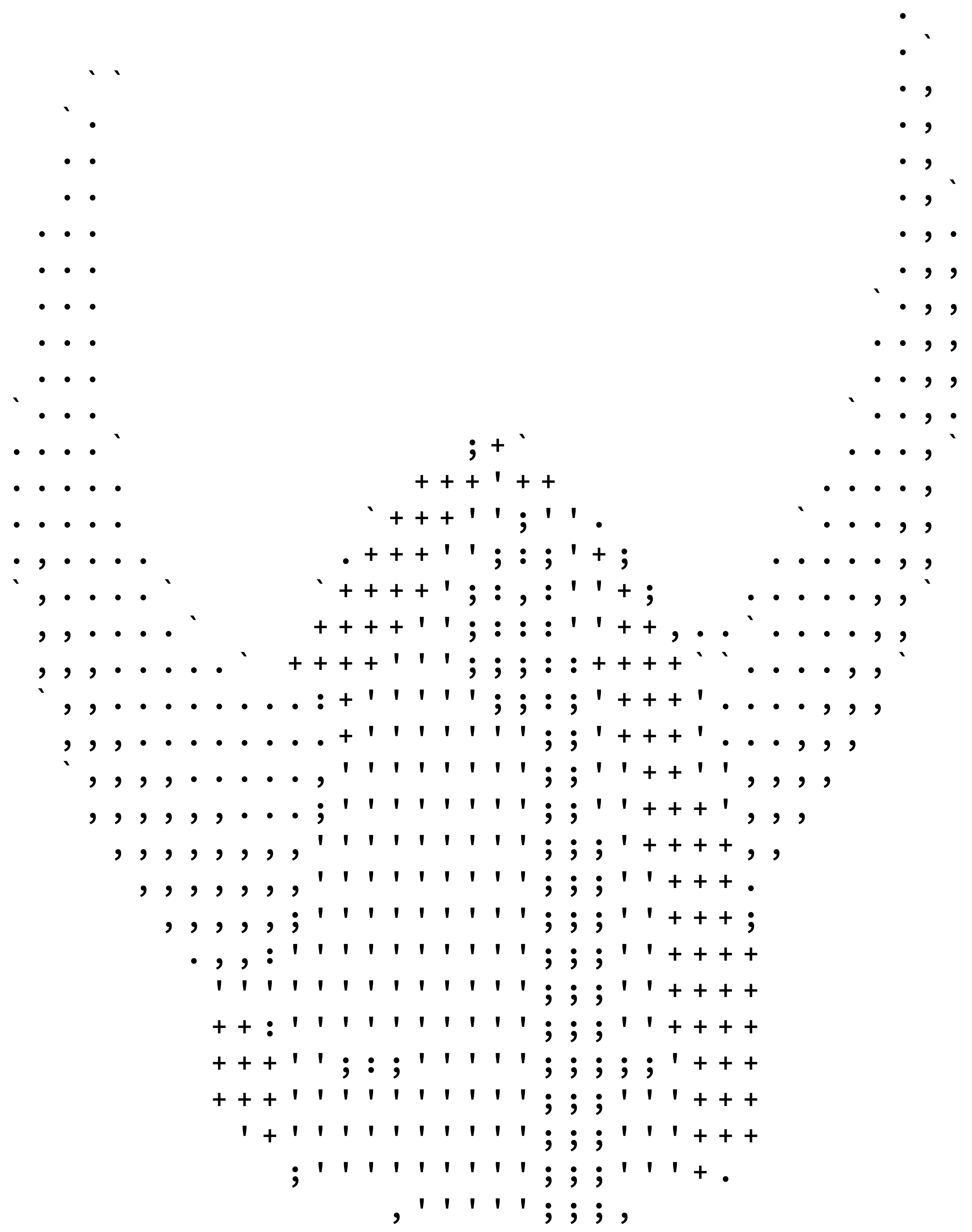
Build Awesome Command-Line Applications in Python

Who is this guy?



