Стандарт на кодирование ПО

«Модуль OCR (оптического распознавания текста)»

Версия 1 от 2019-10-20

Москва 2019 г.

СОДЕРЖАНИЕ

1 ВВЕДЕНИЕ

1.1 Назначение и область применения

Настоящий стандарт устанавливает соглашение о том, как писать код на языке Python.

Стандарт создан на основе PEP8 (python enhanced proposal — заявки на улучшение языка Python, pep-8 – руководство по стилю написания кода на Python), устранена избыточность путем исключения ненужных рекомендаций, из альтернативных вариантов, предлагаемых рекомендациями по оформлению, отобраны единственные наиболее подходящие. Данные модификации производились, руководствуясь собственными соображениями о наглядности и удобстве.

1.2 Ссылки

[1] СТАНДАРТ НА ТРЕБОВАНИЯ

[2] САМИ ТРЕБОВАНИЯ (ВЫСОКОГО УРОВНЯ)

[3] СЕРТИФИКАЦИОННЫЕ ТРЕБОВАНИЯ

[4] UML

1.3 Термины и определения

(Исходный) код — текст компьютерной программы на каком-либо языке программирования (в данном случае – Python) или языке разметки, который может быть прочтён человеком. Исходный код транслируется в исполняемый код целиком до запуска программы при помощи компилятора или (в данном случае) может исполняться сразу при помощи интерпретатора.

Модуль — это файл, содержащий определения и операторы Python.

Пакет — способ структурирования пространств имён модулей Python за счёт использования имён модулей, разделённых точками.

(Соглашение) CamelCase — стиль написания составных слов, при котором несколько слов пишутся слитно без пробелов, при этом каждое слово внутри фразы пишется с прописной буквы.

2 ВНЕШНИЙ ВИД КОДА

2.1 Отступы

Следует использовать 4 пробела на один уровень отступа. В старом коде, который не рекомендуется менять, можно продолжить использовать 8 пробелами для отступа.

2.2 Смешивание символов отступов

Во избежание ошибок в работе интерпретатора не следует смешивать символы табуляции и пробелов. Для отступов рекомендуется использовать только пробелы.

2.3 Максимальная длина строки

Следует ограничить максимальную длину строки 79 символами.

Предпочтительный способ переноса длинных строк — использование подразумевающегося продолжения строки между обычными, квадратными и фигурными скобками. В случае необходимости можно добавить еще одну пару скобок вокруг выражения, но рекомендуется использовать для этого обратный слэш. Рекомендуется также сохранять правильные отступы для перенесённой строки. Предпочтительнее вставить перенос строки после бинарного оператора, но не перед ним.

2.4 Пустые строки

Следует отделять функции (верхнего уровня, не функции внутри функций) и определения классов двумя пустыми строчками.

Определения методов внутри класса следует отделять одной пустой строкой.

Дополнительные отступы строками могут быть изредка использованы для выделения группы логически связанных функций. Пустые строки могут быть пропущены, между несколькими выражениями, записанными в одну строку, например, «заглушки» функций.

Допускаются пустые строки в коде функций, чтобы отделить друг от друга логические части, однако, этим не следует злоупотреблять.

2.5 Import-секции

Импортирование разных модулей должно быть на разных строчках.

Импортирование всегда следует делать сразу после комментариев к модулю и строк документации, перед объявлением глобальных переменных и констант.

Группировать импорты следует следующем порядке:

- импорты стандартной библиотеки
- импорты сторонних библиотек
- импорты модулей текущего проекта

Следует вставлять пустую строку между каждой группой импортов.

Относительные импорты крайне не рекомендуются — по возможности рекомендуется указывать абсолютный путь к модулю для всех импортирований.

2.6 Пробелы в выражениях и инструкциях

Следует избегать использования пробелов в следующих ситуациях:

- Сразу после или перед скобками (обычными, фигурными и квадратными)
- Сразу перед запятой, точкой с запятой, двоеточием
- Сразу перед открывающей скобкой, после которой начинается список аргументов при вызове функции
- Сразу перед открывающей скобкой, после которой следует индекс или срез

- Использование более одного пробела вокруг оператора присваивания (или любого другого) для того, чтобы выровнять его с другим таким же оператором на соседней строке

Следует всегда окружать эти бинарные операторы одним пробелом с каждой стороны: присваивание (=, +=, -= и прочие), сравнения (==, <, >, !=, <>, <=, >=, in, not in, is, is not), логические операторы (and, or, not).

Следует ставить пробелы вокруг арифметических операций.

Не следует использовать пробелы для отделения знака =, когда он употребляется для обозначения аргумента-ключа (keyword argument) или значения параметра по умолчанию.

Не следует использовать составные инструкции (несколько команд в одной строке).

