Python Setup and Usage

Release 2.7.18

Guido van Rossum
and the Python development team

maio 07, 2020

Python Software Foundation
Email: docs@python.org

Sumário

Esta parte da documentação é dedicada a informações gerais sobre a configuração do ambiente Python em diferentes plataformas, a chamada do interpretador e outras coisas que facilitam o trabalho com o Python.

CAPÍTULO 1

Linha de comando e ambiente

The CPython interpreter scans the command line and the environment for various settings.

CPython implementation detail: Other implementations' command line schemes may differ. See implementations for further resources.

1.1 Linha de comando

Ao invocar o Python, você pode especificar qualquer uma destas opções:

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

The most common use case is, of course, a simple invocation of a script:

```
python myscript.py
```

1.1.1 Opções de interface

The interpreter interface resembles that of the UNIX shell, but provides some additional methods of invocation:

- When called with standard input connected to a tty device, it prompts for commands and executes them until an EOF (an end-of-file character, you can produce that with `Ctrl-D` on UNIX or `Ctrl-Z, Enter` on Windows) is read.
- Quando chamado com um argumento de nome de arquivo ou com um arquivo como entrada padrão, ele lê e executa um script desse arquivo.
- When called with a directory name argument, it reads and executes an appropriately named script from that directory.
- When called with `-c command`, it executes the Python statement(s) given as *command*. Here *command* may contain multiple statements separated by newlines. Leading whitespace is significant in Python statements!

- When called with `-m module-name`, the given module is located on the Python module path and executed as a script.

In non-interactive mode, the entire input is parsed before it is executed.

An interface option terminates the list of options consumed by the interpreter, all consecutive arguments will end up in `sys.argv` – note that the first element, subscript zero (`sys.argv[0]`), is a string reflecting the program's source.

-c <command>
Execute the Python code in *command*. *command* can be one or more statements separated by newlines, with significant leading whitespace as in normal module code.

If this option is given, the first element of `sys.argv` will be `"-c"` and the current directory will be added to the start of `sys.path` (allowing modules in that directory to be imported as top level modules).

-m <module-name>
Search `sys.path` for the named module and execute its contents as the `__main__` module.

Since the argument is a *module* name, you must not give a file extension (`.py`). The `module-name` should be a valid Python module name, but the implementation may not always enforce this (e.g. it may allow you to use a name that includes a hyphen).

Package names are also permitted. When a package name is supplied instead of a normal module, the interpreter will execute <pkg>`.__main__` as the main module. This behaviour is deliberately similar to the handling of directories and zipfiles that are passed to the interpreter as the script argument.

Nota: This option cannot be used with built-in modules and extension modules written in C, since they do not have Python module files. However, it can still be used for precompiled modules, even if the original source file is not available.

If this option is given, the first element of `sys.argv` will be the full path to the module file. As with the *-c* option, the current directory will be added to the start of `sys.path`.

Many standard library modules contain code that is invoked on their execution as a script. An example is the `timeit` module:

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

Ver também:

runpy.run_module() Equivalent functionality directly available to Python code

PEP 338 – Executando módulos como scripts

Novo na versão 2.4.

Alterado na versão 2.5: The named module can now be located inside a package.

Alterado na versão 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"`)

-

Read commands from standard input (`sys.stdin`). If standard input is a terminal, *-i* is implied.

If this option is given, the first element of `sys.argv` will be `"-"` and the current directory will be added to the start of `sys.path`.

Ver também:

runpy.run_path() Equivalent functionality directly available to Python code

<script>
Execute the Python code contained in *script*, which must be a filesystem path (absolute or relative) referring to either a Python file, a directory containing a `__main__.py` file, or a zipfile containing a `__main__.py` file.

If this option is given, the first element of `sys.argv` will be the script name as given on the command line.

If the script name refers directly to a Python file, the directory containing that file is added to the start of `sys.path`, and the file is executed as the `__main__` module.

If the script name refers to a directory or zipfile, the script name is added to the start of `sys.path` and the `__main__.py` file in that location is executed as the `__main__` module.

Alterado na versão 2.5: Directories and zipfiles containing a `__main__.py` file at the top level are now considered valid Python scripts.

If no interface option is given, *-i* is implied, `sys.argv[0]` is an empty string (`""`) and the current directory will be added to the start of `sys.path`.

Ver também:

tut-invoking

1.1.2 Opções genéricas

-?
-h
--help
Print a short description of all command line options.

Alterado na versão 2.5: The `--help` variant.

-V
--version
Print the Python version number and exit. Example output could be:

```
Python 2.5.1
```

Alterado na versão 2.5: The `--version` variant.

1.1.3 Miscellaneous options

-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.

Novo na versão 2.6.

-B
If given, Python won't try to write `.pyc` or `.pyo` files on the import of source modules. See also *PYTHONDONTWRITEBYTECODE*.

Novo na versão 2.6.

-d
Turn on parser debugging output (for wizards only, depending on compilation options). See also *PYTHONDEBUG*.

-E

Ignore all `PYTHON*` environment variables, e.g. *PYTHONPATH* and *PYTHONHOME*, that might be set.

Novo na versão 2.2.

-i

When a script is passed as first argument or the *-c* option is used, enter interactive mode after executing the script or the command, even when `sys.stdin` does not appear to be a terminal. The *PYTHONSTARTUP* file is not read.

This can be useful to inspect global variables or a stack trace when a script raises an exception. See also *PYTHONINSPECT*.

-O

Turn on basic optimizations. This changes the filename extension for compiled (*bytecode*) files from `.pyc` to `.pyo`. See also *PYTHONOPTIMIZE*.

-OO

Discard docstrings in addition to the *-O* optimizations.

-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

Ver também:

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

PEP 238 – Changing the division operator

-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.

See also *PYTHONHASHSEED*.

Novo na versão 2.6.8.

-s

Don't add the `user site-packages directory` to `sys.path`.

Novo na versão 2.6.

Ver também:

PEP 370 – Per user site-packages directory

-S

Disable the import of the module `site` and the site-dependent manipulations of `sys.path` that it entails.

-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 bltin-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.

