
Python Setup and Usage

Version 2.7.15

**Guido van Rossum
and the Python development team**

janvier 07, 2019

Python Software Foundation
Email : docs@python.org

Table des matières

1	Ligne de commande et environnement	3
1.1	Ligne de commande	3
1.2	Variables d'environnement	8
2	Using Python on Unix platforms	11
2.1	Getting and installing the latest version of Python	11
2.2	Building Python	12
2.3	Python-related paths and files	12
2.4	Miscellaneous	12
2.5	Editors and IDEs	13
3	Utiliser Python sur Windows	15
3.1	Installer Python	15
3.2	Alternative bundles	15
3.3	Configuring Python	16
3.4	Additional modules	18
3.5	Compiler Python sous Windows	18
3.6	Autres ressources	19
4	Utilisation de Python sur un Macintosh	21
4.1	Obtenir et installer MacPython	21
4.2	The IDE	22
4.3	Installing Additional Python Packages	22
4.4	GUI Programming on the Mac	23
4.5	Distributing Python Applications on the Mac	23
4.6	Autres ressources	23
A	Glossaire	25
B	À propos de ces documents	35
B.1	Contributeurs de la documentation Python	35
C	Histoire et licence	37
C.1	Histoire du logiciel	37
C.2	Conditions générales pour accéder à, ou utiliser, Python	38
C.3	Licences et Remerciements pour les logiciels inclus	41

D Copyright	53
Index	55

This part of the documentation is devoted to general information on the setup of the Python environment on different platforms, the invocation of the interpreter and things that make working with Python easier.

Ligne de commande et environnement

L'interpréteur CPython analyse la ligne de commande et l'environnement à la recherche de différents paramètres.

Le format des lignes de commande utilisé par d'autres implémentations peut s'avérer différent. Voir implémentations pour plus d'informations.

1.1 Ligne de commande

Quand vous invoquez Python, vous pouvez spécifier n'importe laquelle de ces options :

```
python [-bBdEiOQsRStuUvVWxX3?] [-c command | -m module-name | script | - ] [args]
```

Le cas d'utilisation le plus courant est, bien entendu, la simple invocation d'un script :

```
python myscript.py
```

1.1.1 Options de l'interface

L'interface de l'interpréteur ressemble à celle du shell UNIX mais fournit quelques méthodes d'invocation supplémentaires :

- Quand l'interpréteur est appelé avec l'entrée standard connectée à un périphérique tty, il lit les lignes de commande et les exécute jusqu'à ce qu'un caractère EOF (caractère fin de fichier, que vous pouvez produire avec **Ctrl-D** sous UNIX ou **Ctrl-Z**, **Enter** sous Windows) soit lu.
- Quand l'interpréteur est appelé avec un argument correspondant à un nom de fichier ou avec un fichier comme entrée standard, il lit et exécute le script contenu dans ce fichier.
- Quand l'interpréteur est appelé avec un argument correspondant à un répertoire, il lit et exécute un script d'un certain nom dans ce répertoire.
- Quand l'interpréteur est appelé avec l'option **-c *commande***, il exécute la ou les instructions Python données comme *commande*. Ici *commande* peut contenir plusieurs instructions séparées par des fins de ligne. Les blancs en début de ligne ne sont pas ignorés dans les instructions Python !
- Quand l'interpréteur est appelé avec l'option **-m *nom-de-module***, le module donné est recherché dans le chemin des modules Python et est exécuté en tant que script.

En mode non-interactif, toute l'entrée est analysée avant d'être exécutée.

Une option d'interface termine la liste des options consommées par l'interpréteur ; tous les arguments atterrissent dans `sys.argv` — notez que le premier élément, à l'indice zéro (`sys.argv[0]`), est une chaîne de caractères indiquant la source du programme.

-c <command>

Exécute le code Python dans *command*. *command* peut être une ou plusieurs instructions, séparées par des fins de ligne, dont les espaces en début de ligne sont significatives, comme dans le code d'un module.

Si cette option est donnée, le premier élément de `sys.argv` est `"-c"` et le répertoire courant est ajouté au début de `sys.path` (permettant aux modules de ce répertoire d'être importés comme des modules de premier niveau).

-m <module-name>

Parcourt `sys.path` à la recherche du module donné et exécute son contenu en tant que module `__main__`.

L'argument étant un nom de *module*, vous ne devez pas fournir d'extension de fichier (`.py`). Le nom-du-module devrait être un nom de module Python valide, mais l'implémentation n'est pas tenue de le vérifier (par exemple, l'utilisation d'un trait d'union peut être autorisée).

Les noms de paquets sont aussi autorisés. Quand un nom de paquet est donné à la place d'un module habituel, l'interpréteur exécutera `<pkg>.__main__` comme module principal. Ce comportement est délibérément identique que pour un répertoire ou un fichier zip donné à l'interpréteur comme argument à exécuter.

Note : cette option ne peut pas être utilisée avec les modules intégrés et les modules d'extension écrits en C, étant donné qu'il ne possèdent pas de fichiers modules en Python. Cependant, elle peut toujours être utilisée pour les modules pré-compilés, même si le fichier source original n'est pas disponible.

Si cette option est donnée, le premier élément de `sys.argv` sera le chemin complet d'accès au fichier du module. Comme pour l'option `-c`, le répertoire courant sera ajouté au début de `sys.path`.

De nombreux modules de la bibliothèque standard contiennent du code qui est invoqué quand ils sont exécutés comme scripts. Un exemple est le module `timeit` :

```
python -mtimeit -s 'setup here' 'benchmarked code here'
python -mtimeit -h # for details
```

Voir aussi :

`runpy.run_module()` Fonctionnalité équivalente directement disponible en code Python

PEP 338 – Exécuter des modules en tant que scripts

Nouveau dans la version 2.4.

Modifié dans la version 2.5 : The named module can now be located inside a package.

Modifié dans la version 2.7 : Supply the package name to run a `__main__` submodule. `sys.argv[0]` is now set to `"-m"` while searching for the module (it was previously incorrectly set to `"-c"`)

-

Lit les commandes depuis l'entrée standard (`sys.stdin`). Si l'entrée standard est un terminal, l'option `-i` est activée implicitement.

Si cette option est donnée, le premier élément de `sys.argv` est `"-"` et le dossier courant est ajouté au début de `sys.path`.

Voir aussi :

`runpy.run_path()` Fonctionnalité équivalente directement disponible en code Python

<script>

Exécute le code Python contenu dans *script*, qui doit être un chemin d'accès (absolu ou relatif) à un fichier, faisant référence à un fichier Python, à un répertoire contenant un fichier `__main__.py` ou à un fichier zip contenant un fichier `__main__.py`.

Si cette option est donnée, le premier élément de `sys.argv` est le nom du script tel que donné sur la ligne de commande.

Si le nom du script se réfère directement à un fichier Python, le répertoire contenant ce fichier est ajouté au début de `sys.path` et le fichier est exécuté en tant que module `__main__`.

Si le nom du script fait référence à un dossier ou à un fichier zip, le nom du script est ajouté au début de `sys.path` et le fichier `__main__.py` à cet endroit est exécuté en tant que module `__main__`.

Modifié dans la version 2.5 : Directories and zipfiles containing a `__main__.py` file at the top level are now considered valid Python scripts.

Si aucune option d'interface est donnée, `-i` est implicite, `sys.argv[0]` est une chaîne vide ("") et le répertoire courant sera ajouté au début de `sys.path`.

Voir aussi :

tut-invoking

1.1.2 Options génériques

`-?`

`-h`

`--help`

Affiche une brève description de toutes les options de la ligne de commande.

Modifié dans la version 2.5 : The `--help` variant.

`-V`

`--version`

Affiche seulement la version de Python. Par exemple :

```
Python 2.5.1
```

Modifié dans la version 2.5 : The `--version` variant.

1.1.3 Options diverses

`-b`

Issue a warning when comparing `unicode` with `bytearray`. Issue an error when the option is given twice (`-bb`).

Note that, unlike the corresponding Python 3.x flag, this will **not** emit warnings for comparisons between `str` and `unicode`. Instead, the `str` instance will be implicitly decoded to `unicode` and Unicode comparison used.

Nouveau dans la version 2.6.

`-B`

Si donné, Python ne tentera pas d'écrire de fichier `.pyc` ou `.pyo` à l'importation des modules source. Voir aussi [PYTHONDONTWRITEBYTECODE](#).

Nouveau dans la version 2.6.

`-d`

Active la sortie de l'analyseur en mode débogage (pour les experts uniquement, en fonction des options de compilation). Voir aussi [PYTHONDEBUG](#).

`-E`

Ignore toutes les variables d'environnement `PYTHON*` qui pourraient être définies. Par exemple, [PYTHONPATH](#) et [PYTHONHOME](#).

Nouveau dans la version 2.2.

-i

Quand un script est passé comme premier argument ou que l'option `-c` est utilisée, entre en mode interactif après avoir exécuté le script ou la commande, même lorsque `sys.stdin` ne semble pas être un terminal. Le fichier `PYTHONSTARTUP` n'est pas lu.

Cela peut être utile pour examiner les variables globales ou une trace de la pile lorsque le script lève une exception. Voir aussi `PYTHONINSPECT`.

-O

Activer les optimisations de base. Cela modifie l'extension du fichier pour les fichiers compilés (*bytecode*) de `.pyc` à `.pyo`. Voir aussi `PYTHONOPTIMIZE`.

-OO

Supprime les *docstrings* en plus des optimisations réalisées par `-O`.

-Q <arg>

Division control. The argument must be one of the following :

old division of int/int and long/long return an int or long (*default*)

new new division semantics, i.e. division of int/int and long/long returns a float

warn old division semantics with a warning for int/int and long/long

warnall old division semantics with a warning for all uses of the division operator

Voir aussi :

`Tools/scripts/fixdiv.py` for a use of `warnall`

PEP 230 – Gestion des alertes

-R

Turn on hash randomization, so that the `__hash__()` values of str, bytes and datetime objects are « salted » with an unpredictable random value. Although they remain constant within an individual Python process, they are not predictable between repeated invocations of Python.

This is intended to provide protection against a denial-of-service caused by carefully-chosen inputs that exploit the worst case performance of a dict construction, $O(n^2)$ complexity. See <http://www.ocert.org/advisories/ocert-2011-003.html> for details.

Changing hash values affects the order in which keys are retrieved from a dict. Although Python has never made guarantees about this ordering (and it typically varies between 32-bit and 64-bit builds), enough real-world code implicitly relies on this non-guaranteed behavior that the randomization is disabled by default.

Voir aussi `PYTHONHASHSEED`.

Nouveau dans la version 2.6.8.

-s

N'ajoute pas le répertoire utilisateur `site-packages` à `sys.path`.

Nouveau dans la version 2.6.

Voir aussi :

PEP 370 – Répertoire `site-packages` propre à l'utilisateur.

-S

Désactiver l'importation du module `site` et les modifications de `sys.path` spécifiques au site qu'elle implique.

-t

Issue a warning when a source file mixes tabs and spaces for indentation in a way that makes it depend on the worth of a tab expressed in spaces. Issue an error when the option is given twice (`-tt`).

-u

Force stdin, stdout and stderr to be totally unbuffered. On systems where it matters, also put stdin, stdout and stderr in binary mode.

Note that there is internal buffering in `file.readlines()` and builtin-file-objects (`for line in sys.stdin`) which is not influenced by this option. To work around this, you will want to use `file.readline()` inside a `while 1: loop`.

Voir aussi [PYTHONUNBUFFERED](#).

-v

Print a message each time a module is initialized, showing the place (filename or built-in module) from which it is loaded. When given twice (`-vv`), print a message for each file that is checked for when searching for a module. Also provides information on module cleanup at exit. See also [PYTHONVERBOSE](#).

-W arg

Warning control. Python's warning machinery by default prints warning messages to `sys.stderr`. A typical warning message has the following form :

```
file:line: category: message
```

Par défaut, chaque avertissement est affiché une seule fois pour chaque ligne de source où il se trouve. Cette option définit à quelle fréquence afficher ces avertissements.

L'option `-W` peut être répétée ; lorsqu'un avertissement correspond à plus d'une option, l'action associée à la dernière correspondance est effectuée. Les options `-W` invalides sont ignorées (cependant, un message d'avertissement est affiché sur les options invalides au moment où le premier avertissement est généré). Starting from Python 2.7, `DeprecationWarning` and its descendants are ignored by default. The `-Wd` option can be used to re-enable them.

Les avertissements peuvent aussi être contrôlés dans le programme Python en utilisant le module `warnings`.

The simplest form of argument is one of the following action strings (or a unique abbreviation) by themselves :

ignore Ignore tous les avertissements.

default Demande explicitement le comportement par défaut (affiche chaque avertissement une fois par ligne de code source).

all Affiche un avertissement à chaque fois qu'il se produit (ce qui peut générer beaucoup de messages si l'avertissement est déclenché à plusieurs reprises, comme à l'intérieur d'une boucle).

module Affiche chaque avertissement uniquement la première fois qu'il apparaît dans chaque module.

once Affiche chaque avertissement uniquement la première fois qu'il apparaît dans le programme.

error Déclenche une exception au lieu d'afficher un message d'avertissement.

La forme complète de l'argument est :

```
action:message:category:module:line
```

Ici, *action* est tel qu'expliqué ci-dessus, mais s'applique uniquement aux messages qui correspondent aux champs restants. Les champs vides correspondent à toutes les valeurs ; les champs vides de fin peuvent être omis. Le champ *message* correspond au début du message d'avertissement affiché, cette expression est insensible à la casse. Le champ *category* correspond à la catégorie d'avertissement. Ce nom doit être un nom complet de classe ; La règle s'applique à tous les messages d'alertes construits avec une classe qui hérite de celle spécifiée. Le nom de la classe complète doit être donnée. Le champ *module* correspond au nom (pleinement qualifié) du module, cette correspondance est sensible à la casse. Le champ *line* correspond au numéro de ligne, où zéro correspond à n'importe quel numéro de ligne et correspond donc à l'option par défaut.

Voir aussi :

`warnings` – le module qui gère les avertissements.

