Documentation in Python

SURP 2022 Python Bootcamp

Ohio State Astronomy

Slides by: James W. Johnson

Why You Should Care

Because you need to know how to read it! Answering your own questions about python is considerably easier if you simply know *where to look*.

Python Enhancement Proposal (PEP) 287 established reStructuredText (RST) as the standard markup language for python documentation

- Whether or not you ever write documentation, you will encounter RST using python

Bonus: What does *args and **kwargs mean?

reStructuredText (RST)

The standard format of python documentation

- Easily ported into HTML and PDF formats, among others
- Relatively easy to read

Online documentation is generally this content ported to HTML, while docstrings *viewed in python* show the raw RST.

Example: numpy.all docstring

```
Test whether all array elements along a given axis evaluate to True.

Parameters
----------
a : array_like
    Input array or object that can be converted to an array.
axis : None or int or tuple of ints, optional
    Axis or axes along which a logical AND reduction is performed.
    The default (`axis` = `None`) is to perform a logical AND over all
    the dimensions of the input array. `axis` may be negative, in
    which case it counts from the last to the first axis.

    .. versionadded:: 1.7.0

    If this is a tuple of ints, a reduction is performed on multiple
    axes, instead of a single axis or all the axes as before.
out : ndarray, optional
    Alternate output array in which to place the result.
    It must have the same shape as the expected output and its
    type is preserved (e.g., if ``dtype(out)`` is float, the result
    will consist of 0.0's and 1.0's).  See `doc.ufuncs` (Section
    "Output arguments") for more details.

keepdims : bool, optional
    If this is set to True, the axes which are reduced are left
    in the result as dimensions with size one. With this option,
    the result will broadcast correctly against the input array.

    If the default value is passed, then `keepdims` will not be
    passed through to the `all` method of sub-classes of
    `ndarray`, however any non-default value will be.  If the
    sub-class' method does not implement `keepdims` any
    exceptions will be raised.

Returns
-------
all : ndarray, bool
    A new boolean or array is returned unless `out` is specified,
    in which case a reference to `out` is returned.
```

reStructuredText (RST)

The standard format of python documentation

- Easily ported into HTML and PDF formats, among others
- Relatively easy to read

Online documentation is generally this content ported to HTML, while docstrings *viewed in python* show the raw RST.

Example: numpy.all docstring

```
See Also
--------
ndarray.all : equivalent method

any : Test whether any element along a given axis evaluates to True.

Notes
-----
Not a Number (NaN), positive infinity and negative infinity
evaluate to `True` because these are not equal to zero.

Examples
--------
>>> np.all([[True,False],[True,True]])
False

>>> np.all([[True,False],[True,True]], axis=0)
array([ True, False])

>>> np.all([-1, 4, 5])
True

>>> np.all([1.0, np.nan])
True

>>> o=np.array(False)
>>> z=np.all([-1, 4, 5], out=o)
>>> id(z), id(o), z
(28293632, 28293632, array(True)) # may vary
```

Example RST

This is the *all* function from NumPy.

Most of the documentation astronomers will see looks very similar to this.

```
Test whether all array elements along a given axis evaluate to True.

Parameters
----------
a : array_like
    Input array or object that can be converted to an array.
axis : None or int or tuple of ints, optional
    Axis or axes along which a logical AND reduction is performed.
    The default (`axis` = `None`) is to perform a logical AND over all
    the dimensions of the input array. `axis` may be negative, in
    which case it counts from the last to the first axis.

    .. versionadded:: 1.7.0

    If this is a tuple of ints, a reduction is performed on multiple
    axes, instead of a single axis or all the axes as before.
out : ndarray, optional
    Alternate output array in which to place the result.
    It must have the same shape as the expected output and its
    type is preserved (e.g., if ``dtype(out)`` is float, the result
    will consist of 0.0's and 1.0's).  See `doc.ufuncs` (Section
    "Output arguments") for more details.

keepdims : bool, optional
    If this is set to True, the axes which are reduced are left
    in the result as dimensions with size one. With this option,
    the result will broadcast correctly against the input array.

    If the default value is passed, then `keepdims` will not be
    passed through to the `all` method of sub-classes of
    `ndarray`, however any non-default value will be.  If the
    sub-class' method does not implement `keepdims` any
    exceptions will be raised.

Returns
-------
all : ndarray, bool
    A new boolean or array is returned unless `out` is specified,
    in which case a reference to `out` is returned.
```