See also *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
```

By default, each warning is printed once for each source line where it occurs. This option controls how often warnings are printed.

Multiple *-W* options may be given; when a warning matches more than one option, the action for the last matching option is performed. Invalid *-W* options are ignored (though, a warning message is printed about invalid options when the first warning is issued).

Starting from Python 2.7, `DeprecationWarning` and its descendants are ignored by default. The `-Wd` option can be used to re-enable them.

Warnings can also be controlled from within a Python program using the `warnings` module.

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

ignore Ignore all warnings.

default Explicitly request the default behavior (printing each warning once per source line).

all Print a warning each time it occurs (this may generate many messages if a warning is triggered repeatedly for the same source line, such as inside a loop).

module Print each warning only the first time it occurs in each module.

once Print each warning only the first time it occurs in the program.

error Raise an exception instead of printing a warning message.

The full form of argument is:

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

Here, *action* is as explained above but only applies to messages that match the remaining fields. Empty fields match all values; trailing empty fields may be omitted. The *message* field matches the start of the warning message printed; this match is case-insensitive. The *category* field matches the warning category. This must be a class name; the match tests whether the actual warning category of the message is a subclass of the specified warning category.

The full class name must be given. The *module* field matches the (fully-qualified) module name; this match is case-sensitive. The *line* field matches the line number, where zero matches all line numbers and is thus equivalent to an omitted line number.

Ver também:

`warnings` – the warnings module

PEP 230 – Warning framework

PYTHONWARNINGS

-x

Skip the first line of the source, allowing use of non-Unix forms of `#!cmd`. This is intended for a DOS specific hack only.

-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.

Novo na versão 2.6.

See /howto/pyporting for more details.

1.1.4 Options you shouldn't use

-J

Reserved for use by 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 Environment variables

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

Change the location of the standard Python libraries. By default, the libraries are searched in *prefix*`/lib/pythonversion` and *exec_prefix*`/lib/pythonversion`, where *prefix* and *exec_prefix* are installation-dependent directories, both defaulting to `/usr/local`.

When *PYTHONHOME* is set to a single directory, its value replaces both *prefix* and *exec_prefix*. To specify different values for these, set *PYTHONHOME* to *prefix*`:`*exec_prefix*.

PYTHONPATH

Augment the default search path for module files. The format is the same as the shell's `PATH`: one or more directory pathnames separated by `os.pathsep` (e.g. colons on Unix or semicolons on Windows). Non-existent directories are silently ignored.

In addition to normal directories, individual *PYTHONPATH* entries may refer to zipfiles containing pure Python modules (in either source or compiled form). Extension modules cannot be imported from zipfiles.

The default search path is installation dependent, but generally begins with *prefix*/lib/python*version* (see *PYTHONHOME* above). It is *always* appended to *PYTHONPATH*.

An additional directory will be inserted in the search path in front of *PYTHONPATH* as described above under *Opções de interface*. The search path can be manipulated from within a Python program as the 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
If this is set to a non-empty string it is equivalent to specifying the *-O* option. If set to an integer, it is equivalent to specifying *-O* multiple times.

PYTHONDEBUG
If this is set to a non-empty string it is equivalent to specifying the *-d* option. If set to an integer, it is equivalent to specifying *-d* multiple times.

PYTHONINSPECT
If this is set to a non-empty string it is equivalent to specifying the *-i* option.

This variable can also be modified by Python code using `os.environ` to force inspect mode on program termination.

PYTHONUNBUFFERED
If this is set to a non-empty string it is equivalent to specifying the *-u* option.

PYTHONVERBOSE
If this is set to a non-empty string it is equivalent to specifying the *-v* option. If set to an integer, it is equivalent to specifying *-v* multiple times.

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

PYTHONDONTWRITEBYTECODE
If this is set, Python won't try to write `.pyc` or `.pyo` files on the import of source modules. This is equivalent to specifying the *-B* option.

Novo na versão 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.

If *PYTHONHASHSEED* is set to an integer value, it is used as a fixed seed for generating the hash() of the types covered by the hash randomization.

Its purpose is to allow repeatable hashing, such as for selftests for the interpreter itself, or to allow a cluster of python processes to share hash values.

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.

Novo na versão 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()`.

Novo na versão 2.6.

PYTHONNOUSERSITE
If this is set, Python won't add the `user site-packages directory` to `sys.path`.

Novo na versão 2.6.

Ver também:

PEP 370 – Per user site-packages directory

PYTHONUSERBASE
Defines the `user base directory`, which is used to compute the path of the `user site-packages directory` and Distutils installation paths for `python setup.py install --user`.

Novo na versão 2.6.

Ver também:

PEP 370 – Per user site-packages directory

PYTHONEXECUTABLE
If this environment variable is set, `sys.argv[0]` will be set to its value instead of the value got through the C runtime. Only works on Mac OS X.

PYTHONWARNINGS
This is equivalent to the *-W* option. If set to a comma separated string, it is equivalent to specifying *-W* multiple times.

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.

Novo na versão 2.7.12.

1.2.1 Debug-mode variables

Setting these variables only has an effect in a debug build of Python, that is, if Python was configured with the `--with-pydebug` build option.

PYTHONTHREADDEBUG
If set, Python will print threading debug info.

Alterado na versão 2.6: Previously, this variable was called `THREADDEBUG`.

PYTHONDUMPREFS
If set, Python will dump objects and reference counts still alive after shutting down the interpreter.

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.

Novo na versão 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.

Novo na versão 2.7.15.

CAPÍTULO 2

Utilizando Python em plataformas Unix

2.1 Obtendo e instalando a versão mais recente do Python

2.1.1 No Linux

O Python vem pré-instalado na maioria das distribuições Linux e está disponível como um pacote em todos as outras. No entanto, existem certos recursos que podemos querer utilizar e que não estão disponíveis no pacote da sua distro. Poderás compilar facilmente a versão mais recente do Python desde a origem.

Nas situações em que o Python não vier pré-instalado e também não estiver nos repositórios, poderás facilmente gerar os pacotes para a sua distro. Veja os seguintes links:

Ver também:

https://www.debian.org/doc/manuals/maint-guide/first.en.html para usuários Debian

https://en.opensuse.org/Portal:Packaging para usuários OpenSuse

https://docs.fedoraproject.org/en-US/Fedora_Draft_Documentation/0.1/html/RPM_Guide/ch-creating-rpms.html
para usuários Fedora

http://www.slackbook.org/html/package-management-making-packages.html para usuários do Slackware

2.1.2 On FreeBSD e OpenBSD

- usuários do FreeBSD, para adicionar a utilização do pacote utilize:

```
pkg install python3
```

- Usuários do OpenBSD, para adicionar pacotes use:

```
pkg_add -r python

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

(continua na próxima página)

(continuação da página anterior)

Por exemplo, usuários i386 podem pegar a versão 2.5.1 do Python usando o comando:

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

2.1.3 No OpenSolaris

Podes baixar o Python desde OpenCSW. Várias versões do Python estão disponíveis e poderás instala-las, por exemplo `pkgutil -i python27`.

