

---

**default***values*

***Release 0.3.1***

**Dominic Davis-Foster**

**Sep 23, 2020**



**CONTENTS**

<b>1</b>	<b>Installation</b>	<b>3</b>
<b>2</b>	<b>Usage</b>	<b>5</b>
<b>3</b>	<b>API Reference</b>	<b>9</b>
<b>4</b>	<b>Demo</b>	<b>11</b>
<b>5</b>	<b>Overview</b>	<b>13</b>
<b>6</b>	<b>Coding style</b>	<b>15</b>
<b>7</b>	<b>Automated tests</b>	<b>17</b>
<b>8</b>	<b>Type Annotations</b>	<b>19</b>
<b>9</b>	<b>Build documentation locally</b>	<b>21</b>
<b>10</b>	<b>Downloading source code</b>	<b>23</b>
10.1	Building from source . . . . .	23
	<b>Python Module Index</b>	<b>25</b>
	<b>Index</b>	<b>27</b>



**Sphinx extension to show default values in documentation.**

Docs	
Tests	
PyPI	
Activity	
Other	



## INSTALLATION

from PyPI

```
$ python3 -m pip install default_values --user
```

from GitHub

```
$ python3 -m pip install git+https://github.com/domdfcoding/default_values@master --user
```

Enable `default_values` by adding the following to the `extensions` variable in your `conf.py`:

```
extensions = [  
    ..  
    'sphinxcontrib.default_values',  
]
```

For more information see <https://www.sphinx-doc.org/en/master/usage/extensions/index.html#third-party-extensions>.





## USAGE

This extension shows the default values in autodoc-formatted docstrings.

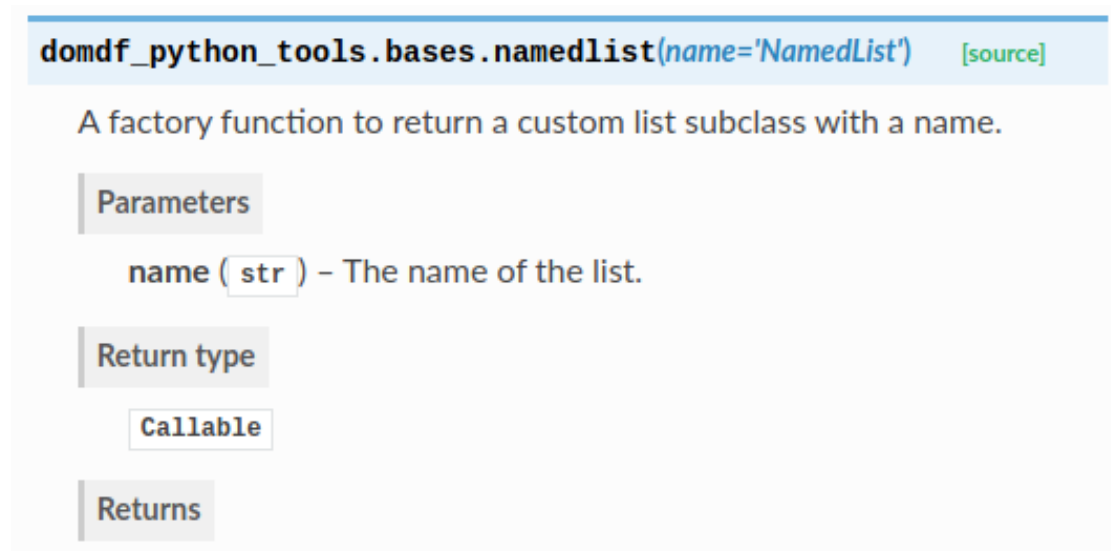
The default behaviour of `autodoc` is to turn this:

```
def namedlist(name: str = "NamedList") -> Callable:
    """
    A factory function to return a custom list subclass with a name.

    :param name: The name of the list.
    :default name: :py:obj:`True`

    :return:
    """
```

into this:



The screenshot shows the rendered documentation for the `namedlist` function. At the top, the function signature `domdf_python_tools.bases.namedlist(name='NamedList')` is displayed in a blue header bar, followed by a [\[source\]](#) link. Below this is the docstring: "A factory function to return a custom list subclass with a name." The documentation is organized into sections: "Parameters" containing "name (str) - The name of the list.", "Return type" containing "Callable", and "Returns".

With `default_values` enabled, the documentation will now look like this:

**domdf\_python\_tools.bases.namedlist**(name='NamedList') [\[source\]](#)

A factory function to return a custom list subclass with a name.

Parameters

name ( str ) – The name of the list. Default `'NamedList'`.

