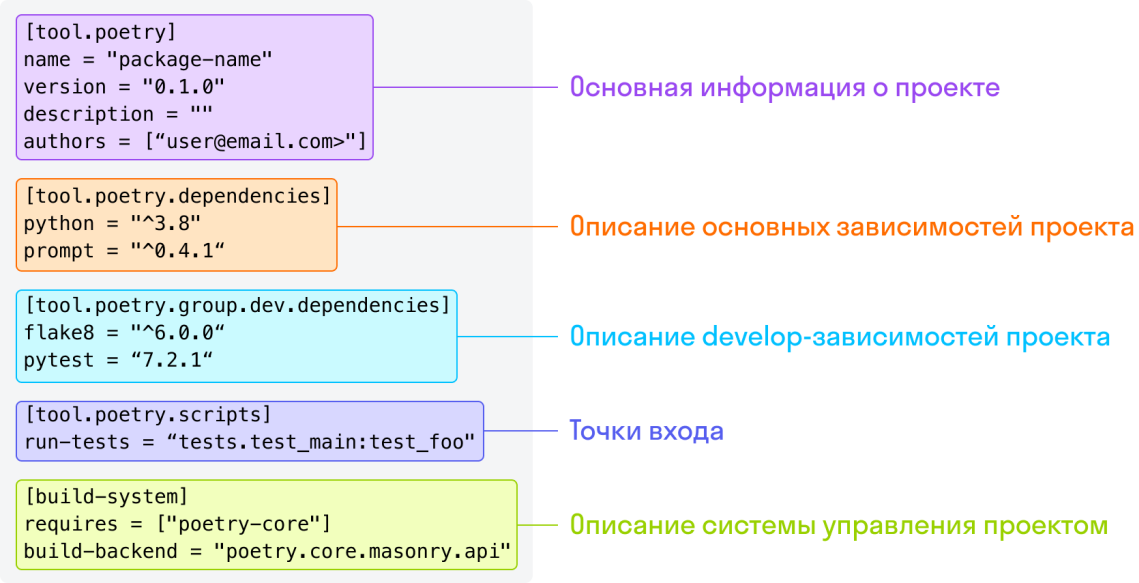
Poetry. Оформление кода. Шпаргалка

Poetry

Poetry — это менеджер пакетов (менеджер зависимостей) Python, который позволяет одновременно создавать виртуальное окружение и управлять зависимостями проекта. В Poetry есть единый файл спецификации pyproject.toml, в котором содержится основная информация по пакету (проекту).

Структура TOML-файла



Команды для работы с Poetry

- `poetry init` — инициализировать пакет в существующем проекте.
- `poetry new package-name` — создать новый проект.
- `poetry env use python3.8` — создать виртуальное окружение, указав версию интерпретатора.
- `poetry shell` — активировать виртуальное окружение.
- `exit` — выйти из виртуального окружения.
- `poetry run python main.py` — запустить файл Python без активации виртуального окружения.

Управление зависимостями

- `poetry add requests` — установить зависимость.
- `poetry install` — первичная установка зависимостей из файла.
- `poetry update` — обновить зависимости.
- `poetry remove requests` — удалить зависимость из проекта.
- `poetry show --tree` — посмотреть всё дерево зависимостей.
- `poetry show --latest` — посмотреть, последние ли версии используются в проекте.

Добавление зависимостей в отдельную группу

- `poetry add --group dev requests` — установка зависимости в группу с именем `dev`.

Основные правила PEP8

Названия переменных пишите в стиле snake_case

Пишем так:

```
user_name = "Alex"

user_salary = 100000

user_age = 33
```

Так нельзя:

```
UserName = "Alex"

user_Salary = 100000

user age = 33
```

Не используйте зарезервированные слова языка

Если вы будете использовать название оператора, то получите ошибку.

Неполный список слов, которые нельзя использовать:

- `False`
- `True`
- `None`
- `and`
- `with/as`
- `break`
- `class`
- `continue`
- `def`
- `del`
- `elif`
- `else`
- `finally`
- `for`
- `from`
- `global`
- `if`
- `import`
- `in`
- `is`
- `lambda`
- `not`
- `or`
- `pass`
- `raise`
- `return`
- `try`
- `while`

Не используйте названия встроенных функций

Несмотря на то что вы можете переписать встроенные имена, лучше этого не делать. Это приведет к ошибкам, о причинах возникновения которых бывает не так просто догадаться.

