

---

# Installing Python Modules

*Yayım 3.14.0rc3*

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

**Ekim 01, 2025**

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







<b>1</b>	<b>Anahtar terimler</b>	<b>3</b>
<b>2</b>	<b>Temel kullanım</b>	<b>5</b>
<b>3</b>	<b>Nasıl yapabilirim ...?</b>	<b>7</b>
3.1	... pip 'i Python 3.4'ten önceki Python sürümlerinde kurmalı mı?	7
3.2	... sadece mevcut kullanıcı için paketler kurabilirim?	7
3.3	... bilimsel Python paketleri kurabilirim?	7
3.4	... paralel olarak yüklenmiş birden çok Python sürümüyle çalışabilirim?	7
<b>4</b>	<b>Genel yükleme sorunları</b>	<b>9</b>
4.1	Python'u Linux'te sisteme kurmak	9
4.2	Pip yüklü değil	9
4.3	İkili uzantıları yükleme	9
<b>A</b>	<b>Sözlük</b>	<b>11</b>
<b>B</b>	<b>About this documentation</b>	<b>29</b>
B.1	Contributors to the Python documentation	29
<b>C</b>	<b>Tarihçe ve Lisans</b>	<b>31</b>
C.1	Yazılımın tarihçesi	31
C.2	Python'a erişmek veya başka bir şekilde kullanmak için şartlar ve koşullar	32
C.2.1	PYTHON SOFTWARE FOUNDATION LICENSE VERSION 2	32
C.2.2	PYTHON 2.0 İÇİN BEOPEN.COM LİSANS SÖZLEŞMESİ	33
C.2.3	PYTHON 1.6.1 İÇİN CNRI LİSANS ANLAŞMASI	34
C.2.4	0.9.0 ARASI 1.2 PYTHON İÇİN CWI LİSANS SÖZLEŞMESİ	35
C.2.5	ZERO-CLAUSE BSD LICENSE FOR CODE IN THE PYTHON DOCUMENTATION	35
C.3	Tüzel Yazılımlar için Lisanslar ve Onaylar	35
C.3.1	Mersenne Twister'i	35
C.3.2	Soketler	36
C.3.3	Asenkron soket hizmetleri	37
C.3.4	Çerez yönetimi	37
C.3.5	Çalıştırma izleme	38
C.3.6	UUencode ve UUdecode fonksiyonları	38
C.3.7	XML Uzaktan Yordam Çağrılar	39
C.3.8	test_epoll	39
C.3.9	kqueue seçin	40
C.3.10	SipHash24	40
C.3.11	strtod ve dtoa	41
C.3.12	OpenSSL	41



C.3.13	expat . . . . .	45
C.3.14	libffi . . . . .	45
C.3.15	zlib . . . . .	46
C.3.16	cfuhash . . . . .	46
C.3.17	libmpdec . . . . .	47
C.3.18	W3C C14N test paketi . . . . .	47
C.3.19	mimalloc . . . . .	48
C.3.20	asyncio . . . . .	48
C.3.21	Global Unbounded Sequences (GUS) . . . . .	49
C.3.22	Zstandard bindings . . . . .	49
<b>D</b>	<b>Telif Hakkı</b>	<b>51</b>
	<b>Dizin</b>	<b>53</b>



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

Popüler bir açık kaynak geliştirme projesi olarak Python, yazılımlarını diğer Python geliştiricilerinin açık kaynak lisansı koşulları altında kullanmasını sağlayan, katkıda bulunanlardan ve kullanıcılardan oluşan aktif bir destekleyici topluluğa sahiptir.

Bu, Python kullanıcılarının etkin bir şekilde paylaşımda bulunmasına ve işbirliği yapmasına, başkalarının yaygın (ve hatta bazen nadir!) sorunlara yönelik oluşturduğu çözümlerden faydalanmasına ve potansiyel olarak ortak havuza kendi çözümleriyle katkıda bulunmasına olanak tanır.

This guide covers the installation part of the process. For a guide to creating and sharing your own Python projects, refer to the [Python packaging user guide](#).

**i Not**

Kurumsal kullanıcılar, birçok kuruluşun açık kaynak yazılımları kullanma ve bunlara katkıda bulunma konusunda kendi politikaları olduğunu unutmayın. Python ile sağlanan dağıtım ve kurulum araçlarını kullanırken lütfen bu tür politikaları dikkate alın.







## Anahtar terimler

- `pip`, tercih edilen yükleyici programıdır. Python 3.4'ten başlayarak, varsayılan olarak Python ikili yükleyicilerine dahil edilmiştir.
- *virtual environment*, paketlerin sistem genelinde kurulmak yerine belirli bir uygulama tarafından kullanılmak üzere kurulmasına izin veren yarı izole bir Python ortamıdır.
- `venv`, sanal ortamlar oluşturmaya yönelik standart bir araçtır ve Python 3.3'ten beri Python'un bir parçasıdır. Python 3.4'ten başlayarak, `pip` 'in tüm oluşturulan sanal ortamlara yüklenmesi varsayılandır.
- `virtualenv`, `venv` 'in üçüncü taraf alternatifidir (ve öncülüdür). `venv` 'i hiç sağlamayan veya oluşturulan ortamlara `pip` 'i otomatik olarak yükleyemeyen 3.4'ten önceki Python sürümlerinde sanal ortamların kullanılmasına izin verir.
- [Python Package Index](#), diğer Python kullanıcılarının kullanımına sunulan, açık kaynaklı lisanslı paketlerin halka açık bir deposudur.
- 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 [GitHub](#).
- `distutils`, Python standart kitaplığına ilk olarak 1998 yılında eklenen orijinal oluşturma ve dağıtım sistemidir. `distutils` 'in doğrudan kullanımı aşamalı olarak kaldırılırken, mevcut paketleme ve dağıtım altyapısının temelini atmaya devam etmektedir, ve yalnızca standard kütüphanenin bir parçası olmakla kalmamakla birlikte adı başka şekillerde de yaşıyor (Python paketleme standartlarının geliştirilmesini koordine etmek için kullanılan posta listesinin adı gibi).

3.5 sürümünde değişti: Artık sanal ortamlar oluşturmak için `venv` 'in kullanılması önerilir.

### Ayrıca bakınız

Python Paketleme Kullanıcı Kılavuzu: Sanal ortamları oluşturma ve kullanma







### Temel kullanım

Standart paketleme araçlarının tümü, komut satırından kullanılmak üzere tasarlanmıştır.