Return type

Callable

Returns

Default values are taken from the function/class signature. They can be overridden using the `:default <argname>: <default value>` option in the docstring:

```
:param name: The name of the list.  
:default name: :py:obj:`True`
```

which will produce:

**domdf\_python\_tools.bases.namedlist**(name='NamedList') [\[source\]](#)

A factory function to return a custom list subclass with a name.

Parameters

name ( str ) – The name of the list. Default `True`.

Return type

Callable

Returns

The value must be formatted how you would like it to be displayed in Sphinx. This can be useful when the default value in the signature is `None` and the true default value is assigned in the function body, such as for a mutable default argument.

The default value can be suppressed using the `:no-default <argname>` option:

```
:param name: The name of the list.  
:no-default name:
```

```
domdf_python_tools.bases.namedlist(name='NamedList') \[source\]
```

A factory function to return a custom list subclass with a name.

#### Parameters

**name** (`str`) – The name of the list.

#### Return type

`Callable`

#### Returns

This allows for default values to be suppressed on a per-argument basis.

No default value is shown if the argument does not have a default value.

The formatting of the default value can be customised using the `default_description_format` option in `setup.py`. By default this is `'Default %s'`.



## API REFERENCE

A Sphinx directive to specify that a module has extra requirements, and show how to install them.

### Functions:

<code>format_default_value(value)</code>	Format the value as a string.
<code>get_arguments(obj)</code>	Returns a dictionary mapping argument names to parameters/arguments for a function.
<code>get_class_defaults(obj)</code>	Obtains the default values for the arguments of a class.
<code>get_function_defaults(obj)</code>	Obtains the default values for the arguments of a function.
<code>process_default_format(app)</code>	Prepare the formatting of the default value.
<code>process_docstring(app, what, name, obj, ...)</code>	Add default values to the docstring.
<code>setup(app)</code>	Setup Sphinx Extension.

### Data:

<code>default_regex</code>	Regular expression to match default values declared in docstrings.
<code>no_default_regex</code>	Regular expression to match flags in docstrings to suppress default values.

### `default_regex`

**Type:** `Pattern`

Regular expression to match default values declared in docstrings.

<b>Pattern</b>	<code>:regex: ^:(default Default) `</code>
<b>Flags</b>	

### `format_default_value (value)`

Format the value as a string.

**Parameters** `value` (`Any`)

New in version 0.3.1.

**Return type** `Optional[str]`

### `get_arguments (obj)`

Returns a dictionary mapping argument names to parameters/arguments for a function.

**Parameters** `obj` (`Callable`) – A function (can be the `__init__` method of a class).

**Return type** `Mapping[str, Parameter]`

**get\_class\_defaults** (*obj*)

Obtains the default values for the arguments of a class.

**Parameters** *obj* (`Type`) – The class.

**Return type** `Iterator[Tuple[str, Any]]`

**Returns** An iterator of 2-element tuples comprising the argument name and its default value.

**get\_function\_defaults** (*obj*)

Obtains the default values for the arguments of a function.

**Parameters** *obj* (`Callable`) – The function.

**Return type** `Iterator[Tuple[str, Any]]`

**Returns** An iterator of 2-element tuples comprising the argument name and its default value.

**no\_default\_regex**

**Type:** `Pattern`

Regular expression to match flags in docstrings to suppress default values.

<b>Pattern</b>	<code>:regex: ^:(No no)[-_](default Default) `</code>
<b>Flags</b>	

**process\_default\_format** (*app*)

Prepare the formatting of the default value.

**Parameters** *app* (`Sphinx`)

**Return type** `None`

**process\_docstring** (*app, what, name, obj, options, lines*)

Add default values to the docstring.

**Parameters**

- **app** (`Sphinx`) – The Sphinx app.
- **what** (`str`)
- **name** (`str`) – The name of the object being documented.
- **obj** (`Any`) – The object being documented.
- **options** (`Dict[str, Any]`) – Mapping of autodoc options to values.
- **lines** (`List[str]`) – List of strings representing the current contents of the docstring.

**Return type** `None`

**setup** (*app*)

Setup Sphinx Extension.

**Parameters** *app* (`Sphinx`)

**Return type** `Dict[str, Any]`

**Returns**

## DEMO

**demo** (*a*, *b*=0.0, *c*="", *d*=' ', *e*='hello world', *f*=(), *g*=Decimal('12.34'), *h*=1234, *i*=None, *j*=None, *k*=None, *l*="", *m*='\t', *n*=...)

