LECTURE 42

ARGPARSE

MCS 275 Spring 2022
Emily Dumas

LECTURE 42: ARGPARSE

Course bulletins:

- Project 4 is due 6pm Friday.
- The project 4 autograder is now open.
- Complete your course evaluations!

END STUFF

Blackboard site closes at the end of May.

Homework, worksheets, solutions, etc. will be archived on this public site:

https://www.dumas.io/teaching/2022/spring/mcs275/

Lecture videos will not be there, but I will email you the links to those before Blackboard closes.

Each lecture video is removed 180 days after recording.

COMMAND LINE INTERFACE

In most settings where programs are developed, basic familiarity and comfort with working in a shell/terminal is important.

This is especially true in Unix/Linux, and a lot of computing involves Unix/Linux in some way.

Today we'll focus on Python scripts that are meant to be run and used entirely in a shell, i.e. that use a command line interface or CLI.

EXECUTABLE PYTHON SCRIPTS

```
#!/usr/bin/python3
# This example works on most Linux
"""Show Python version and exit"""
import sys
print(sys.version)
```

and then marking the file as executable, using shell command

```
chmod +x myscript.py
```

EXECUTABLE PYTHON SCRIPTS

In Unix/Linux you can make a Python script file directly executable by adding an interpreter specification line (starting `#!`) at the beginning of the file

```
#!/usr/bin/env python3
# This example works on MacOS and most Linux
"""Show Python version and exit"""
import sys
print(sys.version)
```

and then marking the file as executable, using shell command

```
chmod +x myscript.py
```

OPTIONS AND ARGUMENTS

CLI programs often want to accept:

- Required positional **arguments** (e.g. input filename, directory to search, ...)
- **Options** (e.g. iterate 5 times, write to "out.txt" instead of terminal, use alternate scrape URL, ...)
- **Flags** (e.g. enable verbose output, allow overwriting an existing file, ...)

OPTIONS

A configurable aspect of the program's operation that can be set or changed by adding command line argument(s).

E.g. A scraper might default to waiting 30 seconds between requests, but allow you to change this on the command line. Some popular syntaxes:

```
scrape --delay 5  # my favorite; human readable!
scrape -d5        # terse but ok
scrape -d 5       # also used
scrape --delay=5  # also used
scrape -delay 5   # less common
scrape /d 5       # rare except in Windows
scrape /delay 5   # rare except in Windows
```

Linux/MacOS examples:

```
# positional argument
cat mcs275/slides/lecture42.html
ls mcs275/public/samplecode
cp lecture_template.html lecture43.html
# flags: turn feature on or off
ls -l
ls --human-readable
# options
find . -name '*.html' # recursive search for HTML files
```

USAGE AND HELP

If invalid or insufficient arguments are given, a good CLI program will display a short **usage message** (explaining how to use it).

It is best to also offer a help flag (e.g. `--help` or `-h`) that prints a more detailed usage message and list of options.

ARGPARSE

Parsing and extracting options, arguments, and flags from `sys.argv` is difficult to do well.

But in Python you can (and should) usually avoid writing command line parsers from scratch.

The standard library module `argparse` is flexible and easy to use.

KEY FEATURES

- Argument and option type checking
- Automatic help and usage messages
- Automatic error messages
- Allows an option to have both short and long names (e.g. `-h` and `--help`)
- Supports many common ways of writing options

Minimal argparse example from the module docs:

```python
import argparse

parser = argparse.ArgumentParser()
parser.add_argument(
    "square",
    help="display a square of a given number",
    type=int  # if not specified, default type is string
)
args = parser.parse_args()  # parse or show error and exit

print(args.square**2) # arguments and options are attributes o
                      # the `args` object returned above
```

REFERENCES

- argparse module documentation
- Section 13.3 of Beazley and Jones (Python Cookbook) discusses argparse and gives some examples.

REVISION HISTORY

- 2022-04-25 Initial publication