*Новое в версии 2.3*

**Обзор**

optparse это более удобный, гибкий и мощный модуль для парсинга опций командной строки, чем getopt. Он использует больше декларативных стилей парсинга. Вы лишь создаете экземпляр класса OptionParser, наполняете его опциями и получаете результат. Модуль позволяет определять опции в традиционном синтаксисе GNU/POSIX и дополнительно генерировать справочные сообщения.

optparse был явно разработан, чтобы поощрить создание программ с прямыми, стандартными интерфейсами командной строки. С этой целью модуль поддерживает только наиболее распространенный синтаксис командной строки и семантику, традиционно используемую под Unix.

В основном, работа с модулем сводится к созданию экземпляра класса OptionParser, наполнением его правилами работы с опциями, разбор командной строки и паередача основной программе полученный опций и аргументов. В общем случае ключевым моментом для написания кода является понимание возможных вариаций задания опций методом add\_options().

**Примечание:** начиная с версии 2.7 модуль считается устаревшим, в пользу argparse. Однако данный модуль существует и в версии Python3, поэтому, вероятно, проблем при переходе на Python3 не должно возникнуть.

**Создание парсера**

Первым шагом для использования optparse является создание экземпляра OptionParser:

parser = OptionParser(...)

Конструктор класса не требует передачи ему каких либо аргументов, однако, при необходимости, можно передать ему ключевые слова аргументы.

usage (По-умолчанию: "%prog [options]")

Краткая справка об использовании программы, которое выводится если программа была запущена неверно или с опцией справки. Когда optparse выводит строку usage он расширяет %prog значениемos.path.basename(sys.argv[0]) или значением prog переданное конструктору. Чтобы запретить вывод usage используйте значение oprparse.SUPPRESS\_USAGE.

option\_list (По-умолчанию: [])

Список объектов Option для заполнения парсера. Опции в option\_listдобавляются после любых опций в standart\_option\_list (атрибут класса который может быть установлен подклассом OptionParser), но перед опциями версии и справки. Устарело, вместо этого используйте add\_option() после создания парсера.

option\_class (по умолчанию: optparse.Option)

Класс используемый при добавлении опций для парсера в add\_option()

version (По умолчанию: None )

Строка с версией программы, котора выводится когда пользователь запрашивает опцию версии. Если значение было задано, то optparse автоматически добавит опцию "--version". Субстрока "%prog" может быть использованна также как и в usage

conflict\_handler (По умолчанию: "error")

Указывает что делать при задании конфликтных опций.   
Если задано значение "error", то при ввозникновение конфликтов будет возбуждено исключение OptionConflictError и работа программы приостановится, например:

parser=optparse.OptionParser(conflict\_handler="error")  
parser.add\_option('-s', '--silence', action='store\_true', dest='silence', default=False, help='silence mode, no any output')  
parser.add\_option('-s', '--skip',action='store\_true', dest='skip', default=False, help='skip some situation in algoritm of prog')  
parser.parse\_args()  
parser.print\_help()

В результате получим сообщение об исклчюении следующего вида:

Traceback (most recent call last):  
  File "./test.py", line 44, in <module>  
    parser.add\_option('-s', '--skip',action='store\_true', dest='skip', default=False, help='skip some situation in algoritm of prog')  
  File "/usr/lib/python2.7/optparse.py", line 1020, in add\_option  
    self.\_check\_conflict(option)  
  File "/usr/lib/python2.7/optparse.py", line 995, in \_check\_conflict  
    option)  
  optparse.OptionConflictError: option -s/--skip: conflicting option string(s): -s

Если задано значение "resolve", то возникающие конфликты будут мягко разрешены. Преобразуя предыдущий пример следующим образом...

parser=optparse.OptionParser(conflict\_handler="resolve")

Получим:

Usage: test.py [options]  
  
Options:  
-h, --help show this help message and exit  
--silence silence mode, no any output  
-s, --skip skip some situation in algoritm of prog

Обратите внимание каким именно образом optparse разрешил конфликты - при совпадении коротких опций -s предпочтение было отдано опции которая была задана позднее.

description (По умолчанию: None)

Абзац текста с кратким обзором программы. optparse переформатирует абзац под ширину текущего терминала и выведет на экран когда пользователь запросит опцию справки (после usage, но перед списком опций).

formatter (По умолчанию: IndentedHelpFormatter)

Эксземпляр класса optparse.HelpFormatter который будет использоваться для вывода справочного текста. optparse предоставляет 2 класса для использования: IndentedHelpFormatter и TitledHelpFormatter.

add\_help\_option (По умолчанию: True)

Если значение истинно, то optparse добавит в парсер опцию справки (опции-h и --help)

prog

Строка с именем программы, используемая для замены "%prog" в usage иversion вместо получаемой по умолчанию из os.path.basename(sys.argv[0])

**Встроенные типы данных применяемые в опциях**

optparse имеет 6 встроенных типов данных: string, int, long, choice, float, complex. Возможно расширение этого списка за счет определения новых типов. Для конвертирования получаемых значений опций в типы Python используются стандартные конверторы ( int() для 'int' и т.д.).   
  
