Best Practices for Command Line Apps

Zack Lalanne

PyTN 2020

About Me

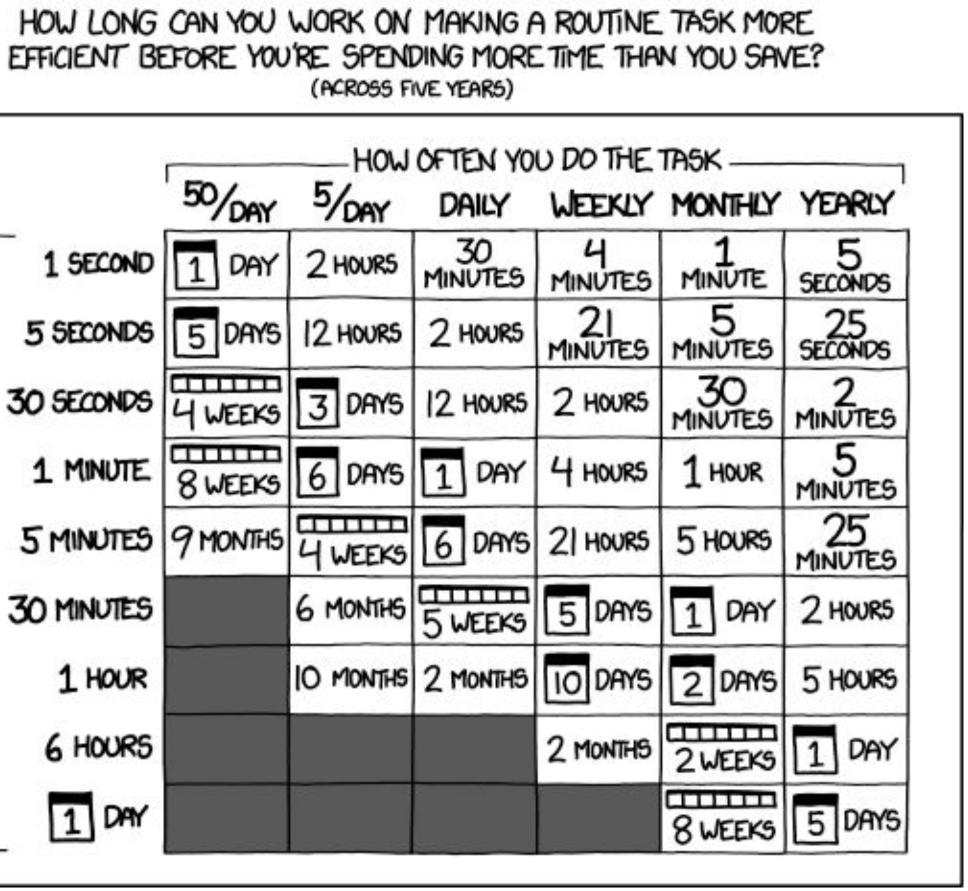
- Developing with Python at Texas Instruments the last ~6 years
- Mainly a C codebase (devices with 4k memory!)
- Manage internal tooling project to improve developer efficiency
    - Documentation
    - Static Analysis
    - Interacting with Services
    - Building / Test Software
    - Code size benchmarking

Outline

- Why developers need tools?
- Parsing Command Line Flags
- Treating Your Tool like an API
- Making Tools Easy to Use
- Odds and Ends

Why Do Developers Need Tools?

- Improve efficiency
- Improve quality
- For automation



HOW LONG CAN YOU WORK ON MAKING A ROUTINE TASK MORE EFFICIENT BEFORE YOU'RE SPENDING MORE TIME THAN YOU SAVE?
(ACROSS FIVE YEARS)

