BUILDING CLIS THAT CLICK

Created by Jason A Myers / @jasonamyers

BUILDING GOOD COMMAND LINE APPLICATIONS IS HARD

IMPORTANT PARTS

NAME

ARGUMENT PARSING AND VALIDATION *

HELP GENERATION *

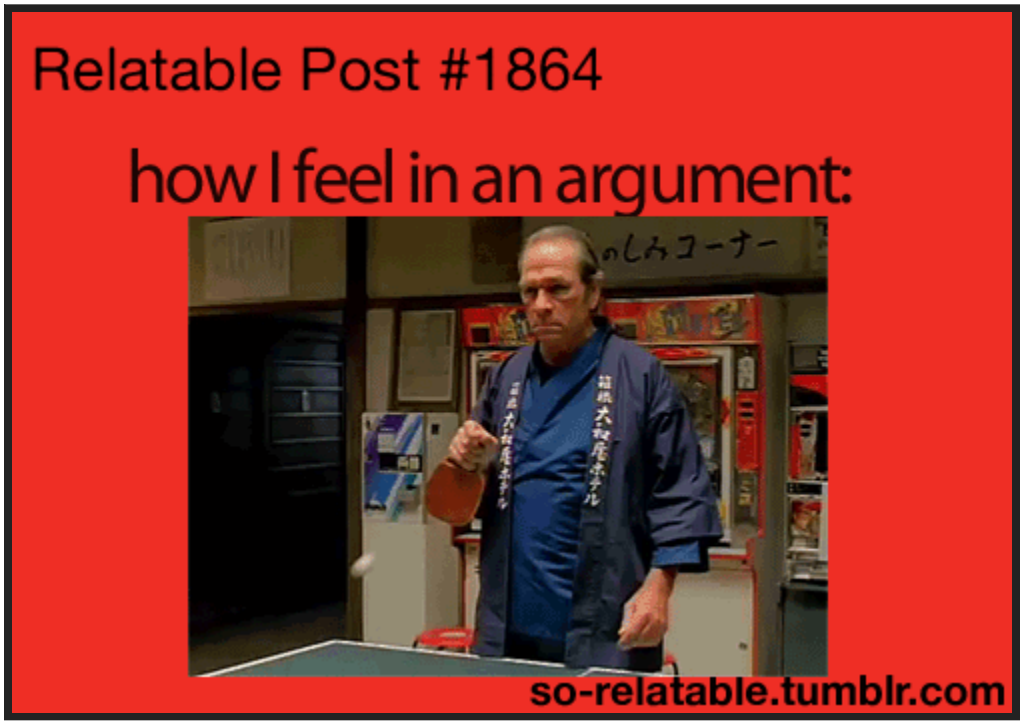
COMMAND STRUCTURE *

AUTOCOMPLETION

NICE OUTPUT

PACKAGING *

ARGUMENTS AND HELP



THREE DIFFERENT PARSERS IN THE STDLIB

- getopt
- optparse
- argparse

I MEAN ARGPARSE IS THE NEW HOTNESS???

SERIOUSLY WHO KNOWS HOW *!@%PARSE WORKS ANYWAY

NO REALLY HAVE YOU LOOKED AT THE DOCS...

15.4.2. ArgumentParser objects

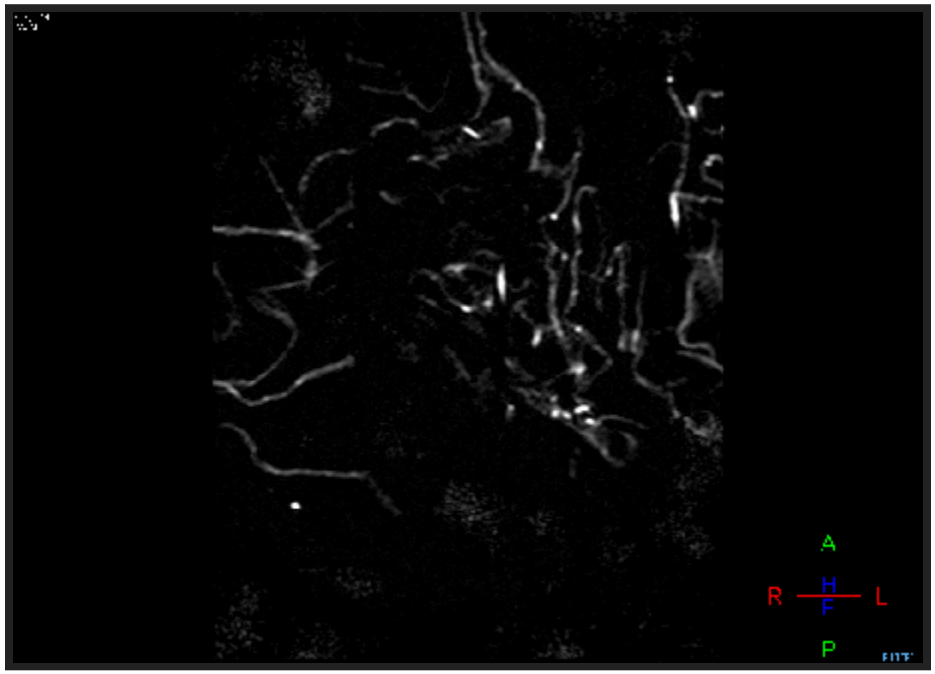
class `argparse.`**`ArgumentParser`**(*prog=None*, *usage=None*, *description=None*, *epilog=None*, *parents=[]*, *formatter_class=argparse.HelpFormatter*, *prefix_chars='-'*, *fromfile_prefix_chars=None*, *argument_default=None*, *conflict_handler='error'*, *add_help=True*)

