 监控告警 | 可集成 Prometheus、Grafana、ELK、Sentry | client.capture\_exception() |
| 40 | C. 允许 | 性能加速 | 可用 Numba、Cython 对计算密集模块加速 | @njit |
| 41 | C. 允许 | 调试工具 | 可以使用 pdb、ipdb、py-spy | python -m pdb app.py |
| 42 | C. 允许 | 代码格式化 | 除 black 外可用 yapf、autopep8 | yapf --in-place file.py |
| 43 | C. 允许 | 包管理器 | 可选 pip、pipenv、poetry | poetry add requests |
| 44 | C. 允许 | 虚拟化平台 | 允许 Docker、Podman、LXC | podman run python:3.11 |
| 45 | C. 允许 | IDE / 编辑器 | 可用 PyCharm、VSCode、Vim、Emacs，团队内保持一致 | 统一配置 EditorConfig |