Aşağıdaki komut, bir modülün en son sürümünü ve bağımlılıklarını Python Paket Dizininden yükleyecektir:

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

#### Not

POSIX kullanıcıları için (macOS ve Linux kullanıcıları dahil), bu kılavuzdaki örneklerde bir *virtual environment* kullanıldığı varsayılmıştır.

Windows kullanıcıları için bu kılavuzdaki örneklerde, Python yüklenirken sistem PATH ortam değişkenini ayarlama seçeneğinin seçildiği varsayılmaktadır.

Doğrudan komut satırında tam veya minimum bir sürüm belirtmek de mümkündür. >, < gibi karşılaştırma operatörleri veya kabuk tarafından yorumlanan diğer bazı özel karakterler kullanılırken, paket adı ve sürüm çift tırnak içine alınmalıdır:

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

Normalde, uygun bir modül zaten kuruluysa, onu tekrar kurmayı denemenin bir etkisi olmaz. Mevcut modüllerin yükseltilmesi açıkça talep edilmelidir:

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

pip ve yapabilecekleri hakkında daha fazla bilgi ve kaynak [Python Paketleme Kullanıcı Kılavuzu](#) 'nda bulunabilir.

Sanal ortamların oluşturulması, `venv` modülü aracılığıyla yapılır. Paketleri aktif bir sanal ortama kurmak, yukarıda gösterilen komutları kullanır.

#### Ayrıca bakınız

[Python Paketleme Kullanıcı Kılavuzu: Python Dağıtma Paketlerini Kurma](#)







### Nasıl yapabilirim ...?

Bunlar, bazı genel görevler için hızlı cevaplar veya bağlantılardır.

#### 3.1 ... `pip` 'i Python 3.4'ten önceki Python sürümlerinde kurmalı mı?

Python, `pip` 'i yalnızca Python 3.4 ile paketlemeye başladı. Önceki sürümler için, `pip` 'in Python Paketleme Kullanıcı Kılavuzu'nda açıklandığı gibi “önyüklenmesi” gerekir.

##### Ayrıca bakınız

Python Paketleme Kullanıcı Kılavuzu: Paketleri Kurmak için Gerekenler

#### 3.2 ... sadece mevcut kullanıcı için paketler kurabilirim?

`--user` seçeneğinin `python -m pip install` 'e geçirilmesi, sistemin tüm kullanıcıları yerine yalnızca geçerli kullanıcı için bir paket yükleyecektir.

#### 3.3 ... bilimsel Python paketleri kurabilirim?

Bir dizi bilimsel Python paketinin karmaşık ikili bağımlılıkları vardır ve şu anda doğrudan `pip` kullanılarak kurulumu kolay değildir. Bu noktada, kullanıcıların bu paketleri `pip` ile kurmaya çalışmak yerine [başka yollar](#) ile kurması genellikle daha kolay olacaktır.

##### Ayrıca bakınız

Python Paketleme Kullanıcı Kılavuzu: Bilimsel Paketler Kurma

#### 3.4 ... paralel olarak yüklenmiş birden çok Python sürümünü çalıştırabilir miyim?

Linux, macOS ve diğer POSIX sistemlerinde, `pip` 'in uygun kopyasını çalıştırmak için `-m` anahtarıyla birlikte sürümlü Python komutlarını kullanın:



```
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
```

Uygun sürümüne sahip `pip` komutları da mevcut olabilir.

Windows'ta, `py` Python başlatıcısını `-m` anahtarıyla birlikte kullanın:

```
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
```



## Genel yükleme sorunları

## 4.1 Python'u Linux'te sisteme kurmak

Linux sistemlerinde, dağıtımın bir parçası olarak genellikle bir Python kurulumu dahil edilir. Bu Python kurulumuna paket yüklemek, sisteme root erişimi gerektirir ve bir bileşen beklenmedik bir şekilde `pip` kullanılarak güncellenirse, sistem paketi yöneticisinin ve sistemin diğer bileşenlerinin çalışmasına müdahale edebilir.

Bu tür sistemlerde, `pip` ile paketleri kurarken sanal ortam veya kullanıcı başına kurulum kullanmak genellikle daha iyidir.

## 4.2 Pip yüklü değil

`pip` 'in varsayılan olarak yüklenmemesi mümkündür. Bir olası düzeltme:

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

`pip`'i yüklemek için ek kaynaklar da vardır.

## 4.3 İkili uzantıları yükleme

Python tipik olarak büyük ölçüde kaynak tabanlı dağıtıma güvenmiştir ve son kullanıcıların kurulum sürecinin bir parçası olarak uzantı modüllerini kaynaktan derlemesi beklenir.

İkili `wheel` formatı için desteğin sunulması ve Python Paket Dizini aracılığıyla en azından Windows ve macOS için `wheel`'lar yayınlanabilmesi ile, kullanıcılar önceden oluşturulmuş uzantıları kendileri inşa etmek yerine daha düzenli bir şekilde yükleyebildiğinden bu sorunun zaman içinde azalması bekleniyor.

**Bilimsel yazılımı** yüklemek için önceden oluşturulmuş (pre-built) `wheel` dosyaları olarak henüz mevcut olmayan çözümlerden bazıları, diğer ikili uzantıları yerel olarak kurmaya gerek kalmadan edinmeye de yardımcı olabilir.

### ➔ Ayrıca bakınız

Python Paketleme Kullanıcı Kılavuzu: İkili Uzantılar







&gt;&gt;&gt;

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

...

Şunlara başvurulabilir:

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

### 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.

### annotate function

A function that can be called to retrieve the *annotations* of an object. This function is accessible as the `__annotate__` attribute of functions, classes, and modules. Annotate functions are a subset of *evaluate functions*.

### dipnot

Bir değişkenle, bir sınıf niteliğiyle veya bir fonksiyon parametresiyle veya bir dönüş değeriyle ilişkilendirilen, gelenek olarak *type hint* biçiminde kullanılan bir etiket.

Annotations of local variables cannot be accessed at runtime, but annotations of global variables, class attributes, and functions can be retrieved by calling `annotationlib.get_annotations()` on modules, classes, and functions, respectively.

See *variable annotation*, *function annotation*, **PEP 484**, **PEP 526**, and **PEP 649**, which describe this functionality. Also see *annotations-howto* for best practices on working with annotations.

### argüman

Fonksiyon çağrılırken bir *function* 'a (veya *method*) geçirilen bir değer. İki tür argüman vardır:



- *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 anahtar 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 fonksiyon 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.

