Docstring Patterns in Python

Documenting code is a good habit, and aspiring developers and programmers should develop a habit of documenting their code in the early phases of their coding journey. Documenting a source code not only improves the readability and management of the source code but also makes it extremely easy for new contributors to the source code to understand it.

Docstrings are string literals written inside source codes. They act as comments or documentation for pieces of code. Docstrings are used to describe classes, functions, and sometimes, even the files. In order words, a docstring act as metadata about the code snippets.

A docstring contains all the relevant information about what they are describing. For a class, it holds information about

- the class
- class functions
- class attributes.

For functions, it holds details about

- parameters
- data types of parameters
- default values of parameters
- short descriptions about parameters
- what the function returns
- data type of what is returned by the function
- errors and exceptions that the function raises and short descriptions about

There are several docstring patterns that professional Python developers use to document their code. Instead of using the existing ones, one can also create their own docstring pattern. Still, this decision solely depends on the individual developer or the team of developers.

In this article, we will talk about some of the best docstring patterns for the Python programming language.

Docstring Patterns in Python

Following are some of the best docstring patterns that are commonly used in the industry by Python professionals.

Epytext Pattern

The Epytext pattern is a docstring pattern similar to the JavaDoc. It is a part of the Epydoc tool that is used to generate documentation for Python modules using their docstrings. Following is an example of the Epytext pattern.

```
"""
Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor
incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud
```

```
exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat.

@param parameter1: this is the first parameter.
@param parameter2: this is the second parameter.
@return: this is a description of what is returned by the function.
@raise KeyError: raises an exception.
@raise TypeError: raises an exception.
"""
```

A brief description of the function is provided at the top. All the parameters of the function are written using the `@param` keyword. An explanation of what is returned by the function is written next to the `@return` keyword. If the function raises any errors or exceptions, they are written using the `@raise` keyword.

reST Pattern

The reSt or reStructuredText is a docstring pattern used by the Sphinx, a tool for generating documentation for the Python programming language. This pattern is one of the most used docstrings patterns in the IT industry. This method is also used as an output format in Pyment, a tool to help Python programmers write enhanced code documentation using docstrings. This tool is very beneficial when the code is partially documented or contains no docstrings. The JetBrains PyCharm IDE or Integrated Development Environment also uses the reST pattern. Following is an example of the reST pattern.

```
"""
Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor
incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud
exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat.

:param parameter1: this is the first parameter.
:param parameter2: this is the second parameter.
:return: this is a description of what is returned by the function.
:raise KeyError: raises an exception.
:raise TypeError: raises an exception.
"""
```

Similar to the Epytext pattern, everything is the same for this pattern except it uses a colon or `:` as a prefix for keywords instead of at sign or `@` in the case of the Epytext pattern. A concise or comprehensive description of the method sits at the top. All the parameters sit next to the `:param` keyword. An explanation of what is returned by the method is written next to the `:return` keyword. And, details about all the errors are placed next to the `:raise` keyword.

Google Pattern

Another pattern on the list is the Google pattern. Technically, its name is not Google pattern, but it is a pattern which was developed by Google. It is a clean pattern that organises details under headings. The Sphinx tool is capable of recognising this pattern too and generating documentation. This pattern is one of the most docstrings patterns too. Following is an example of the Google pattern.

```
"""
Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor
incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud
exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat.

Args:
    parameter1: this is the first parameter.
    parameter2: this is the second parameter.

Returns:
    this is a description of what is returned by the function.

Raises:
    KeyError: raises an exception.
    TypeError: raises an exception.
"""
```

Like the previous patterns, a succinct description of the method sits at the top. The description is followed by heading such as `Args`, `Returns`, and `Raises`. Under the `Args` heading, all the parameters, along with some details about them, such as their type and default values, are placed. A description of what is returned by the function is placed under the `Returns` heading. Lastly, under the `Raises` heading, errors or exceptions and details about them are written.

Numpydoc Pattern

The `numpy` module has its own docstring patterns known as the Numpydoc pattern. This pattern is similar to the Google pattern, and it is recognised by the Sphinx tool. Similar to the Google pattern, information is organised under headings. Following is an example of the Numpydoc pattern.

```
"""
Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor
incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud
exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat.

Parameters
----------
parameter1 : int
    this is the first parameter.
parameter2 : str, "default value"
    this is the second parameter.

Returns
-------
string
    returns a string value.

Raises
------
KeyError
    raises an exception.
TypeError
    raises an exception.
"""
```

The description of the method is written at the top. Other details about the method are organised under headings: `Parameters`, `Returns` and `Raises`. All the details about the parameters, including their default value, value type, etc., are placed under the `Parameters` heading. All the details about what is returned by the function, including the data type, sit under the `Returns` heading. Lastly, information about the errors or exceptions, along with some description about them, is written under the `Raises` heading.