

---

# Python Setup and Usage

*Δημοσίευση 3.10.18*

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

Ιουλίου 08, 2025

**Python Software Foundation  
Email: [docs@python.org](mailto:docs@python.org)**



<b>1</b>	<b>Command line and environment</b>	<b>3</b>
1.1	Command line	3
1.1.1	Interface options	3
1.1.2	Generic options	5
1.1.3	Miscellaneous options	6
1.1.4	Options you shouldn't use	9
1.2	Environment variables	9
1.2.1	Debug-mode variables	13
<b>2</b>	<b>Χρήση της Python σε πλατφόρμες Unix</b>	<b>15</b>
2.1	Λήψη και εγκατάσταση της πιο πρόσφατης έκδοσης Python	15
2.1.1	Σε Linux	15
2.1.2	Σε FreeBSD και OpenBSD	15
2.1.3	On OpenSolaris	16
2.2	Μεταγλώττιση της Python	16
2.3	Διαδρομές και αρχεία που σχετίζονται με την Python	16
2.4	Διάφορα	17
2.5	Custom OpenSSL	17
<b>3</b>	<b>Configure Python</b>	<b>19</b>
3.1	Configure Options	19
3.1.1	General Options	19
3.1.2	Install Options	21
3.1.3	Performance options	21
3.1.4	Python Debug Build	22
3.1.5	Debug options	23
3.1.6	Linker options	23
3.1.7	Libraries options	24
3.1.8	Security Options	25
3.1.9	macOS Options	25
3.2	Python Build System	26
3.2.1	Main files of the build system	26
3.2.2	Main build steps	26
3.2.3	Main Makefile targets	26
3.2.4	C extensions	27
3.3	Compiler and linker flags	27
3.3.1	Preprocessor flags	27
3.3.2	Compiler flags	28
3.3.3	Linker flags	29
<b>4</b>	<b>Using Python on Windows</b>	<b>31</b>

4.1	The full installer . . . . .	31
4.1.1	Installation steps . . . . .	31
4.1.2	Removing the MAX_PATH Limitation . . . . .	33
4.1.3	Installing Without UI . . . . .	33
4.1.4	Installing Without Downloading . . . . .	35
4.1.5	Modifying an install . . . . .	35
4.2	The Microsoft Store package . . . . .	36
4.2.1	Known issues . . . . .	36
4.3	The nuget.org packages . . . . .	37
4.4	The embeddable package . . . . .	38
4.4.1	Python Application . . . . .	38
4.4.2	Embedding Python . . . . .	38
4.5	Alternative bundles . . . . .	39
4.6	Configuring Python . . . . .	39
4.6.1	Excursus: Setting environment variables . . . . .	39
4.6.2	Finding the Python executable . . . . .	40
4.7	UTF-8 mode . . . . .	40
4.8	Python Launcher for Windows . . . . .	41
4.8.1	Getting started . . . . .	41
4.8.2	Shebang Lines . . . . .	42
4.8.3	Arguments in shebang lines . . . . .	43
4.8.4	Customization . . . . .	43
4.8.5	Diagnostics . . . . .	44
4.9	Finding modules . . . . .	45
4.10	Additional modules . . . . .	46
4.10.1	PyWin32 . . . . .	46
4.10.2	cx_Freeze . . . . .	47
4.11	Compiling Python on Windows . . . . .	47
4.12	Other Platforms . . . . .	47
<b>5</b>	<b>Using Python on a Mac</b>	<b>49</b>
5.1	Getting and Installing MacPython . . . . .	49
5.1.1	How to run a Python script . . . . .	50
5.1.2	Running scripts with a GUI . . . . .	50
5.1.3	Configuration . . . . .	50
5.2	The IDE . . . . .	50
5.3	Installing Additional Python Packages . . . . .	51
5.4	GUI Programming on the Mac . . . . .	51
5.5	Distributing Python Applications on the Mac . . . . .	51
5.6	Other Resources . . . . .	51
<b>6</b>	<b>Επεξεργαστές Κειμένου και IDEs</b>	<b>53</b>
<b>A'</b>	<b>Γλωσσάρι</b>	<b>55</b>
<b>B'</b>	<b>About these documents</b>	<b>71</b>
B'.1	Contributors to the Python Documentation . . . . .	71
<b>Γ'</b>	<b>Ιστορία και Άδεια</b>	<b>73</b>
Γ'.1	Η ιστορία του λογισμικού . . . . .	73
Γ'.2	Όροι και προϋποθέσεις για την πρόσβαση ή την χρήση της Python με άλλους τρόπους . . . . .	74
Γ'.2.1	PSF LICENSE AGREEMENT FOR PYTHON 3.10.18 . . . . .	74
Γ'.2.2	ΣΥΜΦΩΝΙΑ ΑΔΕΙΑΣ BEOPEN.COM ΓΙΑ PYTHON 2.0 . . . . .	75
Γ'.2.3	ΣΥΜΦΩΝΙΑ ΑΔΕΙΑΣ CNRI ΓΙΑ PYTHON 1.6.1 . . . . .	76
Γ'.2.4	ΣΥΜΦΩΝΙΑ ΑΔΕΙΑΣ CWI ΓΙΑ PYTHON 0.9.0 ΕΩΣ 1.2 . . . . .	77
Γ'.2.5	ZERO-CLAUSE BSD LICENSE FOR CODE IN THE PYTHON 3.10.18 DOCUMENTATION . . . . .	77
Γ'.3	Άδειες και Ευχαριστίες για Ενσωματωμένο Λογισμικό . . . . .	78
Γ'.3.1	Mersenne Twister . . . . .	78

Γ'.3.2	Sockets . . . . .	79
Γ'.3.3	Ασύγχρονες socket υπηρεσίες . . . . .	79
Γ'.3.4	Διαχείριση Cookie . . . . .	80
Γ'.3.5	Ανίχνευση εκτέλεσης . . . . .	80
Γ'.3.6	Συναρτήσεις UUencode και UUdecode . . . . .	81
Γ'.3.7	Κλήσεις Απομακρυσμένης Διαδικασίας XML . . . . .	81
Γ'.3.8	test_epoll . . . . .	82
Γ'.3.9	Επιλογή kqueue . . . . .	82
Γ'.3.10	SipHash24 . . . . .	83
Γ'.3.11	strtod και dtoa . . . . .	83
Γ'.3.12	OpenSSL . . . . .	84
Γ'.3.13	expat . . . . .	86
Γ'.3.14	libffi . . . . .	86
Γ'.3.15	zlib . . . . .	87
Γ'.3.16	cfuhash . . . . .	87
Γ'.3.17	libmpdec . . . . .	88
Γ'.3.18	W3C C14N σουίτα δοκιμής . . . . .	88
Γ'.3.19	Audioop . . . . .	89
<b>Δ' Copyright</b>		<b>91</b>
<b>Ευρετήριο</b>		<b>93</b>



Αυτό το μέρος της τεκμηρίωσης είναι αφιερωμένο σε γενικές πληροφορίες σχετικά με τη ρύθμιση του περιβάλλοντος της Python σε διαφορετικές πλατφόρμες, την εκκίνηση του διεργαστή και πράγματα που κάνουν το να δουλεύεις με την Python ευκολότερα.





---

## Command line and environment

---

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

**Λεπτομέρεια υλοποίησης CPython:** 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 [-bBdEhiIOqsSuvVWx?] [-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).

Raises an auditing event `cpython.run_command` with argument `command`.

**-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 absolute 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 (including namespace packages) 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.

---

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

---

If this option is given, the first element of `sys.argv` will be the full path to the module file (while the module file is being located, the first element will be set to `"-m"`). As with the `-c` option, the current directory will be added to the start of `sys.path`.

`-I` option can be used to run the script in isolated mode where `sys.path` contains neither the current directory nor the user's site-packages directory. All `PYTHON*` environment variables are ignored, too.

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

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

Raises an auditing event `cpython.run_module` with argument `module-name`.

**Δείτε επίσης:**

**runpy.run\_module()** Equivalent functionality directly available to Python code

**PEP 338** – Executing modules as scripts

Αλλαξε στην έκδοση 3.1: Supply the package name to run a `__main__` submodule.

Αλλαξε στην έκδοση 3.4: namespace packages are also supported

–

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`.

Raises an auditing event `cpython.run_stdin` with no arguments.

**<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.

`-I` option can be used to run the script in isolated mode where `sys.path` contains neither the script's directory nor the user's site-packages directory. All `PYTHON*` environment variables are ignored, too.

Raises an auditing event `cpython.run_file` with argument `filename`.

**Δείτε επίσης:**

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

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`. Also, tab-completion and history editing is automatically enabled, if available on your platform (see `rlcompleter-config`).

**Δείτε επίσης:**

tut-invoking

Αλλαξε στην έκδοση 3.4: Automatic enabling of tab-completion and history editing.

## 1.1.2 Generic options

`-?`

`-h`

`--help`

Print a short description of all command line options.

`-V`

`--version`

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

```
Python 3.8.0b2+
```

When given twice, print more information about the build, like:

```
Python 3.8.0b2+ (3.8:0c076caaa8, Apr 20 2019, 21:55:00)
[GCC 6.2.0 20161005]
```

Νέο στην έκδοση 3.6: The `-VV` option.

### 1.1.3 Miscellaneous options

**-b**

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

Άλλαξε στην έκδοση 3.5: Affects comparisons of `bytes` with `int`.

**-B**

If given, Python won't try to write `.pyc` files on the import of source modules. See also [`PYTHONDONTWRITEBYTECODE`](#).

**--check-hash-based-pycs** `default|always|never`

Control the validation behavior of hash-based `.pyc` files. See `pyc`-invalidation. When set to `default`, checked and unchecked hash-based bytecode cache files are validated according to their default semantics. When set to `always`, all hash-based `.pyc` files, whether checked or unchecked, are validated against their corresponding source file. When set to `never`, hash-based `.pyc` files are not validated against their corresponding source files.

The semantics of timestamp-based `.pyc` files are unaffected by this option.

**-d**

Turn on parser debugging output (for expert only, depending on compilation options). See also [`PYTHONDEBUG`](#).

**-E**

Ignore all `PYTHON*` environment variables, e.g. [`PYTHONPATH`](#) and [`PYTHONHOME`](#), that might be set.

**-i**

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

This can be useful to inspect global variables or a stack trace when a script raises an exception. See also [`PYTHONINSPECT`](#).

**-I**

Run Python in isolated mode. This also implies `-E` and `-s`. In isolated mode `sys.path` contains neither the script's directory nor the user's site-packages directory. All `PYTHON*` environment variables are ignored, too. Further restrictions may be imposed to prevent the user from injecting malicious code.

Νέο στην έκδοση 3.4.

**-O**

Remove assert statements and any code conditional on the value of `__debug__`. Augment the filename for compiled (*bytecode*) files by adding `.opt-1` before the `.pyc` extension (see [PEP 488](#)). See also [`PYTHONOPTIMIZE`](#).

Άλλαξε στην έκδοση 3.5: Modify `.pyc` filenames according to [PEP 488](#).

**-OO**

Do `-O` and also discard docstrings. Augment the filename for compiled (*bytecode*) files by adding `.opt-2` before the `.pyc` extension (see [PEP 488](#)).

Άλλαξε στην έκδοση 3.5: Modify `.pyc` filenames according to [PEP 488](#).

**-q**

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

Νέο στην έκδοση 3.2.

**-R**

Turn on hash randomization. This option only has an effect if the [`PYTHONHASHSEED`](#) environment variable is set to 0, since hash randomization is enabled by default.

On previous versions of Python, this option turns on hash randomization, so that the `__hash__()` values of `str` and `bytes` 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.

Hash randomization 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.

`PYTHONHASHSEED` allows you to set a fixed value for the hash seed secret.

Άλλαξε στην έκδοση 3.7: The option is no longer ignored.

Νέο στην έκδοση 3.2.3.

**-s**

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

**Δείτε επίσης:**

**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. Also disable these manipulations if `site` is explicitly imported later (call `site.main()` if you want them to be triggered).

**-u**

Force the `stdout` and `stderr` streams to be unbuffered. This option has no effect on the `stdin` stream.

See also `PYTHONUNBUFFERED`.

Άλλαξε στην έκδοση 3.7: The text layer of the `stdout` and `stderr` streams now is unbuffered.

**-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.

Άλλαξε στην έκδοση 3.10: The `site` module reports the site-specific paths and `.pth` files being processed.

See also `PYTHONVERBOSE`.

**-W arg**

Warning control. Python's warning machinery by default prints warning messages to `sys.stderr`.

The simplest settings apply a particular action unconditionally to all warnings emitted by a process (even those that are otherwise ignored by default):

```
-Wdefault # Warn once per call location
-Werror   # Convert to exceptions
-Walways  # Warn every time
-Wmodule  # Warn once per calling module
-Wonce    # Warn once per Python process
-Wignore  # Never warn
```

The action names can be abbreviated as desired and the interpreter will resolve them to the appropriate action name. For example, `-Wi` is the same as `-Wignore`.

The full form of argument is:

```
action:message:category:module:lineno
```

Empty fields match all values; trailing empty fields may be omitted. For example `-W ignore::DeprecationWarning` ignores all `DeprecationWarning` warnings.

The *action* field is as explained above but only applies to warnings that match the remaining fields.

The *message* field must match the whole warning message; this match is case-insensitive.

The *category* field matches the warning category (ex: `DeprecationWarning`). This must be a class name; the match test whether the actual warning category of the message is a subclass of the specified warning category.

The *module* field matches the (fully qualified) module name; this match is case-sensitive.

The *lineno* field matches the line number, where zero matches all line numbers and is thus equivalent to an omitted line number.

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

Warnings can also be controlled using the *PYTHONWARNINGS* environment variable and from within a Python program using the `warnings` module. For example, the `warnings.filterwarnings()` function can be used to use a regular expression on the warning message.

See `warning-filter` and `describing-warning-filters` for more details.

**-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.

**-X**

Reserved for various implementation-specific options. CPython currently defines the following possible values:

- `-X faulthandler` to enable `faulthandler`;
- `-X showrefcount` to output the total reference count and number of used memory blocks when the program finishes or after each statement in the interactive interpreter. This only works on *debug builds*.
- `-X tracemalloc` to start tracing Python memory allocations using the `tracemalloc` module. By default, only the most recent frame is stored in a traceback of a trace. Use `-X tracemalloc=NFRAME` to start tracing with a traceback limit of *NFRAME* frames. See the `tracemalloc.start()` for more information.
- `-X int_max_str_digits` configures the integer string conversion length limitation. See also *PYTHONINTMAXSTRDIGITS*.
- `-X importtime` to show how long each import takes. It shows module name, cumulative time (including nested imports) and self time (excluding nested imports). Note that its output may be broken in multi-threaded application. Typical usage is `python3 -X importtime -c 'import asyncio'`. See also *PYTHONPROFILEIMPORTTIME*.
- `-X dev`: enable Python Development Mode, introducing additional runtime checks that are too expensive to be enabled by default.
- `-X utf8` enables the Python UTF-8 Mode. `-X utf8=0` explicitly disables Python UTF-8 Mode (even when it would otherwise activate automatically).
- `-X pycache_prefix=PATH` enables writing `.pyc` files to a parallel tree rooted at the given directory instead of to the code tree. See also *PYTHONPYCACHEPREFIX*.
- `-X warn_default_encoding` issues a `EncodingWarning` when the locale-specific default encoding is used for opening files. See also *PYTHONWARNDEFAULTENCODING*.

It also allows passing arbitrary values and retrieving them through the `sys._xoptions` dictionary.

Αλλάξε στην έκδοση 3.2: The `-X` option was added.

Νέο στην έκδοση 3.3: The `-X faulthandler` option.

Νέο στην έκδοση 3.4: The `-X showrefcount` and `-X tracemalloc` options.

Νέο στην έκδοση 3.6: The `-X showalloccount` option.

Νέο στην έκδοση 3.7: The `-X importtime`, `-X dev` and `-X utf8` options.

Νέο στην έκδοση 3.8: The `-X pycache_prefix` option. The `-X dev` option now logs `close()` exceptions in `io.IOBase` destructor.

Αλλάξε στην έκδοση 3.9: Using `-X dev` option, check *encoding* and *errors* arguments on string encoding and decoding operations.

The `-X showalloccount` option has been removed.

Νέο στην έκδοση 3.10: The `-X warn_default_encoding` option.

Νέο στην έκδοση 3.10.7: The `-X int_max_str_digits` option.

Deprecated since version 3.9, removed in version 3.10: The `-X oldparser` option.

## 1.1.4 Options you shouldn't use

`-J`

Reserved for use by *Jython*.

## 1.2 Environment variables

These environment variables influence Python's behavior, they are processed before the command-line switches other than `-E` or `-I`. 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/pythonversion* (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`.

### **PYTHONPLATLIBDIR**

If this is set to a non-empty string, it overrides the `sys.platlibdir` value.

Νέο στην έκδοση 3.9.

### **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` and the hook `sys.__interactivehook__` in this file.

Raises an auditing event `cpython.run_startup` with the filename as the argument when called on startup.

### **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.

### **PYTHONBREAKPOINT**

If this is set, it names a callable using dotted-path notation. The module containing the callable will be imported and then the callable will be run by the default implementation of `sys.breakpointhook()` which itself is called by built-in `breakpoint()`. If not set, or set to the empty string, it is equivalent to the value `«pdb.set_trace»`. Setting this to the string `«0»` causes the default implementation of `sys.breakpointhook()` to do nothing but return immediately.

Νέο στην έκδοση 3.7.

### **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.

Raises an auditing event `cpython.run_stdin` with no arguments.

Άλλαξε στην έκδοση 3.10.15: (also 3.9.20, and 3.8.20) Emits audit events.

### **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 and macOS.

### **PYTHONDONTWRITEBYTECODE**

If this is set to a non-empty string, Python won't try to write `.pyc` files on the import of source modules. This is equivalent to specifying the `-B` option.

### **PYTHONPYCACHEPREFIX**

If this is set, Python will write `.pyc` files in a mirror directory tree at this path, instead of in `__pycache__` directories within the source tree. This is equivalent to specifying the `-X pycache_prefix=PATH` option.

Νέο στην έκδοση 3.8.

### **PYTHONHASHSEED**

If this variable is not set or set to `random`, a random value is used to seed the hashes of `str` and `bytes` 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 disable hash randomization.

Νέο στην έκδοση 3.2.3.

### **PYTHONINTMAXSTRDIGITS**

If this variable is set to an integer, it is used to configure the interpreter's global integer string conversion length limitation.

Νέο στην έκδοση 3.10.7.

### **PYTHONIOENCODING**

If this is set before running the interpreter, it overrides the encoding used for `stdin/stdout/stderr`, in the syntax `encodingname:errorhandler`. Both the `encodingname` and the `:errorhandler` parts are optional and have the same meaning as in `str.encode()`.

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



Αλλάξε στην έκδοση 3.4: The `encodingname` part is now optional.

Αλλάξε στην έκδοση 3.6: On Windows, the encoding specified by this variable is ignored for interactive console buffers unless `PYTHONLEGACYWINDOWSSTDIO` is also specified. Files and pipes redirected through the standard streams are not affected.

#### **PYTHONNOUSERSITE**

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

Δείτε επίσης:

**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`.

Δείτε επίσης:

**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 macOS.

#### **PYTHONWARNINGS**

This is equivalent to the `-W` option. If set to a comma separated string, it is equivalent to specifying `-W` multiple times, with filters later in the list taking precedence over those earlier in the list.

The simplest settings apply a particular action unconditionally to all warnings emitted by a process (even those that are otherwise ignored by default):

```
PYTHONWARNINGS=default    # Warn once per call location
PYTHONWARNINGS=error      # Convert to exceptions
PYTHONWARNINGS=always     # Warn every time
PYTHONWARNINGS=module     # Warn once per calling module
PYTHONWARNINGS=once       # Warn once per Python process
PYTHONWARNINGS=ignore     # Never warn
```

See warning-filter and describing-warning-filters for more details.

#### **PYTHONFAULTHANDLER**

If this environment variable is set to a non-empty string, `faulthandler.enable()` is called at startup: install a handler for `SIGSEGV`, `SIGFPE`, `SIGABRT`, `SIGBUS` and `SIGILL` signals to dump the Python traceback. This is equivalent to `-X faulthandler` option.

Νέο στην έκδοση 3.3.

#### **PYTHONTRACEMALLOC**

If this environment variable is set to a non-empty string, start tracing Python memory allocations using the `tracemalloc` module. The value of the variable is the maximum number of frames stored in a traceback of a trace. For example, `PYTHONTRACEMALLOC=1` stores only the most recent frame. See the `tracemalloc.start()` for more information.

Νέο στην έκδοση 3.4.

#### **PYTHONPROFILEIMPORTTIME**

If this environment variable is set to a non-empty string, Python will show how long each import takes. This is exactly equivalent to setting `-X importtime` on the command line.

Νέο στην έκδοση 3.7.

#### **PYTHONASYNCIODEBUG**

If this environment variable is set to a non-empty string, enable the debug mode of the `asyncio` module.

Νέο στην έκδοση 3.4.

### **PYTHONMALLOC**

Set the Python memory allocators and/or install debug hooks.

Set the family of memory allocators used by Python:

- `default`: use the default memory allocators.
- `malloc`: use the `malloc()` function of the C library for all domains (`PYMEM_DOMAIN_RAW`, `PYMEM_DOMAIN_MEM`, `PYMEM_DOMAIN_OBJ`).
- `pymalloc`: use the `pymalloc` allocator for `PYMEM_DOMAIN_MEM` and `PYMEM_DOMAIN_OBJ` domains and use the `malloc()` function for the `PYMEM_DOMAIN_RAW` domain.

Install debug hooks:

- `debug`: install debug hooks on top of the default memory allocators.
- `malloc_debug`: same as `malloc` but also install debug hooks.
- `pymalloc_debug`: same as `pymalloc` but also install debug hooks.

Άλλαξε στην έκδοση 3.7: Added the "default" allocator.

Νέο στην έκδοση 3.6.

### **PYTHONMALLOCSTATS**

If set to a non-empty string, Python will print statistics of the `pymalloc` memory allocator every time a new `pymalloc` object arena is created, and on shutdown.

This variable is ignored if the `PYTHONMALLOC` environment variable is used to force the `malloc()` allocator of the C library, or if Python is configured without `pymalloc` support.

Άλλαξε στην έκδοση 3.6: This variable can now also be used on Python compiled in release mode. It now has no effect if set to an empty string.

### **PYTHONLEGACYWINDOWSFSENCODING**

If set to a non-empty string, the default *filesystem encoding and error handler* mode will revert to their pre-3.6 values of "mbcs" and "replace", respectively. Otherwise, the new defaults "utf-8" and "surrogatepass" are used.

This may also be enabled at runtime with `sys._enablelegacywindowsfsencoding()`.

Availability: Windows.

Νέο στην έκδοση 3.6: See [PEP 529](#) for more details.

### **PYTHONLEGACYWINDOWSSTDIO**

If set to a non-empty string, does not use the new console reader and writer. This means that Unicode characters will be encoded according to the active console code page, rather than using utf-8.

This variable is ignored if the standard streams are redirected (to files or pipes) rather than referring to console buffers.

Availability: Windows.

Νέο στην έκδοση 3.6.

### **PYTHONCOERCECLOCALE**

If set to the value 0, causes the main Python command line application to skip coercing the legacy ASCII-based C and POSIX locales to a more capable UTF-8 based alternative.

If this variable is *not* set (or is set to a value other than 0), the `LC_ALL` locale override environment variable is also not set, and the current locale reported for the `LC_CTYPE` category is either the default C locale, or else the explicitly ASCII-based POSIX locale, then the Python CLI will attempt to configure the following locales for the `LC_CTYPE` category in the order listed before loading the interpreter runtime:

- `C.UTF-8`
- `C.utf8`
- `UTF-8`

If setting one of these locale categories succeeds, then the `LC_CTYPE` environment variable will also be set accordingly in the current process environment before the Python runtime is initialized. This ensures that in addition to being seen by both the interpreter itself and other locale-aware components running in the same process (such as the GNU `readline` library), the updated setting is also seen in subprocesses (regardless of whether or not those processes are running a Python interpreter), as well as in operations that query the environment rather than the current C locale (such as Python's own `locale.getdefaultlocale()`).

Configuring one of these locales (either explicitly or via the above implicit locale coercion) automatically enables the `surrogateescape` error handler for `sys.stdin` and `sys.stdout` (`sys.stderr` continues to use `backslashreplace` as it does in any other locale). This stream handling behavior can be overridden using `PYTHONIOENCODING` as usual.

