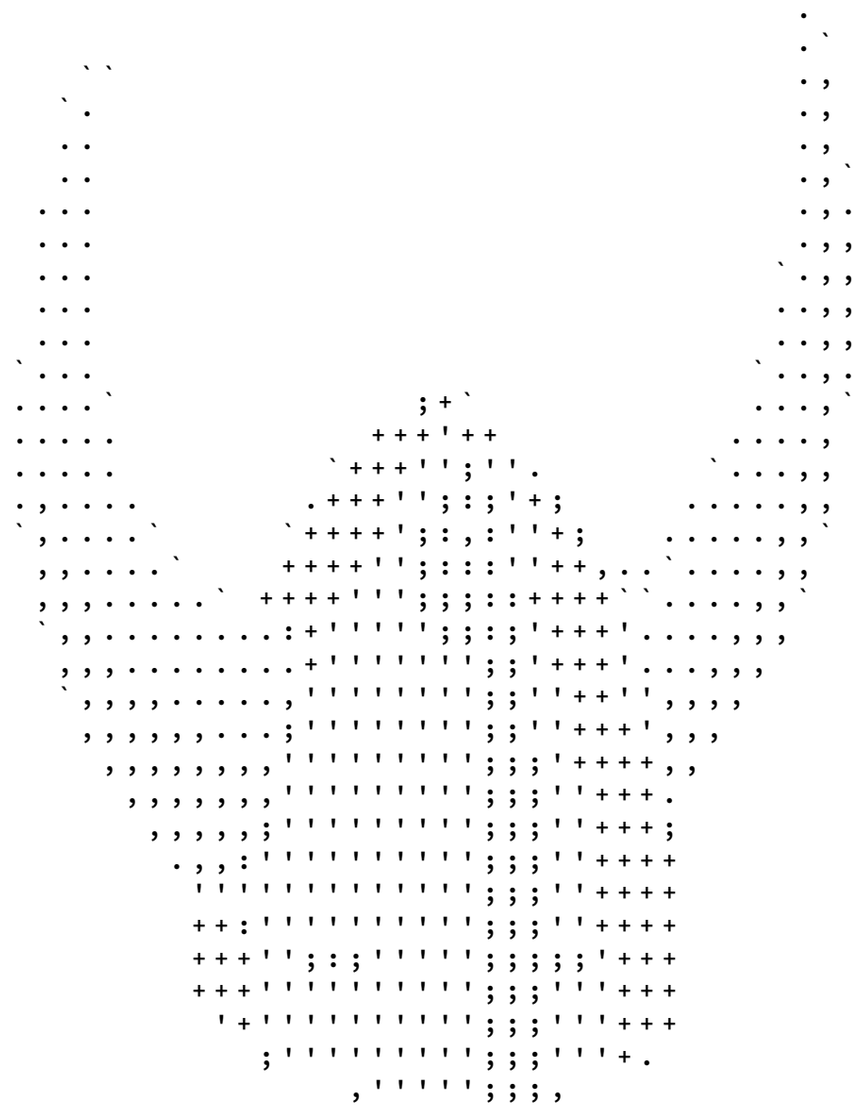
Command-Line Programs in Python

Who is this guy?



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

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

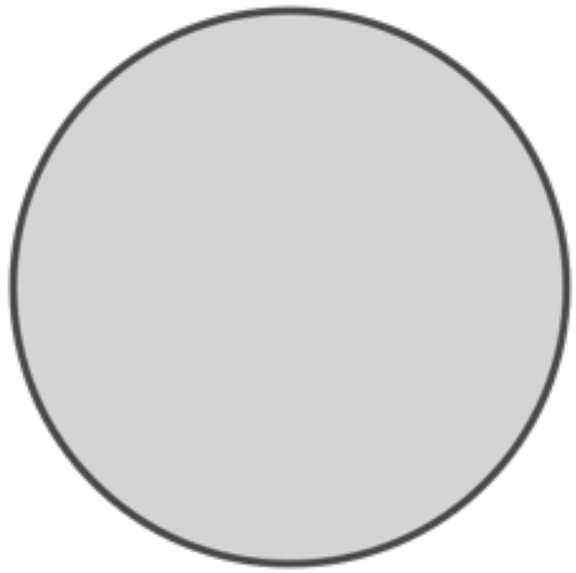
Why Write Command-Line Programs?

