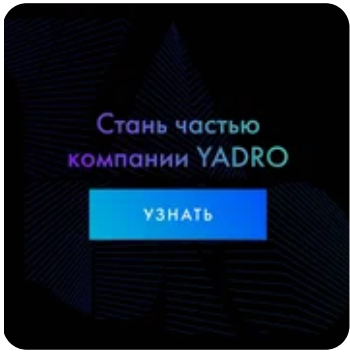


Модуль getopt, извлечение параметров командной строки в Python



oneweekoffer.yadro.com

РЕКЛАМА

Ищем ведущих программистов в команду YADRO.

Ждем амбициозных и талантливых, которые горят инновационными идеями

Заполни анкету

Узнать больше

[Стандартная библиотека Python3.](#) / Модуль getopt, извлечение параметров командной строки в Python

Разбор параметров командной строки в стиле языка C

[Модуль getopt](#) помогает скриптам Python анализировать аргументы командной строки, возвращаемые [функцией sys.argv\(\)](#).

Он поддерживает те же соглашения, что и функция getopt() в Unix, включая специальные значения аргументов в форме '-' и '--'. Длинные параметры, подобные тем, которые поддерживаются программным обеспечением GNU, также могут использоваться через необязательный третий аргумент.

Этот модуль предоставляет две функции и одно исключение.

Примечание. [Модуль getopt](#) - это синтаксический анализатор параметров командной строки, API которого разработан в стиле функции языка C getopt(). Пользователи, которые не знакомы с функцией C getopt() или хотели бы писать меньше кода и получать более качественную подсказку и сообщения об ошибках, должны рассмотреть возможность использования [модуля argparse](#).

Примеры использования модуля getopt:

Пример использования только параметров в стиле Unix:

```
>>> import getopt
>>> args = '-a -b -cfoo -d bar a1 a2'.split()
>>> args
# ['-a', '-b', '-cfoo', '-d', 'bar', 'a1', 'a2']
>>> optlist, args = getopt.getopt(args, 'abc:d:')
>>> optlist
# [('-a', ''), ('-b', ''), ('-c', 'foo'), ('-d', 'bar')]
>>> args
# ['a1', 'a2']
```

Использование длинных имен опций:

```
>>> import getopt
>>> s = '--condition=foo --testing --output-file abc.def -x a1 a2'
>>> args = s.split()
>>> args
# ['--condition=foo', '--testing', '--output-file',
# 'abc.def', '-x', 'a1', 'a2']
>>> optlist, args = getopt.getopt(args, 'x', [
# 'condition=', 'output-file=', 'testing'])
>>> optlist
# [('--condition', 'foo'), ('--testing', ''),
# ('--output-file', 'abc.def'), ('-x', '')]
>>> args
# ['a1', 'a2']
```

В сценарии типичное использование выглядит примерно так:

```
import getopt, sys

def main():
```

```
try:
    opts, args = getopt.getopt(sys.argv[1:], "ho:v", ["help", "output="])
except getopt.GetoptError as err:
    print(err) # выведет что-то вроде "option-a not recognized"
    # показать справку и выйти
    # `usage()` здесь не определена
    # usage()
    sys.exit(2)
output = None
verbose = False
for o, a in opts:
    if o == "-v":
        verbose = True
    elif o in ("-h", "--help"):
        usage()
        sys.exit()
    elif o in ("-o", "--output"):
        output = a
    else:
        assert False, "необработанный вариант"

if __name__ == "__main__":
    main()
```

Обратите внимание, что эквивалентный интерфейс командной строки может быть создан с меньшим количеством кода и более информативными справками и сообщениями об ошибках с помощью [модуля argparse](#):

```
import argparse

if __name__ == '__main__':
    parser = argparse.ArgumentParser()
    parser.add_argument('-o', '--output')
    parser.add_argument('-v', dest='verbose', action='store_true')
    args = parser.parse_args()
    # ... делать что-нибудь с args.output ...
    # ... делать что-нибудь с args.verbose ..
```

Содержание раздела:

- [КРАТКИЙ ОБЗОР МАТЕРИАЛА.](#)
- [Функция getopt\(\) модуля getopt](#)
- [Функция gnu_getopt\(\) модуля getopt](#)
- [Исключение GetoptError модуля getopt](#)

ХОЧУ ПОМОЧЬ
ПРОЕКТУ

Вверх



Вверх