2.2 Compilando o 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.

Aviso: `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/python`*`version`*.

2.3 Paths e arquivos relacionados com o Python

Estes estão sujeitos a diferenças dependendo das convenções de instalação local; `prefix` (`${prefix}`) e `exec_prefix` (`${exec_prefix}`) dependem da instalação e devem ser interpretados da mesma forma que para o software GNU; eles poderão ser os mesmos.

Por exemplo, na maioria dos sistemas Linux, o padrão para ambos é `/usr`.

Arquivo/diretório	Significado
`exec_prefix``/bin/python`	Localização recomendada do interpretador.
`prefix``/lib/python`*`version`*, *`exec_prefix`*`/lib/python`*`version`*	A localização recomendada dos diretórios contendo os módulos padrão.
`prefix``/include/python`*`version`*, *`exec_prefix`*`/include/python`*`version`*	Localizações recomendadas dos diretórios contendo os arquivos de inclusão necessários para o desenvolvimento de extensões Python e incorporação do interpretador.
`~/.pythonrc.py`	User-specific initialization file loaded by the user module; not used by default or by most applications.

2.4 Diversos

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

```
$ chmod +x script
```

e colocar uma linha Shebang apropriada no topo do script. Uma boa escolha normalmente é

```
#!/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.

Para usar comandos Shell em seus scripts Python, veja o módulo `subprocess`.

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.

CAPÍTULO 3

Utilizando Python no Windows

Este documento pretende dar uma visão geral do comportamento específico do Windows que você deve conhecer quando fores utilizar o Python no sistema operacional Microsoft Windows.

3.1 Installing 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.

Com o desenvolvimento do Python em andamento, algumas plataformas que costumavam ser suportadas anteriormente não são mais suportadas (devido à falta de usuário ou desenvolvedores). Confira a **PEP 11** para detalhes de todas as plataformas não suportadas.

- 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 ainda é suportado.
- O instalador Cygwin também oferece instalação do interpretador do Python (cf. Cygwin package source, Maintainer releases)

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

Ver também:

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 Pacotes Alternativos

À parte da distribuição padrão CPython, existem pacotes modificados incluindo funcionalidades adicionais. A seguir está uma lista de versões populares e seus recursos chave:

ActivePython Instalador com compatibilidade multi-plataforma, documentação, 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 Configurando o Python

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

3.3.1 Excursus: Configurando variáveis de ambiente

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.

Ver também:

https://support.microsoft.com/kb/100843 Variáveis de ambiente no Windows NT

https://support.microsoft.com/kb/310519 Como Gerenciar Variáveis de Ambiente no Windows XP

https://www.chem.gla.ac.uk/~louis/software/faq/q1.html Configurando variáveis de ambiente, Louis J. Farrugia

3.3.2 Encontrando o executável do 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 *Linha de comando* documentation.

3.3.3 Encontrando módulos

O Python geralmente armazena sua biblioteca (e assim sua pasta de site-packages) no diretório de instalação. Então, se você instalou o Python em `C:\Python\`, a biblioteca padrão irá residir em `C:\Python\Lib\` e módulos de terceiros serão armazenados em `C:\Python\Lib\site-packages\`.

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

- Uma entrada em branco é adicionada ao início, que corresponde ao diretório atual.
- Se a variável de ambiente *`PYTHONPATH`* existe, como descrito em *Environment variables*, suas entradas são adicionadas em seguida. Note que no Windows, caminhos nessa variável devem ser separados por ponto e vírgula, para distinguir eles dos dois pontos usados nos identificadores de drivers (`C:\` etc.).
- "Caminhos da aplicação" adicionais podem ser adicionados ao registro como subchaves de `\SOFTWARE\Python\PythonCore{version}\PythonPath` sob ambas `HKEY_CURRENT_USER` e `HKEY_LOCAL_MACHINE`. Subchaves que possuem string de caminhos delimitados por ponto e vírgula como seu valor padrão farão com que cada caminho seja adicionado ao `sys.path`. (Note que todos os instaladores conhecidos usam apenas HKLM, portanto HKCU está tipicamente vazio.)
- 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.
- Se o Python Home não puder ser localizado, nenhum *`PYTHONPATH`* está especificado no ambiente, e nenhuma entrada de registro pôde ser encontrada, um caminho padrão com entradas relativas é usado (por exemplo, `.\Lib;.\plat-win`, etc).

O resultado final de tudo isso é:

- Quando executando `python.exe`, ou qualquer outro .exe no diretório principal do Python (ou uma versão instalada, ou diretamente do diretório PCbuild), o caminho núcleo é deduzido, e os caminhos núcleo no registro são ignorados. Outros "caminhos da aplicação" no registro são sempre lidos.
- Quando Python é hospedado em outro .exe (diretório diferente, embutido via COM, etc), o "Python Home" não será deduzido, então o caminho núcleo do registro é usado. Outros "caminhos da aplicação" no registro sempre são lidos.
- 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 Módulos adicionais

Mesmo que o Python tenha como objetivo ser portável através de todas as plataformas, existem recursos que são únicos para o Windows. Alguns módulos, em ambas as bibliotecas padrão e externa, e trechos de código existem para usar esses recursos.

Os módulos padrão específicos para Windows estão documentados em mswin-specific-services.

3.4.1 PyWin32

O módulo PyWin32 de Mark Hammond é uma coleção de módulos para suporte avançado específico para Windows. Isso inclui utilitários para:

- Component Object Model (COM)
- Chamadas à API Win32
- Registro
- Log de Eventos
- Microsoft Foundation Classes (MFC) interface de usuário

PythonWin é uma aplicação MFC de exemplo enviada com o PyWin32. É uma IDE embutível com um depurador embutido.

Ver também:

Win32 How Do I…? por Tim Golden

Python and COM by David and 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

Dado que a camada de manipulação de terminal avançada do Python, `curses`, é restrita a sistemas tipo Unix, existe uma biblioteca exclusiva para o Windows: Windows Console I/O para Python.

WConio é um encapsulador para o `CONIO.H` do Turbo-C usado para criar texto sob interfaces de usuário.

3.5 Compilando Python no 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:

Directory	MSVC version	Visual Studio version
`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.

Check `PC/readme.txt` for general information on the build process.

Para módulos de extensão, consulte building-on-windows.

Ver também:

Python + Windows + distutils + SWIG + gcc MinGW ou "Creating Python extensions in C/C++ with SWIG and compiling them with MinGW gcc under Windows" ou "Installing Python extension with distutils and without Microsoft Visual C++" por Sébastien Sauvage, 2003

MingW – Python extensions by Trent Apted et al, 2007

3.6 Other resources

Ver também:

Python Programming On Win32 "Help for Windows Programmers" by Mark Hammond and Andy Robinson, O'Reilly Media, 2000, ISBN 1-56592-621-8

A Python for Windows Tutorial by Amanda Birmingham, 2004

CAPÍTULO 4

Utilizando Python em um Mac

Autor Bob Savage <bobsavage@mac.com>

Python on a Macintosh running Mac OS X is in principle very similar to Python on any other Unix platform, but there are a number of additional features such as the IDE and the Package Manager that are worth pointing out.

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 Getting and Installing 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.

What you get after installing is a number of things:

- 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 Other Resources

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

APÊNDICE A

Glossário

>>> O prompt padrão do shell interativo do Python. Normalmente visto em exemplos de código que podem ser executados interativamente no interpretador.

... 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 Uma ferramenta que tenta converter código Python 2.x em código Python 3.x tratando a maioria das incompatibilidades que podem se detectadas com análise do código-fonte e navegação na árvore sintática.

O 2to3 está disponível na biblioteca padrão como `lib2to3`; um ponto de entrada é disponibilizado como `Tools/scripts/2to3`. Veja 2to3-reference.

