

Introduction à Python

Eric BERTHOMIER

`eric.berthomier@free.fr`

1^{er} janvier 2016





Préférer le beau au laid,
l'explicite à l'implicite,
le simple au complexe,

le complexe au compliqué,
le déroulé à l'imbriqué,
l'aéré au compact.



Préférer le beau au laid,
l'explicite à l'implicite,
le simple au complexe,

le complexe au compliqué,
le déroulé à l'imbriqué,
l'aéré au compact.

La lisibilité compte.



Préférer le beau au laid,
l'explicite à l'implicite,
le simple au complexe,

le complexe au compliqué,
le déroulé à l'imbriqué,
l'aéré au compact.

La lisibilité compte.

<http://legacy.python.org/dev/peps/pep-0020/>



- Les cas particuliers ne le sont jamais assez pour violer les règles, même s'il faut privilégier l'aspect pratique à la pureté.



- Les cas particuliers ne le sont jamais assez pour violer les règles, même s'il faut privilégier l'aspect pratique à la pureté.
- Ne jamais passer les erreurs sous silence, ou les faire taire explicitement.



- Les cas particuliers ne le sont jamais assez pour violer les règles, même s'il faut privilégier l'aspect pratique à la pureté.
- Ne jamais passer les erreurs sous silence, ou les faire taire explicitement.
- Face à l'ambiguïté, ne pas se laisser tenter à deviner.



- Les cas particuliers ne le sont jamais assez pour violer les règles, même s'il faut privilégier l'aspect pratique à la pureté.
- Ne jamais passer les erreurs sous silence, ou les faire taire explicitement.
- Face à l'ambiguïté, ne pas se laisser tenter à deviner.
- Si l'implémentation s'explique difficilement, c'est une mauvaise idée.



- Les cas particuliers ne le sont jamais assez pour violer les règles, même s'il faut privilégier l'aspect pratique à la pureté.
- Ne jamais passer les erreurs sous silence, ou les faire taire explicitement.
- Face à l'ambiguïté, ne pas se laisser tenter à deviner.
- Si l'implémentation s'explique difficilement, c'est une mauvaise idée.
- Si l'implémentation s'explique facilement, c'est peut-être une bonne idée.



- Les cas particuliers ne le sont jamais assez pour violer les règles, même s'il faut privilégier l'aspect pratique à la pureté.
- Ne jamais passer les erreurs sous silence, ou les faire taire explicitement.
- Face à l'ambiguïté, ne pas se laisser tenter à deviner.
- Si l'implémentation s'explique difficilement, c'est une mauvaise idée.
- Si l'implémentation s'explique facilement, c'est peut-être une bonne idée.

<http://legacy.python.org/dev/peps/pep-0020/>



Duck Typing

*When I see a bird that walks like a duck
and swims like a duck
and quacks like a duck,*



Duck Typing

*When I see a bird that walks like a duck
and swims like a duck
and quacks like a duck,
I call that bird a duck.*

https://en.wikipedia.org/wiki/Duck_typing



Documentation en console

```
Python 3.4.1 Shell
File Edit Shell Debug Options Windows Help
Python 3.4.1 (v3.4.1:c0e311e010fc, May 18 2014, 10:38:22) [MSC v.1600 32 bit (Intel)] on win32
Type "copyright", "credits" or "license()" for more information.
>>> help (print)
Help on built-in function print in module builtins:

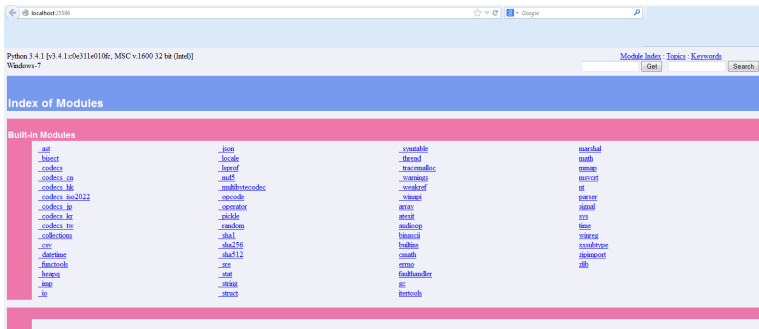
print(...)
    print(value, ..., sep=' ', end='\n', file=sys.stdout, flush=False)

    Prints the values to a stream, or to sys.stdout by default.
    Optional keyword arguments:
    file: a file-like object (stream); defaults to the current sys.stdout.
    sep:  string inserted between values, default a space.
    end:  string appended after the last value, default a newline.
    flush: whether to forcibly flush the stream.

>>>
```