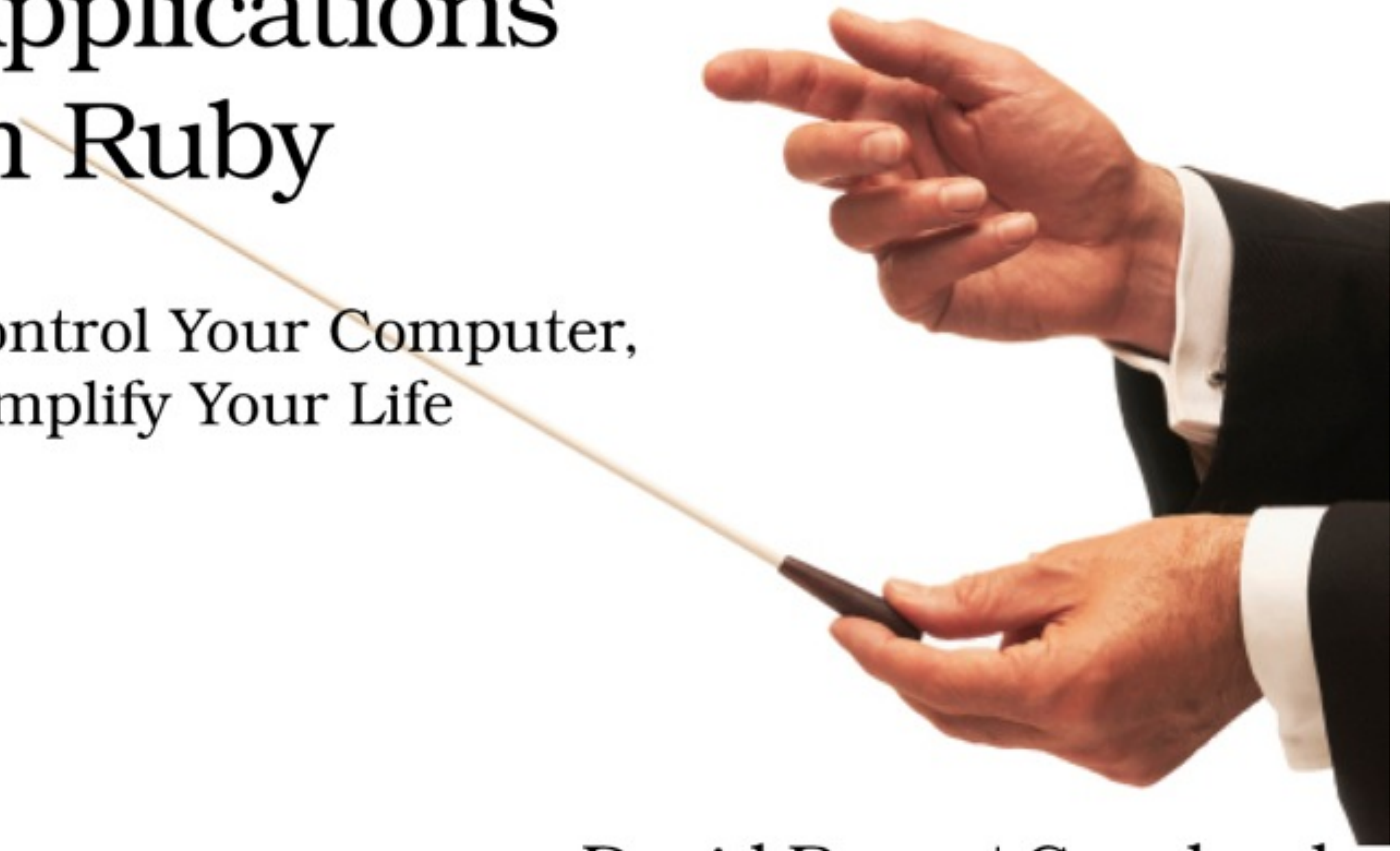
Name: Mark Smith
Company: FanDuel (we’re hiring)

