

---

# **dummyspackage Documentation**

***Release 1.2.1***

**Dummy Dumson**

**Mar 24, 2019**

CONTENTS:

1 dummypackage package 1

1.1 Subpackages 1

1.1.1 dummypackage.nested package 1

1.1.1.1 Submodules 1

1.1.1.2 dummypackage.nested.baz\_module module 1

1.1.1.3 Module contents 2

1.2 Submodules 2

1.3 dummypackage.bar\_module module 2

1.4 dummypackage.foo\_module module 2

1.5 Module contents 2

1.5.1 Section about sections: 3

1.5.1.1 Subsection: 3

1.5.1.2 Emphasis: 3

1.5.1.3 Function descriptions: 3

1.5.2 Other structures: 4

2 Indices and tables 6

Python Module Index 7

Index 8

## 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 dummyspackage.bar\_module module

Module mimicking foo with more expensive memory and runtime requirements.

**class** dummyspackage.bar\_module.**Bar** (*size=1000000*)

Bases: *dummyspackage.foo\_module.Foo*

Similar to Foo, with higher memory and runtime requirements.

### 1.4 dummyspackage.foo\_module module

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

**class** dummyspackage.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:

- 1 2 3 4
- *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 better than 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:

`dummyspackage.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.	<ul style="list-style-type: none"> <li>• Cells</li> <li>• contain</li> <li>• blocks.</li> </ul>
body row 4		

Simple table:

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

## INDICES AND TABLES

- `genindex`
- `modindex`
- `search`



## PYTHON MODULE INDEX

### d

- `dummyspackage`, [2](#)
- `dummyspackage.bar_module`, [2](#)
- `dummyspackage.foo_module`, [2](#)
- `dummyspackage.nested`, [2](#)
- `dummyspackage.nested.baz_module`, [1](#)

## INDEX

### A

`a` (*dummypackage.nested.baz\_module.Baz* attribute), 1  
`add()` (in module *dummypackage*), 3

### B

`b` (*dummypackage.nested.baz\_module.Baz* attribute), 1  
`Bar` (class in *dummypackage.bar\_module*), 2  
`Baz` (class in *dummypackage.nested.baz\_module*), 1

### C

`c` (*dummypackage.nested.baz\_module.Baz* attribute), 1

### D

`d` (*dummypackage.nested.baz\_module.Baz* attribute), 1  
`dummypackage` (module), 2  
`dummypackage.bar_module` (module), 2  
`dummypackage.foo_module` (module), 2  
`dummypackage.nested` (module), 2  
`dummypackage.nested.baz_module` (module), 1

### E

`e` (*dummypackage.nested.baz\_module.Baz* attribute), 1

### F

`f` (*dummypackage.nested.baz\_module.Baz* attribute), 1  
`Foo` (class in *dummypackage.foo\_module*), 2

### G

`g` (*dummypackage.nested.baz\_module.Baz* attribute), 1  
`get_result()` (*dummypackage.foo\_module.Foo* method), 2

### H

`h` (*dummypackage.nested.baz\_module.Baz* attribute), 1

### I

`i` (*dummypackage.nested.baz\_module.Baz* attribute), 1

### J

`j` (*dummypackage.nested.baz\_module.Baz* attribute), 1

### K

`k` (*dummypackage.nested.baz\_module.Baz* attribute), 1

### L

`loop()` (*dummypackage.foo\_module.Foo* method), 2

### M

`m` (*dummypackage.nested.baz\_module.Baz* attribute), 1

### N

`n` (*dummypackage.nested.baz\_module.Baz* attribute), 1

### O

`o` (*dummypackage.nested.baz\_module.Baz* attribute), 1

### P

`p` (*dummypackage.nested.baz\_module.Baz* attribute), 1

### Q

`q` (*dummypackage.nested.baz\_module.Baz* attribute), 1

### R

`r` (*dummypackage.nested.baz\_module.Baz* attribute), 1