For debugging purposes, setting `PYTHONCOERCECLOCALE=warn` will cause Python to emit warning messages on `stderr` if either the locale coercion activates, or else if a locale that *would* have triggered coercion is still active when the Python runtime is initialized.

Also note that even when locale coercion is disabled, or when it fails to find a suitable target locale, `PYTHONUTF8` will still activate by default in legacy ASCII-based locales. Both features must be disabled in order to force the interpreter to use ASCII instead of UTF-8 for system interfaces.

Availability: \*nix.

Νέο στην έκδοση 3.7: See [PEP 538](#) for more details.

#### **PYTHONDEVMODE**

If this environment variable is set to a non-empty string, enable Python Development Mode, introducing additional runtime checks that are too expensive to be enabled by default.

Νέο στην έκδοση 3.7.

#### **PYTHONUTF8**

If set to 1, enable the Python UTF-8 Mode.

If set to 0, disable the Python UTF-8 Mode.

Setting any other non-empty string causes an error during interpreter initialisation.

Νέο στην έκδοση 3.7.

#### **PYTHONWARNDEFAULTENCODING**

If this environment variable is set to a non-empty string, issue a `EncodingWarning` when the locale-specific default encoding is used.

See `io-encoding-warning` for details.

Νέο στην έκδοση 3.10.

## 1.2.1 Debug-mode variables

#### **PYTHONTHREADDEBUG**

If set, Python will print threading debug info into stdout.

Need a *debug build of Python*.

Deprecated since version 3.10, will be removed in version 3.12.

#### **PYTHONDUMPREFS**

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

Need Python configured with the `--with-trace-refs` build option.



---

### Χρήση της Python σε πλατφόρμες Unix

---

## 2.1 Λήψη και εγκατάσταση της πιο πρόσφατης έκδοσης Python

### 2.1.1 Σε 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:

**Δείτε επίσης:**

<https://www.debian.org/doc/manuals/maint-guide/first.en.html> για τους χρήστες του Debian

<https://en.opensuse.org/Portal:Packaging> για τους χρήστες του OpenSuse

[https://docs-old.fedoraproject.org/en-US/Fedora\\_Draft\\_Documentation/0.1/html/RPM\\_Guide/ch-creating-rpms.html](https://docs-old.fedoraproject.org/en-US/Fedora_Draft_Documentation/0.1/html/RPM_Guide/ch-creating-rpms.html)  
για τους χρήστες του Fedora

<http://www.slackbook.org/html/package-management-making-packages.html> για τους χρήστες του Slackware

### 2.1.2 Σε FreeBSD και OpenBSD

- Χρήστες του FreeBSD, για την προσθήκη του πακέτου χρησιμοποιήστε:

```
pkg install python3
```

- Χρήστες του OpenBSD, για την προσθήκη του πακέτου χρησιμοποιήστε:

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

Για παράδειγμα, χρήστες του i386 κάνουν λήψη της έκδοσης 2.5.1 χρησιμοποιώντας:

```
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 Μεταγλώττιση της Python

Αν θέλετε να κάνετε compile το CPython μόνοι σας, το πρώτο πράγμα που πρέπει να κάνετε είναι να πάρετε τον πηγαίο κώδικα <<https://www.python.org/downloads/source/>>\_. Μπορείτε να κατεβάσετε είτε τον πηγαίο κώδικα της τελευταίας έκδοσης είτε απλά να πάρετε έναν καινούργιο `clone`. (Αν θέλετε να συνεισφέρετε διορθώσεις, θα χρειαστείτε έναν κλώνο).

Η διαδικασία της μεταγλώττισης απαρτίζεται από τις συνήθεις εντολές:

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

Το *Configuration options* και οι όροι για συγκεκριμένες πλατφόρμες Unix τεκμηριώνονται εκτενώς στο αρχείο `README.rst` στην βάση του πηγαίου δέντρου της Python.

**Προειδοποίηση:** Το `make install` μπορεί να αντικαταστήσει ή να μεταμφιέσει το `python3` σε δυαδικό. Επομένως προτείνεται το `make altinstall` σε σχέση με το `make install` που μπορεί να εγκαταστήσει μόνο το `:file`{exec_prefix}/bin/python{version}``.

## 2.3 Διαδρομές και αρχεία που σχετίζονται με την Python

Αυτά ενδέχεται να διαφέρουν ανάλογα με τις τοπικές συμβάσεις εγκατάστασης• τα `prefix` και `exec_prefix` εξαρτώνται από την εγκατάσταση και θα πρέπει να ερμηνεύονται όπως για το λογισμικό GNU• μπορεί να είναι τα ίδια.

Για παράδειγμα, στα περισσότερα Linux συστήματα, η προεπιλογή είναι και για τα δύο `/usr`.

File/directory	Που σημαίνει
<code>exec_prefix/bin/python3</code>	Προτεινόμενη θέση του διεργαστή.
<code>prefix/lib/pythonversion,</code> <code>exec_prefix/lib/pythonversion</code>	Προτεινόμενες θέσεις για τους καταλόγους που περιέχουν τα βασικά modules.
<code>prefix/include/pythonversion,</code> <code>exec_prefix/include/pythonversion</code>	Προτεινόμενες θέσεις των καταλόγων που περιέχουν τα αρχεία κεφαλίδων (include files) που απαιτούνται για την ανάπτυξη επεκτάσεων της Python και την ενσωμάτωση του διεργαστή.

## 2.4 Διάφορα

Για να εκτελείτε εύκολα τα Python scripts σε συστήματα Unix, πρέπει να τα κάνετε εκτελέσιμα, για παράδειγμα με

```
$ chmod +x script
```

και να προσθέσετε μια κατάλληλη γραμμή Shebang στην αρχή του script. Μια καλή επιλογή είναι συνήθως

```
#!/usr/bin/env python3
```

που κάνει αναζήτηση για τον διερμηνέα της Python σε ολόκληρο το PATH. Ωστόσο, ορισμένα Unixes μπορεί να μην έχουν την εντολή **env**, οπότε μπορεί να χρειαστεί να κωδικοποιήσετε το `/usr/bin/python3` ως διαδρομή του διερμηνέα.

Για να χρησιμοποιήσετε εντολές shell στα Python script σας, δείτε την ενότητα `subprocess`.

## 2.5 Custom OpenSSL

1. Για να χρησιμοποιήσετε τις ρυθμίσεις του OpenSSL και το αποθετήριο εμπιστοσύνης συστήματος, εντοπίστε τον κατάλογο με το αρχείο `openssl.cnf` ή τον συμβολικό σύνδεσμο στο `/etc`. Στις περισσότερες διανομές το αρχείο βρίσκεται είτε στο `/etc/ssl` είτε στο `/etc/pki/tls`. Ο κατάλογος θα πρέπει επίσης να περιέχει ένα αρχείο `cert.pem` και/ή έναν κατάλογο `certs`.

```
$ find /etc/ -name openssl.cnf -printf "%h\n"
/etc/ssl
```

2. Λήψη, δημιουργία και εγκατάσταση του OpenSSL. Βεβαιωθείτε ότι χρησιμοποιείτε το `install_sw` και όχι το `install`. Ο στόχος `install_sw` δεν παρακάμπτει το αρχείο `openssl.cnf`.

```
$ curl -O https://www.openssl.org/source/openssl-VERSION.tar.gz
$ tar xzf openssl-VERSION
$ pushd openssl-VERSION
$ ./config \
  --prefix=/usr/local/custom-openssl \
  --libdir=lib \
  --openssldir=/etc/ssl
$ make -j1 depend
$ make -j8
$ make install_sw
$ popd
```

3. Μεταγλώττιση της Python με προσαρμοσμένο OpenSSL (δείτε τις επιλογές `configure` `---with-openssl` και `---with-openssl-rpath`)

```
$ pushd python-3.x.x
$ ./configure -C \
  --with-openssl=/usr/local/custom-openssl \
  --with-openssl-rpath=auto \
  --prefix=/usr/local/python-3.x.x
$ make -j8
$ make altinstall
```

**Σημείωση:** Οι διορθωτικές εκδόσεις του OpenSSL έχουν μια δυαδική διεπαφή εφαρμογής (ABI) συμβατή προς τα πίσω. Δεν χρειάζεται να κάνετε `compile` εκ νέου την Python για να ενημερώσετε το OpenSSL. Αρκεί να αντικαταστήσετε την προσαρμοσμένη εγκατάσταση του OpenSSL με μια νεότερη έκδοση.





## Configure Python

### 3.1 Configure Options

List all `./configure` script options using:

```
./configure --help
```

See also the `Misc/SpecialBuilds.txt` in the Python source distribution.

#### 3.1.1 General Options

##### **--enable-loadable-sqlite-extensions**

Support loadable extensions in the `_sqlite` extension module (default is no).

See the `sqlite3.Connection.enable_load_extension()` method of the `sqlite3` module.

Νέο στην έκδοση 3.6.

##### **--disable-ipv6**

Disable IPv6 support (enabled by default if supported), see the `socket` module.

##### **--enable-big-digits=[15|30]**

Define the size in bits of Python `int` digits: 15 or 30 bits.

By default, the number of bits is selected depending on `sizeof(void*)`: 30 bits if `void*` size is 64-bit or larger, 15 bits otherwise.

Define the `PYLONG_BITS_IN_DIGIT` to 15 or 30.

See `sys.int_info.bits_per_digit`.

##### **--with-cxx-main**

##### **--with-cxx-main=COMPILER**

Compile the Python `main()` function and link Python executable with C++ compiler: `$CXX`, or `COMPILER` if specified.

##### **--with-suffix=SUFFIX**

Set the Python executable suffix to `SUFFIX`.

The default suffix is `.exe` on Windows and macOS (`python.exe` executable), and an empty string on other platforms (`python` executable).

**--with-tzpath=<list of absolute paths separated by pathsep>**

Select the default time zone search path for `zoneinfo.TZPATH`. See the Compile-time configuration of the `zoneinfo` module.

Default: `/usr/share/zoneinfo:/usr/lib/zoneinfo:/usr/share/lib/zoneinfo:/etc/zoneinfo`.

See `os.pathsep` path separator.

Νέο στην έκδοση 3.9.

**--without-decimal-contextvar**

Build the `_decimal` extension module using a thread-local context rather than a coroutine-local context (default), see the `decimal` module.

See `decimal.HAVE_CONTEXTVAR` and the `contextvars` module.

Νέο στην έκδοση 3.9.

**--with-dbmliborder=db1:db2:...**

Override order to check db backends for the `dbm` module

A valid value is a colon (:) separated string with the backend names:

- `ndbm`;
- `gdbm`;
- `bdb`.

**--without-c-locale-coercion**

Disable C locale coercion to a UTF-8 based locale (enabled by default).

Don't define the `PY_COERCE_C_LOCALE` macro.

See [PYTHONCOERCECLOCALE](#) and the [PEP 538](#).

**--with-platlibdir=DIRNAME**

Python library directory name (default is `lib`).

Fedora and SuSE use `lib64` on 64-bit platforms.

See `sys.platlibdir`.

Νέο στην έκδοση 3.9.

**--with-wheel-pkg-dir=PATH**

Directory of wheel packages used by the `ensurepip` module (none by default).

Some Linux distribution packaging policies recommend against bundling dependencies. For example, Fedora installs wheel packages in the `/usr/share/python-wheels/` directory and don't install the `ensurepip._bundled` package.

Νέο στην έκδοση 3.10.

### 3.1.2 Install Options

#### **--prefix=PREFIX**

Install architecture-independent files in PREFIX. On Unix, it defaults to `/usr/local`.

This value can be retrieved at runtime using `sys.prefix`.

As an example, one can use `--prefix="$HOME/.local/"` to install a Python in its home directory.

#### **--exec-prefix=EPREFIX**

Install architecture-dependent files in EPREFIX, defaults to `--prefix`.

This value can be retrieved at runtime using `sys.exec_prefix`.

#### **--disable-test-modules**

Don't build nor install test modules, like the `test` package or the `_testcapi` extension module (built and installed by default).

Νέο στην έκδοση 3.10.

#### **--with-ensurepip=[upgrade|install|no]**

Select the `ensurepip` command run on Python installation:

- `upgrade` (default): `run python -m ensurepip --altinstall --upgrade command`.
- `install`: `run python -m ensurepip --altinstall command`;
- `no`: don't run `ensurepip`;

Νέο στην έκδοση 3.6.

### 3.1.3 Performance options

Configuring Python using `--enable-optimizations --with-lto` (PGO + LTO) is recommended for best performance.

#### **--enable-optimizations**

Enable Profile Guided Optimization (PGO) using `PROFILE_TASK` (disabled by default).

The C compiler Clang requires `llvm-profdata` program for PGO. On macOS, GCC also requires it: GCC is just an alias to Clang on macOS.

Disable also semantic interposition in libpython if `--enable-shared` and GCC is used: add `-fno-semantic-interposition` to the compiler and linker flags.

Νέο στην έκδοση 3.6.

Αλλάξε στην έκδοση 3.10: Use `-fno-semantic-interposition` on GCC.

#### **PROFILE\_TASK**

Environment variable used in the Makefile: Python command line arguments for the PGO generation task.

Default: `-m test --pgo --timeout=$(TESTTIMEOUT)`.

Νέο στην έκδοση 3.8.

#### **--with-lto**

Enable Link Time Optimization (LTO) in any build (disabled by default).

The C compiler Clang requires `llvm-ar` for LTO (`ar` on macOS), as well as an LTO-aware linker (`ld.gold` or `lld`).

Νέο στην έκδοση 3.6.

#### **--with-computed-gotos**

Enable computed gotos in evaluation loop (enabled by default on supported compilers).

### **--without-pymalloc**

Disable the specialized Python memory allocator pymalloc (enabled by default).

See also `PYTHONMALLOC` environment variable.

### **--without-doc-strings**

Disable static documentation strings to reduce the memory footprint (enabled by default). Documentation strings defined in Python are not affected.

Don't define the `WITH_DOC_STRINGS` macro.

See the `PyDoc_STRVAR()` macro.

### **--enable-profiling**

Enable C-level code profiling with `gprof` (disabled by default).

## 3.1.4 Python Debug Build

A debug build is Python built with the `--with-pydebug` configure option.

Effects of a debug build:

- Display all warnings by default: the list of default warning filters is empty in the `warnings` module.
- Add `d` to `sys.abiflags`.
- Add `sys.gettotalrefcount()` function.
- Add `-X showrefcount` command line option.
- Add `PYTHONTHREADDEBUG` environment variable.
- Add support for the `__ltrace__` variable: enable low-level tracing in the bytecode evaluation loop if the variable is defined.
- Install debug hooks on memory allocators to detect buffer overflow and other memory errors.
- Define `Py_DEBUG` and `Py_REF_DEBUG` macros.
- Add runtime checks: code surrounded by `#ifdef Py_DEBUG` and `#endif`. Enable `assert(..)` and `_PyObject_ASSERT(...)` assertions: don't set the `NDEBUG` macro (see also the `--with-assertions` configure option). Main runtime checks:
  - Add sanity checks on the function arguments.
  - Unicode and int objects are created with their memory filled with a pattern to detect usage of uninitialized objects.
  - Ensure that functions which can clear or replace the current exception are not called with an exception raised.
  - The garbage collector (`gc.collect()` function) runs some basic checks on objects consistency.
  - The `Py_SAFE_DOWNCAST()` macro checks for integer underflow and overflow when downcasting from wide types to narrow types.

See also the Python Development Mode and the `--with-trace-refs` configure option.

Αλλάξε στην έκδοση 3.8: Release builds and debug builds are now ABI compatible: defining the `Py_DEBUG` macro no longer implies the `Py_TRACE_REFS` macro (see the `--with-trace-refs` option), which introduces the only ABI incompatibility.

### 3.1.5 Debug options

#### **--with-pydebug**

*Build Python in debug mode:* define the `Py_DEBUG` macro (disabled by default).

#### **--with-trace-refs**

Enable tracing references for debugging purpose (disabled by default).

Effects:

- Define the `Py_TRACE_REFS` macro.
- Add `sys.getobjects()` function.
- Add `PYTHONDUMPREFS` environment variable.

This build is not ABI compatible with release build (default build) or debug build (`Py_DEBUG` and `Py_REF_DEBUG` macros).

Νέο στην έκδοση 3.8.

#### **--with-assertions**

Build with C assertions enabled (default is no): `assert(...);` and `_PyObject_ASSERT(...);`.

If set, the `NDEBUG` macro is not defined in the `OPT` compiler variable.

See also the *--with-pydebug* option (*debug build*) which also enables assertions.

Νέο στην έκδοση 3.6.

#### **--with-valgrind**

Enable Valgrind support (default is no).

#### **--with-dtrace**

Enable DTrace support (default is no).

See Instrumenting CPython with DTrace and SystemTap.

Νέο στην έκδοση 3.6.

#### **--with-address-sanitizer**

Enable AddressSanitizer memory error detector, `asan` (default is no).

Νέο στην έκδοση 3.6.

#### **--with-memory-sanitizer**

Enable MemorySanitizer allocation error detector, `msan` (default is no).

Νέο στην έκδοση 3.6.

#### **--with-undefined-behavior-sanitizer**

Enable UndefinedBehaviorSanitizer undefined behaviour detector, `ubsan` (default is no).

Νέο στην έκδοση 3.6.

### 3.1.6 Linker options

#### **--enable-shared**

Enable building a shared Python library: `libpython` (default is no).

#### **--without-static-libpython**

Do not build `libpythonMAJOR.MINOR.a` and do not install `python.o` (built and enabled by default).

Νέο στην έκδοση 3.10.

### 3.1.7 Libraries options

**--with-libs='lib1 ...'**

Link against additional libraries (default is no).

**--with-system-expat**

Build the `pyexpat` module using an installed `expat` library (default is no).

**--with-system-ffi**

Build the `_ctypes` extension module using an installed `ffi` library, see the `ctypes` module (default is system-dependent).

**--with-system-libmpdec**

Build the `_decimal` extension module using an installed `mpdec` library, see the `decimal` module (default is no).

Νέο στην έκδοση 3.3.

**--with-readline=editline**

Use `editline` library for backend of the `readline` module.

Define the `WITH_EDITLINE` macro.

Νέο στην έκδοση 3.10.

**--without-readline**

Don't build the `readline` module (built by default).

Don't define the `HAVE_LIBREADLINE` macro.

Νέο στην έκδοση 3.10.

**--with-tcltk-includes='-I...'**

Override search for Tcl and Tk include files.

**--with-tcltk-libs='-L...'**

Override search for Tcl and Tk libraries.

**--with-libm=STRING**

Override `libm` math library to *STRING* (default is system-dependent).

**--with-libc=STRING**

Override `libc` C library to *STRING* (default is system-dependent).

**--with-openssl=DIR**

Root of the OpenSSL directory.

Νέο στην έκδοση 3.7.

**--with-openssl-rpath=[no|auto|DIR]**

Set runtime library directory (rpath) for OpenSSL libraries:

- `no` (default): don't set rpath;
- `auto`: auto-detect rpath from `--with-openssl` and `pkg-config`;
- `DIR`: set an explicit rpath.

Νέο στην έκδοση 3.10.

### 3.1.8 Security Options

**--with-hash-algorithm**=[fnv|siphhash24]

Select hash algorithm for use in Python/pyhash.c:

- siphhash24 (default).
- fnv;

Νέο στην έκδοση 3.4.

**--with-builtin-hashlib-hashes**=md5,sha1,sha256,sha512,sha3,blake2

Built-in hash modules:

- md5;
- sha1;
- sha256;
- sha512;
- sha3 (with shake);
- blake2.

Νέο στην έκδοση 3.9.

**--with-ssl-default-suites**=[python|openssl|STRING]

Override the OpenSSL default cipher suites string:

- python (default): use Python's preferred selection;
- openssl: leave OpenSSL's defaults untouched;
- *STRING*: use a custom string

See the `ssl` module.

Νέο στην έκδοση 3.7.

Αλλάξε στην έκδοση 3.10: The settings `python` and *STRING* also set TLS 1.2 as minimum protocol version.

### 3.1.9 macOS Options

See `Mac/README.rst`.

**--enable-universalsdk**

**--enable-universalsdk**=SDKDIR

Create a universal binary build. *SDKDIR* specifies which macOS SDK should be used to perform the build (default is no).

**--enable-framework**

**--enable-framework**=INSTALLDIR

Create a Python.framework rather than a traditional Unix install. Optional *INSTALLDIR* specifies the installation path (default is no).

**--with-universal-archs**=ARCH

Specify the kind of universal binary that should be created. This option is only valid when `--enable-universalsdk` is set.

Options:

- universal2;
- 32-bit;
- 64-bit;

- 3-way;
- intel;
- intel-32;
- intel-64;
- all.

**--with-framework-name=FRAMEWORK**

Specify the name for the python framework on macOS only valid when `--enable-framework` is set (default: Python).

## 3.2 Python Build System

### 3.2.1 Main files of the build system

- `configure.ac` => `configure`;
- `Makefile.pre.in` => `Makefile` (created by `configure`);
- `pyconfig.h` (created by `configure`);
- `Modules/Setup`: C extensions built by the Makefile using `Module/makesetup` shell script;
- `setup.py`: C extensions built using the `distutils` module.

### 3.2.2 Main build steps

- C files (`.c`) are built as object files (`.o`).
- A static `libpython` library (`.a`) is created from objects files.
- `python.o` and the static `libpython` library are linked into the final `python` program.
- C extensions are built by the Makefile (see `Modules/Setup`) and `python setup.py build`.

### 3.2.3 Main Makefile targets

- `make`: Build Python with the standard library.
- `make platform::` build the `python` program, but don't build the standard library extension modules.
- `make profile-opt`: build Python using Profile Guided Optimization (PGO). You can use the `configure --enable-optimizations` option to make this the default target of the `make` command (`make all` or just `make`).
- `make buildbottest`: Build Python and run the Python test suite, the same way than `buildbots` test Python. Set `TESTTIMEOUT` variable (in seconds) to change the test timeout (1200 by default: 20 minutes).
- `make install`: Build and install Python.
- `make regen-all`: Regenerate (almost) all generated files; `make regen-stdlib-module-names` and `autoconf` must be run separately for the remaining generated files.
- `make clean`: Remove built files.
- `make distclean`: Same than `make clean`, but remove also files created by the `configure` script.



### 3.2.4 C extensions

Some C extensions are built as built-in modules, like the `sys` module. They are built with the `Py_BUILD_CORE_BUILTIN` macro defined. Built-in modules have no `__file__` attribute:

```
>>> import sys
>>> sys
<module 'sys' (built-in)>
>>> sys.__file__
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
AttributeError: module 'sys' has no attribute '__file__'
```

Other C extensions are built as dynamic libraries, like the `_asyncio` module. They are built with the `Py_BUILD_CORE_MODULE` macro defined. Example on Linux x86-64:

```
>>> import _asyncio
>>> _asyncio
<module '_asyncio' from '/usr/lib64/python3.9/lib-dynload/_asyncio.cpython-39-x86_
↳ 64-linux-gnu.so'>
>>> _asyncio.__file__
'/usr/lib64/python3.9/lib-dynload/_asyncio.cpython-39-x86_64-linux-gnu.so'
```

`Modules/Setup` is used to generate Makefile targets to build C extensions. At the beginning of the files, C extensions are built as built-in modules. Extensions defined after the `*shared*` marker are built as dynamic libraries.

The `setup.py` script only builds C extensions as shared libraries using the `distutils` module.

The `PyAPI_FUNC()`, `PyAPI_API()` and `PyMODINIT_FUNC()` macros of `Include/pyport.h` are defined differently depending if the `Py_BUILD_CORE_MODULE` macro is defined:

- Use `Py_EXPORTED_SYMBOL` if the `Py_BUILD_CORE_MODULE` is defined
- Use `Py_IMPORTED_SYMBOL` otherwise.

If the `Py_BUILD_CORE_BUILTIN` macro is used by mistake on a C extension built as a shared library, its `PyInit_xxx()` function is not exported, causing an `ImportError` on import.

## 3.3 Compiler and linker flags

Options set by the `./configure` script and environment variables and used by Makefile.

### 3.3.1 Preprocessor flags

#### **CONFIGURE\_CPPFLAGS**

Value of `CPPFLAGS` variable passed to the `./configure` script.

Νέο στην έκδοση 3.6.

#### **CPPFLAGS**

(Objective) C/C++ preprocessor flags, e.g. `-I<include dir>` if you have headers in a nonstandard directory `<include dir>`.

Both `CPPFLAGS` and `LDFLAGS` need to contain the shell's value for `setup.py` to be able to build extension modules using the directories specified in the environment variables.

#### **BASECPPFLAGS**

Νέο στην έκδοση 3.4.

#### **PY\_CPPFLAGS**

Extra preprocessor flags added for building the interpreter object files.

Default: `$(BASECPPFLAGS) -I. -I$(srcdir)/Include $(CONFIGURE_CPPFLAGS) $(CPPFLAGS)`.

Νέο στην έκδοση 3.2.

### **3.3.2 Compiler flags**

#### **CC**

C compiler command.

Example: `gcc -pthread`.

#### **MAINCC**

C compiler command used to build the `main()` function of programs like `python`.

Variable set by the `--with-cxx-main` option of the configure script.

Default: `$(CC)`.

#### **CXX**

C++ compiler command.

Used if the `--with-cxx-main` option is used.

Example: `g++ -pthread`.

#### **CFLAGS**

C compiler flags.

#### **CFLAGS\_NODIST**

`CFLAGS_NODIST` is used for building the interpreter and stdlib C extensions. Use it when a compiler flag should *not* be part of the distutils `CFLAGS` once Python is installed ([bpo-21121](#)).

In particular, `CFLAGS` should not contain:

- the compiler flag `-I` (for setting the search path for include files). The `-I` flags are processed from left to right, and any flags in `CFLAGS` would take precedence over user- and package-supplied `-I` flags.
- hardening flags such as `-Werror` because distributions cannot control whether packages installed by users conform to such heightened standards.

Νέο στην έκδοση 3.5.

#### **EXTRA\_CFLAGS**

Extra C compiler flags.

#### **CONFIGURE\_CFLAGS**

Value of `CFLAGS` variable passed to the `./configure` script.

Νέο στην έκδοση 3.2.

#### **CONFIGURE\_CFLAGS\_NODIST**

Value of `CFLAGS_NODIST` variable passed to the `./configure` script.

Νέο στην έκδοση 3.5.

#### **BASECFLAGS**

Base compiler flags.

#### **OPT**

Optimization flags.

#### **CFLAGS\_ALIASING**

Strict or non-strict aliasing flags used to compile `Python/dtoa.c`.

Νέο στην έκδοση 3.7.

**CCSHARED**

Compiler flags used to build a shared library.

For example, `-fPIC` is used on Linux and on BSD.

**CFLAGSFORSHARED**

Extra C flags added for building the interpreter object files.

Default: `$(CCSHARED)` when `--enable-shared` is used, or an empty string otherwise.

**PY\_CFLAGS**

Default: `$(BASECFLAGS) $(OPT) $(CONFIGURE_CFLAGS) $(CFLAGS) $(EXTRA_CFLAGS)`.

**PY\_CFLAGS\_NODIST**

Default: `$(CONFIGURE_CFLAGS_NODIST) $(CFLAGS_NODIST) -I$(srcdir)/Include/internal`.

Νέο στην έκδοση 3.5.

**PY\_STDMODULE\_CFLAGS**

C flags used for building the interpreter object files.

Default: `$(PY_CFLAGS) $(PY_CFLAGS_NODIST) $(PY_CPPFLAGS) $(CFLAGSFORSHARED)`.

Νέο στην έκδοση 3.7.

**PY\_CORE\_CFLAGS**

Default: `$(PY_STDMODULE_CFLAGS) -DPy_BUILD_CORE`.

Νέο στην έκδοση 3.2.

**PY\_BUILTIN\_MODULE\_CFLAGS**

Compiler flags to build a standard library extension module as a built-in module, like the `posix` module.

Default: `$(PY_STDMODULE_CFLAGS) -DPy_BUILD_CORE_BUILTIN`.

Νέο στην έκδοση 3.8.

**PURIFY**

Purify command. Purify is a memory debugger program.

Default: empty string (not used).

### 3.3.3 Linker flags

**LINKCC**

Linker command used to build programs like `python` and `_testembed`.

Default: `$(PURIFY) $(MAINCC)`.

**CONFIGURE\_LDFLAGS**

Value of `LD_FLAGS` variable passed to the `./configure` script.

Avoid assigning `CFLAGS`, `LD_FLAGS`, etc. so users can use them on the command line to append to these values without stomping the pre-set values.

Νέο στην έκδοση 3.2.

**LD\_FLAGS\_NODIST**

`LD_FLAGS_NODIST` is used in the same manner as `CFLAGS_NODIST`. Use it when a linker flag should *not* be part of the distutils `LD_FLAGS` once Python is installed ([bpo-35257](#)).

In particular, `LD_FLAGS` should not contain:

- the compiler flag `-L` (for setting the search path for libraries). The `-L` flags are processed from left to right, and any flags in `LD_FLAGS` would take precedence over user- and package-supplied `-L` flags.

### **CONFIGURE\_LDFLAGS\_NODIST**

Value of `LDFLAGS_NODIST` variable passed to the `./configure` script.

Νέο στην έκδοση 3.8.

### **LDFLAGS**

Linker flags, e.g. `-L<lib dir>` if you have libraries in a nonstandard directory `<lib dir>`.

Both `CPPFLAGS` and `LDFLAGS` need to contain the shell's value for `setup.py` to be able to build extension modules using the directories specified in the environment variables.

### **LIBS**

Linker flags to pass libraries to the linker when linking the Python executable.

Example: `-lrt`.

### **LDSHARED**

Command to build a shared library.

Default: `@LDSHARED@ $(PY_LDFLAGS)`.

### **BLDSHARED**

Command to build `libpython` shared library.

Default: `@BLDSHARED@ $(PY_CORE_LDFLAGS)`.

### **PY\_LDFLAGS**

Default: `$(CONFIGURE_LDFLAGS) $(LDFLAGS)`.

### **PY\_LDFLAGS\_NODIST**

Default: `$(CONFIGURE_LDFLAGS_NODIST) $(LDFLAGS_NODIST)`.

Νέο στην έκδοση 3.8.

### **PY\_CORE\_LDFLAGS**

Linker flags used for building the interpreter object files.

Νέο στην έκδοση 3.8.

---

## 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.

Unlike most Unix systems and services, Windows does not include a system supported installation of Python. To make Python available, the CPython team has compiled Windows installers (MSI packages) with every [release](#) for many years. These installers are primarily intended to add a per-user installation of Python, with the core interpreter and library being used by a single user. The installer is also able to install for all users of a single machine, and a separate ZIP file is available for application-local distributions.

As specified in [PEP 11](#), a Python release only supports a Windows platform while Microsoft considers the platform under extended support. This means that Python 3.10 supports Windows 8.1 and newer. If you require Windows 7 support, please install Python 3.8.

There are a number of different installers available for Windows, each with certain benefits and downsides.

*The full installer* contains all components and is the best option for developers using Python for any kind of project.

*The Microsoft Store package* is a simple installation of Python that is suitable for running scripts and packages, and using IDLE or other development environments. It requires Windows 10 and above, but can be safely installed without corrupting other programs. It also provides many convenient commands for launching Python and its tools.

*The [nuget.org](#) packages* are lightweight installations intended for continuous integration systems. It can be used to build Python packages or run scripts, but is not updateable and has no user interface tools.

*The embeddable package* is a minimal package of Python suitable for embedding into a larger application.

### 4.1 The full installer

#### 4.1.1 Installation steps

Four Python 3.10 installers are available for download - two each for the 32-bit and 64-bit versions of the interpreter. The *web installer* is a small initial download, and it will automatically download the required components as necessary. The *offline installer* includes the components necessary for a default installation and only requires an internet connection for optional features. See [Installing Without Downloading](#) for other ways to avoid downloading during installation.

After starting the installer, one of two options may be selected:



If you select «Install Now»:

- You will *not* need to be an administrator (unless a system update for the C Runtime Library is required or you install the *Python Launcher for Windows* for all users)
- Python will be installed into your user directory
- The *Python Launcher for Windows* will be installed according to the option at the bottom of the first page
- The standard library, test suite, launcher and pip will be installed
- If selected, the install directory will be added to your `PATH`
- Shortcuts will only be visible for the current user

Selecting «Customize installation» will allow you to select the features to install, the installation location and other options or post-install actions. To install debugging symbols or binaries, you will need to use this option.

To perform an all-users installation, you should select «Customize installation». In this case:

- You may be required to provide administrative credentials or approval
- Python will be installed into the Program Files directory
- The *Python Launcher for Windows* will be installed into the Windows directory
- Optional features may be selected during installation
- The standard library can be pre-compiled to bytecode
- If selected, the install directory will be added to the system `PATH`
- Shortcuts are available for all users

### 4.1.2 Removing the MAX\_PATH Limitation

Windows historically has limited path lengths to 260 characters. This meant that paths longer than this would not resolve and errors would result.

In the latest versions of Windows, this limitation can be expanded to approximately 32,000 characters. Your administrator will need to activate the «Enable Win32 long paths» group policy, or set `LongPathsEnabled` to 1 in the registry key `HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\FileSystem`.

This allows the `open()` function, the `os` module and most other path functionality to accept and return paths longer than 260 characters.

After changing the above option, no further configuration is required.

Άλλαξε στην έκδοση 3.6: Support for long paths was enabled in Python.

### 4.1.3 Installing Without UI

All of the options available in the installer UI can also be specified from the command line, allowing scripted installers to replicate an installation on many machines without user interaction. These options may also be set without suppressing the UI in order to change some of the defaults.

To completely hide the installer UI and install Python silently, pass the `/quiet` option. To skip past the user interaction but still display progress and errors, pass the `/passive` option. The `/uninstall` option may be passed to immediately begin removing Python - no confirmation prompt will be displayed.

All other options are passed as `name=value`, where the value is usually 0 to disable a feature, 1 to enable a feature, or a path. The full list of available options is shown below.

Name	Description	Default
InstallAllUsers	Perform a system-wide installation.	0
TargetDir	The installation directory	Selected based on InstallAllUsers
DefaultAllUsersTargetDir	The default installation directory for all-user installs	%ProgramFiles%\Python X.Y or %ProgramFiles(x86)%\Python X.Y
DefaultJustForMeTargetDir	The default install directory for just-for-me installs	%LocalAppData%\Programs\Python\PythonXY or %LocalAppData%\Programs\Python\PythonXY-32 or %LocalAppData%\Programs\Python\PythonXY-64
DefaultCustomTargetDir	The default custom install directory displayed in the UI	(empty)
AssociateFiles	Create file associations if the launcher is also installed.	1
CompileAll	Compile all .py files to .pyc.	0
PrependPath	Add install and Scripts directories to PATH and .PY to PATHEXT	0
Shortcuts	Create shortcuts for the interpreter, documentation and IDLE if installed.	1
Include_doc	Install Python manual	1
Include_debug	Install debug binaries	0
Include_dev	Install developer headers and libraries. Omitting this may lead to an unusable installation.	1
Include_exe	Install python.exe and related files. Omitting this may lead to an unusable installation.	1
Include_launcher	Install <i>Python Launcher for Windows</i> .	1
InstallLauncherForAllUsers	Install the launcher for all users. Also requires Include_launcher to be set to 1	1
Include_lib	Install standard library and extension modules. Omitting this may lead to an unusable installation.	1
Include_pip	Install bundled pip and setuptools	1
Include_symbols	Install debugging symbols (*.pdb)	0
Include_tcltk	Install Tcl/Tk support and IDLE	1
Include_test	Install standard library test suite	1
Include_tools	Install utility scripts	1
LauncherOnly	Only installs the launcher. This will override most other options.	0
SimpleInstall	Disable most install UI	0
SimpleInstallDescription	Description message to display when the simplified install UI is used.	(empty)

For example, to silently install a default, system-wide Python installation, you could use the following command (from an elevated command prompt):



```
python-3.9.0.exe /quiet InstallAllUsers=1 PrependPath=1 Include_test=0
```

To allow users to easily install a personal copy of Python without the test suite, you could provide a shortcut with the following command. This will display a simplified initial page and disallow customization:

```
python-3.9.0.exe InstallAllUsers=0 Include_launcher=0 Include_test=0
SimpleInstall=1 SimpleInstallDescription="Just for me, no test suite."
```

(Note that omitting the launcher also omits file associations, and is only recommended for per-user installs when there is also a system-wide installation that included the launcher.)

The options listed above can also be provided in a file named `unattend.xml` alongside the executable. This file specifies a list of options and values. When a value is provided as an attribute, it will be converted to a number if possible. Values provided as element text are always left as strings. This example file sets the same options as the previous example:

```
<Options>
  <Option Name="InstallAllUsers" Value="no" />
  <Option Name="Include_launcher" Value="0" />
  <Option Name="Include_test" Value="no" />
  <Option Name="SimpleInstall" Value="yes" />
  <Option Name="SimpleInstallDescription">Just for me, no test suite</Option>
</Options>
```

#### 4.1.4 Installing Without Downloading

As some features of Python are not included in the initial installer download, selecting those features may require an internet connection. To avoid this need, all possible components may be downloaded on-demand to create a complete *layout* that will no longer require an internet connection regardless of the selected features. Note that this download may be bigger than required, but where a large number of installations are going to be performed it is very useful to have a locally cached copy.

Execute the following command from Command Prompt to download all possible required files. Remember to substitute `python-3.9.0.exe` for the actual name of your installer, and to create layouts in their own directories to avoid collisions between files with the same name.

```
python-3.9.0.exe /layout [optional target directory]
```

You may also specify the `/quiet` option to hide the progress display.

#### 4.1.5 Modifying an install

Once Python has been installed, you can add or remove features through the Programs and Features tool that is part of Windows. Select the Python entry and choose «Uninstall/Change» to open the installer in maintenance mode.

«Modify» allows you to add or remove features by modifying the checkboxes - unchanged checkboxes will not install or remove anything. Some options cannot be changed in this mode, such as the install directory; to modify these, you will need to remove and then reinstall Python completely.

«Repair» will verify all the files that should be installed using the current settings and replace any that have been removed or modified.

«Uninstall» will remove Python entirely, with the exception of the *Python Launcher for Windows*, which has its own entry in Programs and Features.

## 4.2 The Microsoft Store package

Νέο στην έκδοση 3.7.2.

The Microsoft Store package is an easily installable Python interpreter that is intended mainly for interactive use, for example, by students.

To install the package, ensure you have the latest Windows 10 updates and search the Microsoft Store app for «Python 3.10». Ensure that the app you select is published by the Python Software Foundation, and install it.

**Προειδοποίηση:** Python will always be available for free on the Microsoft Store. If you are asked to pay for it, you have not selected the correct package.

After installation, Python may be launched by finding it in Start. Alternatively, it will be available from any Command Prompt or PowerShell session by typing `python`. Further, `pip` and `IDLE` may be used by typing `pip` or `idle`. `IDLE` can also be found in Start.

All three commands are also available with version number suffixes, for example, as `python3.exe` and `python3.x.exe` as well as `python.x.exe` (where `3.x` is the specific version you want to launch, such as 3.10). Open «Manage App Execution Aliases» through Start to select which version of Python is associated with each command. It is recommended to make sure that `pip` and `idle` are consistent with whichever version of `python` is selected.

Virtual environments can be created with `python -m venv` and activated and used as normal.

If you have installed another version of Python and added it to your `PATH` variable, it will be available as `python.exe` rather than the one from the Microsoft Store. To access the new installation, use `python3.exe` or `python3.x.exe`.

The `py.exe` launcher will detect this Python installation, but will prefer installations from the traditional installer.

To remove Python, open Settings and use Apps and Features, or else find Python in Start and right-click to select Uninstall. Uninstalling will remove all packages you installed directly into this Python installation, but will not remove any virtual environments

### 4.2.1 Known issues

#### Redirection of local data, registry, and temporary paths

Because of restrictions on Microsoft Store apps, Python scripts may not have full write access to shared locations such as `TEMP` and the registry. Instead, it will write to a private copy. If your scripts must modify the shared locations, you will need to install the full installer.

At runtime, Python will use a private copy of well-known Windows folders and the registry. For example, if the environment variable `%APPDATA%` is `c:\Users\<user>\AppData\`, then when writing to `C:\Users\<user>\AppData\Local` will write to `C:\Users\<user>\AppData\Local\Packages\PythonSoftwareFoundation.Python.3.8_qbz5n2kfra8p0\LocalCache\Local\`.

When reading files, Windows will return the file from the private folder, or if that does not exist, the real Windows directory. For example reading `C:\Windows\System32` returns the contents of `C:\Windows\System32` plus the contents of `C:\Program Files\WindowsApps\package_name\VFS\SystemX86`.

You can find the real path of any existing file using `os.path.realpath()`:

```
>>> import os
>>> test_file = 'C:\\Users\\example\\AppData\\Local\\test.txt'
>>> os.path.realpath(test_file)
'C:\\Users\\example\\AppData\\Local\\Packages\\PythonSoftwareFoundation.Python.3.8_
↪qbz5n2kfra8p0\\LocalCache\\Local\\test.txt'
```

When writing to the Windows Registry, the following behaviors exist:

- Reading from HKLM\\Software is allowed and results are merged with the `registry.dat` file in the package.
- Writing to HKLM\\Software is not allowed if the corresponding key/value exists, i.e. modifying existing keys.
- Writing to HKLM\\Software is allowed as long as a corresponding key/value does not exist in the package and the user has the correct access permissions.

For more detail on the technical basis for these limitations, please consult Microsoft's documentation on packaged full-trust apps, currently available at [docs.microsoft.com/en-us/windows/msix/desktop/desktop-to-uwp-behind-the-scenes](https://docs.microsoft.com/en-us/windows/msix/desktop/desktop-to-uwp-behind-the-scenes)

## 4.3 The nuget.org packages

Νέο στην έκδοση 3.5.2.

The nuget.org package is a reduced size Python environment intended for use on continuous integration and build systems that do not have a system-wide install of Python. While nuget is «the package manager for .NET», it also works perfectly fine for packages containing build-time tools.

Visit [nuget.org](https://nuget.org) for the most up-to-date information on using nuget. What follows is a summary that is sufficient for Python developers.

The `nuget.exe` command line tool may be downloaded directly from <https://aka.ms/nugetclidl>, for example, using curl or PowerShell. With the tool, the latest version of Python for 64-bit or 32-bit machines is installed using:

```
nuget.exe install python -ExcludeVersion -OutputDirectory .
nuget.exe install pythonx86 -ExcludeVersion -OutputDirectory .
```

To select a particular version, add a `-Version 3.x.y`. The output directory may be changed from `.`, and the package will be installed into a subdirectory. By default, the subdirectory is named the same as the package, and without the `-ExcludeVersion` option this name will include the specific version installed. Inside the subdirectory is a `tools` directory that contains the Python installation:

```
# Without -ExcludeVersion
> .\python.3.5.2\tools\python.exe -V
Python 3.5.2

# With -ExcludeVersion
> .\python\tools\python.exe -V
Python 3.5.2
```

In general, nuget packages are not upgradeable, and newer versions should be installed side-by-side and referenced using the full path. Alternatively, delete the package directory manually and install it again. Many CI systems will do this automatically if they do not preserve files between builds.

Alongside the `tools` directory is a `build\native` directory. This contains a MSBuild properties file `python.props` that can be used in a C++ project to reference the Python install. Including the settings will automatically use the headers and import libraries in your build.

The package information pages on nuget.org are [www.nuget.org/packages/python](https://www.nuget.org/packages/python) for the 64-bit version and [www.nuget.org/packages/pythonx86](https://www.nuget.org/packages/pythonx86) for the 32-bit version.

## 4.4 The embeddable package

Νέο στην έκδοση 3.5.

The embedded distribution is a ZIP file containing a minimal Python environment. It is intended for acting as part of another application, rather than being directly accessed by end-users.

When extracted, the embedded distribution is (almost) fully isolated from the user's system, including environment variables, system registry settings, and installed packages. The standard library is included as pre-compiled and optimized `.pyc` files in a ZIP, and `python3.dll`, `python37.dll`, `python.exe` and `pythonw.exe` are all provided. Tcl/tk (including all dependants, such as Idle), pip and the Python documentation are not included.

---

**Σημείωση:** The embedded distribution does not include the [Microsoft C Runtime](#) and it is the responsibility of the application installer to provide this. The runtime may have already been installed on a user's system previously or automatically via Windows Update, and can be detected by finding `ucrtbase.dll` in the system directory.

---

Third-party packages should be installed by the application installer alongside the embedded distribution. Using pip to manage dependencies as for a regular Python installation is not supported with this distribution, though with some care it may be possible to include and use pip for automatic updates. In general, third-party packages should be treated as part of the application («vendoring») so that the developer can ensure compatibility with newer versions before providing updates to users.

The two recommended use cases for this distribution are described below.

### 4.4.1 Python Application

An application written in Python does not necessarily require users to be aware of that fact. The embedded distribution may be used in this case to include a private version of Python in an install package. Depending on how transparent it should be (or conversely, how professional it should appear), there are two options.

Using a specialized executable as a launcher requires some coding, but provides the most transparent experience for users. With a customized launcher, there are no obvious indications that the program is running on Python: icons can be customized, company and version information can be specified, and file associations behave properly. In most cases, a custom launcher should simply be able to call `Py_Main` with a hard-coded command line.

The simpler approach is to provide a batch file or generated shortcut that directly calls the `python.exe` or `pythonw.exe` with the required command-line arguments. In this case, the application will appear to be Python and not its actual name, and users may have trouble distinguishing it from other running Python processes or file associations.

With the latter approach, packages should be installed as directories alongside the Python executable to ensure they are available on the path. With the specialized launcher, packages can be located in other locations as there is an opportunity to specify the search path before launching the application.

### 4.4.2 Embedding Python

Applications written in native code often require some form of scripting language, and the embedded Python distribution can be used for this purpose. In general, the majority of the application is in native code, and some part will either invoke `python.exe` or directly use `python3.dll`. For either case, extracting the embedded distribution to a subdirectory of the application installation is sufficient to provide a loadable Python interpreter.

As with the application use, packages can be installed to any location as there is an opportunity to specify search paths before initializing the interpreter. Otherwise, there is no fundamental differences between using the embedded distribution and a regular installation.

## 4.5 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

**Anaconda** Popular scientific modules (such as numpy, scipy and pandas) and the `conda` package manager.

**Enthought Deployment Manager** «The Next Generation Python Environment and Package Manager».

Previously Enthought provided Canopy, but it [reached end of life in 2016](#).

**WinPython** Windows-specific distribution with prebuilt scientific packages and tools for building packages.

Note that these packages may not include the latest versions of Python or other libraries, and are not maintained or supported by the core Python team.

## 4.6 Configuring Python

To run Python conveniently from a command prompt, you might consider changing some default environment variables in Windows. While the installer provides an option to configure the `PATH` and `PATHEXT` variables for you, this is only reliable for a single, system-wide installation. If you regularly use multiple versions of Python, consider using the *Python Launcher for Windows*.

### 4.6.1 Excursus: Setting environment variables

Windows allows environment variables to be configured permanently at both the User level and the System level, or temporarily in a command prompt.

To temporarily set environment variables, open Command Prompt and use the **set** command:

```
C:\>set PATH=C:\Program Files\Python 3.9;%PATH%
C:\>set PYTHONPATH=%PYTHONPATH%;C:\My_python_lib
C:\>python
```

These changes will apply to any further commands executed in that console, and will be inherited by any applications started from the console.

Including the variable name within percent signs will expand to the existing value, allowing you to add your new value at either the start or the end. Modifying `PATH` by adding the directory containing **python.exe** to the start is a common way to ensure the correct version of Python is launched.

To permanently modify the default environment variables, click Start and search for “edit environment variables”, or open System properties, *Advanced system settings* and click the *Environment Variables* button. 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).

---

**Σημείωση:** Windows will concatenate User variables *after* System variables, which may cause unexpected results when modifying `PATH`.

The `PYTHONPATH` variable is used by all versions of Python, so you should not permanently configure it unless the listed paths only include code that is compatible with all of your installed Python versions.

---

**Δείτε επίσης:**

<https://docs.microsoft.com/en-us/windows/win32/procthread/environment-variables> Overview of environment variables on Windows

[https://docs.microsoft.com/en-us/windows-server/administration/windows-commands/set\\_1](https://docs.microsoft.com/en-us/windows-server/administration/windows-commands/set_1) The `set` command, for temporarily modifying environment variables

<https://docs.microsoft.com/en-us/windows-server/administration/windows-commands/setx> The `setx` command, for permanently modifying environment variables

### 4.6.2 Finding the Python executable

Αλλάξε στην έκδοση 3.5.

Besides using the automatically created start menu entry for the Python interpreter, you might want to start Python in the command prompt. The installer has an option to set that up for you.

On the first page of the installer, an option labelled «Add Python to PATH» may be selected to have the installer add the install location into the `PATH`. The location of the `Scripts\` folder is also added. This allows you to type **python** to run the interpreter, and **pip** for the package installer. Thus, you can also execute your scripts with command line options, see *Command line* documentation.

If you don't enable this option at install time, you can always re-run the installer, select Modify, and enable it. Alternatively, you can manually modify the `PATH` using the directions in *Excursus: Setting environment variables*. You need to set your `PATH` environment variable to include the directory of your Python installation, delimited by a semicolon from other entries. An example variable could look like this (assuming the first two entries already existed):

