

---

# Python Setup and Usage

*Yayım 3.10.19*

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

**Ekim 16, 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 . . . . .	14
<b>2</b>	<b>Using Python on Unix platforms</b>	<b>15</b>
2.1	Getting and installing the latest version of Python . . . . .	15
2.1.1	On Linux . . . . .	15
2.1.2	On FreeBSD and OpenBSD . . . . .	15
2.1.3	On OpenSolaris . . . . .	16
2.2	Building Python . . . . .	16
2.3	Python-related paths and files . . . . .	16
2.4	Miscellaneous . . . . .	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 . . . . .	24
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 . . . . .	27
3.2.4	C extensions . . . . .	27
3.3	Compiler and linker flags . . . . .	28
3.3.1	Preprocessor flags . . . . .	28

3.3.2	Compiler flags . . . . .	28
3.3.3	Linker flags . . . . .	30
<b>4</b>	<b>Using Python on Windows</b>	<b>33</b>
4.1	The full installer . . . . .	34
4.1.1	Installation steps . . . . .	34
4.1.2	Removing the MAX_PATH Limitation . . . . .	35
4.1.3	Installing Without UI . . . . .	35
4.1.4	Installing Without Downloading . . . . .	37
4.1.5	Modifying an install . . . . .	37
4.2	The Microsoft Store package . . . . .	38
4.2.1	Known issues . . . . .	38
4.3	The nuget.org packages . . . . .	39
4.4	The embeddable package . . . . .	40
4.4.1	Python Application . . . . .	40
4.4.2	Embedding Python . . . . .	41
4.5	Alternative bundles . . . . .	41
4.6	Configuring Python . . . . .	41
4.6.1	Excursus: Setting environment variables . . . . .	41
4.6.2	Finding the Python executable . . . . .	42
4.7	UTF-8 mode . . . . .	42
4.8	Python Launcher for Windows . . . . .	43
4.8.1	Getting started . . . . .	43
4.8.2	Shebang Lines . . . . .	45
4.8.3	Arguments in shebang lines . . . . .	45
4.8.4	Customization . . . . .	45
4.8.5	Diagnostics . . . . .	47
4.9	Finding modules . . . . .	47
4.10	Additional modules . . . . .	48
4.10.1	PyWin32 . . . . .	48
4.10.2	cx_Freeze . . . . .	49
4.11	Compiling Python on Windows . . . . .	49
4.12	Other Platforms . . . . .	49
<b>5</b>	<b>Using Python on a Mac</b>	<b>51</b>
5.1	Getting and Installing MacPython . . . . .	51
5.1.1	How to run a Python script . . . . .	52
5.1.2	Running scripts with a GUI . . . . .	52
5.1.3	Configuration . . . . .	52
5.2	The IDE . . . . .	52
5.3	Installing Additional Python Packages . . . . .	52
5.4	GUI Programming on the Mac . . . . .	53
5.5	Distributing Python Applications on the Mac . . . . .	53
5.6	Other Resources . . . . .	53
<b>6</b>	<b>Editors and IDEs</b>	<b>55</b>
<b>A</b>	<b>Sözlük</b>	<b>57</b>
<b>B</b>	<b>Dokümanlar hakkında</b>	<b>71</b>
B.1	Python Dokümantasyonuna Katkıda Bulunanlar . . . . .	71
<b>C</b>	<b>Tarihçe ve Lisans</b>	<b>73</b>
C.1	Yazılımın tarihçesi . . . . .	73
C.2	Python'a erişmek veya başka bir şekilde kullanmak için şartlar ve koşullar . . . . .	74

C.2.1	PYTHON İÇİN PSF LİSANS ANLAŞMASI 3.10.19 . . . . .	74
C.2.2	PYTHON 2.0 İÇİN BEOPEN.COM LİSANS SÖZLEŞMESİ . . . . .	75
C.2.3	PYTHON 1.6.1 İÇİN CNRI LİSANS ANLAŞMASI . . . . .	76
C.2.4	0.9.0 ARASI 1.2 PYTHON İÇİN CWI LİSANS SÖZLEŞMESİ . . . . .	77
C.2.5	PYTHON 3.10.19 BELGELERİNDEKİ KOD İÇİN SIFIR MADDE BSD LİSANSI . . . . .	78
C.3	Tüzel Yazılımlar için Lisanslar ve Onaylar . . . . .	78
C.3.1	Mersenne Twister'ı . . . . .	78
C.3.2	Soketler . . . . .	79
C.3.3	Asenkron soket hizmetleri . . . . .	79
C.3.4	Çerez yönetimi . . . . .	80
C.3.5	Çalıştırma izleme . . . . .	80
C.3.6	UUencode ve UUdecode fonksiyonları . . . . .	81
C.3.7	XML Uzaktan Yordam Çağrıları . . . . .	82
C.3.8	test_epoll . . . . .	82
C.3.9	kqueue seçin . . . . .	83
C.3.10	SipHash24 . . . . .	83
C.3.11	strtod ve dtoa . . . . .	84
C.3.12	OpenSSL . . . . .	84
C.3.13	expat . . . . .	86
C.3.14	libffi . . . . .	87
C.3.15	zlib . . . . .	88
C.3.16	cfuhash . . . . .	88
C.3.17	libmpdec . . . . .	89
C.3.18	W3C C14N test paketi . . . . .	89
C.3.19	Audioop . . . . .	90
<b>D</b>	<b>Telif Hakkı</b>	<b>91</b>
<b>Dizin</b>		<b>93</b>



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





---

## Command line and environment

---

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

**CPython uygulama ayrıntısı:** 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.

---

**Not:** 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`.

**Ayrıca bakınız:**

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

**PEP 338** – Executing modules as scripts

3.1 sürümünde değişti: Supply the package name to run a `__main__` submodule.

3.4 sürümünde değişti: 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`.

**Ayrıca bakınız:**

**`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`).

**Ayrıca bakınız:**

tut-invoking

3.4 sürümünde değişti: 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]
```

3.6 sürümünde geldi: 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`).

3.5 sürümünde değişti: 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.

3.4 sürümünde geldi.

**-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](#).

3.5 sürümünde değişti: 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](#)).

3.5 sürümünde değişti: Modify `.pyc` filenames according to [PEP 488](#).

**-q**

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

3.2 sürümünde geldi.

**-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.

3.7 sürümünde değişti: The option is no longer ignored.

3.2.3 sürümünde geldi.

**-s**

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

**Ayrıca bakınız:**

**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`.

3.7 sürümünde değişti: 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.

3.10 sürümünde değişti: 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.

3.2 sürümünde değişti: The `-X` option was added.

3.3 sürümünde geldi: The `-X faulthandler` option.

3.4 sürümünde geldi: The `-X showrefcount` and `-X tracemalloc` options.

3.6 sürümünde geldi: The `-X showalloccount` option.

3.7 sürümünde geldi: The `-X importtime`, `-X dev` and `-X utf8` options.

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

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

The `-X showalloccount` option has been removed.

3.10 sürümünde geldi: The `-X warn_default_encoding` option.

3.10.7 sürümünde geldi: 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.

3.9 sürümünde geldi.

### **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.

3.7 sürümünde geldi.

### **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.

3.10.15 sürümünde değişti: (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.

3.8 sürümünde geldi.

### **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.

3.2.3 sürümünde geldi.

#### **PYTHONINTMAXSTRDIGITS**

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

3.10.7 sürümünde geldi.

#### **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'`.

3.4 sürümünde değişti: The `encodingname` part is now optional.

3.6 sürümünde değişti: On Windows, the encoding specified by this variable is ignored for interactive console buffers unless `PYTHONLEGACYWINDOWSSSTDIO` 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`.

**Ayrıca bakınız:**

**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`.

**Ayrıca bakınız:**

**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.

3.3 sürümünde geldi.

### **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.

3.4 sürümünde geldi.

### **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.

3.7 sürümünde geldi.

### **PYTHONASYNCIODEBUG**

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

3.4 sürümünde geldi.

### **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.

3.7 sürümünde değişti: Added the "default" allocator.

3.6 sürümünde geldi.

### **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.

3.6 sürümünde değişti: 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.

3.6 sürümünde geldi: See [PEP 529](#) for more details.

**PYTHONLEGACYWINDOWSTDIO**

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.

3.6 sürümünde geldi.

**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.

3.7 sürümünde geldi: 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.

3.7 sürümünde geldi.

**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.

3.7 sürümünde geldi.

### **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.

3.10 sürümünde geldi.

## **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.

---

## Using Python on Unix platforms

---

### 2.1 Getting and installing the latest version of Python

#### 2.1.1 On Linux

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

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

**Ayrıca bakınız:**

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

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

[https://docs-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) for Fedora users

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

#### 2.1.2 On FreeBSD and OpenBSD

- FreeBSD users, to add the package use:

```
pkg install python3
```

- OpenBSD users, to add the package use:

```
pkg_add -r python
```

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

(sonraki sayfaya devam)

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

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

### 2.1.3 On OpenSolaris

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

## 2.2 Building Python

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

The build process consists of the usual commands:

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

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

**Uyarı:** `make install` can overwrite or masquerade the `python3` binary. `make altinstall` is therefore recommended instead of `make install` since it only installs `exec_prefix/bin/pythonversion`.

### 2.3 Python-related paths and files

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

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

File/directory	Meaning
<code>exec_prefix/bin/python3</code>	Recommended location of the interpreter.
<code>prefix/lib/pythonversion</code> , <code>exec_prefix/lib/pythonversion</code>	Recommended locations of the directories containing the standard modules.
<code>prefix/include/pythonversion</code> , <code>exec_prefix/include/pythonversion</code>	Recommended locations of the directories containing the include files needed for developing Python extensions and embedding the interpreter.

## 2.4 Miscellaneous

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

```
$ chmod +x script
```

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

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

which searches for the Python interpreter in the whole `PATH`. However, some Unices may not have the `env` command, so you may need to hardcode `/usr/bin/python3` as the interpreter path.

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

## 2.5 Custom OpenSSL

1. To use your vendor's OpenSSL configuration and system trust store, locate the directory with `openssl.cnf` file or symlink in `/etc`. On most distribution the file is either in `/etc/ssl` or `/etc/pki/tls`. The directory should also contain a `cert.pem` file and/or a `certs` directory.

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

2. Download, build, and install OpenSSL. Make sure you use `install_sw` and not `install`. The `install_sw` target does not override `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. Build Python with custom OpenSSL (see the configure `--with-openssl` and `--with-openssl-rpath` options)

```
$ 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
```

**Not:** Patch releases of OpenSSL have a backwards compatible ABI. You don't need to recompile Python to update OpenSSL. It's sufficient to replace the custom OpenSSL installation with a newer version.





## 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.

3.6 sürümünde geldi.

#### **--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.

3.9 sürümünde geldi.

### **--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.

3.9 sürümünde geldi.

### **--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`.

3.9 sürümünde geldi.

### **--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.

3.10 sürümünde geldi.

### 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).

3.10 sürümünde geldi.

#### **--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`;

3.6 sürümünde geldi.

### 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.

3.6 sürümünde geldi.

3.10 sürümünde değişti: 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)`.

3.8 sürümünde geldi.

#### **--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`).

3.6 sürümünde geldi.

### **--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.

3.8 sürümünde değişti: 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).

3.8 sürümünde geldi.

#### **--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.

3.6 sürümünde geldi.

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

Enable Valgrind support (default is no).

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

Enable DTrace support (default is no).

See Instrumenting CPython with DTrace and SystemTap.

3.6 sürümünde geldi.

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

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

3.6 sürümünde geldi.

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

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

3.6 sürümünde geldi.

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

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

3.6 sürümünde geldi.

### 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).

3.10 sürümünde geldi.

### 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).

3.3 sürümünde geldi.

**--with-readline=editline**

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

Define the `WITH_EDITLINE` macro.

3.10 sürümünde geldi.

**--without-readline**

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

Don't define the `HAVE_LIBREADLINE` macro.

3.10 sürümünde geldi.

**--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.

3.7 sürümünde geldi.

**--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.

3.10 sürümünde geldi.

### 3.1.8 Security Options

**--with-hash-algorithm**=[fnv|siphash24]

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

- `siphash24` (default).
- `fnv`;

3.4 sürümünde geldi.

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

Built-in hash modules:

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

3.9 sürümünde geldi.

**--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.

3.7 sürümünde geldi.

3.10 sürümünde değişti: 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.

3.6 sürümünde geldi.

#### **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**

3.4 sürümünde geldi.

#### **PY\_CPPFLAGS**

Extra preprocessor flags added for building the interpreter object files.

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

3.2 sürümünde geldi.

### 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.

3.5 sürümünde geldi.

#### **EXTRA\_CFLAGS**

Extra C compiler flags.

#### **CONFIGURE\_CFLAGS**

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

3.2 sürümünde geldi.

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

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

3.5 sürümünde geldi.

#### **BASECFLAGS**

Base compiler flags.

#### **OPT**

Optimization flags.

#### **CFLAGS\_ALIASING**

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

3.7 sürümünde geldi.

#### **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`.

3.5 sürümünde geldi.

#### **PY\_STDMODULE\_CFLAGS**

C flags used for building the interpreter object files.

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

3.7 sürümünde geldi.

#### **PY\_CORE\_CFLAGS**

Default: `$(PY_STDMODULE_CFLAGS) -DPy_BUILD_CORE`.

3.2 sürümünde geldi.

#### **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`.

3.8 sürümünde geldi.

#### **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 `C_FLAGS`, `LD_FLAGS`, etc. so users can use them on the command line to append to these values without stomping the pre-set values.

3.2 sürümünde geldi.

#### **LD\_FLAGS\_NODIST**

`LD_FLAGS_NODIST` is used in the same manner as `C_FLAGS_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 `LD_FLAGS_NODIST` variable passed to the `./configure` script.

3.8 sürümünde geldi.

#### **LD\_FLAGS**

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

Both `CPP_FLAGS` and `LD_FLAGS` 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`.

#### **LD\_SHARED**

Command to build a shared library.

Default: `@LD_SHARED@ $(PY_LD_FLAGS)`.

#### **BLD\_SHARED**

Command to build `libpython` shared library.

Default: `@BLD_SHARED@ $(PY_CORE_LD_FLAGS)`.

#### **PY\_LD\_FLAGS**

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

**PY\_LDFLAGS\_NODIST**

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

3.8 sürümünde geldi.

**PY\_CORE\_LDFLAGS**

Linker flags used for building the interpreter object files.

3.8 sürümünde geldi.



---

## 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.

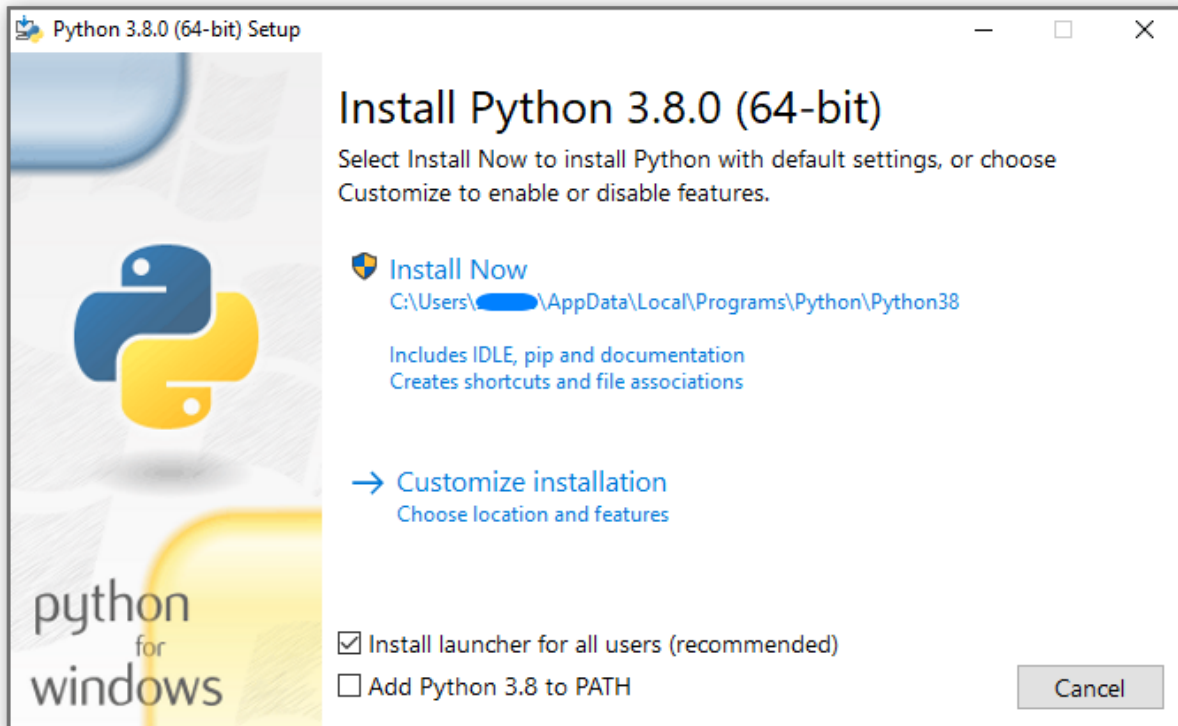
*The embeddable package* is a minimal package of Python suitable for embedding into a larger application.

## 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.

3.6 sürümünde değişti: 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
Install-LauncherAllUsers	Installs 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
Simple-Install	Disable most install UI	0

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

3.7.2 sürümünde geldi.

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.

**Uyarı:** 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.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'
```