PEP 230 – Gestion des alertes*PYTHONWARNINGS***-x**

Saute la première ligne du code source, autorisant ainsi les directives de type `#!cmd` non conformes au standard Unix. L'objectif est de proposer une astuce spécifique pour le DOS.

-3

Warn about Python 3.x possible incompatibilities by emitting a `DeprecationWarning` for features that are removed or significantly changed in Python 3 and can't be detected using static code analysis.

Nouveau dans la version 2.6.

See `/howto/pyporting` for more details.

1.1.4 Options à ne pas utiliser

-J

Utilisation réservée à `Jython`.

-U

Turns all string literals into unicodes globally. Do not be tempted to use this option as it will probably break your world. It also produces `.pyc` files with a different magic number than normal. Instead, you can enable unicode literals on a per-module basis by using :

```
from __future__ import unicode_literals
```

at the top of the file. See `__future__` for details.

-X

Reserved for alternative implementations of Python to use for their own purposes.

1.2 Variables d'environnement

These environment variables influence Python's behavior, they are processed before the command-line switches other than `-E`. It is customary that command-line switches override environmental variables where there is a conflict.

PYTHONHOME

Modifie l'emplacement des bibliothèques standards de Python. Par défaut, les bibliothèques sont recherchées dans `préfixe/lib/pythonversion` et `préfixe_exec/lib/pythonversion` où `préfixe` et `préfixe_exec` sont des répertoires qui dépendent de l'installation (leur valeur par défaut étant `/usr/local`).

Quand `PYTHONHOME` est défini à un simple répertoire, sa valeur remplace à la fois `préfixe` et `préfixe_exec`. Pour spécifier des valeurs différentes à ces variables, définissez `PYTHONHOME` à `prefix:exec_prefix`.

PYTHONPATH

Augmente le chemin de recherche par défaut des fichiers de modules. Le format est le même que pour `PATH` du shell : un ou plusieurs chemins de répertoires séparés par `os.pathsep` (par exemple, deux points sous Unix et point-virgule sous Windows). Les répertoires qui n'existent pas sont ignorés silencieusement.

En plus des répertoires normaux, des entrées individuelles de `PYTHONPATH` peuvent faire référence à des fichiers zip contenant des modules en pur Python (soit sous forme de code source, soit sous forme compilée). Les modules d'extensions ne peuvent pas être importés à partir de fichiers zip.

Le chemin de recherche par défaut dépend de l'installation mais commence généralement par `préfixe/lib/pythonversion` (voir `PYTHONHOME` ci-dessus). Il est *toujours* ajouté à `PYTHONPATH`.

Comme indiqué ci-dessus dans *Options de l'interface*, un répertoire supplémentaire est inséré dans le chemin de recherche devant `PYTHONPATH`. Le chemin de recherche peut être manipulé depuis un programme Python avec la variable `sys.path`.

PYTHONSTARTUP

If this is the name of a readable file, the Python commands in that file are executed before the first prompt is displayed in interactive mode. The file is executed in the same namespace where interactive commands are executed so that objects defined or imported in it can be used without qualification in the interactive session. You can also change the prompts `sys.ps1` and `sys.ps2` in this file.

PYTHONY2K

Set this to a non-empty string to cause the `time` module to require dates specified as strings to include 4-digit years, otherwise 2-digit years are converted based on rules described in the `time` module documentation.

PYTHONOPTIMIZE

Si elle est définie à une chaîne non vide, c'est équivalent à spécifier l'option `-O`. Si elle est définie à un entier, c'est équivalent à spécifier l'option `-O` plusieurs fois.

PYTHONDEBUG

Si elle est définie à une chaîne non vide, c'est équivalent à spécifier l'option `-d`. Si elle est définie à un entier, c'est équivalent à spécifier l'option `-d` plusieurs fois.

PYTHONINSPECT

Si elle est définie à une chaîne non vide, C'est équivalent à spécifier l'option `-i`.

Cette variable peut aussi être modifiée par du code Python en utilisant `os.environ` pour forcer le mode introspectif à la fin du programme.

PYTHONUNBUFFERED

Si elle est définie à une chaîne non vide, c'est équivalent à spécifier l'option `-u`.

PYTHONVERBOSE

Si elle est définie à une chaîne non vide, c'est équivalent à spécifier l'option `-v`. Si elle est définie à un entier, c'est équivalent à spécifier l'option `-v` plusieurs fois.

PYTHONCASEOK

If this is set, Python ignores case in `import` statements. This only works on Windows, OS X, OS/2, and RiscOS.

PYTHONDONTWRITEBYTECODE

Si donné, Python ne tentera pas d'écrire de fichier `.pyc` ou `.pyo` à l'importation des modules source. Voir aussi *PYTHONDONTWRITEBYTECODE*.

Nouveau dans la version 2.6.

PYTHONHASHSEED

If this variable is set to `random`, the effect is the same as specifying the `-R` option : a random value is used to seed the hashes of str, bytes and datetime objects.

Si *PYTHONHASHSEED* est définie à une valeur entière, elle est utilisée comme valeur de salage pour générer les empreintes des types utilisant la randomisation du hachage.

L'objectif est d'avoir des empreintes reproductibles, pour des tests de l'interpréteur lui-même ou pour qu'un groupe de processus Python puisse partager des empreintes.

The integer must be a decimal number in the range [0,4294967295]. Specifying the value 0 will lead to the same hash values as when hash randomization is disabled.

Nouveau dans la version 2.6.8.

PYTHONIOENCODING

Overrides the encoding used for stdin/stdout/stderr, in the syntax `encodingname:errorhandler`. The `:errorhandler` part is optional and has the same meaning as in `str.encode()`.

Nouveau dans la version 2.6.

PYTHONNOUSERSITE

Si elle est définie, Python n'ajoute pas le répertoire `site-packages` utilisateur à `sys.path`.

Nouveau dans la version 2.6.

Voir aussi :

PEP 370 – Répertoire `site-packages` propre à l'utilisateur.

PYTHONUSERBASE

Définit le répertoire `base` utilisateur. Celui-ci est utilisé pour déterminer le chemin du répertoire utilisateur `site-packages` et Installation alternative : le schéma `user` pour `python setup.py install --user`.

Nouveau dans la version 2.6.

Voir aussi :

PEP 370 – Répertoire `site-packages` propre à l'utilisateur.

PYTHONEXECUTABLE

Si cette variable d'environnement est définie, `sys.argv[0]` est définie à la même valeur au lieu de la valeur fournie par l'exécutable. Ne fonctionne que sur Mac OS X.

PYTHONWARNINGS

C'est équivalent à spécifier l'option `-W`. Si la valeur est une chaîne séparée par des virgules, c'est équivalent à spécifier l'option `-W` plusieurs fois.

PYTHONHTTPSVERIFY

If this environment variable is set specifically to 0, then it is equivalent to implicitly calling `ssl._https_verify_certificates()` with `enable=False` when `ssl` is first imported.

Refer to the documentation of `ssl._https_verify_certificates()` for details.

Nouveau dans la version 2.7.12.

1.2.1 Variables en mode débogage

Définir ces variables n'a d'effet que si Python a été compilé en mode débogage, c'est-à-dire que l'option de compilation `--with-pydebug` a été spécifiée.

PYTHONTHREADDEBUG

Si elle est définie, Python affiche des informations de débogage relatives aux différents fils d'exécution.

Modifié dans la version 2.6 : Previously, this variable was called `THREADDEBUG`.

PYTHONDUMPREFS

Si elle est définie, Python affiche (de manière brute) les objets et les compteurs de références toujours existant après la fermeture de l'interpréteur.

PYTHONMALLOCSTATS

If set, Python will print memory allocation statistics every time a new object arena is created, and on shutdown.

PYTHONSHOWALLOCCOUNT

If set and Python was compiled with `COUNT_ALLOCS` defined, Python will dump allocations counts into `stderr` on shutdown.

Nouveau dans la version 2.7.15.

PYTHONSHOWREFCOUNT

If set, Python will print the total reference count when the program finishes or after each statement in the interactive interpreter.

Nouveau dans la version 2.7.15.

2.1 Getting and installing the latest version of Python

2.1.1 On Linux

Python comes preinstalled on most Linux distributions, and is available as a package on all others. However there are certain features you might want to use that are not available on your distro's package. You can easily compile the latest version of Python from source.

In the event that Python doesn't come preinstalled and isn't in the repositories as well, you can easily make packages for your own distro. Have a look at the following links :

Voir aussi :

<https://www.debian.org/doc/manuals/maint-guide/first.en.html> for Debian users

<https://en.opensuse.org/Portal:Packaging> for OpenSuse users

https://docs.fedoraproject.org/en-US/Fedora_Draft_Documentation/0.1/html/RPM_Guide/ch-creating-packages.html for Fedora users

<http://www.slackbook.org/html/package-management-making-packages.html> for Slackware users

2.1.2 On FreeBSD and OpenBSD

— FreeBSD users, to add the package use :

```
pkg install python3
```

— OpenBSD users, to add the package use :

```
pkg_add -r python
```

```
pkg_add ftp://ftp.openbsd.org/pub/OpenBSD/4.2/packages/<insert your architecture_>
here>/python-<version>.tgz
```

For example i386 users get the 2.5.1 version of Python using :

```
pkg_add ftp://ftp.openbsd.org/pub/OpenBSD/4.2/packages/i386/python-2.5.1p2.tgz
```

2.1.3 On OpenSolaris

You can get Python from [OpenCSW](#). Various versions of Python are available and can be installed with e.g. `pkgutil -i python27`.

2.2 Building Python

If you want to compile CPython yourself, first thing you should do is get the [source](#). You can download either the latest release's source or just grab a fresh [clone](#). (If you want to contribute patches, you will need a clone.)

The build process consists in the usual

```
./configure
make
make install
```

invocations. Configuration options and caveats for specific Unix platforms are extensively documented in the [README](#) file in the root of the Python source tree.

Avertissement : `make install` can overwrite or masquerade the `python` binary. `make altinstall` is therefore recommended instead of `make install` since it only installs `exec_prefix/bin/pythonversion`.

2.3 Python-related paths and files

These are subject to difference depending on local installation conventions; `prefix` (`${prefix}`) and `exec_prefix` (`${exec_prefix}`) are installation-dependent and should be interpreted as for GNU software; they may be the same.

For example, on most Linux systems, the default for both is `/usr`.

File/directory	Signification
<code>exec_prefix/bin/python</code>	Recommended location of the interpreter.
<code>prefix/lib/pythonversion</code> , <code>exec_prefix/lib/pythonversion</code>	Recommended locations of the directories containing the standard modules.
<code>prefix/include/pythonversion</code> , <code>exec_prefix/include/pythonversion</code>	Recommended locations of the directories containing the include files needed for developing Python extensions and embedding the interpreter.
<code>~/.pythonrc.py</code>	User-specific initialization file loaded by the user module; not used by default or by most applications.

2.4 Miscellaneous

To easily use Python scripts on Unix, you need to make them executable, e.g. with

```
$ chmod +x script
```

and put an appropriate Shebang line at the top of the script. A good choice is usually


```
#!/usr/bin/env python
```

which searches for the Python interpreter in the whole `PATH`. However, some Unices may not have the `env` command, so you may need to hardcode `/usr/bin/python` as the interpreter path.

To use shell commands in your Python scripts, look at the `subprocess` module.

2.5 Editors and IDEs

There are a number of IDEs that support Python programming language. Many editors and IDEs provide syntax highlighting, debugging tools, and **PEP 8** checks.

Please go to [Python Editors](#) and [Integrated Development Environments](#) for a comprehensive list.

Utiliser Python sur Windows

This document aims to give an overview of Windows-specific behaviour you should know about when using Python on Microsoft Windows.

3.1 Installer Python

Unlike most Unix systems and services, Windows does not require Python natively and thus does not pre-install a version of Python. However, the CPython team has compiled Windows installers (MSI packages) with every [release](#) for many years.

With ongoing development of Python, some platforms that used to be supported earlier are no longer supported (due to the lack of users or developers). Check [PEP 11](#) for details on all unsupported platforms.

- DOS and Windows 3.x are deprecated since Python 2.0 and code specific to these systems was removed in Python 2.1.
- Up to 2.5, Python was still compatible with Windows 95, 98 and ME (but already raised a deprecation warning on installation). For Python 2.6 (and all following releases), this support was dropped and new releases are just expected to work on the Windows NT family.
- [Windows CE](#) is still supported.
- The [Cygwin](#) installer offers to install the Python interpreter as well (cf. [Cygwin package source](#), [Maintainer releases](#))

See [Python for Windows \(and DOS\)](#) for detailed information about platforms with precompiled installers.

Voir aussi :

[Python on XP](#) « 7 Minutes to « Hello World! » » by Richard Dooling, 2006

[Installing on Windows](#) in « [Dive into Python : Python from novice to pro](#) » by Mark Pilgrim, 2004, ISBN 1-59059-356-1

[For Windows users](#) in « [Installing Python](#) » in « [A Byte of Python](#) » by Swaroop C H, 2003

3.2 Alternative bundles

Besides the standard CPython distribution, there are modified packages including additional functionality. The following is a list of popular versions and their key features :

ActivePython Installer with multi-platform compatibility, documentation, PyWin32

Enthought Python Distribution Popular modules (such as PyWin32) with their respective documentation, tool suite for building extensible Python applications

Notice that these packages are likely to install *older* versions of Python.

3.3 Configuring Python

In order to run Python flawlessly, you might have to change certain environment settings in Windows.

3.3.1 Excursus : Setting environment variables

Windows has a built-in dialog for changing environment variables (following guide applies to XP classical view) : Right-click the icon for your machine (usually located on your Desktop and called « My Computer ») and choose *Properties* there. Then, open the *Advanced* tab and click the *Environment Variables* button.

In short, your path is :

My Computer → *Properties* → *Advanced* → *Environment Variables*

In this dialog, you can add or modify User and System variables. To change System variables, you need non-restricted access to your machine (i.e. Administrator rights).

Another way of adding variables to your environment is using the **set** command :

```
set PYTHONPATH=%PYTHONPATH%;C:\My_python_lib
```

To make this setting permanent, you could add the corresponding command line to your **autoexec.bat**. **msconfig** is a graphical interface to this file.

