gpt4docstrings

Miguel Otero Pedrido

Oct 28, 2023

USER GUIDE

1 Requirements 3

2 Installation 5

3 Usage 7

4 Example 9

5 Contributing 13

6 License 15

7 Issues 17

8 Credits 19

Python Module Index 27

Index 29

`gpt4docstrings` is a library that helps you to write docstrings for your Python code. Select a path / paths where you want `gpt4docstrings` to be applied and wait for the results!!

CHAPTER

ONE

REQUIREMENTS

`gpt4docstrings` supports Python 3.9 and above.

CHAPTER

TWO

INSTALLATION

Warning: At the moment, this library is under heavy development, so it is recommended to always install the latest version.

You can install *gpt4docstrings* via pip from PyPI:

```
$ pip install -U gpt4docstrings
```

CHAPTER

THREE

USAGE

To run `gpt4docstrings` on a specific file, run this command.

```
gpt4docstrings my_file.py
```

Remember that, if you don't have your OpenAI API Key defined as an Environment Variable (OPENAI_API_KEY), `gpt4docstrings` can accept the API Key as an option.

```
gpt4docstrings --api_key sk-xxxxxxxxxxx my_file.py
```

Be aware that, as a safety measure , `gpt4docstrings` won't overwrite the file in place. Instead, it will generate a `patch` called `gpt4docstring_docstring_generator_patch.diff` that contains the information about all the changes.

If you want to apply the `patch` to the file, simply run:

```
patch -p1 < gpt4docstring_docstring_generator_patch.diff
```

In case you don't want to generate a `patch` file and modify the files in place, you'll need to add the `--overwrite, -w` option:

```
gpt4docstrings -w my_file.py
```

You can also apply `gpt4docstrings` to folders recursively.

```
gpt4docstrings src/
```

Another quite common situation is that you may want to exclude the `tests/` folder, for example, from the generation of docstrings. Doing this is very simple.

```
gpt4docstrings --exclude tests/ src/
```

By default, `gpt4docstrings` generates `google` style docstrings. However, you can choose between: `google`, `numpy`, `epytext`, `reStructuredText`.

You can specify the docstring style using the `-st` option. For example:

```
gpt4docstrings -st epytext my_file.py
```

For more information about all the available options, you can check the `help` info:

```
Usage: gpt4docstrings [OPTIONS] [PATHS]...

Options:
  -h, --help                      Show this message and exit.
```

(continues on next page)

(continued from previous page)

```
  -S, --ignore-setters            Ignore methods with property setter
                                  decorators.
  -P, --ignore-property-decorators
                                  Ignore methods with property setter/getter
                                  decorators.
  -n, --ignore-nested-functions   Ignore nested functions and methods.
  -C, --ignore-nested-classes     Ignore nested classes.
  -i, --ignore-init-method        Ignore `__init__` method of classes.
  -s, --ignore-semiprivate        Ignore semiprivate classes, methods, and
                                  functions starting with a single underscore.
  -p, --ignore-private            Ignore private classes, methods, and
                                  functions starting with two underscores.
                                  [default: False]
  -w, --overwrite                 If `True`, it will directly write the
                                  docstrings into the files (it will not
                                  generate git patches)
  -v, --verbose INTEGER           Verbosity parameter. Defaults to 0.
  -e, --exclude PATH              Exclude PATHs of files and/or directories.
                                  Multiple `-e/--exclude` invocations
                                  supported.
  -k, --api_key TEXT              OpenAI's API key. If not provided,
                                  `gpt4docstrings` will try to access
                                  `OPENAI_API_KEY` environment variable.
  -st, --style TEXT               Docstring style, which must be one of
                                  'google', 'reStructuredText', 'epytext',
                                  'numpy'
  -m, --model TEXT                The model to be used by `gpt4docstrings`. By
                                  default, `gpt-3.5-turbo`.
```

I also encourage you to see the [Command-line Reference] for more details!!

CHAPTER

FOUR

EXAMPLE

Here is a full example using `gpt4docstring` to generate docstrings for the python code inside `example/example.py`.

```python
import asyncio


async def async_example():
    await asyncio.sleep(2)


class MyClass:

    def __init__(self, value):
        self.value = value

    @staticmethod
    def nested_method():
        def inner_function():
            print("Nested method inner function")
        print("Nested method start")
        inner_function()
        print("Nested method completed")
```

We'll create `numpy` docstrings in this case and will generate a patch file (default) instead of directly overwriting the file directly. We'll also increase the level of verbosity to see some additional information.