(sonraki sayfaya devam)

(önceki sayfadan devam)

```
>>> 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

3.5.2 sürümünde geldi.

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/nugetclidl1>, 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](http://nuget.org) are [www.nuget.org/packages/python](http://www.nuget.org/packages/python) for the 64-bit version and [www.nuget.org/packages/pythonx86](http://www.nuget.org/packages/pythonx86) for the 32-bit version.

## 4.4 The embeddable package

3.5 sürümünde geldi.

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.

---

**Not:** 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).

---

**Not:** 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.

---

### Ayrıca bakınız:

<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

3.5 sürümünde değişti.

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

3.7 sürümünde geldi.

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.

---

**Not:** 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

3.3 sürümünde geldi.

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

3.6 sürümünde değişti.

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

3.5 sürümünde geldi.

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.

3.7 sürümünde geldi: 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 `pyenv.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.

3.6 sürümünde değişti:

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

3.6 sürümünden beri kullanım dışı: 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.

**Ayrıca bakınız:**

[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>



---

### Editors and IDEs

---

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

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



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

... Şunlara başvurulabilir:

- Girintili bir kod bloğu için kod girerken, eşleşen bir çift sol ve sağ sınırlayıcı (parantez, köşeli parantez, kaşlı ayraç veya üçlü tırnak) içindeyken veya bir dekoratör belirttikten sonra etkileşimli kabuğun varsayılan Python istemi.
- Elipsis yerleşik sabiti.

**2to3** Kaynağı ayrıştırarak ve ayrıştırma ağacında gezinerek tespit edilebilecek uyumsuzlukların çoğunu işleyerek Python 2.x kodunu Python 3.x koduna dönüştürmeye çalışan bir araç.

2to3, standart kitaplıkta `lib2to3`; bağımsız bir giriş noktası şu şekilde sağlanır: `file:Tools/scripts/2to3`. Bakınız `2to3-reference`.

**soyut temel sınıf** Soyut temel sınıflar *duck-typing* 'i, `hasattr()` gibi diğer teknikler beceriksiz veya tamamen yanlış olduğunda arayüzleri tanımlamanın bir yolunu sağlayarak tamamlar (örneğin sihirli yöntemlerle). ABC'ler, bir sınıftan miras almayan ancak yine de `isinstance()` ve `issubclass()` tarafından tanınan sınıflar olan sanal alt sınıfları tanıtır; `abc` modül belgelerine bakın. Python comes with many built-in ABCs for data structures (in the `collections.abc` module), numbers (in the `numbers` module), streams (in the `io` module), import finders and loaders (in the `importlib.abc` module). `abc` modülü ile kendi ABC'lerinizi oluşturabilirsiniz.

**dipnot** A label associated with a variable, a class attribute or a function parameter or return value, used by convention as a *type hint*.

Yerel değişkenlerin açıklamalarına çalışma zamanında erişilemez, ancak global değişkenlerin, sınıf niteliklerinin ve işlevlerin açıklamaları, sırasıyla modüllerin, sınıfların ve işlevlerin `__annotations__` özel özelliğinde saklanır.

Bu işlevi açıklayan *variable annotation*, *function annotation*, **PEP 484** ve **PEP 526**'e bakın. Ek açıklamalarla çalışmaya ilişkin en iyi uygulamalar için ayrıca bkz. `annotations-howto`.

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

- *keyword argument*: bir işlev çağırısında bir tanımlayıcının (ör. `ad =`) önüne geçen veya bir sözlükte `**` ile başlayan bir değer olarak geçirilen bir argüman. Örneğin, 3 ve 5, aşağıdaki `complex()`: çağrılarında anah-tar kelimenin argümanleridir:

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

- *positional argument*: anahtar kelime argümanı olmayan bir argüman. Konumsal argümanlar, bir argüman listesinin başında görünebilir ve/veya \* ile başlayan bir *iterable* öğesinin öğeleri olarak iletilebilir. Örneğin, 3 ve 5, aşağıdaki çağrılarda konumsal argümanlardır:

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

argümanlar, bir işlev gövdesindeki adlandırılmış yerel değişkenlere atanır. Bu atamayı yöneten kurallar için calls bölümüne bakın. Sözdizimsel olarak, bir argümanı temsil etmek için herhangi bir ifade kullanılabilir; değerlendirilen değer yerel değişkene atanır.

See also the *parameter* glossary entry, the FAQ question on the difference between arguments and parameters, and **PEP 362**.

**asenkron bağlam yöneticisi** *async with* ifadesinde görülen ortamı `__aenter__()` ve `__aexit__()` yöntemlerini tanımlayarak kontrol eden bir nesne. **PEP 492** de anlatıldı.

**asenkron jeneratör** *asynchronous generator iterator* döndüren bir işlev. Bir *async for* döngüsünde kullanılabilen bir dizi değer üretmek için *yield* ifadeleri içermesi dışında *async def* ile tanımlanmış bir eşyordam işlevine benziyor.

Genellikle bir asenkron üretici işlevine atıfta bulunur, ancak bazı bağlamlarda bir *asynchronous generator iterator* 'e karşılık gelebilir. Amaçlanan anlamın net olmadığı durumlarda, tam terimlerin kullanılması belirsizliği önler.

Bir asenkron üretici fonksiyonu, *await* ifadelerinin yanı sıra *async for* ve *async with* ifadeleri içerebilir.

**asenkron jeneratör yineleyici** Bir *asynchronous generator* işlevi tarafından oluşturulan bir nesne.

Bu, `__anext__()` yöntemi kullanılarak çağrıldığında, bir sonraki *yield* ifadesine kadar *asynchronous generator* işlevinin gövdesini yürütecek, beklenebilir bir nesne döndüren bir *asynchronous iterator*.

Her *yield*, konum yürütme durumunu hatırlayarak (yerel değişkenler ve bekleyen *try* ifadeleri dahil) işlemeyi geçici olarak askıya alır. *asynchronous generator iterator*, `__anext__()` tarafından döndürülen başka bir beklenebilir ile etkili bir şekilde devam ettiğinde, kaldığı yerden devam eder. Bkz. **PEP 492** ve **PEP 525**.

**eşzamansız yinelenebilir** Bir *async for* ifadesinde kullanılabilen bir nesne. `__aiter__()` yönteminden bir *asynchronous iterator* döndürmelidir. **PEP 492** 'de tanıtıldı.

**asenkron yineleyici** 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**.

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

Bir nesneye, eğer nesne izin veriyorsa, örneğin `setattr()` kullanarak, adı identifiers tarafından tanımlandığı gibi tanımlayıcı olmayan bir öznitelik vermek mümkündür. Böyle bir öznitelige noktalı bir ifade kullanılarak erişilemez ve bunun yerine `getattr()` ile alınması gerekir.

**beklenebilir** *await* ifadesinde kullanılabilen bir nesne. Bir *coroutine* veya `__await__()` yöntemine sahip bir nesne olabilir. Ayrıca bakınız **PEP 492**.

**BDFL** Benevolent Dictator For Life, namı diğer Guido van Rossum, Python'un yaratıcısı.

**ikili dosya** Bir *dosya nesnesi* *bayt benzeri nesneler* okuyabilir ve yazabilir. İkili dosya örnekleri, ikili modda açılan dosyalardır ('rb', 'wb' veya 'rb+'), `sys.stdin.buffer`, `sys.stdout.buffer` ve `io.BytesIO` ve `gzip.GzipFile` örnekleri.

Ayrıca *str* nesnelerini okuyabilen ve yazabilen bir dosya nesnesi için *text file* 'a bakın.



**borrowed reference** In Python's C API, a borrowed reference is a reference to an object, where the code using the object does not own the reference. It becomes a dangling pointer if the object is destroyed. For example, a garbage collection can remove the last *strong reference* to the object and so destroy it.

*borrowed reference* üzerinde `Py_INCREF()` çağırmak, nesnenin ödünç alınan son kullanımından önce yok edilemediği durumlar dışında, onu yerinde bir *strong reference* 'a dönüştürmek için tavsiye edilir. referans. `Py_NewRef()` işlevi, yeni bir *strong reference* oluşturmak için kullanılabilir.

**bayt benzeri nesne** `bufferobjects` 'i destekleyen ve bir *C-contiguous* arabelleğini dışa aktarabilen bir nesne. Bu, tüm `bytes`, `bytearray` ve `array.array` nesnelerinin yanı sıra birçok yaygın `memoryview` nesnesini içerir. Bayt benzeri nesneler, ikili verilerle çalışan çeşitli işlemler için kullanılabilir; bunlara sıkıştırma, ikili dosyaya kaydetme ve bir soket üzerinden gönderme dahildir.

Bazı işlemler, değişken olması için ikili verilere ihtiyaç duyar. Belgeler genellikle bunlara “okuma-yazma bayt benzeri nesneler” olarak atıfta bulunur. Örnek değiştirilebilir arabellek nesneleri `bytearray` ve bir `bytearray` `memoryview` içerir. Diğer işlemler, ikili verilerin değişmez nesnelerde (“salt okunur bayt benzeri nesneler”) depolanmasını gerektirir; bunların örnekleri arasında `bytes` ve bir `bytes` nesnesinin `memoryview` bulunur.

**bayt kodu** Python kaynak kodu, bir Python programının CPython yorumlayıcısındaki dahili temsili olan bayt kodunda derlenir. Bayt kodu ayrıca `.pyc` dosyalarında önbelleğe alınır, böylece aynı dosyanın ikinci kez çalıştırılması daha hızlı olur (kaynaktan bayt koduna yeniden derleme önlenir). Bu “ara dilin”, her bir bayt koduna karşılık gelen makine kodunu yürüten bir *sanal makine* üzerinde çalıştığı söylenir. Bayt kodlarının farklı Python sanal makineleri arasında çalışması veya Python sürümleri arasında kararlı olması beklenmediğini unutmayın.

Bayt kodu talimatlarının bir listesi `bytecodes` dokümanında bulunabilir.

**callable** Bir çağrılabilir, muhtemelen bir dizi argümanla (bkz. *argument*) ve aşağıdaki sözdizimiyle çağrılabilen bir nesnedir:

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

Bir *fonksiyon* ve uzantısı olarak bir *metot* bir çağrılabilir. `__call__()` yöntemini uygulayan bir sınıf örneği de bir çağrılabilir.

**geri çağırmak** Gelecekte bir noktada yürütülecek bir argüman olarak iletilen bir alt program işlevi.

**sınıf** Kullanıcı tanımlı nesneler oluşturmak için bir şablon. Sınıf tanımları normalde sınıfın örnekleri üzerinde çalışan yöntem tanımlarını içerir.

**sınıf değişkeni** Bir sınıfta tanımlanmış ve yalnızca sınıf düzeyinde (yani sınıfın bir örneğinde değil) değiştirilmesi amaçlanan bir değişken.

**zorlama** Aynı türden iki argüman içeren bir işlem sırasında bir tür örneğinin diğerine örtük olarak dönüştürülmesi. Örneğin, `int(3.15)`, kayan noktalı sayıyı 3 tamsayısına dönüştürür, ancak `3+4.5` 'te her argüman farklı türdedir (bir `int`, bir kayan nokta), ve her ikisi de eklenmeden önce aynı türe dönüştürülmelidir, aksi takdirde bir `TypeError` yükseltir. Zorlama olmadan, uyumlu türlerin bile tüm argümanlarının programcı tarafından aynı değere normalleştirilmesi gerekir, örneğin: `3+4, 5` yerine `float(3)+4, 5`.

**karmaşık sayı** Tüm sayıların bir reel kısım ve bir sanal kısım toplamı olarak ifade edildiği bilinen gerçek sayı sisteminin bir uzantısı. Hayali sayılar, hayali birimin gerçek katlarıdır ( $-1$  'in karekökü), genellikle matematikte  $i$  veya mühendislikte  $j$  ile yazılır. Python, bu son gösterimle yazılan karmaşık sayılar için yerleşik desteğe sahiptir; hayali kısım bir  $j$  son ekiyle yazılır, örneğin `3+1j`. `math` modülünün karmaşık eşdeğerlerine erişmek için `cmath` kullanın. Karmaşık sayıların kullanımı oldukça gelişmiş bir matematiksel özelliktir. Onlara olan ihtiyacın farkında değilseniz, onları güvenle görmezden gelebileceğiniz neredeyse kesindir.

**bağlam yöneticisi** `with` ifadesinde görülen ortamı `__enter__()` ve `__exit__()` yöntemlerini tanımlayarak kontrol eden bir nesne. Bakınız [PEP 343](#).

**bağlam değişkeni** Bağlamına bağlı olarak farklı değerler alabilen bir değişken. Bu, her yürütme iş parçasının bir değişken için farklı bir değere sahip olabileceği Thread-Local Storage'a benzer. Bununla birlikte, bağlam değiş-

kenleriyle, bir yürütme iş parçacığında birkaç bağlam olabilir ve bağlam değişkenlerinin ana kullanımı, eşzamanlı zaman uyumsuz görevlerde değişkenleri izlemektir. Bakınız `contextvars`.

