[Argparse Tutorial](http://www.pythonforbeginners.com/modules-in-python/argparse-tutorial/)***2***

12 Jun 2013   | [Modules](http://www.pythonforbeginners.com/category/modules-in-python/)

Tags: [argparse](http://www.pythonforbeginners.com/tag/argparse/)

What is it?

Parser for command-line options, arguments and subcommands

Why use it?

The argparse module makes it easy to write user-friendly command-line interfaces.

How does it do that?

The program defines what arguments it requires, and argparse will figure out how

to parse those out of sys.argv.

The argparse module also automatically generates help and usage messages and

issues errors when users give the program invalid arguments.

Concept

When you run the "ls" command without any options, it will default displaying the

contents of the current directory

If you run "ls" on a different directory that you currently are in, you would type

"ls directory\_name". The "directory\_name" is a "positional argument", which means

that the program know what to do with the value.

To get more information about a file we can use the "-l" switch.

The "-l" is knowns as an "optional argument"

If you want to display the help text of the ls command, you would type "ls --help"

Argparse

To start using the argparse module, we first have to import it.

|  |  |
| --- | --- |
| 1  2  3 | import argparse  parser = argparse.ArgumentParser()  parser.parse\_args() |

Run the code

Run the code with the --help option (Running the script without any options

results in nothing displayed to stdout)

|  |  |
| --- | --- |
| 1  2  3  4  5 | python program.py --help (or python program.py -h)  usage: program.py [-h]    optional arguments:    -h, --help  show this help message and exit |

As seen above, even though we didnt specify any help arguments in our

script, its still giving us a nice help message.

This is the only option we get for free.

Positional arguments

In our "ls" example above, we made use of the positional arguments

"ls directory\_name".

Whenever we want to specify which command-line options the program will accept,

we use the "add\_argument()" method.

|  |  |
| --- | --- |
| 1  2  3 | parser.add\_argument("echo")     # naming it "echo"  args = parser.parse\_args()  # returns data from the options specified (echo)  print(args.echo) |

If we now run the code, we can see it requires us to specify an option

|  |  |
| --- | --- |
| 1  2  3  4 | $ python program.py    usage: program.py [-h] echo  program.py: error: too few arguments |

When we specify the echo option it will display "echo"

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12 | $ python program.py echo  echo    #Using the --help option  $ python program.py --help  usage: program.py [-h] echo    positional arguments:    echo    optional arguments:    -h, --help  show this help message and exit |

Extending the help text

To get more help about our positional argument (echo), we have to change our script.

|  |  |
| --- | --- |
| 1  2  3  4  5 | import argparse  parser = argparse.ArgumentParser()  parser.add\_argument("echo", help="echo the string you use here")  args = parser.parse\_args()  print(args.echo) |

Result in this:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8 | $ python program.py --help  usage: program.py [-h] echo    positional arguments:    echo        echo the string you use here    optional arguments:    -h, --help  show this help message and exit |

Note:

Argparse treats the options we give as a string, but we can change that.

Running the code with the type set to Integer

This code will treat the input as an integer.

|  |  |
| --- | --- |
| 1  2  3  4  5  6 | import argparse  parser = argparse.ArgumentParser()  parser.add\_argument("square", help="display a square of a given number",                      type=int)  args = parser.parse\_args()  print(args.square\*\*2) |

If we run the program with the --help option, we can see:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8 | $ python program.py -h  usage: program.py [-h] square    positional arguments:    square      display a square of a given number    optional arguments:    -h, --help  show this help message and exit |

Run the program

From the help text, we can see that if we give the program a number, it will

give us the square back.

Cool, lets try it out:

|  |  |
| --- | --- |
| 1  2  3  4  5 | $ python program.py 4  16    $ python program.py 10  100 |

If we would use a string instead of a number, the program will return an error

|  |  |
| --- | --- |
| 1  2  3  4  5 | $ python program.py four    usage: program.py [-h] square    program.py: error: argument square: invalid int value: 'four' |

Optional arguments

In our "ls" example above, we made use of the optional argument "-l" to get more

information about a file.

The program below will display something when --verbosity is specified and display

nothing when not.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7 | import argparse  parser = argparse.ArgumentParser()  parser.add\_argument("--verbose", help="increase output verbosity",                      action="store\_true")  args = parser.parse\_args()  if args.verbose:      print("verbosity turned on") |

An optional argument (or option) is (by default) given None as a value when its

not being used.

Using the --verbosity option, only two values are actually useful, True or False.

The keyword "action" is being given the value "store\_true" which means that if

the option is specifed, then assign the value "True" to args.verbose

Not specifying the option implies False.

If we run the program with the --help option, we can see:

|  |  |
| --- | --- |
| 1  2  3  4  5  6 | $ python program.py -h  usage: program.py [-h] [--verbose]    optional arguments:    -h, --help  show this help message and exit    --verbose   increase output verbosity |

Run the program using the --verbose option

|  |  |
| --- | --- |
| 1  2 | $ python program.py --verbose  verbosity turned on |

Short options

Using short versions of the options is as easy as:

|  |  |
| --- | --- |
| 1  2 | parser.add\_argument("-v", "--verbose", help="increase output verbosity",                      action="store\_true") |

The help text will updated with the short version.

Source

<http://docs.python.org/dev/library/argparse.html>