HOW MUCH TIME YOU SHAVE OFF	HOW OFTEN YOU DO THE TASK					
	50/DAY	5/DAY	DAILY	WEEKLY	MONTHLY	YEARLY
1 SECOND	1 DAY	2 HOURS	30 MINUTES	4 MINUTES	1 MINUTE	5 SECONDS
5 SECONDS	5 DAYS	12 HOURS	2 HOURS	21 MINUTES	5 MINUTES	25 SECONDS
30 SECONDS	4 WEEKS	3 DAYS	12 HOURS	2 HOURS	30 MINUTES	2 MINUTES
1 MINUTE	8 WEEKS	6 DAYS	1 DAY	4 HOURS	1 HOUR	5 MINUTES
5 MINUTES	9 MONTHS	4 WEEKS	6 DAYS	21 HOURS	5 HOURS	25 MINUTES
30 MINUTES	▨	6 MONTHS	5 WEEKS	5 DAYS	1 DAY	2 HOURS
1 HOUR	▨	10 MONTHS	2 MONTHS	10 DAYS	2 DAYS	5 HOURS
6 HOURS	▨	▨	▨	2 MONTHS	2 WEEKS	1 DAY
1 DAY	▨	▨	▨	▨	8 WEEKS	5 DAYS

Let’s Go on a Journey...

- Converts Markdown to HTML
- Company branding
- Follow our best practices

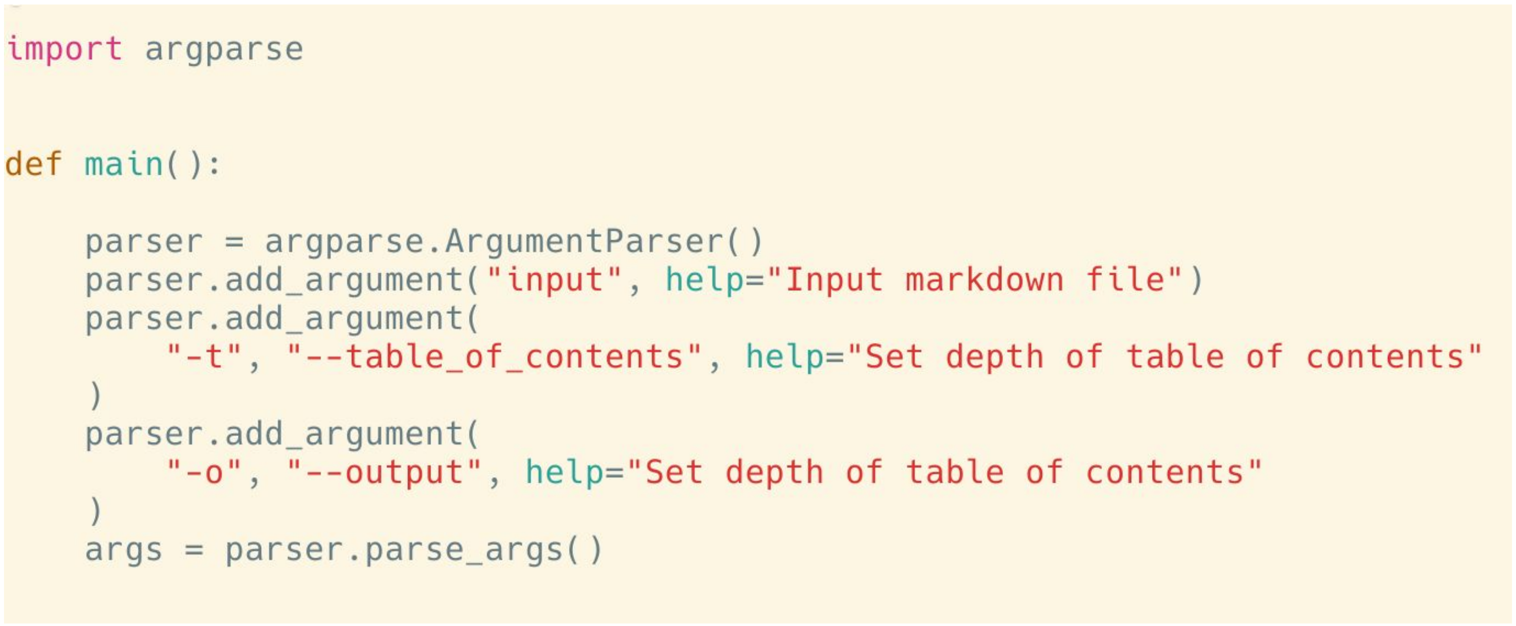
```
~> mdhtml --table_of_contents=3 --output=README.html README.md
```