**bitişik** Bir arabellek, *C-bitişik* veya *Fortran bitişik* ise tam olarak bitişik olarak kabul edilir. Sıfır boyutlu arabellekler C ve Fortran bitişiktir. Tek boyutlu dizilerde, öğeler sıfırdan başlayarak artan dizinler sırasına göre bellekte yan yana yerleştirilmelidir. Çok boyutlu C-bitişik dizilerde, öğeleri bellek adresi sırasına göre ziyaret ederken son dizin en hızlı şekilde değişir. Ancak, Fortran bitişik dizilerinde, ilk dizin en hızlı şekilde değişir.

**eşyordam** Eşyordamlar, altıyordamların daha genelleştirilmiş bir biçimidir. Alt programlara bir noktada girilir ve başka bir noktada çıkılır. Eşyordamlar birçok farklı noktada girilebilir, çıkılabilir ve devam ettirilebilir. `async def` ifadesi ile uygulanabilirler. Ayrıca bakınız **PEP 492**.

**eşyordam işlevi** Bir *coroutine* nesnesi döndüren bir işlev. Bir eşyordam işlevi `async def` ifadesiyle tanımlanabilir ve `await`, `async for` ve `async with` anahtar kelimelerini içerebilir. Bunlar **PEP 492** tarafından tanıtıldı.

**CPython** Python programlama dilinin [python.org](http://python.org) üzerinde dağıtıldığı şekilde kurallı uygulaması. “CPython” terimi, gerektiğinde bu uygulamayı Jython veya IronPython gibi diğerlerinden ayırmak için kullanılır.

**dekoratör** Genellikle `@wrapper` sözdizimi kullanılarak bir işlev dönüşümü olarak uygulanan, başka bir işlevi döndüren bir işlev. Dekoratörler için yaygın örnekler şunlardır: `classmethod()` ve `staticmethod()`.

Dekoratör sözdizimi yalnızca sözdizimsel şekerdir, aşağıdaki iki işlev tanımı anlamsal olarak eşdeğerdir:

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

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

Aynı kavram sınıflar için de mevcuttur, ancak orada daha az kullanılır. Dekoratörler hakkında daha fazla bilgi için `function definitions` ve `class definitions` belgelerine bakın.

**tanımlayıcı** `__get__()`, `__set__()` veya `__delete__()` yöntemlerini tanımlayan herhangi bir nesne. Bir sınıf özneliği bir tanımlayıcı olduğunda, öznelik araması üzerine özel bağlama davranışı tetiklenir. Normalde, bir özneliği almak, ayarlamak veya silmek için *a.b* kullanmak, *a* için sınıf sözlüğünde *b* adlı nesneyi arar, ancak *b* bir tanımlayıcı ise, ilgili tanımlayıcı yöntemi çağırılır. Tanımlayıcıları anlamak, Python’u derinlemesine anlamamanın anahtarıdır çünkü bunlar, işlevler, yöntemler, özellikler, sınıf yöntemleri, statik yöntemler ve süper sınıflara başvuru gibi birçok özelliğin temelidir.

Tanımlayıcıların yöntemleri hakkında daha fazla bilgi için, bkz. `descriptors` veya `Descriptor How To Guide`.

**sözlük** Rasgele anahtarların değerlerle eşlendiği ilişkisel bir dizi. Anahtarlar, `__hash__()` ve `__eq__()` yöntemleriyle herhangi bir nesne olabilir. Perl’de karma denir.

**sözlük anlama** Öğelerin tümünü veya bir kısmını yinelenebilir bir şekilde işlemenin ve sonuçları içeren bir sözlük döndürmenin kompakt bir yolu. `results = {n: n ** 2 for range(10)}`, `n ** 2` değerine eşlenmiş *n* anahtarını içeren bir sözlük oluşturur. Bkz. `comprehensions`.

**sözlük görünümü** `dict.keys()`, `dict.values()` ve `dict.items()` ‘den döndürülen nesnelere sözlük görünümü denir. Sözlüğün girişleri üzerinde dinamik bir görünüm sağlarlar; bu, sözlük değiştiğinde görünümün bu değişiklikleri yansıttığı anlamına gelir. Sözlük görünümünü tam liste olmaya zorlamak için `list(dictview)` kullanın. Bakınız `dict-views`.

**belge dizisi** Bir sınıf, işlev veya modülde ilk ifade olarak görünen bir dize değişmez. Paket yürütüldüğünde yoksayılırken, derleyici tarafından tanınır ve çevreleyen sınıfın, işlevin veya modülün `__doc__` özneliğine yerleştirilir. İç gözlem yoluyla erişilebilir olduğundan, nesnenin belgelenmesi için kurallı yerdir.

**duck-typing** Doğru arayüze sahip olup olmadığını belirlemek için bir nesnenin türüne bakmayan bir programlama stili; bunun yerine, yöntem veya nitelik basitçe çağrılır veya kullanılır (“Ördek gibi görüntüyorsa ve ördek gibi vaklıyorsa,

ördek olmalıdır.”) İyi tasarlanmış kod, belirli türlerden ziyade arayüzleri vurgulayarak, polimorfik ikameye izin vererek esnekliğini artırır. Ördek yazma, `type()` veya `isinstance()` kullanan testleri önler. (Ancak, ördek yazmanın *abstract base class* ile tamamlanabileceğini unutmayın.) Bunun yerine, genellikle `hasattr()` testleri veya *EAFP* programlamasını kullanır.

**EAFP** Af dilemek izin almaktan daha kolaydır. Bu yaygın Python kodlama stili, geçerli anahtarların veya niteliklerin varlığını varsayar ve varsayımın yanlış çıkması durumunda istisnaları yakalar. Bu temiz ve hızlı stil, birçok `try` ve `except` ifadesinin varlığı ile karakterize edilir. Teknik, C gibi diğer birçok dilde ortak olan *LBYL* stiliyle çelişir.

**ifade (değer döndürür)** Bir değere göre değerlendirilebilecek bir sözdizimi parçası. Başka bir deyişle, bir ifade, tümü bir değer döndüren sabit değerler, adlar, öznitelik erişimi, işleçler veya işlev çağrılarını gibi ifade öğelerinin bir toplamıdır. Diğer birçok dilin aksine, tüm dil yapıları ifade değildir. Ayrıca `while` gibi kullanılamayan *ifadeler* de vardır. Atamalar da değer döndürmeyen ifadelerdir (statement).

**uzatma modülü** Çekirdekle ve kullanıcı koduyla etkileşim kurmak için Python’un C API’sini kullanan, C veya C++ ile yazılmış bir modül.

**f-string** Ön eki `'f'` veya `'F'` olan dize değişmezleri genellikle “f-strings” olarak adlandırılır; bu, formatted string literals ‘in kısaltmasıdır. Ayrıca bkz. **PEP 498**.

**dosya nesnesi** Dosya yönelimli bir API’yi (`read()` veya `write()` gibi yöntemlerle) temel alınan bir kaynağa gösteren bir nesne. Oluşturulma şekline bağlı olarak, bir dosya nesnesi gerçek bir disk üzerindeki dosyaya veya başka bir tür depolama veya iletişim aygıtına (örneğin standart giriş/çıkış, bellek içi arabellekler, yuvalar, borular vb.) erişime aracılık edebilir. Dosya nesneleri ayrıca *file-like objects* veya *streams* olarak da adlandırılır.

Aslında üç dosya nesnesi kategorisi vardır: ham *binary files*, arabelleğe alınmış *binary files* ve *text files*. Arayüzleri `io` modülünde tanımlanmıştır. Bir dosya nesnesi yaratmanın kurallı yolu `open()` işlevini kullanmaktır.

**dosya benzeri nesne** *dosya nesnesi* ile eşanlamlıdır.

**dosya sistemi kodlaması ve hata işleyicisi** Python tarafından işletim sistemindeki baytların kodunu çözmek ve Unicode’u işletim sistemine kodlamak için kullanılan kodlama ve hata işleyici.

Dosya sistemi kodlaması, 128’in altındaki tüm baytların kodunu başarıyla çözmeyi garanti etmelidir. Dosya sistemi kodlaması bu garantiyi sağlayamazsa, API işlevleri `UnicodeError` değerini yükseltebilir.

`sys.getfilesystemencoding()` ve `sys.getfilesystemencodeerrors()` işlevleri, dosya sistemi kodlamasını ve hata işleyicisini almak için kullanılabilir.

*filesystem encoding and error handler* Python başlangıcında `PyConfig_Read()` işleviyle yapılandırılır: bkz. `filesystem_encoding` ve `filesystem_errors` üyeleri `PyConfig`.

Ayrıca bkz. *locale encoding*.

**bulucu** İçer aktarılmakta olan bir modül için *loader* ‘ı bulmaya çalışan bir nesne.

Python 3.3’ten beri, iki çeşit bulucu vardır: `sys.meta_path` ile kullanılmak üzere *meta yol bulucular*, ve `sys.path_hooks` ile kullanılmak üzere *yol girişi bulucular*.

Daha fazla ayrıntı için **PEP 302**, **PEP 420** ve **PEP 451** bakın.

**kat bölümü** En yakın tam sayıya yuvarlayan matematiksel bölme. Kat bölme operatörü `//` şeklindedir. Örneğin, `11 // 4` ifadesi, gerçek yüzer bölme tarafından döndürülen `2.75` değerinin aksine `2` olarak değerlendirilir. `(-11) // 4` ‘ün `-3` olduğuna dikkat edin, çünkü bu `-2.75` yuvarlatılmış *aşağı*. Bakınız **PEP 238**.

**fonksiyon** Bir araya bir değer döndüren bir dizi ifade. Ayrıca, gövdenin yürütülmesinde kullanılabilen sıfır veya daha fazla *argüman* iletebilir. Ayrıca *parameter*, *method* ve *function* bölümüne bakın.

**fonksiyon açıklaması** Bir işlev parametresinin veya dönüş değerinin *ek açıklaması*.

İşlev ek açıklamaları genellikle *type hints* için kullanılır: örneğin, bu fonksiyonun iki `int` argüman alması ve ayrıca bir `int` dönüş değerine sahip olması beklenir

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

İşlev açıklama sözdizimi function bölümünde açıklanmaktadır.

Bu işlevi açıklayan *variable annotation* ve **PEP 484**’e bakın. Ek açıklamalarla çalışmaya ilişkin en iyi uygulamalar için ayrıca annotations-howto konusuna bakın.

**\_\_future\_\_** Bir future ifadesi, `from __future__ import <feature>`, derleyiciyi, Python’un gelecekteki bir sürümünde standart hale gelecek olan sözdizimini veya semantiği kullanarak mevcut modülü derlemeye yönlendirir. `__future__` modülü, *feature*’ın olası değerlerini belgeler. Bu modülü içe aktararak ve değişkenlerini değerlendirek, dile ilk kez yeni bir özelliğin ne zaman eklendiğini ve ne zaman varsayılan olacağını (ya da yaptığını) görebilirsiniz:

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

**çöp toplama** Artık kullanılmadığında belleği boşaltma işlemi. Python, referans sayımı ve referans döngülerini algılayıp kırabilen bir döngüsel çöp toplayıcı aracılığıyla çöp toplama gerçekleştirir. Çöp toplayıcı `gc` modülü kullanılarak kontrol edilebilir.

**jeneratör** Bir *generator iterator* döndüren bir işlev. Bir for döngüsünde kullanılabilen bir dizi değer üretmek için `yield` ifadeleri içermesi veya `next()` işleviyle birer birer alınabilmesi dışında normal bir işleve benziyor.

Genellikle bir üretici işlevine atıfta bulunur, ancak bazı bağlamlarda bir *jeneratör yineleyicisine* atıfta bulunabilir. Amaçlanan anlamın net olmadığı durumlarda, tam terimlerin kullanılması belirsizliği önler.

**jeneratör yineleyici** Bir *generator* işlevi tarafından oluşturulan bir nesne.

Her `yield`, konum yürütme durumunu hatırlayarak (yerel değişkenler ve bekleyen `try` ifadeleri dahil) işlemeyi geçici olarak askıya alır. *jeneratör yineleyici* devam ettiğinde, kaldığı yerden devam eder (her çağrıda yeniden başlayan işlevlerin aksine).

**jeneratör ifadesi** Yineleyici döndüren bir ifade. Bir döngü değişkenini, aralığı ve isteğe bağlı bir `if` yan tümcesini tanımlayan bir `for` yan tümcesinin takip ettiği normal bir ifadeye benziyor. Birleştirilmiş ifade, bir çevreleyen için değerler üretir:

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

**genel işlev** Farklı türler için aynı işlemi uygulayan birden çok işlevden oluşan bir işlev. Bir çağrı sırasında hangi uygulamanın kullanılması gerektiği, gönderme algoritması tarafından belirlenir.

Ayrıca *single dispatch* sözlük girdisine, `functools singledispatch()` dekoratörüne ve **PEP 443**’e bakın.

**genel tip** Parametrelendirilebilen bir *type*; tipik olarak bir konteyner sınıfı, örneğin `list` veya `dict`. *type hint* ve *annotation* için kullanılır.

Daha fazla ayrıntı için generic alias types, **PEP 483**, **PEP 484**, **PEP 585** ve `typing` modülüne bakın.

**GIL** Bakınız *global interpreter lock*.

**genel tercüman kilidi** *CPython* yorumlayıcısı tarafından aynı anda yalnızca bir iş parçacığının Python *bytecode*’u yürütmesini sağlamak için kullanılan mekanizma. Bu, nesne modelini (`dict` gibi kritik yerleşik türler dahil) eşzamanlı erişime karşı örtük olarak güvenli hale getirerek *CPython* uygulamasını basitleştirir. Tüm yorumlayıcıyı kilitlemek, çok işlemcili makinelerin sağladığı paralelliğin çoğu pahasına, yorumlayıcının çok iş parçacıklı olmasını kolaylaştırır.

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

