

---

# Installing Python Modules

*Sürüm 3.9.20*

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

**Eylül 08, 2024**

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



<b>1</b>	<b>Key terms</b>	<b>3</b>
<b>2</b>	<b>Basic usage</b>	<b>5</b>
<b>3</b>	<b>How do I ...?</b>	<b>7</b>
3.1	... install <code>pip</code> in versions of Python prior to Python 3.4? . . . . .	7
3.2	... install packages just for the current user? . . . . .	7
3.3	... install scientific Python packages? . . . . .	7
3.4	... work with multiple versions of Python installed in parallel? . . . . .	8
<b>4</b>	<b>Common installation issues</b>	<b>9</b>
4.1	Installing into the system Python on Linux . . . . .	9
4.2	Pip not installed . . . . .	9
4.3	Installing binary extensions . . . . .	9
<b>A</b>	<b>Sözlük</b>	<b>11</b>
<b>B</b>	<b>Dokümanlar hakkında</b>	<b>25</b>
B.1	Python Dokümantasyonuna Katkıda Bulunanlar . . . . .	25
<b>C</b>	<b>Tarihçe ve Lisans</b>	<b>27</b>
C.1	Yazılımın tarihçesi . . . . .	27
C.2	Python'a erişmek veya başka bir şekilde kullanmak için şartlar ve koşullar . . . . .	28
C.2.1	PYTHON İÇİN PSF LİSANS ANLAŞMASI 3.9.20 . . . . .	28
C.2.2	PYTHON 2.0 İÇİN BEOPEN.COM LİSANS SÖZLEŞMESİ . . . . .	29
C.2.3	PYTHON 1.6.1 İÇİN CNRI LİSANS ANLAŞMASI . . . . .	30
C.2.4	0.9.0 ARASI 1.2 PYTHON İÇİN CWI LİSANS SÖZLEŞMESİ . . . . .	31
C.2.5	PYTHON 3.9.20 BELGELERİNDEKİ KOD İÇİN SIFIR MADDE BSD LİSANSI . . . . .	32
C.3	Tüzel Yazılımlar için Lisanslar ve Onaylar . . . . .	32
C.3.1	Mersenne Twister'ı . . . . .	32
C.3.2	Soketler . . . . .	33
C.3.3	Asenkron soket hizmetleri . . . . .	33
C.3.4	Çerez yönetimi . . . . .	34
C.3.5	Çalıştırma izleme . . . . .	34
C.3.6	UUencode ve UUdecode fonksiyonları . . . . .	35
C.3.7	XML Uzaktan Yordam Çağrıları . . . . .	36
C.3.8	test_epoll . . . . .	36

C.3.9	kqueue seçin . . . . .	37
C.3.10	SipHash24 . . . . .	37
C.3.11	strtod ve dtoa . . . . .	38
C.3.12	OpenSSL . . . . .	38
C.3.13	expat . . . . .	40
C.3.14	libffi . . . . .	41
C.3.15	zlib . . . . .	41
C.3.16	cfuhash . . . . .	42
C.3.17	libmpdec . . . . .	43
C.3.18	W3C C14N test paketi . . . . .	43
<b>D</b>	<b>Telif Hakkı</b>	<b>45</b>
	<b>Dizin</b>	<b>47</b>

**Email** [distutils-sig@python.org](mailto:distutils-sig@python.org)

As a popular open source development project, Python has an active supporting community of contributors and users that also make their software available for other Python developers to use under open source license terms.

This allows Python users to share and collaborate effectively, benefiting from the solutions others have already created to common (and sometimes even rare!) problems, as well as potentially contributing their own solutions to the common pool.

This guide covers the installation part of the process. For a guide to creating and sharing your own Python projects, refer to the distribution guide.

---

**Not:** For corporate and other institutional users, be aware that many organisations have their own policies around using and contributing to open source software. Please take such policies into account when making use of the distribution and installation tools provided with Python.