Допустимо писать тело циклов while, for или ветку if в той же строке, если команда короткая, но если команд несколько, так делать не следует.

3 КОММЕНТАРИИ

Всегда следует исправлять комментарии, если производится изменение сопровождаемого ими кода.

Комментарии должны являться законченными предложениями. Если комментарий — фраза или предложение, первое слово должно быть написано с большой буквы, если только это не имя переменной, которая начинается с маленькой буквы.

Если комментарий короткий, можно опустить точку в конце предложения. Блок комментариев обычно состоит из одного или более абзацев, составленных из полноценных предложений, поэтому каждое предложение должно оканчиваться точкой.

Следует два пробела после точки в конце предложения.

3.1 Блок комментариев

Блок комментариев как правило должен объясняет код (весь, или только некоторую часть), идущий после блока, и должен иметь тот же отступ, что и сам код. Каждая строчка такого блока должна начинаться с символа # и одного пробела после него (если только сам текст комментария не имеет отступа).

Абзацы внутри блока комментариев лучше отделять строкой, состоящей из одного символа #.

3.2 Комментарии в строке с кодом

Рекомендуется как можно реже использовать подобные комментарии.

Такой комментарий должен находиться в той же строке, что и инструкция, и отделяться хотя бы двумя пробелами от инструкции. Они должны начинаться с символа # и одного пробела.

Комментарии в строке с кодом не рекомендуется использовать, если они объясняют очевидное.

4 ИМЕНА

4.1 Имена, которых следует избегать

Не рекомендуется использовать символы l (маленькая латинская буква «эль»), O (заглавная латинская буква «о») или I (заглавная латинская буква «ай») как однобуквенные идентификаторы.

В некоторых шрифтах эти символы неотличимы от цифры один и нуля; если очень нужно использовать l имена, следует писать вместо неё заглавную L.

4.2 Имена модулей

Модулям следует давать короткие имена, состоящие из маленьких букв. Можно использовать и символы подчеркивания, если это улучшает читабельность. То же, за исключением символов подчеркивания, относится и к именам пакетов.

4.3 Имена классов

Все имена классов должны следовать соглашению CamelCase. Классы внутреннего использования могут начинаться с символа подчеркивания.

4.4 Имена исключений

Так как исключения являются классами, к исключениями применяется стиль именования классов. Однако допустимо добавить «Error» в конце имени (если конечно исключение действительно является ошибкой).

4.5 Имена функций

Имена функций должны состоять из маленьких букв, а слова разделяться символами подчеркивания — это необходимо, чтобы увеличить читабельность.

Стиль CamelCase допускается только в тех местах, где уже преобладает такой стиль.

4.6 Имена глобальных переменных

Следует руководствоваться теми же соглашениями, что и для имен функций.

Настоятельно рекомендуется добавлять в модули, которые написаны так, чтобы их использовали с помощью from M import *, механизм __all__ чтобы предотвратить экспортирование глобальных переменных. Или же, используя старое соглашение, добавлять перед именами таких глобальных переменных один символ подчеркивания, которым можно обозначить те глобальные переменные, которые используются только внутри модуля.

4.6.1 Аргументы функций и методов

Всегда следует использовать self в качестве первого аргумента метода экземпляра объекта (instance method).

Всегда следует использовать cls в качестве первого аргумента метода класса (class method).

Если имя аргумента конфликтует с зарезервированным ключевым словом python, рекомендуется в конец имени символ подчеркивания, нежели чем исказить написание слова или использовать аббревиатуру. будет подобрать синоним).

4.6.2 Имена методов и переменные экземпляров классов

Следует тот же стиль, что и для имен функций: имена должны состоять из маленьких букв, а слова разделяться символами подчеркивания.

Чтобы избежать конфликта имен с подклассами, рекомендуется добавлять два символа подчеркивания, чтобы включить механизм изменения имен. Вообще, двойное подчеркивание в именах следует использовать, чтобы избежать конфликта имен с атрибутами классов, спроектированных так, чтобы от них наследовали подклассы.

4.6.3 Константы

Константы следует объявлять на уровне модуля и записывать только заглавными буквами, а слова разделять символами подчеркивания.

5 ОБРАБОТКА ИСКЛЮЧЕНИЙ

Когда код перехватывает исключения, следует отлавливать конкретные ошибки вместо простого выражения except.

Однако, допустимо использование чистого 'except:' в двух случаях:

- Если обработчик исключения выводит пользователю всё о случившейся ошибке (например, traceback)
- Если нужно выполнить некоторый код после перехвата исключения, а потом вновь «бросить» его для обработки где-то в другом месте. Обычно же лучше пользоваться конструкцией 'try...finally'.

Рекомендуется заключать в каждую конструкцию try...except минимум кода, для облегчения отлавливания ошибки.