“Serbest iş parçacıklı” bir yorumlayıcı (paylaşılan verileri çok daha ince bir ayrıntı düzeyinde kilitleyen) oluşturma çabaları, ortak tek işlemcili durumda performans düştüğü için başarılı olmamıştır. Bu performans sorununun üstesinden gelinmesinin uygulamayı çok daha karmaşık hale getireceğine ve dolayısıyla bakımını daha maliyetli hale getireceğine inanılmaktadır.

**karma tabanlı pyc** Geçerliliğini belirlemek için ilgili kaynak dosyanın son değiştirilme zamanı yerine karma değerini kullanan bir bayt kodu önbellek dosyası. Bakınız `pyc-invalidation`.

**yıkanabilir** Bir nesne, ömrü boyunca asla değişmeyen bir karma değere sahipse (bir `__hash__()` yöntemine ihtiyaç duyar) ve diğer nesnelerle karşılaştırılabilirse (bir `__eq__()` yöntemine ihtiyaç duyar) *hashable* olur. . Eşit karşılaştıran Hashable nesneleri aynı karma değerine sahip olmalıdır.

Hashability, bir nesneyi bir sözlük anahtarı ve bir set üyesi olarak kullanılabilir hale getirir, çünkü bu veri yapıları hash değerini dahili olarak kullanır.

Python’un değişmez yerleşik nesnelerinin çoğu, yıkanabilir; değiştirilebilir kaplar (listeler veya sözlükler gibi) değildir; değişmez kaplar (tüpler ve donmuş kümeler gibi) yalnızca öğelerinin yıkanabilir olması durumunda yıkanabilir. Kullanıcı tanımlı sınıfların örnekleri olan nesneler varsayılan olarak hash edilebilir. Hepsisi eşit olmayı karşılaştırır (kendileriyle hariç) ve hash değerleri `id()` ‘lerinden türetilir.

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

**değişmez** Sabit değeri olan bir nesne. Değişmez nesneler arasında sayılar, dizeler ve demetler bulunur. Böyle bir nesne değiştirilemez. Farklı bir değerin saklanması gerekiyorsa yeni bir nesne oluşturulmalıdır. Örneğin bir sözlükte anahtar olarak, sabit bir karma değerinin gerekli olduğu yerlerde önemli bir rol oynarlar.

**içe aktarım yolu** İçe aktarılacak modüller için *path based finder* tarafından aranan konumların (veya *path entries*) listesi. İçe aktarma sırasında, bu konum listesi genellikle `sys.path` adresinden gelir, ancak alt paketler için üst paketin `__path__` özelliğinden de gelebilir.

**içe aktarma** Bir modüldeki Python kodunun başka bir modüldeki Python koduna sunulması süreci.

**içe aktarıcı** Bir modülü hem bulan hem de yükleyen bir nesne; hem bir *finder* hem de *loader* nesnesi.

**etkileşimli** Python’un etkileşimli bir yorumlayıcısı vardır; bu, yorumlayıcı isteminde ifadeler ve ifadeler girebileceğiniz, bunları hemen çalıştırabileceğiniz ve sonuçlarını görebileceğiniz anlamına gelir. Herhangi bir argüman olmadan `python` ‘u başlatmanız yeterlidir (muhtemelen bilgisayarınızın ana menüsünden seçerek). Yeni fikirleri test etmenin veya modülleri ve paketleri incelemenin çok güçlü bir yoludur (`help(x)` ‘i unutmayın).

**yorumlanmış** Python, derlenmiş bir dilin aksine yorumlanmış bir dildir, ancak bayt kodu derleyicisinin varlığı nedeniyle ayrım bulanık olabilir. Bu, kaynak dosyaların daha sonra çalıştırılacak bir yürütülebilir dosya oluşturmadan doğrudan çalıştırılabileceği anlamına gelir. Yorumlanan diller genellikle derlenmiş dillerden daha kısa bir geliştirme/hata ayıklama döngüsüne sahiptir, ancak programları genellikle daha yavaş çalışır. Ayrıca bkz. *interactive*.

**tercüman kapatma** Kapatılması istendiğinde, Python yorumlayıcısı, modüller ve çeşitli kritik iç yapılar gibi tahsis edilen tüm kaynakları kademeli olarak serbest bıraktığı özel bir aşamaya girer. Ayrıca *garbage collector* için birkaç çağrı yapar. Bu, kullanıcı tanımlı yıkıcılarda veya zayıf referans geri aramalarında kodun yürütülmesini tetikleyebilir. Kapatma aşamasında yürütülen kod, dayandığı kaynaklar artık çalışmayabileceğinden çeşitli istisnalarla karşılaşabilir (yaygın örnekler kitaplık modülleri veya uyarı makineleridir).

Yorumlayıcının kapatılmasının ana nedeni, `__main__` modülünün veya çalıştırılan betiğin yürütmeyi bitirmiş olmasıdır.

**yinelenebilir** Üyelerini teker teker döndürebilen bir nesne. Yineleme örnekleri, tüm dizi türlerini (`list`, `str`, and `tuple` gibi) ve `dict`, *dosya objeleri* gibi bazı dizi olmayan türleri ve bir `__iter__()` yöntemiyle veya *dizi* semantiğini uygulayan bir `__getitem__()` yöntemiyle tanımladığınız tüm sınıfların nesnelerini içerir.

Yinelenebilirler bir `for` döngüsünde ve bir dizinin gerekli olduğu diğer birçok yerde kullanılabilir (`zip()`, `map()`, ...). Yerleşik `iter()` işlevine argüman olarak yinelenebilir bir nesne iletildiğinde, nesne için bir yineleyici döndürür. Bu yineleyici, değerler kümesi üzerinden bir geçiş için iyidir. Yinelenebilirleri kullanırken,

genellikle `iter()` çağırmanız veya yineleyici nesnelerle kendiniz ilgilenmeniz gerekmez. `for` ifadesi bunu sizin için otomatik olarak yapar ve yineleyiciyi döngü süresince tutmak için geçici bir adsız değişken oluşturur. Ayrıca bkz. *iterator*, *sequence* ve *generator*.

**yineleyici** Bir veri akışını temsil eden bir nesne. Yineleyicinin `__next__()` yöntemine (veya yerleşik `next()` işlevine iletilmesi) yinelenen çağrılar, akıştaki ardışık öğeleri döndürür. Daha fazla veri bulunmadığında, bunun yerine bir `StopIteration` istisnası oluşturulur. Bu noktada, yineleyici nesnesi tükenir ve `__next__()` yöntemine yapılan diğer çağrılar yalnızca `StopIteration` öğesini yeniden yükseltir. Yineleyicilerin, yineleyici nesnesinin kendisini döndüren bir `__iter__()` yöntemine sahip olmaları gerekir, böylece her yineleyici de yinelenabilir ve diğer yinelenebilirlerin kabul edildiği çoğu yerde kullanılabilir. Dikkate değer bir istisna, birden çok yineleme geçişini deneyen koddur. Bir kapsayıcı nesnesi (örneğin bir `list`), onu `iter()` işlevine her ilettiğinizde veya onu bir `for` döngüsünde kullandığınızda yeni bir yineleyici üretir. Bunu bir yineleyiciyle denemek, önceki yineleme geçişinde kullanılan aynı tükenmiş yineleyici nesnesini döndürerek boş bir kap gibi görünmesini sağlar.

Daha fazla bilgi `typeiter` içinde bulunabilir.

**CPython uygulama ayrıntısı:** CPython, bir yineleyicinin `__iter__()` tanımlaması gereksinimini tutarlı bir şekilde uygulamaz.

**anahtar işlev** Anahtar işlevi veya harmanlama işlevi, sıralama veya sıralama için kullanılan bir değeri döndüren bir çağrılabilir. Örneğin, `locale.strxfrm()`, yerel ayara özgü sıralama kurallarının farkında olan bir sıralama anahtarı üretmek için kullanılır.

Python'daki bir dizi araç, öğelerin nasıl sıralandığını veya gruptandırıldığını kontrol etmek için temel işlevleri kabul eder. Bunlar `min()`, `max()`, `sorted()`, `list.sort()`, `heapq.merge()`, `heapq.nsmallest()`, `heapq.nlargest()` ve `itertools.groupby()`.

Bir tuş işlevi oluşturmanın birkaç yolu vardır. Örneğin, `str.lower()` yöntemi, büyük/küçük harfe duyarlı olmayan sıralamalar için bir anahtar işlev işlevi görebilir. Alternatif olarak, `lambda r: (r[0], r[2])` gibi bir `lambda` ifadesinden bir anahtar işlevi oluşturulabilir. Ayrıca, `operator` modülü üç temel işlev kurucusu sağlar: `attrgetter()`, `itemgetter()` ve `methodcaller()`. Anahtar işlevlerin nasıl oluşturulacağı ve kullanılacağına ilişkin örnekler için *Sorting HOW TO* bölümüne bakın.

**anahtar kelime argümanı** Bakınız *argument*.

**lambda** İşlev çağrıldığında değerlendirilen tek bir *expression* 'dan oluşan anonim bir satır içi işlev. Bir `lambda` işlevi oluşturmak için sözdizimi `lambda [parametreler]: ifade` şeklindedir

**LBYL** Zıplamadan önce Bak. Bu kodlama stili, arama veya arama yapmadan önce ön koşulları açıkça test eder. Bu stil, *EAFP* yaklaşımıyla çelişir ve birçok `if` ifadesinin varlığı ile karakterize edilir.

Çok iş parçacıklı bir ortamda, LBYL yaklaşımı "bakan" ve "sıçrayan" arasında bir yarış koşulu getirme riskini taşıyabilir. Örneğin, `if key in mapping: return mapping[key]` kodu, testten sonra, ancak aramadan önce başka bir iş parçacığı *eşlemeden key* kaldırırsa başarısız olabilir. Bu sorun, kilitlerle veya *EAFP* yaklaşımı kullanılarak çözülebilir.

**yerel kodlama** Unix'te, `LC_CTYPE` yerel ayarının kodlamasıdır. `locale.setlocale(locale.LC_CTYPE, new_locale)` ile ayarlanabilir.

Windows'ta bu, ANSI kod sayfasıdır (ör. `cp1252`).

`locale.getpreferredencoding(False)` yerel ayar kodlamasını almak için kullanılabilir.

Python, Unicode dosya adları ile bayt dosya adları arasında dönüştürme yapmak için *filesystem encoding and error handler* kullanır.

**liste** Yerleşik bir Python *dizi*. Adına rağmen, öğelere erişim  $O(1)$  olduğundan, diğer dillerdeki bir diziye, bağlantılı bir listeden daha yakındır.

**liste anlama** Bir dizideki öğelerin tümünü veya bir kısmını işlemenin ve sonuçları içeren bir liste döndürmenin kompakt bir yolu. `sonuç = ['{:04x}'.format(x) for range(256) if x % 2 == 0]`, dizinde



çift onaltılık sayılar (0x..) içeren bir diziler listesi oluşturur. 0 ile 255 arasındadır. `if` yan tümcesi isteğe bağlıdır. Atlanırsa, “aralık(256)” içindeki tüm öğeler işlenir.

**yükleyici** Modül yükleyen bir nesne. `load_module()` adında bir yöntem tanımlanmalıdır. Bir yükleyici genellikle bir *finder* ile döndürülür. Ayrıntılar için [PEP 302](#) ve bir *soyut temel sınıf* için `importlib.abc.Loader` bölümüne bakın.

**sihirli yöntem** *special method* için gayri resmi bir eşanlamlı.

**haritalama** Keyfi anahtar aramalarını destekleyen ve Mapping veya MutableMapping collections-abstract-base-classes içinde belirtilen yöntemleri uygulayan bir kapsayıcı nesnesi temel sınıflar. Örnekler arasında `dict`, `collections.defaultdict`, `collections.OrderedDict` ve `collections.Counter` sayılabilir.

**meta yol bulucu** Bir *finder*, `sys.meta_path` aramasıyla döndürülür. Meta yol bulucular, *yol girişi bulucuları* ile ilişkilidir, ancak onlardan farklıdır.

Meta yol bulucuların uyguladığı yöntemler için `importlib.abc.MetaPathFinder` bölümüne bakın.

**metasınıf** Bir sınıfın sınıfı. Sınıf tanımları, bir sınıf adı, bir sınıf sözlüğü ve temel sınıfların bir listesini oluşturur. Metasınıf, bu üç argümanı almaktan ve sınıfı oluşturmaktan sorumludur. Çoğu nesne yönelimli programlama dili, varsayılan bir uygulama sağlar. Python’u özel yapan şey, özel metasınıflar oluşturma mümkün olmasıdır. Çoğu kullanıcı bu araca hiçbir zaman ihtiyaç duymaz, ancak ihtiyaç duyulduğunda, metasınıflar güçlü ve zarif çözümler sağlayabilir. Nitelik erişimini günlüğe kaydetmek, iş parçacığı güvenliği eklemek, nesne oluşturmaya izlemek, tekilleri uygulamak ve diğer birçok görev için kullanılmışlardır.

Daha fazla bilgi metaclasses içinde bulunabilir.

**metot** Bir sınıf gövdesi içinde tanımlanan bir işlev. Bu sınıfın bir örneğinin özniteliği olarak çağrılırsa, yöntem örnek nesnesini ilk *argument* (genellikle `self` olarak adlandırılır) olarak alır. Bkz. *function* ve *nested scope*.

**metot kalite sıralaması** Metot Çözüm Sırası, arama sırasında bir üye için temel sınıfların arandığı sıradır. 2.3 sürümünden bu yana Python yorumlayıcısı tarafından kullanılan algoritmanın ayrıntıları için bkz. [The Python 2.3 Method Resolution Order](#)

**modül** Python kodunun kuruluş birimi olarak hizmet eden bir nesne. Modüller, rastgele Python nesneleri içeren bir ad alanına sahiptir. Modüller, *importing* işlemiyle Python’a yüklenir.

Ayrıca bakınız *package*.

**modül özelliği** Bir modülü yüklemek için kullanılan içe aktarmayla ilgili bilgileri içeren bir ad alanı. Bir `importlib.machinery.ModuleSpec` örneği.

**MRO** Bakınız *metot çözüm sırası*.

**değiştirilebilir** Değiştirilebilir (mutable) nesneler değerlerini değiştirebilir ancak idlerini koruyabilirler. Ayrıca bkz. *immutable*.

**adlandırılmış demet** “named tuple” terimi, demetten miras alan ve dizinlenebilir öğelerine de adlandırılmış nitelikler kullanılarak erişilebilen herhangi bir tür veya sınıf için geçerlidir. Tür veya sınıfın başka özellikleri de olabilir.

Çeşitli yerleşik türler, `time.localtime()` ve `os.stat()` tarafından döndürülen değerler de dahil olmak üzere, tanımlama grupları olarak adlandırılır. Başka bir örnek `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
```