Email: judy@judy.co.uk
Twitter: judy2k
Github: judy2k

The Pragmatic Programmers

Build Awesome Command-Line Applications in Ruby

Control Your Computer, Simplify Your Life



David Bryant Copeland

Edited by John Osborn

The Facets of Ruby Series

Build Awesome Command-Line Applications in Python

Control Your Computer,
Simplify Your Life

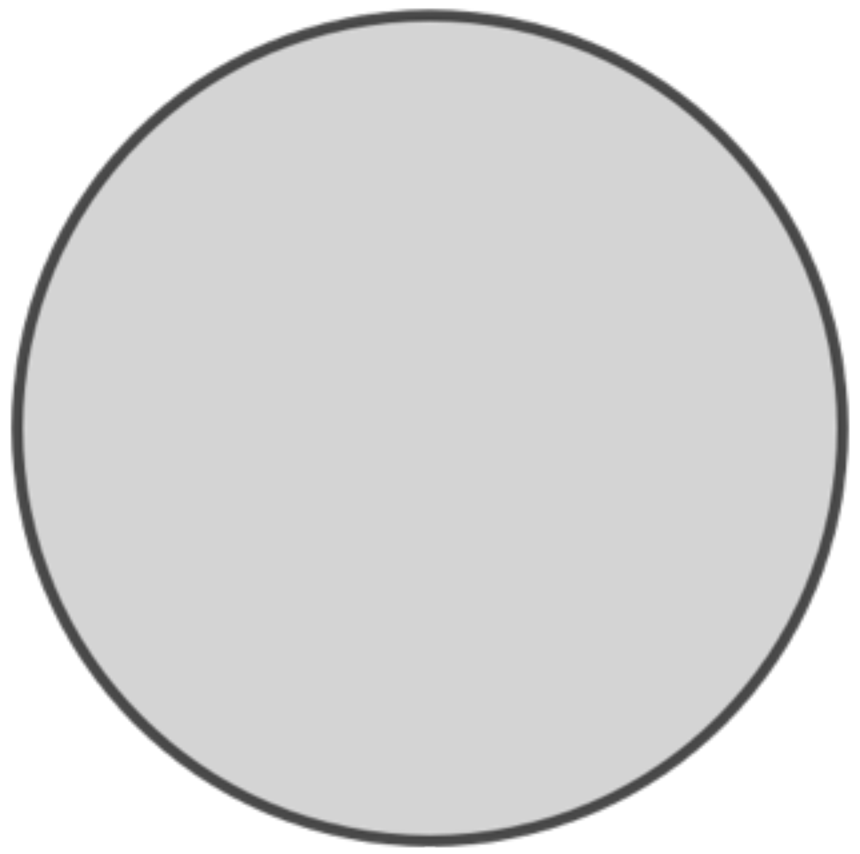
Me

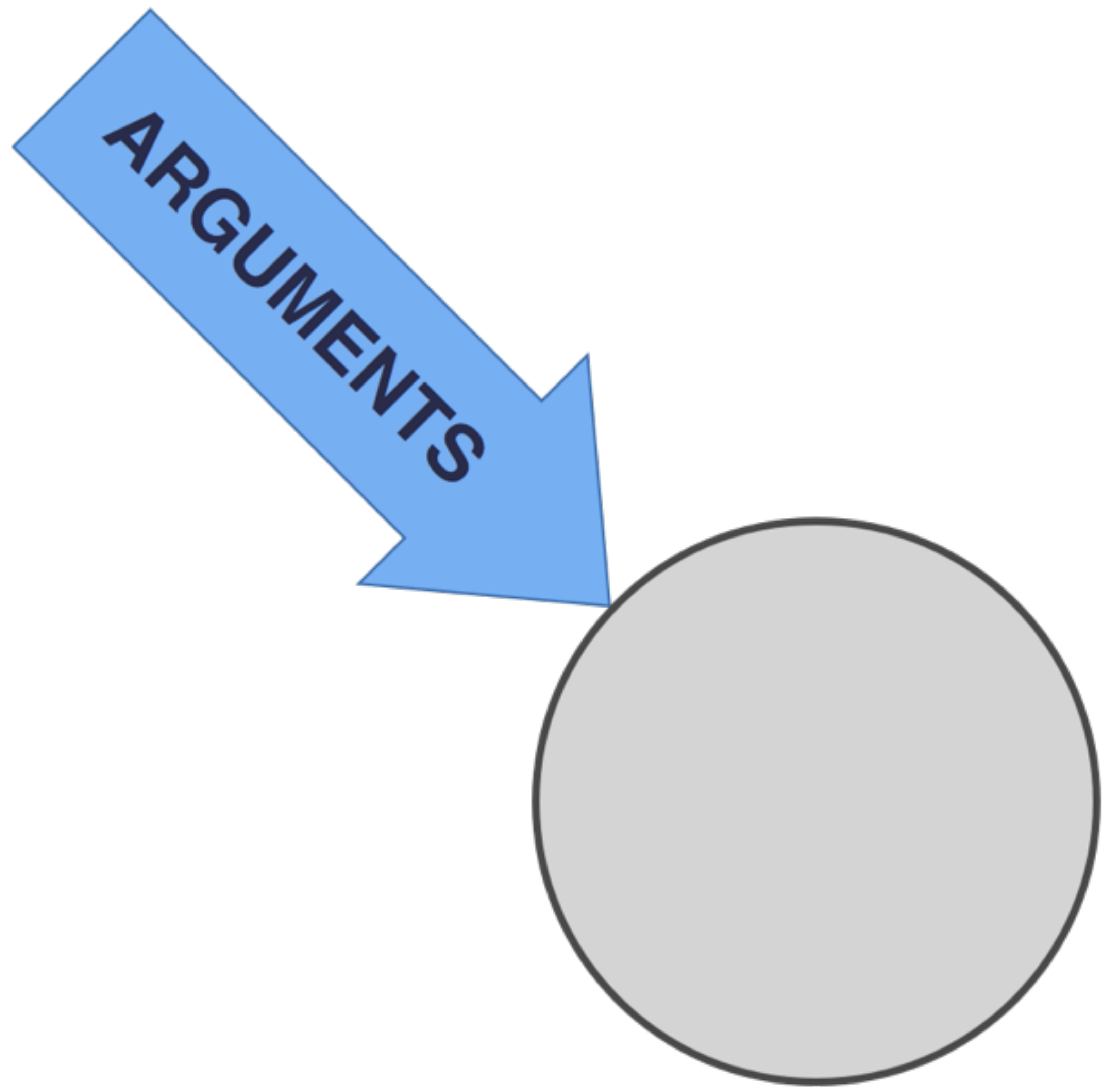
Why Write Command-Line Programs?

