# Notas da Tradução

Esta tradução é derivada de conteúdos de terceiros, consulte o documento original para mais informações e sua licença

# Grossário

### >>>

O `prompt` padrão do terminal interativo do Python. Visto frequentemente em exemplos código que podem ser executados interativamente pelo interpretador.

### ...

Pode referir a:

- O `prompt` padrão do Python ao digitar o código para um bloco de código indentado, quando com um par de delimitadores esquerdo e direito correspondentes (parênteses, colchetes, chaves ou aspas tripas) ou após especificar um decorador.
- A contante embutida Ellipsis (Reticências)

### 2to3

Uma ferramenta que tenta converter código escrito em Python 2.x para um código escrito em Python 3.x, gerenciando a maioria das incopatibilidades que podem ser detectadas ao analisizar sintaticamente o código e atravesar a arvóre sintática. 2to3 está disponível na blibioteca pradão como lib2to3; um ponto de entrada autônomo é fornecido como `Tools/scripts/2t03`. Veja 2to3 - Automated Python 2 to 3 code translation

### Classe abstrata base

Classes básicas abstratas complementam a tipagem duck-typing ao fornecer uma maneira de definir interface quando outras técnicas como hsattr() seriam desajeitadas e sultimente erradas (como por exemplo  magic methods ). ABCs (Abstract Base Classes) introduzem subclasses virtuais, que são classes que não herdam a partir de uma classe, mas continuam sendo reconhecidas pelas funcões 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 modúlos (no módulo importlib.abc) . Você pode criar suas próprias ABCs com o módulo abc

## Anotação

Um rótulo associado a uma variável, um atributo de uma classe ou um parâmetro de uma função, usado por convecção como type-hint. Anotações de variáveis locais não podem serem acessadas durante o tempo de execução, mas anotações globais de variáveis, atributos de classes, e funções são amarzenadas no atributo especial `__annotations__` de módulos, classes e funções, respectivamente. Veja Anotação de uma Variável, Anotação de uma Função, PEP 484 e PEP 526, ao quais descrevem esta funcionalidade. Veja também Melhores Práticas de Anotação que descreve as melhores práticas ao trabalhar com anotações.

## Argumento

Um valor passado para uma função ou método ao ser chamado. Há dois tipos de argumentos:

- argumentos nomeados (ou argumentos de palavra-chave, em tradução literal de ***keywords arguments***): um argumento precedido por um exemplo (e.g. `name=`) em uma chamada de uma função ou passados como um valor em um dicionário precedido por `**`. Por exemplo, `3` e `5` são ambos argumentos nomeados na seguinte chamada a complex():

```
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 começo de uma lista de argumentos ou serem passados como elementos de um itinerável precedido por `*`. Por exemplo, `3` e `5` são ambos argumentos posicionais na seguintes chamadas:

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

Argumentos são atribuidos a variáveis locais no corpo da função. Veja a seção Chamadas com as regras que governam essa atribuição. Sintaticamente, qualquer expressão pode ser usada para representar um argumento, e o valor resultante é atribuído à variável local. Veja também a entrada do glossário parâmetro, as questões FAQ (Perguntas Frequentes, de "Frequently Asked Questions") sobre a diferência entre argumentos e parâmetros e PEP 362