Python Setup and Usage

출시 버전 *2.7.18*

Guido van Rossum
and the Python development team

5월 20, 2020

Contents

도큐멘테이션의 이 부분은 여러 플랫폼에서 파이썬 환경을 설정하고, 인터프리터를 호출하며, 파이썬으로 작업하기 더 쉽게 만드는 것들에 관한 일반적인 정보를 다루는데 할당되었습니다.

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

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

```
python myscript.py
```

1.1.1 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.

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

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

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

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

더 보기:

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

PEP 338 – Executing modules as scripts

버전 2.4에 추가.

버전 2.5에서 변경: The named module can now be located inside a package.

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

-

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

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

더 보기:

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

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

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

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

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

버전 2.5에서 변경: Directories and zipfiles containing a __main__.py file at the top level are now considered valid Python scripts.

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

더 보기:

tut-invoking

1.1.2 Generic options

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

버전 2.5에서 변경: The --help variant.

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

```
Python 2.5.1
```

버전 2.5에서 변경: The --version variant.

1.1.3 Miscellaneous options

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

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

버전 2.6에 추가.

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

버전 2.6에 추가.

-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.

버전 2.2에 추가.

-i

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

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

-O

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

-OO

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

-Q <arg>

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

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

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

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

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

더 보기:

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

PEP 238 – Changing the division operator

-R

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

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

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

See also *PYTHONHASHSEED*.

버전 2.6.8에 추가.

-s

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

버전 2.6에 추가.

더 보기:

PEP 370 – Per user site-packages directory

-S

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

-t

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

-u

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

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

See also *PYTHONUNBUFFERED*.

-v

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

-W `arg`

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

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

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

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

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

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

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

ignore Ignore all warnings.

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

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

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

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

error Raise an exception instead of printing a warning message.

The full form of argument is:

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

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

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

더 보기:

`warnings` – the warnings module

PEP 230 – Warning framework

PYTHONWARNINGS

-x

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

-3

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

버전 2.6에 추가.

See /howto/pyporting for more details.

1.1.4 Options you shouldn't use

-J

Reserved for use by Jython.

-U

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

```
from __future__ import unicode_literals
```

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

-X

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

1.2 Environment variables

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

PYTHONHOME

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

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

PYTHONPATH

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

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

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

An additional directory will be inserted in the search path in front of *PYTHONPATH* as described above under *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, OS X, OS/2, and RiscOS.

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

버전 2.6에 추가.

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

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

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

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

버전 2.6.8에 추가.

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

버전 2.6에 추가.

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

버전 2.6에 추가.

더 보기:

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`.

버전 2.6에 추가.

더 보기:

PEP 370 – Per user site-packages directory

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

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

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

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

버전 2.7.12에 추가.

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.

버전 2.6에서 변경: Previously, this variable was called `THREADDEBUG`.

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

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

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

버전 2.7.15에 추가.

PYTHONSHOWREFCOUNT

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

버전 2.7.15에 추가.

CHAPTER 2

Using Python on Unix platforms

2.1 Getting and installing the latest version of Python

2.1.1 On Linux

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

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

더 보기:

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

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

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

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

2.1.2 On FreeBSD and OpenBSD

- FreeBSD users, to add the package use:

```
pkg install python3
```

- OpenBSD users, to add the package use:

```
pkg_add -r python

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

(다음 페이지에 계속)

(이전 페이지에서 계속)

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

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

2.1.3 On OpenSolaris

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

2.2 Building Python

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

The build process consists in the usual

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

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

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

2.3 Python-related paths and files

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

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

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

2.4 Miscellaneous

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

```
$ chmod +x script
```

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

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

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

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

2.5 Editors and IDEs

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

Please go to Python Editors and Integrated Development Environments for a comprehensive list.

CHAPTER 3

Using Python on Windows

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

3.1 Installing Python

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

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

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

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

더 보기:

Python on XP 《7 Minutes to《Hello World!》》 by Richard Dooling, 2006

Installing on Windows in《Dive into Python: Python from novice to pro》 by Mark Pilgrim, 2004, ISBN 1-59059-356-1

For Windows users in《Installing Python》in《A Byte of Python》 by Swaroop C H, 2003

3.2 Alternative bundles

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

ActivePython Installer with multi-platform compatibility, documentation, PyWin32

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

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

3.3 Configuring Python

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

3.3.1 Excursus: Setting environment variables

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

In short, your path is:

My Computer ▸ *Properties* ▸ *Advanced* ▸ *Environment Variables*

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

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

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

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

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