Ayrıca *parameter* sözlüğü girişine, the difference between arguments and parameters hakkındaki SSS sorusuna ve **PEP 362** 'ye bakın.

### asenkron bağlam yöneticisi

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

### 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 üreteç 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

An object created by an *asynchronous generator* function.

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

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

### eşzamansız yinelenbilir

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

### 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**.

### attached thread state

A *thread state* that is active for the current OS thread.

When a *thread state* is attached, the OS thread has access to the full Python C API and can safely invoke the bytecode interpreter.

Unless a function explicitly notes otherwise, attempting to call the C API without an attached thread state will result in a fatal error or undefined behavior. A thread state can be attached and detached explicitly by the user through the C API, or implicitly by the runtime, including during blocking C calls and by the bytecode interpreter in between calls.



On most builds of Python, having an attached thread state implies that the caller holds the *GIL* for the current interpreter, so only one OS thread can have an attached thread state at a given moment. In *free-threaded* builds of Python, threads can concurrently hold an attached thread state, allowing for true parallelism of the bytecode interpreter.

#### nitelik

Noktalı ifadeler kullanılarak adıyla başvurulmuş bir nesneyle ilişkili değer. Örneğin, *o* nesnesinin *a* özneliği varsa, bu nesneye *o.a* olarak başvurulur.

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

#### beklenebilir

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

#### BDFL

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

#### ikili dosya

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

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

#### ödünç alınan referans

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

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

#### bayt benzeri nesne

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

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

#### bayt kodu

Python kaynak kodu, bir Python programının CPython yorumlayıcısındaki dahili temsili olan bayt kodunda derlenir. Bayt kodu ayrıca `.pyc` dosyalarında ön belleğe alınır, böylece aynı dosyanın ikinci kez çalıştırılması daha hızlı olur (kaynak kodu 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.

#### çağrılabilir

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

```
callable(argument1, argument2, argumentN)
```

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



**geri çağırmak**

Gelecekte bir noktada yürütülecek bir argüman olarak iletilen bir alt program işlevi.

**sınıf**

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

**sınıf değişkeni**

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

**closure variable**

A *free variable* referenced from a *nested scope* that is defined in an outer scope rather than being resolved at runtime from the globals or builtin namespaces. May be explicitly defined with the `nonlocal` keyword to allow write access, or implicitly defined if the variable is only being read.

For example, in the `inner` function in the following code, both `x` and `print` are *free variables*, but only `x` is a *closure variable*:

```
def outer():
    x = 0
    def inner():
        nonlocal x
        x += 1
        print(x)
    return inner
```

Due to the `codeobject.co_freevars` attribute (which, despite its name, only includes the names of closure variables rather than listing all referenced free variables), the more general *free variable* term is sometimes used even when the intended meaning is to refer specifically to closure variables.

**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 eş değerlerine erişmek için `cmath` kullanın. Karmaşık sayıların kullanımı oldukça gelişmiş bir matematiksel özelliktir. Onlara olan ihtiyacın farkında değilseniz, onları güvenle görmezden gelebileceğiniz neredeyse kesindir.

**context**

This term has different meanings depending on where and how it is used. Some common meanings:

- The temporary state or environment established by a *context manager* via a `with` statement.
- The collection of keyvalue bindings associated with a particular `contextvars.Context` object and accessed via `ContextVar` objects. Also see *context variable*.
- A `contextvars.Context` object. Also see *current context*.

**context management protocol**

The `__enter__()` and `__exit__()` methods called by the `with` statement. See [PEP 343](#).

**bağlam yöneticisi**

An object which implements the *context management protocol* and controls the environment seen in a `with` statement. See [PEP 343](#).

**bağlam değişkeni**

A variable whose value depends on which context is the *current context*. Values are accessed via `contextvars.ContextVar` objects. Context variables are primarily used to isolate state between concurrent asynchronous tasks.

**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, altyordamların daha genelleştirilmiş bir biçimidir. Alt programlara bir noktada girilir ve başka bir noktada çıkılır. Eşyordamlar birçok farklı noktada girilebilir, çıkılabilir ve devam ettirilebilir. `async def` ifadesi ile uygulanabilirler. Ayrıca bakınız [PEP 492](#).

### eşyordam işlevi

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

### CPython

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

### current context

The *context* (`contextvars.Context` object) that is currently used by `ContextVar` objects to access (get or set) the values of *context variables*. Each thread has its own current context. Frameworks for executing asynchronous tasks (see `asyncio`) associate each task with a context which becomes the current context whenever the task starts or resumes execution.

### cyclic isolate

A subgroup of one or more objects that reference each other in a reference cycle, but are not referenced by objects outside the group. The goal of the *cyclic garbage collector* is to identify these groups and break the reference cycles so that the memory can be reclaimed.