Example RST

1. Short description
2. Parameters (in the order they should be passed) and subsequent keyword arguments. Their data type is noted along with a description of what it represents.
3. What it returns, if anything, and a data type and description for that too

```
Test whether all array elements along a given axis evaluate to True.

Parameters
----------
a : array_like
    Input array or object that can be converted to an array.
axis : None or int or tuple of ints, optional
    Axis or axes along which a logical AND reduction is performed.
    The default (`axis` = `None`) is to perform a logical AND over all
    the dimensions of the input array. `axis` may be negative, in
    which case it counts from the last to the first axis.

    .. versionadded:: 1.7.0

    If this is a tuple of ints, a reduction is performed on multiple
    axes, instead of a single axis or all the axes as before.
out : ndarray, optional
    Alternate output array in which to place the result.
    It must have the same shape as the expected output and its
    type is preserved (e.g., if ``dtype(out)`` is float, the result
    will consist of 0.0's and 1.0's).  See `doc.ufuncs` (Section
    "Output arguments") for more details.

keepdims : bool, optional
    If this is set to True, the axes which are reduced are left
    in the result as dimensions with size one. With this option,
    the result will broadcast correctly against the input array.

    If the default value is passed, then `keepdims` will not be
    passed through to the `all` method of sub-classes of
    `ndarray`, however any non-default value will be.  If the
    sub-class' method does not implement `keepdims` any
    exceptions will be raised.

Returns
-------
all : ndarray, bool
    A new boolean or array is returned unless `out` is specified,
    in which case a reference to `out` is returned.
```