Viewing environment variables can also be done more straight-forward : The command prompt will expand strings wrapped into percent signs automatically :

```
echo %PATH%
```

Consult **set /?** for details on this behaviour.

Voir aussi :

<https://support.microsoft.com/kb/100843> Environment variables in Windows NT

<https://support.microsoft.com/kb/310519> Comment gérer les variables d'environnement sous Windows XP

<https://www.chem.gla.ac.uk/~louis/software/faq/q1.html> Définir les variables d'environnement, Louis J. Farrugia

3.3.2 Trouver l'exécutable Python

Besides using the automatically created start menu entry for the Python interpreter, you might want to start Python in the DOS prompt. To make this work, you need to set your **%PATH%** environment variable to include the directory of your Python distribution, delimited by a semicolon from other entries. An example variable could look like this (assuming the first two entries are Windows' default) :

```
C:\WINDOWS\system32;C:\WINDOWS;C:\Python25
```

Typing **python** on your command prompt will now fire up the Python interpreter. Thus, you can also execute your scripts with command line options, see *Ligne de commande* documentation.

3.3.3 Finding modules

Python usually stores its library (and thereby your site-packages folder) in the installation directory. So, if you had installed Python to `C:\Python\`, the default library would reside in `C:\Python\Lib\` and third-party modules should be stored in `C:\Python\Lib\site-packages\`.

This is how `sys.path` is populated on Windows :

- An empty entry is added at the start, which corresponds to the current directory.
- If the environment variable `PYTHONPATH` exists, as described in *Variables d'environnement*, its entries are added next. Note that on Windows, paths in this variable must be separated by semicolons, to distinguish them from the colon used in drive identifiers (`C:\` etc.).
- Additional « application paths » can be added in the registry as subkeys of `\SOFTWARE\Python\PythonCore{version}\PythonPath` under both the `HKEY_CURRENT_USER` and `HKEY_LOCAL_MACHINE` hives. Subkeys which have semicolon-delimited path strings as their default value will cause each path to be added to `sys.path`. (Note that all known installers only use `HKLM`, so `HKCU` is typically empty.)
- If the environment variable `PYTHONHOME` is set, it is assumed as « Python Home ». Otherwise, the path of the main Python executable is used to locate a « landmark file » (`Lib\os.py`) to deduce the « Python Home ». If a Python home is found, the relevant sub-directories added to `sys.path` (`Lib`, `plat-win`, etc) are based on that folder. Otherwise, the core Python path is constructed from the `PythonPath` stored in the registry.
- If the Python Home cannot be located, no `PYTHONPATH` is specified in the environment, and no registry entries can be found, a default path with relative entries is used (e.g. `.\Lib;.\plat-win`, etc).

The end result of all this is :

- When running `python.exe`, or any other `.exe` in the main Python directory (either an installed version, or directly from the PCbuild directory), the core path is deduced, and the core paths in the registry are ignored. Other « application paths » in the registry are always read.
- When Python is hosted in another `.exe` (different directory, embedded via COM, etc), the « Python Home » will not be deduced, so the core path from the registry is used. Other « application paths » in the registry are always read.
- If Python can't find its home and there is no registry (eg, frozen `.exe`, some very strange installation setup) you get a path with some default, but relative, paths.

3.3.4 Executing scripts

Python scripts (files with the extension `.py`) will be executed by `python.exe` by default. This executable opens a terminal, which stays open even if the program uses a GUI. If you do not want this to happen, use the extension `.pyw` which will cause the script to be executed by `pythonw.exe` by default (both executables are located in the top-level of your Python installation directory). This suppresses the terminal window on startup.

You can also make all `.py` scripts execute with `pythonw.exe`, setting this through the usual facilities, for example (might require administrative rights) :

1. Launch a command prompt.
2. Associate the correct file group with `.py` scripts :

```
assoc .py=Python.File
```

3. Redirect all Python files to the new executable :

```
ftype Python.File=C:\Path\to\pythonw.exe "%1" %*
```

3.4 Additional modules

Even though Python aims to be portable among all platforms, there are features that are unique to Windows. A couple of modules, both in the standard library and external, and snippets exist to use these features.

The Windows-specific standard modules are documented in `mswin-specific-services`.

3.4.1 PyWin32

The `PyWin32` module by Mark Hammond is a collection of modules for advanced Windows-specific support. This includes utilities for :

- `Component Object Model` (COM)
- Appels à l'API Win32
- Registre
- journal d'événement
- `Microsoft Foundation Classes` (MFC) interfaces utilisateur

`PythonWin` est une exemple d'application MFC livrée avec `PyWin32`. Il s'agit d'un IDE embarqué avec débogueur intégré.

Voir aussi :

`Win32 How Do I... ?` par Tim Golden

`Python and COM` par David et Paul Boddie

3.4.2 Py2exe

`Py2exe` is a `distutils` extension (see `extending-distutils`) which wraps Python scripts into executable Windows programs (`*.exe` files). When you have done this, you can distribute your application without requiring your users to install Python.

3.4.3 WConio

Since Python's advanced terminal handling layer, `curses`, is restricted to Unix-like systems, there is a library exclusive to Windows as well : Windows Console I/O for Python.

`WConio` is a wrapper for Turbo-C's `CONIO.H`, used to create text user interfaces.

3.5 Compiler Python sous Windows

If you want to compile CPython yourself, first thing you should do is get the `source`. You can download either the latest release's source or just grab a fresh `checkout`.

For Microsoft Visual C++, which is the compiler with which official Python releases are built, the source tree contains solutions/project files. View the `readme.txt` in their respective directories :

Répertoire	version de MSVC	Version de Visual Studio
PC/VC6/	6.0	97
PC/VS7.1/	7.1	2003
PC/VS8.0/	8.0	2005
PCbuild/	9.0	2008

Note that not all of these build directories are fully supported. Read the release notes to see which compiler version the official releases for your version are built with.

Vérifiez `PC/readme.txt` pour des informations générales sur le processus de construction.

For extension modules, consult [building-on-windows](#).

Voir aussi :

[Python + Windows + distutils + SWIG + gcc MinGW](#) or « Creating Python extensions in C/C++ with SWIG and compiling them with MinGW gcc under Windows » or « Installing Python extension with distutils and without Microsoft Visual C++ » by Sébastien Sauvage, 2003

[MingW – Python extensions](#) par Trent Apted et al, 2007

3.6 Autres ressources

Voir aussi :

[Python Programming On Win32](#) « Help for Windows Programmers » de Mark Hammond et Andy Robinson, O'Reilly Media, 2000, ISBN 1-56592-621-8

[A Python for Windows Tutorial](#) par Amanda Birmingham, 2004

Utilisation de Python sur un Macintosh

Author Bob Savage <bobsavage@mac.com>

Python sur un Macintosh exécutant Mac OS X est en principe très similaire à Python sur n'importe quelle autre plateforme Unix, mais il y a un certain nombre de fonctionnalités additionnelle telle que l'IDE et le gestionnaire de packages qui méritent d'être soulignées.

The Mac-specific modules are documented in `mac-specific-services`.

Python on Mac OS 9 or earlier can be quite different from Python on Unix or Windows, but is beyond the scope of this manual, as that platform is no longer supported, starting with Python 2.4. See <http://www.cwi.nl/~jack/macpython> for installers for the latest 2.3 release for Mac OS 9 and related documentation.

4.1 Obtenir et installer MacPython

Mac OS X 10.8 comes with Python 2.7 pre-installed by Apple. If you wish, you are invited to install the most recent version of Python from the Python website (<https://www.python.org>). A current « universal binary » build of Python, which runs natively on the Mac's new Intel and legacy PPC CPU's, is available there.

Vous obtiendrez un certain nombre de choses après installation :

- A **MacPython 2.7** folder in your **Applications** folder. In here you find IDLE, the development environment that is a standard part of official Python distributions; PythonLauncher, which handles double-clicking Python scripts from the Finder; and the « Build Applet » tool, which allows you to package Python scripts as standalone applications on your system.
- A framework `/Library/Frameworks/Python.framework`, which includes the Python executable and libraries. The installer adds this location to your shell path. To uninstall MacPython, you can simply remove these three things. A symlink to the Python executable is placed in `/usr/local/bin/`.

The Apple-provided build of Python is installed in `/System/Library/Frameworks/Python.framework` and `/usr/bin/python`, respectively. You should never modify or delete these, as they are Apple-controlled and are used by Apple- or third-party software. Remember that if you choose to install a newer Python version from `python.org`, you will have two different but functional Python installations on your computer, so it will be important that your paths and usages are consistent with what you want to do.

IDLE includes a help menu that allows you to access Python documentation. If you are completely new to Python you should start reading the tutorial introduction in that document.

If you are familiar with Python on other Unix platforms you should read the section on running Python scripts from the Unix shell.

4.1.1 How to run a Python script

Your best way to get started with Python on Mac OS X is through the IDLE integrated development environment, see section *The IDE* and use the Help menu when the IDE is running.

If you want to run Python scripts from the Terminal window command line or from the Finder you first need an editor to create your script. Mac OS X comes with a number of standard Unix command line editors, **vim** and **emacs** among them. If you want a more Mac-like editor, **BBEdit** or **TextWrangler** from Bare Bones Software (see <http://www.barebones.com/products/bbedit/index.html>) are good choices, as is **TextMate** (see <https://macromates.com/>). Other editors include **Gvim** (<http://macvim.org>) and **Aquamacs** (<http://aquamacs.org/>).

To run your script from the Terminal window you must make sure that `/usr/local/bin` is in your shell search path.

To run your script from the Finder you have two options :

- Drag it to **PythonLauncher**
- Select **PythonLauncher** as the default application to open your script (or any .py script) through the finder Info window and double-click it. **PythonLauncher** has various preferences to control how your script is launched. Option-dragging allows you to change these for one invocation, or use its Preferences menu to change things globally.

4.1.2 Running scripts with a GUI

With older versions of Python, there is one Mac OS X quirk that you need to be aware of : programs that talk to the Aqua window manager (in other words, anything that has a GUI) need to be run in a special way. Use **pythonw** instead of **python** to start such scripts.

With Python 2.7, you can use either **python** or **pythonw**.

4.1.3 Configuration

Python on OS X honors all standard Unix environment variables such as `PYTHONPATH`, but setting these variables for programs started from the Finder is non-standard as the Finder does not read your `.profile` or `.cshrc` at startup. You need to create a file `~/MacOSX/environment.plist`. See Apple's Technical Document QA1067 for details.

For more information on installation Python packages in MacPython, see section *Installing Additional Python Packages*.

4.2 The IDE

MacPython ships with the standard IDLE development environment. A good introduction to using IDLE can be found at https://hkn.eecs.berkeley.edu/~dyoo/python/idle_intro/index.html.

4.3 Installing Additional Python Packages

There are several methods to install additional Python packages :

- Packages can be installed via the standard Python distutils mode (`python setup.py install`).

- Many packages can also be installed via the **setuptools** extension or **pip** wrapper, see <https://pip.pypa.io/>.

4.4 GUI Programming on the Mac

There are several options for building GUI applications on the Mac with Python.

PyObjC is a Python binding to Apple's Objective-C/Cocoa framework, which is the foundation of most modern Mac development. Information on PyObjC is available from <https://pythonhosted.org/pyobjc/>.

The standard Python GUI toolkit is **Tkinter**, based on the cross-platform Tk toolkit (<https://www.tcl.tk>). An Aqua-native version of Tk is bundled with OS X by Apple, and the latest version can be downloaded and installed from <https://www.activestate.com>; it can also be built from source.

wxPython is another popular cross-platform GUI toolkit that runs natively on Mac OS X. Packages and documentation are available from <http://www.wxpython.org>.

PyQt is another popular cross-platform GUI toolkit that runs natively on Mac OS X. More information can be found at <https://riverbankcomputing.com/software/pyqt/intro>.

4.5 Distributing Python Applications on the Mac

The « Build Applet » tool that is placed in the MacPython 2.7 folder is fine for packaging small Python scripts on your own machine to run as a standard Mac application. This tool, however, is not robust enough to distribute Python applications to other users.

The standard tool for deploying standalone Python applications on the Mac is **py2app**. More information on installing and using py2app can be found at <http://undefined.org/python/#py2app>.

4.6 Autres ressources

The MacPython mailing list is an excellent support resource for Python users and developers on the Mac : <https://www.python.org/community/sigs/current/pythonmac-sig/>

Another useful resource is the MacPython wiki :

<https://wiki.python.org/moin/MacPython>

>>> L'invite de commande utilisée par défaut dans l'interpréteur interactif. On la voit souvent dans des exemples de code qui peuvent être exécutés interactivement dans l'interpréteur.

... The default Python prompt of the interactive shell when entering code for an indented code block, when within a pair of matching left and right delimiters (parentheses, square brackets, curly braces or triple quotes), or after specifying a decorator.

2to3 Outil qui essaie de convertir du code pour Python 2.x en code pour Python 3.x en gérant la plupart des incompatibilités qui peuvent être détectées en analysant la source et parcourant son arbre syntaxique.

2to3 est disponible dans la bibliothèque standard sous le nom de `lib2to3`; un point d'entrée indépendant est fourni via `Tools/scripts/2to3`. Cf. *2to3-reference*.

classe de base abstraite Les classes de base abstraites (ABC, suivant l'abréviation anglaise *Abstract Base Class*) complètent le *duck-typing* en fournissant un moyen de définir des interfaces pour les cas où d'autres techniques comme `hasattr()` seraient inélégantes, ou subitement fausse (par exemple avec les méthodes magiques). Les ABC introduisent des sous-classes virtuelles, qui n'héritent pas d'une classe mais qui sont quand même reconnues par `isinstance()` ou `issubclass()` (Voir la documentation du module `abc`). Python contient de nombreuses ABC pour les structures de données (dans le module `collections`), les nombres (dans le module `numbers`), les flux (dans le module `io`). Vous pouvez créer vos propres ABC avec le module `abc`.

argument Une valeur, donnée à une *fonction* ou à une *méthode* lors de son appel. Il existe deux types d'arguments :

— *argument nommé* : un argument précédé d'un identifiant (comme `name=`) ou un dictionnaire précédé de `**`, lors d'un appel de fonction. Par exemple, 3 et 5 sont tous les deux des arguments nommés dans l'appel à `complex()` ici :

```
complex(real=3, imag=5)
complex(**{'real': 3, 'imag': 5})
```

— *argument positionnel* : Un argument qui n'est pas nommé. Les arguments positionnels apparaissent au début de la liste des arguments, ou donnés sous forme d'un *itérable* précédé par `*`. Par exemple, 3 et 5 sont tous les deux des arguments positionnels dans les appels suivants :

```
complex(3, 5)
complex(*(3, 5))
```

Les arguments se retrouvent dans le corps de la fonction appelée parmi les variables locales. Voir la section [calls](#) à propos des règles dictant cette affectation. Syntaxiquement, toute expression est acceptée comme argument, et c'est la valeur résultante de l'expression qui sera affectée à la variable locale.

