Python Setup and Usage

Release 3.2.2

Guido van Rossum
Fred L. Drake, Jr., editor

August 02, 2015

Python Software Foundation
Email: docs@python.org

Contents

[U+6587][U+6863][U+7684][U+8FD9][U+4E00][U+90E8][U+5206][U+81F4][U+529B][U+4E8E][U+5728]
Python [U+73AF][U+5883][U+7684][U+5E38][U+89C4][U+4FE1][U+606F],
[U+89E3][U+91CA][U+5668][U+7684][U+8C03][U+7528][U+65B9][U+6CD5][U+4EE5][U+53CA][U+4E00]
Python [U+5DE5][U+4F5C][U+5F97][U+66F4][U+5BB9][U+6613][U+7684][U+4E8B][U+60C5].

CHAPTER 1

Command line and environment

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 Command line

When invoking Python, you may specify any of these options:

```
python [-bBdEhiOsSuvVWx?] [-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 Interface options

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.
- When called with a file name argument or with a file as standard input, it reads and executes a script from that file.
- 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.

Note: 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 (while the module file is being located, the first element will be set to "-m"). 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
```

See also:

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

PEP 338 – Executing modules as scripts

Changed in version 3.1: Supply the package name to run a __main__ submodule.

-
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.

<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.

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.

See also:

tut-invoking

1.1.2 Generic options

-?
-h

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

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

```
Python 3.0
```

1.1.3 Miscellaneous options

-b
Issue a warning when comparing str and bytes. Issue an error when the option is given twice (`-bb`).

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

-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.

-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
Don't display the copyright and version messages even in interactive mode.

New in version 3.2.

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

See also:

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.

-u
Force the binary layer of the stdin, stdout and stderr streams (which is available as their `buffer` attribute) to be unbuffered. The text I/O layer will still be line-buffered.

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).

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):

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.

See also:

`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.

Note: The line numbers in error messages will be off by one.

-X
Reserved for various implementation-specific options. CPython currently defines none of them, but allows to pass arbitrary values and retrieve them through the `sys._xoptions` dictionary.

Changed in version 3.2: It is now allowed to pass *-X* with CPython.

1.1.4 Options you shouldn't use

-J
Reserved for use by Jython.

1.2 Environment variables

These environment variables influence Python's behavior.

PYTHONHOME
Change the location of the standard Python libraries. By default, the libraries are searched in *prefix*/lib/python*version* and *exec_prefix*/lib/python*version*, 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 *Interface options*. 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.

PYTHONDONTWRITEBYTECODE
If this is set, Python won't try to write `.pyc` or `.pyo` files on the import of source modules.

PYTHONIOENCODING
If this is set before running the interpreter, it 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()`.

For stderr, the `:errorhandler` part is ignored; the handler will always be `'backslashreplace'`.

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

See also:

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`.

See also:

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.

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.

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.

CHAPTER 2

[U+5728] Unix [U+5E73] [U+53F0] [U+4E0A] [U+4F7F] [U+7528] Python

2.1 [U+83B7] [U+53D6] [U+548C] [U+5B89] [U+88C5] Python [U+7684] [U+6700] [U+65B0] [U+7248] [U+672C]

2.1.1 [U+5728] Linux [U+4E0A]

[U+5728] [U+5927] [U+591A] [U+6570] Linux [U+53D1] [U+884C] [U+7248] [U+4E0A] [U+9884] [U+88C5] [U+
Python, [U+5176] [U+5B83] [U+6CA1] [U+6709] [U+9884] [U+88C5] [U+7684] [U+53D1] [U+884C] [U+7248] [U
Python. [U+7136] [U+800C], [U+5728] [U+4F60] [U+4F7F] [U+7528] [U+7684] [U+53D1] [U+884C] [U+7248] [
Python [U+5305] [U+4E0D] [U+5305] [U+542B] [U+67D0] [U+4E9B] [U+4F60] [U+60F3] [U+4F7F] [U+7528] [U+
[U+4F60] [U+53EF] [U+4EE5] [U+8F7B] [U+6613] [U+5730] [U+4ECE] [U+6E90] [U+4EE3] [U+7801] [U+91CC]
Python.

[U+5728] [U+6CA1] [U+6709] [U+9884] [U+88C5] Python, [U+8F6F] [U+4EF6] [U+4ED3] [U+5E93] [U+91CC] [U
Python [U+7684] [U+60C5] [U+51B5] [U+4E0B], [U+4F60] [U+53EF] [U+4EE5] [U+8F7B] [U+6613] [U+5730] [U

See also:

http://www.linux.com/articles/60383 for Debian users

http://linuxmafia.com/pub/linux/suse-linux-internals/chapter35.html for OpenSuse users

http://docs.fedoraproject.org/drafts/rpm-guide-en/ch-creating-rpms.html for Fedora users

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

2.1.2 [U+5728] FreeBSD [U+548C] OpenBSD [U+4E0A]

- FreeBSD [U+7528] [U+6237], [U+4F7F] [U+7528]:

  ```
  pkg_add -r python
  ```

  [U+6DFB] [U+52A0] Python [U+5305].

- OpenBSD [U+7528] [U+6237] [U+4F7F] [U+7528]:

  ```
  pkg_add ftp://ftp.openbsd.org/pub/OpenBSD/4.2/packages/<insert your architecture her
  ```

  [U+4F8B] [U+5982] i386 [U+7528] [U+6237] [U+8981] [U+83B7] [U+5F97] Python [U+7684] 2.5.1 [U+7248] [U+672C], [U+53EF] [U+4EE5] [U+4F7F] [U+7528]:

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

2.1.3 [U+5728] OpenSolaris [U+4E0A]

[U+8981][U+5728] OpenSolaris [U+4E0A][U+5B89][U+88C5][U+6700][U+65B0][U+7248][U+672C][U+7684] Python, [U+5148][U+5B89][U+88C5] blastwave (http://www.blastwave.org/howto.html), [U+518D][U+5728][U+547D][U+4EE4][U+63D0][U+793A][U+91CC][U+952E][U+5165] "pkg_get -i python".

2.2 [U+5EFA][U+9020] Python

[U+5982][U+679C][U+4F60][U+60F3][U+81EA][U+5DF1][U+7F16][U+8BD1] CPython, [U+9996][U+5148][U+4F60][U+8981][U+83B7][U+53D6]'[U+6E90][U+4EE3][U+7801] <http://python.org/download/source/>'_. [U+4F60][U+53EF][U+4EE5][U+4E0B][U+8F7D][U+6700][U+65B0][U+7 [U+4E5F][U+53EF][U+4EE5][U+6293][U+53D6][U+4E00][U+4E2A][U+65B0][U+9C9C][U+7684] checkout.

[U+5EFA][U+9020][U+6D41][U+7A0B][U+901A][U+5E38][U+662F][U+8C03][U+7528][U+4EE5][U+4E0B]

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

[U+7279][U+6B8A] Unix [U+5E73][U+53F0][U+7684][U+914D][U+7F6E][U+9009][U+9879][U+548C][U+6 README [U+6587][U+4EF6][U+91CC], [U+8BE5][U+6587][U+4EF6][U+4F4D][U+4E8E] Python [U+6E90][U+4EE3][U+7801][U+6811][U+7684][U+6839][U+76EE][U+5F55][U+4E0B].

Warning: make install [U+53EF][U+80FD][U+4F1A][U+5BF9] python [U+4E8C][U+8FDB][U+5236][U+6587][U+4EF6][U+91CD][U+5199][U+6216][U+5316][U+5986]. [U+56E0][U+6B64][U+63A8][U+8350][U+4F7F][U+7528] make altinstall [U+66FF][U+4EE3] make install, [U+56E0][U+4E3A][U+524D][U+8005][U+53EA][U+4F1A][U+5B89][U *exec_prefix*/bin/python*version*.

2.3 [U+4E0E] Python [U+76F8][U+5173][U+7684][U+8DEF][U+5F84][U+

[U+8FD9][U+4E9B][U+662F][U+5173][U+4E8E][U+57FA][U+4E8E][U+672C][U+5730][U+5B89][U+88C5] :envar:'prefix' (${prefix}) [U+548C] exec_prefix (${exec_prefix}) [U+90FD][U+662F][U+5B89][U+88C5][U+4F9D][U+8D56], [U+5E76][U+5C06][U+5E94][U+5F53][U+88AB GNU [U+8F6F][U+4EF6]; [U+5B83][U+4EEC][U+53EF][U+80FD][U+76F8][U+540C].

[U+4F8B][U+5982], [U+5728][U+5927][U+591A][U+6570] Linux [U+7CFB][U+7EDF][U+4E0A], [U+4E24][U+8005][U+7684][U+9ED8][U+8BA4][U+503C][U+90FD][U+4E3A /usr.