> **Warning:** We are assuming you already have the OpenAI API Key set as an Environment Variable. Otherwise, this example won't work.

```
gpt4docstrings example/example.py -v 1 -st numpy
```

After it finishes documenting, we should see a new patch file on our directory called `gpt4docstring_docstring_generator_patch.diff`.

To apply the patch, simply run:

```
patch -p1 < gpt4docstring_docstring_generator_patch.diff
```

The result should be similar to the following (`gpt-3.5-turbo` temperature is set to 1, so you should expect different results every time you run the command)

```python
import asyncio


async def async_example():
    """
    An asynchronous example function.

    This function asynchronously sleeps for 2 seconds.

    Returns
    -------
    None
        This function does not return any value.
    """
    await asyncio.sleep(2)


class MyClass:
    """
    A class representing MyClass.

    Parameters
    ----------
    value : any
        The initial value for MyClass.

    Methods
    -------
    nested_method : static method
        A nested static method within MyClass.
    """
    def __init__(self, value):
        """
        Initialize a new instance of the class.

        Parameters
        ----------
        value : any
            The initial value for the instance.

        Returns
        -------
        None

        Raises
        ------
        None
        """
        self.value = value

    @staticmethod
    def nested_method():
        """
```

(continues on next page)

(continued from previous page)

```
        This static method demonstrates a nested method.

        Raises
        ------
        None

        Returns
        -------
        None
        """
        def inner_function():
            """
            Inner function.

            This function performs a nested method inner function.

            Parameters
            ----------
            None

            Returns
            -------
            None
            """
            print("Nested method inner function")

        print("Nested method start")
        inner_function()
        print("Nested method completed")
```

Suppose now that we want to modify the docstring type. For example, suppose we want to use `epytext` style. That's very easy with `gpt4docstrings`, we just need to run the same command changing the docstring style.

> Be aware that, by default, `gpt4docstrings` will allways generate docstrings for undocumented functions / classes and also translate the existing ones to follow the provided style. If you just want to generate docstrings (no translation), simply set the `-t` flag to `False`.

```
gpt4docstrings example/example.py -st epytext
```

If we apply the patch, we'll get the previous code with the docstrings translated:

```
import asyncio


async def async_example():
    """
    An asynchronous example function.

    This function asynchronously sleeps for 2 seconds.

    @rtype: None
    @return: This function does not return any value.
    """
```

(continues on next page)

(continued from previous page)

```
    await asyncio.sleep(2)


class MyClass:
    """
    A class representing MyClass.

    @type value: any
    @ivar value: The initial value for MyClass.

    @type nested_method: static method
    @ivar nested_method: A nested static method within MyClass.
    """
    def __init__(self, value):
        """
        Initialize a new instance of the class.

        @param value: The initial value for the instance.
        @type value: any

        @return: None

        @raise None
        """
        self.value = value

    @staticmethod
    def nested_method():
        """
        This static method demonstrates a nested method.

        @rtype: None
        @return: None

        @raise: None
        """
        def inner_function():
            """
            Inner function.

            This function performs a nested method inner function.

            @rtype: None
            """
            print("Nested method inner function")

        print("Nested method start")
        inner_function()
        print("Nested method completed")
```

CHAPTER

FIVE

CONTRIBUTING

Contributions are very welcome. To learn more, see the *Contributor Guide*.

CHAPTER

SIX

LICENSE

Distributed under the terms of the *MIT license*, *gpt4docstrings* is free and open source software.

CHAPTER

SEVEN

ISSUES

If you encounter any problems, please file an issue along with a detailed description.

CHAPTER

EIGHT

CREDITS

This project was generated from @cjolowicz's Hypermodern Python Cookiecutter template.

8.1 Command Line Interface (CLI)

The first option when using `gpt4docstrings` is to use it as a Command Line Interface (CLI). The options available for the CLI can be seen below.

8.1.1 gpt4docstrings

```
gpt4docstrings [OPTIONS] [PATHS]...
```

Options

-h, --help
Show this message and exit.

