Python argparse

- https://ellibrodepython.com/python-argparse

argparse

- o = argparse.ArgumentParser:
  - description: se muestra en la ayuda del programa
  - prefix_chars: cambia el prefijo de los argumentos
  - fromfile_prefix_chars: establece el prefijo para importar un archivo que contiene los argumentos
  - version: numero de versión
- g = o.add_mutually_exclusive_group(): argumentos exluyentes entre si
  - g.add_argument
- o.add_argument
  - -: argumento corto
  - –: argumento largo
  - type: tipo int, str…
  - help: texto de ayuda
  - choices: lista de opciones válidas
  - default: valor por defecto si no se pasa el parámetro
  - required: True|False
  - dest: cambia el nombre de la variable donde se almacena el valor
  - nargs
    - n: número de argumentos
    - ?: argumento opcional
    - +: al menos 1 argumento
    - `argparse.REMAINDER`: recoge el resto de valores, usar como última opción (de todos los `add_argument`)
  - action
    - store: Es el comportamiento por defecto que hemos visto anteriormente. Simplemente almacena el valor que se pasa con el argumento en una variable.
    - store_const: Almacena una constante en la variable, cuyo valor debemos especificar en const.
    - store_true: Almacena el booleano True en la variable. Muy útil para definir flags que no reciben un valor concreto.
    - store_false: Análogo al anterior pero almacena False.
    - append: Añade el argumento a una lista. Útil cuando un argumento es pasado múltiples veces.
    - append_const: Similar a append pero almacena en la lista la constante especificada en const.
    - count: Cuenta el número de veces que un determinado argumento es pasado.
    - help: Muestra la ayuda del programa y finaliza sin hacer nada más.
    - version: Muestra la versión del programa y finaliza la ejecución.
- se pueden crear acciones personalizadas con `argparse.Action`

ejemplos

```
# calculadora.py
import argparse

parser = argparse.ArgumentParser(description='Calculadora, suma/resta/multiplica a
```

```
y b')
parser.add_argument('-a', '--numero_a', type=int, help='Parámetro a')
parser.add_argument('-b', '--numero_b', type=int, help='Parámetro b')
parser.add_argument('-o', '--operacion',
                    type=str,
                    choices=['suma', 'resta', 'multiplicacion'],
                    default='suma', required=False,
                    help='Operación a realizar con a y b')

args = parser.parse_args()
```

```
# cambio_prefijo.py
import argparse

parser = argparse.ArgumentParser(prefix_chars='+')
parser.add_argument('+a')
parser.add_argument('+b')

args = parser.parse_args()
print(vars(args))
```

argumentos_fichero.py

```
# argumentos_fichero.py
import argparse

parser = argparse.ArgumentParser(fromfile_prefix_chars='@')

parser.add_argument('-a')
parser.add_argument('-b')
parser.add_argument('-c')
parser.add_argument('-d')
parser.add_argument('-e')
parser.add_argument('-f')

args = parser.parse_args()
print(vars(args))

# argumentos_fichero.py @args.txt
```

excluyentes.py

```
# excluyentes.py
import argparse

parser = argparse.ArgumentParser()
grupo = parser.add_mutually_exclusive_group()

grupo.add_argument('-f', '--foo')
grupo.add_argument('-b', '--bar')

args = parser.parse_args()
print(vars(args))
```

argumentos_variables1.py

```
# help_version.py
import argparse
parser = argparse.ArgumentParser()
parser.version = '1.0'
parser.add_argument('--ayuda', action='help')
parser.add_argument('--version', action='version')
args = parser.parse_args()
print(vars(args))
```

argumentos_variables1.py

```
# argumentos_variables1.py
import argparse
parser = argparse.ArgumentParser()
parser.add_argument('-a', nargs=5)
args = parser.parse_args()
print(vars(args))
```

argumentos_variables2.py

```
# argumentos_variables2.py
import argparse
parser = argparse.ArgumentParser()
parser.add_argument('-a', nargs='?')
args = parser.parse_args()
print(vars(args))
```

argumentos_variables3.py

```
# argumentos_variables3.py
import argparse
parser = argparse.ArgumentParser()
parser.add_argument('-a', nargs='?')
args = parser.parse_args()
print(vars(args))
```

argumentos_variables4.py

```
# argumentos_variables4.py
import argparse
parser = argparse.ArgumentParser()
parser.add_argument('-a', nargs='+')
args = parser.parse_args()
print(vars(args))
```

argumentos_variables5.py

```
# argumentos_variables5.py
import argparse
```

```
parser = argparse.ArgumentParser()
parser.add_argument('-a')
parser.add_argument('-b', nargs=argparse.REMAINDER)
args = parser.parse_args()
print(vars(args))
```

ejemplos

/via: https://docs.python.org/2/library/argparse.html#module-argparse

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-

import argparse

parser = argparse.ArgumentParser()
parser.add_argument("-v", "--verbose", help="Mostrar información de depuración",
action="store_true")
args = parser.parse_args()
```

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-

import argparse

parser = argparse.ArgumentParser()
parser.add_argument("-v", "--verbose", help="Mostrar información de depuración",
action="store_true")
parser.add_argument("-f", "--file", help="Nombre de archivo a procesar")
args = parser.parse_args()

# Aquí procesamos lo que se tiene que hacer con cada argumento
if args.verbose:
    print "depuración activada!!!"

if args.file:
    print "El nombre de archivo a procesar es: ", args.file
```

From:
https://matebcn.eu/ - **miguel angel torres egea**

Permanent link:
https://matebcn.eu/development:python:argparse

Last update: **26/09/2025 10:05**