

# Epydoc

# Epydoc

- ▶ Generates documentation (API) for code
- ▶ Uses docstrings
- ▶ Uses introspection
- ▶ Takes variety of text formats
- ▶ Commandline (epydoc) & GUI (epydocgui)

# Commandline

epydoc [options] NAMES...

**NAMES** The Python modules to document.

- |                     |                                 |
|---------------------|---------------------------------|
| <b>--html</b>       | Generate HTML output (default). |
| <b>--latex</b>      | Generate LaTeX output.          |
| <b>--pdf</b>        | Generate pdf output, via LaTeX. |
| <b>--output DIR</b> | The output directory.           |
| <b>--docformat</b>  | Docstring format                |

# Epytext

- ▶ Assumed format for docstrings
- ▶ Blank lines separate paragraphs
- ▶ Markup with `X{text}`
- ▶ ReST-like lists, headings & block literals

## Epytext example

```
def __init__(self, root):
    """
    Create a new documentation index, based on
    C{ValueDoc}s.  If any I{does not} have a
    canonical name, it will be assigned.

    @param root: A list of C{ValueDoc}s.
    """
    for apidoc in root:
        ...
```

# Epytext markup

- ▶ `I{italics}`, `B{bold}`, `C{code or literal}`, `L{cross reference}`
- ▶ Cross references link internally to other documentation
- ▶ Also URLs and math
- ▶ `@fields`: special keywords / properties used by epydoc

# Epydoc fields

`@param <p>` A description of the parameter p

`@return` The return value for function

`@type <x>` Expected type of parameter x

`@raise <e>` Description how exception e might be raised.

## Epytext example 2

```
def example():  
    """  
    @param x: This is a description of  
        the parameter x to a function.  
    @type x: This is a description of  
        x's type.  
    @return: This is a description of  
        the function's return value.  
  
    It contains two paragraphs.  
    """  
    [...]
```

## Also

- ▶ `epydoc --check <MODULES>` to check completeness
- ▶ Can use plaintext or javadoc instead of epytext
- ▶ Use restructured text instead

# ReST in Epydoc

- ▶ Set on commandline
- ▶ Set in code

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-
"""
Various conversion utility functions.

"""

__docformat__ = 'restructuredtext'
```

## ReSt fields

Use `:Field:` notation for single items or (consolidated) lists.

```
def fox_speed(size, weight, age):  
    """  
    Return the maximum speed for a fox.  
  
    :Parameters:  
        size  
            The size of the fox (in meters)  
        weight : float  
            The weight of the fox (in stones)  
        age : int  
            The age of the fox (in years)  
    """
```

## ReST example

```
def seqToDelimStr (seq, delim, pad_space=True):  
    """  
    Convert sequence into a string of delimited items.
```

Note that this does not attempt to convert or escape the item text in any way.

:Params:

seq

The sequence to be converted.

delim

The string marking the boundary  
between items.

For example::

```
>>> theSeq = ['1', '2', '3']
```

```
>>> seqToDelimStr (theSeq, ',')
```

# Suggested conventions

- ▶ Docstrings in triple quotes
- ▶ One line brief leader then main body
- ▶ Notes, catches and to-dos.
- ▶ Use ReST
- ▶ Embed doctests
  - ▶ Within reason, or use unittest
- ▶ Module docstrings

# Resources

- ▶ Epydoc
- ▶ Other markup
- ▶ ReST field lists