Create a new **`ArgumentParser`** object. All parameters should be passed as keyword arguments. Each parameter has its own more detailed description below, but in short they are:

- prog - The name of the program (default: `sys.argv[0]`)
- usage - The string describing the program usage (default: generated from arguments added to parser)
- description - Text to display before the argument help (default: none)
- epilog - Text to display after the argument help (default: none)
- parents - A list of **`ArgumentParser`** objects whose arguments should also be included
- formatter_class - A class for customizing the help output
- prefix_chars - The set of characters that prefix optional arguments (default: '-')
- fromfile_prefix_chars - The set of characters that prefix files from which additional arguments should be read (default: `None`)
- argument_default - The global default value for arguments (default: `None`)
- conflict_handler - The strategy for resolving conflicting optionals (usually unnecessary)
- add_help - Add a -h/–help option to the parser (default: `True`)

The following sections describe how each of these are used.

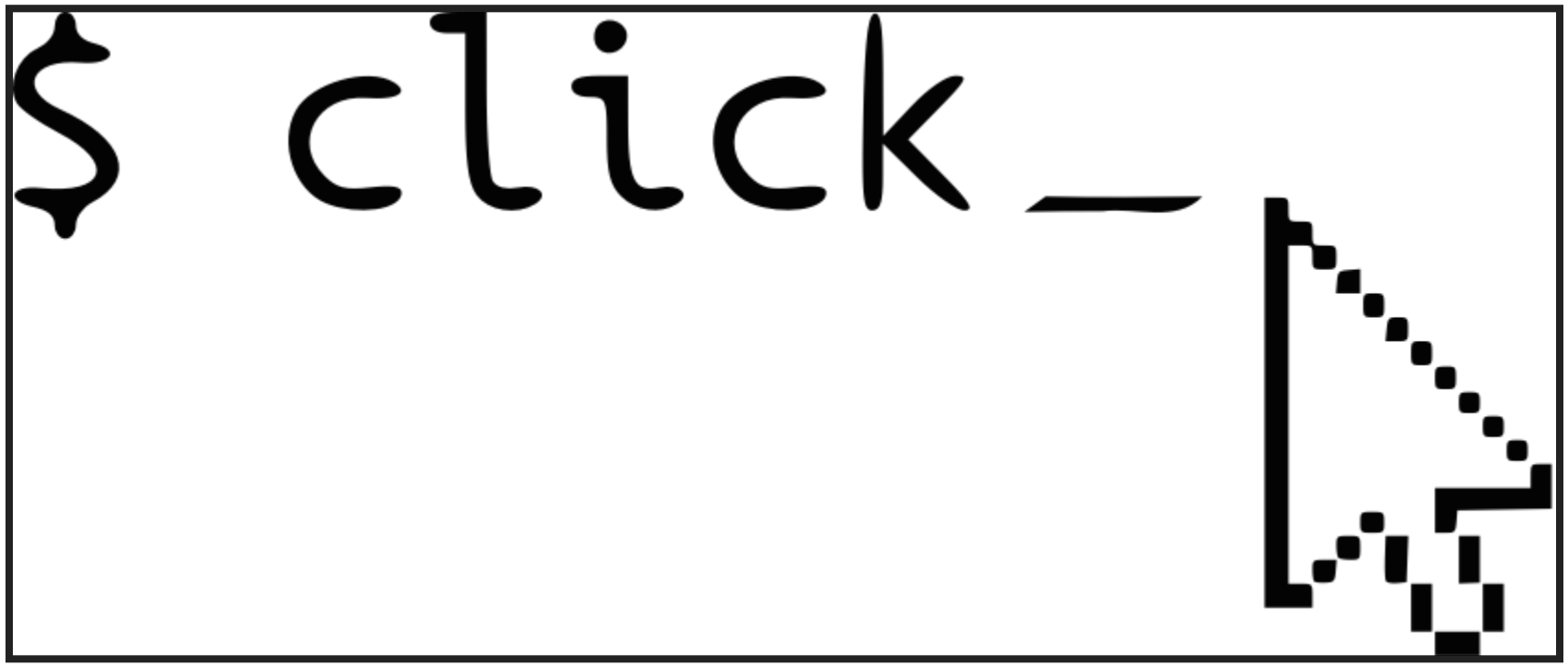
SERIOUSLY BRAIN CELLS EXPLODE



HOW BAD IS IT?