+	[U+6587][U+4EF6]/[U+76EE][U+5F55]	[U+542B][U+4E49]

exec_prefix/bin/python	[U+89E3][U+91CA][U+5668][U+7684][U+63A8][U+8350][U+4F4D][U+7F6E]
prefix/lib/python*version*,	[U+5305][U+542B][U+6807][U+51C6][U+6A21][U+5757][U+7684][U+76E
exec_prefix/lib/python*version*	
prefix/include/python*version*,	[U+5305][U+542B][U+5728][U+5F00][U+53D1] Python [U+6269][U+5C55][U+53CA][U+5D4C][U+5165][U
exec_prefix/include/python*version*	include [U+6587][U+4EF6][U+7684][U+76EE][U+5F55][U+7684]
~/.pythonrc.py	[U+7279][U+5B9A][U+7528][U+6237][U+7684][U+521D][U+59CB][U+5316][U+6587][U+4EF6], [U+901A][U+8FC7][U+7528][U+6237][U+6A21][U+5757][U+8F7D][U+5165];
	[U+4E0D][U+7528][U+4E8E][U+9ED8][U+8BA4][U+53CA][U+5927][U+591A][U+6570][U+5E94][U+7528]

2.4 [U+6742][U+9879]

[U+8981][U+66F4][U+5BB9][U+6613][U+7684][U+5728] Unix [U+4E0A][U+4F7F][U+7528] Python [U+811A][U+672C], [U+4F60][U+9700][U+8981][U+4F7F][U+5B83][U+4EEC][U+53EF][U+6267][U+4F8B][U+5982][U+901A][U+8FC7]

```
$ chmod +x script
```

[U+5E76][U+4E14][U+5728][U+811A][U+672C][U+7684][U+5F00][U+5934][U+6DFB][U+52A0][U+4E00] Shebang [U+884C]. [U+4E00][U+4E2A][U+597D][U+7684][U+9009][U+62E9][U+662F]

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

[U+8FD9][U+4F1A][U+5728][U+6574][U+4E2A] **:envar:'PATH'** [U+91CC][U+641C][U+7D22] Python [U+89E3][U+91CA][U+5668]. [U+7136][U+800C], [U+6709][U+4E9B] Unix [U+7CFB][U+7EDF][U+6CA1][U+6709] **env** [U+547D][U+4EE4], [U+56E0][U+6B64][U+4F60][U+53EF][U+80FD][U+9700][U+8981][U+786C][U+7F16][U+7801] /usr/bin/python [U+4F5C][U+4E3A][U+89E3][U+91CA][U+5668][U+8DEF][U+5F84].

[U+8981][U+5728] Python [U+811A][U+672C][U+91CC][U+4F7F][U+7528] shell [U+547D][U+4EE4], [U+53C2][U+89C1] subprocess' [U+6A21][U+5757].

2.5 [U+7F16][U+8F91][U+5668]

Vim [U+548C] Emacs [U+90FD][U+662F][U+4F18][U+79C0][U+7684][U+7F16][U+8F91][U+5668], [U+5B83][U+4EEC][U+90FD][U+80FD][U+5F88][U+597D][U+7684][U+652F][U+6301] Python. [U+60F3][U+8981][U+66F4][U+591A][U+5982][U+4F55][U+5728][U+8FD9][U+4E24][U+6B3E][U+4EE3][U+7801][U+7684][U+4FE1][U+606F], [U+53C2][U+89C1]:

- http://www.vim.org/scripts/script.php?script_id=790
- http://sourceforge.net/projects/python-mode

Geany [U+662F][U+4E2A][U+4F18][U+79C0][U+7684] IDE, [U+5B83][U+652F][U+6301][U+5F88][U+591A][U+60F3][U+8981][U+66F4][U+591A][U+4FE1][U+606F], [U+9605][U+8BFB]: http://geany.uvena.de/

Komodo edit [U+662F][U+53E6][U+4E00][U+4E2A][U+6781][U+597D][U+7684] IDE. [U+5B83][U+540C][U+6837][U+652F][U+6301][U+5F88][U+591A][U+539F][U+56E0][U+60F3][U+8981][U+66F4][U+591A][U+4FE1][U+606F], [U+9605][U+8BFB]: http://www.activestate.com/store/productdetail.aspx?prdGuid=20f4ed15-6684-4118-a78b-d37ff4058c5f

CHAPTER 3

[U+5728] Windows [U+4E0A][U+4F7F][U+7528] Python

[U+8FD9][U+4EFD][U+6587][U+6863][U+65E8][U+5728][U+7ED9][U+51FA][U+4E00][U+4EFD] Python [U+5728] Windows [U+7279][U+5B9A][U+7684][U+884C][U+4E3A][U+7684][U+6982][U+8981], [U+8FD9][U+4E9B][U+662F][U+4F60][U+5728] Microsoft Windows [U+4E0A][U+4F7F][U+7528] Python [U+5E94][U+8BE5][U+77E5][U+9053][U+7684].

3.1 [U+5B89][U+88C5] Python

[U+4E0D][U+50CF][U+5927][U+591A] Unix [U+7CFB][U+7EDF][U+548C][U+670D][U+52A1], Windows [U+4E0D][U+539F][U+751F][U+7684][U+9700][U+6C42] Python, [U+56E0][U+6B64][U+4E0D][U+4F1A][U+9884][U+5148][U+5B89][U+88C5] Python [U+7684][U+4E00][U+4E2A][U+7248][U+672C]. [U+7136][U+800C], [U+591A][U+5E74][U+4EE5][U+6765], CPython [U+56E2][U+961F][U+4E3A][U+6BCF][U+4E2A]‘[U+53D1] <http://www.python.org/download/releases/>‘_ [U+7F16][U+8BD1][U+4E86] Windows [U+5B89][U+88C5][U+7A0B][U+5E8F] (MSI [U+5305]).

Check **PEP 11** for details on all unsupported platforms. [U+968F][U+7740] Python [U+6301][U+7EED][U+5730][U+5F00][U+53D1], [U+6709][U+4E9B][U+5E73][U+53F0][U+66FE][U+88AB] ([U+56E0][U+4E3A][U+7F3A][U+5C11][U+7528][U+6237][U+6216][U+5F00][U+53D1][U+8005]). [U+68C0][U+67E5] **PEP 11** [U+83B7][U+5F97][U+6240][U+6709][U+4E0D][U+88AB][U+652F][U+6301][U

- [U+76F4][U+5230] 2.5, Python [U+8FD8][U+53EF][U+4EE5][U+517C][U+5BB9] Windows 95, 98 [U+548C] ME ([U+4F46][U+5728][U+5B89][U+88C5][U+65F6][U+4F1A][U+629B][U+51FA][U+ [U+5728] Python 2.6 ([U+4EE5][U+53CA][U+6240][U+6709][U+968F][U+540E][U+7248][U+672C])[U+ [U+653E][U+5F03][U+4E86][U+5BF9][U+5B83][U+4EEC][U+7684][U+652F][U+6301], [U+65B0][U+7248][U+672C][U+53EA][U+88AB][U+9884][U+671F][U+5728] Windows NT [U+5BB6][U+65CF][U+91CC][U+5DE5][U+4F5C].
- Windows CE [U+4F9D][U+65E7][U+88AB][U+652F][U+6301].
- Cygwin [U+5B89][U+88C5][U+7A0B][U+5E8F][U+540C][U+6837][U+63D0][U+4F9B] Python [U+89E3][U+91CA][U+5668]; [U+5B83][U+4F4D][U+4E8E] “Interpreters” [U+4E0B]. (cf. Cygwin [U+5305][U+6E90], Maintainer releases)

[U+53C2][U+9605] Python for Windows ([U+548C] DOS) [U+83B7][U+53D6][U+6709][U+5173][U+9884][U+7

See also:

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 [U+66FF][U+4EE3][U+8F6F][U+4EF6][U+96C6]

[U+9664][U+4E86][U+6807][U+51C6] CPython [U+53D1][U+884C][U+7248],
[U+8FD8][U+6709][U+5305][U+542B][U+9644][U+52A0][U+529F][U+80FD][U+7684][U+4FEE][U+6539]
[U+4EE5][U+4E0B][U+662F][U+4E00][U+4E9B][U+6D41][U+884C][U+7684][U+7248][U+672C][U+4EE5]

ActivePython [U+517C][U+5BB9][U+591A][U+5E73][U+53F0][U+7684][U+5B89][U+88C5][U+5305],
[U+6587][U+6863], PyWin32

Enthought Python Distribution [U+5E26][U+6709][U+5404][U+81EA][U+6587][U+6863][U+7684][U+6D41]
([U+5982] PyWin32), [U+5EFA][U+9020][U+53EF][U+6269][U+5C55] Python
[U+5E94][U+7528][U+7684][U+5DE5][U+5177][U+5305]

[U+6CE8][U+610F], [U+8FD9][U+4E9B][U+5305][U+6709][U+53EF][U+80FD][U+5B89][U+88C5]
Python [U+7684]*[U+8001]*[U+7248][U+672C].

3.3 [U+914D][U+7F6E] Python

[U+4E3A][U+4E86][U+5B8C][U+7F8E][U+7684][U+8FD0][U+884C]
Python, [U+4F60][U+53EF][U+80FD][U+8981][U+4FEE][U+6539] Windows
[U+5F53][U+524D][U+7684][U+73AF][U+5883][U+8BBE][U+7F6E].