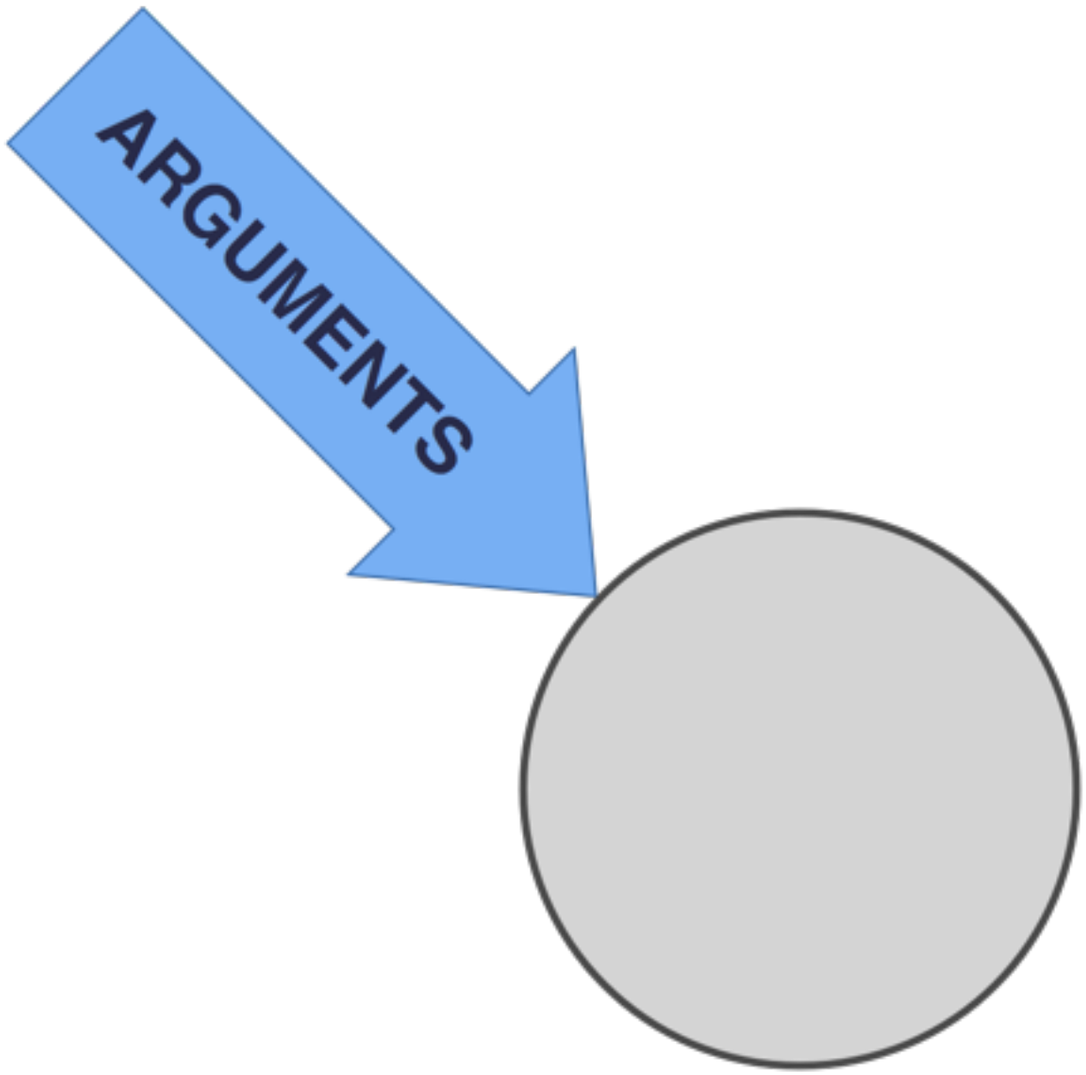
“Command-Line UI
is not inferior
to GUI UI
– just different.”