```
echo %PATH%
```

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

더 보기:

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

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

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

3.3.2 Finding the Python executable

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

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

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

3.3.3 Finding modules

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

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

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

The end result of all this is:

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

3.3.4 Executing scripts

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

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

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

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

3. Redirect all Python files to the new executable:

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

3.4 Additional modules

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

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

3.4.1 PyWin32

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

- Component Object Model (COM)
- Win32 API calls
- Registry
- Event log
- Microsoft Foundation Classes (MFC) user interfaces

PythonWin is a sample MFC application shipped with PyWin32. It is an embeddable IDE with a built-in debugger.

더 보기:

Win32 How Do I…? by Tim Golden

Python and COM by David and Paul Boddie

3.4.2 Py2exe

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

3.4.3 WConio

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

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

3.5 Compiling Python on Windows

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

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

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

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

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

For extension modules, consult building-on-windows.

더 보기:

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 Other resources

더 보기:

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.

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

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

4.1 Getting and Installing MacPython

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

What you get after installing is a number of things:

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

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

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

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

4.1.1 How to run a Python script

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

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

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

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

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

4.1.2 Running scripts with a GUI

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

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

4.1.3 Configuration

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

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

4.2 The IDE

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

4.3 Installing Additional Python Packages

There are several methods to install additional Python packages:

- Packages can be installed via the standard Python distutils mode (`python setup.py install`).
- Many packages can also be installed via the **`setuptools`** extension or **`pip`** wrapper, see https://pip.pypa.io/.

4.4 GUI Programming on the Mac

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

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

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

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

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

4.5 Distributing Python Applications on the Mac

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

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

4.6 Other Resources

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

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

Another useful resource is the MacPython wiki:

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

APPENDIX A

용어집

>>> 대화형 셸의 기본 파이썬 프롬프트. 인터프리터에서 대화형으로 실행될 수 있는 코드 예에서 자주 볼 수 있다.

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

2to3 파이썬 2.x 코드를 파이썬 3.x 코드로 변환하려고 시도하는 도구인데, 소스를 파싱하고 파스 트리를 탐색해서 감지할 수 있는 대부분의 비호환성을 다룬다.

2to3 는 표준 라이브러리에서 `lib2to3` 로 제공된다; 독립적으로 실행할 수 있는 스크립트는 `Tools/scripts/2to3` 로 제공된다. 2to3-reference 를 보세요.

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

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

- 키워드 인자 *(keyword argument)*: 함수 호출 때 식별자가 앞에 붙은 인자 (예를 들어, `name=`) 또는 `**` 를 앞에 붙인 딕셔너리로 전달되는 인자. 예를 들어, 다음과 같은 `complex()` 호출에서 `3` 과 `5` 는 모두 키워드 인자다:

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

- 위치 인자 *(positional argument)*: 키워드 인자가 아닌 인자. 위치 인자들은 인자 목록의 처음에 나오거나 이터러블 의 앞에 `*` 를 붙여 전달할 수 있다. 예를 들어, 다음과 같은 호출에서 `3` 과 `5` 는 모두 위치 인자다.

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

인자는 함수 바의 이름 붙은 지역 변수에 대입된다. 이 대입에 적용되는 규칙들에 대해서는 calls 섹션을 보세요. 문법적으로, 어떤 표현식이건 인자로 사용될 수 있다; 구해진 값이 지역 변수에 대입된다.

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

attribute (어트리뷰트) 점표현식을 사용하는 이름으로 참조되는 객체와 결합한 값. 예를 들어, 객체 *o* 가 어트리뷰트 *a* 를 가지면, *o.a* 처럼 참조된다.

BDFL 자비로운 종신 독재자 (Benevolent Dictator For Life), 즉 Guido van Rossum, 파이썬의 창시자.

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

bytecode (바이트 코드) 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.

바이트 코드 명령어들의 목록은 dis 모듈 도큐멘테이션에 나온다.

class (클래스) 사용자 정의 객체들을 만들기 위한 주형. 클래스 정의는 보통 클래스의 인스턴스를 대상으로 연산하는 메서드 정의들을 포함한다.

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

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

complex number (복소수) 익숙한 실수 시스템의 확장인데, 모든 숫자가 실수부와 허수부의 합으로 표현된다. 허수부는 실수에 허수 단위 (`-1` 의 제곱근)를 곱한 것인데, 종종 수학에서는 `i` 로, 공학에서는 `j` 로 표기한다. 파이썬은 후자의 표기법을 쓰는 복소수를 기본 지원한다; 허수부는 `j` 접미사를 붙여서 표기한다, 예를 들어, `3+1j`. `math` 모듈의 복소수 버전이 필요하면, `cmath` 를 사용한다. 복소수의 활용은 꽤 수준 높은 수학적 기능이다. 필요하다고 느끼지 못한다면, 거의 확실히 무시해도 좋다.

context manager (컨텍스트 관리자) `__enter__()` 와 `__exit__()` 메서드를 정의함으로써 `with` 문에서 보이는 환경을 제어하는 객체. **PEP 343** 로 도입되었다.

CPython 파이썬 프로그래밍 언어의 규범적인 구현인데, python.org 에서 배포된다. 이 구현을 Jython 이나 IronPython 과 같은 다른 것들과 구별할 필요가 있을 때 용어《CPython》이 사용된다.

decorator (데코레이터) 다른 함수를 돌려주는 함수인데, 보통 `@wrapper` 문법을 사용한 함수 변환으로 적용된다. 데코레이터의 흔한 예는 `classmethod()` 과 `staticmethod()` 다.

데코레이터 문법은 단지 편의 문법일 뿐이다. 다음 두 함수 정의는 의미상으로 동등하다:

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

@staticmethod
```