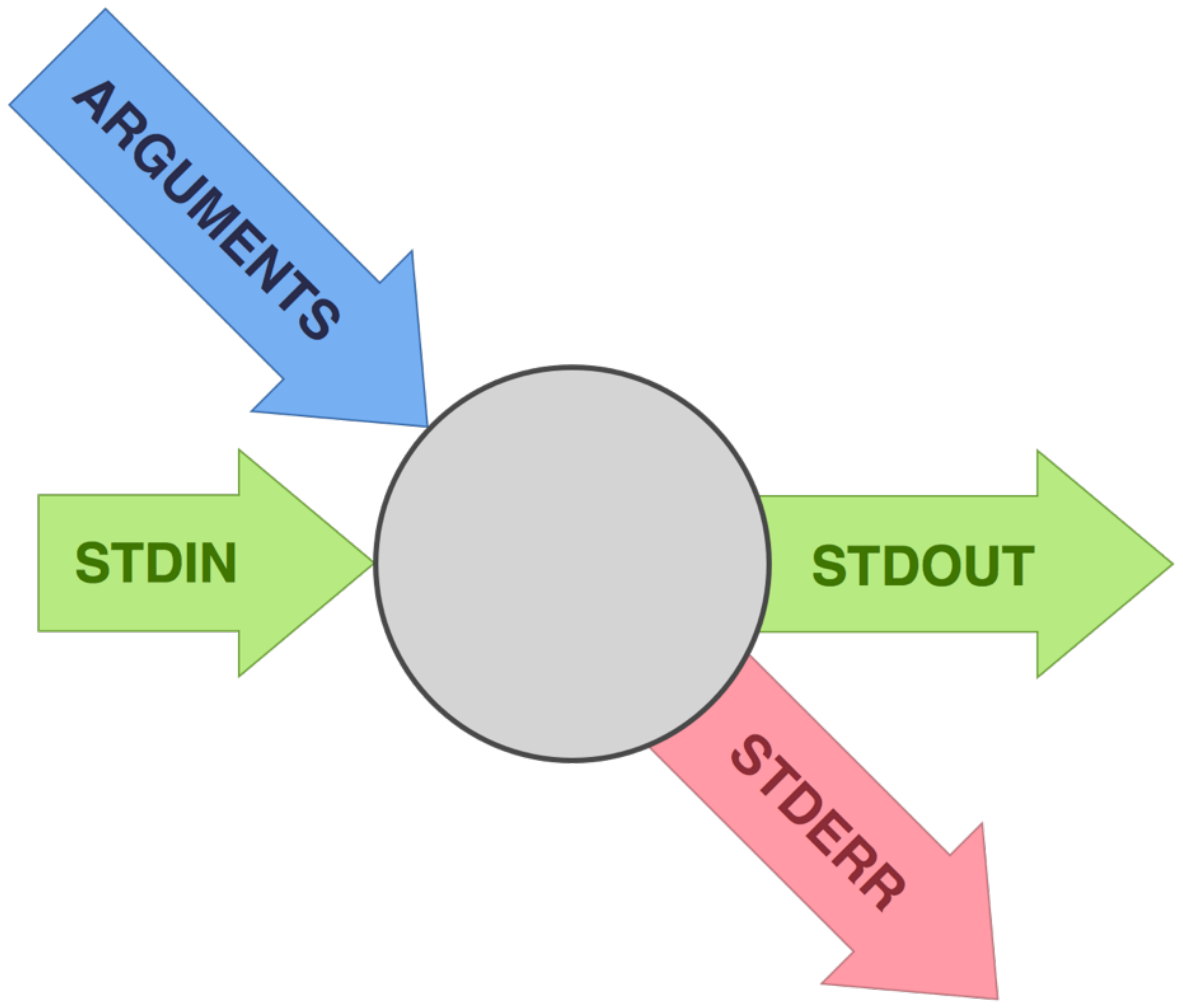
Scripts Are Vampires

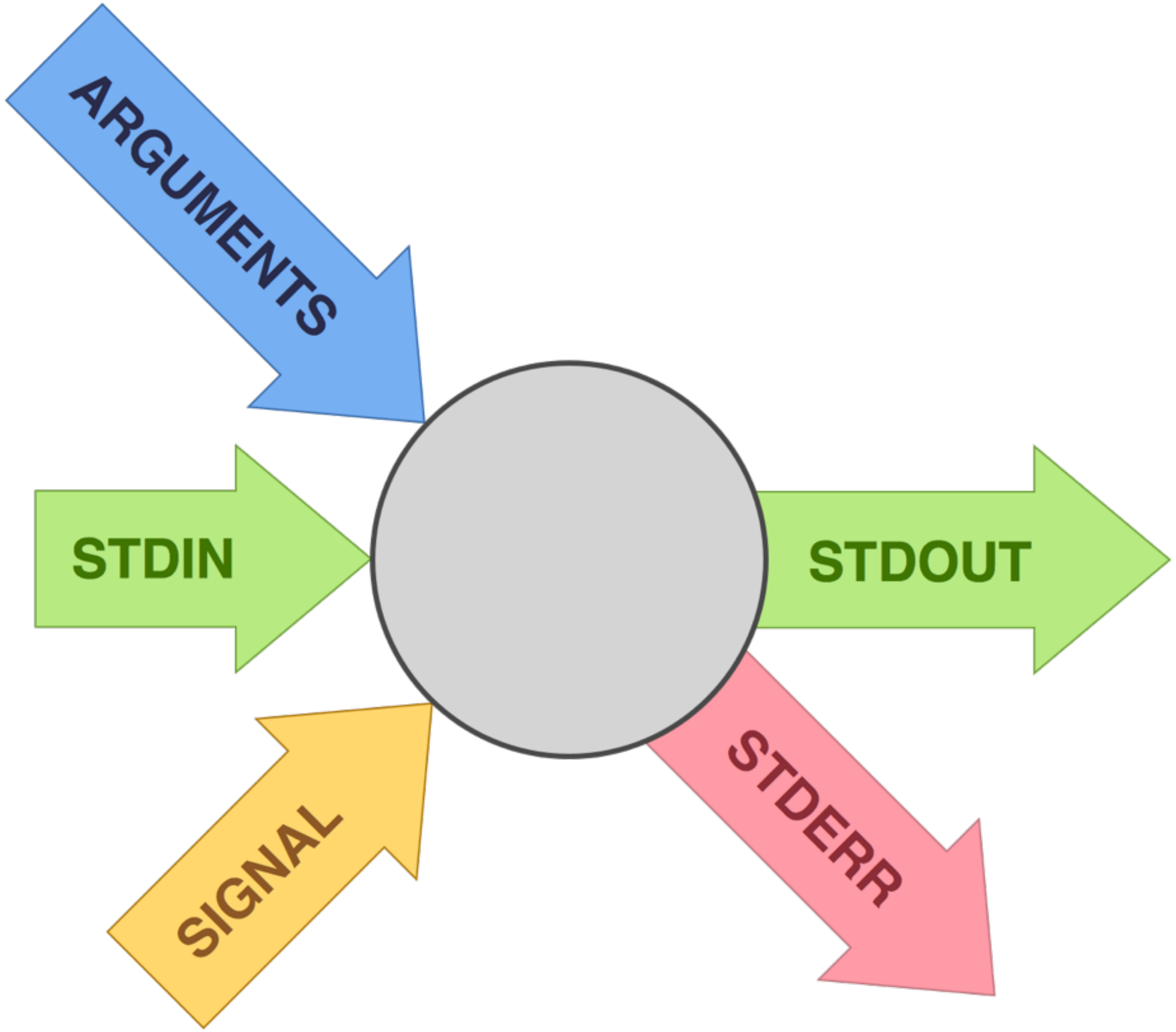