classe base abstrata Abstract base classes complement *duck-typing* by providing a way to define interfaces when other techniques like `hasattr()` would be clumsy or subtly wrong (for example with magic methods). ABCs introduce virtual subclasses, which are classes that don't inherit from a class but are still recognized by `isinstance()` and `issubclass()`; see the `abc` module documentation. Python comes with many built-in ABCs for data structures (in the `collections` module), numbers (in the `numbers` module), and streams (in the `io` module). You can create your own ABCs with the `abc` module.

argumento A value passed to a *function* (or *method*) when calling the function. There are two types of arguments:

- *argumento nomeado*: um argumento precedido por um identificador (por exemplo, `nome=`) na chamada de uma função ou passada como um valor em um dicionário precedido por `**`. Por exemplo, `3` e `5` são ambos argumentos nomeados na chamada da função `complex()` a seguir:

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

- *argumento posicional*: um argumento que não é um argumento nomeado. Argumentos posicionais podem aparecer no início da lista de argumentos e/ou podem ser passados com elementos de um *iterável* precedido por `*`. Por exemplo, `3` e `5` são ambos argumentos posicionais nas chamadas a seguir:

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

Argumentos são atribuídos às variáveis locais nomeadas no corpo da função. Veja a seção calls para as regras de atribuição. Sintaticamente, qualquer expressão pode ser usada para representar um argumento; avaliada a expressão, o valor é atribuído à variável local.

See also the *parameter* glossary entry and the FAQ question on the difference between arguments and parameters.

atributo Um valor associado a um objeto que é referenciado pelo nome separado por um ponto. Por exemplo, se um objeto *o* tem um atributo *a* esse seria referenciado como *o.a*.

BDFL Benevolent Dictator For Life, a.k.a. Guido van Rossum, Python's creator.

objeto byte ou similar An object that supports the buffer protocol, like `str`, `bytearray` or `memoryview`. Bytes-like objects can be used for various operations that expect binary data, such as compression, saving to a binary file or sending over a socket. Some operations need the binary data to be mutable, in which case not all bytes-like objects can apply.

bytecode Python source code is compiled into bytecode, the internal representation of a Python program in the CPython interpreter. The bytecode is also cached in `.pyc` and `.pyo` files so that executing the same file is faster the second time (recompilation from source to bytecode can be avoided). This "intermediate language" is said to run on a *virtual machine* that executes the machine code corresponding to each bytecode. Do note that bytecodes are not expected to work between different Python virtual machines, nor to be stable between Python releases.

Uma lista de instruções bytecode pode ser encontrada na documentação para o módulo dis.

Classe Um modelo para criação de objetos definidos pelo usuário. Definições de classe normalmente contém definições de métodos que operam sobre instâncias da classe.

classic class Any class which does not inherit from `object`. See *new-style class*. Classic classes have been removed in Python 3.

coerção 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`.

número complexo Uma extensão ao familiar sistema de números reais em que todos os números são expressos como uma soma de uma parte real e uma parte imaginária. Números imaginários são múltiplos reais da unidade imaginária (a raiz quadrada de -1), normalmente escrita como `i` em matemática ou `j` em engenharia. O Python tem suporte nativo para números complexos, que são escritos com esta última notação; a parte imaginária escrita com um sufixo `j`, p.ex., `3+1j`. Para ter acesso aos equivalentes para números complexos do módulo `math`, utilize `cmath`. O uso de números complexos é uma funcionalidade matemática bastante avançada. Se você não sabe se irá precisar deles, é quase certo que você pode ignorá-los sem problemas.

gerenciador de contexto Um objeto que controla o ambiente visto numa instrução `with` por meio da definição dos métodos `__enter__()` e `__exit__()`. Veja **PEP 343**.

CPython A implementação canônica da linguagem de programação Python, como disponibilizada pelo python.org. O termo "CPython" é quando for necessário distinguir esta implementação de outras como Jython ou IronPython.

decorador Uma função que retorna outra função, geralmente aplicada como uma transformação de função usando a sintaxe `@wrapper`. Exemplos comuns para decoradores são `classmethod()` e `staticmethod()`.

A sintaxe do decorador é meramente um açúcar-sintático, as duas definições de funções a seguir são semanticamente equivalentes:

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

(continua na próxima página)

(continuação da página anterior)

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

O mesmo conceito existe para as classes, mas não é comumente utilizado. Veja a documentação de function definitions e class definitions para obter mais informações sobre decoradores.

descritor 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.

Para obter mais informações sobre os métodos dos descritores, veja: descriptors.

dicionário 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.

visualização de dicionário 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 Uma string literal que aparece como primeira expressão numa classe, função ou módulo. Ainda que sejam ignoradas quando a suíte é executada, é reconhecida pelo compilador que a coloca no atributo `__doc__` da classe, função ou módulo que a encapsula. Como ficam disponíveis por meio de introspecção, docstrings são o lugar canônico para documentação do objeto.

duck-typing (tipagem pato) Um estilo de programação que não verifica o tipo do objeto para determinar se ele possui a interface correta; em vez disso, o método ou atributo é simplesmente chamado ou utilizado ("Se se parece com um pato e grasna como um pato, então deve ser um pato.") Enfatizando interfaces ao invés de tipos específicos, o código bem desenvolvido aprimora sua flexibilidade por permitir substituição polimórfica. Tipagem pato evita necessidade de testes que usem `type()` ou `isinstance()`. (Note, porém, que a a tipagem pato pode ser complementada com o uso de *classes base abstratas*.) Ao invés disso, são normalmente empregados testes `hasattr()` ou programação *EAFP*.

EAFP Iniciais da expressão em inglês "easier to ask for forgiveness than permission" que significa "é mais fácil pedir perdão que permissão". Este estilo de codificação comum em Python assume a existência de chaves ou atributos válidos e captura exceções caso essa premissa se prove falsa. Este estilo limpo e rápido se caracteriza pela presença de várias instruções `try` e `except`. A técnica diverge do estilo *LBYL*, comum em outras linguagens como C, por exemplo.

expressão 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 *statement*s which cannot be used as expressions, such as `print` or `if`. Assignments are also statements, not expressions.

extension module (módulo de extensão) Um módulo escrito em C ou C++, usando a API C de Python para interagir tanto com código de usuário quanto do núcleo.

objeto arquivo Um objeto que expõe uma API orientada a arquivos (com métodos tais como `read()` ou `write()`) para um recurso subjacente. Dependendo da maneira como foi criado, um objeto arquivo pode mediar o acesso a um arquivo real no disco ou outro tipo de dispositivo de armazenamento ou de comunicação (por exemplo a entrada/saída padrão, buffers em memória, soquetes, pipes, etc.). Objetos arquivo também são chamados de *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.

objeto como-arquivo Um sinônimo do termo *file object*.

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

divisão pelo piso Divisão matemática que arredonda para baixo para o inteiro mais próximo. O operador de divisão pelo piso é `//`. Por exemplo, a expressão `11 // 4` retorna o valor `2` ao invés de `2.75`, que seria retornado pela divisão de ponto flutuante. Note que `(-11) // 4` é `-3` porque é `-2.75` arredondado *para baixo*. Consulte a **PEP 238**.

função Uma série de instruções que retorna algum valor para um chamador. Também pode ser passado zero ou mais *argumentos* que podem ser usados na execução do corpo. Veja também *parâmetro*, *métodos* e a seção function.

__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)
```

