
Migrando código optparse para argparse

Release 3.13.7

Guido van Rossum and the Python development team

outubro 01, 2025

Python Software Foundation
Email: docs@python.org

Sumário

O módulo `argparse` oferece vários recursos de nível mais alto não fornecidos nativamente pelo módulo `optparse`, incluindo:

- Tratando argumentos posicionais.
- Prover suporte a subcomandos.
- Permitir prefixos alternativos de opções como `+` e `/`.
- Manipular argumentos de estilo “zero ou mais” e “um ou mais”.
- Produzir mensagens de uso mais informativas.
- Fornecer uma interface muito mais simples para `type` e `action` personalizados.

Originalmente, o módulo `argparse` tentou manter a compatibilidade com `optparse`. No entanto, as diferenças fundamentais de design entre suportar o processamento de opções de linha de comando declarativas (enquanto deixa o processamento de argumentos posicionais para o código da aplicação) e suportar opções nomeadas e argumentos posicionais na interface declarativa significam que a API divergiu daquela de `optparse` ao longo do tempo.

Conforme descrito em [choosing-an-argument-parser](#), as aplicações que atualmente usam `optparse` e estão satisfeitos com a forma como ele funciona podem continuar usando `optparse`.

Os desenvolvedores de aplicações que estão pensando em migrar também devem revisar a lista de diferenças comportamentais intrínsecas descritas nessa seção antes de decidir se a migração é desejável ou não.

Para aplicações que optam por migrar de `optparse` para `argparse`, as seguintes sugestões devem ser úteis:

- Substituir todas as chamadas de `optparse.OptionParser.add_option()` por chamadas de `ArgumentParser.add_argument()`.
- Substituir `(options, args) = parser.parse_args()` por `args = parser.parse_args()` e adicionar chamadas adicionais a `ArgumentParser.add_argument()` para os argumentos posicionais. Tenha em mente que o que anteriormente era chamado de `options`, agora no contexto do `argparse` é chamado de `args`.
- Substituir `optparse.OptionParser.disable_interspersed_args()` usando `parser.intermixed_args()` em vez de `parse_args()`.
- Substituir ações de função de retorno e argumentos nomeados `callback_*` por argumentos `type` ou `action`.

- Substituir nomes de strings para argumentos nomeados `type` pelos objetos de tipo correspondentes (por exemplo, `int`, `float`, `complex`, etc).
- Substituir `optparse.Values` por `Namespace` e `optparse.OptionError` e `optparse.OptionValueError` por `ArgumentError`.
- Substituir strings com argumentos implícitos tal como `%default` ou `%prog` pela sintaxe padrão do Python para usar dicionários para formatar strings, ou seja, `%(default)s` e `%(prog)s`.
- Substituir o argumento `version` do construtor do `OptionParser` por uma chamada a `parser.add_argument('--version', action='version', version='<the version>')`.