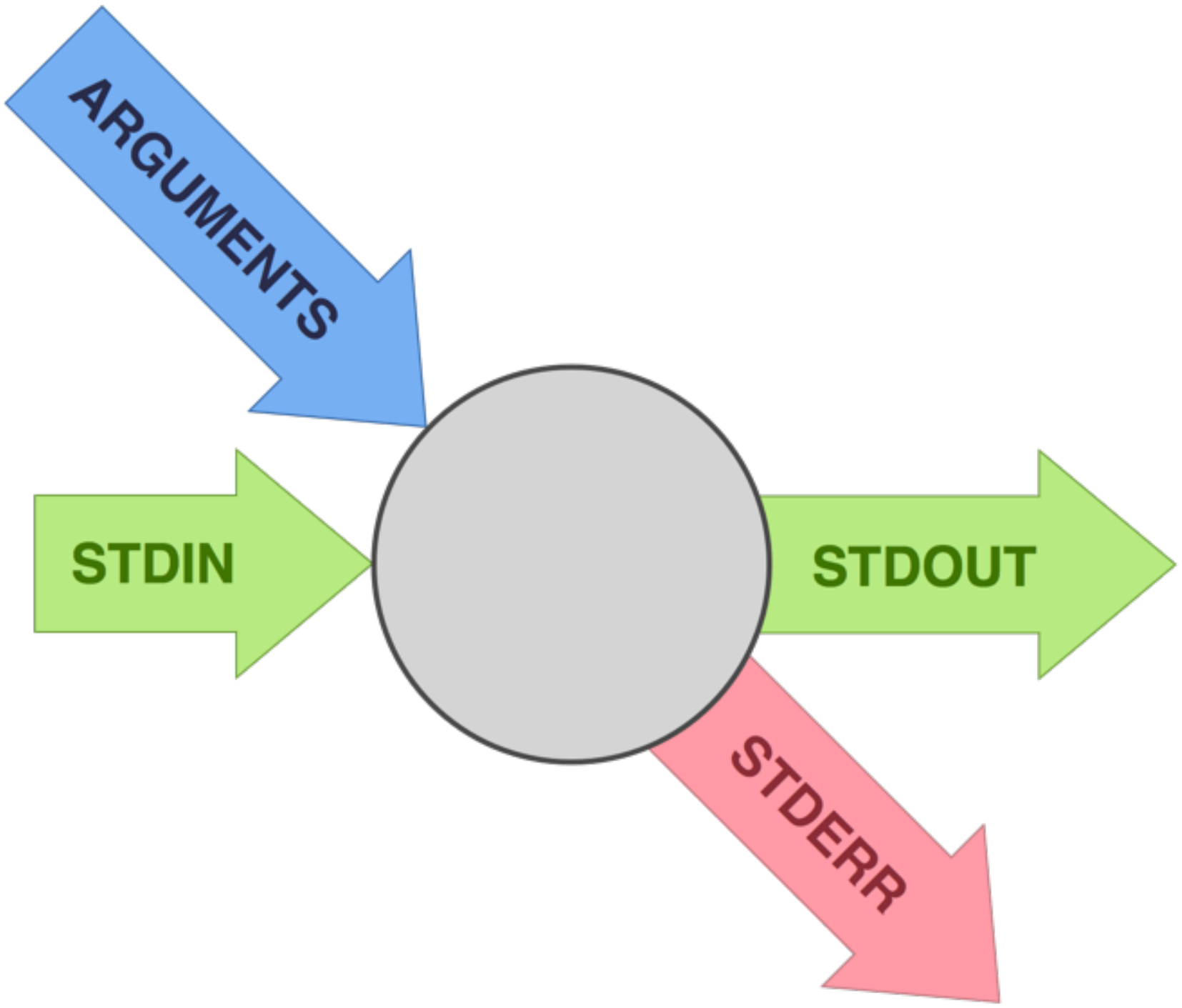
– Some Guy on Twitter