Voir aussi [parameter](#) dans le glossaire, et la question dans la FAQ à propos de la différence entre argument et paramètre.

attribut Valeur associée à un objet et désignée par son nom via une notation utilisant des points. Par exemple, si un objet *o* possède un attribut *a*, il sera référencé par *o.a*.

BDFL Dictateur bienveillant à vie (*Benevolent Dictator For Life* en anglais). Pseudonyme de [Guido van Rossum](#), le créateur de Python.

Objet bytes-compatible Un objet gérant le bufferobjects, comme les classes `str`, `bytearray`, ou `memoryview`. Les objets bytes-compatibles peuvent manipuler des données binaires et ainsi servir à leur compression, sauvegarde, ou envoi sur une socket. Certaines actions nécessitent que la donnée binaire soit modifiable, ce qui n'est pas possible avec tous les objets byte-compatibles.

code intermédiaire (*bytecode*) Le code source, en Python, est compilé en un bytecode, la représentation interne à CPython d'un programme Python. Le bytecode est stocké dans un fichier nommé `.pyc` ou `.pyo`. Ces caches permettent de charger les fichiers plus rapidement lors de la deuxième exécution (en évitant ainsi de recommencer la compilation en bytecode). On dit que ce *langage intermédiaire* est exécuté sur une *machine virtuelle* qui exécute des instructions machine pour chaque instruction du bytecode. Notez que le bytecode n'a pas vocation à fonctionner entre différentes machines virtuelle Python, encore moins entre différentes version de Python.

La documentation du module `dis` fournit une liste des instructions du code intermédiaire.

classe Modèle pour créer des objets définis par l'utilisateur. Une définition de classe (*class*) contient normalement des définitions de méthodes qui agissent sur les instances de la classe.

classic class Any class which does not inherit from `object`. See [new-style class](#). Classic classes have been removed in Python 3.

coercition The implicit conversion of an instance of one type to another during an operation which involves two arguments of the same type. For example, `int(3.15)` converts the floating point number to the integer 3, but in `3+4.5`, each argument is of a different type (one int, one float), and both must be converted to the same type before they can be added or it will raise a `TypeError`. Coercion between two operands can be performed with the `coerce` built-in function; thus, `3+4.5` is equivalent to calling `operator.add(*coerce(3, 4.5))` and results in `operator.add(3.0, 4.5)`. Without coercion, all arguments of even compatible types would have to be normalized to the same value by the programmer, e.g., `float(3)+4.5` rather than just `3+4.5`.

nombre complexe Extension des nombres réels familiers, dans laquelle tous les nombres sont exprimés sous la forme d'une somme d'une partie réelle et d'une partie imaginaire. Les nombres imaginaires sont les nombres réels multipliés par l'unité imaginaire (la racine carrée de -1 , souvent écrite *i* en mathématiques ou *j* par les ingénieurs). Python comprend nativement les nombres complexes, écrits avec cette dernière notation : la partie imaginaire est écrite avec un suffixe *j*, exemple, `3+1j`. Pour utiliser les équivalents complexes de `math`, utilisez `cmath`. Les nombres complexes sont un concept assez avancé en mathématiques. Si vous ne connaissez pas ce concept, vous pouvez tranquillement les ignorer.

gestionnaire de contexte Objet contrôlant l'environnement à l'intérieur d'un bloc `with` en définissant les méthodes `__enter__()` et `__exit__()`. Consultez la [PEP 343](#).

CPython L'implémentation canonique du langage de programmation Python, tel que distribué sur [python.org](#). Le terme « CPython » est utilisé dans certains contextes lorsqu'il est nécessaire de distinguer cette implémentation des autres comme *Jython* ou *IronPython*.

décorateur Fonction dont la valeur de retour est une autre fonction. Un décorateur est habituellement utilisé pour transformer une fonction via la syntaxe `@wrapper`, dont les exemples typiques sont : `classmethod()` et `staticmethod()`.

La syntaxe des décorateurs est simplement du sucre syntaxique, les définitions des deux fonctions suivantes sont sémantiquement équivalentes :

```
def f(...):
    ...
f = staticmethod(f)

@staticmethod
def f(...):
    ...
```

Quoique moins fréquemment utilisé, le même concept existe pour les classes. Consultez la documentation définitions de fonctions et définitions de classes pour en savoir plus sur les décorateurs.

descripteur Any *new-style* object which defines the methods `__get__()`, `__set__()`, or `__delete__()`. When a class attribute is a descriptor, its special binding behavior is triggered upon attribute lookup. Normally, using `a.b` to get, set or delete an attribute looks up the object named `b` in the class dictionary for `a`, but if `b` is a descriptor, the respective descriptor method gets called. Understanding descriptors is a key to a deep understanding of Python because they are the basis for many features including functions, methods, properties, class methods, static methods, and reference to super classes.

Pour plus d'informations sur les méthodes des descripteurs, consultez `descriptors`.

dictionnaire An associative array, where arbitrary keys are mapped to values. The keys can be any object with `__hash__()` and `__eq__()` methods. Called a hash in Perl.

vue de dictionnaire The objects returned from `dict.viewkeys()`, `dict.viewvalues()`, and `dict.viewitems()` are called dictionary views. They provide a dynamic view on the dictionary's entries, which means that when the dictionary changes, the view reflects these changes. To force the dictionary view to become a full list use `list(dictview)`. See `dict-views`.

docstring Première chaîne littérale qui apparaît dans l'expression d'une classe, fonction, ou module. Bien qu'ignorée à l'exécution, elle est reconnue par le compilateur et placée dans l'attribut `__doc__` de la classe, de la fonction ou du module. Comme cette chaîne est disponible par introspection, c'est l'endroit idéal pour documenter l'objet.

duck-typing Style de programmation qui ne prend pas en compte le type d'un objet pour déterminer s'il respecte une interface, mais qui appelle simplement la méthode ou l'attribut (*Si ça a un bec et que ça cancanne, ça doit être un canard*, *duck* signifie canard en anglais). En se concentrant sur les interfaces plutôt que les types, du code bien construit améliore sa flexibilité en autorisant des substitutions polymorphiques. Le *duck-typing* évite de vérifier les types via `type()` ou `isinstance()`. Notez cependant que le *duck-typing* peut travailler de pair avec les *classes de base abstraites*. À la place, le *duck-typing* utilise plutôt `hasattr()` ou la programmation *EAFP*.

EAFP Il est plus simple de demander pardon que demander la permission (*Easier to Ask for Forgiveness than Permission* en anglais). Ce style de développement Python fait l'hypothèse que le code est valide et traite les exceptions si cette hypothèse s'avère fausse. Ce style, propre et efficace, est caractérisé par la présence de beaucoup de mots clés `try` et `except`. Cette technique de programmation contraste avec le style *LYBL* utilisé couramment dans les langages tels que C.

expression A piece of syntax which can be evaluated to some value. In other words, an expression is an accumulation of expression elements like literals, names, attribute access, operators or function calls which all return a value. In contrast to many other languages, not all language constructs are expressions. There are also *statements* which cannot be used as expressions, such as `print` or `if`. Assignments are also statements, not expressions.

module d'extension Module écrit en C ou C++, utilisant l'API C de Python pour interagir avec Python et le code de l'utilisateur.

objet fichier Objet exposant une ressource via une API orientée fichier (avec les méthodes `read()` ou `write()`). En fonction de la manière dont il a été créé, un objet fichier peut interfacer l'accès à un fichier sur le disque ou à un autre type de stockage ou de communication (typiquement l'entrée standard, la sortie standard, un tampon en mémoire, une socket réseau, ...). Les objets fichiers sont aussi appelés *file-like-objects* ou *streams*.

There are actually three categories of file objects : raw binary files, buffered binary files and text files. Their interfaces are defined in the `io` module. The canonical way to create a file object is by using the `open()` function.

objet fichier-compatible Synonyme de *objet fichier*.

chercheur An object that tries to find the *loader* for a module. It must implement a method named `find_module()`. See [PEP 302](#) for details.

division entière Division mathématique arrondissant à l'entier inférieur. L'opérateur de la division entière est `//`. Par exemple l'expression `11 // 4` vaut 2, contrairement à `11 / 4` qui vaut 2.75. Notez que `(-11) // 4` vaut -3 car l'arrondi se fait à l'entier inférieur. Voir la [PEP 328](#).

fonction Suite d'instructions qui renvoie une valeur à son appelant. On peut lui passer des *arguments* qui pourront être utilisés dans le corps de la fonction. Voir aussi *paramètre*, *méthode* et *fonction*.

__future__ A pseudo-module which programmers can use to enable new language features which are not compatible with the current interpreter. For example, the expression `11/4` currently evaluates to 2. If the module in which it is executed had enabled *true division* by executing :

```
from __future__ import division
```

the expression `11/4` would evaluate to 2.75. By importing the `__future__` module and evaluating its variables, you can see when a new feature was first added to the language and when it will become the default :

```
>>> import __future__
>>> __future__.division
_Feature((2, 2, 0, 'alpha', 2), (3, 0, 0, 'alpha', 0), 8192)
```

ramasse-miettes (*garbage collection*) Le mécanisme permettant de libérer de la mémoire lorsqu'elle n'est plus utilisée. Python utilise un ramasse-miettes par comptage de référence, et un ramasse-miettes cyclique capable de détecter et casser les références circulaires.

générateur A function which returns an iterator. It looks like a normal function except that it contains `yield` statements for producing a series of values usable in a `for`-loop or that can be retrieved one at a time with the `next()` function. Each `yield` temporarily suspends processing, remembering the location execution state (including local variables and pending try-statements). When the generator resumes, it picks up where it left off (in contrast to functions which start fresh on every invocation).

expression génératrice Expression qui donne un itérateur. Elle ressemble à une expression normale, suivie d'une expression `for` définissant une variable de boucle, un intervalle et une expression `if` optionnelle. Toute cette expression génère des valeurs pour la fonction qui l'entoure :

```
>>> sum(i*i for i in range(10))      # sum of squares 0, 1, 4, ... 81
285
```

GIL Voir *global interpreter lock*.

verrou global de l'interpréteur (*global interpreter lock* en anglais) Mécanisme utilisé par l'interpréteur *CPython* pour s'assurer qu'un seul fil d'exécution (*thread* en anglais) n'exécute le *bytecode* à la fois. Cela simplifie l'implémentation de *CPython* en rendant le modèle objet (incluant des parties critiques comme la classe native `dict`) implicitement protégé contre les accès concourants. Verrouiller l'interpréteur entier rend plus facile l'implémentation de multiples fils d'exécution (*multi-thread* en anglais), au détriment malheureusement de beaucoup du parallélisme possible sur les machines ayant plusieurs processeurs.

Cependant, certains modules d'extension, standards ou non, sont conçus de manière à libérer le GIL lorsqu'ils effectuent des tâches lourdes tel que la compression ou le hachage. De la même manière, le GIL est toujours libéré lors des entrées / sorties.

Les tentatives précédentes d'implémenter un interpréteur Python avec une granularité de verrouillage plus fine ont toutes échouées, à cause de leurs mauvaises performances dans le cas d'un processeur unique. Il est admis que corriger ce problème de performance induit mènerait à une implémentation beaucoup plus compliquée et donc plus coûteuse à maintenir.

hachable An object is *hashable* if it has a hash value which never changes during its lifetime (it needs a `__hash__()` method), and can be compared to other objects (it needs an `__eq__()` or `__cmp__()` method). Hashable objects which compare equal must have the same hash value.

La hachabilité permet à un objet d'être utilisé comme clé de dictionnaire ou en tant que membre d'un ensemble (type *set*), car ces structures de données utilisent ce *hash*.

Tous les types immuables fournis par Python sont hachables, et aucun type mutable (comme les listes ou les dictionnaires) ne l'est. Toutes les instances de classes définies par les utilisateurs sont hachables par défaut, elles sont toutes différentes selon `__eq__`, sauf comparées à elles mêmes, et leur empreinte (*hash*) est calculée à partir de leur `id()`.

