

Sphinx

Python 3.0's documentation system

(works for Python 2.4 and later)

<http://sphinx.pocoo.org/>

Sphinx output

- Customisable HTML and LaTeX
- HTML in multiple files with:
 - Extensive cross-references
 - Hierarchical structure
- Indices and search facility
- Code highlighting with Pygments

(Python 3.0 example on next slide)

Table Of Contents

- Built-in Types
 - Truth Value Testing
 - Boolean Operations — `and`, `or`, `not`
 - Comparisons
 - Numeric Types — `int`, `float`, `complex`
 - Bit-string Operations on Integer Types
 - Iterator Types
 - Sequence Types — `str`, `bytes`, `bytearray`, `list`, `tuple`, `range`
 - String Methods
 - Old String Formatting Operations
 - Range Type
 - Mutable Sequence Types
 - Bytes and Byte Array Methods
 - Set Types — `set`, `frozenset`
 - Mapping Types — `dict`
 - Dictionary view objects
 - File Objects
 - Context Manager Types
 - Other Built-in Types
 - Modules
 - Classes and Class Instances
 - Functions
 - Methods
 - Code Objects
 - Type Objects
 - The Null Object
 - The Ellipsis Object
 - Boolean Values
 - Internal Objects
 - Special Attributes

Previous topic

Built-in Objects

Next topic

Built-in Exceptions

Built-in Types

The following sections describe the standard types that are built into the interpreter.

The principal built-in types are numerics, sequences, mappings, files, classes, instances and exceptions.

Some operations are supported by several object types; in particular, practically all objects can be compared, tested for truth value, and converted to a string (with the `repr()` function or the slightly different `str()` function). The latter function is implicitly used when an object is written by the `print()` function.

Truth Value Testing

Any object can be tested for truth value, for use in an `if` or `while` condition or as operand of the Boolean operations below. The following values are considered false:

- `None`
- `False`
- zero of any numeric type, for example, `0`, `0L`, `0.0`, `0j`.
- any empty sequence, for example, `''`, `()`, `[]`.
- any empty mapping, for example, `{}`.
- instances of user-defined classes, if the class defines a `__bool__()` or `__len__()` method, when that method returns the integer zero or `bool` value `False`. [1]

All other values are considered true — so objects of many types are always true.

Operations and built-in functions that have a Boolean result always return `0` or `False` for false and `1` or `True` for true, unless otherwise stated. (Important exception: the Boolean operations `or` and `and` always return one of their operands.)

Boolean Operations — `and`, `or`, `not`

These are the Boolean operations, ordered by ascending priority:

Operation	Result	Notes
<code>x or y</code>	if <code>x</code> is false, then <code>y</code> , else <code>x</code>	(1)
<code>x and y</code>	if <code>x</code> is false, then <code>x</code> , else <code>y</code>	(2)
<code>not x</code>	if <code>x</code> is false, then <code>True</code> , else <code>False</code>	(3)

Sphinx input

- Text files in reStructuredText format
- Extensions to reStructuredText for “semantic markup”
- Documentation can be in separate files, or can be automatically pulled from docstrings in your code

reStructuredText

- Half way between something like HTML and plain text, e.g.:

Title

=====

Blah, **blah**, ****blah****.

Section

* Bullet

* List

Title

Blah, *blah*, **blah**.

Section

•Bullet

•List

Sphinx semantic markup

```
.. class:: Shark(numberofteeth=Many)
    Creates a shark, has the following
    methods:
    .. method:: bite(victim='passerby')
        Tells the shark who to bite, after a
        biting :attr:`number_of_victims`
        increases by 1.
    and attributes:
    .. attribute:: number_of_victims
        What it sounds like...
```

Avoid using the :meth:`Shark.bite` method.

Using docstrings

- Just use the following Sphinx markup:

```
.. autoclass:: Shark
```

```
.. autofunction:: rescue_mission
```

Etc.