Command Line Flags

- argparse
- click
- docopt

argparse

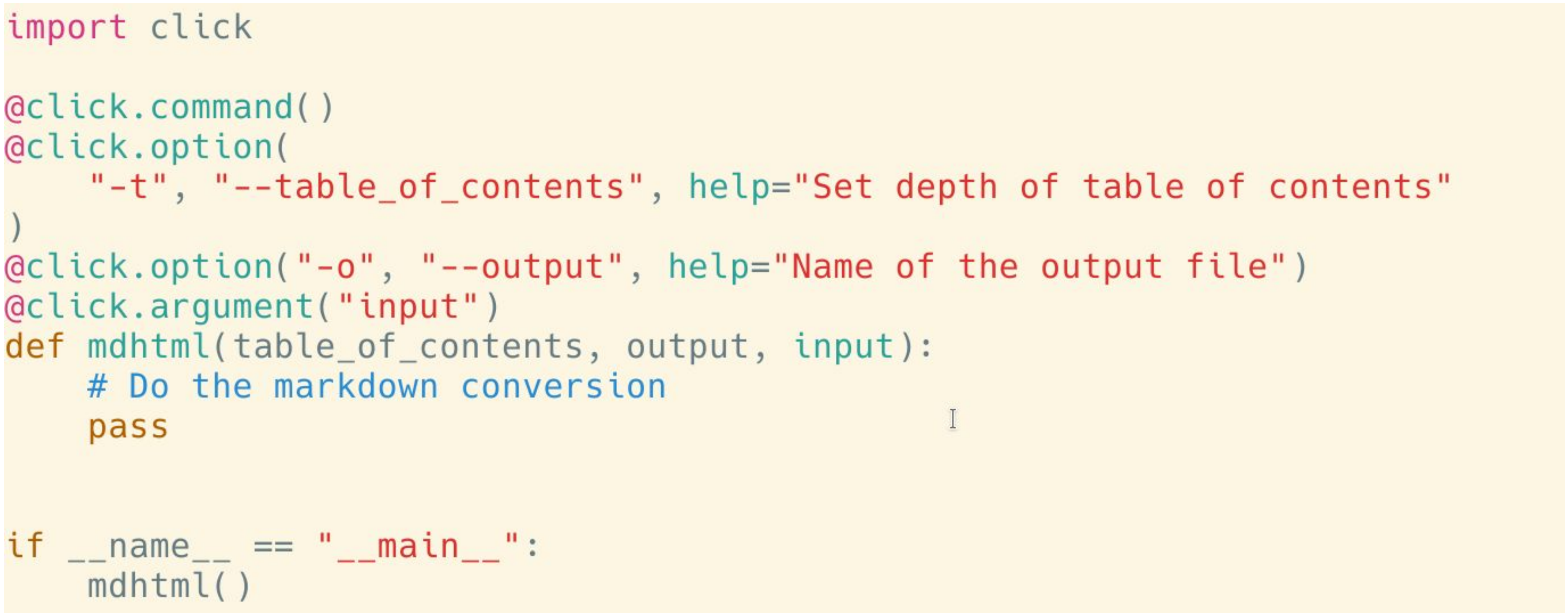
- Great place to start
- Easy to self contain
- Lots of features



```python
import argparse


def main():

    parser = argparse.ArgumentParser()
    parser.add_argument("input", help="Input markdown file")
    parser.add_argument(
        "-t", "--table_of_contents", help="Set depth of table of contents"
    )
    parser.add_argument(
        "-o", "--output", help="Set depth of table of contents"
    )
    args = parser.parse_args()
```