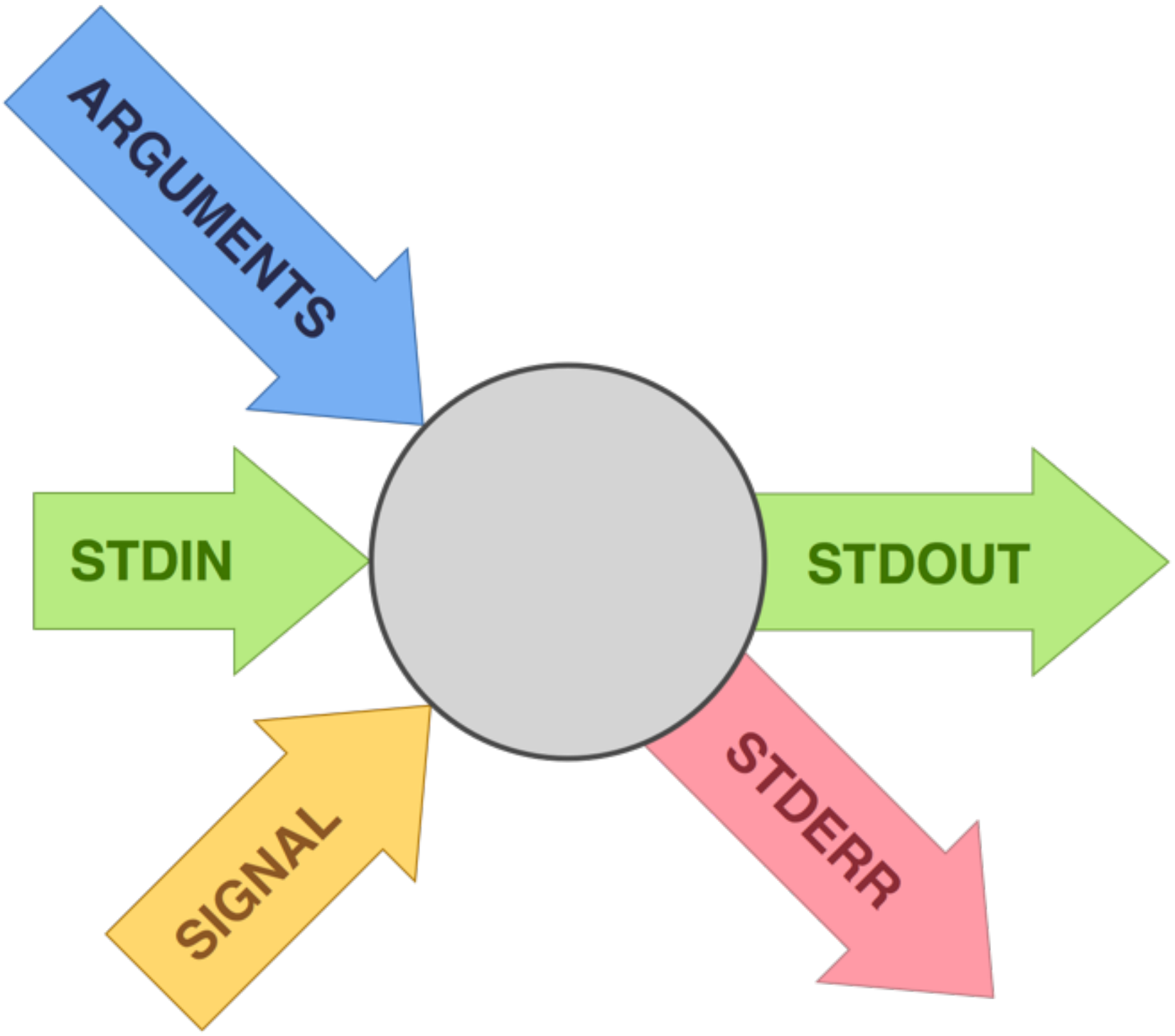
Why Write Command-Line Programs?