garbage collection (coletor de lixo) The process of freeing memory when it is not used anymore. Python performs garbage collection via reference counting and a cyclic garbage collector that is able to detect and break reference cycles.

gerador 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).

generator expression An expression that returns an iterator. It looks like a normal expression followed by a `for` expression defining a loop variable, range, and an optional `if` expression. The combined expression generates values for an enclosing function:

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

GIL Veja *global interpreter lock*.

global interpreter lock (bloqueio global do interpretador) O mecanismo utilizado pelo interpretador *CPython* para garantir que apenas uma thread execute o *bytecode* Python por vez. Isto simplifica a implementação do CPython ao fazer com que o modelo de objetos (incluindo tipos internos críticos como o `dict`) ganhem segurança implícita contra acesso concorrente. Travar todo o interpretador facilita que o interpretador em si seja multitarefa, às custas de muito do paralelismo já provido por máquinas multiprocessador.

No entanto, alguns módulos de extensão, tanto da biblioteca padrão quanto de terceiros, são desenvolvidos de forma a liberar o GIL ao realizar tarefas computacionalmente muito intensas, como compactação ou cálculos de hash. Além disso, o GIL é sempre liberado nas operações de E/S.

No passado, esforços para criar um interpretador que lidasse plenamente com threads (travando dados compartilhados numa granularidade bem mais fina) não foram bem sucedidos devido a queda no desempenho ao serem

executados em processadores de apenas um núcleo. Acredita-se que superar essa questão de desempenho acabaria tornando a implementação muito mais complicada e bem mais difícil de manter.

hasheável 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.

A hashabilidade faz com que um objeto possa ser usado como chave de um dicionário e como membro de um conjunto, pois estas estruturas de dados utilizam os valores de hash internamente.

All of Python's immutable built-in objects are hashable, while no mutable containers (such as lists or dictionaries) are. Objects which are instances of user-defined classes are hashable by default; they all compare unequal (except with themselves), and their hash value is derived from their `id()`.

IDLE Um ambiente de desenvolvimento integrado para Python. IDLE é um editor básico e um ambiente interpretador que vem junto com a distribuição padrão do Python.

imutável Um objeto que possui um valor fixo. Objetos imutáveis incluem números, strings e tuplas. Estes objetos não podem ser alterados. Um novo objeto deve ser criado se um valor diferente tiver de ser armazenado. Objetos imutáveis têm um papel importante em lugares onde um valor constante de hash seja necessário, como por exemplo uma chave em um dicionário.

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__*.

importing (importando) O processo pelo qual o código Python em um módulo é disponibilizado para o código Python em outro módulo.

importer Um objeto que localiza e carrega um módulo; Tanto um *finder* e o objeto *loader*.

interactive Python tem um interpretador interativo, o que significa que você pode digitar comandos e expressões no prompt do interpretador, executá-los imediatamente e ver seus resultados. Apenas execute `python` sem argumentos (possivelmente selecionando-o a partir do menu de aplicações de seu sistema operacional). O interpretador interativo é uma maneira poderosa de testar novas ideias ou aprender mais sobre módulos e pacotes (lembre-se do comando `help(x)`).

interpreted Python é uma linguagem interpretada, em oposição àquelas que são compiladas, embora esta distinção possa ser nebulosa devido à presença do compilador de bytecode. Isto significa que os arquivos-fontes podem ser executados diretamente sem necessidade explícita de se criar um arquivo executável. Linguagens interpretadas normalmente têm um ciclo de desenvolvimento/depuração mais curto que as linguagens compiladas, apesar de seus programas geralmente serem executados mais lentamente. Veja também *interativo*.

iterável 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*.

iterador 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.