IDLE Environnement de développement intégré pour Python. IDLE est un éditeur basique et un interpréteur livré avec la distribution standard de Python.

immuable Objet dont la valeur ne change pas. Les nombres, les chaînes et les n-uplets sont immuables. Ils ne peuvent être modifiés. Un nouvel objet doit être créé si une valeur différente doit être stockée. Ils jouent un rôle important quand une valeur de *hash* constante est requise, typiquement en clé de dictionnaire.

integer division Mathematical division discarding any remainder. For example, the expression `11/4` currently evaluates to 2 in contrast to the 2.75 returned by float division. Also called *floor division*. When dividing two integers the outcome will always be another integer (having the floor function applied to it). However, if one of the operands is another numeric type (such as a `float`), the result will be coerced (see *coercion*) to a common type. For example, an integer divided by a float will result in a float value, possibly with a decimal fraction. Integer division can be forced by using the `//` operator instead of the `/` operator. See also [future](#).

importer Processus rendant le code Python d'un module disponible dans un autre.

importateur Objet qui trouve et charge un module, en même temps un *chercheur* et un *chargeur*.

interactif Python a un interpréteur interactif, ce qui signifie que vous pouvez écrire des expressions et des instructions à l'invite de l'interpréteur. L'interpréteur Python va les exécuter immédiatement et vous en présenter le résultat. Démarrez juste `python` (probablement depuis le menu principal de votre ordinateur). C'est un moyen puissant pour tester de nouvelles idées ou étudier de nouveaux modules (souvenez-vous de `help(x)`).

interprété Python est un langage interprété, en opposition aux langages compilés, bien que la frontière soit floue en raison de la présence d'un compilateur en code intermédiaire. Cela signifie que les fichiers sources peuvent être exécutés directement, sans avoir à compiler un fichier exécutable intermédiaire. Les langages interprétés ont généralement un cycle de développement / débogage plus court que les langages compilés. Cependant, ils s'exécutent généralement plus lentement. Voir aussi *interactif*.

itérable An object capable of returning its members one at a time. Examples of iterables include all sequence types (such as `list`, `str`, and `tuple`) and some non-sequence types like `dict` and `file` and objects of any classes you define with an `__iter__()` or `__getitem__()` method. Iterables can be used in a `for` loop and in many other places where a sequence is needed (`zip()`, `map()`, ...). When an iterable object is passed as an argument to the built-in function `iter()`, it returns an iterator for the object. This iterator is good for one pass over the set of values. When using iterables, it is usually not necessary to call `iter()` or deal with iterator objects yourself. The `for` statement does that automatically for you, creating a temporary unnamed variable to hold the iterator for the duration of the loop. See also *iterator*, *sequence*, and *generator*.

itérateur An object representing a stream of data. Repeated calls to the iterator's `next()` method return successive items in the stream. When no more data are available a `StopIteration` exception

is raised instead. At this point, the iterator object is exhausted and any further calls to its `next()` method just raise `StopIteration` again. Iterators are required to have an `__iter__()` method that returns the iterator object itself so every iterator is also iterable and may be used in most places where other iterables are accepted. One notable exception is code which attempts multiple iteration passes. A container object (such as a `list`) produces a fresh new iterator each time you pass it to the `iter()` function or use it in a `for` loop. Attempting this with an iterator will just return the same exhausted iterator object used in the previous iteration pass, making it appear like an empty container.

Vous trouverez davantage d'informations dans `typeiter`.

fonction clé Une fonction clé est un objet callable qui renvoie une valeur à fins de tri ou de classement. Par exemple, la fonction locale `strxfrm()` est utilisée pour générer une clé de classement prenant en compte les conventions de classement spécifiques aux paramètres régionaux courants.

Plusieurs outils dans Python acceptent des fonctions clé pour maîtriser comment les éléments sont triés ou groupés. Typiquement les fonctions `min()`, `max()`, `sorted()`, `list.sort()`, `heapq.nsmallest()`, `heapq.nlargest()`, et `itertools.groupby()`.

La méthode `str.lower()` peut servir en fonction clé pour effectuer des recherches insensibles à la casse. Aussi, il est possible de créer des fonctions clé au besoin avec des expressions `lambda`, comme `lambda r: (r[0], r[2])`. Finalement le module `operator` fournit des constructeurs de fonctions clé : `attrgetter()`, `itemgetter()`, et `methodcaller()`. Voir Comment Trier pour avoir des exemples de création et d'utilisation de fonctions clés.

argument nommé Voir *argument*.

lambda An anonymous inline function consisting of a single *expression* which is evaluated when the function is called. The syntax to create a lambda function is `lambda [parameters]: expression`

LBYL Regarde avant de tomber, (*Look before you leap* en anglais). Ce style de programmation consiste à vérifier des conditions avant d'effectuer des appels ou des accès. Ce style contraste avec le style *EAFP* et se caractérise par la présence de beaucoup d'instructions `if`.

Dans un environnement avec plusieurs fils d'exécution (*multi-threaded* en anglais), le style *LBYL* peut engendrer un séquençage critique (*race condition* en anglais) entre le « regarde » et le « tomber ». Par exemple, le code `if key in mapping: return mapping[key]` peut échouer si un autre fil d'exécution supprime la clé `key` du `mapping` après le test mais avant l'accès. Ce problème peut être résolu avec des verrous (*locks*) ou avec l'approche *EAFP*.

list A built-in Python *sequence*. Despite its name it is more akin to an array in other languages than to a linked list since access to elements is $O(1)$.

liste en compréhension (ou liste en intension) A compact way to process all or part of the elements in a sequence and return a list with the results. `result = ["0x%02x" % x for x in range(256) if x % 2 == 0]` generates a list of strings containing even hex numbers (0x..) in the range from 0 to 255. The `if` clause is optional. If omitted, all elements in `range(256)` are processed.

chargeur An object that loads a module. It must define a method named `load_module()`. A loader is typically returned by a *finder*. See [PEP 302](#) for details.

Tableau de correspondances Un conteneur permettant d'accéder à des éléments par clé et implémente les méthodes spécifiées dans `Mapping` ou `~collections.MutableMapping` :ref:`classes de base abstraites`. Les classes suivantes sont des exemples de mapping : `dict`, `collections.defaultdict`, `collections.OrderedDict`, et `collections.Counter`.

métaclasses Classe d'une classe. Les définitions de classe créent un nom pour la classe, un dictionnaire de classe et une liste de classes parentes. La métaclasses a pour rôle de réunir ces trois paramètres pour construire la classe. La plupart des langages orientés objet fournissent une implémentation par défaut. La particularité de Python est la possibilité de créer des métaclasses personnalisées. La plupart des utilisateurs n'aura jamais besoin de cet outil, mais lorsque le besoin survient, les métaclasses offrent des solutions élégantes et puissantes. Elles sont utilisées pour journaliser les accès à des propriétés, rendre sûr les environnements *multi-threads*, suivre la création d'objets, implémenter des singletons et bien d'autres tâches.

Plus d'informations sont disponibles dans : *metaclasses*.

méthode Fonction définie à l'intérieur d'une classe. Lorsqu'elle est appelée comme un attribut d'une instance de cette classe, la méthode reçoit l'instance en premier *argument* (qui, par convention, est habituellement nommé `self`). Voir *function* et *nested scope*.

ordre de résolution des méthodes L'ordre de résolution des méthodes (*MRO* pour *Method Resolution Order* en anglais) est, lors de la recherche d'un attribut dans les classes parentes, la façon dont l'interpréteur Python classe ces classes parentes. Voir [The Python 2.3 Method Resolution Order](#) pour plus de détails sur l'algorithme utilisé par l'interpréteur Python depuis la version 2.3.

module Objet utilisé pour organiser une portion unitaire de code en Python. Les modules ont un espace de noms et peuvent contenir n'importe quels objets Python. Charger des modules est appelé *importer*. Voir aussi *paquet*.

MRO Voir *ordre de résolution des méthodes*.

muable Un objet muable peut changer de valeur tout en gardant le même `id()`. Voir aussi *immuable*.

n-uplet nommé (*named-tuple* en anglais) Classe qui, comme un *n-uplet* (*tuple* en anglais), a ses éléments accessibles par leur indice. Et en plus, les éléments sont accessibles par leur nom. Par exemple, `time.localtime()` donne un objet ressemblant à un *n-uplet*, dont `year` est accessible par son indice : `t[0]` ou par son nom : `t.tm_year`).

Un *n-uplet nommé* peut être un type natif tel que `time.struct_time` ou il peut être construit comme une simple classe. Un *n-uplet nommé* complet peut aussi être créé via la fonction `collections.namedtuple()`. Cette dernière approche fournit automatiquement des fonctionnalités supplémentaires, tel qu'une représentation lisible comme `Employee(name='jones', title='programmer')`.

espace de noms The place where a variable is stored. Namespaces are implemented as dictionaries. There are the local, global and built-in namespaces as well as nested namespaces in objects (in methods). Namespaces support modularity by preventing naming conflicts. For instance, the functions `__builtin__.open()` and `os.open()` are distinguished by their namespaces. Namespaces also aid readability and maintainability by making it clear which module implements a function. For instance, writing `random.seed()` or `itertools.izip()` makes it clear that those functions are implemented by the `random` and `itertools` modules, respectively.

portée imbriquée The ability to refer to a variable in an enclosing definition. For instance, a function defined inside another function can refer to variables in the outer function. Note that nested scopes work only for reference and not for assignment which will always write to the innermost scope. In contrast, local variables both read and write in the innermost scope. Likewise, global variables read and write to the global namespace.

nouvelle classe Any class which inherits from `object`. This includes all built-in types like `list` and `dict`. Only new-style classes can use Python's newer, versatile features like `__slots__`, descriptors, properties, and `__getattr__()`. More information can be found in newstyle.

objet N'importe quelle donnée comportant des états (sous forme d'attributs ou d'une valeur) et un comportement (des méthodes). C'est aussi (`object`) l'ancêtre commun à absolument toutes les *nouvelles classes*.

paquet *module* Python qui peut contenir des sous-modules ou des sous-paquets. Techniquement, un paquet est un module qui possède un attribut `__path__`.

paramètre A named entity in a *function* (or method) definition that specifies an *argument* (or in some cases, arguments) that the function can accept. There are four types of parameters :

- *positional-or-keyword* : l'argument peut être passé soit par sa *position*, soit en tant que *argument nommé*. C'est le type de paramètre par défaut. Par exemple, `foo` et `bar` dans l'exemple suivant :

```
def func(foo, bar=None): ...
```

- *positional-only* : l'argument ne peut être donné que par sa position. Python n'a pas de syntaxe pour déclarer de tels paramètres, cependant des fonctions natives, comme `abs()`, en utilisent.

- *var-positional* : une séquence d'arguments positionnels peut être fournie (en plus de tous les arguments positionnels déjà acceptés par d'autres paramètres). Un tel paramètre peut être défini en préfixant son nom par une *. Par exemple *args* ci-après :

```
def func(*args, **kwargs): ...
```

- *var-keyword* : une quantité arbitraire d'arguments peut être passée, chacun étant nommé (en plus de tous les arguments nommés déjà acceptés par d'autres paramètres). Un tel paramètre est défini en préfixant le nom du paramètre par **. Par exemple, *kwargs* ci-dessus.

Les paramètres peuvent spécifier des arguments obligatoires ou optionnels, ainsi que des valeurs par défaut pour les arguments optionnels.

See also the [argument](#) glossary entry, the FAQ question on the difference between arguments and parameters, and the function section.

PEP Python Enhancement Proposal. A PEP is a design document providing information to the Python community, or describing a new feature for Python or its processes or environment. PEPs should provide a concise technical specification and a rationale for proposed features.

PEPs are intended to be the primary mechanisms for proposing major new features, for collecting community input on an issue, and for documenting the design decisions that have gone into Python. The PEP author is responsible for building consensus within the community and documenting dissenting opinions.

See [PEP 1](#).

argument positionnel Voir [argument](#).

Python 3000 Surnom donné à la série des Python 3.x (très vieux surnom donné à l'époque où Python 3 représentait un futur lointain). Aussi abrégé *Py3k*.

Pythonique Idée, ou bout de code, qui colle aux idiomes de Python plutôt qu'aux concepts communs rencontrés dans d'autres langages. Par exemple, il est idiomatique en Python de parcourir les éléments d'un itérable en utilisant `for`. Beaucoup d'autres langages n'ont pas cette possibilité, donc les gens qui ne sont pas habitués à Python utilisent parfois un compteur numérique à la place :

```
for i in range(len(food)):
    print food[i]
```

Plutôt qu'utiliser la méthode, plus propre et élégante, donc *Pythonique* :

```
for piece in food:
    print piece
```

nombre de références Nombre de références à un objet. Lorsque le nombre de références à un objet descend à zéro, l'objet est désalloué. Le comptage de référence n'est généralement pas visible dans le code Python, mais c'est un élément clé de l'implémentation *CPython*. Le module `sys` définit une fonction `getrefcount()` que les développeurs peuvent utiliser pour obtenir le nombre de références à un objet donné.

__slots__ A declaration inside a *new-style class* that saves memory by pre-declaring space for instance attributes and eliminating instance dictionaries. Though popular, the technique is somewhat tricky to get right and is best reserved for rare cases where there are large numbers of instances in a memory-critical application.

séquence An *iterable* which supports efficient element access using integer indices via the `__getitem__()` special method and defines a `len()` method that returns the length of the sequence. Some built-in sequence types are `list`, `str`, `tuple`, and `unicode`. Note that `dict` also supports `__getitem__()` and `__len__()`, but is considered a mapping rather than a sequence because the lookups use arbitrary *immutable* keys rather than integers.

tranche An object usually containing a portion of a *sequence*. A slice is created using the subscript notation, `[]` with colons between numbers when several are given, such as in `variable_name[1:3:5]`.

The bracket (subscript) notation uses `slice` objects internally (or in older versions, `__getslice__()` and `__setslice__()`).

méthode spéciale (*special method* en anglais) Méthode appelée implicitement par Python pour exécuter une opération sur un type, comme une addition. De telles méthodes ont des noms commençant et terminant par des doubles tirets bas. Les méthodes spéciales sont documentées dans `specialnames`.

instruction Une instruction (*statement* en anglais) est un composant d'un « bloc » de code. Une instruction est soit une *expression*, soit une ou plusieurs constructions basées sur un mot-clé, comme `if`, `while` ou `for`.

struct sequence A tuple with named elements. Struct sequences expose an interface similar to *named tuple* in that elements can be accessed either by index or as an attribute. However, they do not have any of the named tuple methods like `_make()` or `_asdict()`. Examples of struct sequences include `sys.float_info` and the return value of `os.stat()`.

chaîne entre triple guillemets Chaîne qui est délimitée par trois guillemets simples (') ou trois guillemets doubles ("). Bien qu'elle ne fournisse aucune fonctionnalité qui ne soit pas disponible avec une chaîne entre guillemets, elle est utile pour de nombreuses raisons. Elle vous autorise à insérer des guillemets simples et doubles dans une chaîne sans avoir à les protéger et elle peut s'étendre sur plusieurs lignes sans avoir à terminer chaque ligne par un `\`. Elle est ainsi particulièrement utile pour les chaînes de documentation (*docstrings*).

type Le type d'un objet Python détermine quel genre d'objet c'est. Tous les objets ont un type. Le type d'un objet peut être obtenu via son attribut `__class__` ou via `type(obj)`.

retours à la ligne universels A manner of interpreting text streams in which all of the following are recognized as ending a line : the Unix end-of-line convention `'\n'`, the Windows convention `'\r\n'`, and the old Macintosh convention `'\r'`. See [PEP 278](#) and [PEP 3116](#), as well as `str.splitlines()` for an additional use.

environnement virtuel Environnement d'exécution isolé (en mode coopératif) qui permet aux utilisateurs de Python et aux applications d'installer et de mettre à jour des paquets sans interférer avec d'autres applications Python fonctionnant sur le même système.

machine virtuelle Ordinateur défini entièrement par du logiciel. La machine virtuelle (*virtual machine*) de Python exécute le *bytecode* produit par le compilateur de *bytecode*.