(다음 페이지에 계속)

(이전 페이지에서 계속)

```
def f(...):
    ...
```

같은 개념이 클래스에도 존재하지만, 덜 자주 쓰인다. 데코레이터에 대한 더 자세한 내용은 함수 정의 와 클래스 정의 의 도큐멘테이션을 보면 된다.

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

디스크립터의 메서드들에 대한 자세한 내용은 descriptors 에 나온다.

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

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

docstring (독스트링) 클래스, 함수, 모듈에서 첫 번째 표현식으로 나타나는 문자열 리터럴. 스위트가 실행될 때는 무시되지만, 컴파일러에 의해 인지되어 둘러싼 클래스, 함수, 모듈의 `__doc__` 어트리뷰트로 삽입된다. 인트로스팩션을 통해 사용할 수 있으므로, 객체의 도큐멘테이션을 위한 규범적인 장소다.

duck-typing (덕 타이핑) 올바른 인터페이스를 가졌는지 판단하는데 객체의 형을 보지 않는 프로그래밍 스타일; 대신, 단순히 메서드나 어트리뷰트가 호출되거나 사용된다 (《오리처럼 보이고 오리처럼 꽥꽥댄다면, 그것은 오리다.》) 특정한 형 대신에 인터페이스를 강조함으로써, 잘 설계된 코드는 다형적인 치환을 허락함으로써 유연성을 개선할 수 있다. 덕 타이핑은 `type()` 이나 `isinstance()` 을 사용한 검사를 피한다. (하지만, 덕 타이핑이 추상 베이스 클래스 로 보완될 수 있음에 유의해야 한다.) 대신에, `hasattr()` 검사나 *EAFP* 프로그래밍을 쓴다.

EAFP 허락보다는 용서를 구하기가 쉽다 (Easier to ask for forgiveness than permission). 이 흔히 볼 수 있는 파이썬 코딩 스타일은, 올바른 키나 어트리뷰트의 존재를 가정하고, 그 가정이 틀리면 예외를 잡는다. 이 깔끔하고 빠른 스타일은 많은 `try` 와 `except` 문의 존재로 특징지어진다. 이 테크닉은 C와 같은 다른 많은 언어에서 자주 사용되는 *LBYL* 스타일과 대비된다.

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

extension module (확장 모듈) C 나 C++ 로 작성된 모듈인데, 파이썬의 C API를 사용해서 핵심이나 사용자 코드와 상호 작용한다.

file object (파일 객체) 하부 자원에 대해 파일 지향적 API (`read()` 나 `write()` 같은 메서드들) 를 드러내는 객체. 만들어진 방법에 따라, 파일 객체는 실제 디스크 상의 파일이나 다른 저장장치나 통신 장치 (예를 들어, 표준 입출력, 인-메모리 버퍼, 소켓, 파이프, 등등) 에 대한 액세스를 중계할 수 있다. 파일 객체는 파일류 객체 *(file-like objects)* 나 스트림 *(streams)* 이라고도 불린다.

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 (파일류 객체) 파일 객체 의 비슷한 말.

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

floor division (정수 나눗셈) 가장 가까운 정수로 내림하는 수학적 나눗셈. 정수 나눗셈 연산자는 `//` 다. 예를 들어, 표현식 `11 // 4` 의 값은 `2` 가 되지만, 실수 나눗셈은 `2.75` 를 돌려준다. `(-11) // 4` 가 `-2.75` 를 내림 한 `-3` 이 됨에 유의해야 한다. **PEP 238** 를 보세요.

function (함수) 호출자에게 어떤 값을 돌려주는 일련의 문장들. 없거나 그 이상의 인자 가 전달될 수 있는데, 바디의 실행에 사용될 수 있다. 파라미터 와 메서드 와 function 섹션도 보세요.