Mais informações podem ser encontradas em typeiter.

key function (função chave) Uma função chave ou função colação é algo que retorna um valor utilizado para ordenação ou classificação. Por exemplo, `locale.strxfrm()` é usada para produzir uma chave de ordenação que leva o locale em consideração para fins de ordenação.

A number of tools in Python accept key functions to control how elements are ordered or grouped. They include `min()`, `max()`, `sorted()`, `list.sort()`, `heapq.nsmallest()`, `heapq.nlargest()`, and `itertools.groupby()`.

There are several ways to create a key function. For example. the `str.lower()` method can serve as a key function for case insensitive sorts. Alternatively, an ad-hoc key function can be built from a `lambda` expression such as `lambda r: (r[0], r[2])`. Also, the `operator` module provides three key function constructors: `attrgetter()`, `itemgetter()`, and `methodcaller()`. See the Sorting HOW TO for examples of how to create and use key functions.

argumento nomeado Veja o *argument*.

lambda Uma função de linha anônima consistindo de uma única *expression*, que é avaliada quando a função é chamada. A sintaxe para criar uma função lambda é `lambda [parameters]: expression`

LBYL Iniciais da expressão em inglês "look before you leap", que significa algo como "olhe antes de pisar". Este estilo de codificação testa as pré-condições explicitamente antes de fazer chamadas ou buscas. Este estilo contrasta com a abordagem *EAFP* e é caracterizada pela presença de muitos comandos `if`.

Em um ambiente multithread, a abordagem LBYL pode arriscar a introdução de uma condição de corrida entre "o olhar" e "o pisar". Por exemplo, o código `if key in mapping: return mapping[key]` pode falhar se outra thread remover *key* do *mapping* após o teste, mas antes da olhada. Esse problema pode ser resolvido com bloqueios ou usando a abordagem EAFP.

lista Uma *sequence* embutida no Python. Apesar do seu nome, é mais próximo de um vetor em outras linguagens do que uma lista encadeada, como o acesso aos elementos é da ordem O(1).

list comprehension 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.

loader 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.

método mágico Um sinônimo informal para um *special method*.

mapeamento A container object that supports arbitrary key lookups and implements the methods specified in the `Mapping` or `MutableMapping` abstract base classes. Examples include `dict`, `collections.defaultdict`, `collections.OrderedDict` and `collections.Counter`.

metaclass A classe de uma classe. Definições de classe criam um nome de classe, um dicionário de classe e uma lista de classes base. A metaclasse é responsável por receber estes três argumentos e criar a classe. A maioria das linguagens de programação orientadas a objetos provê uma implementação default. O que torna o Python especial é o fato de ser possível criar metaclasses personalizadas. A maioria dos usuários nunca vai precisar deste recurso, mas quando houver necessidade, metaclasses possibilitam soluções poderosas e elegantes. Metaclasses têm sido utilizadas para gerar registros de acesso a atributos, para incluir proteção contra acesso concorrente, rastrear a criação de objetos, implementar singletons, dentre muitas outras tarefas.

More information can be found in metaclasses.

method (método) Uma função que é definida dentro do corpo de uma classe. Se chamada como um atributo de uma instância daquela classe, o método receberá a instância do objeto como seu primeiro *argumento* (que comumente é chamado de `self`). Veja *função* e *nested scope*.

method resolution order (ordem de resolução de método) Ordem de resolução de métodos é a ordem em que os membros de uma classe base são buscados durante a pesquisa. Veja A ordem de resolução de métodos do Python 2.3.

módulo Um objeto que serve como uma unidade organizacional de código Python. Os módulos têm um namespace contendo objetos Python arbitrários. Os módulos são carregados pelo Python através do processo de *importação*.

Veja também *pacote*.

MRO Veja *method resolution order*.

mutável Objeto mutável é aquele que pode modificar seus valor mas manter seu `id()`. Veja também *immutable*.

tupla nomeada Any tuple-like class whose indexable elements are also accessible using named attributes (for example, `time.localtime()` returns a tuple-like object where the *year* is accessible either with an index such as `t[0]` or with a named attribute like `t.tm_year`).

A named tuple can be a built-in type such as `time.struct_time`, or it can be created with a regular class definition. A full featured named tuple can also be created with the factory function `collections.namedtuple()`. The latter approach automatically provides extra features such as a self-documenting representation like `Employee(name='jones', title='programmer')`.

namespace 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.

nested scope (escopo aninhado) 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.

new-style class (novo estilo de classes) 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 `__getattribute__()`.

More information can be found in newstyle.

object (objeto) Qualquer dado que tenha estado (atributos ou valores) e comportamento definidos (métodos). Também a última classe base de qualquer *new-style class*.

pacote Um *module* Python é capaz de conter submódulos ou recursivamente, sub-pacotes. Tecnicamente, um pacote é um módulo Python com um atributo `__path__`.

parâmetro 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:

- *posicional-ou-nomeado*: especifica um argumento que pode ser tanto *posicional* quanto *nomeado*. Esse é o tipo padrão de parâmetro, por exemplo *foo* e *bar* a seguir:

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

- *positional-only*: specifies an argument that can be supplied only by position. Python has no syntax for defining positional-only parameters. However, some built-in functions have positional-only parameters (e.g. `abs()`).

- *var-posicional*: especifica quem uma sequência arbitrária de argumentos posicionais pode ser fornecida (em adição a qualquer argumento posicional já aceito por outros parâmetros). Tal parâmetro pode ser definido colocando um `*` antes do nome, por exemplo *args* a seguir:

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

- *var-nomeado*: especifica que, arbitrariamente, muitos argumentos nomeados podem ser fornecidos (em adição a qualquer argumento nomeado já aceito por outros parâmetros). Tal parâmetro pode definido colocando-se `**` antes do nome, por exemplo *kwargs* no exemplo acima.

Parâmetros podem especificar tanto argumentos opcionais quanto obrigatórios, assim como valores padrões para alguns argumentos opcionais.