They live **forever**

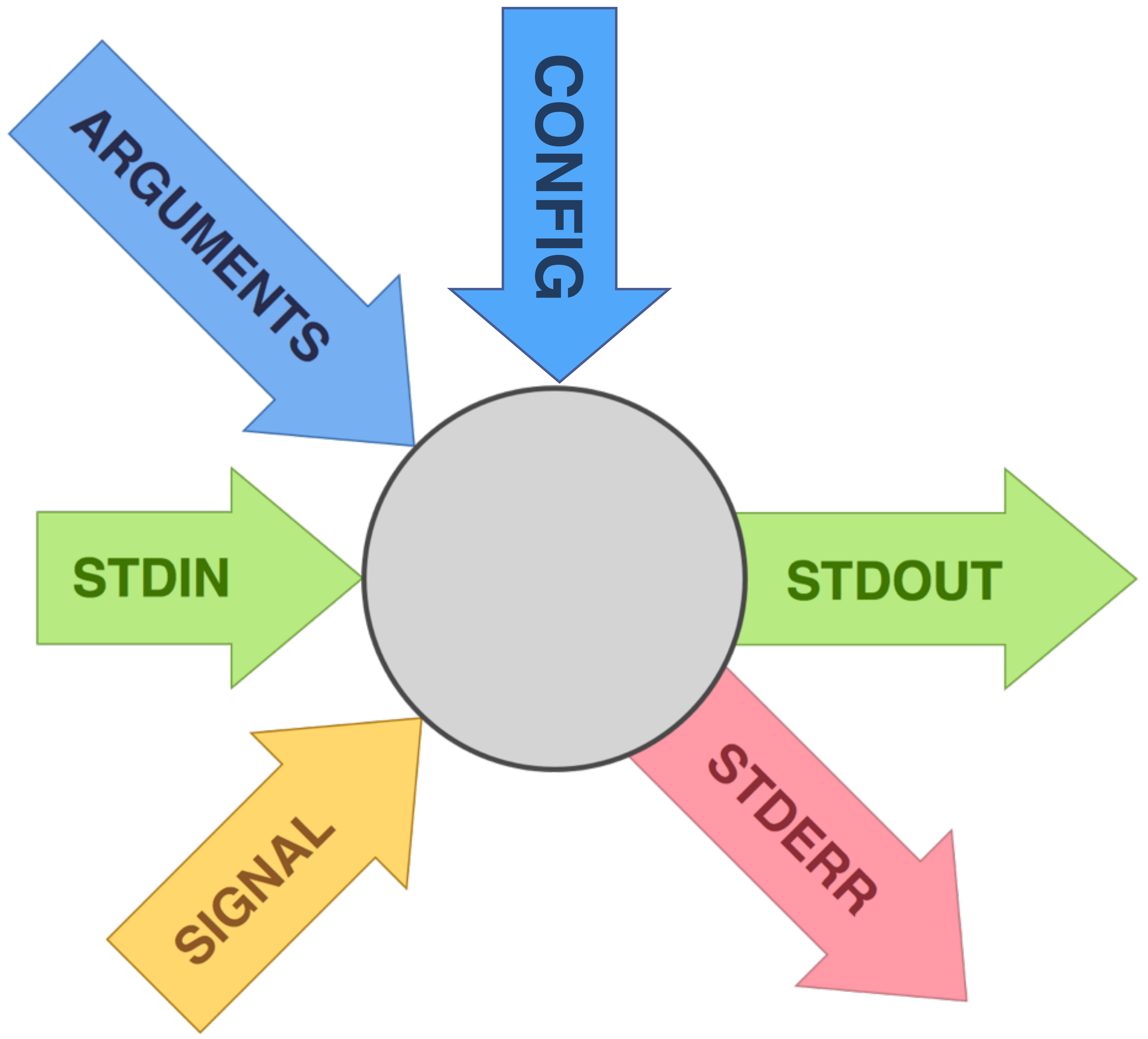
What Makes Up a Command-Line Program?

