Make your command line scripts / apps user-friendly

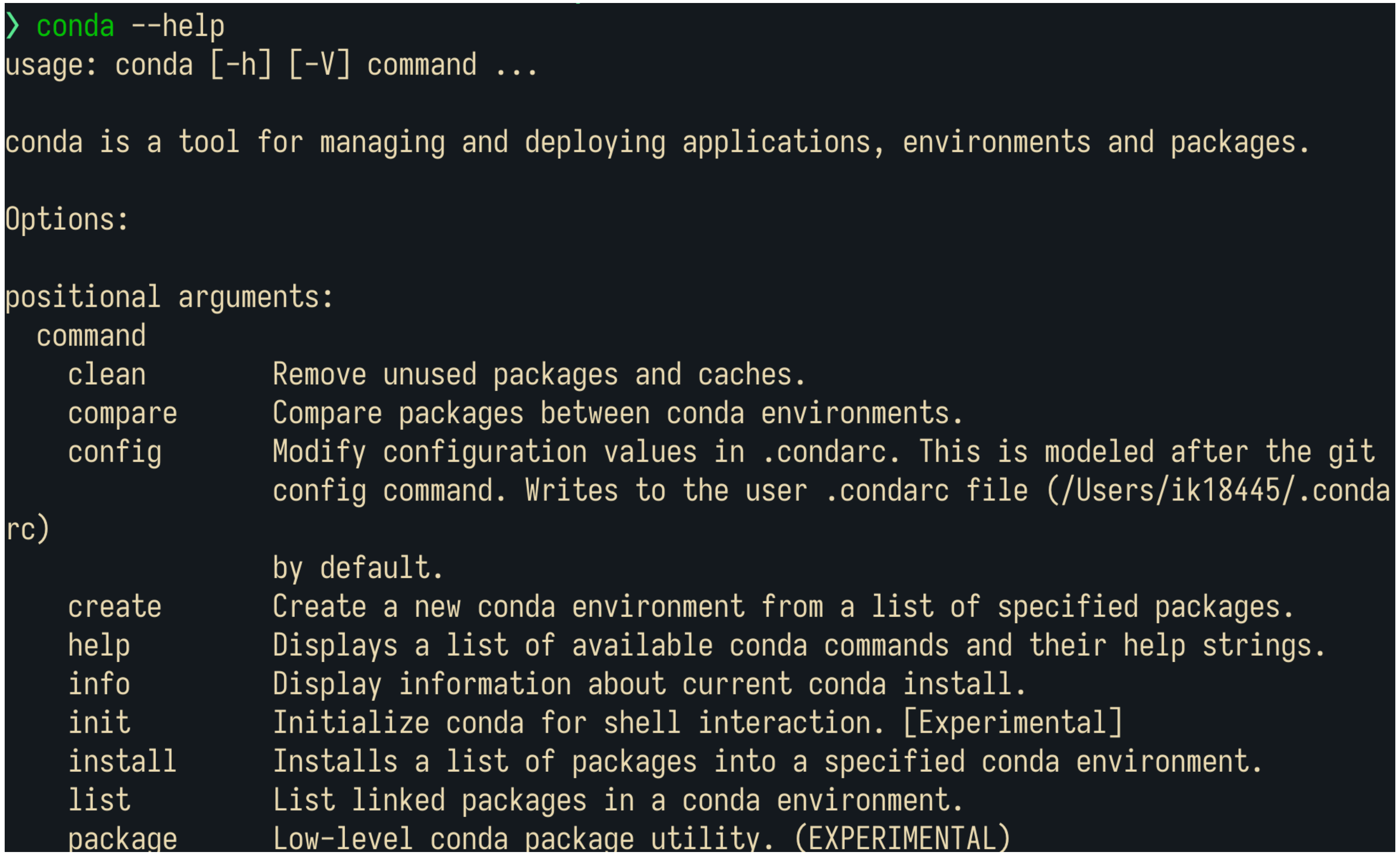
Yi Liu

part of the workshop “How to design user-friendly software”

- Some of the apps (command line, GUI, web) we use are just a bunch of scripts glued together
- The scope of this talk is limited to command line applications, i.e. “processing pipelines”
- How command line apps interface with users?
    - git status
    - git commit -m “I have done something”
    - rm -rf /*
- This talk covers the basics of command line arguments and some 3rd party libraries