'choice' это последовательность строк, применяемая в нескольких типах действий(описано разделом ниже).

**Наполнение опциями парсера**

Следующим шагом необходимо наполнить парсер списком необходимых опций. Опции задаются в различных вариациях, например для опции заданной следующим образом (Короткая и длинная формы записи опции, аттрибут указывающий на тип действия , тип возвращаемого аргумента, имя переменной в которую сохранится значение опции):

parser.add\_option("-f", "--file", action="store", type="string", dest="filename")

будут соответствовать следующие вызовы:

-ffoo  
-f foo  
--file=foo  
--file foo

**Определение аттрибутов опций**

Ниже перечисленные аттрибуты опций могут быть заданы в качестве именнованных аргументов для метода add\_option(). Если вы передадите ошибочный или не существующий аттрибут опции, то optparse возбудит исключение OptionValueError или OptionError.

* **action** (По умолчанию: "store")

Определяет действие для заданной в командной строке опции. Подробно расписано ниже.

* **type** (По умолчанию: "string")

Преобразует значение опции в заданный тип. Подробно описано ниже.

* **dest** (По умолчанию получается из строки опции)

optparse сохранит значениее опции как атрибут с именем dest. Если значение не указано, то имя переменной будет образовано из строки опции. Например, если опция была задана как "--input-file", то по умолчанию имя переменной присвоется как input\_file. Если длинная опция не была задана, то будет использована короткая, например, если была задана опция "-f", то переменная примет имя f.

* **default** (Считается устаревшим)

Значении опции по умолчанию, если она не была задана в командной строке. Считается устаревшей и рекомендуется использовать parser.set\_defaults(). Однако это всего лишь рекомендация и ничего не мешает использовать ее в вашем коде.

* **nargs** (По умолчанию: 1)

Количество аргументов принимающее опцией. Если больше 1, то в dest сохранится кортеж значений.

* **const**

Используется совместно с действиями где сохраняются значения как константы.

* **choices**

Для опций с типом действия "choice", предопределенный список строк значения которых могут быть выбраны пользователем.

* **callback**

Для опций с типом действия "callback", вызываемый объект.

* **callback\_args**, **callback\_kwargs**

Дополнительный позиционные и именнованные аргументы передающиеся в вызываемый объект callback после 4 стандартных аргументов.

* **help**

Справочный текст который выводится если пользователь введет опцию "--help". Если текст не задан, то опция будет перечисленна без справочного текста. Что бы скрыть подобную опцию вовсе используйте специальное значение SUPPRESS\_HELP

* **metavar** (По умолчанию получается из строки опции)

Текст приписываемый при выводе справки к опциям который принимают аргументы. Смотри пример ниже для разъяснений.

**Определение типов действий опций**

Перечисленные ниже типы действий определяют основное поведений при интерпретации параметров командной строки. Разные типы действий могут иметь как обязательные парметры так и дополнительные.

* **store** [дополнительные: type, dest, nargs, choices]

За опцией должен следовать аргумент, который будет конвертирован в значение в соответствием с типом type и сохранен в dest. Если nargs больше 1, то в dest сохранится кортеж аргументов.

* **store\_const** [обязательные: const; дополнительные: dest]

Если указана опция, то в dest сохранится значение const.

* **store\_true** [дополнительные: dest]

Специальный случай store\_const, когда в dest сохраняется значение True

* **store\_false** [дополнительные: dest]

Подобно store\_true, только сохраняется значение False

* **append** [дополнительные: type, dest, nargs, choices]

Следующий за опцией аргумент будет добавлен в список dest. Если не задано значение по умолчанию, то будет создан пустой список при первом упоминании опции в командной строке. Если nargs больше 1, то кортеж аргументов будет добавлен в dest.

* **append\_const** [обязательные: const; дополнительные: dest]

Подобно store\_const, но значение const добавляется в список dest.

* **count** [дополнительные: dest]

Увеличить целочисленное значение находящееся в dest. Если не задано значение по умолчанию, то сперва dest приравнивается нулю.

* **callback** [обязательные: callback; дополнительные: types, nargs, callback\_args, callback\_kwargs]

Вызвать функцию указанную в callback следующим образом:

func(option, opt\_str, value, parser, \*args, \*\*kwargs)

* **help**

Вывести окончательно скомпанованное справочное сообщение. Данная опция автоматически включается в парсер, поэтому ее не нужно указывать явно.

* **version**

Выводит информацию о версии и выходит из программы. Версия формируется и выводится на экран методом print\_version(). Если в конструктор парсера была передана строка с версией программы, то данная опци автоматически будет добавлена в парсер.