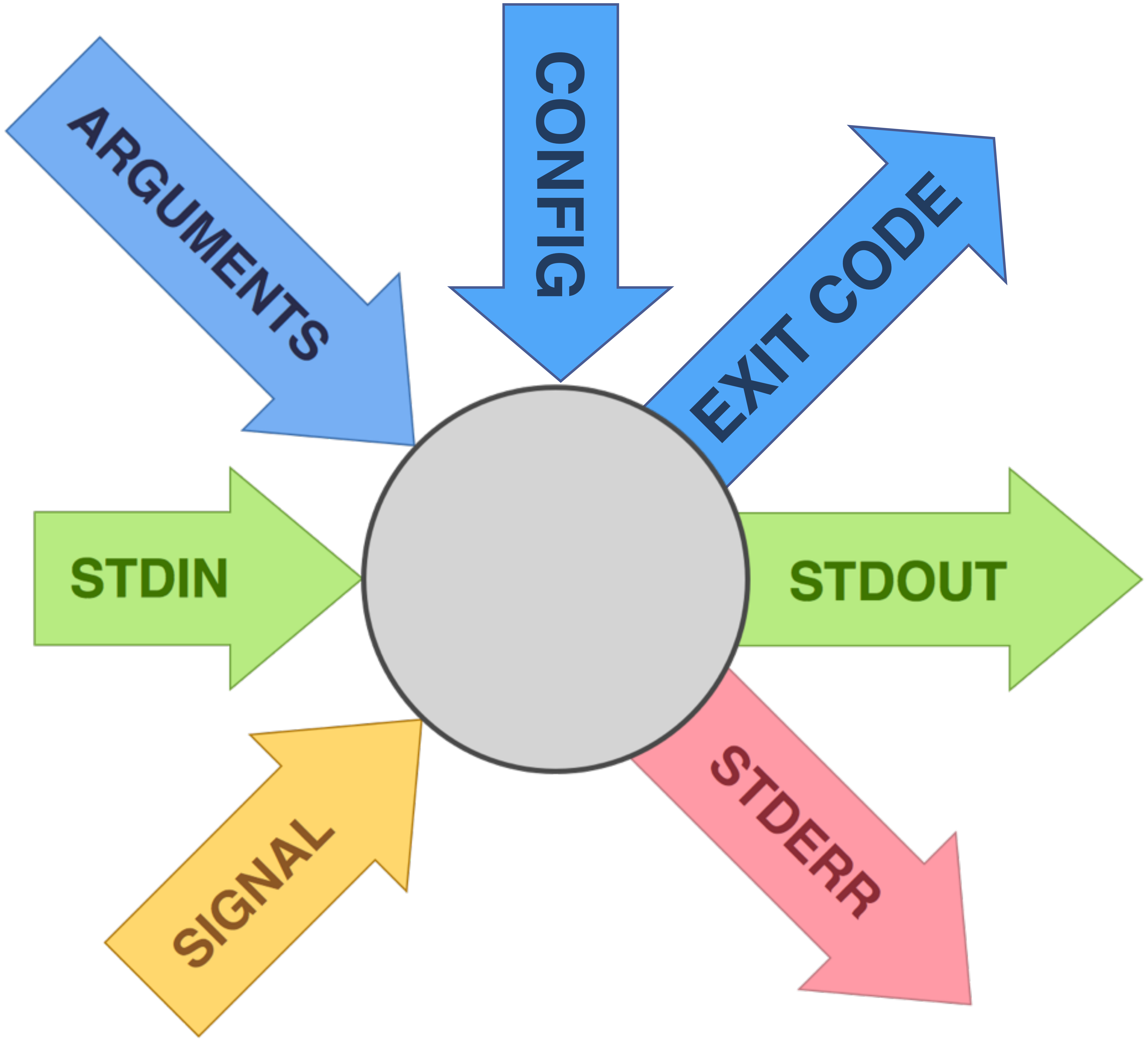



ARGUMENTS


ARGUMENTS
STDIN
STDOUT
STDERR


ARGUMENTS
STDIN
SIGNAL
STDOUT
STDERR


ARGUMENTS
CONFIG
STDIN
STDOUT
SIGNAL
STDERR


ARGUMENTS
CONFIG
EXIT CODE
STDIN
STDOUT
SIGNAL
STDERR

Command-line Parsing

```
 ______________
< Speak to me! >
 --------------
        \   ^__^
         \  (oo)\_______
            (__)\       )\/\
                ||----w |
                ||     ||
```

CLA Libraries

In The Box	3rd Party
Do-it-yourself	docopt
getopt	clint
optparse	click
argparse	**compago**

argparse

```
from argparse import ArgumentParser

def main():
    ap = ArgumentParser()
    ap.add_argument('name', nargs='?')
    args = ap.parse_args()
    name = (args.name or 'World')
    print "Hello,", name, "!"
```

```
$ ./hello_world4.py
Hello, World !


$ ./hello_world4.py Douglas
Hello, Douglas !
```

```
$ ./hello_world4.py --help
usage: hello_world4.py [-h] [name]

positional arguments:
  name

optional arguments:
  -h, --help  show this help message and exit
```

```
ap = ArgumentParser()
ap.add_argument('-v', '--verbose',
    default=False, action='store_true',
    help='Increase verbosity’)
ap.add_argument('-n', '--number',
    type=int, default=1,
    help="The number of times to greet NAME")
ap.add_argument('name', help="The person to greet")
args = ap.parse_args()

for index in range(args.number):