3.3.1 [U+9644][U+5F55]: [U+8BBE][U+7F6E][U+73AF][U+5883][U+53D8][U+91CF]

Windows [U+6709][U+4E00][U+4E2A][U+5185][U+5EFA][U+7684][U+5BF9][U+8BDD][U+6846][U+6765][
([U+4E0B][U+9762][U+6559][U+7A0B][U+4F7F][U+7528][U+4E0E] XP
[U+7ECF][U+5178][U+89C6][U+56FE]): [U+53F3][U+51FB][U+4F60][U+7684][U+673A][U+5668][U+56F
([U+5728][U+684C][U+9762][U+7535][U+8111][U+4E0A][U+901A][U+5E38][U+53EB]
"[U+6211][U+7684][U+7535][U+8111]"), [U+9009][U+62E9]:menuselection:*[U+5C5E][U+6027]*.
[U+7136][U+540E][U+6253][U+5F00] **:guilaber:'[U+9AD8][U+7EA7]'**
[U+9009][U+9879][U+5361], [U+70B9][U+51FB] **:guilaber:'[U+73AF][U+5883][U+53D8][U+91CF]'**
[U+6309][U+94AE].

[U+7B80][U+5355][U+5730][U+8BF4], [U+8FC7][U+7A0B][U+4E3A]:

> *[U+6211][U+7684][U+7535][U+8111]* → *[U+5C5E][U+6027]* → *[U+9AD8][U+7EA7]* → *[U+73AF][U+5883][U+53D8][U+91CF]*

[U+5728][U+8FD9][U+4E2A][U+5BF9][U+8BDD][U+6846][U+91CC],
[U+4F60][U+53EF][U+4EE5][U+6DFB][U+52A0][U+6216][U+66F4][U+6539][U+7528][U+6237][U+548C]
[U+8981][U+6539][U+53D8][U+7CFB][U+7EDF][U+53D8][U+91CF],
[U+4F60][U+9700][U+8981][U+6709][U+673A][U+5668][U+7684][U+975E][U+9650][U+5236][U+6027]
([U+5982] Administrator [U+6743][U+9650]).

[U+53E6][U+4E00][U+4E2A][U+6DFB][U+52A0][U+73AF][U+5883][U+53D8][U+91CF][U+7684][U+65B9]
set [U+547D][U+4EE4]:

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

[U+8981][U+4F7F][U+8FD9][U+4E2A][U+8BBE][U+5B9A][U+6C38][U+4E45][U+751F][U+6548],
[U+4F60][U+53EF][U+4EE5][U+628A][U+76F8][U+5E94][U+7684][U+547D][U+4EE4][U+884C][U+6DFB]
autoexec.bat [U+6587][U+4EF6]. **msconfig** [U+662F][U+8FD9][U+4E2A][U+6587][U+4EF6][U+7684][U

Viewing environment variables can also be done more straight-forward: The command prompt will expand strings wrapped into percent signs automatically
[U+67E5][U+770B][U+73AF][U+5883][U+53D8][U+91CF][U+53EF][U+4EE5][U+66F4][U+4E3A][U+76F4]
[U+547D][U+4EE4][U+63D0][U+793A][U+4F1A][U+81EA][U+52A8][U+7684][U+6269][U+5C55][U+4EE5]

```
echo %PATH%
```

[U+67E5][U+8BE2] **set /?** [U+83B7][U+5F97][U+5B83][U+884C][U+4E3A][U+7684][U+66F4][U+591A][

See also:

http://support.microsoft.com/kb/100843 Environment variables in Windows NT

http://support.microsoft.com/kb/310519 How To Manage Environment Variables in Windows XP

http://www.chem.gla.ac.uk/~louis/software/faq/q1.html Setting Environment variables, Louis J. Farrugia

3.3.2 [U+627E][U+5230][U+53EF][U+6267][U+884C][U+7684] Python

[U+9664][U+4E86][U+4F7F][U+7528][U+5F00][U+59CB][U+83DC][U+5355][U+91CC][U+81EA][U+52A8]
Python [U+89E3][U+91CA][U+5668], [U+4F60][U+53EF][U+80FD][U+8FD8][U+8981][U+4ECE]
DOS [U+547D][U+4EE4][U+4E0B][U+6253][U+5F00] Python. [U+8981][U+4F7F][U+5F97][U+8FD9][U+4E2
[U+4F60][U+9700][U+8981][U+8BBE][U+7F6E][U+4F60][U+7684]
%PATH% [U+73AF][U+5883][U+53D8][U+91CF][U+5305][U+542B] Python
[U+7684][U+76EE][U+5F55], [U+901A][U+8FC7][U+5206][U+53F7][U+5206][U+9694][U+5176][U+5B83
[U+4E00][U+4E2A][U+53C2][U+8003][U+7684][U+53D8][U+91CF][U+5982][U+4E0B][U+6240][U+793A]
([U+5047][U+8BBE][U+6700][U+521D][U+4E24][U+4E2A][U+8BB0][U+5F55][U+662F]
Windows [U+9ED8][U+8BA4][U+7684]):

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

[U+73B0][U+5728][U+5728][U+547D][U+4EE4][U+884C][U+4E0B][U+952E][U+5165]
python [U+5C06][U+4F1A][U+6253][U+5F00] Python [U+89E3][U+91CA][U+5668].
[U+56E0][U+6B64], [U+4F60][U+4E5F][U+53EF][U+4EE5][U+4F7F][U+7528][U+547D][U+4EE4][U+884C
[U+53C2][U+9605] *Command line*.

3.3.3 [U+627E][U+5230][U+6A21][U+5757]

Python [U+901A][U+5E38][U+628A][U+5B83][U+7684][U+5E93][U+653E][U+5728][U+5B89][U+88C5][U+
([U+56E0][U+6B64][U+4F60][U+7684] site-packages [U+6587][U+4EF6][U+5939][U+4E5F][U+5728][U+90A
[U+56E0][U+6B64], [U+5982][U+679C][U+4F60][U+628A] Python
[U+5B89][U+88C5][U+5728] C:\Python\ [U+4E0B], [U+9ED8][U+8BA4][U+5E93][U+5C31][U+653E][U+5
C:\Python\Lib\ [U+4E0B], [U+800C][U+7B2C][U+4E09][U+65B9][U+6A21][U+5757][U+5E94][U+8BE5]
C:\Python\Lib\site-packages\.

[U+8FD9][U+662F] Windows [U+4E0A] sys.path [U+7684][U+6784][U+6210]:

- [U+4E00][U+4E2A][U+7A7A][U+7684][U+8BB0][U+5F55][U+6DFB][U+52A0][U+5728][U+5F00][U+5
  [U+5B83][U+5BF9][U+5E94][U+4E8E][U+5F53][U+524D][U+76EE][U+5F55].
- [U+5982][U+679C][U+5B58][U+5728][U+73AF][U+5883][U+53D8][U+91CF]
  PYTHONPATH, [U+5982] *Environment variables* [U+6240][U+63CF][U+8FF0][U+7684],
  [U+5B83][U+7684][U+6761][U+76EE][U+4F1A][U+5728][U+63A5][U+4E0B][U+6765][U+7684][U+5
  [U+6CE8][U+610F][U+5728] Windows [U+4E0A], [U+8FD9][U+4E2A][U+53D8][U+91CF][U+4E0A][U+7
  [U+4EE5][U+533A][U+522B][U+5728][U+76D8][U+7B26] (C:\ [U+7B49])
  [U+4E2D][U+4F7F][U+7528][U+7684][U+5192][U+53F7].
- [U+9644][U+52A0][U+7684] “[U+5E94][U+7528][U+8DEF][U+5F84]”
  [U+53EF][U+4EE5][U+4F5C][U+4E3A] HKEY_CURRENT_USER [U+6216]
  HKEY_LOCAL_MACHINE [U+4E0B][U+7684] \SOFTWARE\Python\PythonCore*version*\PythonPath
  [U+7684][U+5B50][U+952E][U+6DFB][U+52A0][U+5230][U+6CE8][U+518C][U+8868][U+91CC].
  [U+7528][U+5206][U+5F00][U+5206][U+9694][U+7684][U+5B57][U+7B26][U+4E32][U+4F5C][U+4
  sys.path. ([U+6CE8][U+610F], [U+6240][U+6709][U+5DF2][U+77E5][U+7684][U+5B89][U+88C5][U
  HKLM, [U+56E0][U+6B64][U+4E00][U+822C] HKCU [U+662F][U+7A7A][U+7684].)
- [U+5982][U+679C][U+8BBE][U+7F6E][U+4E86][U+73AF][U+5883][U+53D8][U+91CF]
  PYTHONHOME, [U+90A3][U+4E48][U+5B83][U+5C31][U+88AB][U+8BA4][U+4E3A][U+662F]
  “Python Home”. [U+5426][U+5219], [U+4E3B] Python [U+53EF][U+6267][U+884C][U+6587][U+4EF6][U
  “[U+5730][U+6807][U+6587][U+4EF6]” (Lib\os.py) [U+6765][U+63A8][U+6D4B]

"Python Home". [U+5982][U+679C][U+627E][U+5230][U+4E86][U+4E00][U+4E2A] Python home, [U+88AB][U+6DFB][U+52A0][U+5230] sys.path [U+7684][U+5B57][U+76EE][U+5F55] (Lib, plat-win [U+7B49]) [U+5C31][U+57FA][U+4E8E][U+90A3][U+4E2A][U+6587][U+4EF6][U+5939]. [U+5426][U+5219], [U+6838][U+5FC3] Python [U+8DEF][U+5F84][U+5C31][U+901A][U+8FC7][U+6C PythonPath [U+6765][U+6784][U+5EFA].