**Парсинг**

Финальный шаг заключает в непосредственном разборе командной строки и получения опций и аргументов:

(options, arguments) = parser.parse\_args(args=None, values=None)

Входные параметры:   
  
*args*

Список аргументов для обработки (по умолчанию sys.argv[1:])

*values*

Объект для хранения аргументов опций (по умолчанию optparse.Values)

возращаемые значения:   
  
*options*

Экземпляр класса optparse.Values, напоминающий словарь, в котором хранятся полученный опции. Для обращения к аттрибутам экземпляра класса как к словарю необходимо использовать options.\_\_dict\_\_

*arguments*

Список аргументов командной строки, расположенных позиционно левее опций.

**Примеры**

Ниже представлен пример реализующий различные вариации типов действий опций:

# -\*- coding: utf-8 -\*-  
import optparse, os  
  
VERSION='0.1'  
  
def print\_hello(\*args):  
    for i in args:  
        print i  
  
parser=optparse.OptionParser(version='0.1', description='Some test program to show what is optparse module of Python')  
  
parser.add\_option('-f','--file', type='string', dest='file', help='Input file for additional data')  
  
parser.add\_option('-b','--buffer-size', action='store', choices=['512','4k','1m'], default='4k', help='Set buffer size for IO operations')  
  
parser.add\_option('-c','--coefficients', nargs=3, metavar='X Y Z', default=[0, 0, 0], help='User defined coefficients of start')  
  
parser.add\_option('-o', action='store\_const', const=10, dest='output\_level', help='Set output information level to above standart, for see additional information about work of program')  
parser.add\_option('-O', action='store\_const', const=50, dest='output\_level', help='Set output information level to maximum')  
  
parser.add\_option('-s','--silence', action='store\_true', dest='silence', help='No print anything in stdout and stderr')  
  
parser.add\_option('-p','--panic', action='store\_false', dest='standart\_work', help='Abort from program if any working and error occured')  
  
parser.add\_option('-e','--exclude', action='append', type='int', dest='exclude\_columns', help='Exclude for execution column from input file')  
  
parser.add\_option('-n','--nothing', action='append\_const', const='Nothing', help='Nothing to do option')  
  
parser.add\_option('-P','--priority', action='count', dest='user\_priorety', help='Increase priority of execution program on 1 point')  
  
parser.add\_option('--hello', action='callback', dest='help\_string', nargs=1, callback=print\_hello)  
  
parser.add\_option('--hidden\_option', action='store\_true', help=optparse.SUPPRESS\_HELP)  
  
  
options,arguments=parser.parse\_args()  
  
for key,value in options.\_\_dict\_\_.items():  
    print key,'=',value  
for i,item in enumerate(arguments):  
    print 'argument #%s = %s'%(i,item)

Примеры использования:

john\_16@linux-home:~/workspace/tmp/src> python optparse\_test.py -f data.txt -b 512 -O -c 10 0 99 --exclude 0 --exclude 1 -e 3 -P -P --hidden\_option Hello world !  
exclude\_columns = [0, 1, 3]  
user\_priorety = 2  
help\_string = None  
hidden\_option = True  
output\_level = 50  
coefficients = ('10', '0', '99')  
file = data.txt  
nothing = None  
buffer\_size = 512  
standart\_work = None  
silence = None  
argument #0 = Hello  
argument #1 = world  
argument #2 = !  
  
  
john\_16@linux-home:~/workspace/tmp/src> python optparse\_test.py --help  
Usage: optparse\_test.py [options]  
Some test program to show what is optparse module of Python  
Options:

--version show program's version number and exit

-h, --help show this help message and exit

-f FILE, --file=FILE Input file for additional data

-b BUFFER\_SIZE, --buffer-size=BUFFER\_SIZE

Set buffer size for IO operations

-c X Y Z, --coefficients=X Y Z

User defined coefficients of start

-o Set output information level to above standart, for

see additional information about work of program

-O Set output information level to maximum

-s, --silence No print anything in stdout and stderr

-p, --panic Abort from program if any working and error occured

-e EXCLUDE\_COLUMNS, --exclude=EXCLUDE\_COLUMNS

Exclude for execution column from input file

-n, --nothing Nothing to do option

-P, --priority Increase priority of execution program on 1 point

--hello

john\_16@linux-home:~/workspace/tmp/src> python optparse\_test.py --version  
0.1

Автор: Евгений на [19:21](http://john16blog.blogspot.ru/2012/04/python-optparse.html) [[http://img1.blogblog.com/img/icon18_email.gif](http://www.blogger.com/email-post.g?blogID=8211958935650643608&postID=1222104317696499172)](http://www.blogger.com/email-post.g?blogID=8211958935650643608&postID=1222104317696499172)