Installing Python Modules

Release 3.13.2

Guido van Rossum and the Python development team

março 25, 2025

Python Software Foundation Email: docs@python.org

Sumário

1	Termos chave				
2	Uso l	básico	5		
3	Come 3.1 3.2 3.3 3.4	o eu? instalo pip em versões do Python anteriores ao Python 3.4? instalo pacotes apenas para o usuário atual? instalo pacotes científicos do Python? trabalho com várias versões do Python instaladas em paralelo?	7 7 7 7		
4	4.1 4.2 4.3	Instalando no sistema Python no Linux	9 9 9		
A	Gloss	sário	11		
В	Sobre B.1	3	29 29		
C	C.1 C.2 C.3	História do software Termos e condições para acessar ou usar Python C.2.1 PYTHON SOFTWARE FOUNDATION LICENSE VERSION 2 C.2.2 ACORDO DE LICENCIAMENTO DA BEOPEN.COM PARA PYTHON 2.0 C.2.3 CONTRATO DE LICENÇA DA CNRI PARA O PYTHON 1.6.1 C.2.4 ACORDO DE LICENÇA DA CWI PARA PYTHON 0.9.0 A 1.2 C.2.5 ZERO-CLAUSE BSD LICENSE FOR CODE IN THE PYTHON DOCUMENTATION Licenças e Reconhecimentos para Software Incorporado C.3.1 Mersenne Twister C.3.2 Soquetes C.3.3 Serviços de soquete assíncrono C.3.4 Gerenciamento de cookies C.3.5 Rastreamento de execução C.3.6 Funções UUencode e UUdecode C.3.7 Chamadas de procedimento remoto XML C.3.8 test_epoll C.3.9 kqueue de seleção C.3.10 SipHash24 C.3.11 strtod e dtoa	31 31 32 33 34 35 35 35 37 37 38 39 40 40 41 41		

Íne	dice		53
D	Direitos autor	rais	51
	C.3.21	Global Unbounded Sequences (GUS)	49
	C.3.20	asyncio	48
	C.3.19	mimalloc	48
	C.3.18	Conjunto de testes C14N do W3C	47
	C.3.17	libmpdec	47
	C.3.16	cfuhash	46
		zlib	
	C.3.14	libffi	45
	C.3.13	expat	45

E-mail

distutils-sig@python.org

Como um projeto popular de desenvolvimento de código aberto, Python tem uma comunidade de apoio ativa de colaboradores e usuários, que também fazem o seu software disponível para outros desenvolvedores de Python para usar sob os termos da licença de código aberto.

Isso permite aos usuários Python compartilhar e colaborar efetivamente, se beneficiando das soluções que outros já tenham criado para os problemas mais comuns (em alguns casos até mesmo os raros), bem como potencialmente contribuindo com suas próprias soluções para o conjunto de soluções comuns.

Este guia cobre a parte do processo de instalação. Para um guia sobre criar e compartilhar seus próprios projetos Python, confira o Guia de Usuário para Empacotamento de Python.



1 Nota

Para corporações e outros usuários institucionais, esteja ciente que muitas organizações têm suas próprias políticas em relação ao uso e contribuição para o software de código aberto. Por favor, leve em conta essas políticas ao usar as ferramentas de distribuição e instalação fornecidas com o Python.

1 Sumário

2 Sumário

Termos chave

- pip é o programa de instalação preferido. A partir do Python 3.4, ele é incluído por padrão com os instaladores binários do Python.
- Um *ambiente virtual* é um ambiente Python semi-isolado que permite que pacotes sejam instalados para uso por uma aplicação específica, em vez de serem instaladas em todo o sistema.
- venv é a ferramenta padrão para criar ambientes virtuais e faz parte do Python desde o Python 3.3. A partir do Python 3.4, o padrão é instalar pip em todos os ambientes virtuais criados.
- virtualenv é uma alternativa de terceiros (e predecessora) ao venv. Ele permite que ambientes virtuais sejam usados em versões do Python anteriores a 3.4, que não fornecem venv de forma alguma, ou não são capazes de instalar automaticamente o pip nos ambientes criados.
- O Python Package Index é um repositório público de pacotes licenciados como código aberto e disponíveis para uso de outros usuários Python
- o Python Packaging Authority é o grupo de desenvolvedores e autores de documentação responsáveis pela manutenção e evolução das ferramentas de empacotamento padrão e os metadados associados e padrões de formato de arquivo. Eles mantêm uma variedade de ferramentas, documentação e rastreadores de problemas no GitHub.
- distutils é o sistema original de construção e distribuição adicionado pela primeira vez à biblioteca padrão Python em 1998. Embora o uso direto de distutils esteja sendo eliminado, ele ainda estabeleceu a base para a infraestrutura de distribuição e empacotamento atual, e não apenas permanece da biblioteca padrão, mas seu nome sobrevive de outras maneiras (como o nome da lista de e-mails usada para coordenar o desenvolvimento de padrões de empacotamento Python).

Alterado na versão 3.5: O uso de venv agora é recomendado para a criação de ambientes virtuais.

→ Ver também

Guia de Usuário para Empacotamento de Python: Criando e usando ambientes virtuais

Uso básico

As ferramentas de empacotamento padrão são todas projetadas para serem usadas na linha de comando.

O comando a seguir instalará a versão mais recente de um módulo e suas dependências do Python Package Index:

```
python -m pip install AlgumPacote
```

1 Nota

Para usuários POSIX (incluindo usuários macOS e Linux), os exemplos neste guia presumem o uso de um *ambiente virtual*.

Para usuários do Windows, os exemplos neste guia presumem que a opção de ajustar a variável de ambiente PATH do sistema foi selecionada durante a instalação do Python.

Também é possível especificar uma versão exata ou mínima diretamente na linha de comando. Ao usar operadores de comparação como >, < ou algum outro caractere especial que é interpretado pelo shell, o nome do pacote e a versão devem ser colocados entre aspas duplas:

```
python -m pip install AlgumPacote==1.0.4  # versão específica
python -m pip install "AlgumPacote>=1.0.4"  # versão mínima
```

Normalmente, se um módulo adequado já estiver instalado, tentar instalá-lo novamente não terá efeito. A atualização de módulos existentes deve ser solicitada explicitamente:

```
python -m pip install --upgrade AlgumPacote
```

Mais informações e recursos sobre o pip e seus recursos podem ser encontrados no Python Packaging User Guide.

A criação de ambientes virtuais é feita através do módulo venv. A instalação de pacotes em um ambiente virtual ativo usa os comandos mostrados acima.

→ Ver também

Guia de Usuário para Empacotamento de Python: Instalando pacotes Python de distribuição

Como eu ...?

Estas são respostas rápidas ou links para algumas tarefas comuns.

3.1 ... instalo pip em versões do Python anteriores ao Python 3.4?

Python apenas começou a empacotar pip com Python 3.4. Para versões anteriores, o pip precisa ser "inicializado" conforme descrito no Python Packaging User Guide.

→ Ver também

Python Packaging User Guide: Requisitos para instalar pacotes

3.2 ... instalo pacotes apenas para o usuário atual?

Passar a opção — user para python — m pip install irá instalar um pacote apenas para o usuário atual, ao invés de para todos os usuários do sistema.

3.3 ... instalo pacotes científicos do Python?

Vários pacotes científicos do Python têm dependências binárias complexas e atualmente não são fáceis de instalar usando o pip diretamente. Neste ponto, frequentemente será mais fácil para os usuários instalarem esses pacotes por outros meios ao invés de tentar instalá-los com pip.

Ver também

Guia de Usuário para Empacotamento de Python: Instalando pacotes científicos

3.4 ... trabalho com várias versões do Python instaladas em paralelo?

No Linux, macOS e outros sistemas POSIX, use os comandos Python com versão em combinação com a opção -m para executar a cópia apropriada de pip

```
python2 -m pip install AlgumPacote # Python 2 padrão
python2.7 -m pip install AlgumPacote # Python 2.7 especificamente
python3 -m pip install AlgumPacote # Python 3 padrão
python3.4 -m pip install AlgumPacote # Python 3.4 especificamente
```

Comandos pip com versão apropriada também podem estar disponíveis.

No Windows, use o iniciador Python py em combinação com a opção -m:

```
py -2 -m pip install AlgumPacote # Python 2 padrão
py -2.7 -m pip install AlgumPacote # Python 2.7 especificamente
py -3 -m pip install AlgumPacote # Python 3 padrão
py -3.4 -m pip install AlgumPacote # Python 3.4 especificamente
```

Problemas comuns de instalação

4.1 Instalando no sistema Python no Linux

Em sistemas Linux, uma instalação Python normalmente será incluída como parte da distribuição. A instalação nesta instalação Python requer acesso root ao sistema, e pode interferir na operação do gerenciador de pacotes do sistema e outros componentes do sistema se um componente for atualizado inesperadamente usando pip.

Em tais sistemas, geralmente é melhor usar um ambiente virtual ou uma instalação por usuário ao instalar pacotes com pip.

4.2 Pip não instalado

É possível que o pip não seja instalado por padrão. Uma solução potencial é:

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

Existem também recursos adicionais para instalar pip.

4.3 Instalando extensões binárias

O Python normalmente depende fortemente da distribuição baseada na fonte, com os usuários finais sendo esperados para compilar os módulos de extensão da fonte como parte do processo de instalação.

Com a introdução do suporte para o formato binário wheel e a capacidade de publicar wheels para pelo menos Windows e macOS através do Python Package Index, espera-se que este problema diminua com o tempo, à medida que os usuários são mais capazes para instalar extensões pré-construídas em vez de precisar construí-las eles próprios.

Algumas das soluções para instalar softwares científicos que ainda não estão disponíveis como arquivos wheel préconstruídos também podem ajudar a obter outras extensões binárias sem a necessidade para construí-los localmente.

Ver também

Guia de Usuário para Empacotamento de Python: Extensões binárias

APÊNDICE A

Glossário

>>>

O prompt padrão do console *interativo* do Python. Normalmente visto em exemplos de código que podem ser executados interativamente no interpretador.

. . .

