Python 编码规范文档（技术管理专项）

> 语言：Python
> 适用人群：团队开发成员

---

a. 强制（必须遵守）【共15条】

1. **文件编码统一为 UTF-8**
   所有 `.py` 文件必须使用 UTF-8 编码。
2. **每个模块必须添加文件头注释**
   包含作者、创建时间、功能说明。
3. **类名使用驼峰式命名法（CapWords）**
   例如：`MyClass`, `UserManager`
4. **函数名、变量名使用小写加下划线命名法（snake_case）**
   例如：`get_user_info`, `max_value`
5. **常量名使用全大写加下划线命名法**
   例如：`MAX_RETRY_COUNT`
6. **模块名一律小写，使用下划线分隔**
   文件名如：`data_loader.py`
7. **每个类、方法必须添加 docstring 说明**
   使用三引号标准格式（Google 或 NumPy 风格均可）。
8. **每行代码不超过 100 个字符**
   超过请使用换行续写。
9. **缩进统一使用 4 个空格**
   不允许使用 Tab。
10. **禁止使用通配符导入（from x import *）**
    必须显式导入所需模块/方法。
11. **禁止在异常中使用空 `except:` 捕获**
    应使用 `except Exception as e:` 并打印/记录日志。
12. **文件末尾保留一个空行**
13. **必须使用类型注解（type hint）**
    对函数参数和返回值强制标注类型。
14. **禁止在生产代码中使用 `print()`，应使用 logging 模块**
15. **敏感信息（如密码、Token、密钥）禁止硬编码进源码**

---

b. 推荐（建议遵守）【共15条】

1. **使用 f-string 格式化字符串（Python 3.6+）**
2. **使用 `is None` 而非 `== None`**
3. **尽量使用 list/dict/set comprehensions 提高代码可读性**
4. **函数长度不超过 40 行，超过请拆分子函数**
5. **类中方法数量不超过 15 个，过多应拆分子类**

6. **逻辑复杂部分建议添加 inline comment 说明**
7. **使用 `with open()` 管理文件操作，避免忘记关闭文件**
8. **数据库连接等资源使用 `with` 或 try-finally 管理关闭**
9. **对外提供的接口函数应捕获异常并返回统一错误格式**
10. **log 记录应包括必要上下文，如 user_id, request_id**
11. **项目应配置 `pyproject.toml` 或 `setup.cfg` 管理格式工具**
12. **使用 `black`、`ruff`、`isort` 等工具自动格式化代码**
13. **避免函数有多个返回点，除非逻辑简单清晰**
14. **尽量使用 `Enum` 替代硬编码常量列表**
15. **变量、函数命名应清晰、语义明确，避免 a、b、x 之类模糊命名**

---

c. 允许（可根据情况使用）【共10条】

1. **单行 lambda 表达式可以适度使用**
   如：`sorted(data, key=lambda x: x.age)`
2. **使用 assert 进行开发期调试检查，但必须在生产环境禁用**
3. **可以使用 `typing.Any`，`Union`，`Optional` 等复杂类型注解**
4. **可以使用三元表达式（condition_if_true if condition else condition_if_false）**
5. **可以使用装饰器，但需保证其可读性与文档说明**
6. **可以使用 `@dataclass` 简化数据结构定义**
7. **对于较小的脚本项目，允许将主逻辑写在一个 `.py` 文件中**
8. **允许使用 `__slots__` 限定类属性提升内存效率（进阶用法）**
9. **允许使用 `Pathlib` 替代 `os.path`，提高跨平台兼容性**
10. **允许项目根据业务场景拓展规范，但需文档化并评审通过**