**Python 技术管理文档**

**一、命名规范**

**a. 强制**

1. 模块名必须使用小写字母，可使用下划线（如：my\_module.py）。
2. 类名必须使用驼峰式命名法（如：MyClass）。
3. 函数名必须使用小写字母+下划线（如：compute\_sum）。
4. 常量名必须使用全大写字母+下划线（如：MAX\_RETRY）。
5. 私有变量必须以下划线开头（如：\_internal\_var）。

**b. 推荐**

1. 尽量避免使用单字符变量名，除非在短小的上下文中（如：循环索引 i, j）。
2. 使用具意义的变量名，如 data\_list 优于 dl。
3. 异常类应以 Error 结尾（如：InputFormatError）。

**c. 允许**

1. 可以在函数中临时使用单字符变量（如 x, y）用于数学操作。
2. 模块名可与主类名一致（如：logger.py 中包含 Logger 类）。

**二、代码格式**

**a. 强制**

1. 每行代码不得超过 100 个字符。
2. 须使用 4 个空格缩进（禁止使用制表符 Tab）。
3. 所有文件必须以空行结尾。
4. 运算符两边必须有空格（如：a + b）。
5. 函数之间必须用一个空行隔开，类之间用两个空行。

**b. 推荐**

1. 使用黑色或 Pylint 自动检查工具进行格式化。
2. 函数参数较多时，建议一行一个参数对齐。
3. 用括号自动折行，而不是使用反斜杠。
4. 逻辑块之间使用空行增强可读性。
5. 单行注释前后需保留一个空格（如：# 注释内容）。

**c. 允许**

1. 可以临时违反缩进规则用于演示性代码，但不得提交。
2. 可使用多行字符串作为段落注释（不推荐用于单行注释）。

**三、文档与注释**

**a. 强制**

1. 所有公共函数/类必须添加 docstring，遵循 Google 风格：

def foo(bar):

"""简要说明.

Args:

bar (int): 参数说明.

Returns:

bool: 返回说明.

"""

1. 注释必须与代码保持同步，过时注释需及时删除。
2. 所有 TODO 必须包含责任人标识（如：# TODO(junjie): 添加测试逻辑）。

**b. 推荐**

1. 使用英文注释为主，适当添加中文解释用于复杂逻辑。
2. 注释应描述 “为什么” 而不是 “做什么”。
3. 对于复杂函数，应描述输入/输出的边界条件。

**c. 允许**

1. 小型脚本可以使用中文 docstring，但不得混用中英文标点。

**四、异常与日志处理**

**a. 强制**

1. 不允许使用裸 except: 捕获所有异常，必须指定异常类型。
2. 不允许在捕获异常后不作处理（如 pass）。
3. 日志输出必须使用 logging 模块，不得使用 print。

**b. 推荐**

1. 日志应区分级别（DEBUG/INFO/WARNING/ERROR）。
2. 异常信息可通过 logger.exception() 自动附带 traceback。
3. 自定义异常类应继承自 Exception。

**c. 允许**

1. 在调试阶段允许使用 print()，提交代码前必须清除。

**五、函数与结构设计**

**a. 强制**

1. 每个函数长度不得超过 50 行，复杂函数需拆分。
2. 单个文件中不应包含多个无关类。
3. 模块必须可被导入执行（避免使用过多全局变量）。

**b. 推荐**

1. 函数应具备单一职责，避免多个逻辑混杂。
2. 使用类型注解（Type Hints）提升代码可读性：

def add(x: int, y: int) -> int:

return x + y

1. 尽量避免使用全局变量。

**c. 允许**

1. 可使用 Lambda 表达式用于短小回调或排序逻辑。
2. 在性能要求极高的场景中，可使用 Cython 或 Numba 优化性能。

**六、版本控制与提交**

**a. 强制**

1. 所有代码需提交至 Git，禁止保存于本地不做版本管理。
2. 提交信息需使用英文并简洁说明改动内容：

feat: add batch processing support

fix: correct typo in error message

1. 不得将敏感信息（如密码、密钥）写入代码。

**b. 推荐**

1. 提交前运行所有测试用例通过。
2. 使用 .gitignore 排除 .pyc、临时数据等无关文件。

**c. 允许**

1. 开发阶段可使用 draft 分支自由尝试，合并需代码审查。