Документирование кода в Python

Документирование кода — неотъемлемая часть разработки на Python. Порой документации в коде может быть больше, чем самого кода. Она помогает понять, что делает функция или класс, какие аргументы принимает и что возвращает.

Когда документация и код находятся в разных местах, сопровождать их становиться довольно тяжело. Поэтому на практике документация находится непосредственно рядом с кодом.

Docstring

Docstring — это строковый литерал, который расположен сразу за объявлением модуля, функции, класса или метода. О том, какие существуют соглашения в документировании Python кода описано в документации PEP257.

Документация для классов

Документация класса создается для самого класса, а также для его методов.

```
class Speaker:
"""Это docstring класса Speaker"""


def say_something(self):
    """Это docstring метода"""


    print("something")
```

После строки документации нужно оставлять пустую строку

Документация для класса может содержать следующую информацию:

- краткое описание класса (+ его поведение);
- описание атрибутов класса;
- описание публичных методов;
- все, что связано с интерфейсом для подклассов.

Для методов класса документация может содержать:

- краткое описание метода (+ его поведение);
- описание аргументов метода;
- побочные эффекты (если таковые возникают при выполнении метода);
- исключения.

Ниже — пример с более подробной документацией класса:

```
class TextSplitter:
"""Класс TextSplitter используется для разбивки текста на слова


Основное применение - парсинг логов на отдельные элементы

по указанному разделителю.


Note:

    Возможны проблемы с кодировкой в Windows
```

```
    Attributes
    ----------
    file_path : str
        полный путь до текстового файла
    lines : list
        список строк исходного файла

    Methods
    -------
    load()
        Читает файл и сохраняет его в виде списка строк в lines
    get_splitted(split_symbol=" ")
        Разделяет строки списка по указанному разделителю
        и возвращает результат в виде списка
    """

    def __init__(self, file_path: str):
        self.file_path = file_path.strip()
        self.lines = []

    def load(self) -> None:
        """Метод для загрузки файла в список строк lines
```

```
        Raises
        ------
        Exception
            Если файл пустой вызовется исключение
        """

        with open(self.file_path, encoding="utf-8") as f:
            for line in f:
                self.lines.append(line.rstrip('\n'))
            if len(self.lines) == 0:
                raise Exception(f"file {self.file_path} is empty")

    def get_splitted(self, split_symbol: str = " ") -> list:
        """Разбивает текстовые строки lines, преобразуя строку в
        список слов по разделителю

        Если аргумент split_symbol не задан, в качестве разделителя
        используется пробел

        Parameters
        ----------
        split_symbol : str, optional
            разделитель
```

```
        """


        split_list = []
        for str_line in self.lines:
            split_list.append(str_line.split(split_symbol))
        return split_list
```

Документация для пакетов

Документация пакета размещается в файле __init__.py в верхней части файла (начиная с 1-й строки). В ней может быть указано:

- описание пакета;
- список модулей и пакетов, экспортируемых этим модулем;
- автор;
- контактные данные;
- лицензия.

```
    """
Пакет Mos помогает создать полноэкранный текстовый интерфейс в консоли.


Alex Ivanov [https://alex.ivanov.ru/]
alex.ivanov@gmail.com
# License: BSD
"""
__author__ = 'Alex Ivanov'
```

```
try:
    from .version import version
except ImportError:
    version = "0.0.0"


__version__ = version
```

Документация для модулей

Документация модулей аналогична документации классов. Вместо класса и методов в данном случае документируется модуль со всеми его функциями. Размещается в верхней части файла (начиная с 1-й строки).

Форматы Docstring

Строки документации могут иметь различное форматирование. В примере выше мы использовали стиль NumPy. Существуют и другие форматы:

- Google styleguide -> Comments and Docstrings
- Numpydoc docstring guide
- Epydoc
- reStructuredText (reST)

Вывод документации на экран — help() и __doc__

Строки документации доступны:

- из атрибута __doc__ для любого объекта;
- с помощью встроенной функции help().