- [U+5982][U+679C][U+65E0][U+6CD5][U+5B9A][U+4F4D] Python Home, [U+73AF][U+5883][U+91CC][U+6CA1][U+6709][U+6307][U+5B9A] *PYTHONPATH*, [U+6CA1][U+6709][U+6CE8][U+518C][U+8868][U+8BB0][U+5F55][U+88AB][U+627E][U+5230], [U+90A3][U+4E48][U+5C31][U+4F7F][U+7528][U+4E00][U+4E2A][U+9ED8][U+8BA4][U+7684][U+ ([U+4F8B][U+5982], .\Lib;.\plat-win [U+7B49]).

[U+8FD9][U+4E00][U+5207][U+7684][U+7ED3][U+679C][U+5982][U+4E0B]:

- [U+5F53][U+5728] Python [U+4E3B][U+76EE][U+5F55] ([U+5373][U+53EF][U+4EE5][U+662F][U+5B [U+4E5F][U+53EF][U+4EE5][U+76F4][U+63A5][U+662F] PCBuild [U+76EE][U+5F55]) [U+4E0B][U+8FD0][U+884C] python.exe, [U+6216][U+4EFB][U+610F][U+5176][U+5B83] .exe [U+6587][U+4EF6], [U+6838][U+5FC3][U+8DEF][U+5F84][U+5C31][U+88AB][U+63A8][U+65AD][U+51FA][U+6765][U+ [U+800C][U+6CE8][U+518C][U+8868][U+91CC][U+7684][U+6838][U+5FC3][U+8DEF][U+5F84][U+5 [U+6CE8][U+518C][U+8868][U+91CC][U+7684][U+5176][U+5B83] "[U+5E94][U+7528][U+8DEF][U+5F84]" [U+4E00][U+76F4][U+88AB][U+8BFB][U+53D6].
- [U+5F53] Python [U+5BC4][U+5BBF][U+5728][U+53E6][U+4E00][U+4E2A] .exe [U+91CC] ([U+4E0D][U+540C][U+7684][U+76EE][U+5F55], [U+901A][U+8FC7] COM [U+5D4C][U+5165][U+7B49]), "Python Home" [U+4E0D][U+4F1A][U+88AB][U+63A8][U+65AD][U+51FA], [U+56E0][U+6B64][U+4F7F][U+7528][U+6CE8][U+518C][U+8868][U+91CC][U+7684][U+6838][U+5 [U+6CE8][U+518C][U+8868][U+91CC][U+7684][U+5176][U+5B83] "[U+5E94][U+7528][U+8DEF][U+5F84]" [U+4E00][U+76F4][U+88AB][U+8BFB][U+53D6].
- [U+5982][U+679C] Python [U+65E0][U+6CD5][U+627E][U+5230][U+5B83][U+7684] home, [U+4E5F][U+6CA1][U+6709][U+6CE8][U+518C][U+8868] ([U+5982], [U+51BB][U+7ED3][U+7684] .exe, [U+4E00][U+4E9B][U+975E][U+5E38][U+5927][U+7684][U+5B89] [U+4F60][U+4F1A][U+5F97][U+5230][U+4E00][U+4E2A][U+9ED8][U+8BA4][U+4F46][U+662F][U+

3.3.4 [U+53EF][U+6267][U+884C][U+811A][U+672C]

Python [U+811A][U+672C] ([U+540E][U+7F00][U+4E3A] .py [U+7684][U+6587][U+4EF6]) [U+9ED8][U+8BA4][U+4E0B][U+88AB] **python.exe** [U+6267][U+884C]. [U+8FD9][U+4E2A][U+53EF][U+6267][U+884C][U+6587][U+4EF6][U+4F1A][U+6253][U+5F00][U+4E00] [U+5373][U+4F7F][U+7A0B][U+5E8F][U+4F7F][U+7528] GUI [U+65F6][U+7EC8][U+7AEF][U+4E5F][U+4 [U+5982][U+679C][U+4F60][U+4E0D][U+60F3][U+8BA9][U+5B83][U+53D1][U+751F], [U+90A3][U+4E48][U+4F7F][U+7528][U+540E][U+7F00] .pyw, [U+8FD9][U+6837][U+811A][U+672C][U+5C31][U+9ED8][U+8BA4][U+7531] **pythonw.exe** [U+6267][U+884C] ([U+8FD9][U+4E24][U+4E2A][U+53EF][U+6267][U+884C][U+6587][U+4EF6][U+90FD Python [U+7684][U+9876][U+7EA7][U+76EE][U+5F55][U+4E0B]). [U+8FD9][U+6837][U+4F1A][U+5728][U+542F][U+52A8][U+65F6][U+963B][U+6B62][U+7EC8][U+7AEF]

[U+4F60][U+53EF][U+80FD][U+60F3][U+8BA9] .py [U+811A][U+672C][U+4E5F][U+7531] **pythonw.exe** [U+6267][U+884C], [U+53EF][U+4EE5][U+5728] usual facilities [U+91CC][U+8BBE][U+7F6E][U+5B83], [U+4F8B][U+5982] ([U+53EF][U+80FD][U+9700][U+8981][U+7BA1][U+7406][U+5458][U+6743][U+9650]):

1. [U+6253][U+5F00][U+4E00][U+4E2A][U+547D][U+4EE4][U+884C].
2. [U+5173][U+8054] .py [U+811A][U+672C][U+5230][U+6B63][U+786E][U+7684][U+6587][U+4EF6]

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

3. [U+91CD][U+5B9A][U+5411][U+6240][U+6709] Python [U+6587][U+4EF6][U+5230][U+65B0][U+768

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

3.4 [U+9644][U+52A0][U+6A21][U+5757]

[U+5C3D][U+7BA1] Python [U+65E8][U+5728][U+6240][U+6709][U+5E73][U+53F0][U+4E0A][U+7684][U+
[U+4F46][U+5728] Windows [U+4E0A][U+8FD8][U+6709][U+4E00][U+4E9B][U+72EC][U+6709][U+7684][
[U+5B58][U+5728][U+51E0][U+4E2A][U+6A21][U+5757], [U+6709][U+6807][U+51C6][U+5E93][U+91CC
[U+548C][U+4EE3][U+7801][U+7247][U+6BB5][U+4F7F][U+7528][U+8FD9][U+4E9B][U+7279][U+6027

Windows [U+72EC][U+6709][U+7684][U+6807][U+51C6][U+6A21][U+5757][U+7684][U+6587][U+6863][
mswin-specific-services.

3.4.1 PyWin32

Mark Hammond [U+7684] PyWin32 [U+6A21][U+5757][U+662F][U+9AD8][U+7EA7] Windows
[U+72EC][U+6709][U+652F][U+6301][U+7684][U+6A21][U+5757][U+7684][U+4E00][U+4E2A][U+96C6]
[U+5305][U+542B][U+7684][U+5DE5][U+5177][U+6709]:

- [U+7EC4][U+4EF6][U+5BF9][U+8C61][U+6A21][U+578B] (COM)
- Win32 API [U+8C03][U+7528]
- [U+6CE8][U+518C][U+8868]
- [U+4E8B][U+4EF6][U+65E5][U+5FD7]
- Microsoft Foundation Classes (MFC) [U+7528][U+6237][U+63A5][U+53E3]

PythonWin [U+662F][U+4E00][U+4E2A][U+4F7F][U+7528] PyWin32
[U+88C5][U+8F7D][U+7684][U+7B80][U+5355] MFC [U+5E94][U+7528].
[U+5B83][U+662F][U+4E00][U+4E2A][U+53EF][U+5D4C][U+5165][U+7684] IDE,
[U+5185][U+5EFA][U+6709][U+4E00][U+4E2A][U+8C03][U+8BD5][U+5668].

See also:

Win32 How Do I...? by Tim Golden

Python and COM by David and Paul Boddie

3.4.2 Py2exe

Py2exe [U+662F][U+4E00][U+4E2A] distutils [U+6269][U+5C55]
([U+53C2][U+9605] extending-distutils), [U+5B83][U+5305] Python
[U+811A][U+672C][U+5305][U+88C5][U+6210][U+4E00][U+4E2A][U+53EF][U+6267][U+884C][U+7684]
Windows [U+7A0B][U+5E8F] (`*.exe` [U+6587][U+4EF6]). [U+4F46][U+4F60][U+505A][U+8FD9][U+70B9][U
[U+4F60][U+53EF][U+4EE5][U+53D1][U+5E03][U+4F60][U+7684][U+5E94][U+7528][U+800C][U+65E0]
Python.