```
C:\WINDOWS\system32;C:\WINDOWS;C:\Program Files\Python 3.9
```

## 4.7 UTF-8 mode

Νέο στην έκδοση 3.7.

Windows still uses legacy encodings for the system encoding (the ANSI Code Page). Python uses it for the default encoding of text files (e.g. `locale.getpreferredencoding()`).

This may cause issues because UTF-8 is widely used on the internet and most Unix systems, including WSL (Windows Subsystem for Linux).

You can use the Python UTF-8 Mode to change the default text encoding to UTF-8. You can enable the Python UTF-8 Mode via the `-X utf8` command line option, or the `PYTHONUTF8=1` environment variable. See *PYTHONUTF8* for enabling UTF-8 mode, and *Excursus: Setting environment variables* for how to modify environment variables.

When the Python UTF-8 Mode is enabled, you can still use the system encoding (the ANSI Code Page) via the «mbcs» codec.

Note that adding `PYTHONUTF8=1` to the default environment variables will affect all Python 3.7+ applications on your system. If you have any Python 3.7+ applications which rely on the legacy system encoding, it is recommended to set the environment variable temporarily or use the `-X utf8` command line option.

---

**Σημείωση:** Even when UTF-8 mode is disabled, Python uses UTF-8 by default on Windows for:

- Console I/O including standard I/O (see **PEP 528** for details).
  - The *filesystem encoding* (see **PEP 529** for details).
-

## 4.8 Python Launcher for Windows

Νέο στην έκδοση 3.3.

The Python launcher for Windows is a utility which aids in locating and executing of different Python versions. It allows scripts (or the command-line) to indicate a preference for a specific Python version, and will locate and execute that version.

Unlike the `PATH` variable, the launcher will correctly select the most appropriate version of Python. It will prefer per-user installations over system-wide ones, and orders by language version rather than using the most recently installed version.

The launcher was originally specified in [PEP 397](#).

### 4.8.1 Getting started

#### From the command-line

Άλλαξε στην έκδοση 3.6.

System-wide installations of Python 3.3 and later will put the launcher on your `PATH`. The launcher is compatible with all available versions of Python, so it does not matter which version is installed. To check that the launcher is available, execute the following command in Command Prompt:

```
py
```

You should find that the latest version of Python you have installed is started - it can be exited as normal, and any additional command-line arguments specified will be sent directly to Python.

If you have multiple versions of Python installed (e.g., 3.7 and 3.10) you will have noticed that Python 3.10 was started - to launch Python 3.7, try the command:

```
py -3.7
```

If you want the latest version of Python 2 you have installed, try the command:

```
py -2
```

You should find the latest version of Python 3.x starts.

If you see the following error, you do not have the launcher installed:

```
'py' is not recognized as an internal or external command,  
operable program or batch file.
```

Per-user installations of Python do not add the launcher to `PATH` unless the option was selected on installation.

The command:

```
py --list
```

displays the currently installed version(s) of Python.

### Virtual environments

Νέο στην έκδοση 3.5.

If the launcher is run with no explicit Python version specification, and a virtual environment (created with the standard library `venv` module or the external `virtualenv` tool) active, the launcher will run the virtual environment's interpreter rather than the global one. To run the global interpreter, either deactivate the virtual environment, or explicitly specify the global Python version.

### From a script

Let's create a test Python script - create a file called `hello.py` with the following contents

```
#!/python
import sys
sys.stdout.write("hello from Python %s\n" % (sys.version,))
```

From the directory in which `hello.py` lives, execute the command:

```
py hello.py
```

You should notice the version number of your latest Python 2.x installation is printed. Now try changing the first line to be:

```
#!/python3
```

Re-executing the command should now print the latest Python 3.x information. As with the above command-line examples, you can specify a more explicit version qualifier. Assuming you have Python 3.7 installed, try changing the first line to `#!/python3.7` and you should find the 3.7 version information printed.

Note that unlike interactive use, a bare `<python>` will use the latest version of Python 2.x that you have installed. This is for backward compatibility and for compatibility with Unix, where the command `python` typically refers to Python 2.

### From file associations

The launcher should have been associated with Python files (i.e. `.py`, `.pyw`, `.pyc` files) when it was installed. This means that when you double-click on one of these files from Windows explorer the launcher will be used, and therefore you can use the same facilities described above to have the script specify the version which should be used.

The key benefit of this is that a single launcher can support multiple Python versions at the same time depending on the contents of the first line.

## 4.8.2 Shebang Lines

If the first line of a script file starts with `#!`, it is known as a «shebang» line. Linux and other Unix like operating systems have native support for such lines and they are commonly used on such systems to indicate how a script should be executed. This launcher allows the same facilities to be used with Python scripts on Windows and the examples above demonstrate their use.

To allow shebang lines in Python scripts to be portable between Unix and Windows, this launcher supports a number of “virtual” commands to specify which interpreter to use. The supported virtual commands are:

- `/usr/bin/env python`
- `/usr/bin/python`
- `/usr/local/bin/python`
- `python`



For example, if the first line of your script starts with

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

The default Python will be located and used. As many Python scripts written to work on Unix will already have this line, you should find these scripts can be used by the launcher without modification. If you are writing a new script on Windows which you hope will be useful on Unix, you should use one of the shebang lines starting with `/usr`.

Any of the above virtual commands can be suffixed with an explicit version (either just the major version, or the major and minor version). Furthermore the 32-bit version can be requested by adding «-32» after the minor version. I.e. `/usr/bin/python3.7-32` will request usage of the 32-bit python 3.7.

Νέο στην έκδοση 3.7: Beginning with python launcher 3.7 it is possible to request 64-bit version by the «-64» suffix. Furthermore it is possible to specify a major and architecture without minor (i.e. `/usr/bin/python3-64`).

The `/usr/bin/env` form of shebang line has one further special property. Before looking for installed Python interpreters, this form will search the executable `PATH` for a Python executable. This corresponds to the behaviour of the Unix `env` program, which performs a `PATH` search.

### 4.8.3 Arguments in shebang lines

The shebang lines can also specify additional options to be passed to the Python interpreter. For example, if you have a shebang line:

```
#!/usr/bin/python -v
```

Then Python will be started with the `-v` option

### 4.8.4 Customization

#### Customization via INI files

Two `.ini` files will be searched by the launcher - `py.ini` in the current user's «application data» directory (i.e. the directory returned by calling the Windows function `SHGetFolderPath` with `CSIDL_LOCAL_APPDATA`) and `py.ini` in the same directory as the launcher. The same `.ini` files are used for both the “console” version of the launcher (i.e. `py.exe`) and for the “windows” version (i.e. `pyw.exe`).

Customization specified in the «application directory» will have precedence over the one next to the executable, so a user, who may not have write access to the `.ini` file next to the launcher, can override commands in that global `.ini` file.

#### Customizing default Python versions

In some cases, a version qualifier can be included in a command to dictate which version of Python will be used by the command. A version qualifier starts with a major version number and can optionally be followed by a period (“.”) and a minor version specifier. Furthermore it is possible to specify if a 32 or 64 bit implementation shall be requested by adding «-32» or «-64».

For example, a shebang line of `#!/python` has no version qualifier, while `#!/python3` has a version qualifier which specifies only a major version.

If no version qualifiers are found in a command, the environment variable `PY_PYTHON` can be set to specify the default version qualifier. If it is not set, the default is «3». The variable can specify any value that may be passed on the command line, such as «3», «3.7», «3.7-32» or «3.7-64». (Note that the «-64» option is only available with the launcher included with Python 3.7 or newer.)

If no minor version qualifiers are found, the environment variable `PY_PYTHON{major}` (where {major} is the current major version qualifier as determined above) can be set to specify the full version. If no such option is found,

the launcher will enumerate the installed Python versions and use the latest minor release found for the major version, which is likely, although not guaranteed, to be the most recently installed version in that family.

On 64-bit Windows with both 32-bit and 64-bit implementations of the same (major.minor) Python version installed, the 64-bit version will always be preferred. This will be true for both 32-bit and 64-bit implementations of the launcher - a 32-bit launcher will prefer to execute a 64-bit Python installation of the specified version if available. This is so the behavior of the launcher can be predicted knowing only what versions are installed on the PC and without regard to the order in which they were installed (i.e., without knowing whether a 32 or 64-bit version of Python and corresponding launcher was installed last). As noted above, an optional «-32» or «-64» suffix can be used on a version specifier to change this behaviour.

Examples:

- If no relevant options are set, the commands `python` and `python2` will use the latest Python 2.x version installed and the command `python3` will use the latest Python 3.x installed.
- The command `python3.7` will not consult any options at all as the versions are fully specified.
- If `PY_PYTHON=3`, the commands `python` and `python3` will both use the latest installed Python 3 version.
- If `PY_PYTHON=3.7-32`, the command `python` will use the 32-bit implementation of 3.7 whereas the command `python3` will use the latest installed Python (`PY_PYTHON` was not considered at all as a major version was specified.)
- If `PY_PYTHON=3` and `PY_PYTHON3=3.7`, the commands `python` and `python3` will both use specifically 3.7

In addition to environment variables, the same settings can be configured in the .INI file used by the launcher. The section in the INI file is called `[defaults]` and the key name will be the same as the environment variables without the leading `PY_` prefix (and note that the key names in the INI file are case insensitive.) The contents of an environment variable will override things specified in the INI file.

For example:

- Setting `PY_PYTHON=3.7` is equivalent to the INI file containing:

```
[defaults]
python=3.7
```

- Setting `PY_PYTHON=3` and `PY_PYTHON3=3.7` is equivalent to the INI file containing:

```
[defaults]
python=3
python3=3.7
```

### 4.8.5 Diagnostics

If an environment variable `PYLAUNCH_DEBUG` is set (to any value), the launcher will print diagnostic information to stderr (i.e. to the console). While this information manages to be simultaneously verbose *and* terse, it should allow you to see what versions of Python were located, why a particular version was chosen and the exact command-line used to execute the target Python.

## 4.9 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\`.

To completely override `sys.path`, create a `._pth` file with the same name as the DLL (`python37._pth`) or the executable (`python._pth`) and specify one line for each path to add to `sys.path`. The file based on the DLL name overrides the one based on the executable, which allows paths to be restricted for any program loading the runtime if desired.

When the file exists, all registry and environment variables are ignored, isolated mode is enabled, and `site` is not imported unless one line in the file specifies `import site`. Blank paths and lines starting with `#` are ignored. Each path may be absolute or relative to the location of the file. Import statements other than to `site` are not permitted, and arbitrary code cannot be specified.

Note that `._pth` files (without leading underscore) will be processed normally by the `site` module when `import site` has been specified.

When no `._pth` file is found, 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» (either `Lib\os.py` or `pythonXY.zip`) 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).

If a `pyvenv.cfg` file is found alongside the main executable or in the directory one level above the executable, the following variations apply:

- If `home` is an absolute path and `PYTHONHOME` is not set, this path is used instead of the path to the main executable when deducing the home location.

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 are no registry value (frozen `.exe`, some very strange installation setup) you get a path with some default, but relative, paths.

For those who want to bundle Python into their application or distribution, the following advice will prevent conflicts with other installations:

- Include a `._pth` file alongside your executable containing the directories to include. This will ignore paths listed in the registry and environment variables, and also ignore `site` unless `import site` is listed.

- If you are loading `python3.dll` or `python37.dll` in your own executable, explicitly call `Py_SetPath()` or (at least) `Py_SetProgramName()` before `Py_Initialize()`.
- Clear and/or overwrite `PYTHONPATH` and set `PYTHONHOME` before launching `python.exe` from your application.
- If you cannot use the previous suggestions (for example, you are a distribution that allows people to run `python.exe` directly), ensure that the landmark file (`Lib\os.py`) exists in your install directory. (Note that it will not be detected inside a ZIP file, but a correctly named ZIP file will be detected instead.)

These will ensure that the files in a system-wide installation will not take precedence over the copy of the standard library bundled with your application. Otherwise, your users may experience problems using your application. Note that the first suggestion is the best, as the others may still be susceptible to non-standard paths in the registry and user site-packages.

Άλλαξε στην έκδοση 3.6:

- Adds `._pth` file support and removes `applocal` option from `pyvenv.cfg`.
- Adds `pythonXX.zip` as a potential landmark when directly adjacent to the executable.

Αποσύρθηκε στην έκδοση 3.6: Modules specified in the registry under Modules (not `PythonPath`) may be imported by `importlib.machinery.WindowsRegistryFinder`. This finder is enabled on Windows in 3.6.0 and earlier, but may need to be explicitly added to `sys.meta_path` in the future.

## 4.10 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`.

### 4.10.1 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.

**Δείτε επίσης:**

**Win32 How Do I...?** by Tim Golden

**Python and COM** by David and Paul Boddie

### 4.10.2 cx\_Freeze

`cx_Freeze` 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.

## 4.11 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](#).

The source tree contains a build solution and project files for Microsoft Visual Studio, which is the compiler used to build the official Python releases. These files are in the `PCbuild` directory.

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

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

## 4.12 Other Platforms

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.

- [Windows CE](#) is [no longer supported](#) since Python 3 (if it ever was).
- The [Cygwin](#) installer offers to install the [Python interpreter](#) as well

See [Python for Windows](#) for detailed information about platforms with pre-compiled installers.



---

## Using Python on a Mac

---

**Author** Bob Savage <bobsavage@mac.com>

Python on a Mac running macOS 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.

### 5.1 Getting and Installing MacPython

macOS used to come with Python 2.7 pre-installed between versions 10.8 and 12.3. You are invited to install the most recent version of Python 3 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 `Python 3.12` folder in your `Applications` folder. In here you find `IDLE`, the development environment that is a standard part of official Python distributions; and `PythonLauncher`, which handles double-clicking Python scripts from the Finder.
- 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.

### 5.1.1 How to run a Python script

Your best way to get started with Python on macOS 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. macOS 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** (<https://macvim-dev.github.io/macvim/>) 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.

### 5.1.2 Running scripts with a GUI

With older versions of Python, there is one macOS 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 3.9, you can use either **python** or **pythonw**.

### 5.1.3 Configuration

Python on macOS 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*.

## 5.2 The IDE

MacPython ships with the standard IDLE development environment. A good introduction to using IDLE can be found at [http://www.hashcollision.org/hkn/python/idle\\_intro/index.html](http://www.hashcollision.org/hkn/python/idle_intro/index.html).



## 5.3 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/>.

## 5.4 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://pypi.org/project/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 macOS. Packages and documentation are available from <https://www.wxpython.org>.

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

## 5.5 Distributing Python Applications on the Mac

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

## 5.6 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>



---

### Επεξεργαστές Κειμένου και IDEs

---

Υπάρχουν αρκετά IDEs που υποστηρίζουν τη γλώσσα προγραμματισμού Python. Αρκετοί επεξεργαστές κειμένου και IDEs παρέχουν επισήμανση σύνταξης, εργαλεία αποσφαλμάτωσης και ελέγχους **PEP 8**.

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



>>> Το προεπιλεγμένο Python prompt του διαδραστικού shell. Συχνά εμφανίζεται για παραδείγματα κώδικα που μπορούν να εκτελεστούν διαδραστικά στον interpreter.

... Μπορεί να αναφέρεται σε:

- Το προεπιλεγμένο Python prompt του διαδραστικού shell κατά την εισαγωγή του κώδικα για ένα μπλοκ κώδικα με εσοχή, όταν βρίσκεται μέσα σε ένα ζεύγος ταιριασμένων αριστερών και δεξιών delimiters (παρενθέσεις, αγκύλες, άγκιστρα ή τριπλά εισαγωγικά), ή μετά τον καθορισμό ενός decorator.
- Η ενσωματωμένη σταθερά Ellipsis.

**2to3** Ένα εργαλείο που προσπαθεί να μετατρέψει τον κώδικα Python 2.x σε κώδικα Python 3.x διαχειρίζοντας τις περισσότερες ασυμβατότητες που μπορούν να εντοπιστούν αναλύοντας την πηγή και διασχίζοντας το δέντρο ανάλυσης.

2to3 είναι διαθέσιμο στην στάνταρ βιβλιοθήκη ως `lib2to3`, παρέχεται ένα σημείο εισόδου ως `Tools/scripts/2to3`. Βλ. [2to3-reference](#).

**αφηρημένη βασική κλάση** Οι αφηρημένες βασικές κλάσεις συμπληρώνουν το *duck-typing* παρέχοντας έναν τρόπο ορισμού interfaces όταν άλλες τεχνικές όπως η `hasattr()` θα ήταν αδέξιες ή ανεπαίσθητα λανθασμένες (για παράδειγμα με magic methods). Τα ABC (abstract base class) εισάγουν εικονικές υποκλάσεις, οι οποίες είναι κλάσεις που δεν κληρονομούνται από μια κλάση, αλλά εξακολουθούν να αναγνωρίζονται από το `isinstance()` και από το `issubclass()`” βλ. την τεκμηρίωση του `module abc`. Η Python διαθέτει πολλά ενσωματωμένα ABC για δομές δεδομένων (στο `module collections.abc`), αριθμούς (στο `module numbers`), ροές (στο `module monada io`), εισαγωγή finders και loaders (στο `module importlib.abc`). Μπορείτε να δημιουργήσετε τα δικά σας ABC με το `module abc`.

**annotation** Μια ετικέτα που σχετίζεται με μια μεταβλητή, ένα χαρακτηριστικό κλάσης ή μια παράμετρος συνάρτησης ή τιμή που επιστρέφεται, που χρησιμοποιείται κατά σύμβαση ως *type hint*.

Δεν είναι δυνατή η πρόσβαση στα annotations των τοπικών μεταβλητών κατά το χρόνο εκτέλεσης, αλλά τα annotations των global μεταβλητών, των χαρακτηριστικών κλάσης και των συναρτήσεων αποθηκεύονται στο ειδικό χαρακτηριστικό `__annotations__` των modules, των κλάσεων και των συναρτήσεων, αντίστοιχα.

Βλ. *variable annotation*, *function annotation*, **PEP 484** και **PEP 526**, τα οποία περιγράφουν την λειτουργικότητα. Επίσης βλ. *annotations-howto* για τις βέλτιστες πρακτικές δουλεύοντας με annotations.

**όρισμα** Μια τιμή μεταβιβάζεται σε μία *function* (ή *method*) κατά την κλήση της συνάρτησης. Υπάρχουν δύο είδη ορισμάτων:

- *keyword argument*: ένα όρισμα πριν από ένα αναγνωριστικό (π.χ. `name=`) σε μια κλήση συνάρτησης ή περνώντας το ως τιμή σε ένα λεξικό πριν από `**`. Για παράδειγμα, το 3 και το 5 αποτελούν ορίσματα λέξεων-κλειδίων στις ακόλουθες κλήσεις προς `complex()`:

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

- *positional argument*: ένα όρισμα που δεν είναι όρισμα keyword. Τα ορίσματα θέσης μπορούν να εμφανίζονται στην αρχή μιας λίστας ορισμάτων ή/και να μεταβιβάζονται ως στοιχεία ενός *iterable* πριν από `*`. Για παράδειγμα, το 3 και το 5 αποτελούν ορίσματα θέσης στις παρακάτω κλήσεις:

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

Τα ορίσματα εκχωρούνται στις ονομασμένες τοπικές μεταβλητές στο σώμα μια συνάρτησης. Βλ. την ενότητα *calls* για τους κανόνες που διέπουν αυτήν την εκχώρηση. Συντακτικά, οποιαδήποτε έκφραση μπορεί να χρησιμοποιηθεί για να αναπαραστήσει ένα όρισμα” η αξιολογούμενη τιμή εκχωρείται σε μια τοπική μεταβλητή.

Βλ. επίσης την εγγραφή του γλωσσarium για το *parameter*, την FAQ ερώτηση στο η διαφορά μεταξύ ορισμάτων και παραμέτρων, και **PEP 362**.

**ασύγχρονος διαχειριστής context** An object which controls the environment seen in an `async with` statement by defining `__aenter__()` and `__aexit__()` methods. Introduced by **PEP 492**.

**ασύγχρονος generator** Μια συνάρτηση που επιστρέφει έναν *asynchronous generator iterator*. Μοιάζει με μια συνάρτηση *coroutine* που ορίζεται με `async def` εκτός από ότι περιέχει εκφράσεις `yield` για την παραγωγή μιας σειράς τιμών που μπορούν να χρησιμοποιηθούν σε έναν `async for` βρόχο.

Συνήθως αναφέρεται σε μια συνάρτηση ασύγχρονου generator, αλλά μπορεί να αναφέρεται σε έναν *asynchronous generator iterator* σε ορισμένα contexts. Σε περιπτώσεις όπου το επιδιωκόμενο νόημα δεν είναι σαφές, με την χρήση των πλήρων όρων αποφεύγεται η ασάφεια.

Μια συνάρτηση ασύγχρονου generator μπορεί να περιέχει εκφράσεις `await`, καθώς και δηλώσεις `async for`, και `async with`.

**ασύγχρονος generator iterator** Ένα αντικείμενο που δημιουργήθηκε από μια συνάρτηση *asynchronous generator*.

This is an *asynchronous iterator* which when called using the `__anext__()` method returns an awaitable object which will execute the body of the asynchronous generator function until the next `yield` expression.

Each `yield` temporarily suspends processing, remembering the location execution state (including local variables and pending try-statements). When the *asynchronous generator iterator* effectively resumes with another awaitable returned by `__anext__()`, it picks up where it left off. See **PEP 492** and **PEP 525**.

**ασύγχρονος iterable** An object, that can be used in an `async for` statement. Must return an *asynchronous iterator* from its `__aiter__()` method. Introduced by **PEP 492**.

**ασύγχρονος iterator** An object that implements the `__aiter__()` and `__anext__()` methods. `__anext__()` must return an *awaitable* object. `async for` resolves the awaitables returned by an asynchronous iterator’s `__anext__()` method until it raises a `StopAsyncIteration` exception. Introduced by **PEP 492**.

**χαρακτηριστικό** Μια τιμή που σχετίζεται με ένα αντικείμενο που συνήθως αναφέρεται με όνομα χρησιμοποιώντας εκφράσεις με κουκκίδες. Για παράδειγμα, εάν ένα αντικείμενο *o* έχει ένα χαρακτηριστικό *a* θα αναφέρεται ως *o.a*.

Είναι δυνατό να δώσουμε σε ένα αντικείμενο ένα χαρακτηριστικό που το όνομα του δεν είναι αναγνωριστικό όπως ορίζεται από *identifiers*, για παράδειγμα χρησιμοποιώντας `setattr()`, αν επιτρέπεται από το αντικείμενο. Ένα τέτοιο χαρακτηριστικό δεν θα είναι προσβάσιμο χρησιμοποιώντας τις τελείες, και αντί αυτού θα πρέπει να ανακτηθεί χρησιμοποιώντας `getattr()`.

**awaitable** An object that can be used in an `await` expression. Can be a *coroutine* or an object with an `__await__()` method. See also **PEP 492**.

**BDFL** Ακρωνύμιο του *Benevolent Dictator For Life*, καλοκάγαθος δικτάτορας της ζωής, δηλαδή [Guido van Rossum](#), ο δημιουργός της Python.

**δυναδικό αρχείο** A *file object* able to read and write *bytes-like objects*. Examples of binary files are files opened in binary mode ('rb', 'wb' or 'rb+'), `sys.stdin.buffer`, `sys.stdout.buffer`, and instances of `io.BytesIO` and `gzip.GzipFile`.

Βλ. επίσης *text file* για ένα αντικείμενο τύπου αρχείο ικανό να διαβάσει και να γράψει `str` αντικείμενα.

**δανεική αναφορά** Στο C API της Python, μια δανεική αναφορά είναι μια αναφορά σε ένα αντικείμενο, όπου ο κώδικας που χρησιμοποιεί το αντικείμενο δεν κατέχει την αναφορά. Γίνεται ένας αχρησιμοποίητος δείκτης εάν το αντικείμενο καταστραφεί. Για παράδειγμα, μια διαδικασία garbage collection μπορεί να αφαιρέσει το τελευταίο *strong reference* από το αντικείμενο και έτσι να το καταστρέψει.

Συνίσταται η κλήση του `Py_INCREF()` στο *δανεική αναφορά* με σκοπό να μετατραπεί σε ένα *ισχυρή αναφορά* επιτόπου, εκτός όταν το αντικείμενο δεν μπορεί να καταστραφεί πριν από την τελευταία χρήση της δανεικής αναφοράς. Η συνάρτηση `Py_NewRef()` μπορεί να χρησιμοποιηθεί ώστε να δημιουργηθεί ένα *ισχυρή αναφορά*.

**bytes-like αντικείμενα** Ένα αντικείμενο που υποστηρίζει το `bufferobjects` και μπορεί να εξάγει ένα *C-contiguous* buffer. Αυτό περιλαμβάνει όλα τα αντικείμενα `bytes`, `bytearray`, και `array.array`, καθώς και πολλά κοινά `memoryview` αντικείμενα. Τα δυναδικού τύπου (bytes-like) αντικείμενα μπορούν να χρησιμοποιηθούν για διάφορες λειτουργίες που διαχειρίζονται δυναδικά δεδομένα" αυτά περιλαμβάνουν συμπίεση αποθήκευση σε δυναδικό αρχείο και αποστολή μέσω socket.

Ορισμένες λειτουργίες χρειάζονται τα δυναδικά δεδομένα να είναι μεταβλητά. Η τεκμηρίωση συχνά αναφέρεται σε αυτά ως «δυναδικά αντικείμενα ανάγνωσης-εγγραφής» (read-write bytes-like objects). Παραδείγματα μεταβλητών αντικειμένων προσωρινής αποθήκευσης περιέχουν `bytearray` και ένα `memoryview` ενός `bytearray`. Άλλες λειτουργίες απαιτούν την αποθήκευσης των δυναδικών δεδομένα σε αμετάβλητα αντικείμενα («δυναδικά αντικείμενα μόνο ανάγνωσης» (read-only bytes-like objects) παραδείγματα αυτών περιέχουν `bytes` και ένα `memoryview` ενός `bytes` αντικειμένου.

**bytecode** Ο πηγαίος κώδικας της Python μεταγλωττίζεται σε *bytecode*, η εσωτερική αναπαράσταση ενός προγράμματος Python στον διερμηνέα CPython. Το *bytecode* αποθηκεύεται επίσης προσωρινά ως `.pyc` αρχεία ώστε η εκτέλεση του ίδιου αρχείου να είναι γρηγορότερη την δεύτερη φορά εκτέλεσης (μπορεί να αποφευχθεί η εκ νέου μεταγλώττιση από τον πηγαίο κώδικα σε *bytecode*). Αυτή η «ενδιάμεση γλώσσα» λέγεται ότι τρέχει σε μια *virtual machine* που εκτελεί τον κώδικα μηχανής που αντιστοιχεί σε κάθε *bytecode*. Λάβετε υπόψη ότι τα *bytecode* δεν αναμένεται να λειτουργούν μεταξύ διαφορετικών εικονικών μηχανών Python, ούτε να είναι σταθερά μεταξύ των εκδόσεων της Python.

Μια λίστα από οδηγίες σχετικά με τα *bytecode* μπορεί να βρεθεί στην τεκμηρίωση για το module `dis`.

**callable** Ένα callable είναι ένα αντικείμενο που μπορεί να καλεστεί, πιθανά με ένα σύνολο ορισμάτων (βλ. *argument*), με την παρακάτω σύνταξη:

```
callable(argument1, argument2, ...)
```

Μια *function*, και κατ'επέκταση μια *method* είναι callable. Ένα στιγμιότυπο μια κλάσης που υλοποιεί τη μέθοδο `__call__()` είναι επίσης callable.

**callback** Μια subroutine συνάρτηση η οποία μεταβιβάζεται ως όρισμα που θα εκτελεστεί κάποια στιγμή στο μέλλον.

**κλάση** Ένα πρότυπο για τη δημιουργία αντικειμένων που ορίζονται από το χρήστη. Οι ορισμοί κλάσεων συνήθως περιέχουν ορισμούς μεθόδων που λειτουργούν σε στιγμιότυπα της κλάσης.

**μεταβλητή κλάσης** Μια μεταβλητή που ορίζεται σε μια κλάση και προορίζεται να τροποποιηθεί μόνο σε επίπεδο κλάσης (δηλ. όχι σε ένα στιγμιότυπο μιας κλάσης).

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

**μυγαδικός αριθμός** Μια επέκταση του γνωστού συστήματος πραγματικών αριθμών στο οποίο όλοι οι αριθμοί εκφράζονται ως άθροισμα ενός πραγματικού μέρους και ενός φανταστικού μέρους. Οι φανταστικοί αριθμοί είναι πραγματικά πολλαπλάσια της φανταστικής μονάδα (η τετραγωνική ρίζα του  $-1$ ), που συχνά γράφονται  $i$  στα μαθηματικά ή  $j$  στη μηχανική. Η Python έχει ενσωματωμένη υποστήριξη για μυγαδικούς αριθμούς, οι οποίοι γράφονται με αυτόν τον τελευταίο συμβολισμό” το φανταστικό μέρος γράφεται με το επίθημα  $j$ , π.χ.,  $3+1j$ . Για να αποκτήσετε πρόσβαση σε σύνθετα ισοδύναμα το module `math`, χρησιμοποιήστε το `cmath`. Η χρήση μυγαδικών αριθμών είναι ένα αρκετά προηγμένο μαθηματικό χαρακτηριστικό. εάν δεν γνωρίζετε την ανάγκη τους, είναι σχεδόν σίγουρο ότι μπορείτε να τα αγνοήσετε με ασφάλεια.

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

**context μεταβλητή** Μια μεταβλητή που μπορεί να έχει πολλές διαφορετικές τιμές ανάλογα με το context. Αυτό είναι κοινό στο Thread-Local Storage όπου κάθε εκτέλεση του νήματος μπορεί να έχει διαφορετική τιμή για μια μεταβλητή. Παρόλα αυτά, με τις context μεταβλητές, μπορεί να υπάρχουν πολλά περιβάλλοντα σε ένα νήμα εκτέλεσης και η κύρια χρήση για τις context μεταβλητές είναι η παρακολούθηση των μεταβλητών σε ταυτόχρονες διεργασίες. Βλ. `contextvars`.

**contiguous** Ένα buffer θεωρείται contiguous ακριβώς εάν είναι είτε *C-contiguous* είτε *Fortran contiguous*. Το buffer μηδενικών διαστάσεων είναι C και Fortran contiguous. Σε μονοδιάστατους πίνακες, τα στοιχεία πρέπει να τοποθετούνται στη μνήμη το ένα δίπλα στο άλλο, με σειρά αύξησης των δεικτών ξεκινώντας από το μηδέν. Σε πολυδιάστατους C-contiguous πίνακες, ο τελευταίος δείκτης μεταβάλλεται ταχύτερα όταν επισκέπτονται τα στοιχεία σε σειρά διεύθυνσης μνήμης. Ωστόσο, σε Fortran contiguous πίνακες, ο πρώτος δείκτης μεταβάλλεται πιο γρήγορα.

**coroutine** Οι coroutines είναι μια πιο γενικευμένη μορφή subroutines. Οι subroutines εισάγονται σε ένα σημείο και εξάγονται σε άλλο σημείο. Οι coroutines μπορεί να εισαχθούν, να εξαχθούν και να συνεχιστούν σε πολλά διαφορετικά σημεία. Μπορούν να υλοποιηθούν με την δήλωση `async def`. Βλ. επίσης [PEP 492](#).

**coroutine συνάρτηση** Μια συνάρτηση που επιστρέφει ένα *coroutine* αντικείμενο. Μια συνάρτηση coroutine μπορεί να ορίζεται από τη δήλωση `async def`, και μπορεί να περιέχει `await`, `async for`, και `async with` λέξεις κλειδιά. Αυτές εισήχθησαν από το [PEP 492](#).

**CPython** Η κανονική υλοποίηση της γλώσσας προγραμματισμού Python, όπως διανέμεται στο [python.org](#). Ο όρος «CPython» χρησιμοποιείται όταν είναι απαραίτητο για την διάκριση αυτής της υλοποίησης από άλλες όπως η *Jython* ή η *IronPython*.

**decorator** Μια συνάρτηση που επιστρέφει μια άλλη συνάρτηση, συνήθως εφαρμόζεται ως μετασχηματισμός συνάρτησης χρησιμοποιώντας την `@wrapper` σύνταξη. Συνηθισμένα παραδείγματα για τους decorators είναι `classmethod()` και `staticmethod()`.

Η σύνταξη του decorator είναι απλώς καλλωπιστική, οι ακόλουθοι δύο ορισμοί συναρτήσεων είναι σημασιολογικά ισοδύναμοι:

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

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

Η ίδια έννοια υπάρχει για τις κλάσεις, αλλά χρησιμοποιείται λιγότερο συχνά εκεί. Βλ. την τεκμηρίωση για function definitions και class definitions για περισσότερα σχετικά με τους decorators.

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



Για περισσότερες πληροφορίες αναφορικά με τις μεθόδους των descriptors, βλ. [see descriptors](#) ή το Πρακτικός οδηγός για τη χρήση του Descriptor.

**λεξικό** 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.

**κατανόηση λεξικού** Ένα συμπαγής τρόπος για να επεξεργαστείτε όλα ή μέρος των στοιχείων σε ένα επαναληπτικό και να επιστραφεί ένα με λεξικό με τα αποτελέσματα. `results = {n: n ** 2 for n in range(10)}` δημιουργεί ένα λεξικό που περιέχει το κλειδί `n` που αντιστοιχίζεται με την τιμή `n ** 2`. Βλ. [comprehensions](#).

**όψη λεξικού** Τα αντικείμενα που επιστρέφονται από `dict.keys()`, `dict.values()`, και `dict.items()` καλούνται όψεις λεξικού. Αυτές παρέχουν μια δυναμική όψη των των εγγραφών του λεξικού, που σημαίνει ότι όταν το λεξικό μεταβάλλεται, η όψη αντικατοπτρίζει αυτές τις αλλαγές. Για να αναγκάσετε την όψη λεξικού να γίνει μια πλήρης λίστα χρησιμοποιήστε το `list(dictview)`. Βλ. [dict-views](#).

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

**duck-typing** Ένα στυλ προγραμματισμού που δεν εξετάζει τον τύπο ενός αντικειμένου για να προσδιορίσει αν έχει τη σωστή διεπαφή αντίθετα, η μέθοδος ή το χαρακτηριστικό καλείται απλώς ή χρησιμοποιείται («If it looks like a duck and quacks like a duck, it must be a duck.») Δίνοντας έμφαση στις διεπαφές και όχι σε συγκεκριμένους τύπους, ο καλά σχεδιασμένος κώδικας βελτιώνει την ευελιξία του επιτρέποντας την πολυμορφική υποκατάσταση. Ο τύπος duck-typing αποφεύγει δοκιμές χρησιμοποιώντας `type()` ή `isinstance()`. (Σημείωση, ωστόσο, ότι ο τύπος πάπιας *duck-typing* μπορεί να συμπληρωθεί με [abstract base classes](#).) Αντί αυτού, συνήθως χρησιμοποιεί δοκιμές `hasattr()` ή προγραμματισμό [EAFP](#).

**EAFP** Πιο εύκολο να ζητήσεις συγχώρεση παρά άδεια. Αυτό το κοινό στυλ προγραμματισμού σε Python προϋποθέτει την ύπαρξη έγκυρων κλειδιών ή χαρακτηριστικών και συλλαμβάνει εξαιρέσεις εάν η υπόθεση αποδεχθεί εσφαλμένη. Αυτό το καθαρό και γρήγορο στυλ χαρακτηρίζεται από την παρουσία πολλών δηλώσεων `try` και `except`. Η τεχνική έρχεται σε αντίθεση με το στυλ που είναι [LBYL](#) κοινό σε πολλές άλλες γλώσσες, όπως η C.

**έκφραση** Ένα κομμάτι σύνταξης που μπορεί να αξιολογηθεί σε κάποια τιμή. Με άλλα λόγια, μια έκφραση είναι μια συσσώρευση στοιχείων έκφρασης όπως κυριολεξία, ονόματα, πρόσβαση χαρακτηριστικών, τελεστές ή κλήσεις συναρτήσεων που όλες επιστρέφουν μια τιμή. Σε αντίθεση με πολλές άλλες γλώσσες, δεν είναι όλες οι γλωσσικές δομές εκφράσεις. Υπάρχουν επίσης [statements](#) που δεν μπορούν να χρησιμοποιηθούν ως εκφράσεις, όπως το `while`. Οι αναθέσεις τιμών είναι επίσης δηλώσεις όχι εκφράσεις.

**module επέκτασης** Ένα module γραμμένο σε C ή C++, που χρησιμοποιείται από το C API της Python για να αλληλεπιδράσουν με τον πυρήνα και με τον κώδικα του χρήστη.

**f-string** Οι κυριολεκτικές συμβολοσειρές χρησιμοποιούν με πρόθεμα `'f'` ή `'F'` ονομάζονται συνήθως «f-strings» που είναι συντομογραφία του formatted string literals. Βλ. επίσης [PEP 498](#).

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

Στην πραγματικότητα υπάρχουν τρεις κατηγορίες αντικειμένων αρχείου raw *δυναμικά αρχεία*, buffered *δυναμικά αρχεία* και *αρχεία κειμένου*. Οι διεπαφές τους ορίζονται στην ενότητα [io](#). Ο κανονικός τρόπος για να δημιουργήσετε ένα αντικείμενο αρχείου είναι χρησιμοποιώντας την συνάρτηση `open()`.

**αντικείμενο που μοιάζει με αρχείο** Ένα συνώνυμο με το [file object](#).

**κωδικοποίηση συστήματος αρχείων και χειριστής σφαλμάτων** Η κωδικοποίηση και ο χειριστής σφαλμάτων χρησιμοποιείται από την Python για την αποκωδικοποίηση των bytes από το λειτουργικό σύστημα και την κωδικοποίηση σε Unicode για το λειτουργικό σύστημα.

Η κωδικοποίηση συστήματος αρχείων μπορεί να εγγυηθεί την επιτυχημένη αποκωδικοποίηση όλων των bytes κάτω από 128. Εάν η κωδικοποίηση συστήματος αρχείων δεν παρέχει αυτήν την εγγύηση, οι συναρτήσεις API μπορούν να εγείρουν ένα `UnicodeError`.

Οι συναρτήσεις `sys.getfilesystemencoding()` και `sys.getfilesystemencodeerrors()` μπορούν να χρησιμοποιηθούν για να λάβετε την κωδικοποίηση του συστήματος αρχείων και του χειριστή σφαλμάτων.

Ο *filesystem encoding and error handler* διαμορφώνονται κατά την εκκίνηση της Python από τη συνάρτηση `PyConfig_Read()` βλ. `filesystem_encoding` και `filesystem_errors` μέλη του `PyConfig`.

Βλ. επίσης το *locale encoding*.

**finder** Ένα αντικείμενο που προσπαθεί να βρει το *loader* για ένα module που εισήχθη.

Since Python 3.3, there are two types of finder: *meta path finders* for use with `sys.meta_path`, and *path entry finders* for use with `sys.path_hooks`.

See [PEP 302](#), [PEP 420](#) and [PEP 451](#) for much more detail.

**ακέραια διαίρεση** Η μαθηματική διαίρεση που στρογγυλοποιεί προς τα κάτω στον κοντινότερο ακέραιο. Ο τελεστής ακέραιας διαίρεσης είναι `//`. Για παράδειγμα, η έκφραση `11 // 4` αξιολογείται σε 2 σε αντίθεση με την τιμή `2.75` που επιστρέφεται από την διαίρεση με υποδιαστολή. Σημείωση ότι `(-11) // 4` κάνει `-3` επειδή αυτή είναι η στρογγυλοποίηση προς τα κάτω του `-2.75`. Βλ. [PEP 238](#).

**συνάρτηση** Μια σειρά από δηλώσεις που επιστρέφουν κάποια τιμή σε αυτόν που την κάλεσε. Σε αυτές μπορούν να περαστούν κανένα ή περισσότερα *ορίσματα* που μπορεί να χρησιμοποιηθεί για την εκτέλεση. Βλ. επίσης τις ενότητες *parameter*, *method*, και *the function*.

**συνάρτηση annotation** Ένας *annotation* μιας παραμέτρου συνάρτησης ή μιας τιμής επιστροφής.

Οι συναρτήσεις annotations συχνά χρησιμοποιούνται για *υποδείξεις τύπου*: για παράδειγμα, αυτή η συνάρτηση αναμένεται να πάρει δύο ορίσματα `int` και επίσης αναμένεται να έχει μία επιστρεφόμενη τιμή `int`:

```
def sum_two_numbers(a: int, b: int) -> int:
    return a + b
```

Η σύνταξη συνάρτησης annotation αναλύεται στην ενότητα *function*.

Βλ. *variable annotation* και [PEP 484](#), που περιγράφει αυτή την λειτουργικότητα. Επίσης βλ. *annotations-howto* για τις καλύτερες πρακτικές δουλεύοντας με annotations.

**\_\_future\_\_** Ένα future statement, `from __future__ import <feature>`, καθοδηγεί τον μεταγλωττιστή να μεταγλωττίσει το τρέχον module χρησιμοποιώντας σύνταξη ή σημασιολογία που θα γίνει η τυπική σε μελλοντική έκδοση της Python. Το module `__future__` τεκμηριώνει τις πιθανές τιμές του *feature*. Με την εισαγωγή αυτής της λειτουργικής μονάδας και την αξιολόγηση των μεταβλητών της, μπορείτε να δείτε πότε μια νέα δυνατότητα προστέθηκε για πρώτη φορά στην γλώσσα και πότε θα γίνει (ή έγινε) η προεπιλογή:

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

**συλλογή απορριμάτων** Η διαδικασία απελευθέρωσης της μνήμης όταν δεν χρησιμοποιείται άλλο. Η Python εκτελεί συλλογή απορριμάτων μέσω καταμέτρησης αναφορών και ενός κυκλικού συλλέκτη σκουπιδιών που είναι σε θέση να ανιχνεύει και να σπάει τους κύκλους αναφοράς. Ο συλλέκτης απορριμάτων μπορεί να ελεγχθεί χρησιμοποιώντας το module `gc`.

**generator** Μια συνάρτηση που επιστρέφει ένα *generator iterator*. Μοιάζει με μια κανονική συνάρτηση εκτός από το ότι περιέχει εκφράσεις `yield` για την παραγωγή μιας σειράς τιμών που μπορούν να χρησιμοποιηθούν σε έναν βρόχο *for* ή που μπορούν να ανακτηθούν μία τη φορά με την συνάρτηση `next()` function.

Συνήθως αναφέρεται σε μια συνάρτηση *generator*, αλλά μπορεί να αναφέρεται σε έναν *generator iterator* σε μερικά contexts. Σε περιπτώσεις όπου το επιδιωκόμενο νόημα δεν είναι σαφές, η χρήση των πλήρων όρων αποφεύγει την ασάφεια.

**generator iterator** Ένα αντικείμενο που δημιουργείται από μια συνάρτηση *generator*.

Each `yield` temporarily suspends processing, remembering the location execution state (including local variables and pending try-statements). When the *generator iterator* resumes, it picks up where it left off (in contrast to functions which start fresh on every invocation).

**generator έκφραση** Μια έκφραση που επιστρέφει έναν iterator. Μοιάζει με κανονική έκφραση που ακολουθείται από μια πρόταση `for` που ορίζει μια μεταβλητή βρόχου, ένα εύρος και μια προαιρετική πρόταση `if`. Η συνδυασμένη έκφραση δημιουργεί τιμές για μια συνάρτηση εγκλεισμού:

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

**γενική συνάρτηση** Μια συνάρτηση που αποτελείται από πολλαπλές συναρτήσεις που υλοποιούν την ίδια λειτουργία για διαφορετικούς τύπους. Ποια υλοποίηση πρέπει να χρησιμοποιηθεί κατά τη διάρκεια μια κλήσης καθορίζεται από τον αλγόριθμο αποστολής.

Βλ. επίσης την καταχώρηση του *single dispatch*, τον decorator `functools singledispatch()` και **PEP 443**.

**γενικός τύπος** Ένας *type* που μπορεί να παραμετροποιηθεί” συνήθως μια container class, όπως `list` ή `dict`. Χρησιμοποιείται για *type hints* και *annotations*.

Για περισσότερες λεπτομέρειες, βλ. generic alias types **PEP 483**, **PEP 484**, **PEP 585**, και το module `typing`.

**GIL** Βλ. *global interpreter lock*.

**global interpreter lock** Ο μηχανισμός που χρησιμοποιείται από τον διερμηνέα *CPython* για να διασφαλίσει ότι μόνο ένα νήμα εκτελεί Python *bytecode* κάθε φορά. Αυτό απλοποιεί την υλοποίηση *CPython* δημιουργώντας το μοντέλο αντικειμένου (συμπεριλαμβανομένων κρίσιμων ενσωματωμένων τύπων όπως π.χ. `dict`) έμμεσα ασφαλές έναντι ταυτόχρονης πρόσβασης. Το κλείδωμα ολόκληρου του διερμηνέα διευκολύνει τον διερμηνέα να είναι πολλαπλών νημάτων, εις βάρος του μεγάλου μέρους του παραλληλισμού που παρέχουν οι μηχανές πολλαπλών επεξεργαστών.

Ωστόσο, ορισμένες λειτουργικές μονάδες επέκτασης, είτε τυπικές είτε τρίτων, έχουν σχεδιαστεί έτσι ώστε να απελευθερώνουν το GIL όταν εκτελούν εργασίες εντατικών υπολογισμών όπως συμπίεση ή κατακερματισμός. Επίσης, το GIL απελευθερώνεται πάντα όταν εκτελείτε I/O.

Προηγούμενες προσπάθειες να δημιουργηθεί ένας διερμηνέας «ελεύθερων-νημάτων» (αυτός που κλειδώνει τα κοινόχρηστα δεδομένα με πολύ πιο λεπτομερή ευαισθησία) δεν ήταν επιτυχείς επειδή η απόδοση υποχώρησε στην κοινή περίπτωση ενός επεξεργαστή. Πιστεύεται ότι η υπέρβαση αυτού του προβλήματος απόδοσης θα κάνουν πολύ πιο περίπλοκη και επομένως πιο δαπανηρή στην συντήρηση.

**hash-based pyc** Ένα αρχείο κρυφής μνήμης *bytecode* που χρησιμοποιεί τον κατακερματισμό και όχι τον χρόνο τροποποίησης του αντίστοιχου αρχείου προέλευσης για να προσδιορίσει την εγκυρότητα του. Βλ. `pyc-invalidation`.

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

Η ύπαρξη *hashable* κάνει ένα αντικείμενο να μπορεί να χρησιμοποιηθεί ως κλειδί λεξικού και ως μέλος ενός συνόλου, επειδή αυτές οι δομές δεδομένων χρησιμοποιούν τιμές κατακερματισμού.

Τα περισσότερα από τα αμετάβλητα ενσωματωμένα αντικείμενα της Python μπορούν να κατακερματιστούν” τα μεταβλητά κοντέινερ (όπως οι λίστες ή τα λεξικά) δεν είναι” τα αμετάβλητα κοντέινερ (όπως πλειάδες και τα frozensets) μπορούν να κατακερματιστούν μόνο εάν τα στοιχεία τους είναι κατακερματισμένα. Τα αντικείμενα που είναι στιγμιότυπα κλάσεων που ορίζονται από το χρήστη μπορούν να κατακερματιστούν από προεπιλογή. Όλα συγκρίνονται άνισα εκτός από τον εαυτό τους) και η τιμή κατακερματισμού τους προέρχεται από το `id()`.

**IDLE** Ένα ολοκληρωμένο περιβάλλον ανάπτυξης και μάθησης για την Python. idle είναι ένα βασικό περιβάλλον επεξεργασίας και διερμηνέα που συνοδεύεται από την τυπική διανομή της Python.

**immutable** Ένα αντικείμενο με σταθερή τιμή. Τα αμετάβλητα αντικείμενα περιλαμβάνουν αριθμούς, συμβολοσειρές και πλειάδες. Ένα τέτοιο αντικείμενο δεν μπορεί να αλλάξει. Ένα νέο αντικείμενο πρέπει να δημιουργηθεί εάν πρέπει να αποθηκευτεί μια διαφορετική τιμή. Παίζουν σημαντικό ρόλο σε μέρη όπου μια σταθερά απαιτείται, για παράδειγμα ως κλειδί σε ένα λεξικό.

**εισαγόμενο path** Μια λίστα από τοποθεσίες (ή *καταχωρίσεις διαδρομής*) που μπορούν να αναζητηθούν *path based finder* για να εισαχθούν modules. Κατά την διαδικασία εισαγωγής, αυτή η λίστα με τοποθεσίες συνήθως έρχεται από `sys.path`, αλλά για τα υποπακέτα μπορεί επίσης να έρθει από το χαρακτηριστικό του πακέτου γονέα `__path__`.

**εισαγωγή** Η διαδικασία κατά την οποία ο κώδικας της Python σε ένα module είναι διαθέσιμη στον κώδικα Python ενός άλλου module.

**εισαγωγέας** Ένα αντικείμενο μπορεί και να αναζητεί και να φορτώνει ένα module” και ένα *finder* και *loader* αντικείμενο.

**διαδραστικός** Η Python έχει έναν διαδραστικό διερμηνέα όπου σημαίνει ότι μπορείς να εισάγεις δηλώσεις και εκφράσεις στην εισαγωγή εντολών του διερμηνέα, εκτελώντας τις άμεσα και εμφανίζοντας τα αντικείμενα. Απλώς εκκινήστε την python χωρίς ορίσματα (πιθανώς επιλέγοντας το από το κύριο μενού του υπολογιστή σας). Αποτελεί έναν αποδοτικό τρόπο για να δοκιμάστε νέες ιδέες ή να εξετάσετε λειτουργικές μονάδες και πακέτα (θυμηθείτε `help(x)`).

**interpreted** Η Python είναι μια interpreted γλώσσα, σε αντίθεση με μια μεταγλωττισμένη, αν και η διάκριση μπορεί να είναι και θολή λόγω της παρουσία του bytecode μεταγλωττιστή. Αυτό σημαίνει ότι τα αρχεία προέλευσης μπορούν να εκτελεστούν απευθείας χωρίς να δημιουργηθεί ρητά ένα εκτελέσιμο αρχείο που στην συνέχεια εκτελείται. Οι interpreted γλώσσες συνήθως έχουν μικρότερο κύκλο ανάπτυξης/εντοπισμού σφαλμάτων από τις μεταγλωττισμένες, αν και τα προγράμματά τους γενικά εκτελούνται πιο αργά. Βλ. επίσης *interactive*.

**τερματισμός λειτουργίας διερμηνέα** Όταν ζητείται τερματισμός λειτουργίας, ο διερμηνέας της Python εισέρχεται σε μια ειδική φάση όπου απελευθερώνει σταδιακά όλους τους διατιθέμενους πόρους, όπως λειτουργικές μονάδες και πολλαπλές κρίσιμες εσωτερικές δομές. Επίσης πραγματοποιεί αρκετές κλήσεις στο *συλλέκτης σκουπιδιών*. Αυτό μπορεί να ενεργοποιήσει την εκτέλεση κώδικα σε καταστροφείς που ορίζονται από το χρήστη ή σε callbacks ασθενούς ανταποκρίσεις. Ο κώδικας που εκτελείται κατά τη φάση τερματισμού λειτουργίας μπορεί να συναντήσει διάφορες εξαιρέσεις, καθώς οι πόροι στους οποίους βασίζεται ενδέχεται να μην λειτουργούν πλέον (συνήθη παραδείγματα είναι οι λειτουργικές μονάδες βιβλιοθήκης ή ο μηχανισμός ειδοποιήσεων).

Ο βασικός λόγος τερματισμού λειτουργίας του διερμηνέα είναι ότι το `__main__` module ή ολοκληρώθηκε η εκτέλεση του κώδικα που έτρεχε.

**iterable** An object capable of returning its members one at a time. Examples of iterables include all sequence types (such as `list`, `str`, and `tuple`) and some non-sequence types like `dict`, *file objects*, and objects of any classes you define with an `__iter__()` method or with a `__getitem__()` method that implements *Sequence* semantics.

Iterables can be used in a `for` loop and in many other places where a sequence is needed (`zip()`, `map()`, ...). When an iterable object is passed as an argument to the built-in function `iter()`, it returns an iterator for the object. This iterator is good for one pass over the set of values. When using iterables, it is usually not necessary to call `iter()` or deal with iterator objects yourself. The `for` statement does that automatically for you, creating a temporary unnamed variable to hold the iterator for the duration of the loop. See also *iterator*, *sequence*, and *generator*.

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

`iter()` function or use it in a `for` loop. Attempting this with an iterator will just return the same exhausted iterator object used in the previous iteration pass, making it appear like an empty container.

Περισσότερες πληροφορίες μπορούν να βρεθούν στο `typeiter`.

**Λεπτομέρεια υλοποίησης CPython:** CPython does not consistently apply the requirement that an iterator define `__iter__()`.

**συνάρτηση key** Μια συνάρτηση κλειδί ή μια συνάρτηση ταξινόμησης είναι μια δυνατότητα κλήσης που επιστρέφει μια τιμή που χρησιμοποιείται για ταξινόμηση ή διάταξη. Για παράδειγμα, `locale.strxfrm()` χρησιμοποιείται για την παραγωγή ενός κλειδιού ταξινόμησης που γνωρίζει τις συμβάσεις ταξινόμησης για συγκεκριμένες τοπικές ρυθμίσεις.

Ένα αριθμός εργαλείων στην Python δέχεται βασικές συναρτήσεις για τον έλεγχο του τρόπου με τον οποίο τα στοιχεία ταξινομούνται ή ομαδοποιούνται. Αυτά περιέχουν `min()`, `max()`, `sorted()`, `list.sort()`, `heapq.merge()`, `heapq.nsmallest()`, `heapq.nlargest()`, και `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, a 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.

**όρισμα keyword** Βλ. *argument*.

**lambda** Μια ανώνυμη ενσωματωμένη συνάρτηση που αποτελείται από μια μοναδική *expression* η οποία αξιολογείται όταν καλείται η συνάρτηση. Η σύνταξη για τη δημιουργία μιας συνάρτησης `lambda` είναι `lambda [parameters]: expression`

**LBYL** Look before you leap. Αυτό το στυλ κωδικοποίησης ελέγχει ρητά τις προϋποθέσεις πριν πραγματοποιήσει κλήσεις ή αναζητήσεις. Αυτό το στυλ έρχεται σε αντίθεση με την προσέγγιση *EAFP* και χαρακτηρίζεται από την παρουσία πολλών δηλώσεων `if`.

Σε ένα περιβάλλον πολλαπλών νημάτων, η προσέγγιση LBYL μπορεί να διακινδυνεύσει να εισάγει μια συνθήκη αγώνα μεταξύ «the Looking» και «the leaping». Για παράδειγμα ο κώδικας, `if key in mapping: return mapping[key]` μπορεί να αποτύχει εάν ένα άλλο νήμα αφαιρέσει το `key` από το `mapping` μετά τη δοκιμή, αλλά πριν από την αναζήτηση. Αυτό το πρόβλημα μπορεί να λυθεί με κλειδώματα ή χρησιμοποιώντας την προσέγγιση EAFP.

**τοπική κωδικοποίηση** On Unix, it is the encoding of the LC\_CTYPE locale. It can be set with `locale.setlocale(locale.LC_CTYPE, new_locale)`.

On Windows, it is the ANSI code page (ex: cp1252).

`locale.getpreferredencoding(False)` can be used to get the locale encoding.

Python uses the *filesystem encoding and error handler* to convert between Unicode filenames and bytes filenames.

**λίστα** 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** Ένα συμπαγής τρόπος για να επεξεργαστείτε όλα ή μέρος των στοιχείων σε μια ακολουθία και να επιστρέψετε μια λίστα με τα αποτελέσματα. `result = ['{: #04x}'.format(x) for x in range(256) if x % 2 == 0]` δημιουργεί μια λίστα συμβολοσειρών που περιέχουν ζυγούς δεκαεξαδικούς αριθμούς (0x..) στο εύρος από 0 έως 255. Η πρόταση `if` είναι προαιρετική. Εάν παραλειφθεί, όλα τα στοιχεία στο `range(256)` υποβάλλονται σε επεξεργασία.

**loader** An object that loads a module. It must define a method named `load_module()`. A loader is typically returned by a *finder*. See **PEP 302** for details and `importlib.abc.Loader` for an *abstract base class*.

**μαγική μέθοδος** Ένα άτυπο συνώνυμο για *special method*.

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



**meta path finder** Ένας *finder* που επιστράφηκε με αναζήτηση στο `sys.meta_path`. Οι *finders* μετα-διαδρομής σχετίζονται, αλλά διαφέρουν από τα *finders entry διαδρομής*.

Βλ. `importlib.abc.MetaPathFinder` για τις μεθόδους που υλοποιούν οι meta path finders.

**μετα-κλάση** Η κλάση μιας κλάσης. Οι ορισμοί κλάσης δημιουργούν ένα όνομα κλάσης, ένα λεξικό κλάσης και μια λίστα βασικών κλάσεων. Η μετα-κλάση είναι υπεύθυνη για την απόκτηση αυτών των τριών ορισμάτων και την δημιουργία της κλάσης. Οι περισσότερες αντικειμενοστρεφείς γλώσσες προγραμματισμού παρέχουν μια προεπιλεγμένη υλοποίηση. Αυτό που κάνει την Python ξεχωριστή είναι ότι είναι δυνατή η δημιουργία προσαρμοσμένων μετακλάσεων. Οι περισσότεροι χρήστες δεν χρειάζονται ποτέ αυτό το εργαλείο, αλλά όταν παραστεί ανάγκη, αυτό το εργαλείο, οι μετα-κλάσεις μπορούν να παρέχουν ισχυρές, κομψές λύσεις. Έχουν χρησιμοποιηθεί για την καταγραφή πρόσβασης χαρακτηριστικών, την προσθήκη ασφάλειας νημάτων, την παρακολούθηση δημιουργίας αντικειμένων, την υλοποίηση *singletons*, και πολλές άλλες εργασίες.

Περισσότερες πληροφορίες μπορούν να βρεθούν στο *metaclasses*.

**μέθοδος** Μια συνάρτηση που ορίζεται μέσα στο σώμα μιας κλάσης. Εάν καλείται ως χαρακτηριστικό μιας περίπτωσης αυτής της κλάσης, η μέθοδος θα λάβει αντικείμενο περίπτωσης ως πρώτο της *argument* (το οποίο συνήθως ονομάζεται `self`). Βλ. *function* και *nested scope*.

**σειρά ανάλυσης μεθόδων** Method Resolution Order is the order in which base classes are searched for a member during lookup. See [The Python 2.3 Method Resolution Order](#) for details of the algorithm used by the Python interpreter since the 2.3 release.

**module** Ένα αντικείμενο που χρησιμεύει ως οργανωτική μονάδα του κώδικα της Python. Τα modules έχουν έναν χώρο ονομάτων που περιέχει αυθαίρετα αντικείμενα Python. Τα modules φορτώνονται στην Python με την διαδικασία *importing*.

Βλ. επίσης *package*.

**τεχνικές προδιαγραφές module** Ένα namespace που περιέχει τις πληροφορίες που σχετίζονται με την εισαγωγή που χρησιμοποιούνται για την φόρτωση ενός module. Μια περίπτωση του `importlib.machinery.ModuleSpec`.

**MRO** Βλ. *method resolution order*.

**mutable** Τα ευμετάβλητα αντικείμενα μπορούν να αλλάξουν τις τιμές αλλά να κρατήσουν τα `id()`. Βλ. επίσης *immutable*.

**named tuple** Ο όρος «named tuple» εφαρμόζεται για οποιονδήποτε τύπο ή κλάση που κληρονομείται από την πλειάδα και των οποίων τα στοιχεία μπορούν να ευρετηριοποιηθούν είναι προσβάσιμα χρησιμοποιώντας επώνυμα χαρακτηριστικά. Ο τύπος ή η κλάση μπορεί να έχει και άλλα χαρακτηριστικά.

Πολλοί ενσωματωμένοι τύποι είναι named tuples, συμπεριλαμβανομένων των τιμών που επιστρέφονται από `time.localtime()` και `os.stat()`. Ένα άλλο παράδειγμα είναι το `sys.float_info`:

```
>>> sys.float_info[1]           # indexed access
1024
>>> sys.float_info.max_exp      # named field access
1024
>>> isinstance(sys.float_info, tuple) # kind of tuple
True
```

Some named tuples are built-in types (such as the above examples). Alternatively, a named tuple can be created from a regular class definition that inherits from `tuple` and that defines named fields. Such a class can be written by hand or it can be created with the factory function `collections.namedtuple()`. The latter technique also adds some extra methods that may not be found in hand-written or built-in named tuples.

**namespace** Το μέρος όπου αποθηκεύεται μια μεταβλητή. Τα namespaces υλοποιούνται ως λεξικά. Υπάρχουν οι τοπικοί, οι καθολικοί και οι ενσωματωμένοι namespaces καθώς και οι ένθετοι namespaces σε αντικείμενα (σε μεθόδους). Για παράδειγμα οι συναρτήσεις `builtins.open` και `os.open()` διακρίνονται από τους χώρους ονομάτων τους. Οι χώροι ονομάτων βοηθούν επίσης την αναγνωσιμότητα και τη συντηρησιμότητα καθιστώντας σαφές ποιο module υλοποιεί μια λειτουργία. Για παράδειγμα, γράφοντας

`random.seed()` ή `itertools.islice()` καθιστά σαφές ότι αυτές οι συναρτήσεις υλοποιούνται από τα module `random` και `itertools`, αντίστοιχα.

**πακέτο namespace** A [PEP 420 package](#) which serves only as a container for subpackages. Namespace packages may have no physical representation, and specifically are not like a [regular package](#) because they have no `__init__.py` file.

Βλ. επίσης [module](#).

**nested scope** Η δυνατότητα αναφοράς σε μια μεταβλητή σε έναν περικλειόμενο ορισμό. Για παράδειγμα μια συνάρτηση που ορίζεται μέσα σε μια άλλη συνάρτηση μπορεί να αναφέρεται σε μεταβλητές στην εξωτερική συνάρτηση. Σημειώστε ότι τα ένθετα πεδία από προεπιλογή λειτουργούν μόνο για αναφορά και όχι για εκχώρηση. Οι τοπικές μεταβλητές διαβάζονται και γράφονται στο εσωτερικό πεδίο εφαρμογής. Ομοίως, οι καθολικές μεταβλητές διαβάζουν και γράφουν στον καθολικό χώρο ονομάτων. Το `nonlocal` επιτρέπει την εγγραφή σε εξωτερικά πεδία.

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

**αντικείμενο** Οποιαδήποτε δεδομένα με κατάσταση (χαρακτηριστικά ή τιμή) και καθορισμένη συμπεριφορά (μέθοδοι). Επίσης, η τελική βασική κλάση οποιασδήποτε [new-style class](#).

**πακέτο** Ένα Python [module](#) που μπορεί να περιέχει submodules ή αναδρομικά, υποπακέτα. Τεχνικά, ένα πακέτο είναι μια λειτουργική μονάδα Python με ένα `__path__` χαρακτηριστικό.

Βλ. επίσης [regular package](#) και [namespace package](#).

**παράμετρος** Μια έγκυρη οντότητα σε έναν ορισμό [function](#) (ή μέθοδος) που καθορίζει ένα [argument](#) (ή σε ορισμένες περιπτώσεις, ορίσματα) που μπορεί να δεχθεί η συνάρτηση. Υπάρχουν πέντε είδη παραμέτρων:

- **λέξη-κλειδί ή θέση:** καθορίζει ένα όρισμα που μπορεί να μεταβιβαστεί είτε *θέσεως* ή ως *όρισμα λέξης-κλειδιού*. Αυτό είναι το προεπιλεγμένο είδος παραμέτρου, για παράδειγμα `foo` και `bar` στα ακόλουθα:

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

- **θέσεως μόνο:** καθορίζει ένα όρισμα που μπορεί να παρέχεται μόνο από τη θέση. Οι παράμετροι μόνο θέσης μπορούν να οριστούν συμπεριλαμβάνοντας έναν χαρακτήρα / στη λίστα παραμέτρων του ορισμού συνάρτησης μετά από αυτές, για παράδειγμα `posonly1` και `posonly2` στα εξής:

```
def func(posonly1, posonly2, /, positional_or_keyword): ...
```

- **λέξης-κλειδί μόνο:** καθορίζει ένα όρισμα που μπορεί να παρέχεται μόνο με λέξη κλειδί. Οι παράμετροι μόνο για λέξη-κλειδί μπορούν να οριστούν συμπεριλαμβάνοντας μια παράμετρο θέσης ή σκέτο `*` στη λίστα παραμέτρων του ορισμού συνάρτησης πριν από αυτές, για παράδειγμα `kw_only1` και `kw_only2` στα ακόλουθα:

```
def func(arg, *, kw_only1, kw_only2): ...
```

- **μεταβλητή θέσης:** καθορίζει ότι μπορεί να παρασχεθεί μια αυθαίρετη ακολουθία ορισμάτων θέσης (επιπλέον των ορισμάτων θέσης που είναι ήδη αποδεκτά από άλλες παραμέτρους). Μια τέτοια παράμετρος μπορεί να οριστεί προσαρτώντας το όνομα της παραμέτρου με `*`, για παράδειγμα `args` στα ακόλουθα:

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

- **μεταβλητή λέξη-κλειδί:** καθορίζει ότι μπορούν να παρέχονται αυθαίρετα πολλά ορίσματα λέξης-κλειδιού (επιπλέον των ορισμάτων λέξης κλειδιού που είναι αποδεκτά από άλλες παραμέτρους). Μια τέτοια παράμετρος μπορεί να οριστεί προσαρτώντας το όνομα της παραμέτρου με `**`, για παράδειγμα `kwargs` όπως παραπάνω.

Οι παράμετροι μπορούν να καθορίσουν τόσο τα προαιρετικά όσο και τα απαιτούμενα ορίσματα, καθώς και προεπιλεγμένες τιμές για ορισμένα προαιρετικά ορίσματα.

Βλ. επίσης την [argument](#) καταχώριση ευρετηρίου, την ερώτηση FAQ σχετικά με τη διαφορά μεταξύ ορισμάτων και παραμέτρων, την κλάση `inspect.Parameter`, την ενότητα `function` και [PEP 362](#).

**path entry** Μια μεμονωμένη τοποθεσία στο `import path` την οποία συμβουλεύεται ο `path based finder` για να βρει modules για εισαγωγή.

**path entry finder** Ένας `finder` που επιστρέφεται από έναν καλούμενο στο `sys.path_hooks` (δηλαδή ένα `path entry hook`) που ξέρει πως να εντοπίζει modules με `path entry`.

Βλ. `importlib.abc.PathEntryFinder` για τις μεθόδους που ο `entry finder` διαδρομής υλοποιεί.

**path entry hook** A callable on the `sys.path_hook` list which returns a `path entry finder` if it knows how to find modules on a specific `path entry`.

**path based finder** Ένα από τα προεπιλεγμένα `meta path finders` που αναζητά ένα `import path` για modules.

**path-like αντικείμενο** Ένα αντικείμενο που αντιπροσωπεύει ένα `path` συστήματος αρχείων. Ένα αντικείμενο `path` είναι είτε ένα αντικείμενο `str` ή `bytes` που αντιπροσωπεύει ένα `path` ή ένα αντικείμενο που υλοποιεί το πρωτόκολλο `os.PathLike`. Ένα αντικείμενο που υποστηρίζει το πρωτόκολλο `os.PathLike` μπορεί να μετατραπεί σε `path` συστήματος αρχείων `str` ή `bytes` καλώντας την συνάρτηση `os.fspath()` τα `os.fsdecode()` και `os.fsencode()` μπορούν να χρησιμοποιηθούν για την εγγύηση ενός αποτελέσματος `str` ή `bytes`, αντίστοιχα. Εισήχθη από τον [PEP 519](#).

**PEP** Πρόταση Βελτίωσης Python. Ένα PEP είναι ένα έγγραφο σχεδιασμού που παρέχει πληροφορίες στην κοινότητα Python ή περιγράφει μια νέα δυνατότητα για την Python ή τις διαδικασίες ή το περιβάλλον της. Τα PEP θα πρέπει να παρέχουν μια συνοπτική τεχνική προδιαγραφή και μια λογική για τα προτεινόμενα χαρακτηριστικά.

Τα PEP προορίζονται να είναι οι κύριοι μηχανισμοί για την πρόταση σημαντικών νέων χαρακτηριστικών, για τη συλλογή πληροφοριών της κοινότητας για ένα ζήτημα και για την τεκμηρίωση των αποφάσεων σχεδιασμού που έχουν εισαχθεί στην Python. Ο συγγραφέας του PEP είναι υπεύθυνος για την οικοδόμηση συναίνεσης εντός της κοινότητας και την τεκμηρίωση αντίθετων απόψεων.

Βλ. [PEP 1](#).

**τιμήμα** Ένα σύνολο από αρχεία σε έναν μόνο κατάλογο (ενδεχομένως αποθηκευμένο σε αρχείο `zip`) που συμβάλλουν σε ένα namespace πακέτο, όπως ορίζεται στο [PEP 420](#).

**όρισμα θέσης** Βλ. [argument](#).

**provisional API** Ένα provisional API είναι αυτό που έχει εσκεμμένα εξαιρεθεί από τις backwards εγγυήσεις συμβατότητας της τυπικής βιβλιοθήκης. Αν και δεν αναμένονται σημαντικές αλλαγές σε τέτοιες διαπαφές, εφόσον επισημαίνονται ως προσωρινές, αλλαγές μη backwards συμβατότητας (μέχρι και κατάργηση της διαπαφής) μπορεί να προκύψουν εάν κριθεί απαραίτητο από τους βασικούς προγραμματιστές. Τέτοιες αλλαγές δεν θα γίνουν άσκοπα – θα συμβούν μόνο εάν αποκαλυφθούν σοβαρά θεμελιώδη ελαττώματα που παραλείφθηκαν πριν από τη συμπερίληψη του API.

Ακόμη και για provisional API, οι μη backwards συμβατές αλλαγές θεωρούνται «λύση έσχατης ανάγκης»- θα εξακολουθεί να γίνεται κάθε προσπάθεια για να βρεθεί μια λύση backwards συμβατή σε τυχόν εντοπισμένα προβλήματα.

Αυτή η διαδικασία επιτρέπει στην τυπική βιβλιοθήκη να συνεχίσει να εξελίσσεται με την πάροδο του χρόνου, χωρίς να κλειδώνει προβληματικά σφάλματα σχεδιασμού για εκτεταμένες χρονικές περιόδους. Βλ. [PEP 411](#) για περισσότερες λεπτομέρειες.

**provisional πακέτο** Βλ. [provisional API](#).

**Python 3000** Ψευδώνυμο για το σύνολο εκδόσεων Python 3.x (επινοήθηκε πριν από πολύ καιρό όταν η κυκλοφορία της έκδοσης 3 ήταν κάτι στο μακρινό μέλλον.) Αυτό ονομάζεται επίσης ως συντομογραφία «Py3k».

**Pythonic** Μια ιδέα ή ένα κομμάτι κώδικα που ακολουθεί πιστά τα πιο κοινά ιδιώματα της γλώσσας Python, αντί να υλοποιεί κώδικα χρησιμοποιώντας έννοιες κοινές σε άλλες γλώσσες. Για παράδειγμα, ένα κοινό



ιδίωμα στην Python είναι να κάνει μια επανάληψη πάνω από όλα τα στοιχεία ενός iterable χρησιμοποιώντας μια δήλωση `for`. Πολλές άλλες γλώσσες που δεν έχουν αυτόν τον τύπο κατασκευής, έτσι οι άνθρωποι που δεν είναι εξοικειωμένοι με την Python χρησιμοποιούν μερικές φορές έναν αριθμητικό μετρητή:

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

Αντίθετα, μια πιο καθαρή μέθοδος Pythonic:

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

**αναγνωρισμένο όνομα** Ένα όνομα με κουκκίδες που δείχνει τη «διαδρομή» από το καθολικό εύρος ενός module σε μια κλάση, συνάρτηση ή μέθοδο που ορίζεται σε αυτήν την ενότητα, όπως ορίζεται στο **PEP 3155**. Για συναρτήσεις και κλάσεις ανώτατου επιπέδου, το αναγνωρισμένο όνομα είναι ίδιο με το όνομα του αντικειμένου:

```
>>> class C:
...     class D:
...         def meth(self):
...             pass
...
>>> C.__qualname__
'C'
>>> C.D.__qualname__
'C.D'
>>> C.D.meth.__qualname__
'C.D.meth'
```

Όταν χρησιμοποιείται για αναφορά σε modules, το *πλήρως αναγνωρισμένο όνομα* σημαίνει ολόκληρο το διακεκομμένο path προς το module, συμπεριλαμβανομένων τυχόν γονικών πακέτων π.χ. `email.mime.text`:

```
>>> import email.mime.text
>>> email.mime.text.__name__
'email.mime.text'
```

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

**κανονικό πακέτο** Ένα παραδοσιακό *package*, όπως ένας κατάλογος που περιέχει ένα `__init__.py` αρχείο.

Βλ. επίσης *namespace package*.

**\_\_slots\_\_** Μια δήλωση μέσα σε μια κλάση που εξοικονομεί μνήμη δηλώνοντας εκ των προτέρων χώρο για παράδειγμα χαρακτηριστικά και εξαλείφοντας λεξικά στιγμιότυπων. Αν και δημοφιλής, η τεχνική είναι κάπως δύσκολο να γίνει σωστή και προορίζεται καλύτερα για σπάνιες περιπτώσεις όπου υπάρχει μεγάλος αριθμός στιγμιότυπων σε μια εφαρμογή κρίσιμης-μνήμης.

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

The `collections.abc.Sequence` abstract base class defines a much richer interface that goes beyond just `__getitem__()` and `__len__()`, adding `count()`, `index()`, `__contains__()`, and `__reversed__()`. Types that implement this expanded interface can be registered explicitly using `register()`.

**set comprehension** Ένας συμπαγής τρόπος για να επεξεργαστείτε όλα ή μέρος των στοιχείων σε ένα iterable και να επιστραφεί ένα σύνολο με τα αποτελέσματα. `results = {c for c in 'abracadabra' if c not in 'abc'}` δημιουργεί το σύνολο συμβολοσειρών {'r', 'd'}. Βλ. [comprehensions](#).

**μοναδικό dispatch** Μια μορφή dispatch [generic function](#) όπου η υλοποίηση επιλέγεται με βάση τον τύπο ενός μεμονωμένου ορίσματος.

**slice** Ένα αντικείμενο που συνήθως περιέχει ένα τμήμα μιας ακολουθίας [sequence](#). Δημιουργείται ένα slice χρησιμοποιώντας τη σημείωση subscript, `[]` με άνω και κάτω τελείες μεταξύ αριθμών όταν δίνονται πολλοί, όπως στο `variable_name[1:3:5]`. Η σημείωση αγκύλης (subscript) χρησιμοποιεί εσωτερικά αντικείμενα slice.

**ειδική μέθοδος** Μια μέθοδος που καλείται σιωπηρά από την Python για να εκτελέσει μια συγκεκριμένη λειτουργία σε έναν τύπο, όπως η προσθήκη. Τέτοιες μέθοδοι έχουν ονόματα που ξεκινούν και τελειώνουν με διπλές κάτω παύλες. Οι ειδικές μέθοδοι τεκμηριώνονται στο `specialnames`.

**δήλωση** Μια πρόταση είναι μέρος μιας σουίτας (ένα «μπλοκ» κώδικα). Μια πρόταση είναι είτε ένας [expression](#) είτε μια από πολλές δομές με μια λέξη-κλειδί όπως `if`, `while` ή `for`.

**strong reference** Στο C API της Python, μια ισχυρή αναφορά είναι μια αναφορά σε ένα αντικείμενο που ανήκει στον κώδικα που περιέχει την αναφορά. Η ισχυρή αναφορά λαμβάνεται καλώντας το `Py_INCREF()` όταν η αναφορά δημιουργείται και απελευθερώνεται με `Py_DECREF()` όταν διαγραφεί η αναφορά.

Η συνάρτηση `Py_NewRef()` μπορεί να χρησιμοποιηθεί για τη δημιουργία ισχυρής αναφοράς σε ένα αντικείμενο. Συνήθως, η συνάρτηση `Py_DECREF()` πρέπει να καλείται στην ισχυρή αναφορά πριν βγει από το εύρος της ισχυρής αναφοράς, για να αποφευχθεί η διαρροή μιας αναφοράς.

Βλ. επίσης [borrowed reference](#).

**κωδικοποίηση κειμένου** Μια συμβολοσειρά στην Python είναι μια ακολουθία σημείων κώδικα Unicode (στο εύρος U+0000–U+10FFFF). Για να αποθηκεύσετε ή να μεταφέρετε μια συμβολοσειρά, πρέπει να σειριοποιηθεί ως δυαδική ακολουθία.

Η σειριοποίηση μιας συμβολοσειράς σε μια δυαδική ακολουθία είναι γνωστή ως «κωδικοποίηση», και η αναδημιουργία της συμβολοσειράς από την δυαδική ακολουθία είναι γνωστή ως «αποκωδικοποίηση».

Υπάρχει μια ποικιλία διαφορετικής σειριοποίησης κειμένου codecs, οι οποίοι συλλογικά αναφέρονται ως «κωδικοποιήσεις κειμένου».

**αρχείο κειμένου** Ένα [file object](#) ικανό να διαβάζει και να γράφει αντικείμενα `str`. Συχνά, ένα αρχείο κειμένου αποκτά πραγματικά πρόσβαση σε μια ροή δυαδική ροή δεδομένων και χειρίζεται αυτόματα την [text encoding](#). Παραδείγματα αρχείων κειμένου είναι αρχεία που ανοίγουν σε λειτουργία κειμένου ('r' ή 'w'), `sys.stdin`, `sys.stdout`, και στιγμιότυπα του `io.StringIO`.

Βλ. επίσης [binary file](#) για ένα αντικείμενο αρχείου με δυνατότητα ανάγνωσης και εγγραφής [δυαδικά αντικείμενα](#).

**συμβολοσειρά τριπλών εισαγωγικών** Μια συμβολοσειρά που δεσμεύεται από τρεις περιπτώσεις είτε ενός εισαγωγικού («») ή μιας αποστρόφου ("). Αν και δεν παρέχουν καμία λειτουργικότητα που δεν είναι διαθέσιμη με συμβολοσειρές με μονά εισαγωγικά, είναι χρήσιμες για διαφόρους λόγους. Σας επιτρέπουν να συμπεριλάβετε μονά και διπλά εισαγωγικά χωρίς διαφυγή σε μια συμβολοσειρά και μπορούν να εκτείνονται σε πολλές γραμμές χωρίς τη χρήση του χαρακτήρα συνέχεια, καθιστώντας τα ιδιαίτερα χρήσιμα κατά τη σύνταξη εγγράφων με συμβολοσειρές.

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

**type alias** Ένα συνώνυμο για έναν τύπο, που δημιουργείται με την ανάθεση τύπου σε ένα αναγνωριστικό.

Τα type aliases είναι χρήσιμα για την απλοποίηση [type alias](#). Για παράδειγμα:

```
def remove_gray_shades(
    colors: list[tuple[int, int, int]]) -> list[tuple[int, int, int]]:
    pass
```

μπορεί να γίνει πιο ευανάγνωστο όπως:

```
Color = tuple[int, int, int]

def remove_gray_shades(colors: list[Color]) -> list[Color]:
    pass
```

Βλ. `typing` και **PEP 484**, που περιγράφει αυτήν την λειτουργικότητα.

**type hint** Ένας *annotation* που καθορίζει τον αναμενόμενο τύπο για μια μεταβλητή, ένα χαρακτηριστικό κλάσης ή μια παράμετρο συνάρτησης ή τιμή επιστροφής.

Type hints are optional and are not enforced by Python but they are useful to static type analysis tools, and aid IDEs with code completion and refactoring.

Υποδείξεις τύπου (type hints) για καθολικές μεταβλητές, χαρακτηριστικά κλάσης και συναρτήσεις, αλλά όχι τοπικές μεταβλητές, μπορούν να προσπελαστούν χρησιμοποιώντας το `typing.get_type_hints()`.

Βλ. `typing` και **PEP 484**, που περιγράφει αυτήν την λειτουργικότητα.

**καθολικές νέες γραμμές** Ένα τρόπος ερμηνείας ροών κειμένου στον οποίο όλα τα ακόλουθα αναγνωρίζονται ως λήξεις μιας γραμμής: η σύμβαση τέλους γραμμής του Unix `'\n'`, η σύμβαση των Windows `'\r\n'`, και την παλιά σύμβαση Macintosh `'\r'`. Βλ. **PEP 278** και **PEP 3116**, καθώς και `bytes.splitlines()` για πρόσθετη χρήση.

**annotation μεταβλητής** Ένας *annotation* μια μεταβλητής ή ενός χαρακτηριστικού κλάσης.

Όταν annotating μια μεταβλητή ή ένα χαρακτηριστικό κλάσης, η ανάθεση είναι προαιρετική:

```
class C:
    field: 'annotation'
```

Τα annotations μεταβλητών χρησιμοποιούνται συνήθως για *type hints*: για παράδειγμα αυτή η μεταβλητή αναμένεται να λάβει τιμές `int`:

```
count: int = 0
```

Η σύνταξη annotation μεταβλητής περιγράφεται στην ενότητα `annassign`.

Βλ. *function annotation*, **PEP 484** και **PEP 526**, που περιγράφουν αυτή τη λειτουργία. Δείτε επίσης `annotations-howto` για βέλτιστες πρακτικές σχετικά με την εργασία με σχολιασμούς.

**virtual environment** Ένα συνεργατικά απομονωμένο περιβάλλον χρόνου εκτέλεσης που επιτρέπει στους χρήστες και τις εφαρμογές της Python να εγκαταστήσουν και να αναβαθμίσουν πακέτα διανομής Python χωρίς να παρεμβαίνουν στη συμπεριφορά άλλων εφαρμογών Python που εκτελούνται στο ίδιο σύστημα.

Βλ. επίσης `venv`.

**virtual machine** Ένας υπολογιστής ορίζεται εξ ολοκλήρου από το λογισμικό. Η εικονική μηχανή της Python εκτελεί το *bytecode* που εκπέμπεται από τον μεταγλωττιστή `bytecode`.

**Zen της Python** Κατάλογος σχεδιαστικών αρχών και φιλοσοφιών που είναι χρήσιμες για την κατανόηση και τη χρήση της γλώσσας. Ο κατάλογος μπορεί να βρεθεί πληκτρολογώντας «`import this`» στην διαδραστική κονσόλα.



## ΠΑΡΑΡΤΗΜΑ Β'

---

### About these documents

---

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

Η ανάπτυξη των εγγράφων και των εργαλείων τους είναι εξ΄ ολοκλήρου εθελοντική προσπάθεια, όπως και η ίδια η Python. Εάν θέλετε να συνεισφέρετε, ρίξτε μια ματιά στη σελίδα [reporting-bugs](#) για πληροφορίες σχετικές με το πως να το κάνετε. Καινούριοι εθελοντές είναι πάντα ευπρόσδεκτοι!

Πολλές ευχαριστίες πηγαίνουν στους:

- Fred L. Drake, Jr., the creator of the original Python documentation toolset and writer of much of the content;
- το [Docutils](#) πρότζεκτ για την δημιουργία των εφαρμογών reStructuredText και Docutils·
- Fredrik Lundh για το δικό του Alternative Python Reference πρότζεκτ από το οποίο το Sphinx πήρε πολύ καλές ιδέες.

### B'.1 Contributors to the Python Documentation

Πολλοί άνθρωποι έχουν συνεισφέρει στη γλώσσα Python, την βιβλιοθήκη της Python, και τα έγγραφα της Python. Δείτε [Misc/ACKS](#) στις πηγές διανομής της Python για μια λίστα των συντελεστών.

Μόνο με τη συμβολή και τις συνεισφορές της κοινότητας της Python, η Python έχει τέτοια υπέροχα έγγραφα – Σας ευχαριστούμε!



## Γ'.1 Η ιστορία του λογισμικού

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.

Έκδοση	Προερχόμενη από	Έτος	Ιδιοκτησία	GPL compatible?
0.9.0 έως 1.2	δ/ν	1991-1995	CWI	ναι
1.3 έως 1.5.2	1.2	1995-1999	CNRI	ναι
1.6	1.5.2	2000	CNRI	όχι
2.0	1.6	2000	BeOpen.com	όχι
1.6.1	1.6	2001	CNRI	όχι
2.1	2.0+1.6.1	2001	PSF	όχι
2.0.1	2.0+1.6.1	2001	PSF	ναι
2.1.1	2.1+2.0.1	2001	PSF	ναι
2.1.2	2.1.1	2002	PSF	ναι
2.1.3	2.1.2	2002	PSF	ναι
2.2 και πάνω	2.1.1	2001-σήμερα	PSF	ναι

**Σημείωση:** 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.

Χάρη, στους πολλούς εξωτερικούς εθελοντές που εργάστηκαν κάτω από τις οδηγίες του Guido, αυτές οι εκδόσεις έγιναν εφικτές.

## Γ'.2 Όροι και προϋποθέσεις για την πρόσβαση ή την χρήση της Python με άλλους τρόπους

Python software and documentation are licensed under the *PSF License Agreement*.

Starting with Python 3.8.6, examples, recipes, and other code in the documentation are dual licensed under the PSF License Agreement and the *Zero-Clause BSD license*.

Κάποιο λογισμικό που είναι ενσωματωμένο στην Python είναι υπό διαφορετικές άδειες χρήσης. Οι άδειες παρατίθενται με κώδικα που εμπίπτει σε αυτήν την άδεια. Δείτε *Άδειες και Ευχαριστίες για Ενσωματωμένο Λογισμικό* για μια ελλιπή λίστα αυτών των αδειών.

### Γ'.2.1 PSF LICENSE AGREEMENT FOR PYTHON 3.10.18

1. This LICENSE AGREEMENT is between the Python Software Foundation,  
→ ("PSF"), and  
the Individual or Organization ("Licensee") accessing and otherwise  
→ using Python  
3.10.18 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 3.10.18 alone or in any derivative  
version, provided, however, that PSF's License Agreement and PSF's  
→ notice of  
copyright, i.e., "Copyright © 2001-2023 Python Software Foundation; All  
→ Rights  
Reserved" are retained in Python 3.10.18 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 3.10.18 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  
3.10.18.
4. PSF is making Python 3.10.18 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 3.10.18 WILL NOT INFRINGE ANY THIRD PARTY RIGHTS.



5. PSF SHALL NOT BE LIABLE TO LICENSEE OR ANY OTHER USERS OF PYTHON 3.10.18 FOR ANY INCIDENTAL, SPECIAL, OR CONSEQUENTIAL DAMAGES OR LOSS AS A RESULT OF MODIFYING, DISTRIBUTING, OR OTHERWISE USING PYTHON 3.10.18, 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 3.10.18, Licensee agrees to be bound by the terms and conditions of this License Agreement.

## Γ'.2.2 ΣΥΜΦΩΝΙΑ ΑΔΕΙΑΣ BEOPEN.COM ΓΙΑ PYTHON 2.0

### ΣΥΜΦΩΝΙΑ ΑΔΕΙΑΣ ΑΝΟΙΧΤΟΥ ΚΩΔΙΚΑ BEOPEN PYTHON ΕΚΔΟΣΗ 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

(συνέχεια στην επόμενη σελίδα)

(συνεχίζεται από την προηγούμενη σελίδα)

Agreement does not grant permission to use BeOpen trademarks or trade names in a trademark sense to endorse or promote products or services of Licensee, or any third party. As an exception, the "BeOpen Python" logos available at <http://www.pythonlabs.com/logos.html> may be used according to the permissions granted on that web page.

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

### Γ'.2.3 ΣΥΜΦΩΝΙΑ ΑΔΕΙΑΣ CNRI ΓΙΑ PYTHON 1.6.1

1. This LICENSE AGREEMENT is between the Corporation for National Research Initiatives, having an office at 1895 Preston White Drive, Reston, VA 20191 ("CNRI"), and the Individual or Organization ("Licensee") accessing and otherwise using Python 1.6.1 software in source or binary form and its associated documentation.
2. Subject to the terms and conditions of this License Agreement, CNRI hereby grants Licensee a nonexclusive, royalty-free, world-wide license to reproduce, analyze, test, perform and/or display publicly, prepare derivative works, distribute, and otherwise use Python 1.6.1 alone or in any derivative version, provided, however, that CNRI's License Agreement and CNRI's notice of copyright, i.e., "Copyright © 1995-2001 Corporation for National Research Initiatives; All Rights Reserved" are retained in Python 1.6.1 alone or in any derivative version prepared by Licensee. Alternately, in lieu of CNRI's License Agreement, Licensee may substitute the following text (omitting the quotes): "Python 1.6.1 is made available subject to the terms and conditions in CNRI's License Agreement. This Agreement together with Python 1.6.1 may be located on the internet using the following unique, persistent identifier (known as a handle): 1895.22/1013. This Agreement may also be obtained from a proxy server on the internet using the following URL: <http://hdl.handle.net/1895.22/1013>."
3. In the event Licensee prepares a derivative work that is based on or incorporates Python 1.6.1 or any part thereof, and wants to make the derivative work available to others as provided herein, then Licensee hereby agrees to include in any such work a brief summary of the changes made to Python 1.6.1.
4. CNRI is making Python 1.6.1 available to Licensee on an "AS IS" basis. CNRI MAKES NO REPRESENTATIONS OR WARRANTIES, EXPRESS OR IMPLIED. BY WAY OF EXAMPLE, BUT NOT LIMITATION, CNRI MAKES NO AND DISCLAIMS ANY REPRESENTATION OR WARRANTY OF MERCHANTABILITY OR FITNESS FOR ANY PARTICULAR PURPOSE OR THAT THE USE OF PYTHON 1.6.1 WILL NOT INFRINGE ANY THIRD PARTY RIGHTS.
5. CNRI SHALL NOT BE LIABLE TO LICENSEE OR ANY OTHER USERS OF PYTHON 1.6.1 FOR ANY INCIDENTAL, SPECIAL, OR CONSEQUENTIAL DAMAGES OR LOSS AS A RESULT OF MODIFYING, DISTRIBUTING, OR OTHERWISE USING PYTHON 1.6.1, OR ANY DERIVATIVE THEREOF, EVEN IF ADVISED OF THE POSSIBILITY THEREOF.
6. This License Agreement will automatically terminate upon a material breach of its terms and conditions.
7. This License Agreement shall be governed by the federal intellectual property law of the United States, including without limitation the federal copyright law, and, to the extent such U.S. federal law does not apply, by the law of the Commonwealth of Virginia, excluding Virginia's conflict of law provisions. Notwithstanding the foregoing, with regard to derivative works based on Python 1.6.1 that incorporate non-separable material that was previously distributed under the GNU General Public License (GPL), the law of the Commonwealth of Virginia shall govern this License Agreement only as to issues arising under or

(συνέχεια στην επόμενη σελίδα)

(συνεχίζεται από την προηγούμενη σελίδα)

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.

## Γ'.2.4 ΣΥΜΦΩΝΙΑ ΑΔΕΙΑΣ CWI ΓΙΑ PYTHON 0.9.0 ΕΩΣ 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.

## Γ'.2.5 ZERO-CLAUSE BSD LICENSE FOR CODE IN THE PYTHON 3.10.18 DOCUMENTATION

Permission to use, copy, modify, and/or distribute this software for any purpose with or without fee is hereby granted.

THE SOFTWARE IS PROVIDED "AS IS" AND THE AUTHOR DISCLAIMS ALL WARRANTIES WITH REGARD TO THIS SOFTWARE INCLUDING ALL IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS. IN NO EVENT SHALL THE AUTHOR BE LIABLE FOR ANY SPECIAL, DIRECT, 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.

## Γ'.3 Άδειες και Ευχαριστίες για Ενσωματωμένο Λογισμικό

Αυτή η ενότητα είναι μια ημιτελής, αλλά αυξανόμενη λίστα αδειών και ευχαριστιών για λογισμικό τρίτων, που ενσωματώνεται στην διανομή της Python.

### Γ'.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).

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)
```

### Γ'.3.2 Sockets

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

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

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

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

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

### Γ'.3.3 Ασύγχρονες socket υπηρεσίες

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.

### Γ'.3.4 Διαχείριση Cookie

Η ενότητα `http.cookies` περιέχει την παρακάτω ειδοποίηση:

```
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.
```

### Γ'.3.5 Ανίχνευση εκτέλεσης

Η ενότητα `trace` περιέχει την παρακάτω ειδοποίηση:

```
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.
```

### Γ'.3.6 Συναρτήσεις UUencode και UUdecode

Η ενότητα uu περιέχει την παρακάτω ειδοποίηση:

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

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

### Γ'.3.7 Κλήσεις Απομακρυσμένης Διαδικασίας XML

Η ενότητα xmlrpc.client περιέχει την παρακάτω ειδοποίηση:

```
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.
```

### Γ'.3.8 test\_epoll

The test\_epoll module contains the following notice:

Copyright (c) 2001-2006 Twisted Matrix Laboratories.

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

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

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

### Γ'.3.9 Επιλογή kqueue

Η ενότητα select περιέχει την παρακάτω ειδοποίηση για την kqueue διεπαφή:

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.



### Γ'.3.10 SipHash24

Το αρχείο `Python/pyhash.c` περιέχει την υλοποίηση του Marek Majkowski του αλγορίθμου του Dan Bernstein, SipHash24. Αυτό περιέχει την παρακάτω σημείωση:

```
<MIT License>
Copyright (c) 2013 Marek Majkowski <marek@popcount.org>

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.
</MIT License>

Original location:
  https://github.com/majek/csiphash/

Solution inspired by code from:
  Samuel Neves (supercop/crypto_auth/siphash24/little)
  djb (supercop/crypto_auth/siphash24/little2)
  Jean-Philippe Aumasson (https://131002.net/siphash/siphash24.c)
```

### Γ'.3.11 strtod και dtoa

Το αρχείο `Python/dtoa.c`, που παρέχει τις συναρτήσεις `dtoa` και `strtod` της C για μετατροπή των C doubles προς και από strings, προέρχεται από το ομώνυμο αρχείο του David M. Gay, προς το παρόν διαθέσιμο από <https://web.archive.org/web/20220517033456/http://www.netlib.org/fp/dtoa.c>. Το αρχικό αρχείο, όπως ανακτήθηκε στις 16 Μαρτίου, 2009, περιέχει τα ακόλουθα πνευματικά δικαιώματα και την ειδοποίηση αδειοδότησης:

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

## Γ'.3.12 OpenSSL

The modules `hashlib`, `posix`, `ssl`, `crypt` use the OpenSSL library for added performance if made available by the operating system. Additionally, the Windows and macOS 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](mailto:openssl-core@openssl.org).