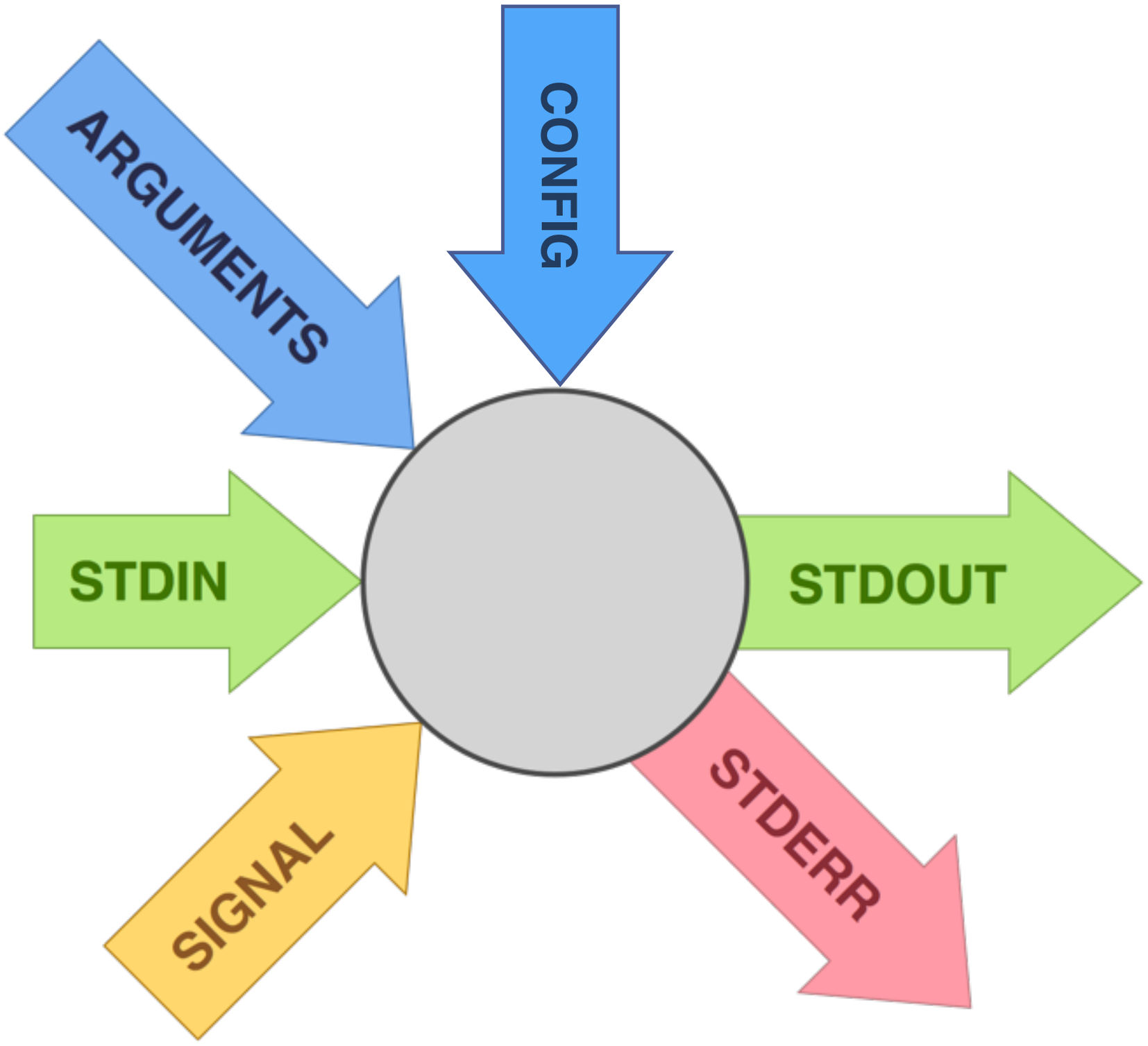
What Makes Up a Command-Line Program?