---



---

## Key terms

---

- `pip` is the preferred installer program. Starting with Python 3.4, it is included by default with the Python binary installers.
- A *virtual environment* is a semi-isolated Python environment that allows packages to be installed for use by a particular application, rather than being installed system wide.
- `venv` is the standard tool for creating virtual environments, and has been part of Python since Python 3.3. Starting with Python 3.4, it defaults to installing `pip` into all created virtual environments.
- `virtualenv` is a third party alternative (and predecessor) to `venv`. It allows virtual environments to be used on versions of Python prior to 3.4, which either don't provide `venv` at all, or aren't able to automatically install `pip` into created environments.
- The [Python Package Index](#) is a public repository of open source licensed packages made available for use by other Python users.
- the [Python Packaging Authority](#) is the group of developers and documentation authors responsible for the maintenance and evolution of the standard packaging tools and the associated metadata and file format standards. They maintain a variety of tools, documentation, and issue trackers on both [GitHub](#) and [Bitbucket](#).
- `distutils` is the original build and distribution system first added to the Python standard library in 1998. While direct use of `distutils` is being phased out, it still laid the foundation for the current packaging and distribution infrastructure, and it not only remains part of the standard library, but its name lives on in other ways (such as the name of the mailing list used to coordinate Python packaging standards development).

3.5 sürümünde değişti: The use of `venv` is now recommended for creating virtual environments.

**Ayrıca bkz.:**

[Python Packaging User Guide: Creating and using virtual environments](#)





---

### Basic usage

---

The standard packaging tools are all designed to be used from the command line.

The following command will install the latest version of a module and its dependencies from the Python Package Index:

```
python -m pip install SomePackage
```

---

**Not:** For POSIX users (including macOS and Linux users), the examples in this guide assume the use of a *virtual environment*.

For Windows users, the examples in this guide assume that the option to adjust the system PATH environment variable was selected when installing Python.

---

It's also possible to specify an exact or minimum version directly on the command line. When using comparator operators such as `>`, `<` or some other special character which get interpreted by shell, the package name and the version should be enclosed within double quotes:

```
python -m pip install SomePackage ==1.0.4    # specific version
python -m pip install "SomePackage>=1.0.4"  # minimum version
```

Normally, if a suitable module is already installed, attempting to install it again will have no effect. Upgrading existing modules must be requested explicitly:

```
python -m pip install --upgrade SomePackage
```

More information and resources regarding `pip` and its capabilities can be found in the [Python Packaging User Guide](#).

Creation of virtual environments is done through the `venv` module. Installing packages into an active virtual environment uses the commands shown above.

**Ayrıca bkz.:**

[Python Packaging User Guide: Installing Python Distribution Packages](#)



These are quick answers or links for some common tasks.

### 3.1 ... install `pip` in versions of Python prior to Python 3.4?

Python only started bundling `pip` with Python 3.4. For earlier versions, `pip` needs to be “bootstrapped” as described in the Python Packaging User Guide.

**Ayrıca bkz.:**

[Python Packaging User Guide: Requirements for Installing Packages](#)

### 3.2 ... install packages just for the current user?

Passing the `--user` option to `python -m pip install` will install a package just for the current user, rather than for all users of the system.

### 3.3 ... install scientific Python packages?

A number of scientific Python packages have complex binary dependencies, and aren't currently easy to install using `pip` directly. At this point in time, it will often be easier for users to install these packages by [other means](#) rather than attempting to install them with `pip`.

**Ayrıca bkz.:**

[Python Packaging User Guide: Installing Scientific Packages](#)

## 3.4 ... work with multiple versions of Python installed in parallel?

On Linux, macOS, and other POSIX systems, use the versioned Python commands in combination with the `-m` switch to run the appropriate copy of `pip`:

```
python2    -m pip install SomePackage # default Python 2
python2.7  -m pip install SomePackage # specifically Python 2.7
python3    -m pip install SomePackage # default Python 3
python3.4  -m pip install SomePackage # specifically Python 3.4
```

Appropriately versioned `pip` commands may also be available.