Le zen de Python Liste de principes et de préceptes utiles pour comprendre et utiliser le langage. Cette liste peut être obtenue en tapant « `import this` » dans une invite Python interactive.

À propos de ces documents

Ces documents sont générés à partir de sources en [reStructuredText](#) par [Sphinx](#), un analyseur de documents spécialement conçu pour la documentation Python.

Le développement de la documentation et de ses outils est entièrement basé sur le volontariat, tout comme Python. Si vous voulez contribuer, allez voir la page [reporting-bugs](#) qui contient des informations pour vous y aider. Les nouveaux volontaires sont toujours les bienvenus !

Merci beaucoup à :

- Fred L. Drake, Jr., créateur des outils originaux de la documentation Python et rédacteur de la plupart de son contenu ;
- le projet [Docutils](#) pour avoir créé *reStructuredText* et la suite d'outils *Docutils* ;
- Fredrik Lundh pour son projet [Alternative Python Reference](#), dont Sphinx a pris beaucoup de bonnes idées.

B.1 Contributeurs de la documentation Python

De nombreuses personnes ont contribué au langage Python, à sa bibliothèque standard et à sa documentation. Consultez [Misc/ACKS](#) dans les sources de la distribution Python pour avoir une liste partielle des contributeurs.

Ce n'est que grâce aux suggestions et contributions de la communauté Python que Python a une documentation si merveilleuse – Merci !

Histoire et licence

C.1 Histoire du logiciel

Python a été créé au début des années 1990 par Guido van Rossum, au Stichting Mathematisch Centrum (CWI, voir <https://www.cwi.nl/>) au Pays-Bas en tant que successeur d'un langage appelé ABC. Guido est l'auteur principal de Python, bien qu'il inclut de nombreuses contributions de la part d'autres personnes.

En 1995, Guido continua son travail sur Python au Corporation for National Research Initiatives (CNRI, voir <https://www.cnri.reston.va.us/>) de Reston, en Virginie, d'où il diffusa plusieurs versions du logiciel.

En mai 2000, Guido et l'équipe de développement centrale de Python sont parti vers BeOpen.com pour former l'équipe BeOpen PythonLabs. En octobre de la même année, l'équipe de PythonLabs est partie vers Digital Creations (désormais Zope Corporation ; voir <http://www.zope.com/>). En 2001, la Python Software Foundation (PSF, voir <http://www.python.org/psf/>) voit le jour. Il s'agit d'une organisation à but non lucratif détenant les droits de propriété intellectuelle de Python. Zope Corporation en est un sponsor.

Toutes les versions de Python sont Open Source (voir <https://www.opensource.org/> pour la définition d'Open Source). Historiquement, la plupart, mais pas toutes, des versions de Python ont également été compatible avec la GPL, le tableau ci-dessous résume les différentes versions.

Version	Dérivé de	Année	Propriétaire	Compatible avec la GPL ?
0.9.0 à 1.2	n/a	1991-1995	CWI	oui
1.3 à 1.5.2	1.2	1995-1999	CNRI	oui
1.6	1.5.2	2000	CNRI	non
2.0	1.6	2000	BeOpen.com	non
1.6.1	1.6	2001	CNRI	non
2.1	2.0+1.6.1	2001	PSF	non
2.0.1	2.0+1.6.1	2001	PSF	oui
2.1.1	2.1+2.0.1	2001	PSF	oui
2.1.2	2.1.1	2002	PSF	oui
2.1.3	2.1.2	2002	PSF	oui
2.2 et supérieur	2.1.1	2001-maintenant	PSF	oui

Note : Compatible GPL ne signifie pas que nous distribuons Python sous licence GPL. Toutes les licences Python, excepté la licence GPL, vous permettent la distribution d'une version modifiée sans rendre open source ces changements. La licence « compatible GPL » rend possible la diffusion de Python avec un autre logiciel qui est lui, diffusé sous la licence GPL ; les licences « non compatible GPL » ne le peuvent pas.

Merci aux nombreux bénévoles qui ont travaillé sous la direction de Guido pour rendre ces versions possibles.

C.2 Conditions générales pour accéder à, ou utiliser, Python

C.2.1 PSF LICENSE AGREEMENT FOR PYTHON 2.7.15

1. This LICENSE AGREEMENT is between the Python Software Foundation ("PSF"), and the Individual or Organization ("Licensee") accessing and otherwise using Python 2.7.15 software in source or binary form and its associated documentation.
2. Subject to the terms and conditions of this License Agreement, PSF hereby grants Licensee a nonexclusive, royalty-free, world-wide license to reproduce, analyze, test, perform and/or display publicly, prepare derivative works, distribute, and otherwise use Python 2.7.15 alone or in any derivative version, provided, however, that PSF's License Agreement and PSF's notice of copyright, i.e., "Copyright © 2001-2019 Python Software Foundation; All Rights Reserved" are retained in Python 2.7.15 alone or in any derivative version prepared by Licensee.
3. In the event Licensee prepares a derivative work that is based on or incorporates Python 2.7.15 or any part thereof, and wants to make the derivative work available to others as provided herein, then Licensee hereby agrees to include in any such work a brief summary of the changes made to Python 2.7.15.
4. PSF is making Python 2.7.15 available to Licensee on an "AS IS" basis. PSF MAKES NO REPRESENTATIONS OR WARRANTIES, EXPRESS OR IMPLIED. BY WAY OF EXAMPLE, BUT NOT LIMITATION, PSF MAKES NO AND DISCLAIMS ANY REPRESENTATION OR WARRANTY OF MERCHANTABILITY OR FITNESS FOR ANY PARTICULAR PURPOSE OR THAT THE USE OF PYTHON 2.7.15 WILL NOT INFRINGE ANY THIRD PARTY RIGHTS.
5. PSF SHALL NOT BE LIABLE TO LICENSEE OR ANY OTHER USERS OF PYTHON 2.7.15 FOR ANY INCIDENTAL, SPECIAL, OR CONSEQUENTIAL DAMAGES OR LOSS AS A RESULT OF MODIFYING, DISTRIBUTING, OR OTHERWISE USING PYTHON 2.7.15, OR ANY DERIVATIVE THEREOF, EVEN IF ADVISED OF THE POSSIBILITY THEREOF.
6. This License Agreement will automatically terminate upon a material breach of its terms and conditions.
7. Nothing in this License Agreement shall be deemed to create any relationship of agency, partnership, or joint venture between PSF and Licensee. This License Agreement does not grant permission to use PSF trademarks or trade name in a trademark sense to endorse or promote products or services of Licensee, or any third party.
8. By copying, installing or otherwise using Python 2.7.15, Licensee agrees to be bound by the terms and conditions of this License Agreement.

C.2.2 LICENCE D'UTILISATION BEOPEN.COM POUR PYTHON 2.0

LICENCE D'UTILISATION LIBRE BEOPEN PYTHON VERSION 1

1. This LICENSE AGREEMENT is between BeOpen.com ("BeOpen"), having an office at 160 Saratoga Avenue, Santa Clara, CA 95051, and the Individual or Organization ("Licensee") accessing and otherwise using this software in source or binary form and its associated documentation ("the Software").
2. Subject to the terms and conditions of this BeOpen Python License Agreement, BeOpen hereby grants Licensee a non-exclusive, royalty-free, world-wide license to reproduce, analyze, test, perform and/or display publicly, prepare derivative works, distribute, and otherwise use the Software alone or in any derivative version, provided, however, that the BeOpen Python License is retained in the Software, alone or in any derivative version prepared by Licensee.
3. BeOpen is making the Software available to Licensee on an "AS IS" basis. BEOPEN MAKES NO REPRESENTATIONS OR WARRANTIES, EXPRESS OR IMPLIED. BY WAY OF EXAMPLE, BUT NOT LIMITATION, BEOPEN MAKES NO AND DISCLAIMS ANY REPRESENTATION OR WARRANTY OF MERCHANTABILITY OR FITNESS FOR ANY PARTICULAR PURPOSE OR THAT THE USE OF THE SOFTWARE WILL NOT INFRINGE ANY THIRD PARTY RIGHTS.
4. BEOPEN SHALL NOT BE LIABLE TO LICENSEE OR ANY OTHER USERS OF THE SOFTWARE FOR ANY INCIDENTAL, SPECIAL, OR CONSEQUENTIAL DAMAGES OR LOSS AS A RESULT OF USING, MODIFYING OR DISTRIBUTING THE SOFTWARE, OR ANY DERIVATIVE THEREOF, EVEN IF ADVISED OF THE POSSIBILITY THEREOF.
5. This License Agreement will automatically terminate upon a material breach of its terms and conditions.
6. This License Agreement shall be governed by and interpreted in all respects by the law of the State of California, excluding conflict of law provisions. Nothing in this License Agreement shall be deemed to create any relationship of agency, partnership, or joint venture between BeOpen and Licensee. This License Agreement does not grant permission to use BeOpen trademarks or trade names in a trademark sense to endorse or promote products or services of Licensee, or any third party. As an exception, the "BeOpen Python" logos available at <http://www.pythonlabs.com/logos.html> may be used according to the permissions granted on that web page.
7. By copying, installing or otherwise using the software, Licensee agrees to be bound by the terms and conditions of this License Agreement.

C.2.3 LICENCE D'UTILISATION CNRI POUR PYTHON 1.6.1

1. This LICENSE AGREEMENT is between the Corporation for National Research Initiatives, having an office at 1895 Preston White Drive, Reston, VA 20191 ("CNRI"), and the Individual or Organization ("Licensee") accessing and otherwise using Python 1.6.1 software in source or binary form and its associated documentation.
2. Subject to the terms and conditions of this License Agreement, CNRI hereby grants Licensee a nonexclusive, royalty-free, world-wide license to reproduce,

(suite sur la page suivante)

(suite de la page précédente)

analyze, test, perform and/or display publicly, prepare derivative works, distribute, and otherwise use Python 1.6.1 alone or in any derivative version, provided, however, that CNRI's License Agreement and CNRI's notice of copyright, i.e., "Copyright © 1995-2001 Corporation for National Research Initiatives; All Rights Reserved" are retained in Python 1.6.1 alone or in any derivative version prepared by Licensee. Alternately, in lieu of CNRI's License Agreement, Licensee may substitute the following text (omitting the quotes): "Python 1.6.1 is made available subject to the terms and conditions in CNRI's License Agreement. This Agreement together with Python 1.6.1 may be located on the Internet using the following unique, persistent identifier (known as a handle): 1895.22/1013. This Agreement may also be obtained from a proxy server on the Internet using the following URL: <http://hdl.handle.net/1895.22/1013>."

3. In the event Licensee prepares a derivative work that is based on or incorporates Python 1.6.1 or any part thereof, and wants to make the derivative work available to others as provided herein, then Licensee hereby agrees to include in any such work a brief summary of the changes made to Python 1.6.1.
4. CNRI is making Python 1.6.1 available to Licensee on an "AS IS" basis. CNRI MAKES NO REPRESENTATIONS OR WARRANTIES, EXPRESS OR IMPLIED. BY WAY OF EXAMPLE, BUT NOT LIMITATION, CNRI MAKES NO AND DISCLAIMS ANY REPRESENTATION OR WARRANTY OF MERCHANTABILITY OR FITNESS FOR ANY PARTICULAR PURPOSE OR THAT THE USE OF PYTHON 1.6.1 WILL NOT INFRINGE ANY THIRD PARTY RIGHTS.
5. CNRI SHALL NOT BE LIABLE TO LICENSEE OR ANY OTHER USERS OF PYTHON 1.6.1 FOR ANY INCIDENTAL, SPECIAL, OR CONSEQUENTIAL DAMAGES OR LOSS AS A RESULT OF MODIFYING, DISTRIBUTING, OR OTHERWISE USING PYTHON 1.6.1, OR ANY DERIVATIVE THEREOF, EVEN IF ADVISED OF THE POSSIBILITY THEREOF.
6. This License Agreement will automatically terminate upon a material breach of its terms and conditions.
7. This License Agreement shall be governed by the federal intellectual property law of the United States, including without limitation the federal copyright law, and, to the extent such U.S. federal law does not apply, by the law of the Commonwealth of Virginia, excluding Virginia's conflict of law provisions. Notwithstanding the foregoing, with regard to derivative works based on Python 1.6.1 that incorporate non-separable material that was previously distributed under the GNU General Public License (GPL), the law of the Commonwealth of Virginia shall govern this License Agreement only as to issues arising under or with respect to Paragraphs 4, 5, and 7 of this License Agreement. Nothing in this License Agreement shall be deemed to create any relationship of agency, partnership, or joint venture between CNRI and Licensee. This License Agreement does not grant permission to use CNRI trademarks or trade name in a trademark sense to endorse or promote products or services of Licensee, or any third party.
8. By clicking on the "ACCEPT" button where indicated, or by copying, installing or otherwise using Python 1.6.1, Licensee agrees to be bound by the terms and conditions of this License Agreement.

C.2.4 LICENCE D'UTILISATION CWI POUR PYTHON 0.9.0 à 1.2

Copyright © 1991 - 1995, Stichting Mathematisch Centrum Amsterdam, The Netherlands. All rights reserved.

Permission to use, copy, modify, and distribute this software and its documentation for any purpose and without fee is hereby granted, provided that the above copyright notice appear in all copies and that both that copyright notice and this permission notice appear in supporting documentation, and that the name of Stichting Mathematisch Centrum or CWI not be used in advertising or publicity pertaining to distribution of the software without specific, written prior permission.