-S, --ignore-setters
Ignore methods with property setter decorators.

Default
`False`

-P, --ignore-property-decorators
Ignore methods with property setter/getter decorators.

Default
`False`

-n, --ignore-nested-functions
Ignore nested functions and methods.

Default
`False`

-C, --ignore-nested-classes
Ignore nested classes.

Default
`False`

-i, --ignore-init-method

Ignore *__init__* method of classes.

Default
`False`

-s, --ignore-semiprivate

Ignore semiprivate classes, methods, and functions starting with a single underscore.

Default
`False`

-p, --ignore-private

Ignore private classes, methods, and functions starting with two underscores. [default: False]

-w, --overwrite

If *True*, it will directly write the docstrings into the files (it will not generate git patches)

Default
`False`

-v, --verbose <verbose>

Verbosity parameter. Defaults to 0.

-e, --exclude <exclude>

Exclude PATHs of files and/or directories. Multiple *-e/–exclude* invocations supported.

-k, --api_key <api_key>

OpenAI's API key. If not provided, *gpt4docstrings* will try to access *OPENAI_API_KEY* environment variable.

-st, --style <style>

Docstring style, which must be one of 'google', 'reStructuredText', 'epytext', 'numpy'

-t, --translate

If *True*, instead of creating new docstrings, it will translate the existing ones into the provided style

Default
`True`

-m, --model <model>

The model to be used by *gpt4docstrings*. By default, *gpt-3.5-turbo*.

Arguments

PATHS

Optional argument(s)

8.2 Contributor Guide

Thank you for your interest in improving this project. This project is open-source under the MIT license and welcomes contributions in the form of bug reports, feature requests, and pull requests.

Here is a list of important resources for contributors:

- Source Code
- Documentation

- Issue Tracker
- *Code of Conduct*

8.2.1 How to report a bug

Report bugs on the Issue Tracker.

When filing an issue, make sure to answer these questions:

- Which operating system and Python version are you using?
- Which version of this project are you using?
- What did you do?
- What did you expect to see?
- What did you see instead?

The best way to get your bug fixed is to provide a test case, and/or steps to reproduce the issue.

8.2.2 How to request a feature

Request features on the Issue Tracker.

8.2.3 How to set up your development environment

You need Python 3.7+ and the following tools:

- Poetry
- Nox
- nox-poetry

Install the package with development requirements:

```
$ poetry install
```

You can now run an interactive Python session, or the command-line interface:

```
$ poetry run python
$ poetry run gpt4docstrings
```

8.2.4 How to test the project

Run the full test suite:

```
$ nox
```

List the available Nox sessions:

```
$ nox --list-sessions
```

You can also run a specific Nox session. For example, invoke the unit test suite like this:

```
$ nox --session=tests
```

Unit tests are located in the *tests* directory, and are written using the pytest testing framework.

8.2.5 How to submit changes

Open a pull request to submit changes to this project.

Your pull request needs to meet the following guidelines for acceptance:

- The Nox test suite must pass without errors and warnings.
- Include unit tests. This project maintains 100% code coverage.
- If your changes add functionality, update the documentation accordingly.

Feel free to submit early, though—we can always iterate on this.

To run linting and code formatting checks before committing your change, you can install pre-commit as a Git hook by running the following command:

```
$ nox --session=pre-commit -- install
```

It is recommended to open an issue before starting work on anything. This will allow a chance to talk it over with the owners and validate your approach.

8.3 Contributor Covenant Code of Conduct

8.3.1 Our Pledge

We as members, contributors, and leaders pledge to make participation in our community a harassment-free experience for everyone, regardless of age, body size, visible or invisible disability, ethnicity, sex characteristics, gender identity and expression, level of experience, education, socio-economic status, nationality, personal appearance, race, caste, color, religion, or sexual identity and orientation.

We pledge to act and interact in ways that contribute to an open, welcoming, diverse, inclusive, and healthy community.