### OpenSSL License

-----

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

(συνέχεια στην επόμενη σελίδα)

(συνεχίζεται από την προηγούμενη σελίδα)

```
* ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
* OF THE POSSIBILITY OF SUCH DAMAGE.
* =====
*
* This product includes cryptographic software written by Eric Young
* (eay@cryptsoft.com). This product includes software written by Tim
* Hudson (tjh@cryptsoft.com).
*
*/
```

## Original SSLeay License

-----

```
/* Copyright (C) 1995-1998 Eric Young (eay@cryptsoft.com)
* All rights reserved.
*
* This package is an SSL implementation written
* by Eric Young (eay@cryptsoft.com).
* The implementation was written so as to conform with Netscapes SSL.
*
* This library is free for commercial and non-commercial use as long as
* the following conditions are aheared to. The following conditions
* apply to all code found in this distribution, be it the RC4, RSA,
* lhash, DES, etc., code; not just the SSL code. The SSL documentation
* included with this distribution is covered by the same copyright terms
* except that the holder is Tim Hudson (tjh@cryptsoft.com).
*
* Copyright remains Eric Young's, and as such any Copyright notices in
* the code are not to be removed.
* If this package is used in a product, Eric Young should be given attribution
* as the author of the parts of the library used.
* This can be in the form of a textual message at program startup or
* in documentation (online or textual) provided with the package.
*
* Redistribution and use in source and binary forms, with or without
* modification, are permitted provided that the following conditions
* are met:
* 1. Redistributions of source code must retain the copyright
* notice, this list of conditions and the following disclaimer.
* 2. Redistributions in binary form must reproduce the above copyright
* notice, this list of conditions and the following disclaimer in the
* documentation and/or other materials provided with the distribution.
* 3. All advertising materials mentioning features or use of this software
* must display the following acknowledgement:
* "This product includes cryptographic software written by
* Eric Young (eay@cryptsoft.com)"
* The word 'cryptographic' can be left out if the rouines from the library
* being used are not cryptographic related :-).
* 4. If you include any Windows specific code (or a derivative thereof) from
* the apps directory (application code) you must include an acknowledgement:
* "This product includes software written by Tim Hudson (tjh@cryptsoft.com)"
*
* THIS SOFTWARE IS PROVIDED BY ERIC YOUNG ``AS IS'' AND
* ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
* IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
* ARE DISCLAIMED. IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
* FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
* DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
* OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
* HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
* LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
```

(συνέχεια στην επόμενη σελίδα)

(συνεχίζεται από την προηγούμενη σελίδα)

```
* 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.]
*/
```

### Γ'.3.13 expat

The pyexpat extension is built using an included copy of the expat sources unless the build is configured `--with-system-expat`:

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

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

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

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

### Γ'.3.14 libffi

The \_ctypes extension is built using an included copy of the libffi sources unless the build is configured `--with-system-libffi`:

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

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

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

THE SOFTWARE IS PROVIDED ``AS IS'', WITHOUT WARRANTY OF ANY KIND,
EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF
MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND
```

(συνέχεια στην επόμενη σελίδα)

(συνεχίζεται από την προηγούμενη σελίδα)

```
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.
```

### Γ'.3.15 zlib

Η επέκταση zlib δημιουργείται χρησιμοποιώντας ένα συμπεριλαμβανόμενου αντίγραφο των πηγών zlib, εάν η έκδοση του zlib που βρίσκεται στο σύστημα είναι πολύ παλιά για να χρησιμοποιηθεί για την κατασκευή:

```
Copyright (C) 1995-2011 Jean-loup Gailly and Mark Adler
```

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

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

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

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

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

### Γ'.3.16 cfuhash

Η υλοποίηση του πίνακα κατακερματισμού που χρησιμοποιείται από το `tracemalloc` βασίζεται στο έργο `cfuhash`:

```
Copyright (c) 2005 Don Owens
All rights reserved.
```

```
This code is released under the BSD license:
```

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

- \* Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
- \* 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.
- \* Neither the name of the author nor the names of its

(συνέχεια στην επόμενη σελίδα)

(συνεχίζεται από την προηγούμενη σελίδα)

```
contributors may 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.
```

### Γ'.3.17 libmpdec

The `_decimal` module is built using an included copy of the libmpdec library unless the build is configured `--with-system-libmpdec`:

```
Copyright (c) 2008-2020 Stefan Krah. 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.
```

### Γ'.3.18 W3C C14N σουίτα δοκιμής

Η σουίτα δοκιμής C14N 2.0 στο πακέτο `test` (`Lib/test/xmltestdata/c14n-20/`) ανακτήθηκε από τον ιστότοπο του W3C <https://www.w3.org/TR/xml-c14n2-testcases/> και διανέμεται με την άδεια 3 ρήτρων BSD:

```
Copyright (c) 2013 W3C(R) (MIT, ERCIM, Keio, Beihang),
All Rights Reserved.
```

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

(συνέχεια στην επόμενη σελίδα)

(συνεχίζεται από την προηγούμενη σελίδα)

```
are met:
```

- \* Redistributions of works must retain the original copyright notice, this list of conditions and the following disclaimer.
- \* Redistributions in binary form must reproduce the original copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
- \* Neither the name of the W3C nor the names of its contributors may be used to endorse or promote products derived from this work 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.
```

### Γ'.3.19 Audioop

Το module audioop χρησιμοποιεί ως βάση κώδικα του αρχείου g771.c του έργου Sox. <https://sourceforge.net/projects/sox/files/sox/12.17.7/sox-12.17.7.tar.gz>

Αυτό ο πηγαίος κώδικας είναι προϊόν της Sun Microsystems, Inc. και παρέχεται για απεριόριστη χρήση. Οι χρήστες μπορούν να αντιγράψουν ή να τροποποιήσουν αυτόν τον πηγαίο κώδικα χωρίς χρέωση.

Ο ΠΗΓΑΙΟΣ ΚΩΔΙΚΑΣ ΤΟΥ SUN ΠΑΡΕΧΕΤΑΙ ΟΠΩΣ ΕΧΕΙ ΧΩΡΙΣ ΚΑΝΕΝΟΣ ΕΙΔΟΥΣ ΕΓΓΥΗΣΕΙΣ ΣΥΜΠΕΡΙΛΑΜΒΑΝΟΜΕΝΩΝ ΕΓΓΥΗΣΕΩΝ ΣΧΕΔΙΑΣΜΟΥ, ΕΜΠΟΡΕΥΣΙΜΟΤΗΤΑΣ ΚΑΙ ΚΑΤΑΛΛΗΛΟΤΗΤΑΣ ΓΙΑ ΣΥΓΚΕΚΡΙΜΕΝΟ ΣΚΟΠΟ Ή ΠΟΥ ΠΡΟΚΥΠΤΕΙ ΑΠΟ ΚΑΠΟΙΑ ΠΟΡΕΙΑ ΣΥΝΑΛΛΑΓΗΣ, ΧΡΗΣΗΣ Ή ΕΜΠΟΡΙΚΗΣ ΠΡΑΚΤΙΚΗΣ.

Ο πηγαίος κώδικας του Sun παρέχεται χωρίς την υποστήριξη και χωρίς καμία υποχρέωση εκ μέρους της Sun Microsystems, Inc. να βοηθήσει στην χρήση, στη διόρθωση, τροποποίηση ή βελτίωση του.

SUN MICROSYSTEMS, INC. SHALL HAVE NO LIABILITY WITH RESPECT TO THE INFRINGEMENT OF COPYRIGHTS, TRADE SECRETS OR ANY PATENTS BY THIS SOFTWARE OR ANY PART THEREOF.

Σε καμία περίπτωση η Sun Microsystems, Inc. δεν φέρει ευθύνη για τυχόν απώλεια εσόδων ή κερδών ή άλλες ειδικές, έμμεσες και επακόλουθες ζημιές, ακόμη και αν η Sun έχει ενημερωθεί για την πιθανότητα τέτοιων ζημιών.

Sun Microsystems, Inc. 2550 Garcia Avenue Mountain View, Καλιφόρνια 94043





## ΠΑΡΑΡΤΗΜΑ Δ'

---

### Copyright

---

Η Python και αυτή η τεκμηρίωση είναι:

Copyright © 2001-2023 Python Software Foundation. Όλα τα δικαιώματα διατηρούνται.

Copyright © 2000 BeOpen.com. Όλα τα δικαιώματα διατηρούνται.

Copyright © 1995-2000 Corporation for National Research Initiatives. Όλα τα δικαιώματα διατηρούνται.

Copyright © 1991-1995 Stichting Mathematisch Centrum. Όλα τα δικαιώματα διατηρούνται.

---

Ανατρέξτε στο [Ιστορία και Άδεια](#) για πλήρης πληροφόρηση σχετικά με την άδεια χρήσης και τις εξουσιοδοτήσεις.



## μη-αλφαβητικά

..., [55](#)  
 -?  
     command line option, [5](#)  
 %APPDATA%, [36](#)  
 2to3, [55](#)  
 >>>, [55](#)  
 -B  
     command line option, [6](#)  
 BDFL, [57](#)  
 CFLAGS, [28](#), [29](#)  
 CFLAGS\_NODIST, [28](#), [29](#)  
 CPPFLAGS, [27](#), [30](#)  
 CPython, [58](#)  
 C-contiguous, [58](#)  
 -E  
     command line option, [6](#)  
 EAFP, [59](#)  
 Fortran contiguous, [58](#)  
 GIL, [61](#)  
 -I  
     command line option, [6](#)  
 IDLE, [62](#)  
 -J  
     command line option, [9](#)  
 LBYL, [63](#)  
 LDFlags, [27](#), [29](#), [30](#)  
 LDFlags\_NODIST, [29](#), [30](#)  
 MRO, [64](#)  
 -O  
     command line option, [6](#)  
 -OO  
     command line option, [6](#)  
 OPT, [23](#)  
 PATH, [9](#), [17](#), [32](#), [34](#), [3941](#), [43](#)  
 PATHEXT, [34](#)  
 PEP, [66](#)  
 PROFILE\_TASK, [21](#)  
 PYTHON\*, [46](#)  
 PYTHONCOERCECLOCALE, [20](#)  
 PYTHONDEBUG, [6](#)  
 PYTHONDONTWRITEBYTECODE, [6](#)  
 PYTHONDUMPPREFS, [23](#)  
 PYTHONHASHSEED, [6](#), [7](#), [10](#)  
 PYTHONHOME, [6](#), [9](#), [45](#), [46](#)  
 PYTHONINSPECT, [6](#)  
 PYTHONINTMAXSTRDIGITS, [8](#)  
 PYTHONIOENCODING, [13](#)  
 PYTHONLEGACYWINDOWSSTDIO, [11](#)  
 PYTHONMALLOC, [12](#), [22](#)  
 PYTHONOPTIMIZE, [6](#)  
 PYTHONPATH, [6](#), [9](#), [39](#), [45](#), [46](#), [50](#)  
 PYTHONPROFILEIMPORTTIME, [8](#)  
 PYTHONPYCACHEPREFIX, [8](#)  
 PYTHONSTARTUP, [6](#)  
 PYTHONTREADDEBUG, [22](#)  
 PYTHONUNBUFFERED, [7](#)  
 PYTHONUTF8, [13](#), [40](#)  
 PYTHONVERBOSE, [7](#)  
 PYTHONWARNDEFAULTENCODING, [8](#)  
 PYTHONWARNINGS, [8](#)  
 PY\_PYTHON, [43](#)  
 Python 3000, [66](#)  
 Python Enhancement Proposals  
     PEP 1, [66](#)  
     PEP 8, [53](#)  
     PEP 11, [31](#), [47](#)  
     PEP 238, [60](#)  
     PEP 278, [69](#)  
     PEP 302, [60](#), [63](#)  
     PEP 338, [4](#)  
     PEP 343, [58](#)  
     PEP 362, [56](#), [66](#)  
     PEP 370, [7](#), [11](#)  
     PEP 397, [41](#)  
     PEP 411, [66](#)  
     PEP 420, [60](#), [65](#), [66](#)  
     PEP 443, [61](#)  
     PEP 451, [60](#)  
     PEP 483, [61](#)  
     PEP 484, [55](#), [60](#), [61](#), [69](#)  
     PEP 488, [6](#)  
     PEP 492, [56](#), [58](#)  
     PEP 498, [59](#)  
     PEP 519, [66](#)  
     PEP 525, [56](#)  
     PEP 526, [55](#), [69](#)

PEP 528, 40  
PEP 529, 12, 40  
PEP 538, 13, 20  
PEP 585, 61  
PEP 3116, 69  
PEP 3155, 67  
Pythonic, 66  
-R  
    command line option, 6  
-S  
    command line option, 7  
TEMP, 36  
-V  
    command line option, 5  
-W arg  
    command line option, 7  
-X  
    command line option, 8  
Zen της Python, 69  
\_\_future\_\_, 60  
\_\_slots\_\_, 67  
annotation, 55  
annotation μεταβλητής, 69  
awaitable, 56  
-b  
    command line option, 6  
bytecode, 57  
bytes-like αντικείμενα, 57  
-c <command>  
    command line option, 4  
callable, 57  
callback, 57  
--check-hash-based-pycs  
    default|always|never  
    command line option, 6  
coercion, 57  
command line option  
    -?, 5  
    -B, 6  
    -E, 6  
    -I, 6  
    -J, 9  
    -O, 6  
    -OO, 6  
    -R, 6  
    -S, 7  
    -V, 5  
    -W arg, 7  
    -X, 8  
    -b, 6  
    -c <command>, 4  
    --check-hash-based-pycs  
        default|always|never, 6  
    -d, 6  
    --disable-ipv6, 19  
    --disable-test-modules, 21  
    --enable-big-digits=[15|30], 19  
    --enable-framework, 25  
    --enable-framework=INSTALLDIR, 25  
    --enable-loadable-sqlite-  
        extensions, 19  
    --enable-optimizations, 21  
    --enable-profiling, 22  
    --enable-shared, 23  
    --enable-universalsdk, 25  
    --enable-universalsdk=SDKDIR, 25  
    --exec-prefix=EPREFIX, 21  
    -h, 5  
    --help, 5  
    -i, 6  
    -m <module-name>, 4  
    --prefix=PREFIX, 21  
    -q, 6  
    -s, 7  
    -u, 7  
    -v, 7  
    --version, 5  
    --with-address-sanitizer, 23  
    --with-assertions, 23  
    --with-builtin-hashlib-  
        hashes=md5, sha1, sha256, sha512, sha3, blake2,  
        25  
    --with-computed-gotos, 21  
    --with-cxx-main, 19  
    --with-cxx-main=COMPILER, 19  
    --with-dbmliborder=db1:db2:..., 20  
    --with-dtrace, 23  
    --with-ensurepip=[upgrade|install|no],  
        21  
    --with-framework-name=FRAMEWORK, 26  
    --with-hash-  
        algorithm=[fnv|siphash24],  
        25  
    --with-libc=STRING, 24  
    --with-libm=STRING, 24  
    --with-libs='lib1 ...', 24  
    --with-lto, 21  
    --with-memory-sanitizer, 23  
    --with-openssl=DIR, 24  
    --with-openssl-  
        rpath=[no|auto|DIR], 24  
    --without-c-locale-coercion, 20  
    --without-decimal-contextvar, 20  
    --without-doc-strings, 22  
    --without-pymalloc, 21  
    --without-readline, 24  
    --without-static-libpython, 23  
    --with-platlibdir=DIRNAME, 20  
    --with-pydebug, 23  
    --with-readline=editline, 24  
    --with-ssl-default-  
        suites=[python|openssl|STRING],  
        25  
    --with-suffix=SUFFIX, 19  
    --with-system-expat, 24  
    --with-system-ffi, 24

```

--with-system-libmpdec, 24
--with-tcltk-includes='-I...', 24
--with-tcltk-libs='-L...', 24
--with-trace-refs, 23
--with-tzpath=<list of absolute
    paths separated by pathsep>, 20
--with-undefined-behavior-
    sanitizer, 23
--with-universal-archs=ARCH, 25
--with-valgrind, 23
--with-wheel-pkg-dir=PATH, 20
-x, 8
context μεταβλητή, 58
contiguous, 58
coroutine, 58
coroutine συνάρτηση, 58
-d
    command line option, 6
decorator, 58
descriptor, 58
--disable-ipv6
    command line option, 19
--disable-test-modules
    command line option, 21
docstring, 59
duck-typing, 59
--enable-big-digits=[15|30]
    command line option, 19
--enable-framework
    command line option, 25
--enable-framework=INSTALLDIR
    command line option, 25
--enable-loadable-sqlite-extensions
    command line option, 19
--enable-optimizations
    command line option, 21
--enable-profiling
    command line option, 22
--enable-shared
    command line option, 23
--enable-universalsdk
    command line option, 25
--enable-universalsdk=SDKDIR
    command line option, 25
--exec-prefix=EPREFIX
    command line option, 21
f-string, 59
finder, 60
generator, 60
generator expression, 61
generator iterator, 61
generator έκφραση, 61
global interpreter lock, 61
-h
    command line option, 5
hash-based pyc, 61
hashable, 61
--help
    command line option, 5
-i
    command line option, 6
immutable, 62
interpreted, 62
iterable, 62
iterator, 62
lambda, 63
list comprehension, 63
loader, 63
-m <module-name>
    command line option, 4
magic
    method, 63
mapping, 63
meta path finder, 64
method
    magic, 63
    special, 68
module, 64
module επέκτασης, 59
mutable, 64
named tuple, 64
namespace, 64
nested scope, 65
path based finder, 66
path entry, 66
path entry finder, 66
path entry hook, 66
path-like αντικείμενο, 66
--prefix=PREFIX
    command line option, 21
provisional API, 66
provisional πακέτο, 66
-q
    command line option, 6
-s
    command line option, 7
set comprehension, 68
slice, 68
special
    method, 68
strong reference, 68
type alias, 68
type hint, 69
-u
    command line option, 7
-v
    command line option, 7
--version
    command line option, 5
virtual environment, 69
virtual machine, 69
--with-address-sanitizer
    command line option, 23
--with-assertions
    command line option, 23

```

```

--with-builtin-hashlib-
    hashes=md5,sha1,sha256,sha512,sha3,blake2
    command line option, 25
--with-computed-gotos
    command line option, 21
--with-cxx-main
    command line option, 19
--with-cxx-main=COMPILER
    command line option, 19
--with-dbmliborder=db1:db2:...
    command line option, 20
--with-dtrace
    command line option, 23
--with-ensurepip=[upgrade|install|no]
    command line option, 21
--with-framework-name=FRAMEWORK
    command line option, 26
--with-hash-algorithm=[fnv|siphash24]
    command line option, 25
--with-libc=STRING
    command line option, 24
--with-libm=STRING
    command line option, 24
--with-libs='lib1 ...'
    command line option, 24
--with-lto
    command line option, 21
--with-memory-sanitizer
    command line option, 23
--with-openssl=DIR
    command line option, 24
--with-openssl-rpath=[no|auto|DIR]
    command line option, 24
--without-c-locale-coercion
    command line option, 20
--without-decimal-contextvar
    command line option, 20
--without-doc-strings
    command line option, 22
--without-pymalloc
    command line option, 21
--without-readline
    command line option, 24
--without-static-libpython
    command line option, 23
--with-platlibdir=DIRNAME
    command line option, 20
--with-pydebug
    command line option, 23
--with-readline=editline
    command line option, 24
--with-ssl-default-
    suites=[python|openssl|STRING]
    command line option, 25
--with-suffix=SUFFIX
    command line option, 19
--with-system-expat
    command line option, 24
--with-system-ffi
    command line option, 24
--with-system-libmpdec
    command line option, 24
--with-tcltk-includes='-I... '
    command line option, 24
--with-tcltk-libs='-L... '
    command line option, 24
--with-trace-refs
    command line option, 23
--with-tzpath=<list of absolute paths
    separated by pathsep>
    command line option, 20
--with-undefined-behavior-sanitizer
    command line option, 23
--with-universal-archs=ARCH
    command line option, 25
--with-valgrind
    command line option, 23
--with-wheel-pkg-dir=PATH
    command line option, 20
-x
    command line option, 8

```

## A

ακέραια διαίρεση, [60](#)  
 ακολουθία, [67](#)  
 αναγνωρισμένο όνομα, [67](#)  
 αντικείμενο, [65](#)  
 αντικείμενο αρχείου, [59](#)  
 αντικείμενο που μοιάζει με αρχείο, [59](#)  
 αρχείο κειμένου, [68](#)  
 ασύγχρονος generator, [56](#)  
 ασύγχρονος generator iterator, [56](#)  
 ασύγχρονος iterable, [56](#)  
 ασύγχρονος iterator, [56](#)  
 ασύγχρονος διαχειριστής context, [56](#)  
 αφηρημένη βασική κλάση, [55](#)

## Γ

γενική συνάρτηση, [61](#)  
 γενικός τύπος, [61](#)

## Δ

δανεική αναφορά, [57](#)  
 δήλωση, [68](#)  
 διαδραστικός, [62](#)  
 διαχειριστής context, [58](#)  
 δυαδικό αρχείο, [57](#)

## Ε

ειδική μέθοδος, [68](#)  
 εισαγόμενο path, [62](#)  
 εισαγωγέας, [62](#)  
 εισαγωγή, [62](#)  
 έκφραση, [59](#)

## Κ

καθολικές νέες γραμμές, [69](#)  
 κανονικό πακέτο, [67](#)  
 κατανόηση λεξικού, [59](#)  
 κλάση, [57](#)  
 κλάση νέου στυλ, [65](#)  
 κωδικοποίηση κειμένου, [68](#)  
 κωδικοποίηση συστήματος αρχείων και  
 χειριστής σφαλμάτων, [59](#)

## Λ

λεξικό, [59](#)  
 λίστα, [63](#)

## Μ

μαγική μέθοδος, [63](#)  
 μέθοδος, [64](#)  
 μετα-κλάση, [64](#)  
 μεταβλητή κλάσης, [57](#)  
 μεταβλητή περιβάλλοντος  
   %APPDATA%, [36](#)  
   BASECFLAGS, [28](#)  
   BASECPPFLAGS, [27](#)  
   BLDSHARED, [30](#)  
   CC, [28](#)  
   CCSHARED, [28](#)  
   CFLAGS, [28, 29](#)  
   CFLAGSFORSHARED, [29](#)  
   CFLAGS\_ALIASING, [28](#)  
   CFLAGS\_NODIST, [28, 29](#)  
   CONFIGURE\_CFLAGS, [28](#)  
   CONFIGURE\_CFLAGS\_NODIST, [28](#)  
   CONFIGURE\_CPPFLAGS, [27](#)  
   CONFIGURE\_LDFLAGS, [29](#)  
   CONFIGURE\_LDFLAGS\_NODIST, [29](#)  
   CPPFLAGS, [27, 30](#)  
   CXX, [28](#)  
   EXTRA\_CFLAGS, [28](#)  
   LDFLAGS, [27, 29, 30](#)  
   LDFLAGS\_NODIST, [29, 30](#)  
   LDSHARED, [30](#)  
   LIBS, [30](#)  
   LINKCC, [29](#)  
   MAINCC, [28](#)  
   OPT, [23, 28](#)  
   PATH, [9, 17, 32, 34, 3941, 43](#)  
   PATHEXT, [34](#)  
   PROFILE\_TASK, [21](#)  
   PURIFY, [29](#)  
   PYTHON\*, [46](#)  
   PYTHONASYNCIODEBUG, [11](#)  
   PYTHONBREAKPOINT, [9](#)  
   PYTHONCASEOK, [10](#)  
   PYTHONCOERCECLOCALE, [12, 20](#)  
   PYTHONDEBUG, [6, 10](#)  
   PYTHONDEVMODE, [13](#)  
   PYTHONDONTWRITEBYTECODE, [6, 10](#)  
   PYTHONDUMPREFS, [13, 23](#)

PYTHONEXECUTABLE, [11](#)  
 PYTHONFAULTHANDLER, [11](#)  
 PYTHONHASHSEED, [6, 7, 10](#)  
 PYTHONHOME, [6, 9, 45, 46](#)  
 PYTHONINSPECT, [6, 10](#)  
 PYTHONINTMAXSTRDIGITS, [8, 10](#)  
 PYTHONIOENCODING, [10, 13](#)  
 PYTHONLEGACYWINDOWSFSENCODING, [12](#)  
 PYTHONLEGACYWINDOWSSSTDIO, [11, 12](#)  
 PYTHONMALLOC, [11, 12, 22](#)  
 PYTHONMALLOCSTATS, [12](#)  
 PYTHONNOUSERSITE, [11](#)  
 PYTHONOPTIMIZE, [6, 9](#)  
 PYTHONPATH, [6, 9, 39, 45, 46, 50](#)  
 PYTHONPLATLIBDIR, [9](#)  
 PYTHONPROFILEIMPORTTIME, [8, 11](#)  
 PYTHONPYCACHEPREFIX, [8, 10](#)  
 PYTHONSTARTUP, [6, 9](#)  
 PYTHONTREADDEBUG, [13, 22](#)  
 PYTHONTRACEMALLOC, [11](#)  
 PYTHONUNBUFFERED, [7, 10](#)  
 PYTHONUSERBASE, [11](#)  
 PYTHONUTF8, [13, 40](#)  
 PYTHONVERBOSE, [7, 10](#)  
 PYTHONWARNDEFAULTENCODING, [8, 13](#)  
 PYTHONWARNINGS, [8, 11](#)  
 PY\_BUILTIN\_MODULE\_CFLAGS, [29](#)  
 PY\_CFLAGS, [29](#)  
 PY\_CFLAGS\_NODIST, [29](#)  
 PY\_CORE\_CFLAGS, [29](#)  
 PY\_CORE\_LDFLAGS, [30](#)  
 PY\_CPPFLAGS, [27](#)  
 PY\_LDFLAGS, [30](#)  
 PY\_LDFLAGS\_NODIST, [30](#)  
 PY\_PYTHON, [43](#)  
 PY\_STDMODULE\_CFLAGS, [29](#)  
 TEMP, [36](#)

μιγαδικός αριθμός, [58](#)  
 μοναδικό dispatch, [68](#)

## Ο

όρισμα, [55](#)  
 όρισμα keyword, [63](#)  
 όρισμα θέσης, [66](#)  
 όψη λεξικού, [59](#)

## Π

πακέτο, [65](#)  
 πακέτο namespace, [65](#)  
 παράμετρος, [65](#)  
 πλήθος αναφοράς, [67](#)

## Σ

σειρά ανάλυσης μεθόδων, [64](#)  
 συλλογή απορριμάτων, [60](#)  
 συμβολοσειρά τριπλών εισαγωγικών, [68](#)  
 συνάρτηση, [60](#)  
 συνάρτηση annotation, [60](#)

συνάρτηση `key`, [63](#)

## **T**

τερματισμός λειτουργίας διερμηνέα, [62](#)

τεχνικές προδιαγραφές `module`, [64](#)

τμήμα, [66](#)

τοπική κωδικοποίηση, [63](#)

τύπος, [68](#)

## **X**

χαρακτηριστικό, [56](#)