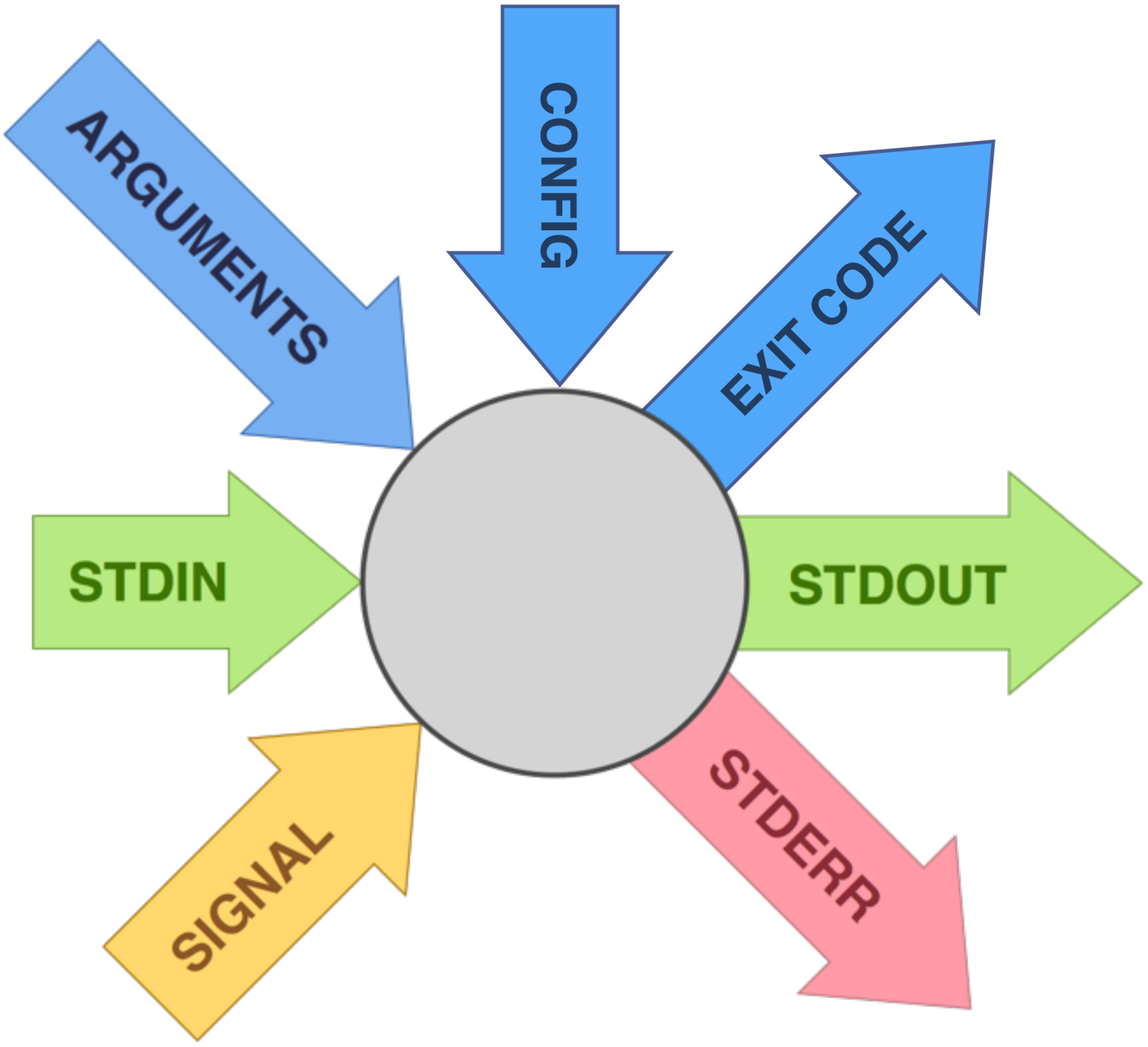



ARGUMENTS


ARGUMENTS
STDIN
STDOUT
STDERR


ARGUMENTS
STDIN
STDOUT
SIGNAL
STDERR


ARGUMENTS
CONFIG
STDIN
STDOUT
SIGNAL
STDERR


ARGUMENTS
CONFIG
EXIT CODE
STDIN
STDOUT
SIGNAL
STDERR

Why Python?

Let’s write a command-line app

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-
print "Hello, World!"
```

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-

def main():
    print "Hello, World!"

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

```
import sys

def main():
    if len(sys.argv) > 1:
        name = sys.argv[1]
    else:
        name = "World"
    print "Hello,", name, "!"

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

```
$ ./hello_world3.py
Hello, World !
```

```
$ ./hello_world3.py Douglas
Hello, Douglas !
```

Command-line Parsing

CLA Libraries

In The Box	3rd Party
Do-it-yourself	docopt
getopt	cliff
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()
```

Input & Output

Boring Bits

```
# Print to stdout:
print "You've seen this all before."

# Print to stderr:
print >>sys.stderr, "This is not *standard* output"

# Read from stdin:
for line in sys.stdin:
    print line,

# Read from a TTY
answer = raw_input("Are you well? ")
```

Formatting

The old way

```
A_LIST_OF_WORDS = ['installation',
                   'Chamicuro',
                   'foliiferous',
                   'spermatic',
                   'intemperately',
                   'pederastically',
                   'proctosigmoidectomy',
                   'begar']

for word in A_LIST_OF_WORDS:
    print "|%02d|%-20s|" % (len(word), word)
```

Output:

```
|12|installation          |
|09|Chamicuro             |
|11|foliiferous           |
|09|spermatic             |
|13|intemperately         |
|14|pederastically        |
|19|proctosigmoidectomy   |
|05|begar                 |
```

The new way

```
for word in A_LIST_OF_WORDS:
    print "{word:15} {length:>2}".format(
        word=word,
        length=len(word))
```

Output

```
installation    12
Chamicuro        9
foliiferous     11
spermatic        9
intemperately   13
pederastically  14
proctosigmoidectomy 19
begar            5
```

