Sphinx doc

Release v.1.0

Nandhagopalan Elangovan

May 28, 2022

CONTENTS:

CHAPTER

ONE

PREPROCESS

CHAPTER

TWO

SRC

2.1 example module

Example Google style docstrings.

This module demonstrates documentation as specified by the Google Python Style Guide. Docstrings may extend over multiple lines. Sections are created with a section header and a colon followed by a block of indented text.

Example

Examples can be given using either the `Example` or `Examples` sections. Sections support any reStructuredText formatting, including literal blocks:

```
$ python example_google.py
```

Section breaks are created by resuming unindented text. Section breaks are also implicitly created anytime a new section starts.

`example.`**`module_level_variable1`**

Module level variables may be documented in either the `Attributes` section of the module docstring, or in an inline docstring immediately following the variable.

Either form is acceptable, but the two should not be mixed. Choose one convention to document module level variables and be consistent with it.

Type int

class `example.`**`ExampleClass`**(*param1*, *param2*, *param3*)

Bases: `object`

The summary line for a class docstring should fit on one line.

If the class has public attributes, they may be documented here in an `Attributes` section and follow the same formatting as a function's `Args` section. Alternatively, attributes may be documented inline with the attribute's declaration (see __init__ method below).

Properties created with the `@property` decorator should be documented in the property's getter method.

`attr1`

Description of *attr1*.

Type str

`attr2`

Description of *attr2*.

Type `int`, optional

Example of docstring on the __init__ method.

The __init__ method may be documented in either the class level docstring, or as a docstring on the __init__ method itself.

Either form is acceptable, but the two should not be mixed. Choose one convention to document the __init__ method and be consistent with it.

Note: Do not include the *self* parameter in the `Args` section.

Parameters

- **param1** (*str*) – Description of *param1*.
- **param2** (`int`, optional) – Description of *param2*. Multiple lines are supported.
- **param3** (`list` of `str`) – Description of *param3*.

`attr3`

Doc comment *inline* with attribute

`attr4`

Doc comment *before* attribute, with type specified

Type list of str

`attr5`

Docstring *after* attribute, with type specified.

Type str

`example_method`(*param1*, *param2*)

Class methods are similar to regular functions.

Note: Do not include the *self* parameter in the `Args` section.

Parameters

- **param1** – The first parameter.
- **param2** – The second parameter.

Returns True if successful, False otherwise.

`property readonly_property`

Properties should be documented in their getter method.

Type str

`property readwrite_property`

Properties with both a getter and setter should only be documented in their getter method.

If the setter method contains notable behavior, it should be mentioned here.

Type list of str

exception example.**ExampleError**(*msg*, *code*)

Bases: Exception

Exceptions are documented in the same way as classes.

The __init__ method may be documented in either the class level docstring, or as a docstring on the __init__ method itself.

Either form is acceptable, but the two should not be mixed. Choose one convention to document the __init__ method and be consistent with it.

Note: Do not include the *self* parameter in the Args section.

Parameters

- **msg** (*str*) – Human readable string describing the exception.
- **code** (int, optional) – Error code.

msg

Human readable string describing the exception.

Type str

code

Exception error code.

Type int

example.**example_generator**(*n*)

Generators have a Yields section instead of a Returns section.

Parameters **n** (*int*) – The upper limit of the range to generate, from 0 to *n* - 1.

Yields *int* – The next number in the range of 0 to *n* - 1.

Examples

Examples should be written in doctest format, and should illustrate how to use the function.

```
>>> print([i for i in example_generator(4)])
[0, 1, 2, 3]
```

example.**function_with_pep484_type_annotations**(*param1: int*, *param2: str*) → bool

Example function with PEP 484 type annotations.

Parameters

- **param1** – The first parameter.
- **param2** – The second parameter.

Returns The return value. True for success, False otherwise.

example.**function_with_types_in_docstring**(*param1*, *param2*)

Example function with types documented in the docstring.

PEP 484 type annotations are supported. If attribute, parameter, and return types are annotated according to PEP 484, they do not need to be included in the docstring:

Parameters

- **param1** (*int*) – The first parameter.
- **param2** (*str*) – The second parameter.

Returns The return value. True for success, False otherwise.

Return type bool

example.**module_level_function**(*param1*, *param2=None*, **args*, ***kwargs*)

This is an example of a module level function.

Function parameters should be documented in the `Args` section. The name of each parameter is required. The type and description of each parameter is optional, but should be included if not obvious.

If *args or **kwargs are accepted, they should be listed as `*args` and `**kwargs`.

The format for a parameter is:

```
name (type): description
    The description may span multiple lines. Following
    lines should be indented. The "(type)" is optional.

    Multiple paragraphs are supported in parameter
    descriptions.
```

Parameters

- **param1** (*int*) – The first parameter.
- **param2** (`str`, optional) – The second parameter. Defaults to None. Second line of description should be indented.
- ***args** – Variable length argument list.
- ****kwargs** – Arbitrary keyword arguments.

Returns

True if successful, False otherwise.

The return type is optional and may be specified at the beginning of the `Returns` section followed by a colon.

The `Returns` section may span multiple lines and paragraphs. Following lines should be indented to match the first line.

The `Returns` section supports any reStructuredText formatting, including literal blocks:

```
{
    'param1': param1,
    'param2': param2
}
```

Return type bool

Raises

- **AttributeError** – The `Raises` section is a list of all exceptions that are relevant to the interface.
- **ValueError** – If *param2* is equal to *param1*.

example.**module_level_variable2 = 98765**

Module level variable documented inline.

The docstring may span multiple lines. The type may optionally be specified on the first line, separated by a colon.

Type int

CHAPTER

THREE

INDICES AND TABLES

- genindex
- modindex
- search

PYTHON MODULE INDEX

INDEX