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

```
from __future__ import division
```

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

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

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

generator expression (제너레이터 표현식) 이터레이터를 돌려주는 표현식. 루프 변수와 범위를 정의하는 `for` 표현식과 생략 가능한 `if` 표현식이 뒤에 붙는 일반 표현식 처럼 보인다. 결합한 표현식은 둘러싼 함수를 위한 값들을 만들어낸다:

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

GIL 전역 인터프리터 록 을 보세요.

global interpreter lock (전역 인터프리터 록) 한 번에 오직 하나의 스레드가 파이썬 바이트 코드 를 실행하도록 보장하기 위해 *CPython* 인터프리터가 사용하는 메커니즘. (`dict` 와 같은 중요한 내장형들을 포함하는) 객체 모델이 묵시적으로 동시 액세스에 대해 안전하도록 만들어서 CPython 구현을 단순하게 만든다. 인터프리터 전체를 로킹하는 것은 인터프리터를 다중스레드화하기 쉽게 만드는 대신, 다중 프로세서 기계가 제공하는 병렬성의 많은 부분을 희생한다.

하지만, 어떤 확장 모듈들은, 표준이나 제삼자 모두, 압축이나 해싱 같은 계산 집약적인 작업을 수행할 때는 GIL 을 반납하도록 설계되었다. 또한, I/O를 할 때는 항상 GIL 을 반납한다.

(훨씬 더 미세하게 공유 데이터를 로킹하는) 《스레드에 자유로운(free-threaded)》 인터프리터를 만들고자 하는 과거의 노력은 성공적이지 못했는데, 흔한 단일 프로세서 경우의 성능 저하가 심하기 때문이다. 이 성능 이슈를 극복하는 것은 구현을 훨씬 복잡하게 만들어서 유지 비용이 더 들어갈 것으로 여겨지고 있다.

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

해시 가능성은 객체를 딕셔너리의 키나 집합의 멤버로 사용할 수 있게 하는데, 이 자료 구조들이 내부적으로 해시값을 사용하기 때문이다.

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

IDLE 파이썬을 위한 통합 개발 환경 (Integrated Development Environment). IDLE은 파이썬의 표준 배포판에 따라오는 기초적인 편집기와 인터프리터 환경이다.

immutable (불변) 고정된 값을 갖는 객체. 불변 객체는 숫자, 문자열, 튜플을 포함한다. 이런 객체들은 변경될 수 없다. 새 값을 저장하려면 새 객체를 만들어야 한다. 변하지 않는 해시값이 있어야 하는 곳에서 중요한 역할을 한다, 예를 들어, 딕셔너리의 키.

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

importing (임포팅) 한 모듈의 파이썬 코드가 다른 모듈의 파이썬 코드에서 사용될 수 있도록 하는 절차.

importer (임포터) 모듈을 찾기도 하고 로드 하기도 하는 객체; 동시에 파인더 이자 로더 객체다.

interactive (대화형) 파이썬은 대화형 인터프리터를 갖고 있는데, 인터프리터 프롬프트에서 문장과 표현식을 입력할 수 있고, 즉각 실행된 결과를 볼 수 있다는 뜻이다. 인자 없이 단지 `python` 을 실행하라 (컴퓨터의 주메뉴에서 선택하는 것도 가능할 수 있다). 새 아이디어를 검사하거나 모듈과 패키지를 들여다보는 매우 강력한 방법이다 (`help(x)` 를 기억하세요).

interpreted (인터프리티드) 바이트 코드 컴파일러의 존재 때문에 그 구분이 흐릿해지기는 하지만, 파이썬은 컴파일 언어가 아니라 인터프리터 언어다. 이것은 명시적으로 실행 파일을 만들지 않고도, 소스 파일을 직접 실행할 수 있다는 뜻이다. 그 프로그램이 좀 더 천천히 실행되기는 하지만, 인터프리터 언어는 보통 컴파일 언어보다 짧은 개발/디버깅 주기를 갖는다. 대화형 도 보세요.

iterable (이터러블) 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 (이터레이터) An object representing a stream of data. Repeated calls to the iterator's `next()` method return successive items in the stream. When no more data are available a `StopIteration` exception is raised instead. At this point, the iterator object is exhausted and any further calls to its `next()` method just raise `StopIteration` again. Iterators are required to have an `__iter__()` method that returns the iterator object itself so every iterator is also iterable and may be used in most places where other iterables are accepted. One notable exception is code which attempts multiple iteration passes. A container object (such as a `list`) produces a fresh new iterator each time you pass it to the `iter()` function or use it in a `for` loop. Attempting this with an iterator will just return the same exhausted iterator object used in the previous iteration pass, making it appear like an empty container.

typeiter 에 더 자세한 내용이 있다.

key function (키 함수) 키 함수 또는 콜레이션 (collation) 함수는 정렬 (sorting) 이나 배열 (ordering) 에 사용되는 값을 돌려주는 콜러블이다. 예를 들어, `locale.strxfrm()` 은 로케일 특정 방식을 따르는 정렬 키를 만드는 데 사용된다.

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

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

keyword argument (키워드 인자) 인자 를 보세요.

lambda (람다) 호출될 때 값이 구해지는 하나의 표현식 으로 구성된 이름 없는 인라인 함수. 람다 함수를 만드는 문법은 `lambda [parameters]: expression` 이다.

