FIELD NOTE / PYTHON

Parsing Command Line Arguments dalam Python

Petruknisme · 3 Feb, 2017 · 3 min to read

Perkenalan Argparse

Pada bahasan kali ini saya akan membahas tentang bagaimana melakukan parsing Command Line Arguments pada python beserta contoh dan juga penjelasan masing-masing penggunaan. Mungkin kita seringkali menjumpai program python dengan dukungan Command Line Arguments seperti contoh:

```
python Belajar.py -u username -p password
```

Terlihat bahwa argumen -u dan -p dipakai untuk mendapatkan user input berupa username dan password. Nah, pada bahasan ini saya akan menggunakan

argparse yang sudah ada pada python untuk melakukan parsing data dari command line yang diberikan. Kita akan memulai dengan contoh sederhana seperti berikut:

```
import argparse
parser = argparse.ArgumentParser(description='Belajar Command Line Parsing
parser.parse_args()
```

Bila kita simpan dan jalankan program tersebut dengan menggunakan *python Belajar.py* maka tidak akan tampil apa-apa karena kita belum menambahkan argumen yang nantinya akan diproses. Pada kasus ini, kita bisa menjalankannya dengan menggunakan *python Belajar.py –help*. Maka akan muncul tampilan help dari sebuah program seperti berikut:

```
usage: Belajar.py [-h]

Belajar Command Line Parsing

optional arguments:
  -h, --help  show this help message and exit
```

Licensed under CC BY-SA 4.0

Oke, dari tampilan diatas menandakan bahwa program tersebut menampilkan pesan help dari setiap argumen yang sudah tersedia. Pada contoh kali ini hanya argumen *–help/-h* yang tersedia karena kita belum menambahkan argumen lainnya.

Argumen Pada argparse

argparse merupakan sebuah pustaka untuk melakukan argument processing. Setiap argumen dapat melakukan trigger dengan aksi yang berbeda, dengan mendifinisikan *action* argumen pada fungsi **add_argument()**. Beberapa aksi yang tersedia seperti menyimpan sebuah nilai variabel ataupun konstanta ketika dilakukan *action* tersebut, tipe data yang disimpan bisa beragam dari boolean, integer, string dan lainnya.

Secara default action akan melakukan penyimpanan nilai dari sebuah argumen. Ketika tipe data sudah terdefinisikan maka data tersebut akan dikonversikan terlebih dahulu barulah setelah itu disimpan. Ketika parameter *dest* diberikan pada sebuah argumen, maka nilai tersebut akan disimpan kedalam atribut dari nama yang diberikan didalam objek Namespace ketika dilakukan parsing. Karena pada dasarnya bahwa nilai yang di return oleh fungsi *parse_args()* merupakan Namespace yang didalamnya terdapat argumen dari sebuah command. Nah, dari objek tersebut yang menghandle nilai dari sebuah argumen sebagai atribut. Mungkin agak sedikit rumit secara teori, tapi prakteknya mudah kok 😀

Oke, mari kita menuju contoh sederhana.

Contoh Sederhana

Pada contoh sederhana ini, kita akan melakukan pendefinisian sebuah argumen dengan 3 tipe data yang berbeda. Contoh, untuk argumen -x akan bernilai boolean, untuk argumen -y akan bernilai string dan untuk argumen -z akan bernilai integer. Berikut contoh code:

```
1  import argparse
2  parser = argparse.ArgumentParser(description='Belajar Command Line Parsing
3
4  parser.add_argument('-x', action="store_true", default=False)
5  parser.add_argument('-y', action="store", dest="y")
6  parser.add_argument('-z', action="store", dest="z", type=int)
7
8  print(parser.parse_args(['-x', '-yinistr', '-z', '100'])) #Memberikan argu
```

Dan bila dijalankan akan menghasilkan output sebagai berikut:

```
1  $ python Belajar.py
```

Licensed under CC BY-SA 4.0

```
Namespace(x=True, y='inistr', z=100)
```

Penjelasan:

- Argumen -x akan menyimpan nilai True berupa tipe data boolean dengan nilai default adalah False
- Argumen -y akan menyimpan nilai kedalam variabel y
- Argumen -z akan menyimpan nilai kedalam variabel z. Tapi, sebelum disimpan data tersebut dikonversikan terlebih dahulu kedalam integer.

Perlu diingat bahwa – digunakan untuk argumen 1 karakter. Bila karakter lebih dari 1 maka gunakan —

Argumen untuk Action

– store

Action ini digunakan untuk menyimpan sebuah nilai, setelah proses konversi tipe data terlebih dahulu. Action ini secara default akan digunakan ketika tidak ada action yang didefinisikan.

– store_const

Action ini digunakan untuk menyimpan sebuah nilai yang sudah didefinisikan sebagai konstanta.

– store_true / store_false

Action ini digunakan untuk menyimpan sebuah nilai berupa boolean true/false sesuai dengan yang kita inginkan.

– append

Action ini digunakan untuk menyimpan nilai kedalam sebuah list. Nilai ganda akan tetap tersimpan ketika argumen dijalankan berulang

– version

Action ini digunakan untuk menampilkan detail versi dari sebuah program

Sebagai contoh seperti berikut:

```
import argparse
parser = argparse.ArgumentParser(description='Belajar Command Line Parsing

parser.add_argument('-s', action="store", dest="nilai_biasa", help='Simpan
parser.add_argument('-c', action="store_const", dest="nilai_const", const=
parser.add_argument('-t', action="store_true", default=False)
```

Licensed under CC BY-SA 4.0

```
7   parser.add_argument('-a', action="append", dest='daftar', default=[], help
8   parser.add_argument('--version', action='version', version='Belajar 1.0')
9
10  print(parser.parse_args(['-snilai', '-c', '-t' ,'-a satu', '-a dua', '-a t
```

Parser diatas sama saja dengan melakukan perintah *python Belajar.py -s nilai -c -t -a satu -a dua -a tiga*. Maka akan tampil sebagai berikut:

```
1   Namespace(daftar=[' satu', ' dua', ' tiga'], nilai_biasa='nilai', nilai_co
```

Mungkin itu saja bahasan tentang Parsing Command Line Arguments pada Python. Saya lampirkan referensi untuk mempelajari argparse lebih lanjut. Semoga artikel ini bermanfaat!

Referensi:

- https://pymotw.com/2/argparse/ (https://pymotw.com/2/argparse/)

Python # Tutorial

Licensed under CC BY-SA 4.0