### 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 eş değerdir:

```
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

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

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

### sözlük

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

### 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 görünümleri 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

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

### ördek yazma

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.

### dunder

An informal short-hand for “double underscore”, used when talking about a *special method*. For example, `__init__` is often pronounced “dunder init”.

### 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.

### evaluate function

A function that can be called to evaluate a lazily evaluated attribute of an object, such as the value of type aliases created with the `type` statement.

### ifade (değer döndürür)

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

### uzatma modülü

Çekirdek ve kullanıcı koduyla etkileşim kurmak için Python'un C API'sini kullanan, C veya C++ ile yazılmış bir modül.

### f-string

#### f-strings

String literals prefixed with `f` or `F` are commonly called “f-strings” which is short for formatted string literals. See also [PEP 498](#).

### dosya nesnesi

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

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

### dosya benzeri nesne

*dosya nesnesi* ile eşanlamlıdır.

### dosya sistemi kodlaması ve hata işleyicisi

Python tarafından işletim sistemindeki baytların kodunu çözmek ve Unicode'ü işletim sistemine kodlamak için kullanılan kodlama ve hata işleyici.



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

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

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

Ayrıca bkz. *locale encoding*.

### bulucu

İçe aktarılmakta olan bir modül için *loader* 'ı bulmaya çalışan bir nesne.

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

See *finders-and-loaders* and `importlib` for much more detail.

### 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**.

### free threading

A threading model where multiple threads can run Python bytecode simultaneously within the same interpreter. This is in contrast to the *global interpreter lock* which allows only one thread to execute Python bytecode at a time. See **PEP 703**.

### free variable

Formally, as defined in the language execution model, a free variable is any variable used in a namespace which is not a local variable in that namespace. See *closure variable* for an example. Pragmatically, due to the name of the `codeobject.co_freevars` attribute, the term is also sometimes used as a synonym for *closure variable*.

### fonksiyon

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

### fonksiyon açıklaması

Bir işlev parametresinin veya dönüş değerinin *ek açıklaması*.

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

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

İşlev açıklama sözdizimi *function* bölümünde açıklanmaktadır.

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

### \_\_future\_\_

Bir *future* ifadesi, `from __future__ import <feature>`, derleyiciyi, Python'un gelecekteki bir sürümünde standart hale gelecek olan sözdizimini veya semantiği kullanarak mevcut modülü derlemeye yönlendirir. `__future__` modülü, *feature*'in olası değerlerini belgeler. Bu modülü içe 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 kı-



rabilen 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.

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

### jeneratör ifadesi

An *expression* that returns an *iterator*. It looks like a normal expression followed by a `for` clause defining a loop variable, range, and an optional `if` clause. The combined expression generates values for an enclosing function:

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

### genel işlev

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

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

### genel tip

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

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

### GIL

Bakınız *global interpreter lock*.

### genel tercüman kilidi

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

Bununla birlikte, standart veya üçüncü taraf bazı genişletme modülleri, sıkıştırma veya karma gibi hesaplama açısından yoğun görevler yaparken GIL’yi serbest bırakacak şekilde tasarlanmıştır. Ayrıca, GIL, G/Ç yaparken her zaman serbest bırakılır.

As of Python 3.13, the GIL can be disabled using the `--disable-gil` build configuration. After building Python with this option, code must be run with `-X gil=0` or after setting the `PYTHON_GIL=0` environment variable. This feature enables improved performance for multi-threaded applications and makes it easier to use multi-core CPUs efficiently. For more details, see **PEP 703**.

In prior versions of Python’s C API, a function might declare that it requires the GIL to be held in order to use it. This refers to having an *attached thread state*.

### karma tabanlı pyc

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

### yıkanabilir

An object is *hashable* if it has a hash value which never changes during its lifetime (it needs a `__hash__()`



method), and can be compared to other objects (it needs an `__eq__()` method). Hashable objects which compare equal must have the same hash value.

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

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

## BOŞTA

Python için Entegre Geliştirme Ortamı. idle, Python'un standart dağıtımıyla birlikte gelen temel bir düzenleyici ve yorumlayıcı ortamıdır.

## immortal

*Immortal objects* are a CPython implementation detail introduced in [PEP 683](#).

If an object is immortal, its *reference count* is never modified, and therefore it is never deallocated while the interpreter is running. For example, `True` and `None` are immortal in CPython.

Immortal objects can be identified via `sys._is_immortal()`, or via `PyUnstable_IsImmortal()` in the C API.

## değişmez

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

## içe aktarım yolu

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

## içe aktarma

Bir modüldeki Python kodunun başka bir modüldeki Python koduna sunulması süreci.

## içe aktarıcı

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

## etkileşimli

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

## yorumlanmış

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

## tercüman kapatma

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

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

## yinelenebilir

An object capable of returning its members one at a time. Examples of iterables include all sequence types (such



as `list`, `str`, and `tuple`) and some non-sequence types like `dict`, *file objects*, and objects of any classes you define with an `__iter__()` method or with a `__getitem__()` method that implements *sequence* semantics.

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

### yineleyici

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

Daha fazla bilgi `typeiter` içinde bulunabilir.

**CPython uygulama ayrıntısı:** CPython does not consistently apply the requirement that an iterator define `__iter__()`. And also please note that the free-threading CPython does not guarantee the thread-safety of iterator operations.

### 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 gruplandırıldığını kontrol etmek için temel işlevleri kabul eder. Bunlar `min()`, `max()`, `sorted()`, `list.sort()`, `heapq.merge()`, `heapq.nsmallest()`, `heapq.nlargest()` ve `itertools.groupby()`.

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

### anahtar kelime argümanı

Bakınız *argument*.

### lambda

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

### LBYL

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

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

### lexical analyzer

Formal name for the *tokenizer*; see *token*.

### liste

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



**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. Atlarsa, “`aralık(256)`” içindeki tüm öğeler işlenir.

**yükleyici**

An object that loads a module. It must define the `exec_module()` and `create_module()` methods to implement the `Loader` interface. A loader is typically returned by a *finder*. See also:

- `finders-and-loaders`
- `importlib.abc.Loader`
- **PEP 302**

**yerel kodlama**

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

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

Android ve VxWorks'te Python, yerel kodlama olarak "`utf-8`" kullanır.

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

Ayrıca *filesystem encoding and error handler* 'ne bakın.

**sihirli yöntem**

*special method* için gayri resmi bir eşanlamlı.

**haritalama**

Keyfi anahtar aramalarını destekleyen ve `Mapping` veya `MutableMapping` `collections-abstract-base-classes` içinde belirtilen yöntemleri uygulayan bir kapsayıcı nesnesi. 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'nın mümkün olmasıdır. Çoğu kullanıcı bu araca hiçbir zaman ihtiyaç duymaz, ancak ihtiyaç duyulduğunda, metasınıflar güçlü ve zarif çözümler sağlayabilir. Nitelik erişimini günlüğe kaydetmek, iş parçacığı güvenliği eklemek, nesne oluşturmaya izlemek, tekilleri uygulamak ve diğer birçok görev için kullanılmışlardır.

Daha fazla bilgi metaclasses içinde bulunabilir.

**metot**

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

**metot kalite sıralaması**

Method Resolution Order is the order in which base classes are searched for a member during lookup. See `python_2.3_mro` for details of the algorithm used by the Python interpreter since the 2.3 release.

**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.

See also `module-specs`.

### 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
```

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

### 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 *package* which serves only as a container for subpackages. Namespace packages may have no physical representation, and specifically are not like a *regular package* because they have no `__init__.py` file.

Namespace packages allow several individually installable packages to have a common parent package. Otherwise, it is recommended to use a *regular package*.

For more information, see **PEP 420** and `reference-namespace-package`.

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

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



**obje**

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

**optimized scope**

A scope where target local variable names are reliably known to the compiler when the code is compiled, allowing optimization of read and write access to these names. The local namespaces for functions, generators, coroutines, comprehensions, and generator expressions are optimized in this fashion. Note: most interpreter optimizations are applied to all scopes, only those relying on a known set of local and nonlocal variable names are restricted to optimized scopes.

