Python 语言规范准则文档

1. 引言

本文档旨在为作者本人提供一套 Python 语言规范准则，保证代码的可读性、可维护性和一致性。文档参考了 Google 的 Python 规范(Google Python Style Guide)，作者本人在未来参与项目开发时应严格遵循本文档中的规范准则。

2. 级别划分

本文档的规范准则分为三个级别：强制、推荐和允许。强制级别的准则是必须遵循的，违反这些规则可能导致一些重大问题；推荐级别的准则虽然不要求强制遵循，但仍然建议采纳这些规则，这有助于提高代码的可读性和可维护性；允许级别的准则可以根据具体情况选择遵循与否，在特定情境下有效提升代码质量。

3. 规范准则

3.1 强制级别

（1）不要在行尾加分号， 也不要用分号把两条语句合并到一行。

（2）除非是用于隐式续行或表示元组，否则不要在返回语句或条件语句中使用括号

（3）每行代码不得超过 80 个字符，除非是较长的导入语句、URL、路径等

（4）使用 4 个空格作为缩进而不是制表符。使用隐式续行时，应该把括起来的元素垂直对齐，或者添加 4 个空格的悬挂缩进。右括号可以置于表达式结尾或者另起一行，且另起一行时应和左括号所在行的缩进相同。

（5）一级函数和类定义之间需要空两行；函数定义之间、类的文档字符串和第一个方法之间，都需要空一行；在函数或方法内部，根据需要使用单个空行。

（6）在括号、方括号或大括号内不要有空格；在逗号、分号或冒号之后要有空格，但在它们之前不要有空格；行尾不能有空格

（7）在表达式的赋值符号、操作符两边各放一个空格。在传递关键字参数或定义默认参数值时，不要在 = 两边使用空格。但如果存在类型注释，对于默认参数值，要在=两边要使用空格。

（8）不要在连续的代码行中使用空格来垂直对齐

（9）不要在循环中用 + 和 += 操作符来累加字符串

（10）文档字符串必须使用三重双引号 """

（11）错误信息需要精确地匹配真正的错误条件，插入的片段一定要能清晰地分辨出来，且要便于简单的自动化处理

（12）使用完文件和套接字以后，显式地关闭它们。

（13）在复杂的代码段开始前写上若干行注释(即段注释)，对于不明显易懂的代码，应该在行尾添加行注释。注释的井号和代码之间应有至少 2 个空格，井号和注释之间应该至少有一个空

格

（14）待办注释以大写 TODO 开头，后面跟着一个冒号以及一个包含上下文的资源链接，理想情况下是一个错误参考。

（15）每个导入语句应该各自独占一行

（16）导入语句必须在文件顶部，位于模块的注释和文档字符串之后、全局变量和全局常量之前

（17）导入语句应该按照从最通用到最不通用的顺序(标准库、第三方库、本地...)分组，每个分组内部按模块的完整包路径的字典序排序

（18）通常每个语句独占一行

（19）访问器和设置器应该遵守命名规范，比如 get_foo() 和 set_foo()

（20）如果之前的代码是通过属性获取数据，就不能把新的访问器/设置器与这一属性绑定，否则任何试图通过老方法访问变量的代码就没法运行。

（21）命名需要避免单字符名称，除非是计数器、迭代器、try/except 语句中的 e、with 语句中的 f 例外；命名需要避免包名/模块名中的包含连字符(-)；命名需要避免首尾都是双下划线的名称

（22）单元测试文件命名用小写加下划线格式的方法名，比如 test_<method_under_test>_<state>

（23）所有文件名都应以 .py 为文件后缀且不能包含连字符 (-)

（24） 所有的文件都应该可以被导入。对不需要作为程序入口地方添加 if __name__ == '__main__' 。

（25）不要用 == 与 True、 False 进行布尔运算。用 if foo is None: (或者 is not None) 来检测 None 值。

（26）避免使用 staticmethod，减少使用 classmethod

3.2 推荐级别

（1）当],), } 和最后一个元素不在同一行时，在序列尾部添加逗号。

（2）最好可以使用括号把元组给括起来， 虽然这不是必要的

（3）对外发布的 public 模块、函数、类、方法等需要包含文档字符串；而内部使用的方法、函数等，使用简单的注释描述功能。

（4）类应该在其定义下有一个用于描述该类的文档字符串。

（5）连接字符串可将每个子串添加到列表中，并在循环结束后用 ".join()，而不是在循环中用 + 和 += 累加。

（6）同一文件中保持字符串引号使用的一致性，统一使用单引号或者双引号。 为了避免在字符串内部需要对引号进行反斜杠转义，可以使用另一种引号。

（7）多行字符串推荐使用 """ 而非 '''