3.4.3 WConio

[U+56E0][U+4E3A] Python [U+9AD8][U+7EA7][U+7EC8][U+7AEF][U+5904][U+7406][U+5C42]
(advanced terminal handling layer), `curses`, [U+4EC5][U+9650][U+4E8E]
Unix-like [U+7CFB][U+7EDF], [U+56E0][U+6B64][U+5728] Windows
[U+4E0A][U+540C][U+6837][U+6709][U+4E00][U+4E2A][U+72EC][U+6709][U+7684][U+5E93]:
Python [U+5728] Windows [U+4E0A][U+7684][U+63A7][U+5236][U+53F0] I/O (Windows Console
I/O for Python).

WConio [U+662F] Turbo-C [U+91CC] `CONIO.H` [U+6587][U+4EF6][U+7684][U+4E00][U+4E2A][U+5305][U+8
[U+7528][U+4E8E][U+521B][U+5EFA][U+6587][U+672C][U+7528][U+6237][U+63A5][U+53E3].

3.5 [U+5728] Windows [U+4E0A][U+7F16][U+8BD1] Python

[U+5982][U+679C][U+4F60][U+60F3][U+81EA][U+5DF1][U+7F16][U+8BD1] CPython, [U+4F60][U+8981][U+505A][U+7684][U+7B2C][U+4E00][U+4EF6][U+4E8B][U+60C5][U+662F][U+83B7][U+6E90][U+4EE3][U+7801]. [U+4F60][U+53EF][U+4EE5][U+9009][U+62E9][U+4E0B][U+8F7D][U+6700[U+4E5F][U+53EF][U+4EE5][U+6293][U+53D6][U+4E00][U+4E2A][U+65B0][U+9C9C][U+7684] checkout.

[U+5BF9][U+4E0E] Microsoft Visual C++, [U+5B98][U+65B9] Python [U+53D1][U+884C][U+7248][U+4F7F][U+7528][U+7684][U+7F16][U+8BD1][U+5668], [U+6E90][U+4EE3][U+7801][U+6811][U+5305][U+542B] solutions/project [U+6587][U+4EF6]. [U+53C2][U+770B][U+5B83][U+4EEC][U+5404][U+81EA][U+76EE][U+5F55][U+4E0E readme.txt [U+6587][U+4EF6]:

[U+76EE][U+5F55]	MSVC [U+7248][U+672C]	Visual Studio [U+7248][U+672C]
PC/VC6/	6.0	97
PC/VS7.1/	7.1	2003
PC/VS8.0/	8.0	2005
PCbuild/	9.0	2008

[U+6CE8][U+610F], [U+4E0D][U+662F][U+6240][U+6709][U+8FD9][U+4E9B][U+76EE][U+7684][U+90FD][U+9605][U+8BFB][U+53D1][U+884C][U+8BF4][U+660E][U+6765], [U+5BF9][U+5E94][U+4F60][U+7684][U+7248][U+672C], [U+770B][U+770B][U+76F8][U+5E94][U+5B98]

[U+68C0][U+67E5] PC/readme.txt [U+83B7][U+5F97][U+5EFA][U+9020][U+6D41][U+7A0B][U+7684][U

[U+5BF9][U+4E8E][U+6269][U+5C55][U+6A21][U+5757], [U+53C2][U+9605] building-on-windows.

See also:

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

MingW – Python extensions by Trent Apted et al, 2007

3.6 [U+5176][U+5B83][U+8D44][U+6E90]

See also:

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

CHAPTER 4

Using Python on a Macintosh

Author 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.

4.1 Getting and Installing MacPython

Mac OS X 10.5 comes with Python 2.5.1 pre-installed by Apple. If you wish, you are invited to install the most recent version of Python from the Python website (http://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 2.5` 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.shtml) are good choices, as is **TextMate** (see http://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 2.5, 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 http://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:

- http://pythonmac.org/packages/ contains selected compiled packages for Python 2.5, 2.4, and 2.3.
- Packages can be installed via the standard Python distutils mode (`python setup.py install`).
- Many packages can also be installed via the **`setuptools`** extension.

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 http://pyobjc.sourceforge.net.

The standard Python GUI toolkit is `tkinter`, based on the cross-platform Tk toolkit (http://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 http://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 http://www.riverbankcomputing.co.uk/software/pyqt/intro.

4.5 Distributing Python Applications on the Mac

The "Build Applet" tool that is placed in the MacPython 2.5 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 Application Scripting

Python can also be used to script other Mac applications via Apple's Open Scripting Architecture (OSA); see http://appscript.sourceforge.net. Appscript is a high-level, user-friendly Apple event bridge that allows you to control scriptable Mac OS X applications using ordinary Python scripts. Appscript makes Python a serious alternative to Apple's own *AppleScript* language for automating your Mac. A related package, *PyOSA*, is an OSA language component for the Python scripting language, allowing Python code to be executed by any OSA-enabled application (Script Editor, Mail, iTunes, etc.). PyOSA makes Python a full peer to AppleScript.

4.7 Other Resources

The MacPython mailing list is an excellent support resource for Python users and developers on the Mac:

http://www.python.org/community/sigs/current/pythonmac-sig/

Another useful resource is the MacPython wiki:

http://wiki.python.org/moin/MacPython

APPENDIX A

Glossary

>>> The default Python prompt of the interactive shell. Often seen for code examples which can be executed interactively in the interpreter.

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

2to3 A tool that tries to convert Python 2.x code to Python 3.x code by handling most of the incompatibilities which can be detected by parsing the source and traversing the parse tree.

2to3 is available in the standard library as `lib2to3`; a standalone entry point is provided as `Tools/scripts/2to3`. See 2to3-reference.

abstract base class 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). Python comes with many built-in ABCs for data structures (in the `collections` module), numbers (in the `numbers` module), streams (in the `io` module), import finders and loaders (in the `importlib.abc` module). You can create your own ABCs with the `abc` module.

argument A value passed to a function or method, assigned to a named local variable in the function body. A function or method may have both positional arguments and keyword arguments in its definition. Positional and keyword arguments may be variable-length: `*` accepts or passes (if in the function definition or call) several positional arguments in a list, while `**` does the same for keyword arguments in a dictionary.

Any expression may be used within the argument list, and the evaluated value is passed to the local variable.

attribute A value associated with an object which is referenced by name using dotted expressions. For example, if an object *o* has an attribute *a* it would be referenced as *o.a*.

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

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.

A list of bytecode instructions can be found in the documentation for the dis module.

class A template for creating user-defined objects. Class definitions normally contain method definitions which operate on instances of the class.

coercion 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`. 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`.

complex number An extension of the familiar real number system in which all numbers are expressed as a sum of a real part and an imaginary part. Imaginary numbers are real multiples of the imaginary unit (the square root of `-1`), often written `i` in mathematics or `j` in engineering. Python has built-in support for complex numbers, which are written with this latter notation; the imaginary part is written with a `j` suffix, e.g., `3+1j`. To get access to complex equivalents of the `math` module, use `cmath`. Use of complex numbers is a fairly advanced mathematical feature. If you're not aware of a need for them, it's almost certain you can safely ignore them.

context manager An object which controls the environment seen in a `with` statement by defining `__enter__()` and `__exit__()` methods. See **PEP 343**.

CPython The canonical implementation of the Python programming language, as distributed on python.org. The term "CPython" is used when necessary to distinguish this implementation from others such as Jython or IronPython.

decorator A function returning another function, usually applied as a function transformation using the `@wrapper` syntax. Common examples for decorators are `classmethod()` and `staticmethod()`.

The decorator syntax is merely syntactic sugar, the following two function definitions are semantically equivalent:

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

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

The same concept exists for classes, but is less commonly used there. See the documentation for function definitions and class definitions for more about decorators.

descriptor Any 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.

For more information about descriptors' methods, see descriptors.

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

docstring A string literal which appears as the first expression in a class, function or module. While ignored when the suite is executed, it is recognized by the compiler and put into the `__doc__` attribute of the enclosing class, function or module. Since it is available via introspection, it is the canonical place for documentation of the object.

duck-typing A programming style which does not look at an object's type to determine if it has the right interface; instead, the method or attribute is simply called or used ("If it looks like a duck and quacks like a duck, it must be a duck.") By emphasizing interfaces rather than specific types, well-designed code improves its flexibility by allowing polymorphic substitution. Duck-typing avoids tests using `type()` or `isinstance()`. (Note, however, that duck-typing can be complemented with *abstract base classes*.) Instead, it typically employs `hasattr()` tests or *EAFP* programming.