Bazı adlandırılmış demetler yerleşik türlerdir (yukarıdaki örnekler gibi). Alternatif olarak, `tuple` ögesinden miras alan ve adlandırılmış alanları tanımlayan normal bir sınıf tanımından adlandırılmış bir tanımlama grubu oluşturulabilir. Böyle bir sınıf elle yazılabilir veya fabrika işlevi `collections.namedtuple()` ile oluşturulabilir. İkinci teknik ayrıca elle yazılmış veya yerleşik adlandırılmış demetlerde bulunmayan bazı ekstra yöntemler ekler.

**ad alanı** Değişkenin saklandığı yer. Ad alanları sözlükler olarak uygulanır. Nesnelerde (yöntemlerde) yerel, genel ve yerleşik ad alanlarının yanı sıra iç içe ad alanları vardır. Ad alanları, adlandırma çakışmalarını önleyerek modülerliği destekler. Örneğin, `builtins.open` ve `os.open()` işlevleri ad alanlarıyla ayırt edilir. Ad alanları, hangi modülün bir işlevi uyguladığını açıkça belirterek okunabilirliğe ve sürdürülebilirliğe de yardımcı olur. Örneğin, `random.seed()` veya `itertools.islice()` yazmak, bu işlevlerin sırasıyla `random` ve `itertools` modülleri tarafından uygulandığını açıkça gösterir.

**ad alanı paketi** A **PEP 420** *package*, yalnızca alt paketler için bir kap olarak hizmet eder. Ad alanı paketlerinin hiçbir fiziksel temsili olmayabilir ve `__init__.py` dosyası olmadığından özellikle *regular package* gibi değildirler.

Ayrıca bkz. *module*.

**iç içe kapsam** Kapsamlı bir tanımdaki bir değişkene atıfta bulunma yeteneği. Örneğin, başka bir fonksiyonun içinde tanımlanan bir fonksiyon, dış fonksiyondaki değişkenlere atıfta bulunabilir. İç içe kapsamların varsayılan olarak yalnızca başvuru için çalıştığını ve atama için çalışmadığını unutmayın. Yerel değişkenler en içteki kapsamda hem okur hem de yazar. Benzer şekilde, global değişkenler global ad alanını okur ve yazar. `nonlocal`, dış kapsamlara yazmaya izin verir.

**yeni stil sınıf** Artık tüm sınıf nesneleri için kullanılan sınıfların lezzetinin eski adı. Önceki Python sürümlerinde, yalnızca yeni stil sınıfları Python'un `__slots__`, tanımlayıcılar, özellikler, `__getattr__()`, sınıf yöntemleri ve statik yöntemler gibi daha yeni, çok yönlü özelliklerini kullanabilirdi.

**obje** Durum (öznitelikler veya değer) ve tanımlanmış davranış (yöntemler) içeren herhangi bir veri. Ayrıca herhangi bir *yeni tarz sınıfın* nihai temel sınıfı.

**paket** A Python *module* which can contain submodules or recursively, subpackages. Technically, a package is a Python module with a `__path__` attribute.

Ayrıca bkz. *regular package* ve *namespace package*.

**parametre** Bir *function* (veya yöntem) tanımında, işlevin kabul edebileceği bir *argument* (veya bazı durumlarda, argümanlar) belirten adlandırılmış bir varlık. Beş çeşit parametre vardır:

- *positional-or-keyword*: *pozisyonel* veya bir *keyword argümanı* olarak iletilebilen bir argüman belirtir. Bu, varsayılan parametre türüdür, örneğin aşağıdakilerde *foo* ve *bar*:

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

- *positional-only*: yalnızca konuma göre sağlanabilen bir argüman belirtir. Yalnızca konumsal parametreler, onlardan sonra fonksiyon tanımının parametre listesine bir `/` karakteri eklenerek tanımlanabilir, örneğin aşağıdakilerde *posonly1* ve *posonly2*:

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

- *keyword-only*: sadece anahtar kelime ile sağlanabilen bir argüman belirtir. Yalnızca anahtar kelime (keyword-only) parametreleri, onlardan önceki fonksiyon tanımının parametre listesine tek bir değişken konumlu parametre veya çıplak `*` dahil edilerek tanımlanabilir, örneğin aşağıdakilerde *kw\_only1* ve *kw\_only2*:

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

- *var-positional*: keyfi bir pozisyonel argüman dizisinin sağlanabileceğini belirtir (diğer parametreler tarafından zaten kabul edilmiş herhangi bir konumsal argümana ek olarak). Böyle bir parametre, parametre adının başına `*` eklenerek tanımlanabilir, örneğin aşağıdakilerde *args*:



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

- *var-keyword*: keyfi olarak birçok anahtar kelime argümanının sağlanabileceğini belirtir (diğer parametreler tarafından zaten kabul edilen herhangi bir anahtar kelime argümanına ek olarak). Böyle bir parametre, parametre adının başına *\*\**, örneğin yukarıdaki örnekte *kwargs* eklenerek tanımlanabilir.

Parametreler, hem isteğe bağlı hem de gerekli argümanları ve ayrıca bazı isteğe bağlı bağımsız değişkenler için varsayılan değerleri belirtebilir.

Ayrıca bkz. *argüman*, argümanlar ve parametreler arasındaki fark, `inspect.Parameter`, `function` ve **PEP 362**.

**yol girişi** *path based finder* içe aktarma modüllerini bulmak için başvurduğu *import path* üzerindeki tek bir konum.

**yol girişi bulucu** Bir *finder* `sys.path_hooks` (yani bir *yol girişi kancası*) üzerinde bir çağrılabilir tarafından döndürülür ve *path entry* verilen modüllerin nasıl bulunacağını bilir.

Yol girişi bulucularının uyguladığı yöntemler için `importlib.abc.PathEntryFinder` bölümüne bakın.

**yol giriş kancası** `sys.path_hook` listesinde, belirli bir *yol girişindeki* modülleri nasıl bulacağını biliyorsa, bir *yol girişi bulucu* döndüren bir çağrılabilir.

**yol tabanlı bulucu** Modüller için bir *import path* arayan varsayılan *meta yol buluculardan* biri.

**yol benzeri nesne** Bir dosya sistemi yolunu temsil eden bir nesne. Yol benzeri bir nesne, bir yolu temsil eden bir `str` veya `bytes` nesnesi veya `os.PathLike` protokolünü uygulayan bir nesnedir. `os.PathLike` protokolünü destekleyen bir nesne, `os.fspath()` işlevi çağrılarak bir `str` veya `bytes` dosya sistemi yoluna dönüştürülebilir; `os.fsdecode()` ve `os.fsencode()`, bunun yerine sırasıyla `str` veya `bytes` sonucunu garanti etmek için kullanılabilir. **PEP 519** tarafından tanıtıldı.

**PEP** Python Geliştirme Önerisi. PEP, Python topluluğuna bilgi sağlayan veya Python veya süreçleri ya da ortamı için yeni bir özelliği açıklayan bir tasarım belgesidir. PEP'ler, önerilen özellikler için özlü bir teknik şartname ve bir gerekçe sağlamalıdır.

PEP'lerin, önemli yeni özellikler önermek, bir sorun hakkında topluluk girdisi toplamak ve Python'a giren tasarım kararlarını belgelemek için birincil mekanizmalar olması amaçlanmıştır. PEP yazarı, topluluk içinde fikir birliği oluşturmaktan ve muhalif görüşleri belgelemekten sorumludur.

Bakınız **PEP 1**.

**kısım** **PEP 420** içinde tanımlandığı gibi, bir ad alanı paketine katkıda bulunan tek bir dizindeki (muhtemelen bir zip dosyasında depolanan) bir dizi dosya.

**konumsal argüman** Bakınız *argument*.

**geçici API** Geçici bir API, standart kitaplığın geriye dönük uyumluluk garantilerinden kasıtlı olarak hariç tutulan bir API'dir. Bu tür arayüzlerde büyük değişiklikler beklenmese de, geçici olarak işaretlendikleri süreç, çekirdek geliştiriciler tarafından gerekli görüldüğü takdirde geriye dönük uyumsuz değişiklikler (arayüzün kaldırılmasına kadar ve buna kadar) meydana gelebilir. Bu tür değişiklikler karşılıksız yapılmayacaktır - bunlar yalnızca API'nin eklenmesinden önce gözden kaçan ciddi temel kusurlar ortaya çıkarsa gerçekleşecektir.

Geçici API'ler için bile, geriye dönük uyumsuz değişiklikler "son çare çözümü" olarak görülür - tanımlanan herhangi bir soruna geriye dönük uyumlu bir çözüm bulmak için her türlü girişimde bulunulacaktır.

Bu süreç, standart kitaplığın, uzun süreler boyunca sorunlu tasarım hatalarına kilitlenmeden zaman içinde gelişmeye devam etmesini sağlar. Daha fazla ayrıntı için bkz. **PEP 411**.

**geçici paket** Bakınız *provisional API*.

**Python 3000** Python 3.x sürüm satırının takma adı (uzun zaman önce sürüm 3'ün piyasaya sürülmesi uzak bir gelecekte olduğu zaman ortaya çıktı.) Bu aynı zamanda "Py3k" olarak da kısaltılır.

**Pythonic** Diğer dillerde ortak kavramları kullanarak kod uygulamak yerine Python dilinin en yaygın deyimlerini yakından takip eden bir fikir veya kod parçası. Örneğin, Python’da yaygın bir deyim, bir `for` ifadesi kullanarak yinelenen bir öğenin tüm öğeleri üzerinde döngü oluşturmaktır. Diğer birçok dilde bu tür bir yapı yoktur, bu nedenle Python’a aşina olmayan kişiler bazen bunun yerine sayısal bir sayaç kullanır:

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

Temizleyicinin aksine, Pythonic yöntemi:

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

**nitelikli isim** **PEP 3155** içinde tanımlandığı gibi, bir modülün genel kapsamından o modülde tanımlanan bir sınıfa, işleve veya yonteme giden “yolu” gösteren noktalı ad. Üst düzey işlevler ve sınıflar için nitelikli ad, nesnenin adıyla aynıdır:

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

Modüllere atıfta bulunmak için kullanıldığında, *tam nitelenmiş ad*, herhangi bir üst paket de dahil olmak üzere, modüle giden tüm noktalı yol anlamına gelir, örn. `email.mime.text`:

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

**referans sayısı** Bir nesneye yapılan başvuruların sayısı. Bir nesnenin referans sayısı sıfıra düştüğünde, yerinden çıkarılır. Referans sayımı genellikle Python kodunda görülmez, ancak *CPython* uygulamasının önemli bir özgesi-dir. `sys` modülü, programcıların belirli bir nesne için referans sayısını döndürmek üzere çağırabilecekleri bir `getrefcount()` işlevini tanımlar.

**sürekli paketleme** `__init__.py` dosyası içeren bir dizin gibi geleneksel bir *package*.

Ayrıca bkz. *ad alanı paketi*.

**\_\_slots\_\_** Örnek öznitelikleri için önceden yer bildirerek ve örnek sözlüklerini ortadan kaldırarak bellekten tasarruf sağlayan bir sınıf içindeki bildirim. Popüler olmasına rağmen, tekniğin doğru olması biraz zor ve en iyi, bellek açısından kritik bir uygulamada çok sayıda örneğin bulunduğu nadir durumlar için ayrılmıştır.

**dizi** `__getitem__()` özel yöntemi aracılığıyla tamsayı dizinlerini kullanarak verimli öğe erişimini destekleyen ve dizinin uzunluğunu döndüren bir `__len__()` yöntemini tanımlayan bir *iterable*. Bazı yerleşik dizi türleri şunlardır: `list`, `str`, `tuple` ve `bytes`. `dict` ayrıca `__getitem__()` ve `__len__()`’i de desteklediğine dikkat edin, ancak aramalar tamsayılar yerine rastgele *immutable* anahtarları kullandığından bir diziden ziyade bir eşleme olarak kabul edilir.