click

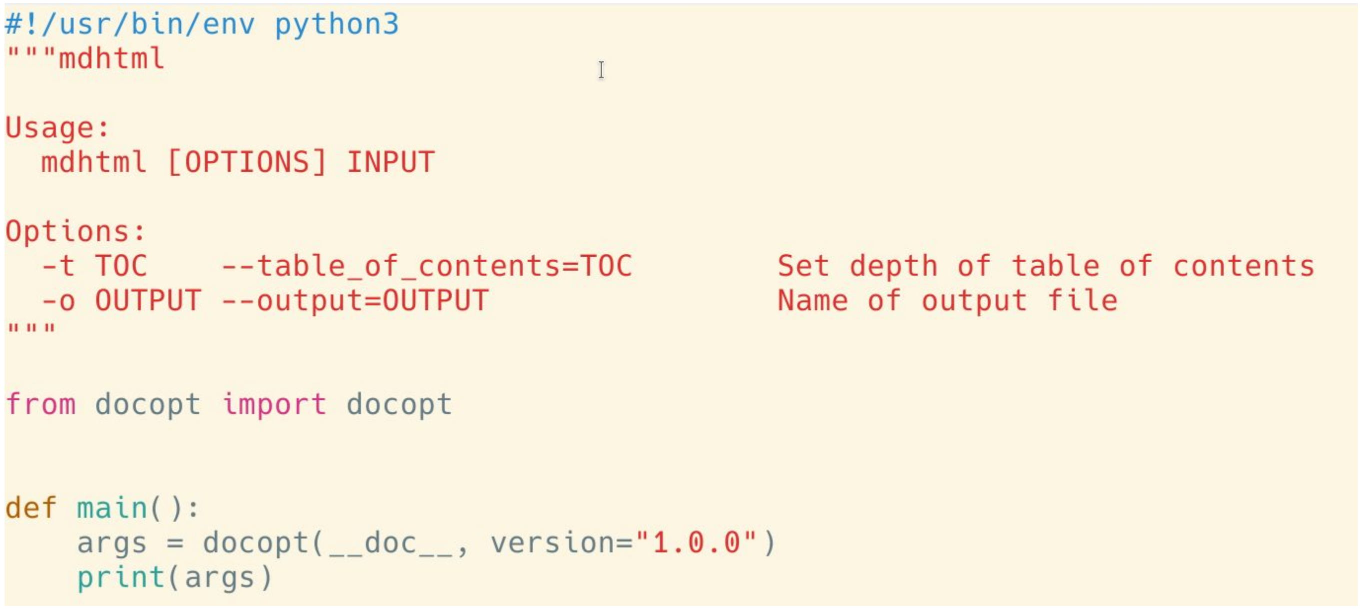
- Lots of community support
- Uses decorators, clean code
- Progress bars / colors
- Great test support
- Bash completion



```python
import click

@click.command()
@click.option(
    "-t", "--table_of_contents", help="Set depth of table of contents"
)
@click.option("-o", "--output", help="Name of the output file")
@click.argument("input")
def mdhtml(table_of_contents, output, input):
    # Do the markdown conversion
    pass


if __name__ == "__main__":
    mdhtml()
```

docopt

- Really fast to get started
- Treats documentation as first class
- Implementations in other languages (Node.js, bash)



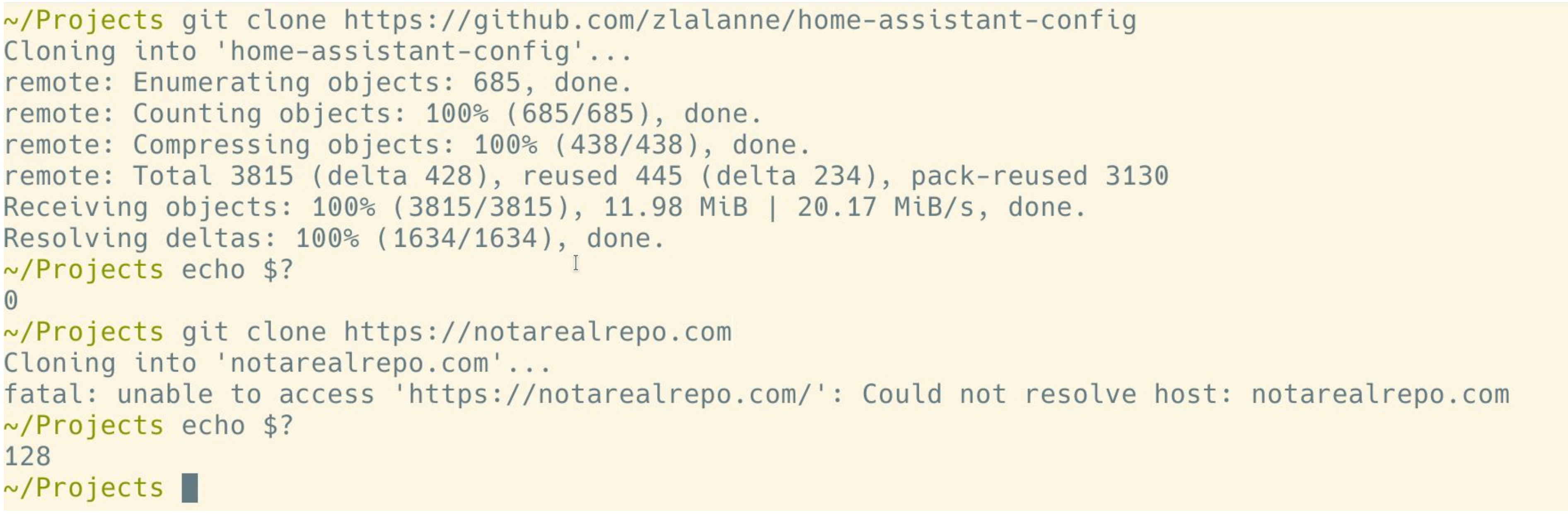
```
#!/usr/bin/env python3
"""mdhtml

Usage:
  mdhtml [OPTIONS] INPUT

Options:
  -t TOC    --table_of_contents=TOC       Set depth of table of contents
  -o OUTPUT --output=OUTPUT               Name of output file
"""

from docopt import docopt


def main():
    args = docopt(__doc__, version="1.0.0")
    print(args)
```