Command line arguments

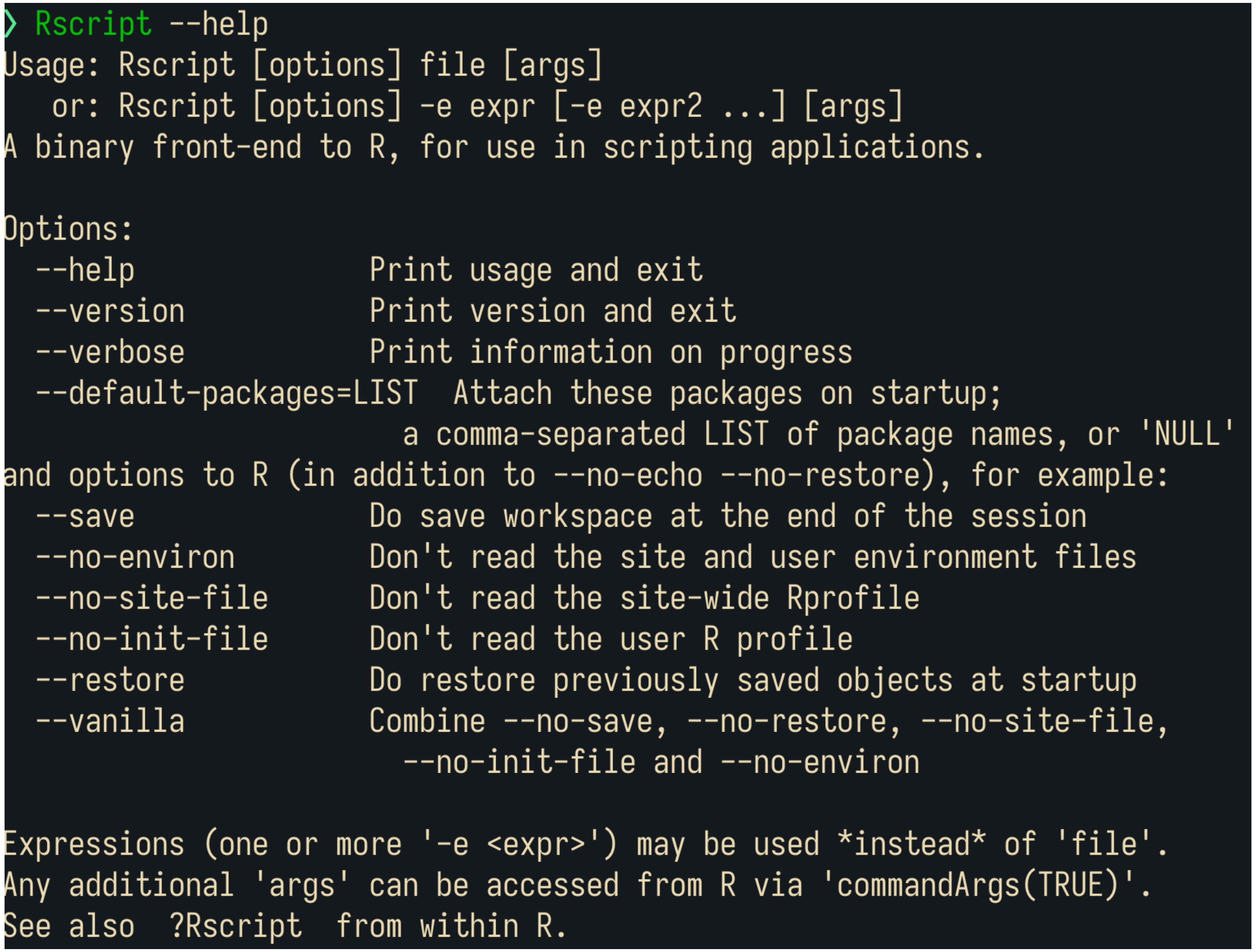
conda --help



```
❯ conda --help
usage: conda [-h] [-V] command ...

conda is a tool for managing and deploying applications, environments and packages.

Options:

positional arguments:
  command
    clean        Remove unused packages and caches.
    compare      Compare packages between conda environments.
    config       Modify configuration values in .condarc. This is modeled after the git
                 config command. Writes to the user .condarc file (/Users/ik18445/.conda
rc)
                 by default.
    create       Create a new conda environment from a list of specified packages.
    help         Displays a list of available conda commands and their help strings.
    info         Display information about current conda install.
    init         Initialize conda for shell interaction. [Experimental]
    install      Installs a list of packages into a specified conda environment.
    list         List linked packages in a conda environment.
    package      Low-level conda package utility. (EXPERIMENTAL)
```