LBYL 뛰기 전에 보라 (Look before you leap). 이 코딩 스타일은 호출이나 조회를 하기 전에 명시적으로 사전 조건들을 검사한다. 이 스타일은 *EAFP* 접근법과 대비되고, 많은 `if` 문의 존재로 특징지어진다.

다중 스레드 환경에서, LBYL 접근법은 《보기》와 《뛰기》 간에 경쟁 조건을 만들게 될 위험이 있다. 예를 들어, 코드 `if key in mapping: return mapping[key]` 는 검사 후에, 하지만 조회 전에, 다른 스레드가 *key* 를 *mapping* 에서 제거하면 실패할 수 있다. 이런 이슈는 록이나 EAFP 접근법을 사용함으로써 해결될 수 있다.

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

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

loader (로더) 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.

magic method An informal synonym for *special method*.

mapping (매핑) 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 (메타 클래스) 클래스의 클래스. 클래스 정의는 클래스 이름, 클래스 딕셔너리, 베이스 클래스들의 목록을 만든다. 메타 클래스는 이 세 인자를 받아서 클래스를 만드는 책임을 진다. 대부분의 객체 지향형 프로그래밍 언어들은 기본 구현을 제공한다. 파이썬을 특별하게 만드는 것은 커스텀 메타 클래스를 만들 수 있다는 것이다. 대부분 사용자에게는 이 도구가 전혀 필요 없지만, 필요가 생길 때, 메타 클래스는 강력하고 우아한 해법을 제공한다. 어트리뷰트 액세스의 로깅 (logging), 스레드 안전성의 추가, 객체 생성 추적, 싱글톤 구현과 많은 다른 작업에 사용됐다.

metaclasses 에서 더 자세한 내용을 찾을 수 있다.

method (메서드) 클래스 바디 안에서 정의되는 함수. 그 클래스의 인스턴스의 어트리뷰트로서 호출되면, 그 메서드는 첫 번째 인자 (보통 `self` 라고 불린다) 로 인스턴스 객체를 받는다. 함수 와 중첩된 스코프 를 보세요.

method resolution order (메서드 결정 순서) 메서드 결정 순서는 조회하는 동안 멤버를 검색하는 베이스 클래스들의 순서다. 2.3 릴리스부터 파이썬 인터프리터에 사용된 알고리즘의 상세한 내용은 The Python 2.3 Method Resolution Order 를 보면 된다.

module (모듈) 파이썬 코드의 조직화 단위를 담당하는 객체. 모듈은 임의의 파이썬 객체들을 담는 이름 공간을 갖는다. 모듈은 임포팅 절차에 의해 파이썬으로 로드된다.

패키지 도 보세요.

MRO 메서드 결정 순서 를 보세요.

mutable (가변) 가변 객체는 값이 변할 수 있지만 `id()` 는 일정하게 유지한다. 불변 도 보세요.

named tuple (네임드 튜플) 인덱싱할 수 있는 요소들을 이름 붙은 어트리뷰트로도 액세스할 수 있는 모든 튜플류 클래스 (예를 들어, `time.localtime()` 은 *year* 가 `t[0]` 처럼 인덱스로도, `t.tm_year` 처럼 어트리뷰트로도 액세스할 수 있는 튜플류 객체를 돌려준다.)

네임드 튜플은 `time.struct_time` 같은 내장형일 수도, 일반 클래스 정의로 만들 수도 있다. 모든 기능이 구현된 네임드 튜플을 팩토리 함수 `collections.namedtuple()` 로도 만들 수 있다. 마지막 접근법은 `Employee(name='jones', title='programmer')` 와 같은 스스로 문서로 만드는 repr 과 같은 확장 기능도 자동 제공한다.

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

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

new-style class (뉴스타일 클래스) Any class which inherits from `object`. This includes all built-in types like `list` and `dict`. Only new-style classes can use Python's newer, versatile features like `__slots__`, descriptors, properties, and `__getattribute__()`.

More information can be found in newstyle.

object (객체) 상태 (어트리뷰트나 값) 를 갖고 동작 (메서드) 이 정의된 모든 데이터. 또한, 모든 뉴스타일 클래스 의 최종적인 베이스 클래스다.

package (패키지) 서브 모듈들이나, 재귀적으로 서브 패키지들을 포함할 수 있는 파이썬 모듈. 기술적으로, 패키지는 `__path__` 어트리뷰트가 있는 파이썬 모듈이다.