Pode se referir a:

- O prompt padrão do console *interativo* do Python ao inserir o código para um bloco de código recuado, quando dentro de um par de delimitadores correspondentes esquerdo e direito (parênteses, colchetes, chaves ou aspas triplas) ou após especificar um decorador.
- A constante embutida Ellipsis.

classe base abstrata

Classes bases abstratas complementam *tipagem pato*, fornecendo uma maneira de definir interfaces quando outras técnicas, como hasattr(), seriam desajeitadas ou sutilmente erradas (por exemplo, com métodos mágicos). ABCs introduzem subclasses virtuais, classes que não herdam de uma classe mas ainda são reconhecidas por isinstance() e issubclass(); veja a documentação do módulo abc. Python vem com muitas ABCs embutidas para estruturas de dados (no módulo collections.abc), números (no módulo numbers), fluxos (no módulo io), localizadores e carregadores de importação (no módulo importlib.abc). Você pode criar suas próprias ABCs com o módulo abc.

anotação

Um rótulo associado a uma variável, um atributo de classe ou um parâmetro de função ou valor de retorno, usado por convenção como *dica de tipo*.

Anotações de variáveis locais não podem ser acessadas em tempo de execução, mas anotações de variáveis globais, atributos de classe e funções são armazenadas no atributo especial __annotations__ de módulos, classes e funções, respectivamente.

Veja *anotação de variável*, *anotação de função*, **PEP 484** e **PEP 526**, que descrevem esta funcionalidade. Veja também annotations-howto para as melhores práticas sobre como trabalhar com anotações.

argumento

Um valor passado para uma função (ou método) ao chamar a função. Existem dois tipos de argumento:

• *argumento nomeado*: um argumento precedido por um identificador (por exemplo, name=) na chamada de uma função ou passada como um valor em um dicionário precedido por **. Por exemplo, 3 e 5 são ambos argumentos nomeados na chamada da função complex() a seguir:

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

• *argumento posicional*: um argumento que não é um argumento nomeado. Argumentos posicionais podem aparecer no início da lista de argumentos e/ou podem ser passados com elementos de um *iterável* precedido por *. Por exemplo, 3 e 5 são ambos argumentos posicionais nas chamadas a seguir:

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

Argumentos são atribuídos às variáveis locais nomeadas no corpo da função. Veja a seção calls para as regras de atribuição. Sintaticamente, qualquer expressão pode ser usada para representar um argumento; avaliada a expressão, o valor é atribuído à variável local.

Veja também o termo *parâmetro* no glossário, a pergunta no FAQ sobre a diferença entre argumentos e parâmetros e **PEP 362**.

gerenciador de contexto assíncrono

Um objeto que controla o ambiente visto numa instrução async with por meio da definição dos métodos __aenter__() e __aexit__(). Introduzido pela PEP 492.

gerador assíncrono

Uma função que retorna um *iterador gerador assíncrono*. É parecida com uma função de corrotina definida com asynd def exceto pelo fato de conter instruções yield para produzir uma série de valores que podem ser usados em um laço asynd for.

Normalmente se refere a uma função geradora assíncrona, mas pode se referir a um *iterador gerador assíncrono* em alguns contextos. Em casos em que o significado não esteja claro, usar o termo completo evita a ambiguidade.

Uma função geradora assíncrona pode conter expressões await e também as instruções async for e async with.

iterador gerador assíncrono

Um objeto criado por uma função geradora assíncrona.

Este é um *iterador assíncrono* que, quando chamado usando o método __anext__(), retorna um objeto aguardável que executará o corpo da função geradora assíncrona até a próxima expressão yield.

Cada yield suspende temporariamente o processamento, lembrando o estado de execução (incluindo variáveis locais e instruções try pendentes). Quando o *iterador gerador assíncrono* é efetivamente retomado com outro aguardável retornado por __anext__ (), ele inicia de onde parou. Veja PEP 492 e PEP 525.

iterável assíncrono

Um objeto que pode ser usado em uma instrução async for. Deve retornar um *iterador assíncrono* do seu método __aiter__(). Introduzido por PEP 492.

iterador assíncrono

Um objeto que implementa os métodos __aiter__() e __anext__(). __anext__() deve retornar um objeto aguardável. async for resolve os aguardáveis retornados por um método __anext__() do iterador assíncrono até que ele levante uma exceção <code>StopAsyncIteration</code>. Introduzido pela PEP 492.

atributo

Um valor associado a um objeto que é geralmente referenciado pelo nome separado por um ponto. Por exemplo, se um objeto o tem um atributo a esse seria referenciado como o.a.

É possível dar a um objeto um atributo cujo nome não seja um identificador conforme definido por identifiers, por exemplo usando setattr(), se o objeto permitir. Tal atributo não será acessível usando uma expressão pontilhada e, em vez disso, precisaria ser recuperado com getattr().

aguardável

Um objeto que pode ser usado em uma expressão await. Pode ser uma *corrotina* ou um objeto com um método __await__(). Veja também a PEP 492.

BDFL

Abreviação da expressão da língua inglesa "Benevolent Dictator for Life" (em português, "Ditador Benevolente Vitalício"), referindo-se a Guido van Rossum, criador do Python.

arquivo binário

Um *objeto arquivo* capaz de ler e gravar em *objetos bytes ou similar*. Exemplos de arquivos binários são arquivos abertos no modo binário ('rb', 'wb' ou 'rb+'), sys.stdin.buffer, sys.stdout.buffer, e instâncias de io.BytesIO e gzip.GzipFile.

Veja também arquivo texto para um objeto arquivo capaz de ler e gravar em objetos str.

referência emprestada

Na API C do Python, uma referência emprestada é uma referência a um objeto que não é dona da referência. Ela se torna um ponteiro solto se o objeto for destruído. Por exemplo, uma coleta de lixo pode remover a última *referência forte* para o objeto e assim destruí-lo.

Chamar Py_INCREF () na *referência emprestada* é recomendado para convertê-lo, internamente, em uma *referência forte*, exceto quando o objeto não pode ser destruído antes do último uso da referência emprestada. A função Py_NewRef () pode ser usada para criar uma nova *referência forte*.

objeto byte ou similar

Um objeto com suporte ao o bufferobjects e que pode exportar um buffer C *contíguo*. Isso inclui todos os objetos bytes, bytearray e array, além de muitos objetos memoryview comuns. Objetos byte ou similar podem ser usados para várias operações que funcionam com dados binários; isso inclui compactação, salvamento em um arquivo binário e envio por um soquete.

Algumas operações precisam que os dados binários sejam mutáveis. A documentação geralmente se refere a eles como "objetos byte ou similar para leitura e escrita". Exemplos de objetos de buffer mutável incluem bytearray e um memoryview de um bytearray. Outras operações exigem que os dados binários sejam armazenados em objetos imutáveis ("objetos byte ou similar para somente leitura"); exemplos disso incluem bytes e a memoryview de um objeto bytes.

bytecode

O código-fonte Python é compilado para bytecode, a representação interna de um programa em Python no interpretador CPython. O bytecode também é mantido em cache em arquivos .pyc e .pyo, de forma que executar um mesmo arquivo é mais rápido na segunda vez (a recompilação dos fontes para bytecode não é necessária). Esta "linguagem intermediária" é adequada para execução em uma *máquina virtual*, que executa o código de máquina correspondente para cada bytecode. Tenha em mente que não se espera que bytecodes sejam executados entre máquinas virtuais Python diferentes, nem que se mantenham estáveis entre versões de Python.

Uma lista de instruções bytecode pode ser encontrada na documentação para o módulo dis.

chamável

Um chamável é um objeto que pode ser chamado, possivelmente com um conjunto de argumentos (veja *argumento*), com a seguinte sintaxe:

```
chamavel(argumento1, argumento2, argumentoN)
```

Uma *função*, e por extensão um *método*, é um chamável. Uma instância de uma classe que implementa o método __call__() também é um chamável.

função de retorno

Também conhecida como callback, é uma função sub-rotina que é passada como um argumento a ser executado em algum ponto no futuro.

classe

Um modelo para criação de objetos definidos pelo usuário. Definições de classe normalmente contém definições de métodos que operam sobre instâncias da classe.

variável de classe

Uma variável definida em uma classe e destinada a ser modificada apenas no nível da classe (ou seja, não em uma instância da classe).

variável de clausura

Uma *variável livre* referenciada de um *escopo aninhado* que é definida em um escopo externo em vez de ser resolvida em tempo de execução a partir dos espaços de nomes embutido ou globais. Pode ser explicitamente definida com a palavra reservada nonlocal para permitir acesso de gravação, ou implicitamente definida se a variável estiver sendo somente lida.

Por exemplo, na função interna no código a seguir, tanto x quanto print são *variáveis livres*, mas somente x é uma *variável de clausura*:

```
def externa():
    x = 0
    def interna():
        nonlocal x
        x += 1
        print(x)
    return interna
```

Devido ao atributo codeobject.co_freevars (que, apesar do nome, inclui apenas os nomes das variáveis de clausura em vez de listar todas as variáveis livres referenciadas), o termo mais geral *variável livre* às vezes é usado mesmo quando o significado pretendido é se referir especificamente às variáveis de clausura.

número complexo

Uma extensão ao familiar sistema de números reais em que todos os números são expressos como uma soma de uma parte real e uma parte imaginária. Números imaginários são múltiplos reais da unidade imaginária (a raiz quadrada de -1), normalmente escrita como i em matemática ou j em engenharia. O Python tem suporte nativo para números complexos, que são escritos com esta última notação; a parte imaginária escrita com um sufixo j, p.ex., 3+1j. Para ter acesso aos equivalentes para números complexos do módulo math, utilize cmath. O uso de números complexos é uma funcionalidade matemática bastante avançada. Se você não sabe se irá precisar deles, é quase certo que você pode ignorá-los sem problemas.

contexto

Este termo tem diferentes significados dependendo de onde e como ele é usado. Alguns significados comuns:

- O estado ou ambiente temporário estabelecido por um *gerenciador de contexto* por meio de uma instrução with.
- A coleção de ligações de chave-valor associadas a um objeto contextvars.Context específico e acessadas por meio de objetos ContextVar. Veja também *variável de contexto*.
- Um objeto contextvars. Context. Veja também contexto atual.

protocolo de gerenciamento de contexto

```
Os métodos __enter__() e __exit__() chamados pela instrução with. Veja PEP 343.
```

gerenciador de contexto

Um objeto que implementa o *protocolo de gerenciamento de contexto* e controla o ambiente visto em uma instrução with. Veja PEP 343.

variável de contexto

Uma variável cujo valor depende de qual contexto é o *contexto atual*. Os valores são acessados por meio de objetos contextvars. Contextvar. Variáveis de contexto são usadas principalmente para isolar o estado entre tarefas assíncronas simultâneas.

contíguo

Um buffer é considerado contíguo exatamente se for *contíguo C* ou *contíguo Fortran*. Os buffers de dimensão zero são contíguos C e Fortran. Em vetores unidimensionais, os itens devem ser dispostos na memória próximos um do outro, em ordem crescente de índices, começando do zero. Em vetores multidimensionais contíguos C, o último índice varia mais rapidamente ao visitar itens em ordem de endereço de memória. No entanto, nos vetores contíguos do Fortran, o primeiro índice varia mais rapidamente.

corrotina

Corrotinas são uma forma mais generalizada de sub-rotinas. Sub-rotinas tem a entrada iniciada em um ponto, e a saída em outro ponto. Corrotinas podem entrar, sair, e continuar em muitos pontos diferentes. Elas podem ser implementadas com a instrução async def. Veja também **PEP 492**.

função de corrotina

Uma função que retorna um objeto do tipo *corrotina*. Uma função de corrotina pode ser definida com a instrução async def, e pode conter as palavras chaves await, async for, e async with. Isso foi introduzido pela PEP 492.