Tool as an API

- We hate when APIs break, don’t break tools!
- Required / optional arguments
- Exit Codes
- Experimental Options
- Be consistent

Tool as an API - Exit Codes

- Good tools make use of error codes



```
~/Projects git clone https://github.com/zlalanne/home-assistant-config
Cloning into 'home-assistant-config'...
remote: Enumerating objects: 685, done.
remote: Counting objects: 100% (685/685), done.
remote: Compressing objects: 100% (438/438), done.
remote: Total 3815 (delta 428), reused 445 (delta 234), pack-reused 3130
Receiving objects: 100% (3815/3815), 11.98 MiB | 20.17 MiB/s, done.
Resolving deltas: 100% (1634/1634), done.
~/Projects echo $?
0
~/Projects git clone https://notarealrepo.com
Cloning into 'notarealrepo.com'...
fatal: unable to access 'https://notarealrepo.com/': Could not resolve host: notarealrepo.com
~/Projects echo $?
128
~/Projects
```

Tool as an API - Exit Codes

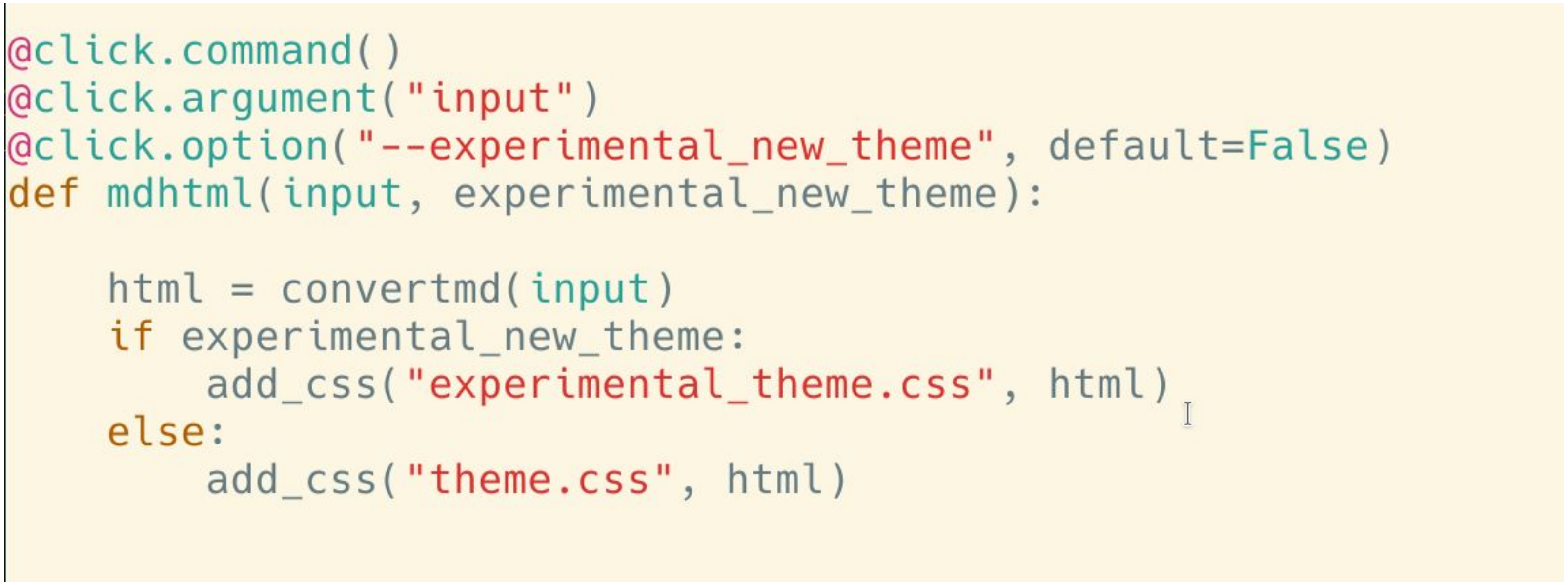
```python
if not os.path.exists(input_file):
    sys.stderr.write("{} does not exist\n".format(input_file))
    sys.exit(3)

try:
    convertmd(input_file)
except InvalidMarkdownError:
    sys.stderr.write("Input file had invalid Markdown\n")
    sys.exit(4)
```