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

- 위치-키워드 *(positional-or-keyword)*: 위치 인자 나 키워드 인자 로 전달될 수 있는 인자를 지정한다. 이것이 기본 형태의 파라미터다, 예를 들어 다음에서 *foo* 와 *bar*:

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

- 위치-전용 *(positional-only)*: 위치로만 제공될 수 있는 인자를 지정한다. 파이썬은 위치-전용 파라미터를 정의하는 문법을 갖고 있지 않다. 하지만, 어떤 매장 함수들은 위치-전용 파라미터를 갖는다 (예를 들어, `abs()`).
- 가변-위치 *(var-positional)*: (다른 파라미터들에 의해서 이미 받아들여진 위치 인자들에 더해) 제공될 수 있는 위치 인자들의 임의의 시퀀스를 지정한다. 이런 파라미터는 파라미터 이름에 `*` 를 앞에 붙여서 정의될 수 있다, 예를 들어 다음에서 *args*:

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

- 가변-키워드 *(var-keyword)*: (다른 파라미터들에 의해서 이미 받아들여진 키워드 인자들에 더해) 제공될 수 있는 임의의 개수 키워드 인자들을 지정한다. 이런 파라미터는 파라미터 이름에 `**` 를 앞에 붙여서 정의될 수 있다, 예를 들어 위의 예 에서 *kwargs*.

파라미터는 선택적 인자들을 위한 기본값뿐만 아니라 선택적이거나 필수 인자들을 지정할 수 있다.

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

PEP 파이썬 개선 제안. PEP는 파이썬 커뮤니티에 정보를 제공하거나 파이썬 또는 그 프로세스 또는 환경에 대한 새로운 기능을 설명하는 설계 문서다. PEP는 제안된 기능에 대한 간결한 기술 사양 및 근거를 제공해야 한다.

PEP는 주요 새로운 기능을 제안하고 문제에 대한 커뮤니티 입력을 수집하며 파이썬에 들어간 설계 결정을 문서로 만들기 위한 기본 메커니즘이다. PEP 작성자는 커뮤니티 내에서 합의를 구축하고 반대 의견을 문서화 할 책임이 있다.

PEP 1 참조하세요.

positional argument (위치 인자) 인자 를 보세요.

Python 3000 (파이썬 3000) 파이썬 3.x 배포 라인의 별명 (버전 3의 배포가 먼 미래의 이야기던 시절에 만들어진 이름이다.) 이것을《Py3k》로 줄여 쓰기도 한다.

Pythonic (파이썬다운) 다른 언어들에서 일반적인 개념들을 사용해서 코드를 구현하는 대신, 파이썬 언어에서 가장 자주 사용되는 이디엄들을 가까이 따르는 아이디어나 코드 조작. 예를 들어, 파이썬에서 자주 쓰는 이디엄은 `for` 문을 사용해서 이터러블의 모든 요소로 루핑하는 것이다. 다른 많은 언어에는 이런 종류의 구성물이 없으므로, 파이썬에 익숙하지 않은 사람들은 대신에 숫자 카운터를 사용하기도 한다:

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

더 깔끔한, 파이썬다운 방법은 이렇다:

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

reference count (참조 횟수) 객체에 대한 참조의 개수. 객체의 참조 횟수가 0으로 떨어지면, 메모리가 반납된다. 참조 횟수 추적은 일반적으로 파이썬 코드에 노출되지는 않지만, *CPython* 구현의 핵심 요소다. `sys` 모듈은 특정 객체의 참조 횟수를 돌려주는 `getrefcount()` 을 정의한다.

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

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

slice (슬라이스) An object usually containing a portion of a *sequence*. A slice is created using the subscript notation, `[]` with colons between numbers when several are given, such as in `variable_name[1:3:5]`. The bracket (subscript) notation uses `slice` objects internally (or in older versions, `__getslice__()` and `__setslice__()`).

special method (특수 메서드) 파이썬이 형에 어떤 연산을, 덧셈 같은, 실행할 때 묵시적으로 호출되는 메서드. 이런 메서드는 두 개의 밑줄로 시작하고 끝나는 이름을 갖고 있다. 특수 메서드는 specialnames 에 문서로 만들어져 있다.

statement (문장) 문장은 스위트 (코드의《블록(block)》) 를 구성하는 부분이다. 문장은 표현식 이거나 키워드를 사용하는 여러 가지 구조물 중의 하나다. 가령 `if`, `while`, `for`.

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

triple-quoted string (삼중 따옴표 된 문자열) 따옴표 (《) 나 작은따옴표 (〈) 세 개로 둘러싸인 문자열. 그냥 따옴표 하나로 둘러싸인 문자열에 없는 기능을 제공하지는 않지만, 여러 가지 이유에서 쓸모가 있다. 이스케이프 되지 않은 작은따옴표나 큰따옴표를 문자열 안에 포함할 수 있도록 하고, 연결 문자를 쓰지 않고도 여러 줄에 걸칠 수 있는데, 독스트링을 쓸 때 특히 쓸모 있다.

type (형) 파이썬 객체의 형은 그것이 어떤 종류의 객체인지를 결정한다; 모든 객체는 형이 있다. 객체의 형은 `__class__` 어트리뷰트로 액세스할 수 있거나 `type(obj)` 로 얻을 수 있다.