CPython

A implementação canônica da linguagem de programação Python, como disponibilizada pelo python.org. O termo "CPython" é usado quando necessário distinguir esta implementação de outras como Jython ou IronPython.

contexto atual

O contexto (objeto contextvars.Context) que é usado atualmente pelos objetos ContextVar para acessar (obter ou definir) os valores de variáveis de contexto. Cada thread tem seu próprio contexto atual. Frameworks para executar tarefas assíncronas (veja asyncio) associam cada tarefa a um contexto que se torna o contexto atual sempre que a tarefa inicia ou retoma a execução.

decorador

Uma função que retorna outra função, geralmente aplicada como uma transformação de função usando a sintaxe @wrapper. Exemplos comuns para decoradores são classmethod() e staticmethod().

A sintaxe do decorador é meramente um açúcar sintático, as duas definições de funções a seguir são semanticamente equivalentes:

O mesmo conceito existe para as classes, mas não é comumente utilizado. Veja a documentação de definições de função e definições de classe para obter mais informações sobre decoradores.

descritor

Qualquer objeto que define os métodos __get__(), __set__() ou __delete__(). Quando um atributo de classe é um descritor, seu comportamento de associação especial é acionado no acesso a um atributo. Normalmente, ao se utilizar a.b para se obter, definir ou excluir, um atributo dispara uma busca no objeto chamado b no dicionário de classe de a, mas se b for um descritor, o respectivo método descritor é chamado. Compreender descritores é a chave para um profundo entendimento de Python pois eles são a base de muitas funcionalidades incluindo funções, métodos, propriedades, métodos de classe, métodos estáticos e referências para superclasses.

Para obter mais informações sobre os métodos dos descritores, veja: descriptors ou o Guia de Descritores.

dicionário

Um vetor associativo em que chaves arbitrárias são mapeadas para valores. As chaves podem ser quaisquer objetos que possuam os métodos __hash__() e __eq__(). Isso é chamado de hash em Perl.

compreensão de dicionário

Uma maneira compacta de processar todos ou parte dos elementos de um iterável e retornar um dicionário com os resultados. results = {n: n ** 2 for n in range(10)} gera um dicionário contendo a chave n mapeada para o valor n ** 2. Veja comprehensions.

visão de dicionário

Os objetos retornados por dict.keys(), dict.values() e dict.items() são chamados de visões de dicionário. Eles fornecem uma visão dinâmica das entradas do dicionário, o que significa que quando o dicionário é alterado, a visão reflete essas alterações. Para forçar a visão de dicionário a se tornar uma lista completa use list(dictview). Veja dict-views.

docstring

Abreviatura de "documentation string" (string de documentação). Uma string literal que aparece como primeira expressão numa classe, função ou módulo. Ainda que sejam ignoradas quando a suíte é executada, é

reconhecida pelo compilador que a coloca no atributo __doc__ da classe, função ou módulo que a encapsula. Como ficam disponíveis por meio de introspecção, docstrings são o lugar canônico para documentação do objeto.

tipagem pato

Também conhecida como *duck-typing*, é um estilo de programação que não verifica o tipo do objeto para determinar se ele possui a interface correta; em vez disso, o método ou atributo é simplesmente chamado ou utilizado ("Se se parece com um pato e grasna como um pato, então deve ser um pato.") Enfatizando interfaces ao invés de tipos específicos, o código bem desenvolvido aprimora sua flexibilidade por permitir substituição polimórfica. Tipagem pato evita necessidade de testes que usem type() ou isinstance(). (Note, porém, que a tipagem pato pode ser complementada com o uso de *classes base abstratas*.) Ao invés disso, são normalmente empregados testes hasattr() ou programação *EAFP*.

EAFP

Iniciais da expressão em inglês "easier to ask for forgiveness than permission" que significa "é mais fácil pedir perdão que permissão". Este estilo de codificação comum no Python presume a existência de chaves ou atributos válidos e captura exceções caso essa premissa se prove falsa. Este estilo limpo e rápido se caracteriza pela presença de várias instruções try e except. A técnica diverge do estilo *LBYL*, comum em outras linguagens como C, por exemplo.

expressão

Uma parte da sintaxe que pode ser avaliada para algum valor. Em outras palavras, uma expressão é a acumulação de elementos de expressão como literais, nomes, atributos de acesso, operadores ou chamadas de funções, todos os quais retornam um valor. Em contraste com muitas outras linguagens, nem todas as construções de linguagem são expressões. Também existem *instruções*, as quais não podem ser usadas como expressões, como, por exemplo, while. Atribuições também são instruções, não expressões.

módulo de extensão

Um módulo escrito em C ou C++, usando a API C do Python para interagir tanto com código de usuário quanto do núcleo.

f-string

Literais string prefixadas com 'f' ou 'F' são conhecidas como "f-strings" que é uma abreviação de formatted string literals. Veja também **PEP 498**.

objeto arquivo

Um objeto que expõe uma API orientada a arquivos (com métodos tais como read() ou write()) para um recurso subjacente. Dependendo da maneira como foi criado, um objeto arquivo pode mediar o acesso a um arquivo real no disco ou outro tipo de dispositivo de armazenamento ou de comunicação (por exemplo a entrada/saída padrão, buffers em memória, soquetes, pipes, etc.). Objetos arquivo também são chamados de *objetos arquivo ou similares* ou *fluxos*.

Atualmente há três categorias de objetos arquivo: *arquivos binários* brutos, *arquivos binários* em buffer e *arquivos textos*. Suas interfaces estão definidas no módulo io. A forma canônica para criar um objeto arquivo é usando a função open().

objeto arquivo ou similar

Um sinônimo do termo objeto arquivo.

tratador de erros e codificação do sistema de arquivos

Tratador de erros e codificação usado pelo Python para decodificar bytes do sistema operacional e codificar Unicode para o sistema operacional.

A codificação do sistema de arquivos deve garantir a decodificação bem-sucedida de todos os bytes abaixo de 128. Se a codificação do sistema de arquivos falhar em fornecer essa garantia, as funções da API podem levantar UnicodeError.

As funções sys.getfilesystemencoding() e sys.getfilesystemencodeerrors() podem ser usadas para obter o tratador de erros e codificação do sistema de arquivos.

O *tratador de erros e codificação do sistema de arquivos* são configurados na inicialização do Python pela função PyConfig_Read(): veja os membros filesystem_encoding e filesystem_errors do PyConfig.

Veja também codificação da localidade.

localizador

Um objeto que tenta encontrar o carregador para um módulo que está sendo importado.

Existem dois tipos de localizador: *localizadores de metacaminho* para uso com sys.meta_path, e *localizadores de entrada de caminho* para uso com sys.path_hooks.

Veja finders-and-loaders e importlib para muito mais detalhes.

divisão pelo piso

Divisão matemática que arredonda para baixo para o inteiro mais próximo. O operador de divisão pelo piso é //. Por exemplo, a expressão 11 // 4 retorna o valor 2 ao invés de 2.75, que seria retornado pela divisão de ponto flutuante. Note que (-11) // 4 é -3 porque é -2.75 arredondado *para baixo*. Consulte a **PEP 238**.

threads livres

Um modelo de threads onde múltiplas threads podem simultaneamente executar bytecode Python no mesmo interpretador. Isso está em contraste com a *trava global do interpretador* que permite apenas uma thread por vez executar bytecode Python. Veja **PEP 703**.

variável livre

Formalmente, conforme definido no modelo de execução de linguagem, uma variável livre é qualquer variável usada em um espaço de nomes que não seja uma variável local naquele espaço de nomes. Veja *variável de clausura* para um exemplo. Pragmaticamente, devido ao nome do atributo codeobject.co_freevars, o termo também é usado algumas vezes como sinônimo de *variável de clausura*.

função

Uma série de instruções que retorna algum valor para um chamador. Também pode ser passado zero ou mais *argumentos* que podem ser usados na execução do corpo. Veja também *parâmetro*, *método* e a seção function.

anotação de função

Uma anotação de um parâmetro de função ou valor de retorno.

Anotações de função são comumente usados por *dicas de tipo*: por exemplo, essa função espera receber dois argumentos int e também é esperado que devolva um valor int:

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

A sintaxe de anotação de função é explicada na seção function.

Veja *anotação de variável* e **PEP 484**, que descrevem esta funcionalidade. Veja também annotations-howto para as melhores práticas sobre como trabalhar com anotações.

future

A instrução future, from __future__ import <feature>, direciona o compilador a compilar o módulo atual usando sintaxe ou semântica que será padrão em uma versão futura de Python. O módulo __future__ documenta os possíveis valores de *feature*. Importando esse módulo e avaliando suas variáveis, você pode ver quando um novo recurso foi inicialmente adicionado à linguagem e quando será (ou se já é) o padrão:

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

coleta de lixo

Também conhecido como *garbage collection*, é o processo de liberar a memória quando ela não é mais utilizada. Python executa a liberação da memória através da contagem de referências e um coletor de lixo cíclico que é capaz de detectar e interromper referências cíclicas. O coletor de lixo pode ser controlado usando o módulo qc.

gerador

Uma função que retorna um *iterador gerador*. É parecida com uma função normal, exceto pelo fato de conter expressões yield para produzir uma série de valores que podem ser usados em um laço "for" ou que podem ser obtidos um de cada vez com a função next ().

Normalmente refere-se a uma função geradora, mas pode referir-se a um *iterador gerador* em alguns contextos. Em alguns casos onde o significado desejado não está claro, usar o termo completo evita ambiguidade.

iterador gerador

Um objeto criado por uma função geradora.