### Parameters

- **a** (*Any*) – No default.
- **b** (*float*) – A float. Default 0.0.
- **c** (*str*) – An empty string. Default ''.
- **d** (*str*) – A space (or a smiley face?). Default '␣'.
- **e** (*str*) – A string. Default 'hello\_world'.
- **f** (*Tuple*) – A Tuple. Default ().
- **g** (*Decimal*) – A Decimal. Default Decimal('12.34').
- **h** (*int*) – An int. Default 1234.
- **i** (*Optional[List[str]]*) – Default None. Default None.
- **j** (*Optional[List[str]]*) – Overridden default. Default [].
- **k** (*Optional[List[str]]*) – Suppressed default.
- **l** (*str*) – This is a really long description. It spans multiple lines. The quick brown fox jumps over the lazy dog. The default value should be added at the end regardless. Default ''.
- **m** (*str*) – Tab. Default '\t'.
- **n** (*Any*) – This argument's default value is undefined.

The description for *d* lacked a fullstop at the end, but one was added automatically.

The default value of *n* was Ellipsis, but it wasn't shown.

The above example was created from the following Python code:

```
1 # stdlib
2 from decimal import Decimal # pragma: no cover
3 from typing import Any, List, Optional, Tuple # pragma: no cover
4
5 __all__ = ["demo"] # pragma: no cover
6
7
8 def demo(
9     a: Any,
```

(continues on next page)

(continued from previous page)

```

10     b: float = 0.0,
11     c: str = '',
12     d: str = ' ',
13     e: str = "hello world",
14     f: Tuple = (),
15     g: Decimal = Decimal("12.34"),
16     h: int = 1234,
17     i: Optional[List[str]] = None,
18     j: Optional[List[str]] = None,
19     k: Optional[List[str]] = None,
20     l: str = '',
21     m: str = "\t",
22     n: Any = ...,
23 ): # pragma: no cover
24 """
25
26 :param a: No default.
27 :param b: A float.
28 :param c: An empty string.
29 :param d: A space (or a smiley face?)
30 :param e: A string.
31 :param f: A Tuple.
32 :param g: A Decimal.
33 :param h: An int.
34 :param i: Default None.
35 :param j: Overridden default.
36 :default j: ``[]``
37 :param k: Suppressed default.
38 :no-default k:
39 :param l: This is a really long description.
40     It spans multiple lines.
41     The quick brown fox jumps over the lazy dog.
42     The default value should be added at the end regardless.
43 :param m: Tab.
44 :param n: This argument's default value is undefined.
45
46     The description for ``d`` lacked a fullstop at the end, but one was added,
47     ↪automatically.
48
49     The default value of ``n`` was ``Ellipsis``, but it wasn't shown.
50 """

```

The **PEP 484** type hints were added by `sphinx-autodoc-typehints`.



## OVERVIEW

`default_values` uses `tox` to automate testing and packaging, and `pre-commit` to maintain code quality.

Install `pre-commit` with `pip` and install the git hook:

```
$ python -m pip install pre-commit
$ pre-commit install
```



## CODING STYLE

`Yapf` is used for code formatting, and `isort` is used to sort imports.

`yapf` and `isort` can be run manually via `pre-commit`:

```
$ pre-commit run yapf -a
$ pre-commit run isort -a
```

The complete autoformatting suite can be run with `pre-commit`:

```
$ pre-commit run -a
```



## AUTOMATED TESTS

Tests are run with `tox` and `pytest`. To run tests for a specific Python version, such as Python 3.6, run:

```
$ tox -e py36
```

To run tests for all Python versions, simply run:

```
$ tox
```



## TYPE ANNOTATIONS

Type annotations are checked using `mypy`. Run `mypy` using `tox`:

```
$ tox -e mypy
```





## BUILD DOCUMENTATION LOCALLY

The documentation is powered by Sphinx. A local copy of the documentation can be built with `tox`:

```
$ tox -e docs
```



## DOWNLOADING SOURCE CODE

The `default_values` source code is available on GitHub, and can be accessed from the following URL: [https://github.com/domdfcoding/default\\_values](https://github.com/domdfcoding/default_values)

If you have `git` installed, you can clone the repository with the following command:

```
$ git clone https://github.com/domdfcoding/default_values"
> Cloning into 'default_values'...
> remote: Enumerating objects: 47, done.
> remote: Counting objects: 100% (47/47), done.
> remote: Compressing objects: 100% (41/41), done.
> remote: Total 173 (delta 16), reused 17 (delta 6), pack-reused 126
> Receiving objects: 100% (173/173), 126.56 KiB | 678.00 KiB/s, done.
> Resolving deltas: 100% (66/66), done.
```

Alternatively, the code can be downloaded in a ‘zip’ file by clicking:

*Clone or download* → *Download Zip*

### 10.1 Building from source

The recommended way to build `default_values` is to use `tox`:

```
$ tox -e build
```

The source and wheel distributions will be in the directory `dist`.