See also the *argument* glossary entry, the FAQ question on the difference between arguments and parameters, and the function section.

PEP Proposta de melhoria do Python. Uma PEP é um documento de design que fornece informação para a comunidade Python, ou descreve uma nova funcionalidade para o Python ou seus predecessores ou ambientes. PEPs devem prover uma especificação técnica concisa e um racional para funcionalidades propostas.

PEPs tem a intenção de ser os mecanismos primários para propor novas funcionalidades significativas, para coletar opiniões da comunidade sobre um problema, e para documentar as decisões de design que foram adicionadas ao Python. O autor da PEP é responsável por construir um consenso dentro da comunidade e documentar opiniões dissidentes.

Veja **PEP 1**.

positional argument (argumento posicional) Veja o *argument*.

Python 3000 Apelido para a versão do Python 3.x linha de lançamento (cunhado há muito tempo, quando o lançamento da versão 3 era algo em um futuro muito distante.) Esse termo possui a seguinte abreviação: “Py3k”.

Pythonic Uma ideia ou um pedaço de código que segue de perto os idiomas mais comuns da linguagem Python, ao invés de implementar códigos usando conceitos comuns a outros idiomas. Por exemplo, um idioma comum em Python é fazer um loop sobre todos os elementos de uma iterável usando a instrução `for`. Muitas outras linguagens não têm esse tipo de construção, então as pessoas que não estão familiarizadas com o Python usam um contador numérico:

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

Ao contrário do método limpo, ou então, Pythônico:

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

reference count O número de referências para um objeto. Quando a contagem de referências de um objeto atinge zero, ele é desalocado. Contagem de referências geralmente não é visível no código Python, mas é um elemento chave da implementação *CPython*. O módulo `sys` define a função `getrefcount()` que programadores podem chamar para retornar a contagem de referências para um objeto em particular.

__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.

sequência 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.

fatia 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__()`).

método especial Um método que é chamado implicitamente pelo Python para executar uma certa operação em um tipo, como uma adição por exemplo. Tais métodos tem nomes iniciando e terminando com dois underscores. Métodos especiais estão documentados em specialnames.

instrução Uma instrução é parte de uma suíte (um "bloco" de código). Uma instrução é ou uma *expression* ou uma de várias construções com uma palavra-chave, tal como `if`, `while` ou `for`.

struct sequence A tuple with named elements. Struct sequences expose an interface similiar 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()`.

aspas triplas Uma string que está definida com três ocorrências de aspas duplas (") ou apóstrofes ('). Enquanto elas não fornecem nenhuma funcionalidade não disponível com strings de aspas simples, elas são úteis para inúmeras razões. Elas permitem que você inclua aspas simples e duplas não encerradas dentro de uma string, e elas podem utilizar múltiplas linhas sem o uso de caractere de continuação, fazendo-as especialmente úteis quando escrevemos documentação em docstrings.

type O tipo de um objeto Python determina qual tipo de objeto ele é; cada objeto tem um tipo. Um tipo de objeto é acessível pelo atributo `__class__` ou pode ser recuperado com `type(obj)`.

Novas linhas universais 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.

ambiente virtual Um ambiente de execução isolado que permite usuários Python e aplicações instalarem e atualizarem pacotes Python sem interferir no comportamento de outras aplicações Python em execução no mesmo sistema.

máquina virtual Um computador definido inteiramente em software. A máquina virtual de Python executa o *bytecode* emitido pelo compilador de bytecode.

Zen of Python Lista de princípios de projeto e filosofias do Python que são úteis para a compreensão e uso da linguagem. A lista é exibida quando se digita "`import this`" no console interativo.

APÊNDICE B

Sobre esses documentos

Esses documentos são gerados a partir de reStructuredText pelo Sphinx, um processador de documentos especificamente escrito para documentação do Python.

O desenvolvimento da documentação e de suas ferramentas é um esforço totalmente voluntário, como o Python em si. Se você quer contribuir, por favor dê uma olhada na página reporting-bugs para informações sobre como fazer. Novos voluntários são sempre bem vindos!

Agradecimentos especiais para:

- Fred L. Drake, Jr., o criador do primeiro conjunto de ferramentas para documentar o Python e escritor de boa parte do conteúdo;
- O projeto Docutils para criar reStructuredText e o pacote Docutils;
- Fredrik Lundh por seu projeto Referência Alternativa para Python do qual Sphinx teve muitas ideias boas.

B.1 Contribuidores da Documentação do Python

Muitas pessoas tem contribuído para a linguagem Python, sua biblioteca padrão e sua documentação. Veja Misc/ACKS na distribuição do código do Python para ver uma lista parcial de contribuidores.

Tudo isso só foi possível com o esforço e a contribuição da comunidade Python, por isso temos essa maravilhosa documentação – Obrigado a todos!

APÊNDICE C

História e Licença

C.1 História do software

O Python foi criado no início dos anos 1990 por Guido van Rossum na Stichting Mathematisch Centrum (CWI, veja https://www.cwi.nl/) na Holanda como um sucessor de uma linguagem chamada ABC. Guido continua a ser o principal autor de Python, embora inclua muitas contribuições de outros.

Em 1995, Guido continuou seu trabalho em Python na Corporação para Iniciativas Nacionais de Pesquisa (CNRI, veja https://www.cnri.reston.va.us/) em Reston, Virgínia, onde lançou várias versões do software.

Em maio de 2000, Guido e a equipe principal de desenvolvimento do Python mudaram-se para o BeOpen.com para formar a equipe BeOpen PythonLabs. Em outubro do mesmo ano, a equipe da PythonLabs mudou para a Digital Creations (agora Zope Corporation; veja https://www.zope.org/). Em 2001, formou-se a Python Software Foundation (PSF, ver https://www.python.org/psf/), uma organização sem fins lucrativos criada especificamente para possuir propriedade intelectual relacionada a Python. A Zope Corporation é um membro patrocinador do PSF.

Todas as versões do Python são de código aberto (consulte https://opensource.org/ para a definição de código aberto). Historicamente, a maioria, mas não todas, versões do Python também são compatíveis com GPL; a tabela abaixo resume os vários lançamentos.

