PEP – это аббревиатура от Python Enterprise Proposal. Написание кода с правильной логикой является ключевым фактором программирования, но многие другие важные факторы могут повлиять на качество кода. Стиль кодирования разработчика делает код очень надежным, и каждый разработчик должен помнить, что Python строго следует порядку и формату строки.

Адаптивный красивый стиль кодирования делает код более читабельным. Код становится простым для конечного пользователя.

PEP 8 – это документ, который предоставляет различные рекомендации по написанию читаемого на Python. PEP 8 описывает, как разработчик может писать красивый код. Он был официально написан в 2001 году Гвидо ван Россумом, Барри Варшавой и Ником Когланом. Основная цель PEP – улучшить читаемость и согласованность кода.

Почему важен PEP 8?

PEP 8 улучшает читаемость кода Python.

Создатель Python Гвидо ван Россум сказал, что код может быть написан за несколько минут, несколько часов или целый день, но как только мы напишем код, мы больше никогда его не переписываем. Но иногда нам нужно читать код снова и снова.

На этом этапе мы должны иметь представление о том, почему мы написали конкретную строку в коде. Код должен отражать значение каждой строки. Вот почему так важна удобочитаемость.

Условные обозначения

Условные обозначения в Python немного запутаны, но есть определенные соглашения, которым мы можем легко следовать. Давайте посмотрим на следующие обозначения.

1. Одна строчная буква
2. Одна заглавная буква
3. Строчные буквы
4. Строчные буквы с подчеркиванием
5. Верхний регистр
6. Верхний регистр с подчеркиванием
7. Заглавные слова(или CamelCase)

Стили имен

Функция. Мы должны использовать слова в нижнем регистре или разделять слова подчеркиванием(myfunction, my_function)/

Переменная. Мы должны использовать строчные буквы, слова или отдельные слова, чтобы улучшить читаемость(a, var, variable_name)

Класс. Первая буква названия класса должна быть заглавной; используйте camel case. Не разделяйте слова подчеркиванием.(MyClass, Form, Model)

Метод. Мы должны использовать строчные буквы, слова или отдельные слова, чтобы улучшить читаемость. class_method, method

Константа Использование заглавных букв, слов или отдельных слов для повышения удобочитаемости.(MYCONSTANT, CONSTANT, MY_CONSTANT)

Модуль. Мы должны использовать строчные буквы, слова или отдельные слова, чтобы улучшить читаемость.(Module_name.py, module.py)

Пакет. Мы должны использовать строчные буквы, слова или отдельные слова, чтобы улучшить читаемость. Не разделяйте слова подчеркиванием.(package, mypackage)

Макет кода

Макет кода определяет, насколько код читается.

Отступ

Отступы – важная часть языка программирования Python, определяющая уровень строк кода. Как правило, мы используем 4 пробела для отступа. Давайте разберемся в следующем примере.
Отступ определяет блок кода и сообщает нам, какие операторы выполняются при вызове функции или триггере условия.

Вкладки против пробела

Мы также можем использовать вкладки, чтобы предоставить последовательные пробелы, чтобы указать отступ, но пробелы являются наиболее предпочтительными. Python 2 позволяет смешивать табуляции и пробелы, но мы получим ошибку в Python 3.

Использование документационной строки

Python предоставляет два типа документационных строк – однострочные и многострочные. Мы используем тройные кавычки для определения однострочного или многострочного типа. В основном они используются для описания функции или конкретной программы.

Должна ли линия разрываться до или после бинарного оператора?

Строки разрываются до или после двоичной операции – это традиционный подход. Но это сильно влияет на удобочитаемость, потому что операторы разбросаны по разным экранам, и каждый оператор находится вдали от своего операнда и на предыдущей строке. Python позволяет нам разбивать строку до или после бинарного оператора, если соглашение согласовано локально.

Модуль импорта

Мы должны импортировать модули в разделительной строке следующим образом.

Оператор импорта должен быть написан в верхней части файла или сразу после любого комментария модуля. Рекомендуется использовать абсолютный импорт, потому что он более удобочитаемый и, как правило, ведет себя лучше.
Однако мы можем использовать явный относительный импорт вместо абсолютного импорта, особенно при работе со сложными пакетами.

Пустые строки

Пустые строки могут улучшить читаемость кода Python. Если много строк кода сгруппированы вместе, код станет труднее читать.

Комментарии

Комментарии являются неотъемлемой частью любого языка программирования. Это лучший способ объяснить код. Когда мы задокументировали наш код с соответствующими комментариями, каждый сможет понять код. Но следует помнить о следующих моментах.

Начните с заглавной буквы и напишите полное предложение.
Обновите комментарий в случае изменения кода.
Ограничьте длину строки комментариев и строк документации 72 символами.

Блочный комментарий

Блочные комментарии – хороший выбор для небольшого участка кода. Такие комментарии полезны, когда мы пишем несколько кодов строк для выполнения одного действия, такого как повторение цикла. Они помогают нам понять цель кода.

PEP 8 предоставляет следующие правила для записи блока комментариев.

Комментарий блока отступа должен быть на одном уровне.
Каждую строку начинайте с символа #, за которым следует один пробел.
Выделите строку одним знаком #.

Встроенные комментарии
Встроенные комментарии используются для объяснения одного оператора в фрагменте кода. Мы можем быстро понять, почему мы написали именно эту строку кода. PEP 8 определяет следующие правила для встроенных комментариев.

Комментарии в Python начинаются с символа # и одного пробела.
Осторожно используйте встроенные комментарии.
Мы должны разделять встроенные комментарии в той же строке, что и утверждение, на которое они ссылаются.
Встроенные комментарии необходимы, но блочные комментарии делают код более читабельным.

Избегайте сравнения логических значений с помощью оператора эквивалентности
Мы не должны использовать оператор эквивалентности == для сравнения логических значений. Он может принимать только Истину или Ложь.