8.3.2 Our Standards

Examples of behavior that contributes to a positive environment for our community include:

- Demonstrating empathy and kindness toward other people
- Being respectful of differing opinions, viewpoints, and experiences
- Giving and gracefully accepting constructive feedback
- Accepting responsibility and apologizing to those affected by our mistakes, and learning from the experience
- Focusing on what is best not just for us as individuals, but for the overall community

Examples of unacceptable behavior include:

- The use of sexualized language or imagery, and sexual attention or advances of any kind
- Trolling, insulting or derogatory comments, and personal or political attacks
- Public or private harassment

- Publishing others' private information, such as a physical or email address, without their explicit permission
- Other conduct which could reasonably be considered inappropriate in a professional setting

8.3.3 Enforcement Responsibilities

Community leaders are responsible for clarifying and enforcing our standards of acceptable behavior and will take appropriate and fair corrective action in response to any behavior that they deem inappropriate, threatening, offensive, or harmful.

Community leaders have the right and responsibility to remove, edit, or reject comments, commits, code, wiki edits, issues, and other contributions that are not aligned to this Code of Conduct, and will communicate reasons for moderation decisions when appropriate.

8.3.4 Scope

This Code of Conduct applies within all community spaces, and also applies when an individual is officially representing the community in public spaces. Examples of representing our community include using an official e-mail address, posting via an official social media account, or acting as an appointed representative at an online or offline event.

8.3.5 Enforcement

Instances of abusive, harassing, or otherwise unacceptable behavior may be reported to the community leaders responsible for enforcement at miguel.otero.pedrido.1993@gmail.com. All complaints will be reviewed and investigated promptly and fairly.

All community leaders are obligated to respect the privacy and security of the reporter of any incident.

8.3.6 Enforcement Guidelines

Community leaders will follow these Community Impact Guidelines in determining the consequences for any action they deem in violation of this Code of Conduct:

1. Correction

Community Impact: Use of inappropriate language or other behavior deemed unprofessional or unwelcome in the community.

Consequence: A private, written warning from community leaders, providing clarity around the nature of the violation and an explanation of why the behavior was inappropriate. A public apology may be requested.

2. Warning

Community Impact: A violation through a single incident or series of actions.

Consequence: A warning with consequences for continued behavior. No interaction with the people involved, including unsolicited interaction with those enforcing the Code of Conduct, for a specified period of time. This includes avoiding interactions in community spaces as well as external channels like social media. Violating these terms may lead to a temporary or permanent ban.

3. Temporary Ban

Community Impact: A serious violation of community standards, including sustained inappropriate behavior.

Consequence: A temporary ban from any sort of interaction or public communication with the community for a specified period of time. No public or private interaction with the people involved, including unsolicited interaction with those enforcing the Code of Conduct, is allowed during this period. Violating these terms may lead to a permanent ban.

4. Permanent Ban

Community Impact: Demonstrating a pattern of violation of community standards, including sustained inappropriate behavior, harassment of an individual, or aggression toward or disparagement of classes of individuals.

Consequence: A permanent ban from any sort of public interaction within the community.

8.3.7 Attribution

This Code of Conduct is adapted from the Contributor Covenant, version 2.1, available at https://www.contributor-covenant.org/version/2/1/code_of_conduct.html.

Community Impact Guidelines were inspired by Mozilla's code of conduct enforcement ladder.

For answers to common questions about this code of conduct, see the FAQ at https://www.contributor-covenant.org/faq. Translations are available at https://www.contributor-covenant.org/translations.

8.4 License

8.5 utils

8.6 docstrings_generators

class gpt4docstrings.docstrings_generators.base.**DocstringGenerator**

An abstract base class for docstring generators that provides a blueprint for generating docstrings for functions and classes.

generate_docstring(source

str) -> dict: Generate a docstring for the provided source code.

class gpt4docstrings.docstrings_generators.chatgpt_generator.**ChatGPTDocstringGenerator**(*api_key: str*, *model_name: str*, *docstring_style: str*)

A class for generating Python docstrings using ChatGPT.

8.7 gpt4docstrings

class gpt4docstrings.generate_docstrings.**GPT4Docstrings**(*paths: str | List[str]*, *excluded=None*, *model: str = 'gpt-3.5-turbo'*, *docstring_style: str = 'google'*, *translate: bool = True*, *api_key: str | None = None*, *verbose: int = 0*, *config: GPT4DocstringsConfig | None = None*)

async generate_file_docstrings(*filename: str*, *file_content: str*, *nodes: List[GPT4DocstringsNode]*) → str

Generates docstrings for a single file.

