

google python docstring style guide

Google Python Docstring Style Guide serves as a comprehensive guideline for writing clear and consistent documentation within Python code. Effective documentation is crucial for enhancing code readability and maintainability, allowing developers and users to understand the purpose and functionality of code components efficiently. The Google Python Docstring Style Guide outlines best practices for documenting functions, classes, modules, and methods, ensuring that Python code adheres to a standardized format that is both informative and easy to follow.

Introduction to Docstrings

Docstrings are string literals that appear right after the definition of a function, method, class, or module. They serve as documentation for the object they describe and are accessible through the built-in `help()` function. In Python, docstrings are denoted by triple quotes (`"""` or `'''`). The style guide encourages developers to write meaningful and concise docstrings that effectively communicate the purpose, parameters, return values, and exceptions raised by the code.

Basic Structure of a Docstring

According to the Google Python Docstring Style Guide, a well-structured docstring should include the following components:

1. Summary Line

The first line of the docstring should be a short summary of the function or method's purpose. This line should be concise and begin with a verb in the imperative mood. For example:

```
```python
def calculate_area(radius):
 """Calculate the area of a circle given its radius."""
    ```
```

2. Extended Description (Optional)

Following the summary line, you can provide an extended description of the function, explaining its

behavior, usage, or any other relevant details. This part is optional but can be useful for complex functions.

```
```python
def calculate_area(radius):
 """Calculate the area of a circle given its radius.
```

This function uses the formula:  $\text{area} = \pi \text{ radius}^2$ .

It is important to ensure that the radius is a positive number.

```
"""
```
```

3. Parameters Section

The parameters section describes the input parameters of the function. Each parameter should include its name, type, and description. The format is as follows:

```
```python
Args:
radius (float): The radius of the circle. Must be positive.
```
```

4. Returns Section

The returns section indicates the type of value the function returns along with a description of that value. This section is crucial for understanding the output of the function.

```
```python
Returns:
float: The area of the circle.
```
```

5. Raises Section (Optional)

If the function raises exceptions, it is essential to document them. This section should list the exception types and conditions under which they are raised.

```
```python
Raises:
```

```
ValueError: If radius is not positive.
```

```
'''
```

## 6. Example Section (Optional)

Including examples can significantly enhance the clarity of the documentation. This section provides usage examples that illustrate how to call the function and what the results look like.

```
```python
```

```
Example:
```

```
>>> calculate_area(5)
```

```
78.53981633974483
```

```
'''
```

Complete Example of a Docstring

Combining all the components discussed, a complete docstring for the `calculate_area` function would look like this:

```
```python
```

```
def calculate_area(radius):
```

```
 """Calculate the area of a circle given its radius.
```

```
 This function uses the formula: $\text{area} = \pi \text{ radius}^2$.
```

```
 It is important to ensure that the radius is a positive number.
```

```
 Args:
```

```
 radius (float): The radius of the circle. Must be positive.
```

```
 Returns:
```

```
 float: The area of the circle.
```

```
 Raises:
```

```
 ValueError: If radius is not positive.
```

```
 Example:
```

```
>>> calculate_area(5)
```

```
78.53981633974483
```

```
 """
```

```
'''
```

# Docstring Conventions

In addition to the basic structure, the Google Python Docstring Style Guide provides several conventions that should be followed to maintain consistency across codebases.

## 1. Use Triple Quotes

Always use triple quotes for docstrings, even for single-line docstrings. This practice ensures that the documentation remains consistent.

## 2. Limit Line Length

Keep lines in the docstring to a maximum of 72 characters. This limitation helps maintain readability in terminal and console settings.

## 3. Use Proper Capitalization and Punctuation

- The summary line should be capitalized and end with a period.
- Use full sentences in the extended description and include punctuation.

## 4. Use Verb Tense Consistency

Use the present tense for the summary line and the imperative mood for instructions. For example, “Returns the area” instead of “Returned the area.”

## 5. Maintain Consistent Formatting

Ensure that all sections (Args, Returns, Raises, Example) are consistently formatted. Use the same indentation and layout for each docstring.

# Using Google Style for Classes and Modules

The Google Python Docstring Style Guide also applies to classes and modules, with minor variations in formatting.

## Class Docstrings

For classes, the docstring should describe the class's purpose and functionality. The parameters section should be replaced with an Attributes section to list class attributes.

```
```python
class Circle:
    """A class representing a circle.
```

Attributes:

radius (float): The radius of the circle.

Methods:

area: Returns the area of the circle.

```
"""
```
```

## Module Docstrings

Module docstrings should describe the overall purpose of the module and can include information about the module's contents and usage. The module docstring should be placed at the top of the module file.

```
```python
"""This module provides functions to calculate geometric properties.
```

It includes functions for calculating the area and perimeter of various shapes.

```
"""
```
```

## Conclusion

The Google Python Docstring Style Guide is an invaluable resource for Python developers aiming to create

clear, comprehensive, and standardized documentation for their code. By adhering to the guidelines outlined in this article, developers can improve the maintainability of their codebases and contribute to a culture of quality documentation within the Python community. Following best practices in docstring writing not only facilitates collaboration among developers but also enhances the user experience for anyone interacting with the code. Ultimately, well-documented code is easier to understand, debug, and extend, ensuring long-term success in software development projects.

## Frequently Asked Questions

### What is the Google Python Docstring Style Guide?

The Google Python Docstring Style Guide is a set of conventions for writing docstrings in Python code, aimed at improving code readability and maintainability. It provides guidelines on formatting, including how to structure docstrings for modules, classes, methods, and functions.

### How should a function docstring be structured according to the Google style?

A function docstring should begin with a short summary of the function's purpose, followed by a more detailed description if necessary. After that, it should list the parameters with their types and descriptions, the return type and description, exceptions raised, and any other relevant notes, all formatted consistently.

### What is the recommended format for parameters in Google-style docstrings?

Parameters in Google-style docstrings should be listed in the format: ``param_name (param_type):`  
Description of the parameter.` Each parameter should be on a new line, and the type should be enclosed in parentheses directly following the parameter name.

### How should you document return values in Google-style docstrings?

Return values in Google-style docstrings should be documented under a section labeled 'Returns:' followed by the return type and a description. The format is: ``Returns: return_type: Description of the return value.`` This section should clearly indicate what the function returns.

### Can you provide an example of a class docstring in Google style?

Certainly! A class docstring in Google style should describe the class briefly, followed by a more detailed explanation if necessary. For example: ``class ExampleClass:`  
`""This class represents an example.

More detailed description of the class.

```
"""
```

## [Google Python Docstring Style Guide](#)

Google Python Docstring Style Guide

## Related Articles

- [grande illusions books i ii](#)
- [grokking the system design interview](#)
- [greek mathematical thought and the origin of algebra dover books on mathematics](#)

[Back to Home](#)