EAFP Easier to ask for forgiveness than permission. This common Python coding style assumes the existence of valid keys or attributes and catches exceptions if the assumption proves false. This clean and fast style is characterized by the presence of many `try` and `except` statements. The technique contrasts with the *LBYL* style common to many other languages such as C.

expression A piece of syntax which can be evaluated to some value. In other words, an expression is an accumulation of expression elements like literals, names, attribute access, operators or function calls which all return a value. In contrast to many other languages, not all language constructs are expressions. There

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

extension module A module written in C or C++, using Python's C API to interact with the core and with user code.

file object An object exposing a file-oriented API (with methods such as `read()` or `write()`) to an underlying resource. Depending on the way it was created, a file object can mediate access to a real on-disk file or to another other type of storage or communication device (for example standard input/output, in-memory buffers, sockets, pipes, etc.). File objects are also called *file-like objects* or *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.

file-like object A synonym for *file object*.

finder An object that tries to find the *loader* for a module. It must implement a method named `find_module()`. See **PEP 302** for details and `importlib.abc.Finder` for an *abstract base class*.

floor division Mathematical division that rounds down to nearest integer. The floor division operator is `//`. For example, the expression `11 // 4` evaluates to `2` in contrast to the `2.75` returned by float true division. Note that `(-11) // 4` is `-3` because that is `-2.75` rounded *downward*. See **PEP 238**.

function A series of statements which returns some value to a caller. It can also be passed zero or more arguments which may be used in the execution of the body. See also *argument* and *method*.

__future__ A pseudo-module which programmers can use to enable new language features which are not compatible with the current interpreter.

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 becomes the default:

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

garbage collection 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.

generator A function which returns an iterator. It looks like a normal function except that it contains `yield` statements for producing a series a 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 See *global interpreter lock*.

global interpreter lock The mechanism used by the *CPython* interpreter to assure that only one thread executes Python *bytecode* at a time. This simplifies the CPython implementation by making the object model (including critical built-in types such as `dict`) implicitly safe against concurrent access. Locking the entire interpreter makes it easier for the interpreter to be multi-threaded, at the expense of much of the parallelism afforded by multi-processor machines.

However, some extension modules, either standard or third-party, are designed so as to release the GIL when doing computationally-intensive tasks such as compression or hashing. Also, the GIL is always released when doing I/O.

Past efforts to create a "free-threaded" interpreter (one which locks shared data at a much finer granularity) have not been successful because performance suffered in the common single-processor case. It is believed that overcoming this performance issue would make the implementation much more complicated and therefore costlier to maintain.

hashable 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__()` method). Hashable objects which compare equal must have the same hash value.

Hashability makes an object usable as a dictionary key and a set member, because these data structures use the hash value internally.

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, and their hash value is their `id()`.

IDLE An Integrated Development Environment for Python. IDLE is a basic editor and interpreter environment which ships with the standard distribution of Python.

immutable An object with a fixed value. Immutable objects include numbers, strings and tuples. Such an object cannot be altered. A new object has to be created if a different value has to be stored. They play an important role in places where a constant hash value is needed, for example as a key in a dictionary.

importer An object that both finds and loads a module; both a *finder* and *loader* object.

interactive Python has an interactive interpreter which means you can enter statements and expressions at the interpreter prompt, immediately execute them and see their results. Just launch `python` with no arguments (possibly by selecting it from your computer's main menu). It is a very powerful way to test out new ideas or inspect modules and packages (remember `help(x)`).

interpreted Python is an interpreted language, as opposed to a compiled one, though the distinction can be blurry because of the presence of the bytecode compiler. This means that source files can be run directly without explicitly creating an executable which is then run. Interpreted languages typically have a shorter development/debug cycle than compiled ones, though their programs generally also run more slowly. See also *interactive*.

iterable 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*.

iterator An object representing a stream of data. Repeated calls to the iterator's `__next__()` method (or passing it to the built-in function `next()`) 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.

More information can be found in typeiter.

key function A key function or collation function is a callable that returns a value used for sorting or ordering. For example, `locale.strxfrm()` is used to produce a sort key that is aware of locale specific sort conventions.

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 constuctors: `attrgetter()`, `itemgetter()`, and `methodcaller()`. See the Sorting HOW TO for examples of how to create and use key functions.

keyword argument Arguments which are preceded with a `variable_name=` in the call. The variable name designates the local name in the function to which the value is assigned. `**` is used to accept or pass a dictionary of keyword arguments. See *argument*.

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

LBYL Look before you leap. This coding style explicitly tests for pre-conditions before making calls or lookups. This style contrasts with the *EAFP* approach and is characterized by the presence of many `if` statements.

In a multi-threaded environment, the LBYL approach can risk introducing a race condition between "the looking" and "the leaping". For example, the code, `if key in mapping: return mapping[key]` can fail if another thread removes *key* from *mapping* after the test, but before the lookup. This issue can be solved with locks or by using the EAFP approach.

list A built-in Python *sequence*. Despite its name it is more akin to an array in other languages than to a linked list since access to elements are 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 = ['{:#04x}'.format(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 and `importlib.abc.Loader` for an *abstract base class*.

mapping 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 The class of a class. Class definitions create a class name, a class dictionary, and a list of base classes. The metaclass is responsible for taking those three arguments and creating the class. Most object oriented programming languages provide a default implementation. What makes Python special is that it is possible to create custom metaclasses. Most users never need this tool, but when the need arises, metaclasses can provide powerful, elegant solutions. They have been used for logging attribute access, adding thread-safety, tracking object creation, implementing singletons, and many other tasks.

More information can be found in metaclasses.

method A function which is defined inside a class body. If called as an attribute of an instance of that class, the method will get the instance object as its first *argument* (which is usually called `self`). See *function* and *nested scope*.

method resolution order Method Resolution Order is the order in which base classes are searched for a member during lookup. See The Python 2.3 Method Resolution Order.

MRO See *method resolution order*.

mutable Mutable objects can change their value but keep their `id()`. See also *immutable*.

named tuple 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 `builtins.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.islice()` makes it clear that those functions are implemented by the `random` and `itertools` modules, respectively.

nested scope 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 by default work only for reference and not for assignment. Local variables both read and write in the innermost scope. Likewise, global variables read and write to the global namespace. The `nonlocal` allows writing to outer scopes.

new-style class Old name for the flavor of classes now used for all class objects. In earlier Python versions, only new-style classes could use Python's newer, versatile features like `__slots__`, descriptors, properties, `__getattribute__()`, class methods, and static methods.

object Any data with state (attributes or value) and defined behavior (methods). Also the ultimate base class of any *new-style class*.

positional argument The arguments assigned to local names inside a function or method, determined by the order in which they were given in the call. `*` is used to either accept multiple positional arguments (when in the definition), or pass several arguments as a list to a function. See *argument*.

Python 3000 Nickname for the Python 3.x release line (coined long ago when the release of version 3 was something in the distant future.) This is also abbreviated "Py3k".

Pythonic An idea or piece of code which closely follows the most common idioms of the Python language, rather than implementing code using concepts common to other languages. For example, a common idiom in Python is to loop over all elements of an iterable using a `for` statement. Many other languages don't have this type of construct, so people unfamiliar with Python sometimes use a numerical counter instead:

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

As opposed to the cleaner, Pythonic method:

```
for piece in food:
    print(piece)
```

reference count The number of references to an object. When the reference count of an object drops to zero, it is deallocated. Reference counting is generally not visible to Python code, but it is a key element of the *CPython* implementation. The `sys` module defines a `getrefcount()` function that programmers can call to return the reference count for a particular object.

__slots__ A declaration inside a 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.

sequence 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 `bytes`. 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.

slice 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.

special method A method that is called implicitly by Python to execute a certain operation on a type, such as addition. Such methods have names starting and ending with double underscores. Special methods are documented in specialnames.

statement A statement is part of a suite (a "block" of code). A statement is either an *expression* or a one of several constructs with a keyword, such as `if`, `while` or `for`.

triple-quoted string A string which is bound by three instances of either a quotation mark (") or an apostrophe ('). While they don't provide any functionality not available with single-quoted strings, they are useful for a

number of reasons. They allow you to include unescaped single and double quotes within a string and they can span multiple lines without the use of the continuation character, making them especially useful when writing docstrings.

type The type of a Python object determines what kind of object it is; every object has a type. An object's type is accessible as its `__class__` attribute or can be retrieved with `type(obj)`.

view The objects returned from `dict.keys()`, `dict.values()`, and `dict.items()` are called dictionary views. They are lazy sequences that will see changes in the underlying dictionary. To force the dictionary view to become a full list use `list(dictview)`. See dict-views.

virtual machine A computer defined entirely in software. Python's virtual machine executes the *bytecode* emitted by the bytecode compiler.

Zen of Python Listing of Python design principles and philosophies that are helpful in understanding and using the language. The listing can be found by typing "`import this`" at the interactive prompt.

APPENDIX B

About these documents

These documents are generated from reStructuredText sources by Sphinx, a document processor specifically written for the Python documentation.