If you wish, you may also use `pep517.build` or another **PEP 517**-compatible build tool.

View the [Function Index](#) or browse the [Source Code](#).

[Browse the GitHub Repository](#)

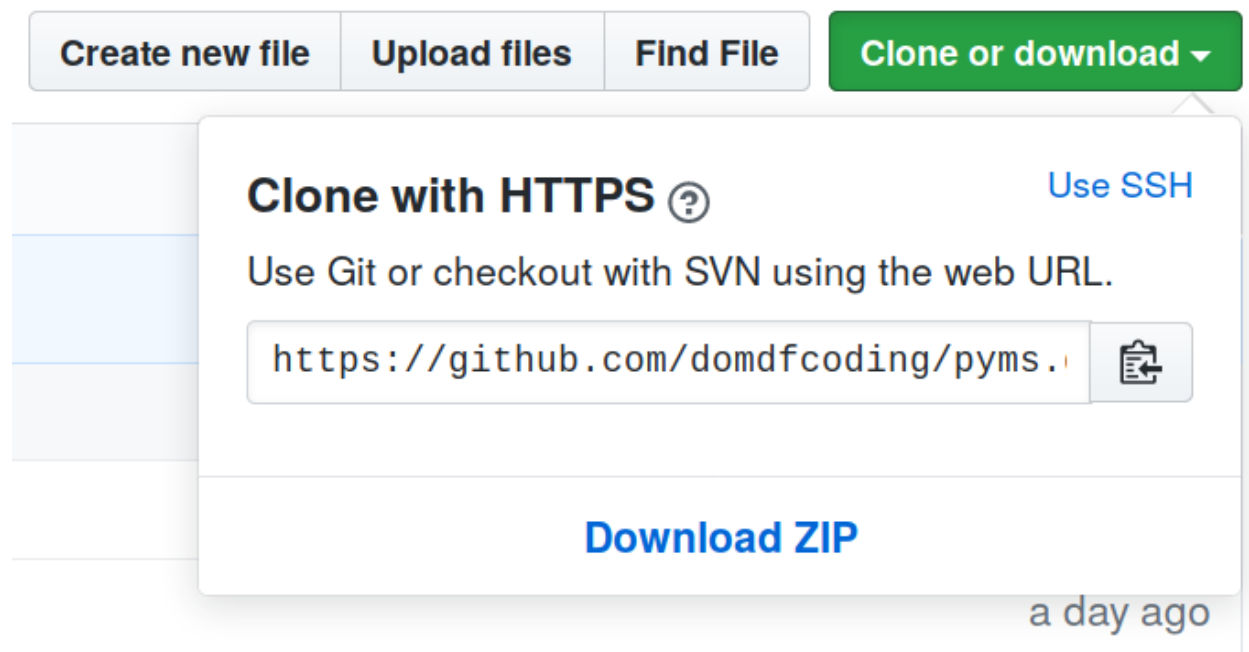


Fig. 1: Downloading a 'zip' file of the source code

## PYTHON MODULE INDEX

### S

`sphinxcontrib.default_values`, [9](#)

`sphinxcontrib.default_values.demo`, [11](#)



## D

`default_regex` (in module *sphinxcontrib.default\_values*), 9  
`demo()` (in module *sphinxcontrib.default\_values.demo*), 11

## F

`format_default_value()` (in module *sphinxcontrib.default\_values*), 9

## G

`get_arguments()` (in module *sphinxcontrib.default\_values*), 9  
`get_class_defaults()` (in module *sphinxcontrib.default\_values*), 10  
`get_function_defaults()` (in module *sphinxcontrib.default\_values*), 10

## M

module  
     *sphinxcontrib.default\_values*, 9  
     *sphinxcontrib.default\_values.demo*, 11

## N

`no_default_regex` (in module *sphinxcontrib.default\_values*), 10

## P

`process_default_format()` (in module *sphinxcontrib.default\_values*), 10  
`process_docstring()` (in module *sphinxcontrib.default\_values*), 10  
 Python Enhancement Proposals  
     PEP 484, 12  
     PEP 517, 23

## S

`setup()` (in module *sphinxcontrib.default\_values*), 10  
*sphinxcontrib.default\_values*  
     module, 9  
*sphinxcontrib.default\_values.demo*  
     module, 11