- /usr/include/sysexits.h
- Advanced bash scripting guide: http://tldp.org/LDP/abs/html/exitcodes.html

Tool as an API - Experimental Options

- Take “feature flag” approach
- Define a standard for “experimental” options

- Requires more maintenance
- Eventually mature or get removed



```
@click.command()
@click.argument("input")
@click.option("--experimental_new_theme", default=False)
def mdhtml(input, experimental_new_theme):

    html = convertmd(input)
    if experimental_new_theme:
        add_css("experimental_theme.css", html)
    else:
        add_css("theme.css", html)
```

Tool as an API - Be Consistent

- `--verbose | -v`
- `--output | -o`
- `--version | -V`
- `--quiet | -q`
- `--help | -h`
- STDIN (-)

Tool as an API

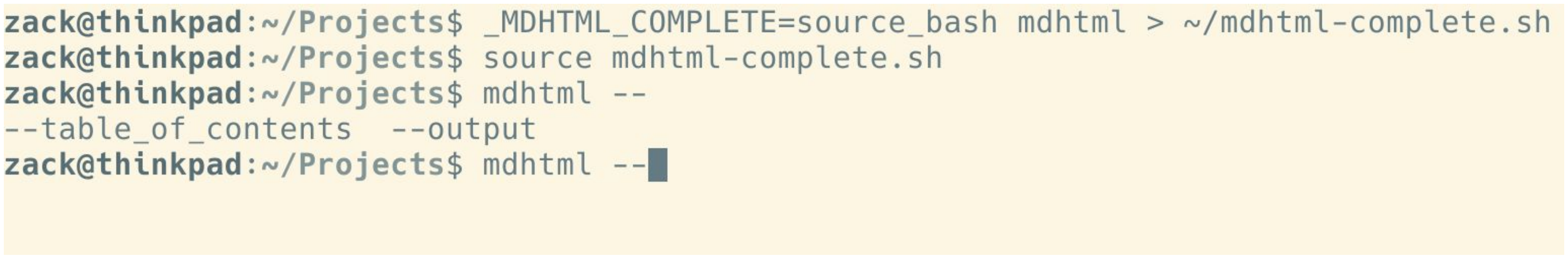
- We hate when APIs break, don’t break tools!
- Required / optional arguments
- Exit Codes
- Experimental Options
- Be consistent

Make Them Easy to Use

- Shell completion Scripts
- STDOUT/STDERR/STDIN
- MAN pages

Make Them Easy to Use - Shell Completion

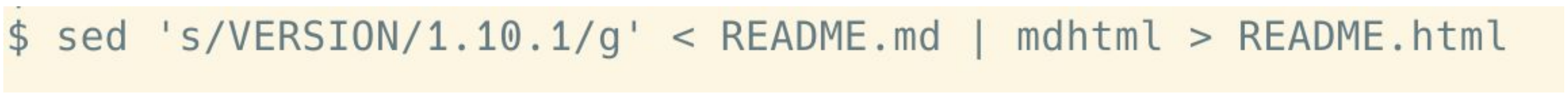
- Ship bash/zsh/fish completion scripts with your tools
- Click makes this really easy to do



```
zack@thinkpad:~/Projects$ _MDHTML_COMPLETE=source_bash mdhtml > ~/mdhtml-complete.sh
zack@thinkpad:~/Projects$ source mdhtml-complete.sh
zack@thinkpad:~/Projects$ mdhtml --
--table_of_contents  --output
zack@thinkpad:~/Projects$ mdhtml --
```