（8）可使用 .startswith() /endswith() 检查字符串的前缀和后缀，而不选择用字符串切片的方法。
（9）推荐使用 with 语句管理文件和类似的资源，对于不支持 with 语句且类似文件的对象，使用 contextlib.closing():
（10）导入语句按照从最通用到最不通用的顺序分组后，可以选择在分组之间插入空行。
（11）如果访问属性后需要复杂的逻辑处理，或者访问开销高昂，那么建议使用访问器和设置器函数。
（12）模块名: module_name; 包名: package_name; 类名: ClassName; 方法名: method_name; 异常名: ExceptionName; 函数名: function_name, query_proper_noun_for_thing, send_acronym_via_https; 全局常量名: GLOBAL_CONSTANT_NAME ; 全局变量名: global_var_name; 实例名: instance_var_name; 函数参数名: function_parameter_name; 局部变量名: local_var_name
（13）在模块变量和函数前加上单个下划线（_）可以在一定程度上起到保护作用
（14）将相关的类和顶级函数放在同一个模块里，没有必要像 Java 那样限制一个类一个模块
（15）类名使用大写字母开头的单词，模块名用小写加下划线的方式
（16）不硬性限制函数的长度，但如果一个函数超过 40 行，应考虑在不破坏程序结构的前提下拆分函数。
（17）if not seq: 比 if len(seq): 更好，if not seq: 比 if not len(seq): 更好，因为多利用空序列是假值的特点。
（18）用.size 属性检查 np.array 是否为空 (例如 if not users.size)，因为把 Numpy 数组转换为布尔值时可能抛出异常。

3.3 允许级别

（1）字符串可以用 + 实现单次拼接，但不要用 + 实现格式化
（2）当且仅当项目中 给常规字符串使用 '单引号时，才能在非文档字符串的多行字符串上使用 '''
（3）多行字符串不会跟进代码其他部分的缩进。若需避免字符串中的额外空格，可用多个单行字符串拼接
（4）避免在代码中使用三重引号 """，因为当使用三重引号时，缩进方式与其他部分不一致，容易引起误导。
（5）较短的注释 (比如行尾注释) 可以不用那么正式，但还是要保持风格一致
（6）导入了模块却不使用它是语法上允许的，但这通常会降低代码的可读性
（7）如果判断语句的主体与判断条件可以挤进一行，可以将它们放在同一行，但这不适用于 try / except
（8）在实例的变量或方法名称前加双下划线 (__) 可以把变量或方法变成类的私有成员 ，但

这会影响可读性和可测试性，而且没有真正实现私有

（9）嵌套函数和嵌套类可以谨慎使用，但还是尽量避免，除非需要捕获 self 和 cls 以外的局部变量

（10）全局变量尽量避免使用，但特殊情况下还是可以使用，应将其声明为模块级变量或者类属性，并在名称前加_以示为内部状态

（11）可以按需使用生成器。生成器的文档字符串中应使用”Yields:”而不是”Returns:”

（12）可使用 Lambda 函数，适用单行函数。若函数体超过 60-80 个字符，最好还是定义为常规的嵌套函数

（13）条件表达式仅可以在简单的情况下使用

（14）if 与 else 尽量一起出现，而不是全为 if 子句。

（15）默认参数值可以使用，但不要在函数或方法中定义可变对象作为默认值。

（16）允许使用特性，可以用来读取或设置涉及简单计算、逻辑的属性。但和运算符重载一样，只能在必要时使用。

（17）静态作用域可以使用，但在产生更清晰优雅的代码的同时，也可能引发让人困惑的 bug

（18）在简单的情况下允许使用推导式。

（19）只在有显著优势时才谨慎地使用装饰器。