- docopt
- Plac
- Cliff
- Clint

```
import sys
if __name__ == "__main__":
    main(sys.argv)
```


$ click_

DEMO

LOGGING

GOOD LOGGING MESSAGES

- Time
- Module
- Level
- Parseable Messages

```
2015-05-28 09:25:18,711 - complex.logger - DEBUG - Creating composit
e: cookies
2015-05-28 09:25:18,711 - complex.logger - DEBUG - Created composite
: cookies
```

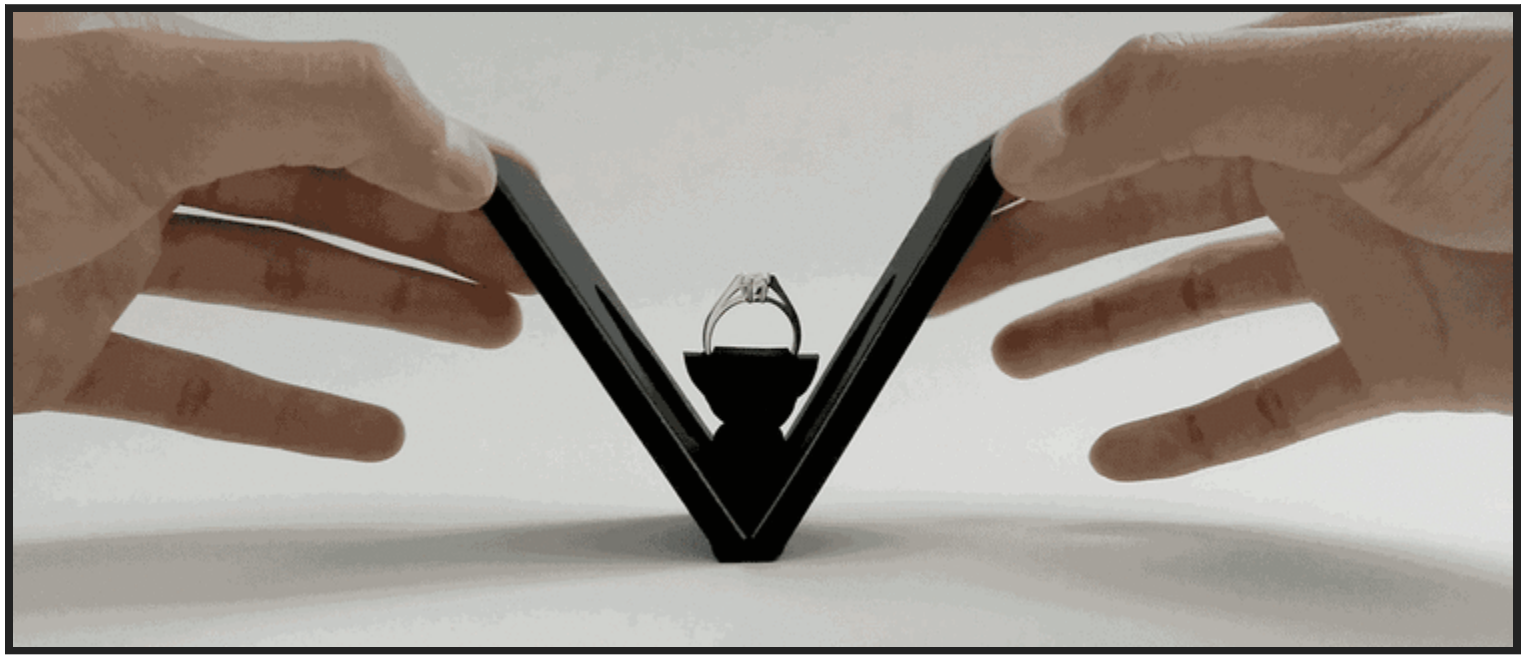
JAM'S LOGGING STYLE

```
import logging

logger = logging.getLogger(__name__)
logger.setLevel(logging.ERROR)
```

```
file_log_handler = logging.FileHandler('complex-cli.log')
logger.addHandler(file_log_handler)

stderr_log_handler = logging.StreamHandler()
logger.addHandler(stderr_log_handler)
```

```
format_string = '%(asctime)s - %(name)s - ' \
                '%(levelname)s - %(message)s'
formatter = logging.Formatter(format_string)
file_log_handler.setFormatter(formatter)
stderr_log_handler.setFormatter(formatter)
```

PACKAGING



FIND OUR MODULE

```
from setuptools import setup, find_packages

setup(
    name='complex',
    version='0.1.2',
    packages=find_packages(),
    include_package_data=True,
```

```
install_requires=[
        'Click==3.3',
    ],
```

```
    description='A description',
    classifiers=[
        'License :: OSI Approved :: BSD License',
        'Programming Language :: Python',
        'Programming Language :: Python :: 3',
    ],
```

```
    entry_points='''
    [console_scripts]
    complex=complex.command:cli
    '''
)
```

COMPLEX DEMO

QUESTIONS

@JASONAMYERS