**paket**

Alt modüller veya yinelemeli olarak alt paketler içerebilen bir Python *module*. Teknik olarak bir paket, `__path__` özneliğine sahip bir Python modülüdür.

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

**parametre**

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

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

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

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

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

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

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

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

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

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

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

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

**yol girişi**

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

**yol girişi bulucu**

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

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



### yol giriş kancası

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

### 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**

The number of references to an object. When the reference count of an object drops to zero, it is deallocated. Some objects are *immortal* and have reference counts that are never modified, and therefore the objects are never deallocated. Reference counting is generally not visible to Python code, but it is a key element of the *CPython* implementation. Programmers can call the `sys.getrefcount()` function to return the reference count for a particular object.

In *CPython*, reference counts are not considered to be stable or well-defined values; the number of references to an object, and how that number is affected by Python code, may be different between versions.

**sürekli paketleme**

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

Ayrıca bkz. *ad alanı paketi*.

**REPL**

An acronym for the “read–eval–print loop”, another name for the *interactive* interpreter shell.

**\_\_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**

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

The `collections.abc.Sequence` abstract base class defines a much richer interface that goes beyond just `__getitem__()` and `__len__()`, adding `count()`, `index()`, `__contains__()`, and `__reversed__()`. Types that implement this expanded interface can be registered explicitly using `register()`. For more documentation on sequence methods generally, see Common Sequence Operations.

**anlamak**

Öğelerin tümünü veya bir kısmını yinelenabilir 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.

### soft deprecated

A soft deprecated API should not be used in new code, but it is safe for already existing code to use it. The API remains documented and tested, but will not be enhanced further.

Soft deprecation, unlike normal deprecation, does not plan on removing the API and will not emit warnings.

See [PEP 387: Soft Deprecation](#).

### 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.

### standard library

The collection of *packages*, *modules* and *extension modules* distributed as a part of the official Python interpreter package. The exact membership of the collection may vary based on platform, available system libraries, or other criteria. Documentation can be found at [library-index](#).

See also `sys.stdlib_module_names` for a list of all possible standard library module names.

### 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.

### static type checker

An external tool that reads Python code and analyzes it, looking for issues such as incorrect types. See also [type hints](#) and the `typing` module.

### stdlib

An abbreviation of *standard library*.

### güçlü referans

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

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

Ayrıca bkz. *ödünç alınan referans*.

### t-string

#### t-strings

String literals prefixed with `t` or `T` are commonly called “t-strings” which is short for template string literals.

### 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 kodları 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.

### thread state

The information used by the *CPython* runtime to run in an OS thread. For example, this includes the current exception, if any, and the state of the bytecode interpreter.

Each thread state is bound to a single OS thread, but threads may have many thread states available. At most, one of them may be *attached* at once.

An *attached thread state* is required to call most of Python's C API, unless a function explicitly documents otherwise. The bytecode interpreter only runs under an attached thread state.

Each thread state belongs to a single interpreter, but each interpreter may have many thread states, including multiple for the same OS thread. Thread states from multiple interpreters may be bound to the same thread, but only one can be *attached* in that thread at any given moment.

See Thread State and the Global Interpreter Lock for more information.

### token

A small unit of source code, generated by the lexical analyzer (also called the *tokenizer*). Names, numbers, strings, operators, newlines and similar are represented by tokens.

The `tokenize` module exposes Python's lexical analyzer. The `token` module contains information on the various types of tokens.

### 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

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

### 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*.

Type hints are optional and are not enforced by Python but they are useful to *static type checkers*. They can also aid IDEs with code completion and refactoring.

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 özneliğinin *ek açıklaması*.

Bir değişkene veya sınıf niteliğine açıklama eklerken atama isteğe bağlıdır:

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

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

```
count: int = 0
```

Değişken açıklama sözdizimi `annassign` bölümünde açıklanmıştır.

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

### sanal ortam

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

Ayrıca bakınız `venv`.

### sanal makine

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

### walrus operator

A light-hearted way to refer to the assignment expression operator `:` because it looks a bit like a walrus if you turn your head.

### Python'un Zen'i

Dili anlamaya ve kullanmaya yardımcı olan Python tasarım ilkeleri ve felsefelerinin listesi. Liste, etkileşimli komut isteminde `"import this"` yazarak bulunabilir.



---

## About this documentation

---

Python's documentation is generated from [reStructuredText](#) sources using [Sphinx](#), a documentation generator originally created for Python and now maintained as an independent project.

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., the creator of the original Python documentation toolset and author of much of the content;
- [Docutils](#) projesi, [reStructuredText](#) ve [Docutils](#) paketini oluşturdıkları için;
- Fredrik Lundh, Sphinx'in pek çok iyi fikir edindiği Alternatif Python Referansı projesi için.

## B.1 Contributors to the Python documentation

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

Python topluluğunun girdileri ve katkıları sayesinde böyle harika bir dokümantasyona sahibiz – Teşekkürler!







## Tarihçe ve Lisans

## C.1 Yazılımın tarihçesi

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

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

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

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

Yayın	Şundan türedi:	Yıl	Sahibi	GPL-compatible? (1)
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	yes (2)
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

**i Not**

- (1) GPL-compatible doesn't mean that we're distributing Python under the GPL. All Python licenses, unlike the GPL, let you distribute a modified version without making your changes open source. The GPL-compatible



licenses make it possible to combine Python with other software that is released under the GPL; the others don't.

- (2) According to Richard Stallman, 1.6.1 is not GPL-compatible, because its license has a choice of law clause. According to CNRI, however, Stallman's lawyer has told CNRI's lawyer that 1.6.1 is "not incompatible" with the GPL.

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 software and documentation are licensed under the Python Software Foundation License Version 2.

Starting with Python 3.8.6, examples, recipes, and other code in the documentation are dual licensed under the PSF License Version 2 and the *Zero-Clause BSD license*.

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 SOFTWARE FOUNDATION LICENSE VERSION 2

1. This LICENSE AGREEMENT is between the Python Software Foundation ("PSF"), and the Individual or Organization ("Licensee") accessing and otherwise using this software ("Python") 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 alone or in any derivative version, provided, however, that PSF's License Agreement and PSF's notice of copyright, i.e., "Copyright © 2001 Python Software Foundation; All Rights Reserved" are retained in Python 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 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.
4. PSF is making Python 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 WILL NOT INFRINGE ANY THIRD PARTY RIGHTS.
5. PSF SHALL NOT BE LIABLE TO LICENSEE OR ANY OTHER USERS OF PYTHON FOR ANY INCIDENTAL, SPECIAL, OR CONSEQUENTIAL DAMAGES OR LOSS AS A RESULT OF MODIFYING, DISTRIBUTING, OR OTHERWISE USING PYTHON, 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

(sonraki sayfaya devam)



(önceki sayfadan devam)

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, 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.
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 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.