On Windows, use the `py` Python launcher in combination with the `-m` switch:

```
py -2      -m pip install SomePackage # default Python 2
py -2.7    -m pip install SomePackage # specifically Python 2.7
py -3      -m pip install SomePackage # default Python 3
py -3.4    -m pip install SomePackage # specifically Python 3.4
```

---

## Common installation issues

---

### 4.1 Installing into the system Python on Linux

On Linux systems, a Python installation will typically be included as part of the distribution. Installing into this Python installation requires root access to the system, and may interfere with the operation of the system package manager and other components of the system if a component is unexpectedly upgraded using `pip`.

On such systems, it is often better to use a virtual environment or a per-user installation when installing packages with `pip`.

### 4.2 Pip not installed

It is possible that `pip` does not get installed by default. One potential fix is:

```
python -m ensurepip --default-pip
```

There are also additional resources for [installing pip](#).

### 4.3 Installing binary extensions

Python has typically relied heavily on source based distribution, with end users being expected to compile extension modules from source as part of the installation process.

With the introduction of support for the binary `wheel` format, and the ability to publish wheels for at least Windows and macOS through the Python Package Index, this problem is expected to diminish over time, as users are more regularly able to install pre-built extensions rather than needing to build them themselves.

Some of the solutions for installing [scientific software](#) that are not yet available as pre-built `wheel` files may also help with obtaining other binary extensions without needing to build them locally.

**Ayrıca bkz.:**

Python Packaging User Guide: Binary Extensions

>>> 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 ayırıştırarak ve ayırıştırma 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.

See *variable annotation*, *function annotation*, **PEP 484** and **PEP 526**, which describe this functionality.

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

- *keyword argument*: bir işlev çağrısında 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 referenced by name using dotted expressions. For example, if an object *o* has an attribute *a* it would be referenced as *o.a*.

**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.

**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.

**geri çağırma** 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 parçacığının bir değişken için farklı bir değere sahip olabileceği Thread-Local Storage’a benzer. Bununla birlikte, bağlam değişkenleriyle, 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 tanımlandı.

**CPython** Python programlama dilinin [python.org](#) üzerinde dağıtıldığı şekliyle 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 görünümüleri 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 değişmezi. Paket yürütüldüğünde yoksayılırken, derleyici tarafından tanınır ve çevreleyen sınıfın, işlevin veya modülün `__doc__` özneliğine yerleştirilir. İç gözetim 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 görünüyorsa 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, öznelik erişimi, işlemler veya işlev çağrıları gibi ifade öğelerinin bir toplamıdır. Diğer birçok dilin aksine, tüm dil yapıları ifade değildir. Ayrıca `while` gibi kullanılamayan *ifadeleler* 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.

**bulucu** İçerik 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.

See *variable annotation* and [PEP 484](#), which describe this functionality.

**\_\_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*'in olası değerlerini belgeler. Bu modülü içeri aktararak ve değişkenlerini değerlendirerek, 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 paralellığın 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 ön bellek 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 edilebilirdir. Hepsisi eşit olmayı karşılaştırır (kendileriyle hariç) ve hash değerleri `id()`’lerinden türetilir.

**BOŞTA** An Integrated Development 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.

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

**İçe 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 çalıştırılabilmesi 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` ögesini yeniden yükseltir. Yineleyicilerin, yineleyici nesnesinin kendisini döndüren bir `__iter__()` yöntemine sahip olmaları gerekir, böylece her yineleyici de yinelenebilir 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.

**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* kaldırırrsa başarısız olabilir. Bu sorun, kilitlerle veya EAFP yaklaşımı kullanılarak çözülebilir.