    print "Hello,", args.name, "!"
if args.verbose:
    print "I've finished now."
```

compago

```
import compago
app = compago.Application()

@app.command
def greet(to="world"):
    print "Hello,", to, "!"

@app.command
def ungreet(to="world"):
    print "Goodbye,", to, "!"

if __name__ == ‘__main__': app.run()
```

```
$ ./program greet --to=Mark
Hello, Mark!

$ ./program ungreet
Goodbye, world!
```

Input & Output

```
 ____________
< I'm hungry >
 ------------
        \   ^__^
         \  (oo)\_______
            (__)\       )\/\
                ||----w |
                ||     ||
```

`stdout` vs. **`stderr`**

logging

```
logging.basicConfig(level=logging.WARNING,
                    format="%(msg)s")

if options.verbose:
    logging.getLogger().setLevel(logging.DEBUG)

LOG = logging.getLogger('logtest')

LOG.debug('Running main')
LOG.info('Everything is okay')
LOG.warning('EVERYTHING HAS GONE WRONG!')
```

Are you talking to a user?

isatty()

```
from sys import stdin, stdout, stderr

print "Piped input:",  not stdin.isatty()
print "Piped output:", not stdout.isatty()
print "Piped error:",  not stderr.isatty()
```

```
$ ./isatty.py
Piped input: False
Piped output: False
Piped error: False

$ ./isatty.py | cat
Piped input: False
Piped output: True
Piped error: False

$ echo 'Hello' | ./isatty.py | cat
Piped input: True
Piped output: True
Piped error: False
```