Cada yield suspende temporariamente o processamento, memorizando o estado da execução (incluindo variáveis locais e instruções try pendentes). Quando o *iterador gerador* retorna, ele se recupera do último ponto onde estava (em contrapartida as funções que iniciam uma nova execução a cada vez que são invocadas).

expressão geradora

Uma *expressão* que retorna um *iterador*. Parece uma expressão normal, seguido de uma cláusula for definindo uma variável de laço, um intervalo, e uma cláusula if opcional. A expressão combinada gera valores para uma função encapsuladora:

```
>>> sum(i*i for i in range(10))  # soma dos quadrados 0, 1, 4, ... 81
285
```

função genérica

Uma função composta por várias funções implementando a mesma operação para diferentes tipos. Qual implementação deverá ser usada durante a execução é determinada pelo algoritmo de despacho.

Veja também a entrada *despacho único* no glossário, o decorador functools.singledispatch(), e a PEP 443.

tipo genérico

Um *tipo* que pode ser parametrizado; tipicamente uma classe contêiner tal como list ou dict. Usado para *dicas de tipo* e *anotações*.

Para mais detalhes, veja tipo apelido genérico, PEP 483, PEP 484, PEP 585, e o módulo typing.

GIL

Veja trava global do interpretador.

trava global do interpretador

O mecanismo utilizado pelo interpretador *CPython* para garantir que apenas uma thread execute o *bytecode* Python por vez. Isto simplifica a implementação do CPython ao fazer com que o modelo de objetos (incluindo tipos embutidos críticos como o dict) ganhem segurança implícita contra acesso concorrente. Travar todo o interpretador facilita que o interpretador em si seja multitarefa, às custas de muito do paralelismo já provido por máquinas multiprocessador.

No entanto, alguns módulos de extensão, tanto da biblioteca padrão quanto de terceiros, são desenvolvidos de forma a liberar a GIL ao realizar tarefas computacionalmente muito intensas, como compactação ou cálculos de hash. Além disso, a GIL é sempre liberado nas operações de E/S.

A partir de Python 3.13, o GIL pode ser desabilitado usando a configuração de construção —disable—gil. Depois de construir Python com essa opção, o código deve ser executado com a opção —X gil=0 ou a variável de ambiente PYTHON_GIL=0 deve estar definida. Esse recurso provê um desempenho melhor para aplicações com múltiplas threads e torna mais fácil o uso eficiente de CPUs com múltiplos núcleos. Para mais detalhes, veja PEP 703.

pyc baseado em hash

Um arquivo de cache em bytecode que usa hash ao invés do tempo, no qual o arquivo de código-fonte foi modificado pela última vez, para determinar a sua validade. Veja pyc-invalidation.

hasheável

Um objeto é *hasheável* se tem um valor de hash que nunca muda durante seu ciclo de vida (precisa ter um método __hash__()) e pode ser comparado com outros objetos (precisa ter um método __eq__()). Objetos hasheáveis que são comparados como iguais devem ter o mesmo valor de hash.

A hasheabilidade faz com que um objeto possa ser usado como uma chave de dicionário e como um membro de conjunto, pois estas estruturas de dados utilizam os valores de hash internamente.

A maioria dos objetos embutidos imutáveis do Python são hasheáveis; containers mutáveis (tais como listas ou dicionários) não são; containers imutáveis (tais como tuplas e frozensets) são hasheáveis apenas se os seus

elementos são hasheáveis. Objetos que são instâncias de classes definidas pelo usuário são hasheáveis por padrão. Todos eles comparam de forma desigual (exceto entre si mesmos), e o seu valor hash é derivado a partir do seu id().

IDLE

Um ambiente de desenvolvimento e aprendizado integrado para Python. idle é um editor básico e um ambiente interpretador que vem junto com a distribuição padrão do Python.

imortal

Objetos imortais são um detalhe da implementação do CPython introduzida na PEP 683.

Se um objeto é imortal, sua *contagem de referências* nunca é modificada e, portanto, nunca é desalocado enquanto o interpretador está em execução. Por exemplo, True e None são imortais no CPython.

imutável

Um objeto que possui um valor fixo. Objetos imutáveis incluem números, strings e tuplas. Estes objetos não podem ser alterados. Um novo objeto deve ser criado se um valor diferente tiver de ser armazenado. Objetos imutáveis têm um papel importante em lugares onde um valor constante de hash seja necessário, como por exemplo uma chave em um dicionário.

caminho de importação

Uma lista de localizações (ou *entradas de caminho*) que são buscadas pelo *localizador baseado no caminho* por módulos para importar. Durante a importação, esta lista de localizações usualmente vem a partir de sys.path, mas para subpacotes ela também pode vir do atributo __path__ de pacotes-pai.

importação

O processo pelo qual o código Python em um módulo é disponibilizado para o código Python em outro módulo.

importador

Um objeto que localiza e carrega um módulo; Tanto um localizador e o objeto carregador.

interativo

Python tem um interpretador interativo, o que significa que você pode digitar instruções e expressões no prompt do interpretador, executá-los imediatamente e ver seus resultados. Apenas execute python sem argumentos (possivelmente selecionando-o a partir do menu de aplicações de seu sistema operacional). O interpretador interativo é uma maneira poderosa de testar novas ideias ou aprender mais sobre módulos e pacotes (lembre-se do comando help(x)). Para saber mais sobre modo interativo, veja tut-interac.

interpretado

Python é uma linguagem interpretada, em oposição àquelas que são compiladas, embora esta distinção possa ser nebulosa devido à presença do compilador de bytecode. Isto significa que os arquivos-fontes podem ser executados diretamente sem necessidade explícita de se criar um arquivo executável. Linguagens interpretadas normalmente têm um ciclo de desenvolvimento/depuração mais curto que as linguagens compiladas, apesar de seus programas geralmente serem executados mais lentamente. Veja também *interativo*.

desligamento do interpretador

Quando solicitado para desligar, o interpretador Python entra em uma fase especial, onde ele gradualmente libera todos os recursos alocados, tais como módulos e várias estruturas internas críticas. Ele também faz diversas chamadas para o *coletor de lixo*. Isto pode disparar a execução de código em destrutores definidos pelo usuário ou função de retorno de referência fraca. Código executado durante a fase de desligamento pode encontrar diversas exceções, pois os recursos que ele depende podem não funcionar mais (exemplos comuns são os módulos de bibliotecas, ou os mecanismos de avisos).

A principal razão para o interpretador desligar, é que o módulo __main__ ou o script sendo executado terminou sua execução.

iterável

Um objeto capaz de retornar seus membros um de cada vez. Exemplos de iteráveis incluem todos os tipos de sequência (tais como list, str e tuple) e alguns tipos de não-sequência, como o dict, *objetos arquivos*, além dos objetos de quaisquer classes que você definir com um método __iter__() ou __getitem__() que implementam a semântica de *sequência*.

Iteráveis podem ser usados em um laço for e em vários outros lugares em que uma sequência é necessária (zip(), map(), ...). Quando um objeto iterável é passado como argumento para a função embutida iter(), ela retorna um iterador para o objeto. Este iterador é adequado para se varrer todo o conjunto de valores. Ao

usar iteráveis, normalmente não é necessário chamar iter() ou lidar com os objetos iteradores em si. A instrução for faz isso automaticamente para você, criando uma variável temporária para armazenar o iterador durante a execução do laço. Veja também *iterador*, *sequência*, e *gerador*.

iterador

Um objeto que representa um fluxo de dados. Repetidas chamadas ao método __next__() de um iterador (ou passando o objeto para a função embutida next()) vão retornar itens sucessivos do fluxo. Quando não houver mais dados disponíveis uma exceção StopIteration será levantada. Neste ponto, o objeto iterador se esgotou e quaisquer chamadas subsequentes a seu método __next__() vão apenas levantar a exceção StopIteration novamente. Iteradores precisam ter um método __iter__() que retorne o objeto iterador em si, de forma que todo iterador também é iterável e pode ser usado na maioria dos lugares em que um iterável é requerido. Uma notável exceção é código que tenta realizar passagens em múltiplas iterações. Um objeto contêiner (como uma list) produz um novo iterador a cada vez que você passá-lo para a função iter() ou utilizá-lo em um laço for. Tentar isso com o mesmo iterador apenas iria retornar o mesmo objeto iterador esgotado já utilizado na iteração anterior, como se fosse um contêiner vazio.

Mais informações podem ser encontradas em typeiter.

O CPython não aplica consistentemente o requisito de que um iterador defina __iter__(). E também observe que o CPython com threads livres não garante a segurança do thread das operações do iterador.

função chave

Uma função chave ou função colação é um chamável que retorna um valor usado para ordenação ou classificação. Por exemplo, locale.strxfrm() é usada para produzir uma chave de ordenação que leva o locale em consideração para fins de ordenação.

Uma porção de ferramentas no Python aceitam funções chave para controlar como os elementos são ordenados ou agrupados. Algumas delas incluem min(), max(), sorted(), list.sort(), heapq.merge(), heapq.nsmallest(), heapq.nlargest() e itertools.groupby().

Há várias maneiras de se criar funções chave. Por exemplo, o método str.lower() pode servir como uma função chave para ordenações insensíveis à caixa. Alternativamente, uma função chave ad-hoc pode ser construída a partir de uma expressão lambda, como lambda r: (r[0], r[2]). Além disso, operator. attrgetter(), operator.itemgetter() e operator.methodcaller() são três construtores de função chave. Consulte o guia de Ordenação para ver exemplos de como criar e utilizar funções chave.

argumento nomeado

Veja argumento.

lambda

Uma função de linha anônima consistindo de uma única *expressão*, que é avaliada quando a função é chamada. A sintaxe para criar uma função lambda é lambda [parameters]: expression

LBYL

Iniciais da expressão em inglês "look before you leap", que significa algo como "olhe antes de pisar". Este estilo de codificação testa as pré-condições explicitamente antes de fazer chamadas ou buscas. Este estilo contrasta com a abordagem *EAFP* e é caracterizada pela presença de muitas instruções if.

