```
"""Docstring for the example.py module.

Modules names should have short, all-lowercase names.  The module name may
have underscores if this improves readability.

Every module should have a docstring at the very top of the file.  The
module's docstring may extend over multiple lines.  If your docstring does
extend over multiple lines, the closing three quotation marks must be on
a line by itself, preferably preceded by a blank line.

"""
from __future__ import division, absolute_import, print_function

import os  # standard library imports first

# Do NOT import using *, e.g. from numpy import *
#
# Import the module using
#
#   import numpy
#
# instead or import individual functions as needed, e.g
#
#  from numpy import array, zeros
#
# If you prefer the use of abbreviated module names, we suggest the
# convention used by NumPy itself::

import numpy as np
import matplotlib as mpl
import matplotlib.pyplot as plt

# These abbreviated names are not to be used in docstrings; users must
# be able to paste and execute docstrings after importing only the
# numpy module itself, unabbreviated.


def foo(var1, var2, *args, long_var_name='hi', **kwargs):
    r"""Summarize the function in one line.

    Several sentences providing an extended description. Refer to
    variables using back-ticks, e.g. `var`.

    Parameters
    ----------
    var1 : array_like
        Array_like means all those objects -- lists, nested lists, etc. --
        that can be converted to an array.  We can also refer to
        variables like `var1`.
    var2 : int
        The type above can either refer to an actual Python type
        (e.g. ``int``), or describe the type of the variable in more
        detail, e.g. ``(N,) ndarray`` or ``array_like``.
    *args : iterable
        Other arguments.
    long_var_name : {'hi', 'ho'}, optional
        Choices in brackets, default first when optional.
    **kwargs : dict
        Keyword arguments.

    Returns
    -------
    type
        Explanation of anonymous return value of type ``type``.
    describe : type
        Explanation of return value named `describe`.
    out : type
        Explanation of `out`.
    type_without_description

    Other Parameters
    ----------------
    only_seldom_used_keywords : type
        Explanation.
    common_parameters_listed_above : type
        Explanation.

    Raises
    ------
    BadException
        Because you shouldn't have done that.

    See Also
    --------
    numpy.array : Relationship (optional).
    numpy.ndarray : Relationship (optional), which could be fairly long, in
                    which case the line wraps here.
    numpy.dot, numpy.linalg.norm, numpy.eye
```

```
    Notes
    -----
    Notes about the implementation algorithm (if needed).

    This can have multiple paragraphs.

    You may include some math:

    .. math:: X(e^{j\omega } ) = x(n)e^{ - j\omega n}

    And even use a Greek symbol like :math:`\omega` inline.

    References
    ----------
    Cite the relevant literature, e.g. [1]_.  You may also cite these
    references in the notes section above.

    .. [1] O. McNoleg, "The integration of GIS, remote sensing,
       expert systems and adaptive co-kriging for environmental habitat
       modelling of the Highland Haggis using object-oriented, fuzzy-logic
       and neural-network techniques," Computers & Geosciences, vol. 22,
       pp. 585-588, 1996.

    Examples
    --------
    These are written in doctest format, and should illustrate how to
    use the function.

    >>> a = [1, 2, 3]
    >>> print([x + 3 for x in a])
    [4, 5, 6]
    >>> print("a\nb")
    a
    b
    """
    # After closing class docstring, there should be one blank line to
    # separate following codes (according to PEP257).
    # But for function, method and module, there should be no blank lines
    # after closing the docstring.
    pass
```

Rendered

Docstring for the example.py module.

Modules names should have short, all-lowercase names. The module name may have underscores if this improves readability.

Every module should have a docstring at the very top of the file. The module's docstring may extend over multiple lines. If your docstring does extend over multiple lines, the closing three quotation marks must be on a line by itself, preferably preceded by a blank line.

example.foo(*var1*, *var2*, **args*, *long_var_name='hi'*, ***kwargs*) **[source]**

Summarize the function in one line.

Several sentences providing an extended description. Refer to variables using back-ticks, e.g. *var*.

Parameters:

var1 : ***array_like***

Array_like means all those objects – lists, nested lists, etc. – that can be converted to an array. We can also refer to variables like *var1*.

var2 : ***int***

The type above can either refer to an actual Python type (e.g. `int`), or describe the type of the variable in more detail, e.g. `(N,) ndarray` or `array_like`.

args** : ***iterable

Other arguments.

long_var_name : ***{'hi', 'ho'}, optional***

Choices in brackets, default first when optional.

****kwargs** : ***dict***

Keyword arguments.

Returns: *type*

Explanation of anonymous return value of type `type`.

describe : *type*

Explanation of return value named *describe*.

out : *type*

Explanation of *out*.

type_without_description

Other Parameters: **only_seldom_used_keywords : *type***

Explanation.

common_parameters_listed_above : *type*

Explanation.

Raises: **BadException**

Because you shouldn't have done that.

See also

`numpy.array`
Relationship (optional).

`numpy.ndarray`
Relationship (optional), which could be fairly long, in which case the line wraps here.

`numpy.dot`, `numpy.linalg.norm`, `numpy.eye`

Notes

Notes about the implementation algorithm (if needed).

This can have multiple paragraphs.

You may include some math:

$$X(e^{j\omega}) = x(n)e^{-j\omega n}$$

And even use a Greek symbol like ω inline.

References

Cite the relevant literature, e.g. [1]. You may also cite these references in the notes section above.

[1] O. McNoleg, "The integration of GIS, remote sensing, expert systems and adaptive co-kriging for environmental habitat modelling of the Highland Haggis using object-oriented, fuzzy-logic and neural-network techniques," Computers & Geosciences, vol. 22, pp. 585-588, 1996.

Examples

These are written in doctest format, and should illustrate how to use the function.

```
>>> a = [1, 2, 3]
>>> print([x + 3 for x in a])
[4, 5, 6]
>>> print("a\nb")
a
b
```