Parameters

- **filename** (*str*) – The path of the file to generate docstrings for.
- **file_content** (*str*) – The content of the file to be processed.
- **nodes** (*List[GPT4DocstringsNode]*) – The list of *GPT4DocstringsNode* containing nodes from classes and functions

Returns
The new file content

get_filenames_from_paths() → List[str]

Retrieves the filenames from the input paths.

Returns
The list of filenames.

Return type
List[str]

`run()`

Generates docstrings for the input files or directories.

`async translate_file_docstrings(`*filename: str*, *file_content: str*, *nodes: List[GPT4DocstringsNode]*`)` → str

Generates docstrings for a single file.

Parameters

- **filename** (*str*) – The path of the file to generate docstrings for.
- **file_content** (*str*) – The content of the file to be processed.
- **nodes** (*List[GPT4DocstringsNode]*) – The list of *GPT4DocstringsNode* containing nodes from classes and functions

Returns

The new file content

PYTHON MODULE INDEX

g
gpt4docstrings.utils, 25

INDEX

Symbols

-C
    gpt4docstrings command line option, 19
-P
    gpt4docstrings command line option, 19
-S
    gpt4docstrings command line option, 19
--api_key
    gpt4docstrings command line option, 20
--exclude
    gpt4docstrings command line option, 20
--help
    gpt4docstrings command line option, 19
--ignore-init-method
    gpt4docstrings command line option, 19
--ignore-nested-classes
    gpt4docstrings command line option, 19
--ignore-nested-functions
    gpt4docstrings command line option, 19
--ignore-private
    gpt4docstrings command line option, 20
--ignore-property-decorators
    gpt4docstrings command line option, 19
--ignore-semiprivate
    gpt4docstrings command line option, 20
--ignore-setters
    gpt4docstrings command line option, 19
--model
    gpt4docstrings command line option, 20
--overwrite
    gpt4docstrings command line option, 20
--style
    gpt4docstrings command line option, 20
--translate
    gpt4docstrings command line option, 20
--verbose
    gpt4docstrings command line option, 20
-e
    gpt4docstrings command line option, 20
-h
    gpt4docstrings command line option, 19
-i
    gpt4docstrings command line option, 19
-k
    gpt4docstrings command line option, 20
-m
    gpt4docstrings command line option, 20
-n
    gpt4docstrings command line option, 19
-p
    gpt4docstrings command line option, 20
-s
    gpt4docstrings command line option, 20
-st
    gpt4docstrings command line option, 20
-t
    gpt4docstrings command line option, 20
-v
    gpt4docstrings command line option, 20
-w
    gpt4docstrings command line option, 20

C

ChatGPTDocstringGenerator (*class in gpt4docstrings.docstrings_generators.chatgpt_generator*), 25

D

DocstringGenerator (*class in gpt4docstrings.docstrings_generators.base*), 25

G

generate_file_docstrings() (*gpt4docstrings.generate_docstrings.GPT4Docstrings method*), 25
get_filenames_from_paths() (*gpt4docstrings.generate_docstrings.GPT4Docstrings method*), 25
GPT4Docstrings (*class in gpt4docstrings.generate_docstrings*), 25
gpt4docstrings command line option
    -C, 19
    -P, 19
    -S, 19

--api_key, 20
--exclude, 20
--help, 19
--ignore-init-method, 19
--ignore-nested-classes, 19
--ignore-nested-functions, 19
--ignore-private, 20
--ignore-property-decorators, 19
--ignore-semiprivate, 20
--ignore-setters, 19
--model, 20
--overwrite, 20
--style, 20
--translate, 20
--verbose, 20
-e, 20
-h, 19
-i, 19
-k, 20
-m, 20
-n, 19
-p, 20
-s, 20
-st, 20
-t, 20
-v, 20
-w, 20
PATHS, 20
gpt4docstrings.utils
module, 25

M

module
gpt4docstrings.utils, 25

P

PATHS
gpt4docstrings command line option, 20

R

run() (*gpt4docstrings.generate_docstrings.GPT4Docstrings method*), 25

T

translate_file_docstrings() (*gpt4docstrings.generate_docstrings.GPT4Docstrings method*), 26