Em um ambiente multithread, a abordagem LBYL pode arriscar a introdução de uma condição de corrida entre "o olhar" e "o pisar". Por exemplo, o código if key in mapping: return mapping[key] pode falhar se outra thread remover *key* do *mapping* após o teste, mas antes da olhada. Esse problema pode ser resolvido com travas ou usando a abordagem EAFP.

analisador léxico

Nome formal para o tokenizador; veja token.

lista

Uma *sequência* embutida no Python. Apesar do seu nome, é mais próximo de um vetor em outras linguagens do que uma lista encadeada, como o acesso aos elementos é da ordem O(1).

compreensão de lista

Uma maneira compacta de processar todos ou parte dos elementos de uma sequência e retornar os resultados em uma lista. result = [' $\{:\#04x\}$ '.format(x) for x in range(256) if x % 2 == 0] gera uma lista de strings contendo números hexadecimais (0x..) no intervalo de 0 a 255. A cláusula if é opcional. Se omitida, todos os elementos no range(256) serão processados.

carregador

Um objeto que carrega um módulo. Ele deve definir os métodos exec_module() e create_module() para implementar a interface Loader. Um carregador é normalmente retornado por um *localizador*. Veja também:

- · finders-and-loaders
- importlib.abc.Loader
- PEP 302

codificação da localidade

No Unix, é a codificação da localidade do LC_CTYPE, que pode ser definida com locale. setlocale(locale.LC_CTYPE, new_locale).

No Windows, é a página de código ANSI (ex: "cp1252").

No Android e no VxWorks, o Python usa "utf-8" como a codificação da localidade.

locale.getencoding() pode ser usado para obter a codificação da localidade.

Veja também tratador de erros e codificação do sistema de arquivos.

método mágico

Um sinônimo informal para um método especial.

mapeamento

Um objeto contêiner que tem suporte a pesquisas de chave arbitrária e implementa os métodos especificados nas collections.abc.Mapping ou collections.abc.MutableMapping classes base abstratas. Exemplos incluem dict, collections.defaultdict, collections.OrderedDict e collections.Counter.

localizador de metacaminho

Um *localizador* retornado por uma busca de sys.meta_path. Localizadores de metacaminho são relacionados a, mas diferentes de, *localizadores de entrada de caminho*.

Veja importlib.abc.MetaPathFinder para os métodos que localizadores de metacaminho implementam.

metaclasse

A classe de uma classe. Definições de classe criam um nome de classe, um dicionário de classe e uma lista de classes base. A metaclasse é responsável por receber estes três argumentos e criar a classe. A maioria das linguagens de programação orientadas a objetos provê uma implementação default. O que torna o Python especial é o fato de ser possível criar metaclasses personalizadas. A maioria dos usuários nunca vai precisar deste recurso, mas quando houver necessidade, metaclasses possibilitam soluções poderosas e elegantes. Metaclasses têm sido utilizadas para gerar registros de acesso a atributos, para incluir proteção contra acesso concorrente, rastrear a criação de objetos, implementar singletons, dentre muitas outras tarefas.

Mais informações podem ser encontradas em metaclasses.

método

Uma função que é definida dentro do corpo de uma classe. Se chamada como um atributo de uma instância daquela classe, o método receberá a instância do objeto como seu primeiro *argumento* (que comumente é chamado de self). Veja *função* e *escopo aninhado*.

ordem de resolução de métodos

Ordem de resolução de métodos é a ordem em que os membros de uma classe base são buscados durante a pesquisa. Veja python_2.3_mro para detalhes do algoritmo usado pelo interpretador do Python desde a versão 2.3.

módulo

Um objeto que serve como uma unidade organizacional de código Python. Os módulos têm um espaço de nomes contendo objetos Python arbitrários. Os módulos são carregados pelo Python através do processo de *importação*.

Veja também pacote.

spec de módulo

Um espaço de nomes que contém as informações relacionadas à importação usadas para carregar um módulo. Uma instância de importlib.machinery.ModuleSpec.

Veja também module-specs.

MRO

Veja ordem de resolução de métodos.

mutável

Objeto mutável é aquele que pode modificar seus valor mas manter seu id(). Veja também imutável.

tupla nomeada

O termo "tupla nomeada" é aplicado a qualquer tipo ou classe que herda de tupla e cujos elementos indexáveis também são acessíveis usando atributos nomeados. O tipo ou classe pode ter outras funcionalidades também.

Diversos tipos embutidos são tuplas nomeadas, incluindo os valores retornados por time.localtime() e os.stat(). Outro exemplo é sys.float_info:

```
>>> sys.float_info[1]  # acesso indexado
1024
>>> sys.float_info.max_exp  # acesso a campo nomeado
1024
>>> isinstance(sys.float_info, tuple)  # tipo de tupla
True
```

Algumas tuplas nomeadas são tipos embutidos (tal como os exemplos acima). Alternativamente, uma tupla nomeada pode ser criada a partir de uma definição de classe regular, que herde de tuple e que defina campos nomeados. Tal classe pode ser escrita a mão, ou ela pode ser criada herdando typing.NamedTuple ou com uma função fábrica collections.namedtuple(). As duas últimas técnicas também adicionam alguns métodos extras, que podem não ser encontrados quando foi escrita manualmente, ou em tuplas nomeadas embutidas.

espaço de nomes

O lugar em que uma variável é armazenada. Espaços de nomes são implementados como dicionários. Existem os espaços de nomes local, global e nativo, bem como espaços de nomes aninhados em objetos (em métodos). Espaços de nomes suportam modularidade ao prevenir conflitos de nomes. Por exemplo, as funções __builtin__.open() e os.open() são diferenciadas por seus espaços de nomes. Espaços de nomes também auxiliam na legibilidade e na manutenibilidade ao torar mais claro quais módulos implementam uma função. Escrever random.seed() ou itertools.izip(), por exemplo, deixa claro que estas funções são implementadas pelos módulos random e itertools respectivamente.

pacote de espaço de nomes

Um *pacote* que serve apenas como contêiner para subpacotes. Pacotes de espaços de nomes podem não ter representação física, e especificamente não são como um *pacote regular* porque eles não tem um arquivo __init__.py.

Pacotes de espaço de nomes permitem que vários pacotes instaláveis individualmente tenham um pacote pai comum. Caso contrário, é recomendado usar um *pacote regular*.

Para mais informações, veja PEP 420 e reference-namespace-package.

Veja também *módulo*.

escopo aninhado

A habilidade de referir-se a uma variável em uma definição de fechamento. Por exemplo, uma função definida dentro de outra pode referenciar variáveis da função externa. Perceba que escopos aninhados por padrão funcionam apenas por referência e não por atribuição. Variáveis locais podem ler e escrever no escopo mais interno. De forma similar, variáveis globais podem ler e escrever para o espaço de nomes global. O nonlocal permite escrita para escopos externos.

classe estilo novo

Antigo nome para o tipo de classes agora usado para todos os objetos de classes. Em versões anteriores do Python, apenas classes estilo podiam usar recursos novos e versáteis do Python, tais como __slots__, descritores, propriedades, __getattribute__(), métodos de classe, e métodos estáticos.

objeto

Qualquer dado que tenha estado (atributos ou valores) e comportamento definidos (métodos). Também a

última classe base de qualquer classe estilo novo.

escopo otimizado

Um escopo no qual os nomes das variáveis locais de destino são conhecidos de forma confiável pelo compilador quando o código é compilado, permitindo a otimização do acesso de leitura e gravação a esses nomes. Os espaços de nomes locais para funções, geradores, corrotinas, compreensões e expressões geradoras são otimizados desta forma. Nota: a maioria das otimizações de interpretador são aplicadas a todos os escopos, apenas aquelas que dependem de um conjunto conhecido de nomes de variáveis locais e não locais são restritas a escopos otimizados.

pacote

Um *módulo* Python é capaz de conter submódulos ou recursivamente, subpacotes. Tecnicamente, um pacote é um módulo Python com um atributo __path__.

Veja também pacote regular e pacote de espaço de nomes.

parâmetro

Uma entidade nomeada na definição de uma *função* (ou método) que específica um *argumento* (ou em alguns casos, argumentos) que a função pode receber. Existem cinco tipos de parâmetros:

• posicional-ou-nomeado: especifica um argumento que pode ser tanto posicional quanto nomeado. Esse é o tipo padrão de parâmetro, por exemplo foo e bar a seguir:

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

• *somente-posicional*: especifica um argumento que pode ser fornecido apenas por posição. Parâmetros somente-posicionais podem ser definidos incluindo o caractere / na lista de parâmetros da definição da função após eles, por exemplo *somentepos1* e *somentepos2* a seguir:

```
def func(somentepos1, somentepos2, /, posicional_ou_nomeado): ...
```

• somente-nomeado: especifica um argumento que pode ser passado para a função somente por nome. Parâmetros somente-nomeados podem ser definidos com um simples parâmetro var-posicional ou um * antes deles na lista de parâmetros na definição da função, por exemplo somente_nom1 and somente_nom2 a seguir:

```
def func(arg, *, somente_nom1, somente_nom2): ...
```

 var-posicional: especifica que uma sequência arbitrária de argumentos posicionais pode ser fornecida (em adição a qualquer argumento posicional já aceito por outros parâmetros). Tal parâmetro pode ser definido colocando um * antes do nome do parâmetro, por exemplo args a seguir:

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

• *var-nomeado*: especifica que, arbitrariamente, muitos argumentos nomeados podem ser fornecidos (em adição a qualquer argumento nomeado já aceito por outros parâmetros). Tal parâmetro pode definido colocando-se ** antes do nome, por exemplo *kwargs* no exemplo acima.

Parâmetros podem especificar tanto argumentos opcionais quanto obrigatórios, assim como valores padrão para alguns argumentos opcionais.

Veja também o termo *argumento* no glossário, a pergunta do FAQ sobre a diferença entre argumentos e parâmetros, a classe inspect.Parameter, a seção function e a PEP 362.

entrada de caminho

Um local único no caminho de importação que o localizador baseado no caminho consulta para encontrar módulos a serem importados.

localizador de entrada de caminho

Um *localizador* retornado por um chamável em sys.path_hooks (ou seja, um *gancho de entrada de caminho*) que sabe como localizar os módulos *entrada de caminho*.

Veja importlib.abc.PathEntryFinder para os métodos que localizadores de entrada de caminho implementam.

gancho de entrada de caminho

Um chamável na lista sys.path_hooks que retorna um *localizador de entrada de caminho* caso saiba como localizar módulos em uma *entrada de caminho* específica.

localizador baseado no caminho

Um dos localizadores de metacaminho padrão que procura por um caminho de importação de módulos.

objeto caminho ou similar

Um objeto representando um caminho de sistema de arquivos. Um objeto caminho ou similar é ou um objeto str ou bytes representando um caminho, ou um objeto implementando o protocolo os.PathLike. Um objeto que suporta o protocolo os.PathLike pode ser convertido para um arquivo de caminho do sistema str ou bytes, através da chamada da função os.fspath(); os.fsdecode() e os.fsencode() podem ser usadas para garantir um str ou bytes como resultado, respectivamente. Introduzido na PEP 519.

