Introduction à Python

Eric BERTHOMIER

eric.berthomier@free.fr

1er janvier 2016

Origine

Philosophie (1/3) - Zen pour Python

Préfèrer 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.

Philosophie (1/3) - Zen pour Python

Préfèrer 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.

Philosophie (1/3) - Zen pour Python

Préfèrer 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/`

Philosophie (2/3) - Zen pour Python

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

Philosophie (2/3) - Zen pour Python

- 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.

Philosophie (2/3) - Zen pour Python

- 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.

Philosophie (2/3) - Zen pour Python

- 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.

Philosophie (2/3) - Zen pour Python

- 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.

Philosophie (2/3) - Zen pour Python

- 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/`

Philosophie (3/3) - Duck Typing

Duck Typing

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

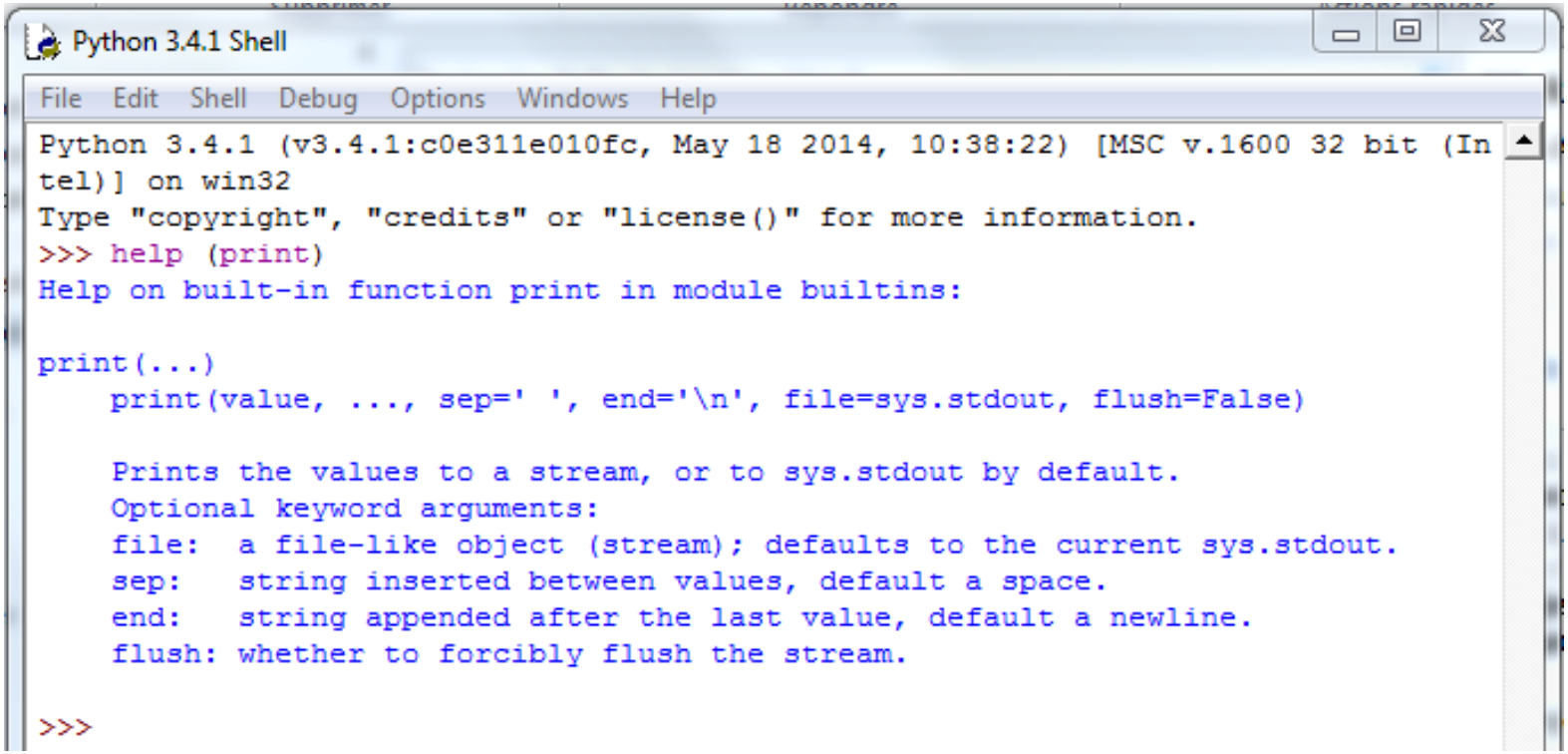
Philosophie (3/3) - Duck Typing

Duck Typing

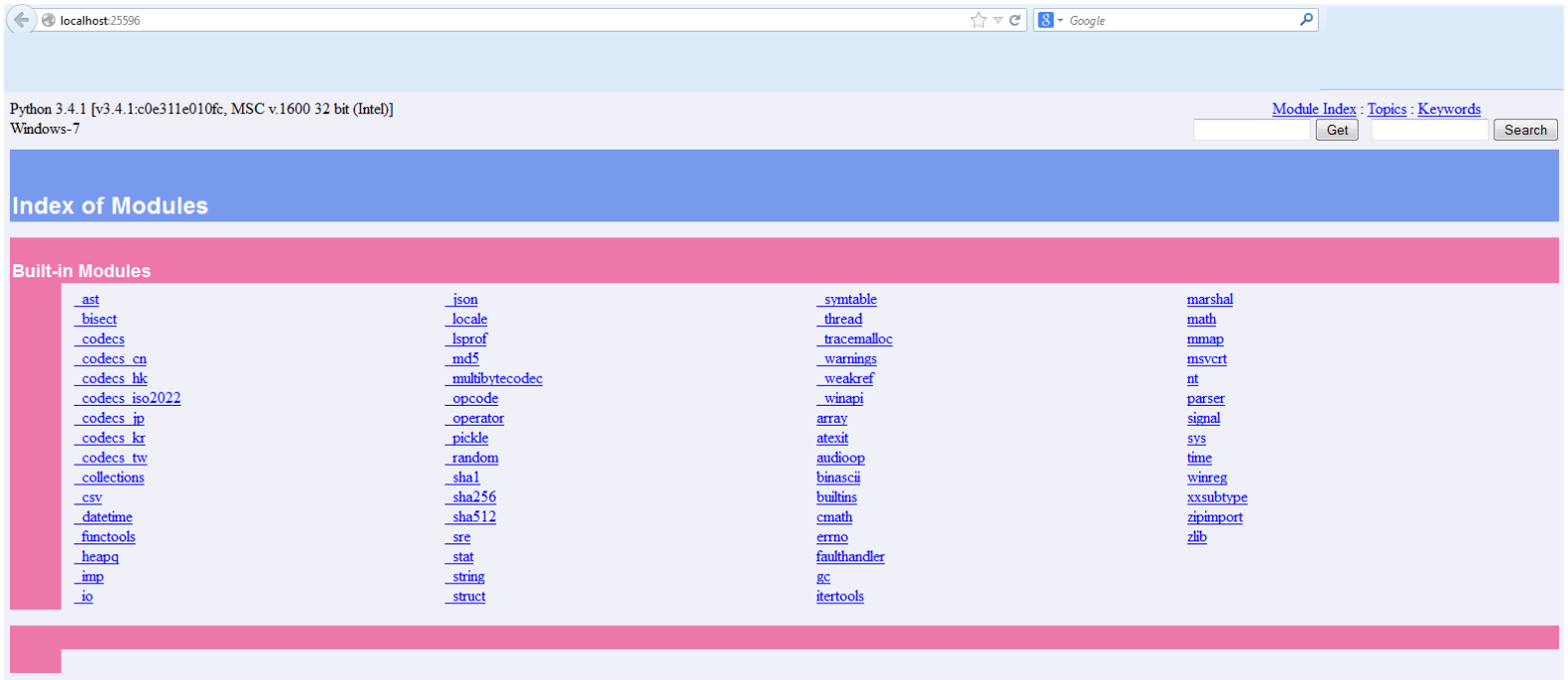
When I see a bird that walks like a duck
and swims like a duck
and quacks like a duck,
I call that bird a duck.