Make Them Easy to Use - With Other Tools

- STDOUT
- STERR
- STDIN



```
$ sed 's/VERSION/1.10.1/g' < README.md | mdhtml > README.html
```

Make Them Easy to Use - With Other Tools

- Reading from STDIN or a File

```python
import click

@click.command()
@click.option(
    "-t", "--table_of_contents", help="Set depth of table of contents"
)
@click.option("-o", "--output", help="Name of the output file")
@click.argument("input_file", type=click.File('r'), default="-")
def mdhtml(table_of_contents, output, input_file):
    # Do the markdown conversion
    pass
```

Make Them Easy to Use - MAN Pages

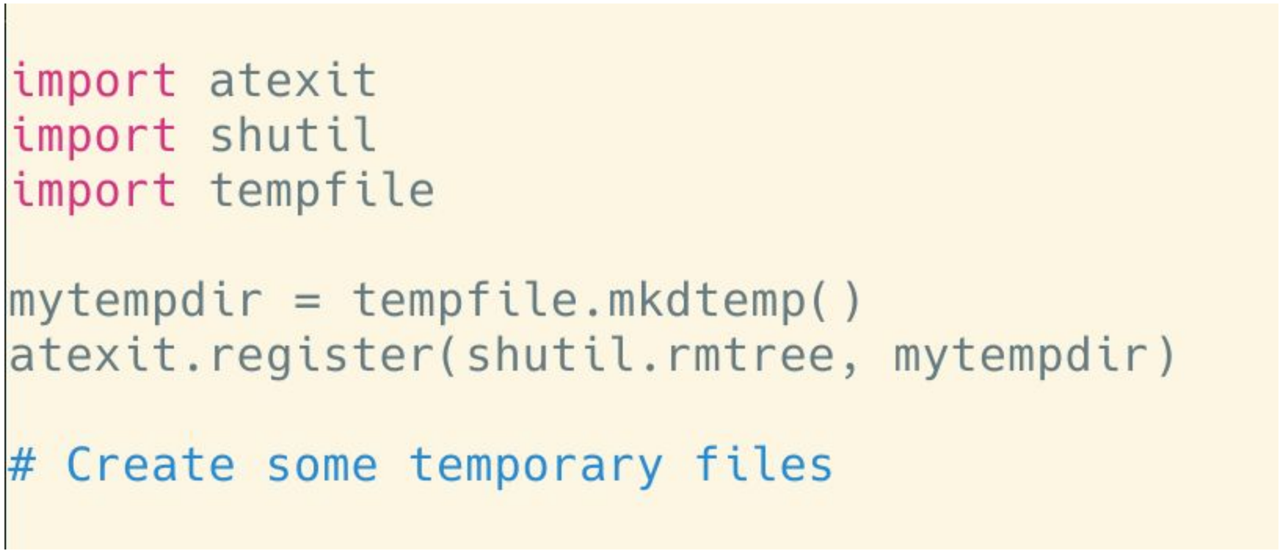
- Great for teams on UNIX
- Lots of ways to generate
  - Click-man
  - `pandoc --from=markdown --to=man mdhtml.md`
- `man mdhtml`

Make Them Easy to Use

- Shell completion Scripts
- STDOUT/STDERR/STDIN
- MAN pages

Other Useful Packages

- Delete temporary workspaces on exit



```python
import atexit
import shutil
import tempfile

mytempdir = tempfile.mkdtemp()
atexit.register(shutil.rmtree, mytempdir)

# Create some temporary files
```

Testing

- Check exit codes
- pytest datadir/tempdir
- Click build in testing

```python
def test_exit(mymodule):
    with pytest.raises(SystemExit) as exp:
        subprocess.check_call(["mdhtml", "bad_input_file.md"])
    assert exp.type == SystemExit
    assert exp.value.code == 3
```

What Makes a Good Tool?

- Tools that are useful
- Tools that are consistent
- Tools that are easy to use
- Tools that are well tested

Questions?