Width Calculation

```
max_width = max(len(w) for w in A_LIST_OF_WORDS)

for word in A_LIST_OF_WORDS:
    print "{word:{col_width}} {length:>2}".format(
        word=word,
        length=len(word),
        col_width=max_width)
```

Output

```
installation          12
Chamicuro              9
foliiferous           11
spermatic              9
intemperately         13
pederastically        14
proctosigmoidectomy   19
begar                  5
```

Are you talking to a user?

```
#!/usr/bin/env python

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
```

User Credentials

```
import getpass

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

print "You are", username, \
    "and you should never use the password
'", password, \
    "' again!"
```

Output

```
$ python credentials.py
Password:
You are mark and you should never use
the password ' passw0rd ' again!
```

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
```

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

```
# 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

```
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'
])

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, sys, time

def siginfo(sig, frame):
    print "Some info"
    return True

signal.siginterrupt(signal.SIGINFO, False)
signal.signal(signal.SIGINFO, siginfo)

def main(argv=sys.argv[1:]):
    print 'start'
    while True:
        print '.',
        sys.stdout.flush()
        time.sleep(1)
    print 'finish'
```

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
        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

Exit Codes

```
# Normal termination exits with 0

# Uncaught exceptions exit with 1

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

Skipped

Using **logging** for output

Using **pyprogress** for progress bars

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

Multithreading and signals

Cross-platform considerations

Summary

Think Context

How will your program be run?

Is your user a user?

What do they want to know?

What will they want to do?

```
#!python

print “Any questions?”

exit(0)

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