**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 eşanlamı.

**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, tekileri 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` öğesinden 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 değildir.

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 an `__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 giriş 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 sürece, 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 yönteme 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, module 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 özelliğidir. `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 öge 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** Ögelerin 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.

**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 faydalıdır. 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.

See *function annotation*, **PEP 484** and **PEP 526**, which describe this functionality.

**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 3.9.20

1. This LICENSE AGREEMENT is between the Python Software Foundation ("PSF"),  
and  
the Individual or Organization ("Licensee") accessing and otherwise using  
Python  
3.9.20 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 3.9.20 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 3.9.20 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 3.9.20 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  
3.9.20.
4. PSF is making Python 3.9.20 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 3.9.20 WILL NOT INFRINGE ANY THIRD PARTY RIGHTS.

5. PSF SHALL NOT BE LIABLE TO LICENSEE OR ANY OTHER USERS OF PYTHON 3.9.20  
 FOR ANY INCIDENTAL, SPECIAL, OR CONSEQUENTIAL DAMAGES OR LOSS AS A RESULT  
 →OF  
 MODIFYING, DISTRIBUTING, OR OTHERWISE USING PYTHON 3.9.20, 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 3.9.20, 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.

(continues on next page)

(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

(continues on next page)

(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 3.9.20 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,

(continues on next page)

(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, <http://www.wide.ad.jp/>.

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

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

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

THIS SOFTWARE IS PROVIDED BY THE PROJECT AND CONTRIBUTORS ``AS IS'' AND 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

(continues on next page)

(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.

(continues on next page)

(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 <http://www.netlib.org/fp/>. The original file, as retrieved on March 16, 2009, contains the following copyright and licensing notice:

```

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

```

### C.3.12 OpenSSL

`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
 * the documentation and/or other materials provided with the

```

(continues on next page)

(önceki sayfadan devam)

```

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

```

Original SSLeay License

```

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

```

(continues on next page)

(önceki sayfadan devam)

```

*
* 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 routines 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
"Software"), to deal in the Software without restriction, including

```

(continues on next page)

(önceki sayfadan devam)

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 sürümü 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.

(continues on next page)

(önceki sayfadan devam)

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

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

Jean-loup Gailly  
jloup@gzip.org

Mark Adler  
madler@alumni.caltech.edu

### 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 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,
  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,
```

(continues on next page)

(önceki sayfadan devam)

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.



---

## 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.



## Non-alphabetical

..., [11](#)  
2to3, [11](#)  
>>>, [11](#)  
\_\_future\_\_, [15](#)  
\_\_slots\_\_, [22](#)

## A

ad alanı, [19](#)  
ad alanı paketi, [19](#)  
adlandırılmış demet, [19](#)  
anahtar işlev, [17](#)  
anahtar kelime argümanı, [18](#)  
anlamak, [22](#)  
argüman, [11](#)  
asenkron bağlam yöneticisi, [12](#)  
asenkron jeneratör, [12](#)  
asenkron jeneratör yineleyici, [12](#)  
asenkron yineleyici, [12](#)

## B

bağlam değişkeni, [13](#)  
bağlam yöneticisi, [13](#)  
bayt benzeri nesne, [12](#)  
bayt kodu, [13](#)  
BDFL, [12](#)  
beklenebilir, [12](#)  
belge dizisi, [14](#)  
bitişik, [13](#)  
BOŞTA, [16](#)  
bulucu, [15](#)

## C

C-contiguous, [13](#)  
CPython, [13](#)

## Ç

çöp toplama, [15](#)

## D

değişken açıklama, [23](#)  
değişmez, [16](#)  
değiştirilebilir, [19](#)  
dekoratör, [14](#)  
dipnot, [11](#)  
dizi, [22](#)  
dosya benzeri nesne, [15](#)  
dosya nesnesi, [15](#)  
duck-typing, [14](#)