PEP

Proposta de melhoria do Python. Uma PEP é um documento de design que fornece informação para a comunidade Python, ou descreve uma nova funcionalidade para o Python ou seus predecessores ou ambientes. PEPs devem prover uma especificação técnica concisa e um racional para funcionalidades propostas.

PEPs têm a intenção de ser os mecanismos primários para propor novas funcionalidades significativas, para coletar opiniões da comunidade sobre um problema, e para documentar as decisões de design que foram adicionadas ao Python. O autor da PEP é responsável por construir um consenso dentro da comunidade e documentar opiniões dissidentes.

Veja PEP 1.

porção

Um conjunto de arquivos em um único diretório (possivelmente armazenado em um arquivo zip) que contribuem para um pacote de espaço de nomes, conforme definido em PEP 420.

argumento posicional

Veja argumento.

API provisória

Uma API provisória é uma API que foi deliberadamente excluída das bibliotecas padrões com compatibilidade retroativa garantida. Enquanto mudanças maiores para tais interfaces não são esperadas, contanto que elas sejam marcadas como provisórias, mudanças retroativas incompatíveis (até e incluindo a remoção da interface) podem ocorrer se consideradas necessárias pelos desenvolvedores principais. Tais mudanças não serão feitas gratuitamente – elas irão ocorrer apenas se sérias falhas fundamentais forem descobertas, que foram esquecidas anteriormente a inclusão da API.

Mesmo para APIs provisórias, mudanças retroativas incompatíveis são vistas como uma "solução em último caso" - cada tentativa ainda será feita para encontrar uma resolução retroativa compatível para quaisquer problemas encontrados.

Esse processo permite que a biblioteca padrão continue a evoluir com o passar do tempo, sem se prender em erros de design problemáticos por períodos de tempo prolongados. Veja PEP 411 para mais detalhes.

pacote provisório

Veja API provisória.

Python 3000

Apelido para a linha de lançamento da versão do Python 3.x (cunhada há muito tempo, quando o lançamento da versão 3 era algo em um futuro muito distante.) Esse termo possui a seguinte abreviação: "Py3k".

Pythônico

Uma ideia ou um pedaço de código que segue de perto as formas de escritas mais comuns da linguagem Python, ao invés de implementar códigos usando conceitos comuns a outras linguagens. Por exemplo, um formato comum em Python é fazer um laço sobre todos os elementos de uma iterável usando a instrução for. Muitas outras linguagens não têm esse tipo de construção, então as pessoas que não estão familiarizadas com o Python usam um contador numérico:

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

Ao contrário do método mais limpo, Pythônico:

```
for parte in comida:
    print(parte)
```

nome qualificado

Um nome pontilhado (quando 2 termos são ligados por um ponto) que mostra o "path" do escopo global de um módulo para uma classe, função ou método definido num determinado módulo, conforme definido pela **PEP** 3155. Para funções e classes de nível superior, o nome qualificado é o mesmo que o nome do objeto:

Quando usado para se referir a módulos, o *nome totalmente qualificado* significa todo o caminho pontilhado para o módulo, incluindo quaisquer pacotes pai, por exemplo: email.mime.text:

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

contagem de referências

O número de referências a um objeto. Quando a contagem de referências de um objeto cai para zero, ele é desalocado. Alguns objetos são *imortais* e têm contagens de referências que nunca são modificadas e, portanto, os objetos nunca são desalocados. A contagem de referências geralmente não é visível para o código Python, mas é um elemento-chave da implementação do *CPython*. Os programadores podem chamar a função sys. getrefcount () para retornar a contagem de referências para um objeto específico.

pacote regular

Um pacote tradicional, como um diretório contendo um arquivo __init__.py.

Veja também pacote de espaço de nomes.

REPL

Um acrônimo para "read-eval-print loop", outro nome para o console interativo do interpretador.

__slots_

Uma declaração dentro de uma classe que economiza memória pré-declarando espaço para atributos de instâncias, e eliminando dicionários de instâncias. Apesar de popular, a técnica é um tanto quanto complicada de acertar, e é melhor se for reservada para casos raros, onde existe uma grande quantidade de instâncias em uma aplicação onde a memória é crítica.

sequência

Um *iterável* com suporte para acesso eficiente a seus elementos através de índices inteiros via método especial __getitem__() e que define o método __len__() que devolve o tamanho da sequência. Alguns tipos de sequência embutidos são: list, str, tuple, e bytes. Note que dict também tem suporte para __getitem__() e __len__(), mas é considerado um mapeamento e não uma sequência porque a busca usa uma chave *hasheável* arbitrária em vez de inteiros.

A classe base abstrata collections.abc.Sequence define uma interface mais rica que vai além de apenas __getitem__() e __len__(), adicionando count(), index(), __contains__(), e __reversed__(). Tipos que implementam essa interface podem ser explicitamente registrados usando register(). Para mais documentação sobre métodos de sequências em geral, veja Operações comuns de sequências.

compreensão de conjunto

Uma maneira compacta de processar todos ou parte dos elementos em iterável e retornar um conjunto com os resultados. results = {c for c in 'abracadabra' if c not in 'abc'} gera um conjunto de strings {'r', 'd'}. Veja comprehensions.

despacho único

Uma forma de despacho de *função genérica* onde a implementação é escolhida com base no tipo de um único argumento.

fatia

Um objeto geralmente contendo uma parte de uma *sequência*. Uma fatia é criada usando a notação de subscrito [] pode conter também até dois pontos entre números, como em variable_name[1:3:5]. A notação de suporte (subscrito) utiliza objetos slice internamente.

suavemente descontinuado

Uma API suavemente descontinuada não deve ser usada em código novo, mas é seguro para código já existente usá-la. A API continua documentada e testada, mas não será aprimorada mais.

A descontinuação suave, diferentemente da descontinuação normal, não planeja remover a API e não emitirá avisos.

Veja PEP 387: Descontinuação suave.

método especial

Um método que é chamado implicitamente pelo Python para executar uma certa operação em um tipo, como uma adição por exemplo. Tais métodos tem nomes iniciando e terminando com dois underscores. Métodos especiais estão documentados em specialnames.

instrução

Uma instrução é parte de uma suíte (um "bloco" de código). Uma instrução é ou uma *expressão* ou uma de várias construções com uma palavra reservada, tal como if, while ou for.

verificador de tipo estático

Uma ferramenta externa que lê o código Python e o analisa, procurando por problemas como tipos incorretos. Consulte também *dicas de tipo* e o módulo typing.

referência forte

Na API C do Python, uma referência forte é uma referência a um objeto que pertence ao código que contém a referência. A referência forte é obtida chamando Py_INCREF () quando a referência é criada e liberada com Py_DECREF () quando a referência é excluída.

A função Py_NewRef() pode ser usada para criar uma referência forte para um objeto. Normalmente, a função Py_DECREF() deve ser chamada na referência forte antes de sair do escopo da referência forte, para evitar o vazamento de uma referência.

Veja também referência emprestada.

codificador de texto

Uma string em Python é uma sequência de pontos de código Unicode (no intervalo U+0000-U+10FFFF). Para armazenar ou transferir uma string, ela precisa ser serializada como uma sequência de bytes.

A serialização de uma string em uma sequência de bytes é conhecida como "codificação" e a recriação da string a partir de uma sequência de bytes é conhecida como "decodificação".

Há uma variedade de diferentes serializações de texto codecs, que são coletivamente chamadas de "codificações de texto".

arquivo texto

Um *objeto arquivo* apto a ler e escrever objetos str. Geralmente, um arquivo texto, na verdade, acessa um fluxo de dados de bytes e captura o *codificador de texto* automaticamente. Exemplos de arquivos texto são: arquivos abertos em modo texto ('r' or 'w'), sys.stdin, sys.stdout, e instâncias de io.StringIO.

Veja também arquivo binário para um objeto arquivo apto a ler e escrever objetos byte ou similar.

token

Uma pequena unidade de código-fonte, gerada pelo analisador léxico (também chamado de *tokenizador*). Nomes, números, strings, operadores, quebras de linha e similares são representados por tokens.

O módulo tokenize expõe o analisador léxico do Python. O módulo token contém informações sobre os vários tipos de tokens.

aspas triplas

Uma string que está definida com três ocorrências de aspas duplas (") ou apóstrofos ('). Enquanto elas não fornecem nenhuma funcionalidade não disponível com strings de aspas simples, elas são úteis para inúmeras razões. Elas permitem que você inclua aspas simples e duplas não escapadas dentro de uma string, e elas podem utilizar múltiplas linhas sem o uso de caractere de continuação, fazendo-as especialmente úteis quando escrevemos documentação em docstrings.

tipo

O tipo de um objeto Python determina qual classe de objeto ele é; cada objeto tem um tipo. Um tipo de objeto é acessível pelo atributo __class__ ou pode ser recuperado com type (obj).

apelido de tipo

Um sinônimo para um tipo, criado através da atribuição do tipo para um identificador.

Apelidos de tipo são úteis para simplificar dicas de tipo. Por exemplo:

pode tornar-se mais legível desta forma:

```
Cor = tuple[int, int, int]

def remove_tons_de_cinza(cores: list[Cor]) -> list[Cor]:
    pass
```

Veja typing e PEP 484, a qual descreve esta funcionalidade.

dica de tipo

Uma *anotação* que especifica o tipo esperado para uma variável, um atributo de classe, ou um parâmetro de função ou um valor de retorno.

Dicas de tipo são opcionais e não são forçadas pelo Python, mas elas são úteis para *verificadores de tipo estático*. Eles também ajudam IDEs a completar e refatorar código.

Dicas de tipos de variáveis globais, atributos de classes, e funções, mas não de variáveis locais, podem ser acessadas usando typing.get_type_hints().

Veja typing e PEP 484, a qual descreve esta funcionalidade.

novas linhas universais

Uma maneira de interpretar fluxos de textos, na qual todos estes são reconhecidos como caracteres de fim de linha: a convenção para fim de linha no Unix '\n', a convenção no Windows '\r\n', e a antiga convenção no Macintosh '\r'. Veja PEP 278 e PEP 3116, bem como bytes.splitlines() para uso adicional.

anotação de variável

Uma anotação de uma variável ou um atributo de classe.