Выведем документацию с помощью функции help():

```
>>> import my_module
>>> help(my_module)

Help on module test:

NAME
    test - Это docstring модуля, он однострочный.

FILE
    /var/www/test.py

CLASSES
    MyClass

    class MyClass
     |  Это docstring класса.
     |
```

```
 |  Methods defined here:
 |
 |  my_method(self)
 |      Это docstring метода


FUNCTIONS
    my_function(a)
        Это многострочный docstring для функции my_function.


        В многострочном docstring первое предложение
        кратко описывает работу функции.
```

Также можно выводить документацию отдельного объекта:

```
>>> import my_module
>>> my_module.__doc__
>>> my_module.my_function.__doc__
>>> my_module.MyClass.__doc__
>>> my_module.MyClass.my_method.__doc__
```

Pydoc

Для более удобной работы с документацией, в Python существует встроенная библиотека pydoc.

Pydoc автоматически генерирует документацию из Python модулей. Информацию по доступным командам модуля pydoc можно получить набрав в терминале:

python -m pydoc

Разберем подробнее, что умеет pydoc.

Вывод текста документации

pydoc <name> — покажет текст документации указанного модуля, пакета, функции, класса и т.д. Если <name> содержит "\", Python будет искать документацию по указанному пути.

Для примера, посмотрим документацию встроенного модуля math:

```
   python -m pydoc math
Help on built-in module math:


NAME

   math


DESCRIPTION

   This module provides access to the mathematical functions

   defined by the C standard.


FUNCTIONS

   acos(x, /)

       Return the arc cosine (measured in radians) of x.
```

```
    acosh(x, /)

        Return the inverse hyperbolic cosine of x.



...
```

В консоль выведется название модуля, его описание и описание всех функций в модуле.

Поиск по документации

pydoc -k <keyword> — найдет ключевое слово в документации всех доступных модулей.

Допустим, нам нужно распаковать gzip файл. Поищем слово "gzip":

```
    python -m pydoc -k gzip
_compression - Internal classes used by the gzip, lzma and bz2 modules

gzip - Functions that read and write gzipped files.

test.test_gzip - Test script for the gzip module.
```

В списке мы видим модуль gzip. Теперь можно посмотреть его документацию:

```
    python -m pydoc gzip
Help on module gzip:



NAME
```

```
    gzip - Functions that read and write gzipped files.


DESCRIPTION

    The user of the file doesn't have to worry about the compression,

    but random access is not allowed.
```

По описанию, данный модуль решит нашу задачу.

HTTP сервер с документацией

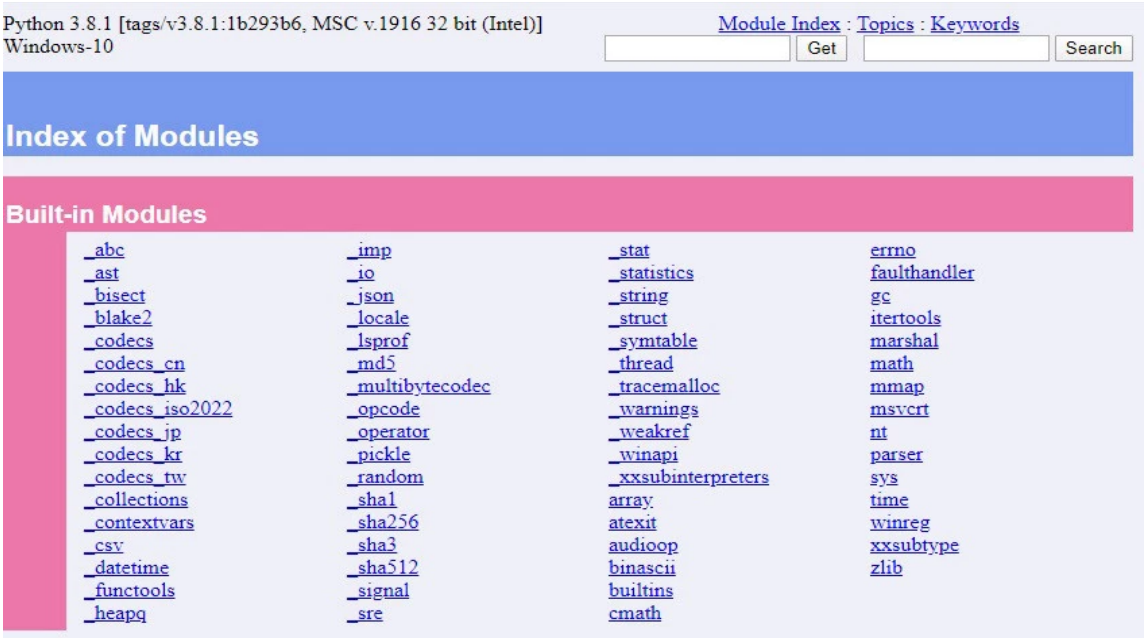
Для удобства просмотра документации, pydoc позволяет одной командой создать HTTP-сервер:

```
    sudo python -m pydoc -p 331
Server ready at http://localhost:331/

Server commands: [b]rowser, [q]uit

server>
```