## E

EAFP, [14](#)  
eşyordam, [13](#)  
eşyordam işlevi, [13](#)  
eşzamansız yinelenenir, [12](#)  
etkileşimli, [17](#)  
evrensel yeni satırlar, [23](#)

## F

f-string, [15](#)  
fonksiyon, [15](#)  
fonksiyon açıklaması, [15](#)  
Fortran contiguous, [13](#)

## G

geçici API, [21](#)  
geçici paket, [21](#)  
genel işlev, [16](#)  
genel tercüman kilidi, [16](#)  
genel tip, [16](#)  
generator, [15](#)  
generator expression, [16](#)  
geri çağırmak, [13](#)  
GIL, [16](#)

## H

haritalama, [18](#)

**I**

iç içe kapsam, **19**  
içe aktarıcı, **17**  
içe aktarım yolu, **16**  
içe aktarma, **17**  
ifade (*değer döndürmez*), **22**  
ifade (*değer döndürür*), **14**  
ikili dosya, **12**

**J**

jeneratör, **15**  
jeneratör ifadesi, **16**  
jeneratör yineleyici, **15**

**K**

karma tabanlı pyc, **16**  
karmaşık sayı, **13**  
kat bölümü, **15**  
kısım, **20**  
konumsal argüman, **20**

**L**

lambda, **18**  
LBYL, **18**  
liste, **18**  
liste anlama, **18**

**M**

magic  
    method, **18**  
meta yol bulucu, **18**  
metasınıf, **18**  
method  
    magic, **18**  
    special, **22**  
metot, **18**  
metot kalite sıralaması, **18**  
modül, **18**  
modül özelliği, **18**  
MRO, **18**

**N**

nitelik, **12**  
nitelikli isim, **21**

**O**

obje, **19**

**Ö**

özel metod, **22**

**P**

paket, **19**

parametre, **19**  
parçalamak, **22**  
PEP, **20**  
Python 3000, **21**  
Python'ı iyileştirme Önerileri  
    PEP 1, **20**  
    PEP 238, **15**  
    PEP 278, **23**  
    PEP 302, **15, 18**  
    PEP 343, **13**  
    PEP 362, **12, 20**  
    PEP 411, **21**  
    PEP 420, **15, 19, 20**  
    PEP 443, **16**  
    PEP 451, **15**  
    PEP 483, **16**  
    PEP 484, **11, 15, 16, 23**  
    PEP 492, **12, 13**  
    PEP 498, **15**  
    PEP 519, **20**  
    PEP 525, **12**  
    PEP 526, **11, 23**  
    PEP 585, **16**  
    PEP 3116, **23**  
    PEP 3155, **21**

Pythonic, **21**  
Python'un Zen'i, **23**

**R**

referans sayısı, **21**

**S**

sanal makine, **23**  
sanal ortam, **23**  
sınıf, **13**  
sınıf değişkeni, **13**  
sihirli yöntem, **18**  
soyut temel sınıf, **11**  
sözlük, **14**  
sözlük anlama, **14**  
sözlük görünümü, **14**  
special  
    method, **22**  
sürekli paketleme, **21**

**T**

tanımlayıcı, **14**  
tek sevk, **22**  
tercüman kapatma, **17**  
tip, **22**  
tip takma adı, **22**  
tür ipucu, **23**

## U

uzatma modülü, [14](#)

## Ü

üç tırnaklı dize, [22](#)

## Y

yazı çözümleme, [22](#)

yazı dosyası, [22](#)

yeni stil sınıf, [19](#)

yıkanabilir, [16](#)

yinelenebilir, [17](#)

yineleyici, [17](#)

yol benzeri nesne, [20](#)

yol giriş kancası, [20](#)

yol girişi, [20](#)

yol girişi bulucu, [20](#)

yol tabanlı bulucu, [20](#)

yorumlanmış, [17](#)

yükleyici, [18](#)

## Z

zorlama, [13](#)