Example RST

4. See Also for other relevant parts of the documentation

5. Notes – any caveats or relevant pieces of information on the function's implementation.

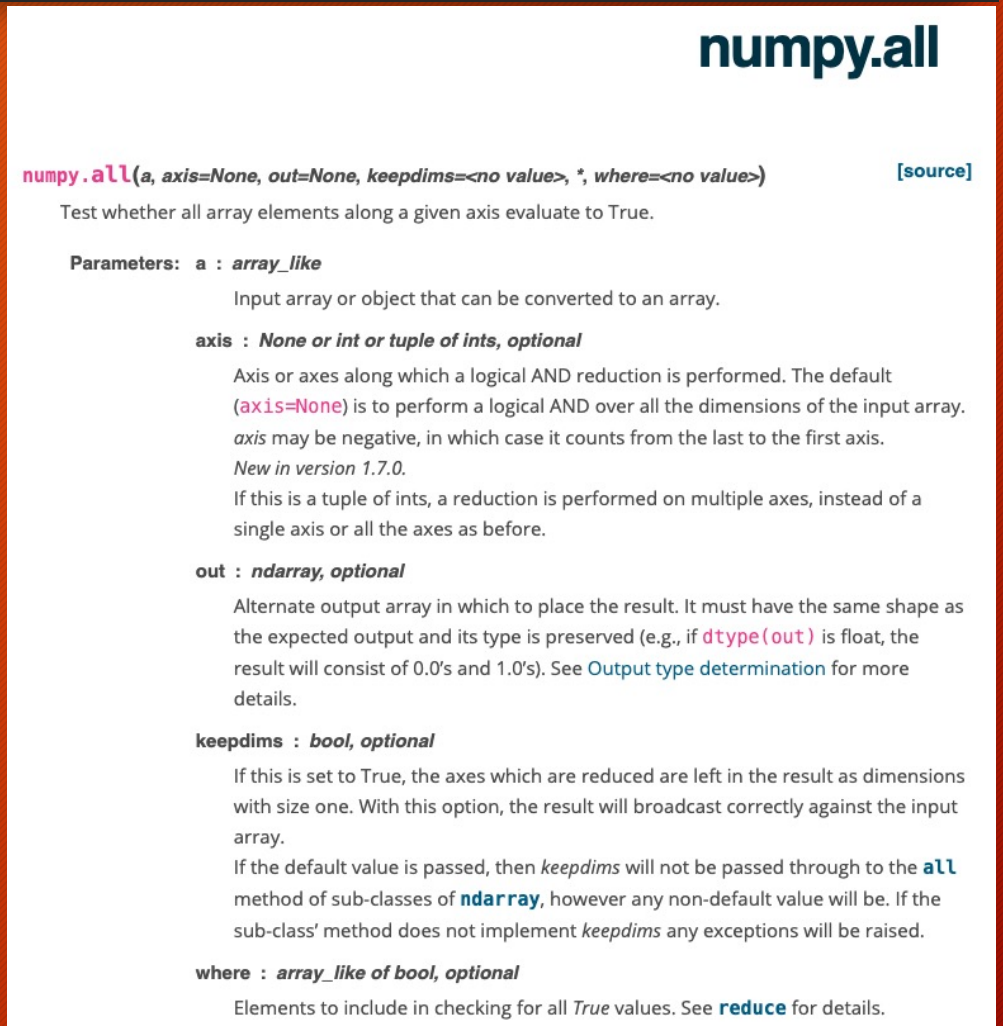
6. Example code

Math is written with *:math:* followed by LaTeX style escape sequences within `` (e.g. *:math:* `\eta`)

```
See Also
--------
ndarray.all : equivalent method

any : Test whether any element along a given axis evaluates to True.

Notes
-----
Not a Number (NaN), positive infinity and negative infinity
evaluate to `True` because these are not equal to zero.

Examples
--------
>>> np.all([[True,False],[True,True]])
False

>>> np.all([[True,False],[True,True]], axis=0)
array([ True, False])

>>> np.all([-1, 4, 5])
True

>>> np.all([1.0, np.nan])
True

>>> o=np.array(False)
>>> z=np.all([-1, 4, 5], out=o)
>>> id(z), id(o), z
(28293632, 28293632, array(True)) # may vary
```

What This Example Looks Like in HTML

Same information as the RST, but formatted nicely.



numpy.all

numpy.all(a, axis=None, out=None, keepdims=<no value>, *, where=<no value>) [source]

Test whether all array elements along a given axis evaluate to True.

Parameters: **a** : *array_like*

Input array or object that can be converted to an array.

axis : *None or int or tuple of ints, optional*

Axis or axes along which a logical AND reduction is performed. The default (axis=None) is to perform a logical AND over all the dimensions of the input array. *axis* may be negative, in which case it counts from the last to the first axis.
New in version 1.7.0.
If this is a tuple of ints, a reduction is performed on multiple axes, instead of a single axis or all the axes as before.

out : *ndarray, optional*

Alternate output array in which to place the result. It must have the same shape as the expected output and its type is preserved (e.g., if dtype(out) is float, the result will consist of 0.0's and 1.0's). See Output type determination for more details.

keepdims : *bool, optional*

If this is set to True, the axes which are reduced are left in the result as dimensions with size one. With this option, the result will broadcast correctly against the input array.
If the default value is passed, then *keepdims* will not be passed through to the **all** method of sub-classes of **ndarray**, however any non-default value will be. If the sub-class' method does not implement *keepdims* any exceptions will be raised.

where : *array_like of bool, optional*

Elements to include in checking for all *True* values. See **reduce** for details.

*args

Indicates that the function accepts an arbitrary number of positional arguments (i.e. non-keyword arguments)

- The data type and description that follows applies to all of them
- *args* in this case is a *tuple*

Example: calculate sum of an unknown number of numbers.

```
def sum(*args):
    r"""
    Calculate the sum of a set of values.

    Parameters
    ----------
    args : real numbers
        The numbers to sum.

    Returns
    -------
    x : float
        The sum of each number passed as an argument.

    Examples
    --------
    >>> sum(1, 2, 3)
    6
    >>> sum(2, 7)
    9
    >>> sum(*(1, 2, 3))
    6
    """
    x = 0
    for i in args:
        x += i
    return x
```

**kwargs

Indicates that the function accepts an arbitrary number of keyword arguments.

Sometimes these are keyword arguments which will be passed to *another function*. This will be made clear in the *Parameters* section of a docstring.