(sonraki sayfaya devam)



(önceki sayfadan devam)

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 ZERO-CLAUSE BSD LICENSE FOR CODE IN THE PYTHON DOCUMENTATION

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

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

## 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` modülünün altyapısını oluşturan `_random` C uzantısı, <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)`

(sonraki sayfaya devam)



(önceki sayfadan devam)

```
or init_by_array(init_key, key_length).
```

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

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

1. Redistributions of source code must retain the above copyright  
notice, this list of conditions and the following disclaimer.
2. Redistributions in binary form must reproduce the above copyright  
notice, this list of conditions and the following disclaimer in the  
documentation and/or other materials provided with the distribution.
3. The names of its contributors may not be used to endorse or promote  
products derived from this software without specific prior written  
permission.

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

Any feedback is very welcome.

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

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

### C.3.2 Soketler

socket modülü, <https://www.wide.ad.jp/> adresindeki WIDE Projesi'nden ayrı kaynak dosyalarında kodlanan  
getaddrinfo() ve getnameinfo() fonksiyonlarını kullanır.

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.

(sonraki sayfaya devam)



(önceki sayfadan devam)

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

The `test.support.asyncchat` and `test.support.asyncore` modules contain the following notice:

Copyright 1996 by Sam Rushing

All Rights Reserved

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

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

### C.3.4 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

(sonraki sayfaya devam)



(önceki sayfadan devam)

```
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.  
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

The uu codec contains the following notice:

```
Copyright 1994 by Lance Ellinghouse  
Cathedral City, California Republic, United States of America.  
All Rights Reserved  
Permission to use, copy, modify, and distribute this software and its  
documentation for any purpose and without fee is hereby granted,  
provided that the above copyright notice appear in all copies and that  
both that copyright notice and this permission notice appear in  
supporting documentation, and that the name of Lance Ellinghouse  
not be used in advertising or publicity pertaining to distribution  
of the software without specific, written prior permission.  
LANCE ELLINGHOUSE DISCLAIMS ALL WARRANTIES WITH REGARD TO  
THIS SOFTWARE, INCLUDING ALL IMPLIED WARRANTIES OF MERCHANTABILITY AND  
FITNESS, IN NO EVENT SHALL LANCE ELLINGHOUSE CENTRUM BE LIABLE
```

(sonraki sayfaya devam)



(önceki sayfadan devam)

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 MERCHANTABILITY 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.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:

(sonraki sayfaya devam)



(önceki sayfadan devam)

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.

(sonraki sayfaya devam)



(önceki sayfadan devam)

&lt;/MIT License&gt;

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

C double'larının dizelere ve dizelerden dönüştürülmesi için `dtoa` ve `strtod` C fonksiyonlarını sağlayan `Python/dtoa.c` dosyası, şu anda <https://web.archive.org/web/20220517033456/http://www.netlib.org/fp/dtoa.c> 'den erişilebilen David M. Gay tarafından aynı adlı dosyadan türetilmiştir. 16 Mart 2009'da alınan orijinal dosya aşağıdaki telif hakkı ve lisans bildirimini içerir:

```

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

```

### C.3.12 OpenSSL

The modules `hashlib`, `posix` and `ssl` use the OpenSSL library for added performance if made available by the operating system. Additionally, the Windows and macOS installers for Python may include a copy of the OpenSSL libraries, so we include a copy of the OpenSSL license here. For the OpenSSL 3.0 release, and later releases derived from that, the Apache License v2 applies:

```

                Apache License
                Version 2.0, January 2004
                https://www.apache.org/licenses/

```

TERMS AND CONDITIONS FOR USE, REPRODUCTION, AND DISTRIBUTION

#### 1. Definitions.

"License" shall mean the terms and conditions for use, reproduction, and distribution as defined by Sections 1 through 9 of this document.

"Licensor" shall mean the copyright owner or entity authorized by the copyright owner that is granting the License.

(sonraki sayfaya devam)



(önceki sayfadan devam)

"Legal Entity" shall mean the union of the acting entity and all other entities that control, are controlled by, or are under common control with that entity. For the purposes of this definition, "control" means (i) the power, direct or indirect, to cause the direction or management of such entity, whether by contract or otherwise, or (ii) ownership of fifty percent (50%) or more of the outstanding shares, or (iii) beneficial ownership of such entity.

"You" (or "Your") shall mean an individual or Legal Entity exercising permissions granted by this License.

"Source" form shall mean the preferred form for making modifications, including but not limited to software source code, documentation source, and configuration files.

"Object" form shall mean any form resulting from mechanical transformation or translation of a Source form, including but not limited to compiled object code, generated documentation, and conversions to other media types.

"Work" shall mean the work of authorship, whether in Source or Object form, made available under the License, as indicated by a copyright notice that is included in or attached to the work (an example is provided in the Appendix below).

"Derivative Works" shall mean any work, whether in Source or Object form, that is based on (or derived from) the Work and for which the editorial revisions, annotations, elaborations, or other modifications represent, as a whole, an original work of authorship. For the purposes of this License, Derivative Works shall not include works that remain separable from, or merely link (or bind by name) to the interfaces of, the Work and Derivative Works thereof.

"Contribution" shall mean any work of authorship, including the original version of the Work and any modifications or additions to that Work or Derivative Works thereof, that is intentionally submitted to Licensor for inclusion in the Work by the copyright owner or by an individual or Legal Entity authorized to submit on behalf of the copyright owner. For the purposes of this definition, "submitted" means any form of electronic, verbal, or written communication sent to the Licensor or its representatives, including but not limited to communication on electronic mailing lists, source code control systems, and issue tracking systems that are managed by, or on behalf of, the Licensor for the purpose of discussing and improving the Work, but excluding communication that is conspicuously marked or otherwise designated in writing by the copyright owner as "Not a Contribution."

"Contributor" shall mean Licensor and any individual or Legal Entity on behalf of whom a Contribution has been received by Licensor and subsequently incorporated within the Work.

2. Grant of Copyright License. Subject to the terms and conditions of this License, each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable copyright license to reproduce, prepare Derivative Works of,

(sonraki sayfaya devam)



(önceki sayfadan devam)

publicly display, publicly perform, sublicense, and distribute the Work and such Derivative Works in Source or Object form.

3. Grant of Patent License. Subject to the terms and conditions of this License, each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable (except as stated in this section) patent license to make, have made, use, offer to sell, sell, import, and otherwise transfer the Work, where such license applies only to those patent claims licensable by such Contributor that are necessarily infringed by their Contribution(s) alone or by combination of their Contribution(s) with the Work to which such Contribution(s) was submitted. If You institute patent litigation against any entity (including a cross-claim or counterclaim in a lawsuit) alleging that the Work or a Contribution incorporated within the Work constitutes direct or contributory patent infringement, then any patent licenses granted to You under this License for that Work shall terminate as of the date such litigation is filed.
4. Redistribution. You may reproduce and distribute copies of the Work or Derivative Works thereof in any medium, with or without modifications, and in Source or Object form, provided that You meet the following conditions:
  - (a) You must give any other recipients of the Work or Derivative Works a copy of this License; and
  - (b) You must cause any modified files to carry prominent notices stating that You changed the files; and
  - (c) You must retain, in the Source form of any Derivative Works that You distribute, all copyright, patent, trademark, and attribution notices from the Source form of the Work, excluding those notices that do not pertain to any part of the Derivative Works; and
  - (d) If the Work includes a "NOTICE" text file as part of its distribution, then any Derivative Works that You distribute must include a readable copy of the attribution notices contained within such NOTICE file, excluding those notices that do not pertain to any part of the Derivative Works, in at least one of the following places: within a NOTICE text file distributed as part of the Derivative Works; within the Source form or documentation, if provided along with the Derivative Works; or, within a display generated by the Derivative Works, if and wherever such third-party notices normally appear. The contents of the NOTICE file are for informational purposes only and do not modify the License. You may add Your own attribution notices within Derivative Works that You distribute, alongside or as an addendum to the NOTICE text from the Work, provided that such additional attribution notices cannot be construed as modifying the License.

You may add Your own copyright statement to Your modifications and may provide additional or different license terms and conditions for use, reproduction, or distribution of Your modifications, or

(sonraki sayfaya devam)



for any such Derivative Works as a whole, provided Your use, reproduction, and distribution of the Work otherwise complies with the conditions stated in this License.

5. Submission of Contributions. Unless You explicitly state otherwise, any Contribution intentionally submitted for inclusion in the Work by You to the Licensor shall be under the terms and conditions of this License, without any additional terms or conditions. Notwithstanding the above, nothing herein shall supersede or modify the terms of any separate license agreement you may have executed with Licensor regarding such Contributions.
6. Trademarks. This License does not grant permission to use the trade names, trademarks, service marks, or product names of the Licensor, except as required for reasonable and customary use in describing the origin of the Work and reproducing the content of the NOTICE file.
7. Disclaimer of Warranty. Unless required by applicable law or agreed to in writing, Licensor provides the Work (and each Contributor provides its Contributions) on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied, including, without limitation, any warranties or conditions of TITLE, NON-INFRINGEMENT, MERCHANTABILITY, or FITNESS FOR A PARTICULAR PURPOSE. You are solely responsible for determining the appropriateness of using or redistributing the Work and assume any risks associated with Your exercise of permissions under this License.
8. Limitation of Liability. In no event and under no legal theory, whether in tort (including negligence), contract, or otherwise, unless required by applicable law (such as deliberate and grossly negligent acts) or agreed to in writing, shall any Contributor be liable to You for damages, including any direct, indirect, special, incidental, or consequential damages of any character arising as a result of this License or out of the use or inability to use the Work (including but not limited to damages for loss of goodwill, work stoppage, computer failure or malfunction, or any and all other commercial damages or losses), even if such Contributor has been advised of the possibility of such damages.
9. Accepting Warranty or Additional Liability. While redistributing the Work or Derivative Works thereof, You may choose to offer, and charge a fee for, acceptance of support, warranty, indemnity, or other liability obligations and/or rights consistent with this License. However, in accepting such obligations, You may act only on Your own behalf and on Your sole responsibility, not on behalf of any other Contributor, and only if You agree to indemnify, defend, and hold each Contributor harmless for any liability incurred by, or claims asserted against, such Contributor by reason of your accepting any such warranty or additional liability.

END OF TERMS AND CONDITIONS



### C.3.13 expat

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

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

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

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

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

### C.3.14 libffi

`ctypes` modülünün altyapısını oluşturan `_ctypes` C uzantısı, `--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.
```

```
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,
```

(sonraki sayfaya devam)



(önceki sayfadan devam)

```
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

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

```
Copyright (c) 2008–2020 Stefan Krah. All rights reserved.
```

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

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

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

### 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.

(sonraki sayfaya devam)



(önceki sayfadan devam)

\* Neither the name of the W3C nor the names of its contributors may be used to endorse or promote products derived from this work without specific prior written permission.

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

### C.3.19 mimalloc

MIT License:

Copyright (c) 2018–2021 Microsoft Corporation, Daan Leijen

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.3.20 asyncio

Parts of the `asyncio` module are incorporated from `uvloop 0.16`, which is distributed under the MIT license:

Copyright (c) 2015–2021 MagicStack Inc. <http://magic.io>

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.

(sonraki sayfaya devam)



(önceki sayfadan devam)

```
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.3.21 Global Unbounded Sequences (GUS)

The file `Python/qsbr.c` is adapted from FreeBSD's "Global Unbounded Sequences" safe memory reclamation scheme in `subr_smr.c`. The file is distributed under the 2-Clause BSD License:

```
Copyright (c) 2019,2020 Jeffrey Roberson <jeff@FreeBSD.org>
```

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 unmodified, 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 "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 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.3.22 Zstandard bindings