`collections.abc.Sequence` soyut temel sınıfı; `count()`, `index()`, `__contains__()`, ve `__reversed__()` ekleyerek sadece `__getitem__()` ve `__len__()`’in ötesine geçen çok daha zengin bir arayüzü tanımlar. Bu genişletilmiş arabirimi uygulayan türler, `register()` kullanılarak açıkça kaydedilebilir.

**anlamak** Öğelerin tümünü veya bir kısmını yinelenebilir bir şekilde işlemenin ve sonuçlarla birlikte bir küme döndürmenin kompakt bir yolu. `results = {c for c in 'abracadabra' if c not in 'abc'}, {'r', 'd'}` dizelerini oluşturur. Bakınız `comprehensions`.

**tek sevk** Uygulamanın tek bir argüman türüne göre seçildiği bir *generic function* gönderimi biçimi.

**parçalamak** Genellikle bir *sequence* 'nin bir bölümünü içeren bir nesne. Bir dilim, örneğin `variable_name[1:3:5]` 'de olduğu gibi, birkaç tane verildiğinde, sayılar arasında iki nokta üst üste koyarak, `[]` alt simge gösterimi kullanılarak oluşturulur. Köşeli ayraç (alt simge) gösterimi, dahili olarak `slice` nesnelerini kullanır.

**özel metod** Toplama gibi bir tür üzerinde belirli bir işlemi yürütmek için Python tarafından örtük olarak çağrılan bir yöntem. Bu tür yöntemlerin çift alt çizgi ile başlayan ve biten adları vardır. Özel yöntemler `specialnames` içinde belgelenmiştir.

**ifade (değer döndürmez)** Bir ifade, bir paketin parçasıdır (kod “bloğu”). Bir ifade, bir *expression* veya `if`, `while` veya `for` gibi bir anahtar kelimeye sahip birkaç yapıdan biridir.

**güçlü referans** In Python's C API, a strong reference is a reference to an object which is owned by the code holding the reference. The strong reference is taken by calling `Py_INCREF()` when the reference is created and released with `Py_DECREF()` when the reference is deleted.

`Py_NewRef()` fonksiyonu, bir nesneye güçlü bir başvuru oluşturmak için kullanılabilir. Genellikle `Py_DECREF()` fonksiyonu, bir referansın sızmasını önlemek için güçlü referans kapsamından çıkmadan önce güçlü referansta çağrılmalıdır.

Ayrıca bkz. *ödüncü alınan referans*.

**yazı çözümleme** Python'da bir dize, bir Unicode kod noktaları dizisidir (U+0000–U+10FFFF aralığında). Bir dizeyi depolamak veya aktarmak için, bir bayt dizisi olarak seri hale getirilmesi gerekir.

Bir dizeyi bir bayt dizisi halinde seri hale getirmek “kodlama (encoding)” olarak bilinir ve dizeyi bayt dizisinden yeniden oluşturmak “kod çözme (decoding)” olarak bilinir.

Toplu olarak “metin kodlamaları” olarak adlandırılan çeşitli farklı metin serileştirme kodekleri vardır.

**yazı dosyası** A *file object* `str` nesnelerini okuyabilir ve yazabilir. Çoğu zaman, bir metin dosyası aslında bir bayt yönelimli veri akışına erişir ve otomatik olarak *text encoding* işler. Metin dosyalarına örnek olarak metin modunda açılan dosyalar ('r' veya 'w'), `sys.stdin`, `sys.stdout` ve `io.StringIO` örnekleri verilebilir.

Ayrıca *ikili dosyaları* okuyabilen ve yazabilen bir dosya nesnesi için *bayt benzeri nesnelere* bakın.

**üç tırnaklı dize** Üç tırnak işareti (") veya kesme işareti (') ile sınırlanan bir dize. Tek tırnaklı dizelerde bulunmayan herhangi bir işlevsellik sağlamasalar da, birkaç nedenden dolayı faydalıdırlar. bir dizeye çıkışsız tek ve çift tırnak eklemeniz gerekir ve bunlar, devam karakterini kullanmadan birden çok satıra yayılabilir, bu da onları özellikle belge dizileri yazarken kullanışlı hale getirir.

**tip** Bir Python nesnesinin türü, onun ne tür bir nesne olduğunu belirler; her nesnenin bir türü vardır. Bir nesnenin tipine `__class__` niteliği ile erişilebilir veya `type(obj)` ile alınabilir.

**tip takma adı** Bir tanımlayıcıya tür atanarak oluşturulan, bir tür için eş anlamlı.

Tür takma adları, *tür ipuçlarını* basitleştirmek için kullanışlıdır. Örneğin:

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

bu şekilde daha okunaklı hale getirilebilir:

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

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

Bu işlevi açıklayan `typing` ve **PEP 484** bölümlerine bakın.

**tür ipucu** Bir değişken, bir sınıf niteliği veya bir işlev parametresi veya dönüş değeri için beklenen türü belirten bir *ek açıklama*.

Tür ipuçları isteğe bağlıdır ve Python tarafından uygulanmaz, ancak bunlar statik tip analiz araçları için faydalıdır ve kod tamamlama ve yeniden düzenleme ile IDE'lere yardımcı olur.

Genel değişkenlerin, sınıf özniteliklerinin ve işlevlerin tür ipuçlarına, yerel değişkenlere değil, `typing.get_type_hints()` kullanılarak erişilebilir.

Bu işlevi açıklayan `typing` ve **PEP 484** bölümlerine bakın.

**evrensel yeni satırlar** Aşağıdakilerin tümünün bir satırın bitişi olarak kabul edildiği metin akışlarını yorumlamanın bir yolu: Unix satır sonu kuralı `\n`, Windows kuralı `\r\n`, ve eski Macintosh kuralı `\r`. Ek bir kullanım için **PEP 278** ve **PEP 3116** ve ayrıca `bytes.splitlines()` bakın.

**değişken açıklama** Bir değişkenin veya bir sınıf özniteliğinin *ek açıklaması*.

Bir değişkene veya sınıf niteliğine açıklama eklerken atama isteğe bağlıdır:

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

Değişken açıklamaları genellikle *tür ipuçları* için kullanılır: örneğin, bu değişkenin `int` değerlerini alması beklenir:

```
count: int = 0
```

Değişken açıklama sözdizimi `annassign` bölümünde açıklanmıştır.

Bu işlevi açıklayan; *function annotation*, **PEP 484** ve **PEP 526** bölümlerine bakın. Ek açıklamalarla çalışmaya ilişkin en iyi uygulamalar için ayrıca bkz. `annotations-howto`.

**sanal ortam** Python kullanıcılarının ve uygulamalarının, aynı sistem üzerinde çalışan diğer Python uygulamalarının davranışına müdahale etmeden Python dağıtım paketlerini kurmasına ve yükseltmesine olanak tanıyan, işbirliği içinde yalıtılmış bir çalışma zamanı ortamı.

Ayrıca bakınız `venv`.

**sanal makine** Tamamen yazılımla tanımlanmış bir bilgisayar. Python'un sanal makinesi, bayt kodu derleyicisi tarafından yayınlanan *bytecode* 'u çalıştırır.

**Python'un Zen'i** Dili anlamaya ve kullanmaya yardımcı olan Python tasarım ilkeleri ve felsefelerinin listesi. Liste, etkileşimli komut isteminde `import this` yazarak bulunabilir.

---

## Dokümanlar hakkında

---

Bu dokümanlar, Python dokümanları için özel olarak yazılmış bir doküman işlemcisi olan [Sphinx](#) tarafından [reStructuredText](#) kaynaklarından oluşturulur.

Dokümantasyonun ve araç zincirinin geliştirilmesi, tıpkı Python'un kendisi gibi tamamen gönüllü bir çabadır. Katkıda bulunmak istiyorsanız, nasıl yapacağınıza ilişkin bilgi için lütfen [reporting-bugs](#) sayfasına göz atın. Yeni gönüllülere her zaman açığız!

Destekleri için teşekkürler:

- Fred L. Drake, Jr., orijinal Python dokümantasyon araç setinin yaratıcısı ve içeriğin çoğunun yazarı;
- the [Docutils](#) project for creating reStructuredText and the Docutils suite;
- Fredrik Lundh for his Alternative Python Reference project from which Sphinx got many good ideas.

### B.1 Python Dokümantasyonuna Katkıda Bulunanlar

Birçok kişi Python diline, Python standart kütüphanesine ve Python belgelerine katkıda bulunmuştur. Katkıda bulunanların kısmi listesi için Python kaynak dağıtımında [Misc/ACKS](#) adresine bakın.

Python topluluğunun girdileri ve katkılarıyla Python böyle harika bir dokümantasyona sahip – Teşekkürler!



## Tarihçe ve Lisans

## C.1 Yazılımın tarihçesi

Python, 1990'ların başında Guido van Rossum tarafından Hollanda'da Stichting Mathematisch Centrum'da (CWI, bkz. <https://www.cwi.nl/>) ABC adlı bir dilin devamı olarak oluşturuldu. Guido, diğerlerinin oldukça katkısı olmasına rağmen, Python'un ana yazarı olmaya devam ediyor.

1995'te Guido, yazılımın çeşitli sürümlerini yayınladığı Virginia, Reston'daki Ulusal Araştırma Girişimleri Kurumu'nda (CNRI, bkz. <https://www.cnri.reston.va.us/>) Python üzerindeki çalışmalarına devam etti.

Mayıs 2000'de, Guido ve Python çekirdek geliştirme ekibi, BeOpen PythonLabs ekibini oluşturmak için BeOpen.com'a taşındı. Aynı yılın Ekim ayında PythonLabs ekibi Digital Creations'a (şimdi Zope Corporation; bkz. <https://www.zope.org/>) taşındı. 2001 yılında, Python Yazılım Vakfı (PSF, bkz. <https://www.python.org/psf/>) kuruldu, özellikle Python ile ilgili Fikri Mülkiyete sahip olmak için oluşturulmuş kar amacı gütmeyen bir organizasyon. Zope Corporation, PSF'nin sponsor üyesidir.

Tüm Python sürümleri Açık Kaynaklıdır (Açık Kaynak Tanımı için bkz. <https://opensource.org/>). Tarihsel olarak, tümü olmasa da çoğu Python sürümleri de GPL uyumluydu; aşağıdaki tablo çeşitli yayınları özetlemektedir.

Yayın	Şundan türedi:	Yıl	Sahibi	GPL uyumlu mu?
0.9.0'dan 1.2'ye	n/a	1991-1995	CWI	evet
1.3 'dan 1.5.2'ye	1.2	1995-1999	CNRI	evet
1.6	1.5.2	2000	CNRI	hayır
2.0	1.6	2000	BeOpen.com	hayır
1.6.1	1.6	2001	CNRI	hayır
2.1	2.0+1.6.1	2001	PSF	hayır
2.0.1	2.0+1.6.1	2001	PSF	evet
2.1.1	2.1+2.0.1	2001	PSF	evet
2.1.2	2.1.1	2002	PSF	evet
2.1.3	2.1.2	2002	PSF	evet
2.2 ve üzeri	2.1.1	2001-Günümüz	PSF	evet

**Not:** GPL uyumlu olması, Python'u GPL kapsamında dağıttığımız anlamına gelmez. Tüm Python lisansları, GPL'den farklı olarak, değişikliklerinizi açık kaynak yapmadan değiştirilmiş bir sürümü dağıtmanıza izin verir. GPL uyumlu lisanslar, Python'u GPL kapsamında yayınlanan diğer yazılımlarla birleştirmeyi mümkün kılar; diğerleri yapmaz.

---

Bu yayınları mümkün kılmak için Guido'nun yönetimi altında çalışan birçok gönüllüye teşekkürler.

## C.2 Python'a erişmek veya başka bir şekilde kullanmak için şartlar ve koşullar

Python yazılımı ve belgeleri *PSF Lisans Anlaşması* kapsamında lisanslanmıştır.

Python 3.8.6'dan başlayarak, belgelerdeki örnekler, tarifler ve diğer kodlar, PSF Lisans Sözleşmesi ve *Zero-Clause BSD license* kapsamında çift lisanslıdır.

Python'a dahil edilen bazı yazılımlar farklı lisanslar altındadır. Lisanslar, bu lisansa giren kodla listelenir. Bu lisansların eksik listesi için bkz. *Tüzel Yazılımlar için Lisanslar ve Onaylar*.

### C.2.1 PYTHON İÇİN PSF LİSANS ANLAŞMASI 3.10.19

1. This LICENSE AGREEMENT is between the Python Software Foundation ("PSF"),  
→and  
the Individual or Organization ("Licensee") accessing and otherwise using  
→Python  
3.10.19 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 3.10.19 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 3.10.19 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 3.10.19 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  
3.10.19.
4. PSF is making Python 3.10.19 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 3.10.19 WILL NOT INFRINGE ANY THIRD PARTY RIGHTS.
5. PSF SHALL NOT BE LIABLE TO LICENSEE OR ANY OTHER USERS OF PYTHON 3.10.19 FOR ANY INCIDENTAL, SPECIAL, OR CONSEQUENTIAL DAMAGES OR LOSS AS A RESULT OF MODIFYING, DISTRIBUTING, OR OTHERWISE USING PYTHON 3.10.19, 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 3.10.19, Licensee agrees to be bound by the terms and conditions of this License Agreement.

## C.2.2 PYTHON 2.0 İÇİN BEOPEN.COM LİSANS SÖZLEŞMESİ

### BEOPEN PYTHON AÇIK KAYNAK LİSANS SÖZLEŞMESİ SÜRÜM 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.

(sonraki sayfaya devam)

(önceki sayfadan devam)

5. This License Agreement will automatically terminate upon a material breach of its terms and conditions.
6. This License Agreement shall be governed by and interpreted in all respects by the law of the State of California, excluding conflict of law provisions. Nothing in this License Agreement shall be deemed to create any relationship of agency, partnership, or joint venture between BeOpen and Licensee. This License Agreement does not grant permission to use BeOpen trademarks or trade names in a trademark sense to endorse or promote products or services of Licensee, or any third party. As an exception, the "BeOpen Python" logos available at <http://www.pythonlabs.com/logos.html> may be used according to the permissions granted on that web page.
7. By copying, installing or otherwise using the software, Licensee agrees to be bound by the terms and conditions of this License Agreement.

