



Python 规范

整理自 Google Python Style Guide (中文版)

参考: <https://zh-google-styleguide.readthedocs.io/en/latest/google-python-styleguide/>

目录

- 1. Python 语言规则
 - 2. 命名规范
 - 3. 格式规范
 - 4. 注释规范
 - 5. Python 特性使用
 - 6. 编程建议
-

1. Python 语言规则

1.1 Lint

代码应通过 `pylint` 或类似工具检查，保持高可读性。

允许在必要时用 `# pylint: disable=xxx` 局部禁用。

示例：

代码块

```
1  pylint my_module.py
```

1.2 导入规则

导入顺序：

1. 标准库
2. 第三方库
3. 本地库

每组之间空一行。不要在一行中写多个导入。

示例：

代码块

```
1  import os
2  import sys
3
4  import requests
5
6  from myproject.submodule import myclass
```

1.3 包与模块

使用**绝对导入**，避免相对导入。

示例：

代码块

```
1  # 推荐
2  from myproject.subpackage import mymodule
3
4  # 不推荐
5  from . import mymodule
```

1.4 异常

捕获具体异常，避免裸 `except:`；链式异常时用 `from` 保持上下文。

示例：

代码块

```
1  # 推荐
2  try:
3      x = int(value)
4  except ValueError as e:
5      raise MyError("Invalid input") from e
6
7  # 不推荐
8  try:
9      x = int(value)
10 except:
11     pass
```

1.5 默认参数

不要使用可变对象作为默认参数。

示例：

代码块

```
1  # 推荐
2  def foo(a, b=None):
3      if b is None:
4          b = []
5
6  # 不推荐
7  def foo(a, b=[]):
8      ...
```

2. 命名规范

2.1 模块与包

- 模块：小写，用下划线（如 `my_module`）。
- 包：小写，不用下划线（如 `mypackage`）。

2.2 类

类名用大驼峰式 `CapWords`。

示例：

代码块

```
1 class MyClass:
2     pass
```

2.3 函数

- 函数与变量名小写，用下划线分隔。

示例：

代码块

```
1 def fetch_data():
2     result_total = 0
```

2.4. 变量

- 小写，可用下划线连接（如 `my_module`）。
- 命名推荐 `动宾` 格式为先保持整洁

2.5. 常量

常量用全大写，下划线分隔。

示例：

代码块

```
1 MAX_OVERFLOW = 10
```

3. 格式规范

3.1 行长度

单行最长 80 字符（注释/字符串例外可到 100）。

示例：

代码块

```
1  # 推荐
2  def long_function_name(
3      var_one, var_two, var_three,
4      var_four, var_five, var_six):
5      print(var_one)
```

3.2 缩进

使用 4 个空格，不要用 Tab。

示例：

代码块

```
1  def foo():
2      print("Hello")
```

3.3 空格

运算符两边加空格，函数调用括号内不加。

示例：

代码块

```
1  a = b + c
2  result = foo(x, y)
```

3.4 空行

- 顶层定义之间两个空行。
- 类中方法之间一个空行。

4. 注释规范

4.1 Docstring

所有模块、类、函数必须有 docstring，采用三重双引号，描述作用、参数、返回值、警告、异常。

示例：

代码块

```
1  def fetch_data(path: str) -> str:
2      """从指定路径读取数据.
3
4      # Args:
5          - path (str): 文件路径
6
7      # Returns:
8          - str: 文件内容
9
10     # Warning:
11         - 要求输入的数据必须符合某个规则
12
13     # Raises:
14         - FileNotFoundError: 文件不存在时抛出
15     """
```

说明：上面 Docstring 的内容是支持 Markdown 语法渲染的，所以才会写成这个样子。

4.2 行内注释

仅在必要时使用，保持简洁。

4.3 TODO 注释

格式：# TODO(username)：描述

示例：

代码块

```
1  # TODO(jian): 优化算法复杂度
```

5. Python 特性使用

5.1 字符串

推荐 f-string, 避免 `+` 拼接。

示例:

代码块

```
1 name = "Tom"
2 print(f"Hello, {name}")
```

5.2 类型注解

推荐使用 `typing`, 避免过度复杂。

示例:

代码块

```
1 from typing import List, Dict
2
3 def add_items(items: List[str]) -> Dict[str, int]:
4     ...
```

5.3 True/False 判断

布尔值直接使用, 不与 True/False 比较。

示例:

代码块

```
1 if items:
2     print("not empty")
```

5.4 None 判断

用 `is None` / `is not None`。

示例：

代码块

```
1  if foo is None:
2      return
```

6. 编程建议

- 函数保持简短（<40 行）。
- 避免重复代码。
- 可读性优先于简洁性。

示例：

代码块

```
1  # 推荐
2  def calculate_area(width: int, height: int) -> int:
3      return width * height
4
5  # 不推荐
6  def cal(w, h): return w*h
```

总结

- 命名：类用驼峰，函数/变量用下划线小写，常量全大写。
- 格式：80 字符、4 空格缩进、注意空格。
- 注释：docstring 必须有，TODO 要署名。
- 异常：捕获具体异常，不要裸 except。
- 字符串：推荐 f-string。

代码块

```
1
2  ---
3
```


- 4 这样调整后：**先讲规则 → 再给示例**，整个笔记会更顺畅。
- 5
- 6 要不要我在每一章最后再加一个 **“常见错误总结”小列表**（比如：默认参数陷阱、裸`except:`、拼接字符串等），方便速查？