STICHTING MATHEMATISCH CENTRUM DISCLAIMS ALL WARRANTIES WITH REGARD TO THIS SOFTWARE, INCLUDING ALL IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS, IN NO EVENT SHALL STICHTING MATHEMATISCH CENTRUM BE LIABLE FOR ANY SPECIAL, INDIRECT OR CONSEQUENTIAL DAMAGES OR ANY DAMAGES WHATSOEVER RESULTING FROM LOSS OF USE, DATA OR PROFITS, WHETHER IN AN ACTION OF CONTRACT, NEGLIGENCE OR OTHER TORTIOUS ACTION, ARISING OUT OF OR IN CONNECTION WITH THE USE OR PERFORMANCE OF THIS SOFTWARE.

C.3 Licences et Remerciements pour les logiciels inclus

Cette section est une liste incomplète mais grandissante de licences et remerciements pour les logiciels tiers incorporés dans la distribution de Python.

C.3.1 Mersenne twister

Le module `_random` inclut du code construit à partir d'un téléchargement depuis <http://www.math.sci.hiroshima-u.ac.jp/~m-mat/MT/MT2002/emt19937ar.html>. Voici mot pour mot les commentaires du code original :

A C-program for MT19937, with initialization improved 2002/1/26.
Coded by Takuji Nishimura and Makoto Matsumoto.

Before using, initialize the state by using `init_genrand(seed)`
or `init_by_array(init_key, key_length)`.

Copyright (C) 1997 - 2002, Makoto Matsumoto and Takuji Nishimura,
All rights reserved.

Redistribution and use in source and binary forms, with or without
modification, are permitted provided that the following conditions
are met:

1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.

(suite sur la page suivante)

(suite de la page précédente)

3. The names of its contributors may not be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

Any feedback is very welcome.

<http://www.math.sci.hiroshima-u.ac.jp/~m-mat/MT/emt.html>

email: m-mat @ math.sci.hiroshima-u.ac.jp (remove space)

C.3.2 Interfaces de connexion (sockets)

Le module `socket` utilise les fonctions `getaddrinfo()` et `getnameinfo()` codées dans des fichiers source séparés et provenant du projet WIDE : <http://www.wide.ad.jp/>.

Copyright (C) 1995, 1996, 1997, and 1998 WIDE Project.
All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
3. Neither the name of the project nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE PROJECT AND CONTRIBUTORS ``AS IS'' AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE PROJECT OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

C.3.3 Virgule flottante et contrôle d'exception

Le code source pour le module `fpectl` inclut la note suivante :

```

-----
/                               Copyright (c) 1996.                               \
|                               The Regents of the University of California.          |
|                               All rights reserved.                                |
|                                                                                 |
| Permission to use, copy, modify, and distribute this software for                |
| any purpose without fee is hereby granted, provided that this en-               |
| tire notice is included in all copies of any software which is or               |
| includes a copy or modification of this software and in all                     |
| copies of the supporting documentation for such software.                       |
|                                                                                 |
| This work was produced at the University of California, Lawrence                 |
| Livermore National Laboratory under contract no. W-7405-ENG-48                  |
| between the U.S. Department of Energy and The Regents of the                   |
| University of California for the operation of UC LLNL.                         |
|                                                                                 |
|                               DISCLAIMER                                           |
|                                                                                 |
| This software was prepared as an account of work sponsored by an                |
| agency of the United States Government. Neither the United States               |
| Government nor the University of California nor any of their em-                |
| ployees, makes any warranty, express or implied, or assumes any                 |
| liability or responsibility for the accuracy, completeness, or                  |
| usefulness of any information, apparatus, product, or process                   |
| disclosed, or represents that its use would not infringe                       |
| privately-owned rights. Reference herein to any specific commer-                |
| cial products, process, or service by trade name, trademark,                   |
| manufacturer, or otherwise, does not necessarily constitute or                 |
| imply its endorsement, recommendation, or favoring by the United               |
| States Government or the University of California. The views and                |
| opinions of authors expressed herein do not necessarily state or                |
| reflect those of the United States Government or the University                 |
| of California, and shall not be used for advertising or product                 |
| \ endorsement purposes.                                                         /
-----

```

C.3.4 MD5 message digest algorithm

The source code for the `md5` module contains the following notice :

```

Copyright (C) 1999, 2002 Aladdin Enterprises. All rights reserved.

This software is provided 'as-is', without any express or implied
warranty. In no event will the authors be held liable for any damages
arising from the use of this software.

Permission is granted to anyone to use this software for any purpose,
including commercial applications, and to alter it and redistribute it
freely, subject to the following restrictions:

```

(suite sur la page suivante)

(suite de la page précédente)

1. The origin of this software must not be misrepresented; you must not claim that you wrote the original software. If you use this software in a product, an acknowledgment in the product documentation would be appreciated but is not required.
2. Altered source versions must be plainly marked as such, and must not be misrepresented as being the original software.
3. This notice may not be removed or altered from any source distribution.

L. Peter Deutsch
ghost@aladdin.com

Independent implementation of MD5 (RFC 1321).

This code implements the MD5 Algorithm defined in RFC 1321, whose text is available at

<http://www.ietf.org/rfc/rfc1321.txt>

The code is derived from the text of the RFC, including the test suite (section A.5) but excluding the rest of Appendix A. It does not include any code or documentation that is identified in the RFC as being copyrighted.

The original and principal author of md5.h is L. Peter Deutsch <ghost@aladdin.com>. Other authors are noted in the change history that follows (in reverse chronological order):

2002-04-13 lpd Removed support for non-ANSI compilers; removed references to Ghostscript; clarified derivation from RFC 1321; now handles byte order either statically or dynamically.
1999-11-04 lpd Edited comments slightly for automatic TOC extraction.
1999-10-18 lpd Fixed typo in header comment (ansi2knr rather than md5); added conditionalization for C++ compilation from Martin Purschke <purschke@bnl.gov>.
1999-05-03 lpd Original version.

C.3.5 Interfaces de connexion asynchrones

Les modules `asynchat` et `asyncore` contiennent la note suivante :

Copyright 1996 by Sam Rushing

All Rights Reserved

Permission to use, copy, modify, and distribute this software and its documentation for any purpose and without fee is hereby granted, provided that the above copyright notice appear in all copies and that both that copyright notice and this permission notice appear in supporting documentation, and that the name of Sam Rushing not be used in advertising or publicity pertaining to distribution of the software without specific, written prior permission.

SAM RUSHING DISCLAIMS ALL WARRANTIES WITH REGARD TO THIS SOFTWARE,

(suite sur la page suivante)

(suite de la page précédente)

INCLUDING ALL IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS, IN NO EVENT SHALL SAM RUSHING BE LIABLE FOR ANY SPECIAL, INDIRECT OR CONSEQUENTIAL DAMAGES OR ANY DAMAGES WHATSOEVER RESULTING FROM LOSS OF USE, DATA OR PROFITS, WHETHER IN AN ACTION OF CONTRACT, NEGLIGENCE OR OTHER TORTIOUS ACTION, ARISING OUT OF OR IN CONNECTION WITH THE USE OR PERFORMANCE OF THIS SOFTWARE.

C.3.6 Gestion de témoin (*cookie*)

The Cookie module contains the following notice :

Copyright 2000 by Timothy O'Malley <timo@alum.mit.edu>

All Rights Reserved

Permission to use, copy, modify, and distribute this software and its documentation for any purpose and without fee is hereby granted, provided that the above copyright notice appear in all copies and that both that copyright notice and this permission notice appear in supporting documentation, and that the name of Timothy O'Malley not be used in advertising or publicity pertaining to distribution of the software without specific, written prior permission.

Timothy O'Malley DISCLAIMS ALL WARRANTIES WITH REGARD TO THIS SOFTWARE, INCLUDING ALL IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS, IN NO EVENT SHALL Timothy O'Malley BE LIABLE FOR ANY SPECIAL, INDIRECT OR CONSEQUENTIAL DAMAGES OR ANY DAMAGES WHATSOEVER RESULTING FROM LOSS OF USE, DATA OR PROFITS, WHETHER IN AN ACTION OF CONTRACT, NEGLIGENCE OR OTHER TORTIOUS ACTION, ARISING OUT OF OR IN CONNECTION WITH THE USE OR PERFORMANCE OF THIS SOFTWARE.

C.3.7 Traçage d'exécution

Le module `trace` contient la note suivante :

portions copyright 2001, Autonomous Zones Industries, Inc., all rights...
err... reserved and offered to the public under the terms of the
Python 2.2 license.
Author: Zooko O'Whielacronx
<http://zooko.com/>
<mailto:zooko@zooko.com>

Copyright 2000, Mojam Media, Inc., all rights reserved.
Author: Skip Montanaro

Copyright 1999, Bioreason, Inc., all rights reserved.
Author: Andrew Dalke

Copyright 1995-1997, Automatrix, Inc., all rights reserved.
Author: Skip Montanaro

(suite sur la page suivante)

(suite de la page précédente)

Copyright 1991-1995, Stichting Mathematisch Centrum, all rights reserved.

Permission to use, copy, modify, and distribute this Python software and its associated documentation for any purpose without fee is hereby granted, provided that the above copyright notice appears in all copies, and that both that copyright notice and this permission notice appear in supporting documentation, and that the name of neither Automatrix, Bioreason or Mojam Media be used in advertising or publicity pertaining to distribution of the software without specific, written prior permission.

C.3.8 Les fonctions UUencode et UUdecode

Le module uu contient la note suivante :

Copyright 1994 by Lance Ellinghouse
Cathedral City, California Republic, United States of America.

All Rights Reserved

Permission to use, copy, modify, and distribute this software and its documentation for any purpose and without fee is hereby granted, provided that the above copyright notice appear in all copies and that both that copyright notice and this permission notice appear in supporting documentation, and that the name of Lance Ellinghouse not be used in advertising or publicity pertaining to distribution of the software without specific, written prior permission.

LANCE ELLINGHOUSE DISCLAIMS ALL WARRANTIES WITH REGARD TO THIS SOFTWARE, INCLUDING ALL IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS, IN NO EVENT SHALL LANCE ELLINGHOUSE CENTRUM BE LIABLE FOR ANY SPECIAL, INDIRECT OR CONSEQUENTIAL DAMAGES OR ANY DAMAGES WHATSOEVER RESULTING FROM LOSS OF USE, DATA OR PROFITS, WHETHER IN AN ACTION OF CONTRACT, NEGLIGENCE OR OTHER TORTIOUS ACTION, ARISING OUT OF OR IN CONNECTION WITH THE USE OR PERFORMANCE OF THIS SOFTWARE.

Modified by Jack Jansen, CWI, July 1995:

- Use binascii module to do the actual line-by-line conversion between ascii and binary. This results in a 1000-fold speedup. The C version is still 5 times faster, though.
- Arguments more compliant with Python standard

C.3.9 Appel de procédures distantes en XML (*RPC*, pour *Remote Procedure Call*)

The xmlrpclib module contains the following notice :

The XML-RPC client interface is

Copyright (c) 1999-2002 by Secret Labs AB
Copyright (c) 1999-2002 by Fredrik Lundh

By obtaining, using, and/or copying this software and/or its associated documentation, you agree that you have read, understood, and will comply with the following terms and conditions:

(suite sur la page suivante)

(suite de la page précédente)

Permission to use, copy, modify, and distribute this software and its associated documentation for any purpose and without fee is hereby granted, provided that the above copyright notice appears in all copies, and that both that copyright notice and this permission notice appear in supporting documentation, and that the name of Secret Labs AB or the author not be used in advertising or publicity pertaining to distribution of the software without specific, written prior permission.

SECRET LABS AB AND THE AUTHOR DISCLAIMS ALL WARRANTIES WITH REGARD TO THIS SOFTWARE, INCLUDING ALL IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS. IN NO EVENT SHALL SECRET LABS AB OR THE AUTHOR BE LIABLE FOR ANY SPECIAL, INDIRECT OR CONSEQUENTIAL DAMAGES OR ANY DAMAGES WHATSOEVER RESULTING FROM LOSS OF USE, DATA OR PROFITS, WHETHER IN AN ACTION OF CONTRACT, NEGLIGENCE OR OTHER TORTIOUS ACTION, ARISING OUT OF OR IN CONNECTION WITH THE USE OR PERFORMANCE OF THIS SOFTWARE.

C.3.10 test_epoll

Le module `test_epoll` contient la note suivante :

Copyright (c) 2001-2006 Twisted Matrix Laboratories.

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

C.3.11 Select kqueue

Le module `select` contient la note suivante pour l'interface `kqueue` :

Copyright (c) 2000 Doug White, 2006 James Knight, 2007 Christian Heimes
All rights reserved.

Redistribution and use in source and binary forms, with or without

(suite sur la page suivante)

(suite de la page précédente)

modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.

THIS SOFTWARE IS PROVIDED BY THE AUTHOR AND CONTRIBUTORS ``AS IS'' AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

C.3.12 *strtod* et *dtoa*

Le fichier `Python/dtoa.c`, qui fournit les fonctions `dtoa` et `strtod` pour la conversions de *doubles C* vers et depuis les chaînes, et tiré d'un fichier du même nom par David M. Gay, actuellement disponible sur <http://www.netlib.org/fp/>. Le fichier original, tel que récupéré le 16 mars 2009, contient la licence suivante :

```

/*****
 *
 * The author of this software is David M. Gay.
 *
 * Copyright (c) 1991, 2000, 2001 by Lucent Technologies.
 *
 * Permission to use, copy, modify, and distribute this software for any
 * purpose without fee is hereby granted, provided that this entire notice
 * is included in all copies of any software which is or includes a copy
 * or modification of this software and in all copies of the supporting
 * documentation for such software.
 *
 * THIS SOFTWARE IS BEING PROVIDED "AS IS", WITHOUT ANY EXPRESS OR IMPLIED
 * WARRANTY. IN PARTICULAR, NEITHER THE AUTHOR NOR LUCENT MAKES ANY
 * REPRESENTATION OR WARRANTY OF ANY KIND CONCERNING THE MERCHANTABILITY
 * OF THIS SOFTWARE OR ITS FITNESS FOR ANY PARTICULAR PURPOSE.
 *
 *****/

```

C.3.13 OpenSSL

Les modules `hashlib`, `posix`, `ssl`, et `crypt` utilisent la bibliothèque OpenSSL pour améliorer les performances, si elle est disponible via le système d'exploitation. Aussi les outils d'installation sur Windows et Mac OS X peuvent inclure une copie des bibliothèques d'OpenSSL, donc on colle une copie de la licence d'OpenSSL ici :

LICENSE ISSUES

=====

The OpenSSL toolkit stays under a dual license, i.e. both the conditions of the OpenSSL License and the original SSLeay license apply to the toolkit. See below for the actual license texts. Actually both licenses are BSD-style Open Source licenses. In case of any license issues related to OpenSSL please contact openssl-core@openssl.org.