Documentation en local, interface web



Python 3.4.1 [v3.4.1.rc6311e010f; MSC v.1600 32 bit (Intel)]
Windows-7

Module Index - Topics - Keywords

Get Search

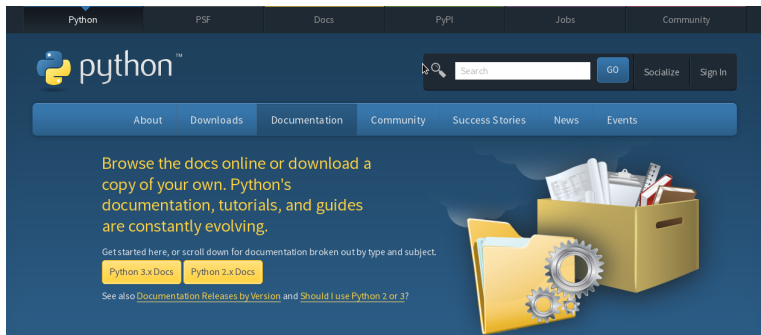
Index of Modules

Built-in Modules

_ast	_json	_syntable	marshal
_bisect	_locale	_thread	math
_codecs	_lxml	_tracemalloc	mmap
_codecs_cn	_md5	_warnings	mover
_codecs_iso2022	_multibytecodec	_weakref	nt
_codecs_jp	_opcode	_winreg	parser
_codecs_kr	_operator	array	signal
_codecs_tw	_pickle	array	sys
_collections	_random	audiopack	time
_cmr	_sha1	binascii	unittest
_datetime	_sha256	binhex	urllib2
_functools	_sha3	cmath	urllib
_heapq	_sre	errno	urllib3
_imp	_stat	faulthandler	uuid
_io	_string	gc	
	_struct	itertools	



Documentation sur la toile



The screenshot shows the Python.org website with a dark blue header. The navigation menu includes links for Python, PSF, Docs, PyPI, Jobs, and Community. The Python logo is on the left, and a search bar with a 'GO' button and 'Socialize' and 'Sign In' links are on the right. Below the header is a secondary navigation bar with links for About, Downloads, Documentation, Community, Success Stories, News, and Events. The main content area features the text: 'Browse the docs online or download a copy of your own. Python's documentation, tutorials, and guides are constantly evolving.' Below this is a link 'Get started here, or scroll down for documentation broken out by type and subject.' and two buttons: 'Python 3.x Docs' and 'Python 2.x Docs'. At the bottom of the main content, there is a link: 'See also [Documentation Releases by Version](#) and [Should I use Python 2 or 3?](#)'. To the right of the text is an illustration of a yellow folder and a box containing papers and gears.

<https://www.python.org/doc/>



- Using the `__future__` module
- The `print` function
- Integer division
- Unicode
- `xrange`
- Raising exceptions

<https://docs.python.org/2/library/2to3.html>

http://nbviewer.ipython.org/github/rasbt/python_reference/blob/master/tutorials/key_differences_between_python_2_and_3.ipynb

- Handling exceptions
- The `next()` function and `.next()` method
- For-loop variables and the global namespace leak
- Comparing unorderable types
- Parsing user inputs via `input()`
- Returning iterable objects instead of lists



Windows

windows.py

```
#!c:\python34\python  
# -*- coding: UTF-8 -*-
```

Linux

linux.py

```
#!/usr/bin/python3  
# -*- coding: UTF-8 -*-
```



EnteteProjet.py

```
__author__ = "Rob Knight, Gavin Huttley, and Peter Maxwell"  
__copyright__ = "Copyright 2007, The Cogent Project"  
__credits__ = ["Rob Knight", "Peter Maxwell", "Gavin Huttley",  
              "Matthew Wakefield"]  
__license__ = "GPL"  
__version__ = "1.0.1"  
__maintainer__ = "Rob Knight"  
__email__ = "rob@spot.colorado.edu"  
__status__ = "Production"
```