### C.2.3 PYTHON 1.6.1 İÇİN CNRI LİSANS ANLAŞMASI

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

(sonraki sayfaya devam)

(önceki sayfadan devam)

THEREOF, EVEN IF ADVISED OF THE POSSIBILITY THEREOF.

6. This License Agreement will automatically terminate upon a material breach of its terms and conditions.
7. This License Agreement shall be governed by the federal intellectual property law of the United States, including without limitation the federal copyright law, and, to the extent such U.S. federal law does not apply, by the law of the Commonwealth of Virginia, excluding Virginia's conflict of law provisions. Notwithstanding the foregoing, with regard to derivative works based on Python 1.6.1 that incorporate non-separable material that was previously distributed under the GNU General Public License (GPL), the law of the Commonwealth of Virginia shall govern this License Agreement only as to issues arising under or with respect to Paragraphs 4, 5, and 7 of this License Agreement. Nothing in this License Agreement shall be deemed to create any relationship of agency, partnership, or joint venture between CNRI and Licensee. This License Agreement does not grant permission to use CNRI trademarks or trade name in a trademark sense to endorse or promote products or services of Licensee, or any third party.
8. By clicking on the "ACCEPT" button where indicated, or by copying, installing or otherwise using Python 1.6.1, Licensee agrees to be bound by the terms and conditions of this License Agreement.

## C.2.4 0.9.0 ARASI 1.2 PYTHON İÇİN CWI LİSANS SÖZLEŞMESİ

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

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

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

## C.2.5 PYTHON 3.10.19 BELGELERİNDEKİ KOD İÇİN SIFIR MADDE BSD LİSANSI

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.

## C.3 Tüzel Yazılımlar için Lisanslar ve Onaylar

Bu bölüm, Python dağıtımına dahil edilmiş üçüncü taraf yazılımlar için tamamlanmamış ancak büyüyen bir lisans ve onay listesidir.

### C.3.1 Mersenne Twister'i

`_random` modülü, <http://www.math.sci.hiroshima-u.ac.jp/~m-mat/MT/MT2002/emt19937ar.html> adresinden indirilen kodu temel alır. Orijinal koddan kelimesi kelimesine yorumlar aşağıdadır:

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,

(sonraki sayfaya devam)

(önceki sayfadan devam)

```
PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR
PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF
LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING
NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
```

Any feedback is very welcome.

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

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

### C.3.2 Soketler

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.

### C.3.3 Asenkron soket hizmetleri

asynchat ve asyncore modülleri aşağıdaki uyarıyı içerir:

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

(sonraki sayfaya devam)

(önceki sayfadan devam)

granted, provided that the above copyright notice appear in all copies and that both that copyright notice and this permission notice appear in supporting documentation, and that the name of Sam Rushing not be used in advertising or publicity pertaining to distribution of the software without specific, written prior permission.

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

### C.3.4 Çerez yönetimi

`http.cookies` modülü aşağıdaki uyarıyı içerir:

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

All Rights Reserved

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

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

### C.3.5 Çalıştırma izleme

`trace` modülü aşağıdaki uyarıyı içerir:

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.

(sonraki sayfaya devam)

(önceki sayfadan devam)

Author: Skip Montanaro

Copyright 1999, Bioreason, Inc., all rights reserved.

Author: Andrew Dalke

Copyright 1995-1997, Automatrix, Inc., all rights reserved.

Author: Skip Montanaro

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

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

### C.3.6 UUencode ve UUdecode fonksiyonları

uu modülü aşağıdaki uyarıyı içerir:

Copyright 1994 by Lance Ellinghouse

Cathedral City, California Republic, United States of America.

All Rights Reserved

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

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

Modified by Jack Jansen, CWI, July 1995:

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

### C.3.7 XML Uzaktan Yordam Çağrıları

`xmlrpc.client` modülü aşağıdaki uyarıyı içerir:

```
The XML-RPC client interface is

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

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

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

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

### C.3.8 test\_epoll

`test_epoll` modülü aşağıdaki uyarıyı içerir:

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

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

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

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



### C.3.9 kqueue seçin

select modülü, kqueue arayüzü için aşağıdaki uyarıyı içerir:

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

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

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

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

### C.3.10 SipHash24

Python/pyhash.c dosyası, Dan Bernstein'in SipHash24 algoritmasının Marek Majkowski uygulamasını içerir. Burada aşağıdaki not yer alır:

<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>)

### C.3.11 strtod ve dtoa

The file `Python/dtoa.c`, which supplies C functions `dtoa` and `strtod` for conversion of C doubles to and from strings, is derived from the file of the same name by David M. Gay, currently available from <https://web.archive.org/web/20220517033456/http://www.netlib.org/fp/dtoa.c>. The original file, as retrieved on March 16, 2009, contains the following copyright and licensing notice:

```

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

```

### C.3.12 OpenSSL

`hashlib`, `posix`, `ssl`, `crypt` modülleri, işletim sistemi tarafından sağlanmışsa ek performans için OpenSSL kütüphanesini kullanır. Ek olarak, Python için Windows ve macOS yükleyicileri, OpenSSL kütüphanelerinin bir kopyasını içerebilir, bu nedenle buraya OpenSSL lisansının bir kopyasını ekliyoruz:

```

LICENSE ISSUES
=====

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

OpenSSL License
-----

/* =====
 * Copyright (c) 1998-2008 The OpenSSL Project. All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * 1. Redistributions of source code must retain the above copyright
 * notice, this list of conditions and the following disclaimer.
 *
 * 2. Redistributions in binary form must reproduce the above copyright
 * notice, this list of conditions and the following disclaimer in

```

(sonraki sayfaya devam)

(önceki sayfadan devam)

```

*   the documentation and/or other materials provided with the
*   distribution.
*
* 3. All advertising materials mentioning features or use of this
*   software must display the following acknowledgment:
*   "This product includes software developed by the OpenSSL Project
*   for use in the OpenSSL Toolkit. (http://www.openssl.org/)"
*
* 4. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to
*   endorse or promote products derived from this software without
*   prior written permission. For written permission, please contact
*   openssl-core@openssl.org.
*
* 5. Products derived from this software may not be called "OpenSSL"
*   nor may "OpenSSL" appear in their names without prior written
*   permission of the OpenSSL Project.
*
* 6. Redistributions of any form whatsoever must retain the following
*   acknowledgment:
*   "This product includes software developed by the OpenSSL Project
*   for use in the OpenSSL Toolkit (http://www.openssl.org/)"
*
* THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT ``AS IS'' AND ANY
* EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
* IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
* PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL THE OpenSSL PROJECT OR
* ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
* SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
* NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
* LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
* HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
* STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
* ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
* OF THE POSSIBILITY OF SUCH DAMAGE.
* =====
*
* This product includes cryptographic software written by Eric Young
* (eay@cryptsoft.com).  This product includes software written by Tim
* Hudson (tjh@cryptsoft.com).
*
*/

```

Original SSLeay License

-----

```

/* Copyright (C) 1995-1998 Eric Young (eay@cryptsoft.com)
* All rights reserved.
*
* This package is an SSL implementation written
* by Eric Young (eay@cryptsoft.com).
* The implementation was written so as to conform with Netscapes SSL.
*
* This library is free for commercial and non-commercial use as long as
* the following conditions are aheared to. The following conditions
* apply to all code found in this distribution, be it the RC4, RSA,
* lhash, DES, etc., code; not just the SSL code. The SSL documentation
* included with this distribution is covered by the same copyright terms

```

(sonraki sayfaya devam)

(önceki sayfadan devam)

```

* except that the holder is Tim Hudson (tjh@cryptsoft.com).
*
* Copyright remains Eric Young's, and as such any Copyright notices in
* the code are not to be removed.
* If this package is used in a product, Eric Young should be given attribution
* as the author of the parts of the library used.
* This can be in the form of a textual message at program startup or
* in documentation (online or textual) provided with the package.
*
* Redistribution and use in source and binary forms, with or without
* modification, are permitted provided that the following conditions
* are met:
* 1. Redistributions of source code must retain the copyright
*   notice, this list of conditions and the following disclaimer.
* 2. Redistributions in binary form must reproduce the above copyright
*   notice, this list of conditions and the following disclaimer in the
*   documentation and/or other materials provided with the distribution.
* 3. All advertising materials mentioning features or use of this software
*   must display the following acknowledgement:
*   "This product includes cryptographic software written by
*    Eric Young (eay@cryptsoft.com)"
*   The word 'cryptographic' can be left out if the rouines from the library
*   being used are not cryptographic related :-).
* 4. If you include any Windows specific code (or a derivative thereof) from
*   the apps directory (application code) you must include an acknowledgement:
*   "This product includes software written by Tim Hudson (tjh@cryptsoft.com)"
*
* THIS SOFTWARE IS PROVIDED BY ERIC YOUNG ``AS IS'' AND
* ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
* IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
* ARE DISCLAIMED. IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
* FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
* DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
* OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
* HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
* LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
* OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
* SUCH DAMAGE.
*
* The licence and distribution terms for any publically available version or
* derivative of this code cannot be changed. i.e. this code cannot simply be
* copied and put under another distribution licence
* [including the GNU Public Licence.]
*/

```

### C.3.13 expat

pyexpat uzantısı, derleme --with-system-expat şeklinde yapılandırılmadığı sürece, expat kaynaklarının dahil edildiği bir kopya kullanılarak oluşturulur:

```

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

```

(sonraki sayfaya devam)

(önceki sayfadan devam)

"Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

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

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

### C.3.14 libffi

`_ctypes` uzantısı, yapı `--with-system-libffi` olarak yapılandırılmadığı sürece libffi kaynaklarının dahil edildiği bir kopya kullanılarak oluşturulur:

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

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

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

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

### C.3.15 zlib

zlib uzantısı, sistemde bulunan zlib sürümünü derleme için kullanılamayacak kadar eskiyse, zlib kaynaklarının dahil edildiği bir kopya kullanılarak oluşturulur:

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

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

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

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

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

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

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

### C.3.16 cfuhash

tracemalloc tarafından kullanılan hash tablosunun uygulanması cfuhash projesine dayanmaktadır:

```
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
  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
```

(sonraki sayfaya devam)

(önceki sayfadan devam)

```
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.
```

### C.3.17 libmpdec

`_decimal` modülü, yapı `--with-system-libmpdec` şeklinde yapılandırılmadığı sürece libmpdec kitaplığının dahil edildiği bir kopya kullanılarak oluşturulur:

```
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.

### C.3.18 W3C C14N test paketi

`test` paketindeki C14N 2.0 test paketi (`Lib/test/xmltestdata/c14n-20/`), <https://www.w3.org/TR/xml-c14n2-testcases/> adresindeki W3C web sitesinden alınmıştır ve 3 maddeli BSD lisansı altında dağıtılmaktadır:

```
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 are met:

- \* Redistributions of works must retain the original copyright notice,

(sonraki sayfaya devam)

(önceki sayfadan devam)

```
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.
```

### C.3.19 Audioop

The audioop module uses the code base in g771.c file of the SoX project. <https://sourceforge.net/projects/sox/files/sox/12.17.7/sox-12.17.7.tar.gz>

This source code is a product of Sun Microsystems, Inc. and is provided for unrestricted use. Users may copy or modify this source code without charge.

SUN SOURCE CODE IS PROVIDED AS IS WITH NO WARRANTIES OF ANY KIND INCLUDING THE WARRANTIES OF DESIGN, MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE, OR ARISING FROM A COURSE OF DEALING, USAGE OR TRADE PRACTICE.

Sun source code is provided with no support and without any obligation on the part of Sun Microsystems, Inc. to assist in its use, correction, modification or enhancement.

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.

In no event will Sun Microsystems, Inc. be liable for any lost revenue or profits or other special, indirect and consequential damages, even if Sun has been advised of the possibility of such damages.

Sun Microsystems, Inc. 2550 Garcia Avenue Mountain View, California 94043



---

## Telif Hakkı

---

Python ve bu dokümantasyon:

Telif Hakkı © 2001-2023 Python Software Foundation. Tüm hakları saklıdır.

Telif Hakkı © 2000 BeOpen.com. Tüm hakları saklıdır.

Telif Hakkı © 1995-2000 Ulusal Araştırma Girişimleri Kurumu. Tüm hakları saklıdır.

Telif Hakkı © 1991-1995 Stichting Mathematisch Centrum. Tüm hakları saklıdır.

---

Bütün lisans ve izin bilgileri için [Tarihçe ve Lisans](#) 'a göz atın.



## Alfabetik olmayan

..., [57](#)

-?

komut satırı seçeneği, [5](#)

%APPDATA%, [38](#)

2to3, [57](#)

>>>, [57](#)

\_\_future\_\_, [62](#)

\_\_slots\_\_, [68](#)

## A

ad alanı, [66](#)

ad alanı paketi, [66](#)

adlandırılmış demet, [65](#)

anahtar işlev, [64](#)

anahtar kelime argümanı, [64](#)

anlamak, [69](#)

argüman, [57](#)

asenkron bağlam yöneticisi, [58](#)

asenkron jeneratör, [58](#)

asenkron jeneratör yineleyici, [58](#)

asenkron yineleyici, [58](#)

## B

-B

komut satırı seçeneği, [6](#)

-b

komut satırı seçeneği, [6](#)

bağlam değişkeni, [59](#)

bağlam yöneticisi, [59](#)

bayt benzeri nesne, [59](#)

bayt kodu, [59](#)

BDFL, [58](#)

beklenebilir, [58](#)

belge dizisi, [60](#)

bitişik, [60](#)

borrowed reference, [59](#)

BOŞTA, [63](#)

bulucu, [61](#)

## C

-c <command>

komut satırı seçeneği, [4](#)

callable, [59](#)

C-contiguous, [60](#)

CFLAGS, [2830](#)

CFLAGS\_NODIST, [2830](#)

--check-hash-based-pycs

default|always|never

komut satırı seçeneği, [6](#)

CPPFLAGS, [28, 30](#)

CPython, [60](#)

## Ç

çöp toplama, [62](#)

## D

-d

komut satırı seçeneği, [6](#)

değişken açıklama, [70](#)

değişmez, [63](#)

değiştirilebilir, [65](#)

dekoratör, [60](#)

dipnot, [57](#)

--disable-ipv6

komut satırı seçeneği, [19](#)

--disable-test-modules

komut satırı seçeneği, [21](#)

dizi, [68](#)

dosya benzeri nesne, [61](#)

dosya nesnesi, [61](#)

dosya sistemi kodlaması ve hata  
işleyicisi, [61](#)

duck-typing, [60](#)

## E

-E

komut satırı seçeneği, [6](#)

EAFP, [61](#)

