
Atualizando código optparse

Release 3.13.0

Guido van Rossum and the Python development team

outubro 29, 2024

**Python Software Foundation
Email: docs@python.org**

Sumário

Originalmente, o módulo `argparse` tentou manter a compatibilidade com `optparse`. No entanto, `optparse` era difícil de estender de forma transparente, particularmente com as mudanças necessárias para prover suporte a especificadores `nargs=` e melhores mensagens de uso. Quando quase tudo em `optparse` tinha sido copiado e colado ou corrigido em tempo de execução, não mais parecia prático tentar manter a compatibilidade com versões anteriores.

O módulo `argparse` melhora em comparação ao módulo `optparse` em vários aspectos, 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.

Um caminho de atualização parcial de `optparse` para `argparse`:

- 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 `parse_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`.
`add_argument('--version', action='version', version='<the version>').`