Index

- Classes, functions, etc. automatically generate index entries.
- In addition, you can add your own index entries as follows:

```
.. index:  
   pair: shark; attack  
   single: Jaws
```


[altzone](#) (in module `time`)
[anchor_bgn\(\)](#) (`htmlib.HTMLParser` method)
[anchor_end\(\)](#) (`htmlib.HTMLParser` method)
and
 bitwise
 operator, [\[Link\]](#), [\[Link\]](#)
[and_\(\)](#) (in module `operator`)
[annotate\(\)](#) (in module `dircache`)
annotations
 function

[AUDIODEV](#)
[audioop](#) (module)
augmented
 assignment
[authenticate\(\)](#) (`imaplib.IMAP4` method)
[authenticators\(\)](#) (`netrc.netrc` method)
[autoGIL](#) (module)
[AutoGILError](#)
[avg\(\)](#) (in module `audioop`)
[avgpp\(\)](#) (in module `audioop`)

B

[b16decode\(\)](#) (in module `base64`)
[b16encode\(\)](#) (in module `base64`)
[b2a_base64\(\)](#) (in module `binascii`)
[b2a_hex\(\)](#) (in module `binascii`)
[b2a_hqx\(\)](#) (in module `binascii`)
[b2a_qp\(\)](#) (in module `binascii`)
[b2a_uu\(\)](#) (in module `binascii`)
[b32decode\(\)](#) (in module `base64`)
[b32encode\(\)](#) (in module `base64`)
[b64decode\(\)](#) (in module `base64`)
[b64encode\(\)](#) (in module `base64`)
[Babyl](#) (class in `mailbox`)
[BabylMailbox](#) (class in `mailbox`)
[BabylMessage](#) (class in `mailbox`)
 backslash character
[backslashreplace_errors\(\)](#) (in module `codecs`)
[backward\(\)](#) (in module `turtle`)
[BadStatusLine](#)
[BadZipfile](#)
[Balloon](#) (class in `Tix`)
base64
 encoding
 module
[base64](#) (module)
[BaseCGIHandler](#) (class in `wsgiref.handlers`)
[BaseCookie](#) (class in `Cookie`)
[BaseException](#)
[BaseHandler](#) (class in `urllib2`)
 (class in `wsgiref.handlers`)

[Boolean](#) (class in `aetypes`)
[boolean\(\)](#) (in module `xmlrpclib`)
[border\(\)](#) (`curses.window` method)
[bottom\(\)](#) (`curses.panel.Panel` method)
[bottom_panel\(\)](#) (in module `curses.panel`)
[BoundaryError](#)
[BoundedSemaphore\(\)](#) (in module `threading`)
[box\(\)](#) (`curses.window` method)
break
 statement, [\[Link\]](#), [\[Link\]](#), [\[Link\]](#), [\[Link\]](#)
[break_anywhere\(\)](#) (`bdb.Bdb` method)
[break_here\(\)](#) (`bdb.Bdb` method)
[break_long_words](#) (`textwrap.TextWrapper` attribute)
[BREAK_LOOP](#) (opcode)
[Breakpoint](#) (class in `bdb`)
[BROWSER](#), [\[Link\]](#)
bsddb
 module, [\[Link\]](#), [\[Link\]](#), [\[Link\]](#)
[bsddb](#) (module)
[BsdDbShelf](#) (class in `shelve`)
[btopen\(\)](#) (in module `bsddb`)
buffer
 object, [\[Link\]](#)
[buffer interface](#)
[buffer size](#), I/O
[buffer_info\(\)](#) (`array.array` method)
[buffer_size](#) (`xml.parsers.expat.xmlparser` attribute)
[buffer_text](#) (`xml.parsers.expat.xmlparser` attribute)
[buffer_used](#) (`xml.parsers.expat.xmlparser` attribute)

And in conclusion...

- Sphinx seems a good choice of documentation system as it is:
 - Easy to use and powerful
 - Generates very nice output
 - Likely to be well supported as it is being used for Python 3.0 docs
- See it in action: <http://docs.python.org/dev/3.0/>
- Download it: <http://sphinx.pocoo.org/index.html>