Так можно:

```
greeting_text = "Добро пожаловать"

number_of_users = 4

max_guests_num = 50
```

Так нельзя:

```
print = "Добро пожаловать"

sum = 4

max = "Happy Birthday"
```

Не используйте обманчивые символы

Не используйте следующие символы как однобуквенные идентификаторы:

- `l` (маленькая латинская буква «эль») и `I` (заглавная латинская «ай»);
- `0` (ноль) и `o` (латинская буква «о»).

Используйте пробелы вокруг операторов

```
students_count = 12
group_id = 1620
teacher_name = "Дмитриева Клементина"
```

Ставьте пробелы после запятой при вызове функций

```
print("оформление", "это", "важно")
```

В блочном коде ставьте четыре пробела

```
if pep == 8:
    print("Всё отлично!")
```

Логические блоки разделяйте пустыми строками

```
price = int(input())

payment = price / 10
payment_round = round(payment)

print(payment_round)
```

Используйте пустую строку в конце кода

```
if pep == 8:
    print("Всё отлично!")

```

Допустимая длина строки

Ограничение длины строки в Python, как рекомендует PEP 8, составляет 79 символов. Однако, согласно тому же PEP 8, максимальная допустимая длина строки увеличивается до 119 символов в некоторых случаях.

Работа с константами

```
# Стандартное количество баночек в упаковке
CANS_IN_PACKAGE = 32
```

```
# Приветствие пользователя
USER_GREETING = "Добро пожаловать в систему"
```

```
# Разрешение экрана
SCREEN_WIDTH = 640
SCREEN_HEIGHT = 480
```

```
# Путь к файлу
SOURCE_PATH = "/docs/source/data.txt"
```

```
# Путь к API
SOURCE_URL = "https://sky.pro/api/load-data/"
```

```
# Количество слов на странице
SYMBOLS_PER_PAGE = 1800
```

Линтер Flake8

Установка в `poetry`:

```
poetry add --group lint flake8
```

Пример конфигурации:

```
[flake8]
max-line-length = 119
ignore = E203, W503
exclude = .git, __pycache__, venv, .venv
```

Запуск:

```
flake8 myfile.py
```

Docstring

> **Docstring** — это строка документации, которая объясняет, что делает функция, метод или модуль.

Docstring должен быть написан внутри тела функции, метода или модуля, и он должен описывать, что делает функция, а не как она это делает.

```
def add_numbers(a, b):
    """Функция, которая складывает два числа."""
    return a + b
```

Аннотации типов

Аннотации для переменных пишут через двоеточие после идентификатора. После этого может идти инициализация значения:

```
price: int = 5
title: str
```

Параметры функции аннотируются так же, как и переменные, а возвращаемое значение указывается после стрелки → и до завершающего двоеточия:

```
def indent_right(s: str, width: int) -> str:
    return " " * (max(0, width - len(s))) + s
```

Модуль typing

> **Модуль** `typing` — это модуль стандартной библиотеки Python, который содержит типы данных и функции для аннотации типов.

Для работы с модулем сделайте импорт используемых объектов:

```
from typing import Union, Any, Optional
```

Union

```
from typing import Union

def foo(arg: Union[str, int]) -> None:
    print(arg)
```

Any

```
from typing import Any

def foo(arg: Any) -> None:
    print(arg)
```

Optional

```
from typing import Optional

def foo(arg: Optional[int] = None) -> None:
    print(arg)
```

Iterable

```
from typing import Iterable

def foo(arg: Iterable[int]) -> None:
    for num in arg:
        print(num)
```

Callable

```
from typing import Callable, Iterable

def foo(func: Callable, array: Iterable[int]) -> None:
    for item in array:
        func(item)
```

mypy

Установка в `poetry`:

```
poetry add --group lint mypy
```

Пример конфигурации:

```
[tool.mypy]
disallow_untyped_defs = true
no_implicit_optional = true
warn_return_any = true
exclude = 'venv'
```

Запуск:

```
mypy foo.py bar.py some_directory/
```

Форматер кода black

Установка в `poetry`:

```
poetry add --group lint black
```

Пример конфигурации:

```
[tool.black]
# Максимальная длина строки
line-length = 119
# Файлы, которые не нужно форматировать
exclude = '''
(
  /(
      \.eggs         # Исключить несколько общих каталогов
    | \.git          # в корне проекта
    | \.mypy_cache
    | \.venv
  )/
  | foo.py           # Также отдельно исключить файл с именем foo.py
                     # в корне проекта
)
'''
```

Запуск:

```
black main.py test.py utils
```

Форматер кода isort

Установка в `poetry`:

```
poetry add --group lint isort
```

Пример конфигурации:

```
[tool.isort]
# максимальная длина строки
line_length = 119
```

Запуск:

```
isort main.py test.py utils.py
```