Требования к коду | Я.Шпора

Основные правила PEP 8

- Длина строки не превышает 79 символов.
- Отступы вложенных блоков кода — 4 пробела.
- Стиль именования переменных и функций соответствует разделу PEP 8: Naming Conventions.
- При переносе строк соблюдены правила отступов.
- Для переноса строк не используются бэкслеши `\`.
- В коде нет неиспользуемых импортов.
- Импорты в коде описаны согласно иерархии:

  1. Импорты из стандартной библиотеки.
  2. Импорты из сторонних библиотек.

  Внутри каждой группы импорты отсортированы в алфавитном порядке.
- Функции, объявленные на верхнем уровне программы (не вложенные в другие блоки кода), разделены двумя пустыми строками.
- В коде одного проекта используются кавычки одного типа.
- Комментарии начинаются со знака `#`. Символ `#` от текста комментария отделён одним пробелом. Комментарий выровнен с кодом, к которому он относится.
- Если комментарий размещён в той же строке, что и код, между кодом и комментарием стоит два пробела.
- Комментарий начинается с заглавной буквы, а после комментария стоит точка.
- Строка комментария не превышает 72 символа.

Линтеры

Программы, которые помогают проконтролировать соответствие кода установленным стандартам.

Расширения для линтеров в VS Code: Pylint, Flake8, Mypy.

Форматеры

Программы, которые автоматически структурируют код, следуя заданным стилевым правилам и конвенциям.

Форматеры для VS Code: autopep8 и Black Formatter.

Как установить форматер по умолчанию для файлов Python в VS Code

1. Откройте в VS Code любой файл с расширением *.py*.
2. Щёлкните правой кнопкой мыши на редакторе, чтобы вызвать контекстное меню.
3. Выберите пункт *Format Document With...*
4. В раскрывающемся меню выберите *Configure Default Formatter...*
5. Выберите из списка нужный форматер.

Чтобы отформатировать код, нужно щёлкнуть правой кнопкой мыши на редакторе и выбрать пункт *Format Document*.

Аннотация типов

Явное указание типа ожидаемых данных при объявлении переменных, классов и функций:

```
# Без аннотации: объявлена переменная,
# а Python сам определил, какой в ней тип данных.
birth_year = 1971

# С аннотацией: объявлена переменная и указано,
# что это переменная только для целых чисел.
birth_year: int = 1971

# Общий синтаксис аннотирования переменных.
<имя переменной>: <принимаемый тип> = <значение>
```

Инструменты для проверки аннотации типов

- **Муру** — активно развивается при участии Гвидо ван Россума, создателя

Python.

- **Pyre** — альтернатива Муру.
- **Pytype** — инструмент для анализа типов, разработанный компанией Google.
- **Mypy Type Checker** — создан корпорацией Microsoft.

Аннотация типов для функций

```
# def <имя функции>(<аргумент>: <тип>) -> <возвращаемый тип>:

def is_rhomb(a_size: float, b_size: float) -> bool:
    """Принимает на вход длины сторон параллелограмма
       и проверяет, является ли параллелограмм ромбом."""
    # Вернёт True или False в зависимости от истинности выражения.
    return a_size == b_size


# Функция print_hi() ожидает строковый аргумент name,
# значение этого аргумента по умолчанию - 'незнакомец'.
# Функция ничего не возвращает, тип возвращаемых данных - None.
def print_hi(name: str = 'незнакомец') -> None:
    print('Привет,' + name + '!')


# Можно напечатать аннотации функций.
print(f'Аннотации для is_rhomb(): {is_rhomb.__annotations__}')
# Аннотации для is_rhomb(): {
#     'a_size': <class 'float'>,
#     'b_size': <class 'float'>,
#     'return': <class 'bool'>
# }
print(f'Аннотации для print_hi(): {print_hi.__annotations__}')
# Аннотации для print_hi(): {'name': <class 'str'>, 'return': None}
```

Модуль typing

Предоставляет набор стандартных типов и инструментов для создания более

сложных аннотаций.

Тип Optional используется, когда аргумент функции может быть определённым типом или `None`:

```
from typing import Optional

# Функция greet() может вернуть строку или None,
# так что надо аннотировать переменную этими двумя типами.
result_no_name: Optional[str] = greet()
```

Тип Union используется, когда переменная может получить значение одного из нескольких типов:

```
from typing import Union

# Переменную, которая готова принять значения типа None и str,
# можно аннотировать и с помощью Union:
result_has_name: Union[str, None] = greet('Гвидо')
```

Полезные ресурсы

PEP 8 – Style Guide for Python Code

Документация модуля typing

Где кончается память — начинается
мастерство применения шпаргалок.

Брюс И, мастер боя на шпаргалках