Release	Derivado de	Ano	Proprietário	GPL compatível?
0.9.0 a 1.2	n/a	1991-1995	CWI	sim
1.3 a 1.5.2	1.2	1995-1999	CNRI	sim
1.6	1.5.2	2000	CNRI	não
2.0	1.6	2000	BeOpen.com	não
1.6.1	1.6	2001	CNRI	não
2.1	2.0+1.6.1	2001	PSF	não
2.0.1	2.0+1.6.1	2001	PSF	sim
2.1.1	2.1+2.0.1	2001	PSF	sim
2.1.2	2.1.1	2002	PSF	sim
2.1.3	2.1.2	2002	PSF	sim
2.2 e acima	2.1.1	2001-agora	PSF	sim

Nota: Compatível com GPL não significa que estamos distribuindo Python sob a GPL. Todas as licenças do Python, ao contrário da GPL, permitem distribuir uma versão modificada sem fazer alterações em código aberto. As licenças compatíveis com GPL possibilitam combinar o Python com outro software lançado sob a GPL; os outros não.

Graças aos muitos voluntários externos que trabalharam sob a direção de Guido para tornar esses lançamentos possíveis.

C.2 Termos e condições para acessar ou usar Python

C.2.1 PSF LICENSE AGREEMENT FOR PYTHON 2.7.18

```
1. This LICENSE AGREEMENT is between the Python Software Foundation ("PSF"),␣
↪and
   the Individual or Organization ("Licensee") accessing and otherwise using␣
↪Python
   2.7.18 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 2.7.18 alone or in any derivative
   version, provided, however, that PSF's License Agreement and PSF's notice␣
↪of
   copyright, i.e., "Copyright © 2001-2020 Python Software Foundation; All␣
↪Rights
   Reserved" are retained in Python 2.7.18 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 2.7.18 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
   2.7.18.

4. PSF is making Python 2.7.18 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 2.7.18 WILL NOT INFRINGE ANY THIRD PARTY RIGHTS.

5. PSF SHALL NOT BE LIABLE TO LICENSEE OR ANY OTHER USERS OF PYTHON 2.7.18
   FOR ANY INCIDENTAL, SPECIAL, OR CONSEQUENTIAL DAMAGES OR LOSS AS A RESULT␣
↪OF
   MODIFYING, DISTRIBUTING, OR OTHERWISE USING PYTHON 2.7.18, 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 2.7.18, Licensee agrees
   to be bound by the terms and conditions of this License Agreement.
```

C.2.2 ACORDO DE LICENÇA DA BEOPEN.COM PARA PYTHON 2.0

CONTRATO DE LICENÇA DE FONTE ABERTA DO BEOPEN PYTHON VERSÃO 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
```

(continua na próxima página)

(continuação da página anterior)

```
   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 CONTRATO DE LICENÇA DA CNRI PARA O 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,
   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
```

(continua na próxima página)

(continuação da página anterior)

```
   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 ACORDO DE LICENÇA DA CWI PARA PYTHON 0.9.0 A 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 Licenças e Reconhecimentos para Software Incorporado

Esta seção é uma lista incompleta, mas crescente, de licenças e confirmações para softwares de terceiros incorporados na distribuição do Python.

C.3.1 Mersenne Twister

O módulo `_random` inclui código baseado em um download de http://www.math.sci.hiroshima-u.ac.jp/~m-mat/MT/MT2002/emt19937ar.html. A seguir estão os comentários literais do código 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).
```

(continua na próxima página)

(continuação da página anterior)

```
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.

 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 Sockets

O módulo `socket` usa as funções `getaddrinfo()` e `getnameinfo()`, que são codificadas em arquivos de origem separados do Projeto 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.
```

(continua na próxima página)

(continuação da página anterior)

```
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 Floating point exception control

The source for the `fpectl` module includes the following notice:

```
  ---------------------------------------------------------------------
 /                      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:

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 Serviços de soquete assíncrono

Os módulos `asynchat` e `asyncore` contêm o seguinte aviso:

```
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,
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 Gerenciamento de cookies

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 Rastreamento de execução

O módulo `trace` contém o seguinte aviso:

```
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

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 Funções UUencode e UUdecode

O módulo `uu` contém o seguinte aviso:

```
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
```

(continua na próxima página)

(continuação da página anterior)

```
  version is still 5 times faster, though.
- Arguments more compliant with Python standard
```

C.3.9 Chamadas de Procedimento Remoto XML

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:

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 MERCHANT-
ABILITY 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

The `test_epoll` contains the following notice:

```
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
```

(continua na próxima página)

(continuação da página anterior)

```
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 Selecione o kqueue

The `select` and contains the following notice for the kqueue interface:

```
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
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 e dtoa

O arquivo `Python/dtoa.c`, que fornece as funções C dtoa e strtod para conversão de duplas de C para e de strings, é derivado do arquivo com o mesmo nome de David M. Gay, atualmente disponível em http://www.netlib.org/fp/. O arquivo original, conforme recuperado em 16 de março de 2009, contém os seguintes avisos de direitos autorais e de licenciamento:

```
/****************************************************************
 *
 * 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.
 *
```

(continua na próxima página)

(continuação da página anterior)

```
 * 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

Os módulos `hashlib`, `posix`, `ssl`, `crypt` usam a biblioteca OpenSSL para desempenho adicional se forem disponibilizados pelo sistema operacional. Além disso, os instaladores do Windows e do Mac OS X para Python podem incluir uma cópia das bibliotecas do OpenSSL, portanto incluímos uma cópia da licença do OpenSSL aqui:

```
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
```

(continua na próxima página)

(continuação da página anterior)

```
   *    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
   * 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
```

(continua na próxima página)

(continuação da página anterior)

```
 *    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 :-).
 * 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

A extensão `pyexpat` é construída usando uma cópia incluída das fontes de expatriadas, a menos que a compilação esteja configurada `--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

A extensão `_ctypes` é construída usando uma cópia incluída das fontes libffi, a menos que a compilação esteja configurada `--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

A extensão `zlib` é construída usando uma cópia incluída das fontes zlib se a versão do zlib encontrada no sistema for muito antiga para ser usada na construção:

```
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        Mark Adler
jloup@gzip.org          madler@alumni.caltech.edu
```

APÊNDICE D

Copyright

Python e essa documentação é:

Veja: *História e Licença* para informações completas de licença e permissões.

Índice

Não alfabético

A

B

C

D

E

Q

R

S

T

U

V

W

X

Z