universal newlines (유니버설 줄 넘김) A manner of interpreting text streams in which all of the following are recognized as ending a line: the Unix end-of-line convention `'\n'`, the Windows convention `'\r\n'`, and the old Macintosh convention `'\r'`. See **PEP 278** and **PEP 3116**, as well as `str.splitlines()` for an additional use.

virtual environment (가상 환경) 파이썬 사용자와 응용 프로그램이, 같은 시스템에서 실행되는 다른 파이썬 응용 프로그램들의 동작에 영향을 주지 않으면서, 파이썬 배포 패키지들을 설치하거나 업그레이드하는 것을 가능하게 하는, 협력적으로 격리된 실행 환경.

virtual machine (가상 기계) 소프트웨어만으로 정의된 컴퓨터. 파이썬의 가상 기계는 바이트 코드 컴파일러가 출력하는 바이트 코드 를 실행한다.

Zen of Python (파이썬 젠) 파이썬 디자인 원리와 철학들의 목록인데, 인어를 이해하고 사용하는 데 도움이 된다. 이 목록은 대화형 프롬프트에서 《`import this`》를 입력하면 보인다.

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 is an entirely volunteer effort, just like Python itself. If you want to contribute, please take a look at the reporting-bugs page for information on how to do so. New volunteers are always welcome!

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.

B.1 Contributors to the Python Documentation

Many people have contributed to the Python language, the Python standard library, and the Python documentation. See Misc/ACKS in the Python source distribution for a partial list of contributors.

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 https://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 https://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 https://www.zope.org/). In 2001, the Python Software Foundation (PSF, see https://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 https://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
2.1.2	2.1.1	2002	PSF	yes
2.1.3	2.1.2	2002	PSF	yes
2.2 and above	2.1.1	2001-now	PSF	yes

참고: 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

C.2.1 PSF LICENSE AGREEMENT FOR PYTHON 2.7.18

```
1. This LICENSE AGREEMENT is between the Python Software Foundation ("PSF"),␣
↪and
   the Individual or Organization ("Licensee") accessing and otherwise using␣
↪Python
   2.7.18 software in source or binary form and its associated documentation.

2. Subject to the terms and conditions of this License Agreement, PSF hereby
   grants Licensee a nonexclusive, royalty-free, world-wide license to␣
↪reproduce,
   analyze, test, perform and/or display publicly, prepare derivative works,
   distribute, and otherwise use Python 2.7.18 alone or in any derivative
   version, provided, however, that PSF's License Agreement and PSF's notice␣
↪of
   copyright, i.e., "Copyright © 2001-2020 Python Software Foundation; All␣
↪Rights
   Reserved" are retained in Python 2.7.18 alone or in any derivative version
   prepared by Licensee.

3. In the event Licensee prepares a derivative work that is based on or
   incorporates Python 2.7.18 or any part thereof, and wants to make the
   derivative work available to others as provided herein, then Licensee␣
↪hereby
   agrees to include in any such work a brief summary of the changes made to␣
↪Python
   2.7.18.

4. PSF is making Python 2.7.18 available to Licensee on an "AS IS" basis.
   PSF MAKES NO REPRESENTATIONS OR WARRANTIES, EXPRESS OR IMPLIED.  BY WAY OF
   EXAMPLE, BUT NOT LIMITATION, PSF MAKES NO AND DISCLAIMS ANY REPRESENTATION␣
↪OR
   WARRANTY OF MERCHANTABILITY OR FITNESS FOR ANY PARTICULAR PURPOSE OR THAT␣
↪THE
   USE OF PYTHON 2.7.18 WILL NOT INFRINGE ANY THIRD PARTY RIGHTS.

5. PSF SHALL NOT BE LIABLE TO LICENSEE OR ANY OTHER USERS OF PYTHON 2.7.18
   FOR ANY INCIDENTAL, SPECIAL, OR CONSEQUENTIAL DAMAGES OR LOSS AS A RESULT␣
↪OF
   MODIFYING, DISTRIBUTING, OR OTHERWISE USING PYTHON 2.7.18, OR ANY␣
↪DERIVATIVE
   THEREOF, EVEN IF ADVISED OF THE POSSIBILITY THEREOF.
```

```
6. This License Agreement will automatically terminate upon a material breach␣
↪of
   its terms and conditions.

7. Nothing in this License Agreement shall be deemed to create any␣
↪relationship
   of agency, partnership, or joint venture between PSF and Licensee.  This␣
↪License
   Agreement does not grant permission to use PSF trademarks or trade name in␣
↪a
   trademark sense to endorse or promote products or services of Licensee, or␣
↪any
   third party.

8. By copying, installing or otherwise using Python 2.7.18, Licensee agrees
   to be bound by the terms and conditions of this License Agreement.
```

C.2.2 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
```

(다음 페이지에 계속)

(이전 페이지에서 계속)

```
   http://www.pythonlabs.com/logos.html may be used according to the permissions
   granted on that web page.