Теперь можно перейти в браузер и зайти на http://localhost:331/



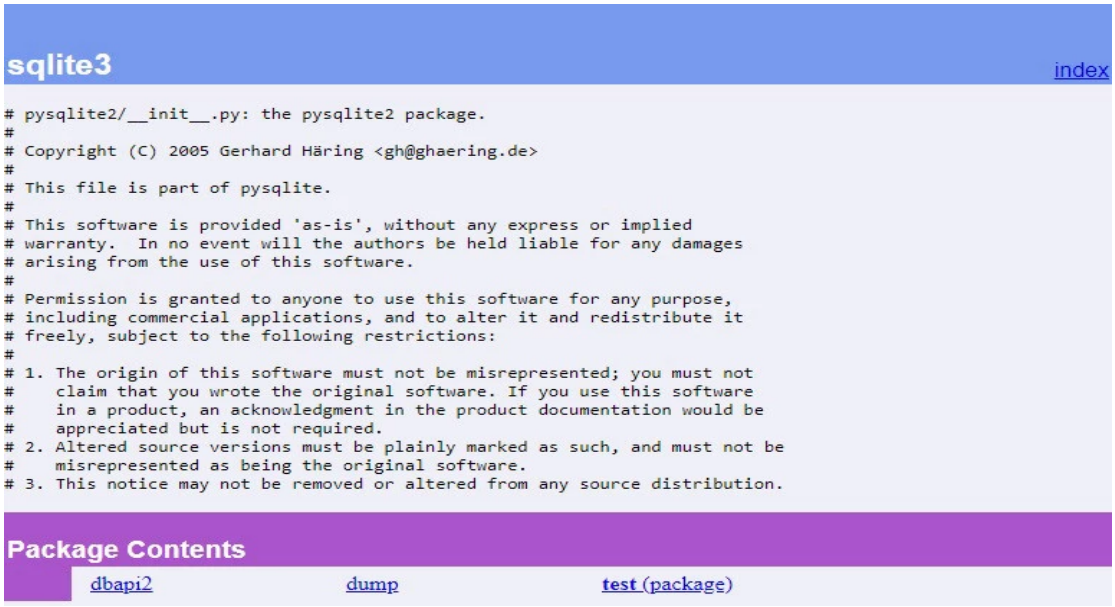
Для остановки сервера введите "q" и нажмите "Enter":

server> q
Server stopped

Также HTTP-сервер доступен через python -m pydoc -b – эта команда создаст сервер на свободном порту, откроет браузер и перейдет на нужную страницу.

Запись документации в файл

python -m pydoc -w sqlite3 — запишем файл с документацией по модулю sqlite3 в html файл.



Автодокументирование кода

Для того чтобы облегчить написание документации и улучшить ее в целом, существуют различные Python-пакеты. Один из них — pyment.

Pyment работает следующим образом:

- Анализирует один или несколько скриптов.
- Получает существующие строки документации.
- Генерирует отформатированные строки документации со всеми параметрами, значениями по умолчанию и т.д.
- Далее вы можете применить сгенерированные строки к своим файлам.

Этот инструмент особенно полезен когда код плохо задокументирован, или когда документация вовсе отсутствует. Также pyment будет полезен в команде разработчиков для форматирования документации в едином стиле.

Установка:

```
pip install pyment
```

Использование:

```
pyment myfile.py    # для файла
pyment -w myfile.py  # для файла + запись в файл
pyment my/folder/  # для всех файлов в папке
```

Для большинства IDE также существуют плагины, помогающие документировать код:

- AutoDocstring – для VS Code.
- AutoDocstring – для SublimeText.
- Python DocBlock Package – для Atom.
- Autodoc – для PyCharm.

В PyCharm существует встроенный функционал добавления документации к коду. Для этого нужно:

- Переместить курсор под объявление функции.
- Написать тройные кавычки """ и нажмите "Enter".