- *kwargs* in this case is a *dict*

Example: print names and ages for an unknown number of people.

```
def ages(**kwargs):
    r"""
    Print the names of people and their ages.

    Parameters
    ----------
    kwargs : integers
        Each person's name as a keyword argument mapped to
        their age in years.

    Examples
    --------
    >>> ages(Sam = 21, Julia = 22)
    Sam is 21 years old.
    Julia is 22 years old.
    >>> ages(Tom = 20)
    Tom is 20 years old.
    >>> ages(**{"Sam": 21, "Julia": 22})
    Sam is 21 years old.
    Julia is 22 years old.
    """
    for i in kwargs.keys():
        print("%s is %d years old." % (i, kwargs[i]))
```

Documentation: Other Aspects

Parameters, returns, raises, etc. for *all* components of a software is often referred to as the “Comprehensive API Reference”

Additional pieces of documentation include, though not limited to:

- Installation instructions
- Tutorials and example code
- Where to submit bug reports
- Developer’s documentation – how to contribute to an open-source project

Markdown

Another markup/type-setting language (RST and HTML are also markup languages)

Can be written directly within jupyter notebooks via *Cell > Cell Type > Markdown*

- Files with extension *.md* (e.g. README.md)

Some useful capabilities:

- Embed hyperlinks to specific locations within your notebook or to external webpages
- Include LaTeX math-type and raw HTML

By using markdown cells in jupyter notebooks and then exporting to a PDF via *File > Download as > PDF via LaTeX (.pdf)*, you can easily make nicely formatted notes for collaborators

Markdown

```
## Fitting the Samples

To fit the samples, we use Markov Chain Monte Carlo methods with the [emcee](https://emcee.readthedocs.io/en/stable/)
python package from [Forman-Mackey et al. (2013)](https://ui.adsabs.harvard.edu/abs/2013PASP..125..306F/abstract).
With 50 walkers starting in the volume defined by $\vec{m} \equiv (\tau_\text{in}, \tau_\star, \eta) = (2, 10, 25) \pm
(0.2, 1, 2.5)$ (i.e. 10% gaussian scatter around the known value), we conduct 50 steps of burn-in on each walker, and
record their next 100 as the $N = 50(100) = 5000$ Markov Chain.

For a given realization of the model with a defined set of parameters $\vec{m}$ and one-zone model predictions
$\vec{\mu} \equiv (\text{[Fe/H], [O/Fe], age})$, the likelihood of the data given the model is equal to the product of
the likelihoods of each individual point given the model. That is:

$$
\begin{aligned}
L(d | m) &\equiv \prod_i L(d_i | m)
\\
\implies \ln L(d | m) &= \ln\left(\prod_i L(d_i | m)\right)
\\
&= \sum_i \ln L(d_i | m)
\end{aligned}
$$
```

Markdown

Fitting the Samples

To fit the samples, we use Markov Chain Monte Carlo methods with the emcee python package from Forman-Mackey et al. (2013). With 50 walkers starting in the volume defined by $\vec{m} \equiv (\tau_{\rm in}, \tau_\star, \eta) = (2, 10, 25) \pm (0.2, 1, 2.5)$ (i.e. 10% gaussian scatter around the known value), we conduct 50 steps of burn-in on each walker, and record their next 100 as the $N = 50(100) = 5000$ Markov Chain.

For a given realization of the model with a defined set of parameters $\vec{m}$ and one-zone model predictions $\vec{\mu} \equiv ([\mathrm{Fe/H}], [\mathrm{O/Fe}], \mathrm{age})$, the likelihood of the data given the model is equal to the product of the likelihoods of each individual point given the model. That is:

$$L(d|m) \equiv \prod_i L(d_i|m)$$

$$\implies \ln L(d|m) = \ln\left(\prod_i L(d_i|m)\right)$$

$$= \sum_i \ln L(d_i|m)$$

Markdown can also include images and tables. Reference: https://www.markdownguide.org/basic-syntax/

Markdown: Stepping-Stone to Web Development

You can build websites in markdown using GitHub Pages

- Having a nice professional website is proving to be increasingly important for getting a foothold in many industries, academia included
- Because you can include raw HTML directly within markdown, it's a stone's throw from CSS and JavaScript, the more "bare bones" tools of web development

We'll talk a little bit about GitHub in the last session!