Colour

colorama

```
from colorama import Fore, Back, Style

print Fore.RED + 'some red text'
print Back.GREEN + 'and with a green background'
print Style.BRIGHT + 'and in bright text',
print Fore.RESET + Back.RESET + Style.RESET_ALL
print 'back to normal now'
```

```
$ python colour.py
some red text
and with a green background
and in bright text
back to normal now
```

```
$ ./colour > saved_data.txt
$ vim saved_data.txt
^[[31msome red text
^[[42mand with a green background
^[[1mand in bright text ^[[39m^[[49m^[[0m
back to normal now
```

Stripping Colour

```
import colorama

if not sys.stdout.isatty():
    colorama.init(strip=True)


print Fore.RED + 'some red text'
print Back.GREEN + ‘and a green background'
```

User Credentials

```
import getpass

username = getpass.getuser()
password = getpass.getpass()

print ( “You are {username}, and you should never use the
password ‘{password}’ again!”.format(
    username=username,
    password=password
)
```

Output

```
$ python credentials.py
Password:

You are mark and you should never use the password
'passw0rd' again!
```

Progress

```
 _____________________
< Should I kill this? >
 ---------------------
        \   ^__^
         \  (oo)\_______
            (__)\       )\/\
                ||----w |
                ||     ||
```

progressbar2

```
from progressbar import *
import time

progress = ProgressBar()
for i in progress(range(80)):
  time.sleep(0.01)

# 100% |#######################################|
```

progressbar2

```
widgets = ['Loading: ', Percentage(), ' ', Bar(),
           ' ', ETA(), ' ', FileTransferSpeed()]

pbar = ProgressBar(widgets=widgets, maxval=20000).start()

for i in range(20000):
    pbar.update(i)
    time.sleep(.005)
pbar.finish()

# Loading:   9% |#         | ETA:  0:01:42 177.08  B/s
```

Think About:

Adding a flag to specify output format.

Adding a flag to control verbosity/quietness.

Be responsive - tell the user how things are going (unless they ask you not to.)

Configuration

Config Choices

In The Box	3rd Party
INI	YAML
Environment Vars	Java Properties
JSON	
CSV	
XML	
Apple Plist	
Do-it-yourself	

INI Files

```
from ConfigParser import SafeConfigParser
from os.path import dirname, join, expanduser

INSTALL_DIR = dirname(__file__)

config = SafeConfigParser()
config.read([
    join(INSTALL_DIR, 'defaults.ini'),
    expanduser(‘~/.tool.ini’),
    'config.ini'
])
```

```
# tool/defaults.ini -----
[server]
# Default host and port:
host=localhost
port=8080
url=http://%(host)s:%(port)s/
```

```
# ~/.tool.ini ------------
[server]
# My servers all use 5000:
port=5000
```

```
# project.ini --------------
[server]
# Special hostname:
host=www.ninjarockstar.guru
```

INI Files

```
print config.get('server', ‘host’)
  => www.ninjarockstar.guru

print config.getint('server', ‘port')
  => 5000

print config.get('server', ‘url’)
  => http://www.ninjarockstar.guru:5000/
```

Signals

Signals Package

```
import signal

signal.siginterrupt(signal.SIGINFO, False)

signal.signal(signal.SIGINFO, mysiginfofunc)
```

KeyboardInterrupt

```
def main():
    try:
        time.sleep(5)
    except KeyboardInterrupt:
        pass


if __name__ == '__main__':
    main()
```

Code Structure & Packaging

Structure

```
mytool-project/
  setup.py
  mytool
  mytoollib/
    __init__.py
    __main__.py
    mytool.py
    utils.py
```

setuptools

```
# setup.py
setup(name = 'mytool',
    version = '2.0',
    url = ‘http://mytool.ninjarockstar.guru/',
    license = 'BSD License',
    author = ‘Mark Smith',
    author_email = ‘judy@judy.co.uk’,
    description = ‘A tool with little purpose.',
    keywords = 'utils',
    packages = ‘mytoollib’,
    scripts = [‘mytool’]
    platforms = ‘any')
```

Exit Codes

```
 _______________________
< let's get out of here >
 -----------------------
        \   ^__^
         \  (oo)\_______
            (__)\       )\/\
                ||----w |
                ||     ||
```

Exit Codes

```
# Normal termination exits with 0

# Uncaught exceptions exit with 1

# … or you can explicitly exit:
sys.exit(exit_code)
```

Skipped

CLI frameworks (**cliff** & **clint**)

Cross-platform considerations

```
#!python

print “Any questions?”

exit(0)

# https://github.com/judy2k/command-line-talk
```