Development of the documentation and its toolchain takes place on the docs@python.org mailing list. We're always looking for volunteers wanting to help with the docs, so feel free to send a mail there!

Many thanks go to:

- Fred L. Drake, Jr., the creator of the original Python documentation toolset and writer of much of the content;
- the Docutils project for creating reStructuredText and the Docutils suite;
- Fredrik Lundh for his Alternative Python Reference project from which Sphinx got many good ideas.

See reporting-bugs for information how to report bugs in this documentation, or Python itself.

B.1 Contributors to the Python Documentation

This section lists people who have contributed in some way to the Python documentation. It is probably not complete – if you feel that you or anyone else should be on this list, please let us know (send email to docs@python.org), and we'll be glad to correct the problem.

Aahz, Michael Abbott, Steve Alexander, Jim Ahlstrom, Fred Allen, A. Amoroso, Pehr Anderson, Oliver Andrich, Heidi Annexstad, Jesús Cea Avión, Manuel Balsera, Daniel Barclay, Chris Barker, Don Bashford, Anthony Baxter, Alexander Belopolsky, Bennett Benson, Jonathan Black, Robin Boerdijk, Michal Bozon, Aaron Brancotti, Georg Brandl, Keith Briggs, Ian Bruntlett, Lee Busby, Lorenzo M. Catucci, Carl Cerecke, Mauro Cicognini, Gilles Civario, Mike Clarkson, Steve Clift, Dave Cole, Matthew Cowles, Jeremy Craven, Andrew Dalke, Ben Darnell, L. Peter Deutsch, Robert Donohue, Fred L. Drake, Jr., Jacques Ducasse, Josip Dzolonga, Jeff Epler, Michael Ernst, Blame Andy Eskilsson, Carey Evans, Martijn Faassen, Carl Feynman, Dan Finnie, Hernán Martínez Foffani, Stefan Franke, Jim Fulton, Peter Funk, Lele Gaifax, Matthew Gallagher, Gabriel Genellina, Ben Gertzfield, Nadim Ghaznavi, Jonathan Giddy, Matt Giuca, Shelley Gooch, Nathaniel Gray, Grant Griffin, Thomas Guettler, Anders Hammarquist, Mark Hammond, Harald Hanche-Olsen, Manus Hand, Gerhard Häring, Travis B. Hartwell, Tim Hatch, Janko Hauser, Ben Hayden, Thomas Heller, Bernhard Herzog, Magnus L. Hetland, Konrad Hinsen, Stefan Hoffmeister, Albert Hofkamp, Gregor Hoffleit, Steve Holden, Thomas Holenstein, Gerrit Holl, Rob Hooft, Brian Hooper, Randall Hopper, Michael Hudson, Eric Huss, Jeremy Hylton, Roger Irwin, Jack Jansen, Philip H. Jensen, Pedro Diaz Jimenez, Kent Johnson, Lucas de Jonge, Andreas Jung, Robert Kern, Jim Kerr, Jan Kim, Kamil Kisiel, Greg Kochanski, Guido Kollerie, Peter A. Koren, Daniel Kozan, Andrew M. Kuchling, Dave Kuhlman, Erno Kuusela, Ross Lagerwall, Thomas Lamb, Detlef Lannert, Piers Lauder, Glyph Lefkowitz, Robert Lehmann, Marc-André Lemburg, Ross Light, Gediminas Liktaras, Ulf A. Lindgren, Everett Lipman, Mirko Liss, Martin von Löwis, Fredrik Lundh, Jeff MacDonald, John Machin, Andrew MacIntyre, Vladimir Marangozov, Vincent Marchetti, Westley Martínez, Laura Matson, Daniel May, Rebecca McCreary, Doug Mennella, Paolo Milani, Skip Montanaro, Paul Moore, Ross Moore, Sjoerd Mullender, Dale Nagata, Michal Nowikowski, Steffen Daode Nurpmeso, Ng Pheng Siong, Koray Oner, Tomas Oppelstrup, Denis S. Otkidach, Zooko O'Whielacronx, Shriphani Palakodety, William Park, Joonas Paalasmaa, Harri Pasanen, Bo Peng, Tim Peters, Benjamin Peterson,

Christopher Petrilli, Justin D. Pettit, Chris Phoenix, François Pinard, Paul Prescod, Eric S. Raymond, Edward K. Ream, Terry J. Reedy, Sean Reifschneider, Bernhard Reiter, Armin Rigo, Wes Rishel, Armin Ronacher, Jim Roskind, Guido van Rossum, Donald Wallace Rouse II, Mark Russell, Nick Russo, Chris Ryland, Constantina S., Hugh Sasse, Bob Savage, Scott Schram, Neil Schemenauer, Barry Scott, Joakim Sernbrant, Justin Sheehy, Charlie Shepherd, SilentGhost, Michael Simcich, Ionel Simionescu, Michael Sloan, Gregory P. Smith, Roy Smith, Clay Spence, Nicholas Spies, Tage Stabell-Kulo, Frank Stajano, Anthony Starks, Greg Stein, Peter Stoehr, Mark Summerfield, Reuben Sumner, Kalle Svensson, Jim Tittsler, David Turner, Sandro Tosi, Ville Vainio, Martijn Vries, Charles G. Waldman, Greg Ward, Barry Warsaw, Corran Webster, Glyn Webster, Bob Weiner, Eddy Welbourne, Jeff Wheeler, Mats Wichmann, Gerry Wiener, Timothy Wild, Paul Winkler, Collin Winter, Blake Winton, Dan Wolfe, Steven Work, Thomas Wouters, Ka-Ping Yee, Rory Yorke, Moshe Zadka, Milan Zamazal, Cheng Zhang, Trent Nelson, Michael Foord.

It is only with the input and contributions of the Python community that Python has such wonderful documentation – Thank You!

APPENDIX C

History and License

C.1 History of the software

Python was created in the early 1990s by Guido van Rossum at Stichting Mathematisch Centrum (CWI, see http://www.cwi.nl/) in the Netherlands as a successor of a language called ABC. Guido remains Python's principal author, although it includes many contributions from others.

In 1995, Guido continued his work on Python at the Corporation for National Research Initiatives (CNRI, see http://www.cnri.reston.va.us/) in Reston, Virginia where he released several versions of the software.

In May 2000, Guido and the Python core development team moved to BeOpen.com to form the BeOpen PythonLabs team. In October of the same year, the PythonLabs team moved to Digital Creations (now Zope Corporation; see http://www.zope.com/). In 2001, the Python Software Foundation (PSF, see http://www.python.org/psf/) was formed, a non-profit organization created specifically to own Python-related Intellectual Property. Zope Corporation is a sponsoring member of the PSF.

All Python releases are Open Source (see http://www.opensource.org/ for the Open Source Definition). Historically, most, but not all, Python releases have also been GPL-compatible; the table below summarizes the various releases.

Release	Derived from	Year	Owner	GPL compatible?
0.9.0 thru 1.2	n/a	1991-1995	CWI	yes
1.3 thru 1.5.2	1.2	1995-1999	CNRI	yes
1.6	1.5.2	2000	CNRI	no
2.0	1.6	2000	BeOpen.com	no
1.6.1	1.6	2001	CNRI	no
2.1	2.0+1.6.1	2001	PSF	no
2.0.1	2.0+1.6.1	2001	PSF	yes
2.1.1	2.1+2.0.1	2001	PSF	yes
2.2	2.1.1	2001	PSF	yes
2.1.2	2.1.1	2002	PSF	yes
2.1.3	2.1.2	2002	PSF	yes
2.2.1	2.2	2002	PSF	yes
2.2.2	2.2.1	2002	PSF	yes
2.2.3	2.2.2	2002-2003	PSF	yes
2.3	2.2.2	2002-2003	PSF	yes
2.3.1	2.3	2002-2003	PSF	yes
2.3.2	2.3.1	2003	PSF	yes
2.3.3	2.3.2	2003	PSF	yes
2.3.4	2.3.3	2004	PSF	yes
2.3.5	2.3.4	2005	PSF	yes
2.4	2.3	2004	PSF	yes
2.4.1	2.4	2005	PSF	yes

Continued on next page

Table C.1 – continued from previous page

Release	Derived from	Year	Owner	GPL compatible?
2.4.2	2.4.1	2005	PSF	yes
2.4.3	2.4.2	2006	PSF	yes
2.4.4	2.4.3	2006	PSF	yes
2.5	2.4	2006	PSF	yes
2.5.1	2.5	2007	PSF	yes
2.6	2.5	2008	PSF	yes
2.6.1	2.6	2008	PSF	yes
2.6.2	2.6.1	2009	PSF	yes
2.6.3	2.6.2	2009	PSF	yes
2.6.4	2.6.3	2009	PSF	yes
3.0	2.6	2008	PSF	yes
3.0.1	3.0	2009	PSF	yes
3.1	3.0.1	2009	PSF	yes
3.1.1	3.1	2009	PSF	yes
3.1.2	3.1.1	2010	PSF	yes
3.1.3	3.1.2	2010	PSF	yes
3.1.4	3.1.3	2011	PSF	yes
3.2	3.1	2011	PSF	yes
3.2.1	3.2	2011	PSF	yes
3.2.2	3.2.1	2011	PSF	yes

Note: GPL-compatible doesn't mean that we're distributing Python under the GPL. All Python licenses, unlike the GPL, let you distribute a modified version without making your changes open source. The GPL-compatible licenses make it possible to combine Python with other software that is released under the GPL; the others don't.

