dummypackage Documentation

Release 0.9.1

Dummy Dumson

Feb 25, 2019

CONTENTS:

CHAPTER

ONE

DUMMYPACKAGE PACKAGE

1.1 Subpackages

1.1.1 dummypackage.nested package

1.1.1.1 Submodules

1.1.1.2 dummypackage.nested.baz_module module

Module with dummy functionality that isn't tested, to show a gap in the code coverage.

class `dummypackage.nested.baz_module.`**`Baz`**

Bases: `object`

A class with many lines that don't get tested

`a = 123`

`b = 123`

`c = 123`

`d = 123`

`e = 123`

`f = 123`

`g = 123`

`h = 123`

`i = 123`

`j = 123`

`k = 123`

`m = 123`

`n = 123`

`o = 123`

`p = 123`

`q = 123`

`r = 123`

1.1.1.3 Module contents

1.2 Submodules

1.3 dummypackage.bar_module module

Module mimicking foo with more expensive memory and runtime requirements.

class `dummypackage.bar_module.`**`Bar`**`(`*size=1000000*`)`

Bases: *`dummypackage.foo_module.Foo`*

Similar to Foo, with higher memory and runtime requirements.

1.4 dummypackage.foo_module module

Module containing a simple class with low memory and runtime requirements.

class `dummypackage.foo_module.`**`Foo`**`(`*size=1000000*`)`

Bases: `object`

A simple class with low memory and runtime requirements.

`get_result``()`

This function does something.

Returns a number stored in `self._result`

Return type integer or float

`loop``(`*times*`)`

Restart result and run computation a number of times.

Parameters **`times`** (*`int`*) – non-negative number.

1.5 Module contents

Main init file docstring.
It exemplifies the usage of restructured text, like:

- *Italics*
- **Bold**
- **Numbered and nested lists:**
    1. This is a numbered list
    2. Nested lists have at least three characters indentation
- `Inline literals`
- Parameter fields: see class and method docstrings.

Note that lines above 80 characters would break `flake8` and therefore have to be wrapped. This can be achieved with `|` blocks.

This is a new line.

1.5.1 Section about sections:

- Surrounding chars have to be at least as long as the title
- No explicit hierarchy, but this recommended: `#, *, =, -, ^, "` (the first two with overline).

1.5.1.1 Subsection:

To exemplify the usage of LaTeX and nested quotes, nothing best that the words of Isaac Newton himself:

> "If I have seen further it is by standing on the shoulders of Giants."

Or, in other words:

$$\sum_{k=1}^{\infty} k = -\frac{1}{12}$$

1.5.1.2 Emphasis:

Note: The sum of all parameters cannot exceed infinity

Warning: If the sum of all parameters exceeds infinity, behaviour is undefined!

1.5.1.3 Function descriptions:

Sphinx formatting:

`dummypackage.`**`add`**`(`*a*, *b=None*`)`

This is a cool function.

Parameters

- **a** (*`int or float`*) – a number
- **b** (*`int, float or None`*) – another number

Returns `a+b`. If b is none, returns `a`

Return type integer or float

Note: Neither `a` nor `b` can be infinity!

Google formatting:

This function does something.

Args: name (str): The name to use.

Kwargs: state (bool): Current state to be in.

Returns: int. The return code:

```
0 -- Success!
1 -- No good.
2 -- Try again.
```

Raises: AttributeError, KeyError

Usage example:

```
>>> print public_fn_with_googley_docstring(name='foo', state=None)
0
```

BTW, this always returns 0. **NEVER** use with `MyPublicClass`.

1.5.2 Other structures:

Field lists:

Author Homer J. Simpson

Email `hjs@compuglobalhypermega.net`

Literal blocks, preceded by double colon:

```
This is a literal block

Markups are **not** rendered here.
```

Doctest blocks can be tested by the doc tool:

```
>>> [factorial(n) for n in range(6)]
[1, 1, 2, 6, 24, 120]
>>> [factorial(long(n)) for n in range(6)]
[1, 1, 2, 6, 24, 120]
```

Grid tables must be indented:

Header 1	Header 2	Header 3
body row 1	column 2	column 3
body row 2	Cells may span columns.	
body row 3	Cells may span rows.	• Cells • contain • blocks.
body row 4		

Simple table:

Inputs		Output
A	B	A or B
False	False	False
True	False	True
False	True	True
True	True	True

CHAPTER

TWO

INDICES AND TABLES

- genindex
- modindex
- search

PYTHON MODULE INDEX

INDEX