Сообщить об ошибке.

# Модуль argparse, интерфейс командной строки в Python





## Создание удобных интерфейсов командной строки

Модуль argparse позволяет легко писать удобные интерфейсы командной строки. Программа определяет, какие аргументы ей требуются, а argparse выяснит, как их разобрать из вывода функции sys.argv.

Модуль argparse также автоматически генерирует сообщения справки и использования, а так же выдает ошибки, когда пользователи передают программе недопустимые аргументы.

Обратите внимание, что есть еще два модуля, которые выполняют ту же задачу, а именно это модуль getopt - эквивалент функции getopt() из языка C и устаревший модуль optparse. Обратите также внимание, что модуль argparse основан на optparse и поэтому очень похож с точки зрения использования.

Примечание.

> - *Рекомендуем посмотреть в сторону стороннего модуля click. Это пакет Python для создания красивых интерфейсов командной строки компонуемым способом с минимальным количеством кода, насколько это возможно.*
> - *Также, для **быстрого создания утилит** командной строки из имеющихся функций можно использовать сторонний модуль fire.*

## Создание парсера.

Первым шагом в использовании модуля argparse является получение объекта ArgumentParser путем создания экземпляра класса argparse.ArgumentParser():

```
>>> parser = argparse.ArgumentParser(description='Process some integers.')
```

Объект ArgumentParser будет содержать всю информацию, необходимую для синтаксического анализа командной строки на типы данных Python.

## Добавление аргументов.

Заполнение объекта ArgumentParser информацией об аргументах программы осуществляется путем вызова метода parser.add_argument(). Как правило, эти вызовы говорят ArgumentParser, как взять строки из командной строки и превратить их в объекты. Эта информация хранится и используется при вызове метода parser.parse_args().

Например:

```
>>> parser.add_argument('integers', metavar='N', type=int, nargs='+',
...                     help='an integer for the accumulator')
>>> parser.add_argument('--sum', dest='accumulate', action='store_const',
...                     const=sum, default=max,
...                     help='sum the integers (default: find the max)')
```

Вверх

Позже, вызов метода parser.parse_args() вернет объект с двумя атрибутами: integers и accumulate. Атрибут integers будет представлять собой список из одного или нескольких int, а атрибут accumulate будет либо функцией sum(), если в командной строке указан параметр --sum, либо функцией max(), если это не так.

## Разбор аргументов.

Объект ArgumentParser анализирует аргументы с помощью метода parser.parse_args(). Это позволит проверить командную строку на ошибки, преобразовать каждый аргумент в соответствующий тип и затем вызвать соответствующее действие.

В большинстве случаев это означает, что простой объект пространства имен будет создан из атрибутов, проанализированных из командной строки:

```
>>> parser.parse_args(['--sum', '7', '-1', '42'])
# Namespace(accumulate=<built-in function sum>, integers=[7, -1, 42])
```

В сценарии метод parser.parse_args() обычно вызывается без аргументов, а объект ArgumentParser автоматически определяет аргументы командной строки из функции sys.argv.

## Пример использования модуля argparse:

Следующий код представляет собой программу Python, которая читает список целых чисел из командной строки и производит либо сумму, либо вычисляет максимум:

```
# prog.py
import argparse

parser = argparse.ArgumentParser(description='Process some integers.')
parser.add_argument('integers', metavar='N', type=int, nargs='+',
                    help='an integer for the accumulator')
parser.add_argument('--sum', dest='accumulate', action='store_const',
                    const=sum, default=max,
                    help='sum the integers (default: find the max)')

args = parser.parse_args()
print(args.accumulate(args.integers))
```

Предположим, что приведенный выше код Python сохранен в файл с именем prog.py и может быть запущен в командной строке и предоставляет полезные справочные сообщения:

```
$ python3 prog.py -h
usage: prog.py [-h] [--sum] N [N ...]

Process some integers.

positional arguments:
 N           an integer for the accumulator

optional arguments:
 -h, --help  show this help message and exit
 --sum       sum the integers (default: find the max)
```

При запуске с соответствующими аргументами он выводит либо сумму, либо максимум целых чисел, прочитанных из командной строки:

```
$ python3 prog.py 1 2 3 4
4

$ python prog.py 1 2 3 4 --sum
10
```

Если будут переданы недопустимые аргументы, то программа выдаст ошибку:

```
$ python3 prog.py a b c
usage: prog.py [-h] [--sum] N [N ...]
prog.py: error: argument N: invalid int value: 'a'
```

Вверх

Содержание раздела:





Вверх

Вверх