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

C.2.3 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
```

(다음 페이지에 계속)

(이전 페이지에서 계속)

```
   under the GNU General Public License (GPL), the law of the Commonwealth of
   Virginia shall govern this License Agreement only as to issues arising under or
   with respect to Paragraphs 4, 5, and 7 of this License Agreement.  Nothing in
   this License Agreement shall be deemed to create any relationship of agency,
   partnership, or joint venture between CNRI and Licensee.  This License Agreement
   does not grant permission to use CNRI trademarks or trade name in a trademark
   sense to endorse or promote products or services of Licensee, or any third
   party.

8. By clicking on the "ACCEPT" button where indicated, or by copying, installing
   or otherwise using Python 1.6.1, Licensee agrees to be bound by the terms and
   conditions of this License Agreement.
```

C.2.4 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.sci.hiroshima-u.ac.jp/~m-mat/MT/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).
```

(다음 페이지에 계속)

(이전 페이지에서 계속)

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

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

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

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

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

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


Any feedback is very welcome.
http://www.math.sci.hiroshima-u.ac.jp/~m-mat/MT/emt.html
email: m-mat @ math.sci.hiroshima-u.ac.jp (remove space)
```

C.3.2 Sockets

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.
```

(다음 페이지에 계속)

(이전 페이지에서 계속)

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

C.3.3 Floating point exception control

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

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

C.3.4 MD5 message digest algorithm

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

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

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

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

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

L. Peter Deutsch
ghost@aladdin.com

Independent implementation of MD5 (RFC 1321).

This code implements the MD5 Algorithm defined in RFC 1321, whose
text is available at
      http://www.ietf.org/rfc/rfc1321.txt
The code is derived from the text of the RFC, including the test suite
(section A.5) but excluding the rest of Appendix A.  It does not include
any code or documentation that is identified in the RFC as being
copyrighted.

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

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

C.3.5 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.3.6 Cookie management

The `Cookie` module contains the following notice:

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

               All Rights Reserved

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

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

C.3.7 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.3.8 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
```

(다음 페이지에 계속)

(이전 페이지에서 계속)

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

C.3.9 XML Remote Procedure Calls

The `xmlrpclib` module contains the following notice:

```
    The XML-RPC client interface is

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

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

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

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

C.3.10 test_epoll

The `test_epoll` contains the following notice:

```
Copyright (c) 2001-2006 Twisted Matrix Laboratories.

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

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

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND,
EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF
```

(다음 페이지에 계속)

(이전 페이지에서 계속)

```
MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND
NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE
LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION
OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION
WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
```

C.3.11 Select kqueue

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

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

Redistribution and use in source and binary forms, with or without
modification, are permitted provided that the following conditions
are met:
1. Redistributions of source code must retain the above copyright
   notice, this list of conditions and the following disclaimer.
2. Redistributions in binary form must reproduce the above copyright
   notice, this list of conditions and the following disclaimer in the
   documentation and/or other materials provided with the distribution.

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

C.3.12 strtod 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
```

(다음 페이지에 계속)

(이전 페이지에서 계속)

```
 * WARRANTY.  IN PARTICULAR, NEITHER THE AUTHOR NOR LUCENT MAKES ANY
 * REPRESENTATION OR WARRANTY OF ANY KIND CONCERNING THE MERCHANTABILITY
 * OF THIS SOFTWARE OR ITS FITNESS FOR ANY PARTICULAR PURPOSE.
 *
 ***************************************************************/
```

C.3.13 OpenSSL

The modules `hashlib`, `posix`, `ssl`, `crypt` use the OpenSSL library for added performance if made available by the operating system. Additionally, the Windows and Mac OS X installers for Python may 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:
```

(다음 페이지에 계속)

(이전 페이지에서 계속)

```
   *    "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
```

(다음 페이지에 계속)

(이전 페이지에서 계속)

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

C.3.14 expat

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.3.15 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.3.16 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) 1995-2010 Jean-loup Gailly and Mark Adler

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

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

1. The origin of this software must not be misrepresented; you must not
   claim that you wrote the original software. If you use this software
   in a product, an acknowledgment in the product documentation would be
   appreciated but is not required.

2. Altered source versions must be plainly marked as such, and must not be
   misrepresented as being the original software.

3. This notice may not be removed or altered from any source distribution.

Jean-loup Gailly        Mark Adler
jloup@gzip.org          madler@alumni.caltech.edu
```

APPENDIX D

저작권

파이썬과 이 도큐멘테이션은:

전체 라이센스 및 사용 권한 정보는 *History and License* 에서 제공한다.

색인

Non-alphabetical

A

B

C

D

E