Ao fazer uma anotação de uma variável ou um atributo de classe, a atribuição é opcional:

```
class C:
campo: 'anotação'
```

Anotações de variáveis são normalmente usadas para *dicas de tipo*: por exemplo, espera-se que esta variável receba valores do tipo int:

```
contagem: int = 0
```

A sintaxe de anotação de variável é explicada na seção annassign.

Veja *anotação de função*, **PEP 484** e **PEP 526**, que descrevem esta funcionalidade. Veja também annotations-howto para as melhores práticas sobre como trabalhar com anotações.

ambiente virtual

Um ambiente de execução isolado que permite usuários Python e aplicações instalarem e atualizarem pacotes Python sem interferir no comportamento de outras aplicações Python em execução no mesmo sistema.

Veja também venv.

máquina virtual

Um computador definido inteiramente em software. A máquina virtual de Python executa o *bytecode* emitido pelo compilador de bytecode.

Zen do Python

Lista de princípios de projeto e filosofias do Python que são úteis para a compreensão e uso da linguagem. A lista é exibida quando se digita "import this" no console interativo.

APÊNDICE B

Sobre esta documentação

A documentação do Python é gerada a partir de fontes reStructuredText usando Sphinx, um gerador de documentação criado originalmente para Python e agora mantido como um projeto independente.

O desenvolvimento da documentação e de suas ferramentas é um esforço totalmente voluntário, como Python em si. Se você quer contribuir, por favor dê uma olhada na página reporting-bugs para informações sobre como fazer. Novos voluntários são sempre bem-vindos!

Agradecimentos especiais para:

- Fred L. Drake, Jr., o criador do primeiro conjunto de ferramentas para documentar Python e autor de boa parte do conteúdo;
- O projeto Docutils por criar reStructuredText e o pacote Docutils;
- Fredrik Lundh, pelo seu projeto de referência alternativa em Python, do qual Sphinx pegou muitas boas ideias.

B.1 Contribuidores da documentação do Python

Muitas pessoas tem contribuído para a linguagem Python, sua biblioteca padrão e sua documentação. Veja Misc/ACKS na distribuição do código do Python para ver uma lista parcial de contribuidores.

Tudo isso só foi possível com o esforço e a contribuição da comunidade Python, por isso temos essa maravilhosa documentação – Obrigado a todos!

APÊNDICE C

História e Licença

C.1 História do software

Python foi criado no início dos anos 1990 por Guido van Rossum no Stichting Mathematisch Centrum (CWI, veja https://www.cwi.nl) na Holanda como sucessor de uma linguagem chamada ABC. Guido continua sendo o principal autor do Python, embora inclua muitas contribuições de outros.

Em 1995, Guido continuou seu trabalho em Python na Corporation for National Research Initiatives (CNRI, veja https://www.cnri.reston.va.us) em Reston, Virgínia, onde lançou várias versões do software.

Em maio de 2000, Guido e a equipe de desenvolvimento do núcleo Python mudaram-se para BeOpen.com para formar a equipe BeOpen PythonLabs. Em outubro do mesmo ano, a equipe PythonLabs mudou-se para a Digital Creations, que se tornou Zope Corporation. Em 2001, a Python Software Foundation (PSF, veja https://www.python.org/psf/) foi formada, uma organização sem fins lucrativos criada especificamente para possuir Propriedade Intelectual relacionada ao Python. A Zope Corporation era um membro patrocinador da PSF.

Todas as versões do Python são de código aberto (consulte https://opensource.org para a definição de código aberto). Historicamente, a maioria, mas não todas, versões do Python também são compatíveis com GPL; a tabela abaixo resume os vários lançamentos.

Versão	Derivada de	Ano	Proprietário	Compatível com a GPL? (1)
0.9.0 a 1.2	n/a	1991-1995	CWI	sim
1.3 a 1.5.2	1.2	1995-1999	CNRI	sim
1.6	1.5.2	2000	CNRI	não
2.0	1.6	2000	BeOpen.com	não
1.6.1	1.6	2001	CNRI	sim (2)
2.1	2.0+1.6.1	2001	PSF	não
2.0.1	2.0+1.6.1	2001	PSF	sim
2.1.1	2.1+2.0.1	2001	PSF	sim
2.1.2	2.1.1	2002	PSF	sim
2.1.3	2.1.2	2002	PSF	sim
2.2 e acima	2.1.1	2001-agora	PSF	sim

1 Nota

- (1) Compatível com a GPL não significa que estamos distribuindo Python sob a GPL. Todas as licenças do Python, ao contrário da GPL, permitem distribuir uma versão modificada sem fazer alterações em código aberto. As licenças compatíveis com a GPL possibilitam combinar o Python com outro software lançado sob a GPL; os outros não.
- (2) De acordo com Richard Stallman, 1.6.1 não é compatível com GPL, porque sua licença tem uma cláusula de escolha de lei. De acordo com a CNRI, no entanto, o advogado de Stallman disse ao advogado da CNRI que 1.6.1 "não é incompatível" com a GPL.

Graças aos muitos voluntários externos que trabalharam sob a direção de Guido para tornar esses lançamentos possíveis.

C.2 Termos e condições para acessar ou usar Python

O software e a documentação do Python são licenciados sob a Python Software Foundation License Versão 2.

A partir do Python 3.8.6, exemplos, receitas e outros códigos na documentação são licenciados duplamente sob o Licença PSF versão 2 e a *Licença BSD de Zero Cláusula*.

Alguns softwares incorporados ao Python estão sob licenças diferentes. As licenças são listadas com o código abrangido por essa licença. Veja *Licenças e Reconhecimentos para Software Incorporado* para uma lista incompleta dessas licenças.

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 © 2001-2024 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.

(continua na próxima página)

- 7. Nothing in this License Agreement shall be deemed to create any relationship of agency, partnership, or joint venture between PSF and Licensee. This License Agreement does not grant permission to use PSF trademarks or trade name in a trademark sense to endorse or promote products or services of Licensee, or any third party.
- 8. By copying, installing or otherwise using Python, Licensee agrees to be bound by the terms and conditions of this License Agreement.

C.2.2 ACORDO DE LICENCIAMENTO DA BEOPEN.COM PARA PYTHON 2.0

ACORDO DE LICENCIAMENTO DA BEOPEN DE FONTE ABERTA DO PYTHON VERSÃO 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 CONTRATO DE LICENÇA DA CNRI PARA O PYTHON 1.6.1

- 1. This LICENSE AGREEMENT is between the Corporation for National Research Initiatives, having an office at 1895 Preston White Drive, Reston, VA 20191 ("CNRI"), and the Individual or Organization ("Licensee") accessing and otherwise using Python 1.6.1 software in source or binary form and its associated documentation.
- 2. Subject to the terms and conditions of this License Agreement, CNRI hereby grants Licensee a nonexclusive, royalty-free, world-wide license to reproduce, analyze, test, perform and/or display publicly, prepare derivative works, distribute, and otherwise use Python 1.6.1 alone or in any derivative version, provided, however, that CNRI's License Agreement and CNRI's notice of copyright, i.e., "Copyright © 1995-2001 Corporation for National Research Initiatives; All Rights Reserved" are retained in Python 1.6.1 alone or in any derivative version prepared by Licensee. Alternately, in lieu of CNRI's License Agreement, Licensee may substitute the following text (omitting the quotes): "Python 1.6.1 is made available subject to the terms and conditions in CNRI's License Agreement. This Agreement together with Python 1.6.1 may be located on the internet using the following unique, persistent identifier (known as a handle): 1895.22/1013. This Agreement may also be obtained from a proxy server on the internet using the following URL: http://hdl.handle.net/1895.22/1013".
- 3. In the event Licensee prepares a derivative work that is based on or incorporates Python 1.6.1 or any part thereof, and wants to make the derivative work available to others as provided herein, then Licensee hereby agrees to include in any such work a brief summary of the changes made to Python 1.6.1.
- 4. CNRI is making Python 1.6.1 available to Licensee on an "AS IS" basis. CNRI MAKES NO REPRESENTATIONS OR WARRANTIES, EXPRESS OR IMPLIED. BY WAY OF EXAMPLE, BUT NOT LIMITATION, CNRI MAKES NO AND DISCLAIMS ANY REPRESENTATION OR WARRANTY OF MERCHANTABILITY OR FITNESS FOR ANY PARTICULAR PURPOSE OR THAT THE USE OF PYTHON 1.6.1 WILL NOT INFRINGE ANY THIRD PARTY RIGHTS.
- 5. CNRI SHALL NOT BE LIABLE TO LICENSEE OR ANY OTHER USERS OF PYTHON 1.6.1 FOR ANY INCIDENTAL, SPECIAL, OR CONSEQUENTIAL DAMAGES OR LOSS AS A RESULT OF MODIFYING, DISTRIBUTING, OR OTHERWISE USING PYTHON 1.6.1, OR ANY DERIVATIVE THEREOF, EVEN IF ADVISED OF THE POSSIBILITY THEREOF.
- 6. This License Agreement will automatically terminate upon a material breach of its terms and conditions.
- 7. This License Agreement shall be governed by the federal intellectual property law of the United States, including without limitation the federal copyright law, and, to the extent such U.S. federal law does not apply, by the law of the Commonwealth of Virginia, excluding Virginia's conflict of law provisions. Notwithstanding the foregoing, with regard to derivative works based on Python 1.6.1 that incorporate non-separable material that was previously distributed under the GNU General Public License (GPL), the law of the Commonwealth of Virginia shall govern this License Agreement only as to issues arising under or with respect to Paragraphs 4, 5, and 7 of this License Agreement. Nothing in this License Agreement shall be deemed to create any relationship of agency, partnership, or joint venture between CNRI and Licensee. This License Agreement does not grant permission to use CNRI trademarks or trade name in a trademark sense to endorse or promote products or services of Licensee, or any third party.

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

C.2.4 ACORDO DE LICENÇA DA CWI PARA PYTHON 0.9.0 A 1.2

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

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

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

C.2.5 ZERO-CLAUSE BSD LICENSE FOR CODE IN THE PYTHON DOCUMENTA-TION

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 Licenças e Reconhecimentos para Software Incorporado

Esta seção é uma lista incompleta, mas crescente, de licenças e reconhecimentos para softwares de terceiros incorporados na distribuição do Python.

C.3.1 Mersenne Twister

A extensão C _random subjacente ao módulo random inclui código baseado em um download de http://www.math.sci.hiroshima-u.ac.jp/~m-mat/MT/MT2002/emt19937ar.html. A seguir estão os comentários literais do código original:

A C-program for MT19937, with initialization improved 2002/1/26. Coded by Takuji Nishimura and Makoto Matsumoto.