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

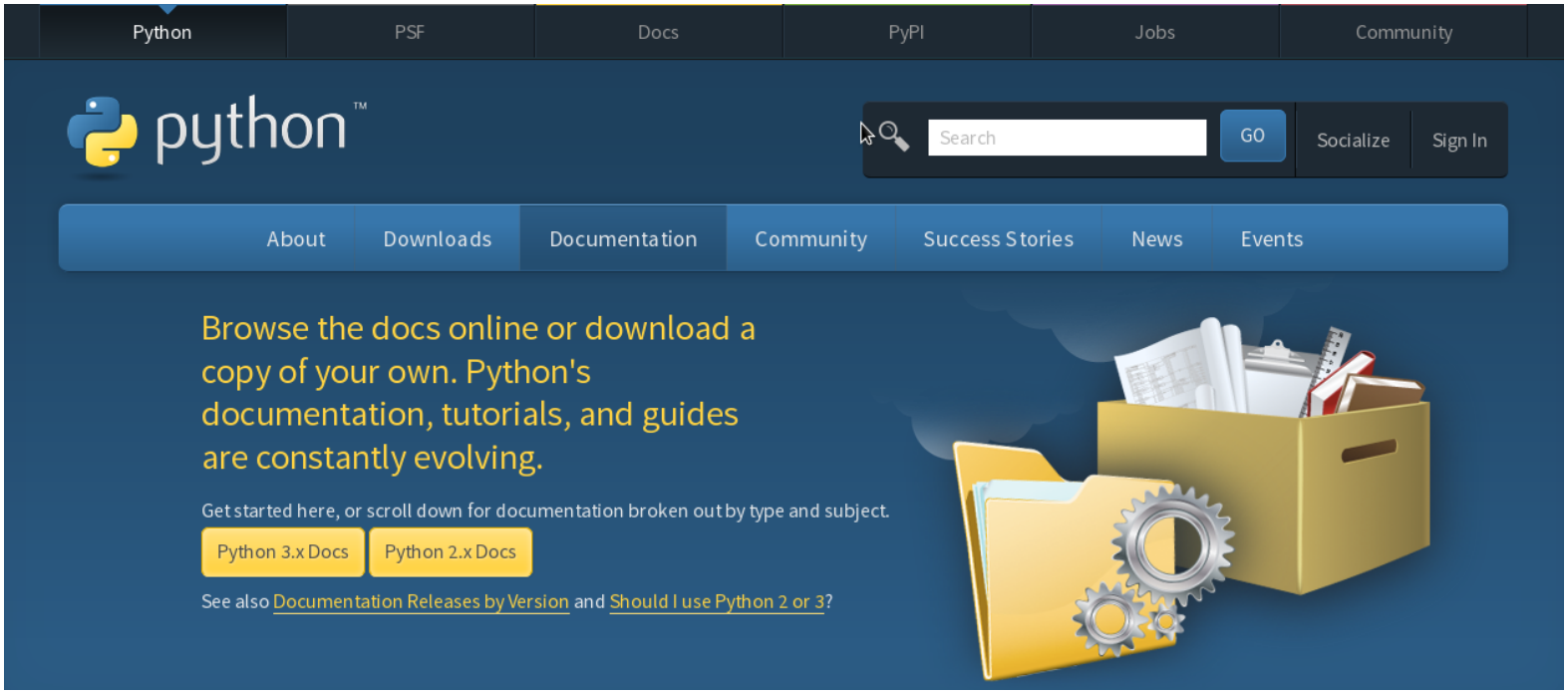
Documentation en console



Documentation en local, interface web



Documentation sur la toile



`https://www.python.org/doc/`

Python : de 2 à 3

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

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

Entête Python : Exécution

Windows

windows.py

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

Linux

linux.py

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

Entête Python : Projet

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"
```