OpenSSL License

```

/* =====
 * Copyright (c) 1998-2008 The OpenSSL Project. All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * 1. Redistributions of source code must retain the above copyright
 * notice, this list of conditions and the following disclaimer.
 *
 * 2. Redistributions in binary form must reproduce the above copyright
 * notice, this list of conditions and the following disclaimer in
 * the documentation and/or other materials provided with the
 * distribution.
 *
 * 3. All advertising materials mentioning features or use of this
 * software must display the following acknowledgment:
 * "This product includes software developed by the OpenSSL Project
 * for use in the OpenSSL Toolkit. (http://www.openssl.org/)"
 *
 * 4. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to
 * endorse or promote products derived from this software without
 * prior written permission. For written permission, please contact
 * openssl-core@openssl.org.
 *
 * 5. Products derived from this software may not be called "OpenSSL"
 * nor may "OpenSSL" appear in their names without prior written
 * permission of the OpenSSL Project.
 *
 * 6. Redistributions of any form whatsoever must retain the following
 * acknowledgment:
 * "This product includes software developed by the OpenSSL Project
 * for use in the OpenSSL Toolkit (http://www.openssl.org/)"
 *
 * THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT ``AS IS'' AND ANY
 * EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE OpenSSL PROJECT OR
 * ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
 * SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT

```

(suite sur la page suivante)

(suite de la page précédente)

```

* NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
* LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
* HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
* STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
* ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
* OF THE POSSIBILITY OF SUCH DAMAGE.
* =====
*
* This product includes cryptographic software written by Eric Young
* (eay@cryptsoft.com).  This product includes software written by Tim
* Hudson (tjh@cryptsoft.com).
*
*/

```

Original SSLeay License

```

/* Copyright (C) 1995-1998 Eric Young (eay@cryptsoft.com)
* All rights reserved.
*
* This package is an SSL implementation written
* by Eric Young (eay@cryptsoft.com).
* The implementation was written so as to conform with Netscapes SSL.
*
* This library is free for commercial and non-commercial use as long as
* the following conditions are aheared to.  The following conditions
* apply to all code found in this distribution, be it the RC4, RSA,
* lhash, DES, etc., code; not just the SSL code.  The SSL documentation
* included with this distribution is covered by the same copyright terms
* except that the holder is Tim Hudson (tjh@cryptsoft.com).
*
* Copyright remains Eric Young's, and as such any Copyright notices in
* the code are not to be removed.
* If this package is used in a product, Eric Young should be given attribution
* as the author of the parts of the library used.
* This can be in the form of a textual message at program startup or
* in documentation (online or textual) provided with the package.
*
* Redistribution and use in source and binary forms, with or without
* modification, are permitted provided that the following conditions
* are met:
* 1. Redistributions of source code must retain the copyright
*    notice, this list of conditions and the following disclaimer.
* 2. Redistributions in binary form must reproduce the above copyright
*    notice, this list of conditions and the following disclaimer in the
*    documentation and/or other materials provided with the distribution.
* 3. All advertising materials mentioning features or use of this software
*    must display the following acknowledgement:
*    "This product includes cryptographic software written by
*     Eric Young (eay@cryptsoft.com)"
*    The word 'cryptographic' can be left out if the rouines from the library
*    being used are not cryptographic related :-).

```

(suite sur la page suivante)

(suite de la page précédente)

```

* 4. If you include any Windows specific code (or a derivative thereof) from
*     the apps directory (application code) you must include an acknowledgement:
*     "This product includes software written by Tim Hudson (tjh@cryptsoft.com)"
*
* THIS SOFTWARE IS PROVIDED BY ERIC YOUNG ``AS IS'' AND
* ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
* IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
* ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
* FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
* DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
* OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
* HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
* LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
* OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
* SUCH DAMAGE.
*
* The licence and distribution terms for any publically available version or
* derivative of this code cannot be changed.  i.e. this code cannot simply be
* copied and put under another distribution licence
* [including the GNU Public Licence.]
*/

```

C.3.14 expat

Le module `pyexpat` est compilé avec une copie des sources d'*expat*, sauf si la compilation est configurée avec `--with-system-expat` :

```

Copyright (c) 1998, 1999, 2000 Thai Open Source Software Center Ltd
and Clark Cooper

Permission is hereby granted, free of charge, to any person obtaining
a copy of this software and associated documentation files (the
"Software"), to deal in the Software without restriction, including
without limitation the rights to use, copy, modify, merge, publish,
distribute, sublicense, and/or sell copies of the Software, and to
permit persons to whom the Software is furnished to do so, subject to
the following conditions:

The above copyright notice and this permission notice shall be included
in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND,
EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF
MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT.
IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY
CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT,
TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE
SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

```

C.3.15 libffi

Le module `_ctypes` est compilé en utilisant une copie des sources de la *libffi*, sauf si la compilation est configurée avec `--with-system-libffi` :

```
Copyright (c) 1996-2008 Red Hat, Inc and others.
```

```
Permission is hereby granted, free of charge, to any person obtaining
a copy of this software and associated documentation files (the
``Software''), to deal in the Software without restriction, including
without limitation the rights to use, copy, modify, merge, publish,
distribute, sublicense, and/or sell copies of the Software, and to
permit persons to whom the Software is furnished to do so, subject to
the following conditions:
```

```
The above copyright notice and this permission notice shall be included
in all copies or substantial portions of the Software.
```

```
THE SOFTWARE IS PROVIDED ``AS IS'', WITHOUT WARRANTY OF ANY KIND,
EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF
MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND
NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT
HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY,
WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER
DEALINGS IN THE SOFTWARE.
```

C.3.16 zlib

Le module `zlib` est compilé en utilisant une copie du code source de *zlib* si la version de *zlib* trouvée sur le système est trop vieille pour être utilisée :

```
Copyright (C) 1995-2010 Jean-loup Gailly and Mark Adler
```

```
This software is provided 'as-is', without any express or implied
warranty. In no event will the authors be held liable for any damages
arising from the use of this software.
```

```
Permission is granted to anyone to use this software for any purpose,
including commercial applications, and to alter it and redistribute it
freely, subject to the following restrictions:
```

1. The origin of this software must not be misrepresented; you must not claim that you wrote the original software. If you use this software in a product, an acknowledgment in the product documentation would be appreciated but is not required.
2. Altered source versions must be plainly marked as such, and must not be misrepresented as being the original software.
3. This notice may not be removed or altered from any source distribution.

```
Jean-loup Gailly  
jloup@gzip.org
```

```
Mark Adler  
madler@alumni.caltech.edu
```


ANNEXE D

Copyright

Python et cette documentation sont :

Copyright © 2001-2019 Python Software Foundation. All rights reserved.

Copyright © 2000 *BeOpen.com*. Tous droits réservés.

Copyright © 1995-2000 *Corporation for National Research Initiatives*. Tous droits réservés.

Copyright © 1991-1995 *Stichting Mathematisch Centrum*. Tous droits réservés.

Voir [Histoire et licence](#) pour des informations complètes concernant la licence et les permissions.

Non-alphabetical

..., [25](#)

-?

option de ligne de commande, [5](#)

%PATH%, [16](#)

2to3, [25](#)

-3

option de ligne de commande, [8](#)

>>>, [25](#)

__future__, [28](#)

__slots__, [32](#)

A

argument, [25](#)

argument nommé, [30](#)

argument positionnel, [32](#)

attribut, [26](#)

B

-B

option de ligne de commande, [5](#)

-b

option de ligne de commande, [5](#)

BDFL, [26](#)

C

-c <command>

option de ligne de commande, [4](#)

chaîne entre triple guillemets, [33](#)

chargeur, [30](#)

chercheur, [28](#)

classe, [26](#)

classe de base abstraite, [25](#)

classic class, [26](#)

code intermédiaire (bytecode), [26](#)

coercition, [26](#)

CPython, [26](#)

D

-d

option de ligne de commande, [5](#)

décorateur, [27](#)

descripteur, [27](#)

dictionnaire, [27](#)

division entière, [28](#)

docstring, [27](#)

duck-typing, [27](#)

E

-E

option de ligne de commande, [5](#)

EAFP, [27](#)

environnement virtuel, [33](#)

espace de noms, [31](#)

exec_prefix, [12](#)

expression, [27](#)

expression génératrice, [28](#)

F

fonction, [28](#)

fonction clé, [30](#)

G

générateur, [28](#)

generator, [28](#)

generator expression, [28](#)

gestionnaire de contexte, [26](#)

GIL, [28](#)

H

-h

option de ligne de commande, [5](#)

hachable, [29](#)

-help

option de ligne de commande, [5](#)

I

-i

option de ligne de commande, [6](#)

IDLE, **29**
 immuable, **29**
 importateur, **29**
 importer, **29**
 instruction, **33**
 integer division, **29**
 interactif, **29**
 interprété, **29**
 itérable, **29**
 itérateur, **29**

J
 -J
 option de ligne de commande, **8**

L
 lambda, **30**
 LBYL, **30**
 Le zen de Python, **33**
 list, **30**
 liste en compréhension (ou liste en intension), **30**

M
 -m <module-name>
 option de ligne de commande, **4**
 machine virtuelle, **33**
 métaclasse, **30**
 méthode, **31**
 méthode spéciale, **33**
 module, **31**
 module d'extension, **27**
 MRO, **31**
 muable, **31**

N
 nombre complexe, **26**
 nombre de références, **32**
 nouvelle classe, **31**
 n-uplet nommé, **31**

O
 -O
 option de ligne de commande, **6**
 objet, **31**
 Objet bytes-compatible, **26**
 objet fichier, **28**
 objet fichier-compatible, **28**
 -OO
 option de ligne de commande, **6**
 option de ligne de commande
 -?, **5**
 -3, **8**
 -B, **5**
 -b, **5**

 -c <command>, **4**
 -d, **5**
 -E, **5**
 -h, **5**
 -help, **5**
 -i, **6**
 -J, **8**
 -m <module-name>, **4**
 -O, **6**
 -OO, **6**
 -Q <arg>, **6**
 -R, **6**
 -S, **6**
 -s, **6**
 -t, **6**
 -U, **8**
 -u, **6**
 -V, **5**
 -v, **7**
 -version, **5**
 -W arg, **7**
 -X, **8**
 -x, **8**
 ordre de résolution des méthodes, **31**

P
 paquet, **31**
 paramètre, **31**
 PATH, **8, 13**
 PEP, **32**
 portée imbriquée, **31**
 prefix, **12**
 Python 3000, **32**
 Python Enhancement Proposals
 PEP 1, **32**
 PEP 8, **13**
 PEP 11, **15**
 PEP 230, **6, 7**
 PEP 278, **33**
 PEP 302, **28, 30**
 PEP 328, **28**
 PEP 338, **4**
 PEP 343, **26**
 PEP 370, **6, 10**
 PEP 3116, **33**
 PYTHON*, **5**
 PYTHONDEBUG, **5**
 PYTHONDONTWRITEBYTECODE, **5, 9**
 PYTHONHASHSEED, **6, 9**
 PYTHONHOME, **5, 8, 17**
 PYTHONINSPECT, **6**
 Pythonique, **32**
 PYTHONOPTIMIZE, **6**
 PYTHONPATH, **5, 8, 9, 17, 22**

PYTHONSTARTUP, 6
 PYTHONUNBUFFERED, 7
 PYTHONVERBOSE, 7
 PYTHONWARNINGS, 8

Q

-Q <arg>
 option de ligne de commande, 6

R

-R
 option de ligne de commande, 6
 ramasse-miettes, 28
 retours à la ligne universels, 33

S

-S
 option de ligne de commande, 6
 -s
 option de ligne de commande, 6
 séquence, 32
 struct sequence, 33

T

-t
 option de ligne de commande, 6
 Tableau de correspondances, 30
 tranche, 32
 type, 33

U

-U
 option de ligne de commande, 8
 -u
 option de ligne de commande, 6

V

-V
 option de ligne de commande, 5
 -v
 option de ligne de commande, 7
 variable d'environnement
 %PATH%, 16
 exec_prefix, 12
 PATH, 8, 13
 prefix, 12
 PYTHON*, 5
 PYTHONCASEOK, 9
 PYTHONDEBUG, 5, 9
 PYTHONDONTWRITEBYTECODE, 5, 9
 PYTHONDUMPREFS, 10
 PYTHONEXECUTABLE, 10
 PYTHONHASHSEED, 6, 9

PYTHONHOME, 5, 8, 17
 PYTHONHTTPSVERIFY, 10
 PYTHONINSPECT, 6, 9
 PYTHONIOENCODING, 9
 PYTHONMALLOCSTATS, 10
 PYTHONNOUSERSITE, 10
 PYTHONOPTIMIZE, 6, 9
 PYTHONPATH, 5, 8, 9, 17, 22
 PYTHONSHOWALLOCCOUNT, 10
 PYTHONSHOWREFCOUNT, 10
 PYTHONSTARTUP, 6, 9
 PYTHONTHREADDEBUG, 10
 PYTHONUNBUFFERED, 7, 9
 PYTHONUSERBASE, 10
 PYTHONVERBOSE, 7, 9
 PYTHONWARNINGS, 8, 10
 PYTHON2K, 9

verrou global de l'interpréteur, 28

-version

option de ligne de commande, 5

vue de dictionnaire, 27

W

-W arg
 option de ligne de commande, 7

X

-X
 option de ligne de commande, 8
 -x
 option de ligne de commande, 8