Before using, initialize the state by using init_genrand(seed)

or init_by_array(init_key, key_length).

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

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

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

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

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

C.3.2 Soquetes

O módulo socket usa as funções getaddrinfo() e getnameinfo(), que são codificadas em arquivos de origem separados do Projeto WIDE, https://www.wide.ad.jp/.

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

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

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

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

C.3.3 Serviços de soquete assíncrono

Os módulos test.support.asynchat e test.support.asyncore contêm o seguinte aviso:

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 Gerenciamento de cookies

O módulo http.cookies contém o seguinte aviso:

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

All Rights Reserved

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

Timothy O'Malley DISCLAIMS ALL WARRANTIES WITH REGARD TO THIS

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

C.3.5 Rastreamento de execução

O módulo trace contém o seguinte aviso:

```
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 Funções UUencode e UUdecode

O codec uu contém o seguinte aviso:

```
Copyright 1994 by Lance Ellinghouse
Cathedral City, California Republic, United States of America.

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

LANCE ELLINGHOUSE DISCLAIMS ALL WARRANTIES WITH REGARD TO
THIS SOFTWARE, INCLUDING ALL IMPLIED WARRANTIES OF MERCHANTABILITY AND
FITNESS, IN NO EVENT SHALL LANCE ELLINGHOUSE CENTRUM BE LIABLE
```

FOR ANY SPECIAL, INDIRECT OR CONSEQUENTIAL DAMAGES OR ANY DAMAGES WHATSOEVER RESULTING FROM LOSS OF USE, DATA OR PROFITS, WHETHER IN AN ACTION OF CONTRACT, NEGLIGENCE OR OTHER TORTIOUS ACTION, ARISING OUT OF OR IN CONNECTION WITH THE USE OR PERFORMANCE OF THIS SOFTWARE.

Modified by Jack Jansen, CWI, July 1995:

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

C.3.7 Chamadas de procedimento remoto XML

O módulo xmlrpc.client contém o seguinte aviso:

The XML-RPC client interface is

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

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

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

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

C.3.8 test_epoll

O módulo test.test_epoll contém o seguinte aviso:

Copyright (c) 2001-2006 Twisted Matrix Laboratories.

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

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

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

C.3.9 kqueue de seleção

O módulo select contém o seguinte aviso para a interface do kqueue:

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

O arquivo Python/pyhash.c contém a implementação de Marek Majkowski do algoritmo SipHash24 de Dan Bernstein. Contém a seguinte nota:

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

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

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

```
</MIT License>
Original location:
   https://github.com/majek/csiphash/
Solution inspired by code from:
   Samuel Neves (supercop/crypto_auth/siphash24/little)
   djb (supercop/crypto_auth/siphash24/little2)
   Jean-Philippe Aumasson (https://131002.net/siphash/siphash24.c)
```

C.3.11 strtod e dtoa

O arquivo Python/dtoa.c, que fornece as funções C dtoa e strtod para conversão de duplas de C para e de strings, é derivado do arquivo com o mesmo nome de David M. Gay, atualmente disponível em https://web.archive.org/web/20220517033456/http://www.netlib.org/fp/dtoa.c. O arquivo original, conforme recuperado em 16 de março de 2009, contém os seguintes avisos de direitos autorais e de licenciamento:

C.3.12 OpenSSL

Os módulos hashlib, posix e ssl usam a biblioteca OpenSSL para desempenho adicional se forem disponibilizados pelo sistema operacional. Além disso, os instaladores do Windows e do Mac OS X para Python podem incluir uma cópia das bibliotecas do OpenSSL, portanto incluímos uma cópia da licença do OpenSSL aqui: Para o lançamento do OpenSSL 3.0, e lançamentos posteriores derivados deste, se aplica a Apache License v2:

```
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.
```

- "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,

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 $% \left(1\right) =\left(1\right) \left(1$ 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

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

A extensão pyexpat é construída usando uma cópia incluída das fontes de expatriadas, a menos que a compilação esteja configurada —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

A extensão C _ctypes subjacente ao módulo ctypes é construída usando uma cópia incluída das fontes do libffi, a menos que a construção esteja configurada com --with-system-libffi:

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

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

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

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

C.3.15 zlib

A extensão zlib é construída usando uma cópia incluída das fontes zlib se a versão do zlib encontrada no sistema for muito antiga para ser usada na construção:

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:

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

Jean-loup Gailly Mark Adler

jloup@gzip.org madler@alumni.caltech.edu

C.3.16 cfuhash

A implementação da tabela de hash usada pelo tracemalloc é baseada no projeto cfuhash:

Copyright (c) 2005 Don Owens All rights reserved.

This code is released under the BSD license:

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

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

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT,

INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

C.3.17 libmpdec

A extensão C _decimal subjacente ao módulo decimal é construída usando uma cópia incluída da biblioteca libmpdec, a menos que a construção esteja configurada com --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 Conjunto de testes C14N do W3C

O conjunto de testes C14N 2.0 no pacote test (Lib/test/xmltestdata/c14n-20/) foi recuperado do site do W3C em https://www.w3.org/TR/xml-c14n2-testcases/ e é distribuído sob a licença BSD de 3 cláusulas:

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

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

- * Redistributions of works must retain the original copyright notice, this list of conditions and the following disclaimer.
- * Redistributions in binary form must reproduce the original copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
- * Neither the name of the W3C nor the names of its contributors may be

used to endorse or promote products derived from this work without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, 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

Licença MIT:

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

Partes do módulo asyncio são incorporadas do uvloop 0.16, que é distribuído sob a licença MIT:

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.

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)

O arquivo Python/qsbr.c é adaptado do esquema de recuperação de memória segura "Global Unbounded Sequences" do FreeBSD em subr_smr.c. O arquivo é distribuído sob a licença BSD de 2 cláusulas:

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:

- 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.

APÊNDICE D

Direitos autorais

Python e essa documentação é:

Copyright © 2001-2024 Python Software Foundation. Todos os direitos reservados.

Copyright © 2000 BeOpen.com. Todos os direitos reservados.

Copyright © 1995-2000 Corporation for National Research Initiatives. Todos os direitos reservados.

Copyright © 1991-1995 Stichting Mathematisch Centrum. Todos os direitos reservados.

Veja: História e Licença para informações completas de licença e permissões.

Índice

Não alfabético	contíguo, 14
,11	contíguo C, 14
>>>, 11	contíguo Fortran, 14 corrotina, 14
future, 17	CPython, 15
slots, 25	02 1 011011, 22
A	D
ambiente virtual, 28 analisador léxico, 20 anotação, 11 anotação de função, 17 anotação de variável, 27	decorador, 15 descritor, 15 desligamento do interpretador, 19 despacho único, 26 dica de tipo, 27 dicionário, 15 divisão pelo piso, 17 docstring, 15
argumento, 11	E
argumento posicional, 24 arquivo binário, 13 arquivo texto, 26 aspas triplas, 27	EAFP, 16 entrada de caminho, 23 escopo aninhado, 22 escopo otimizado, 23 espaço de nomes, 22 especial
В	método, 26
, -	expressão, 16
bytecode, 13	expressão geradora, 18
C	F
caminho de importação, 19 carregador, 21 chamável, 13 classe, 13 classe base abstrata, 11 classe estilo novo, 22 codificação da localidade, 21	f-string, 16 fatia, 26 função, 17 função chave, 20 função de corrotina, 15 função de retorno, 13 função genérica, 18
codificador de texto, 26 coleta de lixo, 17	G
compreensão de conjunto, 26 compreensão de dicionário, 15 compreensão de lista, 20 contagem de referências, 25 contexto, 14	gancho de entrada de caminho, 24 gerador, 17 gerador assíncrono, 12 gerenciador de contexto, 14 gerenciador de contexto assíncrono, 12 GIL, 18

H	Р
hasheável, 18	pacote, 23
1	pacote de espaço de nomes, 22
l	pacote provisório, 24
IDLE, 19	pacote regular, 25
imortal, 19	parâmetro, 23
importação, 19	PEP, 24
importador, 19	porção, 24
imutável, 19	Propostas de Melhorias do Python
instrução, 26	PEP 1,24
interativo, 19	PEP 238, 17
interpretado, 19	PEP 278, 27
iterador, 20	PEP 302,21
iterador assíncrono, 12	PEP 343, 14
iterador gerador, 18	PEP 362, 12, 23
iterador gerador assíncrono, 12	PEP 411,24
iterável, 19	PEP 420, 22, 24
iterável assíncrono, 12	PEP 443, 18
1	PEP 483, 18
L	PEP 484, 11, 17, 18, 27, 28
lambda, 20	PEP 492, 12, 14, 15
LBYL, 20	PEP 498, 16
lista, 20	PEP 519, 24
localizador, 17	PEP 525, 12
localizador baseado no caminho, 24	PEP 526, 11, 28
localizador de entrada de caminho, 23	PEP 585, 18
localizador de metacaminho, 21	PEP 683, 19
	PEP 703, 17, 18
M	PEP 3116,27
mágico	PEP 3155, 25
método, 21	protocolo de gerenciamento de contexto, 14
mapeamento, 21	pyc baseado em hash, 18
máquina virtual, 28	Python 3000, 24
metaclasse, 21	PYTHON_GIL, 18
método, 21	Pythônico, 24
especial, 26	D
mágico, 21	R
método especial, 26	referência emprestada, 13
método mágico, 21	referência forte, 26
módulo, 21	REPL, 25
módulo de extensão, 16	
MRO, 22	S
mutável, 22	sequência, 25
N.I.	spec de módulo, 21
N	suavemente descontinuado, 26
nome qualificado, 25	
novas linhas universais, 27	T
número complexo, 14	threads livres, 17
_	tipagem pato, 16
O	tipo, 27
objeto, 22	tipo genérico, 18
objeto arquivo, 16	token, 26
objeto arquivo ou similar, 16	tratador de erros e codificação do
objeto byte ou similar, 13	sistema de arquivos, 16
objeto caminho ou similar, 24	trava global do interpretador, 18
ordem de resolução de métodos, 21	tupla nomeada, 22

54 Índice

٧

```
variável de ambiente
PYTHON_GIL, 18

variável de classe, 13

variável de clausura, 14

variável de contexto, 14

variável livre, 17

verificador de tipo estático, 26

visão de dicionário, 15

Z

Zen do Python, 28
```

Índice 55