Zstandard bindings in `Modules/_zstd` and `Lib/compression/zstd` are based on code from the [pyzstd library](#), copyright Ma Lin and contributors. The `pyzstd` code is distributed under the 3-Clause BSD License:

```
Copyright (c) 2020-present, Ma Lin and contributors.
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 copyright holder nor the names of its

(sonraki sayfaya devam)



(önceki sayfadan devam)

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 HOLDER 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.



---

## Telif Hakkı

---

Python ve bu dokümantasyon:

Copyright © 2001 Python Software Foundation. All rights reserved.

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

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

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

---

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







## Alfabetik olmayan

..., [11](#)  
    ellipsis literal, [11](#)  
>>>, [11](#)  
\_\_future\_\_, [17](#)  
\_\_slots\_\_, [25](#)

## A

ad alanı, [22](#)  
ad alanı paketi, [22](#)  
adlandırılmış demet, [22](#)  
anahtar işlev, [20](#)  
anahtar kelime argümanı, [20](#)  
anlamak, [25](#)  
annotate function, [11](#)  
argüman, [11](#)  
asenkron bağlam yöneticisi, [12](#)  
asenkron jeneratör, [12](#)  
asenkron jeneratör yineleyici, [12](#)  
asenkron yineleyici, [12](#)  
attached thread state, [12](#)

## B

bağlam değişkeni, [14](#)  
bağlam yöneticisi, [14](#)  
bayt benzeri nesne, [13](#)  
bayt kodu, [13](#)  
BDFL, [13](#)  
beklenebilir, [13](#)  
belge dizisi, [16](#)  
bitişik, [14](#)  
BOŞTA, [19](#)  
bulucu, [17](#)

## C

C-contiguous, [14](#)  
closure variable, [14](#)  
context, [14](#)  
context management protocol, [14](#)  
CPython, [15](#)  
current context, [15](#)  
cyclic isolate, [15](#)

## Ç

çağırılabilir, [13](#)  
çöp toplama, [17](#)

## D

değişken açıklama, [28](#)  
değişmez, [19](#)  
değiştirilebilir, [22](#)  
dekoratör, [15](#)  
dipnot, [11](#)  
dizi, [25](#)  
dosya benzeri nesne, [16](#)  
dosya nesnesi, [16](#)  
dosya sistemi kodlaması ve hata  
    işleyicisi, [16](#)  
dunder, [16](#)

## E

EAFP, [16](#)  
eşyordam, [15](#)  
eşyordam işlevi, [15](#)  
eşzamansız yinelenebilir, [12](#)  
etkileşimli, [19](#)  
evaluate function, [16](#)  
evrensel yeni satırlar, [28](#)

## F

f-string, [16](#)  
f-strings, [16](#)  
fonksiyon, [17](#)  
fonksiyon açıklaması, [17](#)  
Fortran contiguous, [14](#)  
free threading, [17](#)  
free variable, [17](#)

## G

geçici API, [24](#)  
geçici paket, [24](#)  
genel işlev, [18](#)  
genel tercüman kilidi, [18](#)  
genel tip, [18](#)  
geri çağırma, [14](#)  
GIL, [18](#)



güçlü referans, [26](#)

## H

haritalama, [21](#)

## I

iç içe kapsam, [22](#)

içe aktarıcı, [19](#)

içe aktarım yolu, [19](#)

içe aktarma, [19](#)

ifade (*değer döndürmez*), [26](#)

ifade (*değer döndürür*), [16](#)

ikili dosya, [13](#)

immortal, [19](#)

## J

jeneratör, [18](#)

jeneratör ifadesi, [18](#)

jeneratör yineleyici, [18](#)

## K

karma tabanlı pyc, [18](#)

karmaşık sayı, [14](#)

kat bölümü, [17](#)

kısım, [24](#)

konumsal argüman, [24](#)

## L

lambda, [20](#)

LBYL, [20](#)

lexical analyzer, [20](#)

liste, [20](#)

liste anlama, [21](#)

## M

magic

metot, [21](#)

meta yol bulucu, [21](#)

metasınıf, [21](#)

metot, [21](#)

magic, [21](#)

special, [26](#)

metot kalite sıralaması, [21](#)

modül, [21](#)

modül özelliği, [22](#)

MRO, [22](#)

## N

nitelik, [13](#)

nitelikli isim, [25](#)

## O

obje, [23](#)

optimized scope, [23](#)

ortam değişkeni

PYTHON\_GIL, [18](#)

## Ö

ödünç alınan referans, [13](#)

ördek yazma, [16](#)

özel metod, [26](#)

## P

paket, [23](#)

parametre, [23](#)

parçalamak, [26](#)

PEP, [24](#)

Python 3000, [24](#)

Python Geliştirme Önerileri

PEP 1, [24](#)

PEP 238, [17](#)

PEP 278, [28](#)

PEP 302, [21](#)

PEP 343, [14](#)

PEP 362, [12](#), [23](#)

PEP 411, [24](#)

PEP 420, [22](#), [24](#)

PEP 443, [18](#)

PEP 483, [18](#)

PEP 484, [11](#), [17](#), [18](#), [27](#), [28](#)

PEP 492, [12](#), [13](#), [15](#)

PEP 498, [16](#)

PEP 519, [24](#)

PEP 525, [12](#)

PEP 526, [11](#), [28](#)

PEP 585, [18](#)

PEP 649, [11](#)

PEP 683, [19](#)

PEP 703, [17](#), [18](#)

PEP 3116, [28](#)

PEP 3155, [25](#)

PYTHON\_GIL, [18](#)

Pythonic, [24](#)

Python'un Zen'i, [28](#)

## R

referans sayısı, [25](#)

REPL, [25](#)

## S

sanal makine, [28](#)

sanal ortam, [28](#)

sınıf, [14](#)

sınıf değişkeni, [14](#)

sihirli yöntem, [21](#)

soft deprecated, [26](#)

soyut temel sınıf, [11](#)

sözlük, [15](#)

sözlük anlama, [15](#)

sözlük görünümü, [16](#)

special

metot, [26](#)

standard library, [26](#)

static type checker, [26](#)



stdlib, [26](#)  
sürekli paketleme, [25](#)

## T

t-string, [26](#)  
t-strings, [26](#)  
tanımlayıcı, [15](#)  
tek sevk, [26](#)  
tercüman kapatma, [19](#)  
thread state, [27](#)  
tip, [27](#)  
tip takma adı, [27](#)  
token, [27](#)  
tür ipucu, [27](#)

## U

uzatma modülü, [16](#)

## Ü

üç tırnaklı dize, [27](#)

## W

walrus operator, [28](#)

## Y

yazı çözümleme, [26](#)  
yazı dosyası, [26](#)  
yeni stil sınıf, [22](#)  
yerel kodlama, [21](#)  
yıkabilir, [18](#)  
yinelenebilir, [19](#)  
yineleyici, [20](#)  
yol benzeri nesne, [24](#)  
yol giriş kancası, [24](#)  
yol girişi, [23](#)  
yol girişi bulucu, [23](#)  
yol tabanlı bulucu, [24](#)  
yorumlanmış, [19](#)  
yükleyici, [21](#)