Thanks to the many outside volunteers who have worked under Guido's direction to make these releases possible.

C.2 Terms and conditions for accessing or otherwise using Python

PSF LICENSE AGREEMENT FOR PYTHON 3.2.2

1. This LICENSE AGREEMENT is between the Python Software Foundation ("PSF"), and the Individual or Organization ("Licensee") accessing and otherwise using Python 3.2.2 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 3.2.2 alone or in any derivative version, provided, however, that PSF's License Agreement and PSF's notice of copyright, i.e., "Copyright © 2001-2011 Python Software Foundation; All Rights Reserved" are retained in Python 3.2.2 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 3.2.2 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 3.2.2.

4. PSF is making Python 3.2.2 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 3.2.2 WILL NOT INFRINGE ANY THIRD PARTY RIGHTS.

5. PSF SHALL NOT BE LIABLE TO LICENSEE OR ANY OTHER USERS OF PYTHON 3.2.2 FOR ANY INCIDENTAL, SPECIAL, OR CONSEQUENTIAL DAMAGES OR LOSS AS A RESULT OF MODIFY-

ING, DISTRIBUTING, OR OTHERWISE USING PYTHON 3.2.2, 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 3.2.2, Licensee agrees to be bound by the terms and conditions of this License Agreement.

BEOPEN.COM LICENSE AGREEMENT FOR PYTHON 2.0

BEOPEN PYTHON OPEN SOURCE LICENSE AGREEMENT VERSION 1

1. This LICENSE AGREEMENT is between BeOpen.com ("BeOpen"), having an office at 160 Saratoga Avenue, Santa Clara, CA 95051, and the Individual or Organization ("Licensee") accessing and otherwise using this software in source or binary form and its associated documentation ("the Software").

2. Subject to the terms and conditions of this BeOpen Python License Agreement, BeOpen hereby grants Licensee a non-exclusive, royalty-free, world-wide license to reproduce, analyze, test, perform and/or display publicly, prepare derivative works, distribute, and otherwise use the Software alone or in any derivative version, provided, however, that the BeOpen Python License is retained in the Software, alone or in any derivative version prepared by Licensee.

3. BeOpen is making the Software available to Licensee on an "AS IS" basis. BEOPEN MAKES NO REPRESENTATIONS OR WARRANTIES, EXPRESS OR IMPLIED. BY WAY OF EXAMPLE, BUT NOT LIMITATION, BEOPEN MAKES NO AND DISCLAIMS ANY REPRESENTATION OR WARRANTY OF MERCHANTABILITY OR FITNESS FOR ANY PARTICULAR PURPOSE OR THAT THE USE OF THE SOFTWARE WILL NOT INFRINGE ANY THIRD PARTY RIGHTS.

4. BEOPEN SHALL NOT BE LIABLE TO LICENSEE OR ANY OTHER USERS OF THE SOFTWARE FOR ANY INCIDENTAL, SPECIAL, OR CONSEQUENTIAL DAMAGES OR LOSS AS A RESULT OF USING, MODIFYING OR DISTRIBUTING THE SOFTWARE, OR ANY DERIVATIVE THEREOF, EVEN IF ADVISED OF THE POSSIBILITY THEREOF.

5. This License Agreement will automatically terminate upon a material breach of its terms and conditions.

6. This License Agreement shall be governed by and interpreted in all respects by the law of the State of California, excluding conflict of law provisions. Nothing in this License Agreement shall be deemed to create any relationship of agency, partnership, or joint venture between BeOpen and Licensee. This License Agreement does not grant permission to use BeOpen trademarks or trade names in a trademark sense to endorse or promote products or services of Licensee, or any third party. As an exception, the "BeOpen Python" logos available at http://www.pythonlabs.com/logos.html may be used according to the permissions granted on that web page.

7. By copying, installing or otherwise using the software, Licensee agrees to be bound by the terms and conditions of this License Agreement.

CNRI LICENSE AGREEMENT FOR 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 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.

ACCEPT

CWI LICENSE AGREEMENT FOR PYTHON 0.9.0 THROUGH 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 Licenses and Acknowledgements for Incorporated Software

This section is an incomplete, but growing list of licenses and acknowledgements for third-party software incorporated in the Python distribution.

C.3.1 Mersenne Twister

The `_random` module includes code based on a download from http://www.math.keio.ac.jp/matumoto/MT2002/emt19937ar.html. The following are the verbatim comments from the original code:

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

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

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

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

 1. Redistributions of source code must retain the above copyright
    notice, this list of conditions and the following disclaimer.

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

 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.keio.ac.jp/matumoto/emt.html
email: matumoto@math.keio.ac.jp
```

C.3.2 Sockets

The `socket` module uses the functions, `getaddrinfo()`, and `getnameinfo()`, which are coded in separate source files from the WIDE Project, http://www.wide.ad.jp/.

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

Redistribution and use in source and binary forms, with or without
modification, are permitted provided that the following conditions
are met:
1. Redistributions of source code must retain the above copyright
```

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

THIS SOFTWARE IS PROVIDED BY THE PROJECT AND CONTRIBUTORS ``AS IS'' AND
GAI_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 GAI_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 GAI_ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN GAI_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 Asynchronous socket services

The `asynchat` and `asyncore` modules contain the following notice:

```
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.3.5 Cookie management

The `http.cookies` 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.3.6 Execution tracing

The `trace` module contains the following notice:

```
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.3.7 UUencode and UUdecode functions

The uu module contains the following notice:

```
Copyright 1994 by Lance Ellinghouse
Cathedral City, California Republic, United States of America.
                        All Rights Reserved
Permission to use, copy, modify, and distribute this software and its
documentation for any purpose and without fee is hereby granted,
provided that the above copyright notice appear in all copies and that
both that copyright notice and this permission notice appear in
supporting documentation, and that the name of Lance Ellinghouse
not be used in advertising or publicity pertaining to distribution
of the software without specific, written prior permission.
LANCE ELLINGHOUSE DISCLAIMS ALL WARRANTIES WITH REGARD TO
THIS SOFTWARE, INCLUDING ALL IMPLIED WARRANTIES OF MERCHANTABILITY AND
FITNESS, IN NO EVENT SHALL LANCE ELLINGHOUSE CENTRUM BE LIABLE
FOR ANY SPECIAL, INDIRECT OR CONSEQUENTIAL DAMAGES OR ANY DAMAGES
WHATSOEVER RESULTING FROM LOSS OF USE, DATA OR PROFITS, WHETHER IN AN
ACTION OF CONTRACT, NEGLIGENCE OR OTHER TORTIOUS ACTION, ARISING OUT
OF OR IN CONNECTION WITH THE USE OR PERFORMANCE OF THIS SOFTWARE.

Modified by Jack Jansen, CWI, July 1995:
- Use binascii module to do the actual line-by-line conversion
  between ascii and binary. This results in a 1000-fold speedup. The C
  version is still 5 times faster, though.
- Arguments more compliant with Python standard
```

C.3.8 XML Remote Procedure Calls

The `xmlrpc.client` 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.3.9 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
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.3.10 Select 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.3.11 strtod and dtoa

The file `Python/dtoa.c`, which supplies C functions dtoa and strtod for conversion of C doubles to and from strings, is derived from the file of the same name by David M. Gay, currently available from http://www.netlib.org/fp/. The original file, as retrieved on March 16, 2009, contains the following copyright and licensing notice:

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

C.3.12 OpenSSL

The modules `hashlib`, `posix`, `ssl`, `crypt` use the OpenSSL library for added performance if made available by the operating system. Additionally, the Windows installers for Python include a copy of the OpenSSL libraries, so we include a copy of the OpenSSL license here:

```
LICENSE ISSUES
==============

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

OpenSSL License
---------------

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

```
    * LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
    * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
    * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
    * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
    * OF THE POSSIBILITY OF SUCH DAMAGE.
    * ====================================================================
    *
    * This product includes cryptographic software written by Eric Young
    * (eay@cryptsoft.com).  This product includes software written by Tim
    * Hudson (tjh@cryptsoft.com).
    *
    */

Original SSLeay License
-----------------------

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

The `pyexpat` extension is built using an included copy of the expat sources unless the build is configured `--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.3.14 libffi

The `_ctypes` extension is built using an included copy of the libffi sources unless the build is configured `--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.3.15 zlib

The `zlib` extension is built using an included copy of the zlib sources if the zlib version found on the system is too old to be used for the build:

```
Copyright (C) 1995-2011 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
```

APPENDIX D

Copyright

See *History and License* for complete license and permissions information.

Symbols

A

B

C

R

S

T

V

Z