

optparse!

Hunter Blanks, hblanks@monetate.com / [@tildehblanks](https://twitter.com/tildehblanks)
github.com/hblanks/talks

15.5. optparse — Parser for command line options

New in version 2.3.

Deprecated since version 2.7: The `optparse` module is deprecated and will not be developed further; development will continue with the `argparse` module.

Source code: [Lib/optparse.py](#)

`optparse` is a more convenient, flexible, and powerful library for parsing command-line options than the old `getopt` module. `optparse` uses a more declarative style of command-line parsing: you create an instance of `OptionParser`, populate it with options, and parse the command line. `optparse` allows users to specify options in the conventional GNU/POSIX syntax, and additionally generates usage and help messages for you.

Here's an example of using `optparse` in a simple script:

```
from optparse import OptionParser
[...]  
parser = OptionParser()  
parser.add_option("-f", "--file", dest="filename",  
                  help="write report to FILE", metavar="FILE")  
parser.add_option("-q", "--quiet",  
                  action="store_false", dest="verbose", default=True,  
                  help="don't print status messages to stdout")  
  
(options, args) = parser.parse_args()
```

```
#!/usr/bin/env python
```

```
"""
```

```
Usage: %prog [options] word [word2 ...]
```

```
Echo's one or more words, optionally truncating or changing their case.
```

```
"""
```

```
from optparse import OptionParser
import sys
```

```
##### Options #####
```

```
parser = OptionParser(usage=__doc__.strip())
parser.add_option('--upper', dest='upper', action='store_true')
parser.add_option('--lower', dest='lower', action='store_true')
parser.add_option('--num-chars', dest='num_chars', type='int')
```

your python version, full path may be warranted

```
#!/usr/bin/env python
```

will become sys.argv[0]

```
"""
```

```
Usage: %prog [options] word [word2 ...]
```

```
Echo's one or more words, optionally truncating or changing their case.
```

```
"""
```

```
from optparse import OptionParser
import sys
```

```
##### Options #####
```

```
parser = OptionParser(usage=__doc__.strip())
parser.add_option('--upper', dest='upper', action='store_true')
parser.add_option('--lower', dest='lower', action='store_true')
parser.add_option('--num-chars', dest='num_chars', type='int')
```

takes lots of arguments

```
##### Functions #####
```

```
def truncate(text, num_chars):  
    if len(text) > num_chars:  
        return text[:num_chars]  
    return text
```

```
##### Main block #####
```


```
if __name__ == '__main__':  
    options, args = parser.parse_args()  
    if options.upper and options.lower:  
        print >> sys.stderr, 'only --upper or --lower can be supplied'  
        parser.print_help()  
        sys.exit(1)
```

```
text = ' '.join(args)  
if options.upper:  
    text = text.upper()  
elif options.lower:  
    text = text.lower()
```

```
if options.num_chars:  
    text = truncate(text, options.num_chars)
```

```
print text
```

you can print the help whenever you want
(NB: it will go to stdout)



15.5. optparse — Parser for command line options

New in version 2.3.

Deprecated since version 2.7: The `optparse` module is deprecated and will not be developed further; development will continue with the `argparse` module.

Source code: [Lib/optparse.py](#)

`optparse` is a more convenient, flexible, and powerful library for parsing command-line options than the old `getopt` module. `optparse` uses a more declarative style of command-line parsing: you create an instance of `OptionParser`, populate it with options, and parse the command line. `optparse` allows users to specify options in the conventional GNU/POSIX syntax, and additionally generates usage and help messages for you.

Here's an example of using `optparse` in a simple script:

```
from optparse import OptionParser
[...]  
parser = OptionParser()  
parser.add_option("-f", "--file", dest="filename",  
                  help="write report to FILE", metavar="FILE")  
parser.add_option("-q", "--quiet",  
                  action="store_false", dest="verbose", default=True,  
                  help="don't print status messages to stdout")  
  
(options, args) = parser.parse_args()
```