```
--enable-big-digits =[15|30]
    komut satırı seçeneği, 19
--enable-framework
    komut satırı seçeneği, 25
--enable-framework =INSTALLDIR
    komut satırı seçeneği, 25
--enable-loadable-sqlite-extensions
    komut satırı seçeneği, 19
--enable-optimizations
    komut satırı seçeneği, 21
--enable-profiling
    komut satırı seçeneği, 22
--enable-shared
    komut satırı seçeneği, 24
--enable-universalsdk
    komut satırı seçeneği, 25
--enable-universalsdk =SDKDIR
    komut satırı seçeneği, 25
eşyordam, 60
eşyordam işlevi, 60
eşzamansız yinelenebilir, 58
etkileşimli, 63
evrensel yeni satırlar, 70
--exec-prefix =EPREFIX
    komut satırı seçeneği, 21
```

## F

f-string, 61  
fonksiyon, 61  
fonksiyon açıklaması, 61  
Fortran contiguous, 60

## G

geçici API, 67  
geçici paket, 67  
genel işlev, 62  
genel tercüman kilidi, 62  
genel tip, 62  
generator, 62  
generator expression, 62  
geri çağırma, 59  
GIL, 62  
güçlü referans, 69

## H

```
-h
    komut satırı seçeneği, 5
haritalama, 65
--help
    komut satırı seçeneği, 5
```

## I

```
-I
```

komut satırı seçeneği, 6

## I

```
-i
    komut satırı seçeneği, 6
iç içe kapsam, 66
iç aktarıcı, 63
iç aktarım yolu, 63
iç aktarma, 63
ifade (değer döndürmez), 69
ifade (değer döndürür), 61
ikili dosya, 58
```

## J

```
-J
    komut satırı seçeneği, 9
jeneratör, 62
jeneratör ifadesi, 62
jeneratör yineleyici, 62
```

## K

karma tabanlı pyc, 63  
karmaşık sayı, 59  
kat bölümü, 61  
kısım, 67  
komut satırı seçeneği  
    -?, 5  
    -B, 6  
    -b, 6  
    -c <command>, 4  
    --check-hash-based-pycs  
        default|always|never, 6  
    -d, 6  
    --disable-ipv6, 19  
    --disable-test-modules, 21  
    -E, 6  
    --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, 24  
    --enable-universalsdk, 25  
    --enable-universalsdk =SDKDIR, 25  
    --exec-prefix =EPREFIX, 21  
    -h, 5  
    --help, 5  
    -I, 6  
    -i, 6  
    -J, 9  
    -m <module-name>, 4

```

-O, 6
-OO, 6
--prefix =PREFIX, 21
-q, 6
-R, 6
-S, 7
-s, 7
-u, 7
-V, 5
-v, 7
--version, 5
-W arg, 7
--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], method, 65
    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, 22
--without-readline, 24
--without-static-libpython, 24
--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
-x, 8
konumsal argüman, 67

L
lambda, 64
LBYL, 64
LDFLAGS, 28, 30
LDFLAGS_NODIST, 30
liste, 64
liste anlama, 64

M
-m <module-name>
    komut satırı seçeneği, 4
magic
meta yol bulucu, 65
metasınıf, 65
method
    magic, 65
    special, 69
metot, 65
metot kalite sıralaması, 65
modül, 65
modül özelliği, 65
MRO, 65

N
nitelik, 58
nitelikli isim, 68

O
-O
    komut satırı seçeneği, 6
obje, 66
-OO
    komut satırı seçeneği, 6
ortam değişkeni
    %APPDATA%, 38
    BASECFLAGS, 29
    BASECPPFLAGS, 28
    BLD_SHARED, 30
    CC, 28
    C_CSHARED, 29
    CFLAGS, 2830
    CFLAGS_ALIASING, 29
    CFLAGS_NODIST, 2830

```

CFLAGSFORSHARED, 29  
CONFIGURE\_CFLAGS, 29  
CONFIGURE\_CFLAGS\_NODIST, 29  
CONFIGURE\_CPPFLAGS, 28  
CONFIGURE\_LDFLAGS, 30  
CONFIGURE\_LDFLAGS\_NODIST, 30  
CPPFLAGS, 28, 30  
CXX, 28  
EXTRA\_CFLAGS, 29  
LDFLAGS, 28, 30  
LDFLAGS\_NODIST, 30  
LD\_SHARED, 30  
LIBS, 30  
LINKCC, 30  
MAINCC, 28  
OPT, 23, 29  
PATH, 9, 17, 3436, 4145  
PATHEXT, 36  
PROFILE\_TASK, 21  
PURIFY, 30  
PY\_BUILTIN\_MODULE\_CFLAGS, 29  
PY\_CFLAGS, 29  
PY\_CFLAGS\_NODIST, 29  
PY\_CORE\_CFLAGS, 29  
PY\_CORE\_LDFLAGS, 31  
PY\_CPPFLAGS, 28  
PY\_LDFLAGS, 30  
PY\_LDFLAGS\_NODIST, 30  
PY\_PYTHON, 46  
PY\_STDMODULE\_CFLAGS, 29  
PYTHON\*, 46  
PYTHONASYNCIODEBUG, 12  
PYTHONBREAKPOINT, 10  
PYTHONCASEOK, 10  
PYTHONCOERCECLOCALE, 13, 20  
PYTHONDEBUG, 6, 10  
PYTHONDEVMODE, 13  
PYTHONDONTWRITEBYTECODE, 6, 10  
PYTHONDUMPPREFS, 14, 23  
PYTHONEXECUTABLE, 11  
PYTHONFAULTHANDLER, 11  
PYTHONHASHSEED, 6, 7, 10  
PYTHONHOME, 6, 9, 47, 48  
PYTHONINSPECT, 6, 10  
PYTHONINTMAXSTRDIGITS, 8, 11  
PYTHONIOENCODING, 11, 13  
PYTHONLEGACYWINDOWSFSENCODING, 12  
PYTHONLEGACYWINDOWSTDIO, 11, 12  
PYTHONMALLOC, 12, 22  
PYTHONMALLOCSTATS, 12  
PYTHONNOUSERSITE, 11  
PYTHONOPTIMIZE, 6, 10  
PYTHONPATH, 6, 9, 42, 47, 48, 52  
PYTHONPLATLIBDIR, 9

PYTHONPROFILEIMPORTTIME, 8, 12  
PYTHONPYCACHEPREFIX, 8, 10  
PYTHONSTARTUP, 6, 9  
PYTHONTHREADDEBUG, 14, 22  
PYTHONTRACEMALLOC, 12  
PYTHONUNBUFFERED, 7, 10  
PYTHONUSERBASE, 11  
PYTHONUTF8, 13, 42  
PYTHONVERBOSE, 7, 10  
PYTHONWARNDEFAULTENCODING, 8, 13  
PYTHONWARNINGS, 8, 11  
TEMP, 38

## Ö

özel metod, 69

## P

paket, 66  
parametre, 66  
parçalamak, 69  
PATH, 9, 17, 3436, 4145  
PATHEXT, 36  
PEP, 67  
--prefix =PREFIX  
    komut satırı seçeneği, 21  
PROFILE\_TASK, 21  
PY\_PYTHON, 46  
Python 3000, 67  
Python Geliştirme Önerileri  
    PEP 1, 67  
    PEP 8, 55  
    PEP 11, 33, 49  
    PEP 238, 61  
    PEP 278, 70  
    PEP 302, 61, 65  
    PEP 338, 4  
    PEP 343, 59  
    PEP 362, 58, 67  
    PEP 370, 7, 11  
    PEP 397, 43  
    PEP 411, 67  
    PEP 420, 61, 66, 67  
    PEP 443, 62  
    PEP 451, 61  
    PEP 483, 62  
    PEP 484, 57, 62, 70  
    PEP 488, 6  
    PEP 492, 58, 60  
    PEP 498, 61  
    PEP 519, 67  
    PEP 525, 58  
    PEP 526, 57, 70  
    PEP 528, 43  
    PEP 529, 12, 43

PEP 538, 13, 20  
 PEP 585, 62  
 PEP 3116, 70  
 PEP 3155, 68

PYTHON\*, 46  
 PYTHONCOERCECLOCALE, 20  
 PYTHONDEBUG, 6  
 PYTHONDONTWRITEBYTECODE, 6  
 PYTHONDUMPPREFS, 23  
 PYTHONHASHSEED, 6, 7, 10  
 PYTHONHOME, 6, 9, 47, 48  
 PYTHONINSPECT, 6  
 PYTHONINTMAXSTRDIGITS, 8  
 PYTHONIOENCODING, 13  
 Pythonic, 68  
 PYTHONLEGACYWINDOWSSTDIO, 11  
 PYTHONMALLOC, 12, 22  
 PYTHONOPTIMIZE, 6  
 PYTHONPATH, 6, 9, 42, 47, 48, 52  
 PYTHONPROFILEIMPORTTIME, 8  
 PYTHONPYCACHEPREFIX, 8  
 PYTHONSTARTUP, 6  
 PYTHONTHREADDEBUG, 22  
 Python'un Zen'i, 70  
 PYTHONUNBUFFERED, 7  
 PYTHONUTF8, 13, 42  
 PYTHONVERBOSE, 7  
 PYTHONWARNDEFAULTENCODING, 8  
 PYTHONWARNINGS, 8

## Q

-q  
 komut satırı seçeneği, 6

## R

-R  
 komut satırı seçeneği, 6  
 referans sayısı, 68

## S

-S  
 komut satırı seçeneği, 7  
 -s  
 komut satırı seçeneği, 7  
 sanal makine, 70  
 sanal ortam, 70  
 sınıf, 59  
 sınıf değişkeni, 59  
 sihirli yöntem, 65  
 soyut temel sınıf, 57  
 sözlük, 60  
 sözlük anlama, 60  
 sözlük görünümü, 60  
 special

method, 69  
 sürekli paketleme, 68

## T

tanımlayıcı, 60  
 tek sevk, 69  
 TEMP, 38  
 tercüman kapatma, 63  
 tip, 69  
 tip takma adı, 69  
 tür ipucu, 70

## U

-u  
 komut satırı seçeneği, 7  
 uzatma modülü, 61

## Ü

üç tırnaklı dize, 69

## V

-V  
 komut satırı seçeneği, 5  
 -v  
 komut satırı seçeneği, 7  
 --version  
 komut satırı seçeneği, 5

## W

-W arg  
 komut satırı seçeneği, 7  
 --with-address-sanitizer  
 komut satırı seçeneği, 23  
 --with-assertions  
 komut satırı seçeneği, 23  
 --with-builtin-hashlib-hashes =md5,sha1,sha256,sha512  
 komut satırı seçeneği, 25  
 --with-computed-gotos  
 komut satırı seçeneği, 21  
 --with-cxx-main  
 komut satırı seçeneği, 19  
 --with-cxx-main =COMPILER  
 komut satırı seçeneği, 19  
 --with-dbmliborder =db1:db2:...  
 komut satırı seçeneği, 20  
 --with-dtrace  
 komut satırı seçeneği, 23  
 --with-ensurepip =[upgrade|install|no]  
 komut satırı seçeneği, 21  
 --with-framework-name =FRAMEWORK  
 komut satırı seçeneği, 26  
 --with-hash-algorithm =[fnv|siphash24]  
 komut satırı seçeneği, 25

```
--with-libc =STRING
    komut satırı seçeneği, 24
--with-libm =STRING
    komut satırı seçeneği, 24
--with-libs = 'lib1 ...'
    komut satırı seçeneği, 24
--with-lto
    komut satırı seçeneği, 21
--with-memory-sanitizer
    komut satırı seçeneği, 23
--with-openssl =DIR
    komut satırı seçeneği, 24
--with-openssl-rpath = [no|auto|DIR]
    komut satırı seçeneği, 24
--without-c-locale-coercion
    komut satırı seçeneği, 20
--without-decimal-contextvar
    komut satırı seçeneği, 20
--without-doc-strings
    komut satırı seçeneği, 22
--without-pymalloc
    komut satırı seçeneği, 22
--without-readline
    komut satırı seçeneği, 24
--without-static-libpython
    komut satırı seçeneği, 24
--with-platlibdir =DIRNAME
    komut satırı seçeneği, 20
--with-pydebug
    komut satırı seçeneği, 23
--with-readline =editline
    komut satırı seçeneği, 24
--with-ssl-default-suites = [python|openssl|STRING]
    komut satırı seçeneği, 25
--with-suffix =SUFFIX
    komut satırı seçeneği, 19
--with-system-expat
    komut satırı seçeneği, 24
--with-system-ffi
    komut satırı seçeneği, 24
--with-system-libmpdec
    komut satırı seçeneği, 24
--with-tcltk-includes = '-I...'
    komut satırı seçeneği, 24
--with-tcltk-libs = '-L...'
    komut satırı seçeneği, 24
--with-trace-refs
    komut satırı seçeneği, 23
--with-tzpath =<list of absolute paths
    separated by pathsep>
    komut satırı seçeneği, 20
--with-undefined-behavior-sanitizer
    komut satırı seçeneği, 23
--with-universal-archs =ARCH
```

```
    komut satırı seçeneği, 25
--with-valgrind
    komut satırı seçeneği, 23
--with-wheel-pkg-dir =PATH
    komut satırı seçeneği, 20
```

## X

```
-X
    komut satırı seçeneği, 8
-x
    komut satırı seçeneği, 8
```

## Y

yazı çözümleme, [69](#)  
yazı dosyası, [69](#)  
yeni stil sınıf, [66](#)  
yerel kodlama, [64](#)  
yıkabilir, [63](#)  
yinelenebilir, [63](#)  
yineleyici, [64](#)  
yol benzeri nesne, [67](#)  
yol giriş kancası, [67](#)  
yol girişi, [67](#)  
yol girişi bulucu, [67](#)  
yol tabanlı bulucu, [67](#)  
yorumlanmış, [63](#)  
yükleyici, [65](#)

## Z

zorlama, [59](#)