- main command (binary / script)
- flags
- positional args

Rscript --help

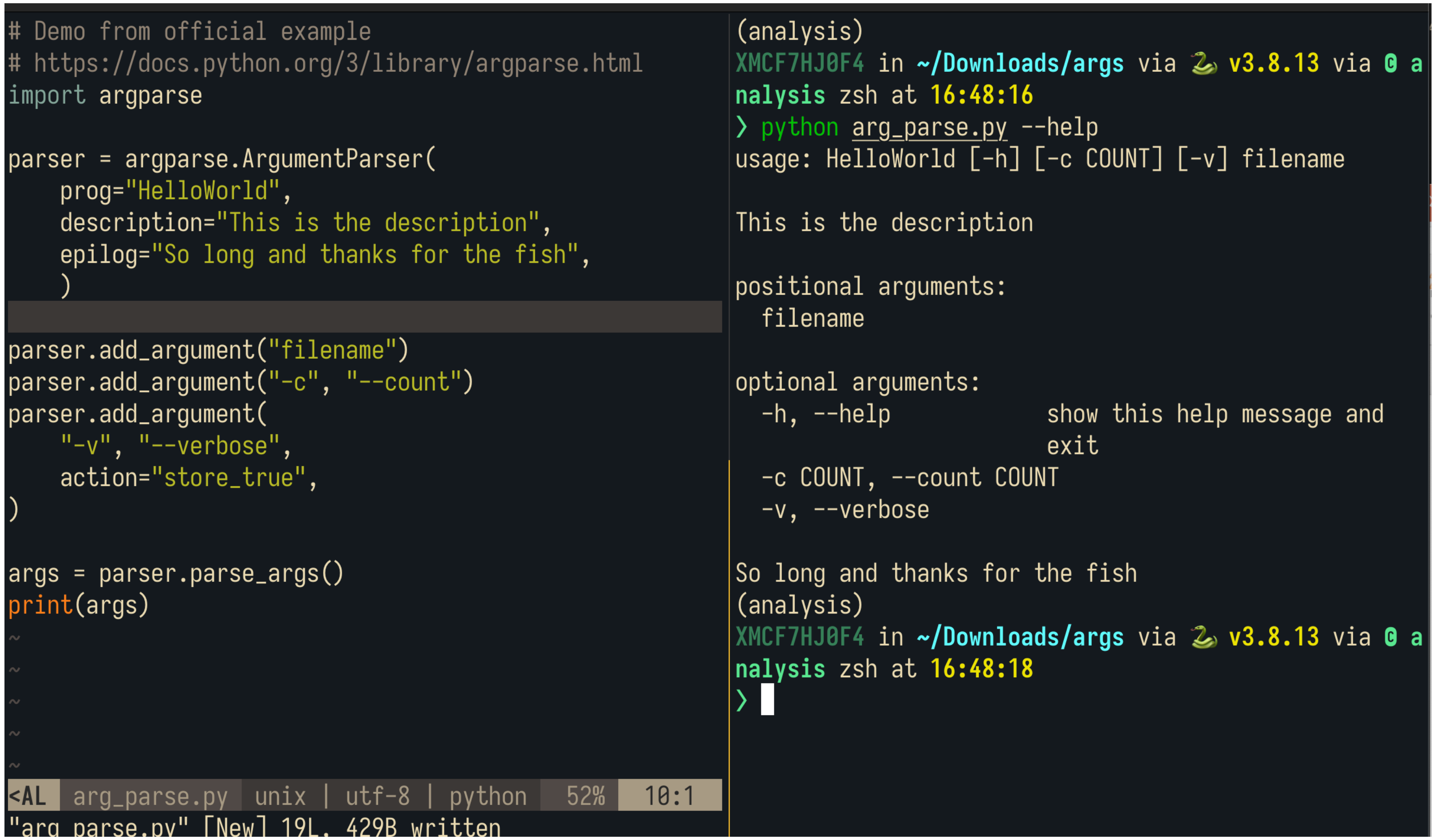


```
❯ Rscript --help
Usage: Rscript [options] file [args]
   or: Rscript [options] -e expr [-e expr2 ...] [args]
A binary front-end to R, for use in scripting applications.

Options:
  --help              Print usage and exit
  --version           Print version and exit
  --verbose           Print information on progress
  --default-packages=LIST  Attach these packages on startup;
                        a comma-separated LIST of package names, or 'NULL'
and options to R (in addition to --no-echo --no-restore), for example:
  --save              Do save workspace at the end of the session
  --no-environ        Don't read the site and user environment files
  --no-site-file      Don't read the site-wide Rprofile
  --no-init-file      Don't read the user R profile
  --restore           Do restore previously saved objects at startup
  --vanilla           Combine --no-save, --no-restore, --no-site-file,
                        --no-init-file and --no-environ

Expressions (one or more '-e <expr>') may be used *instead* of 'file'.
Any additional 'args' can be accessed from R via 'commandArgs(TRUE)'.
See also  ?Rscript  from within R.
```

- main command (binary / script)
- flags
- positional args

Default ways

python argparse



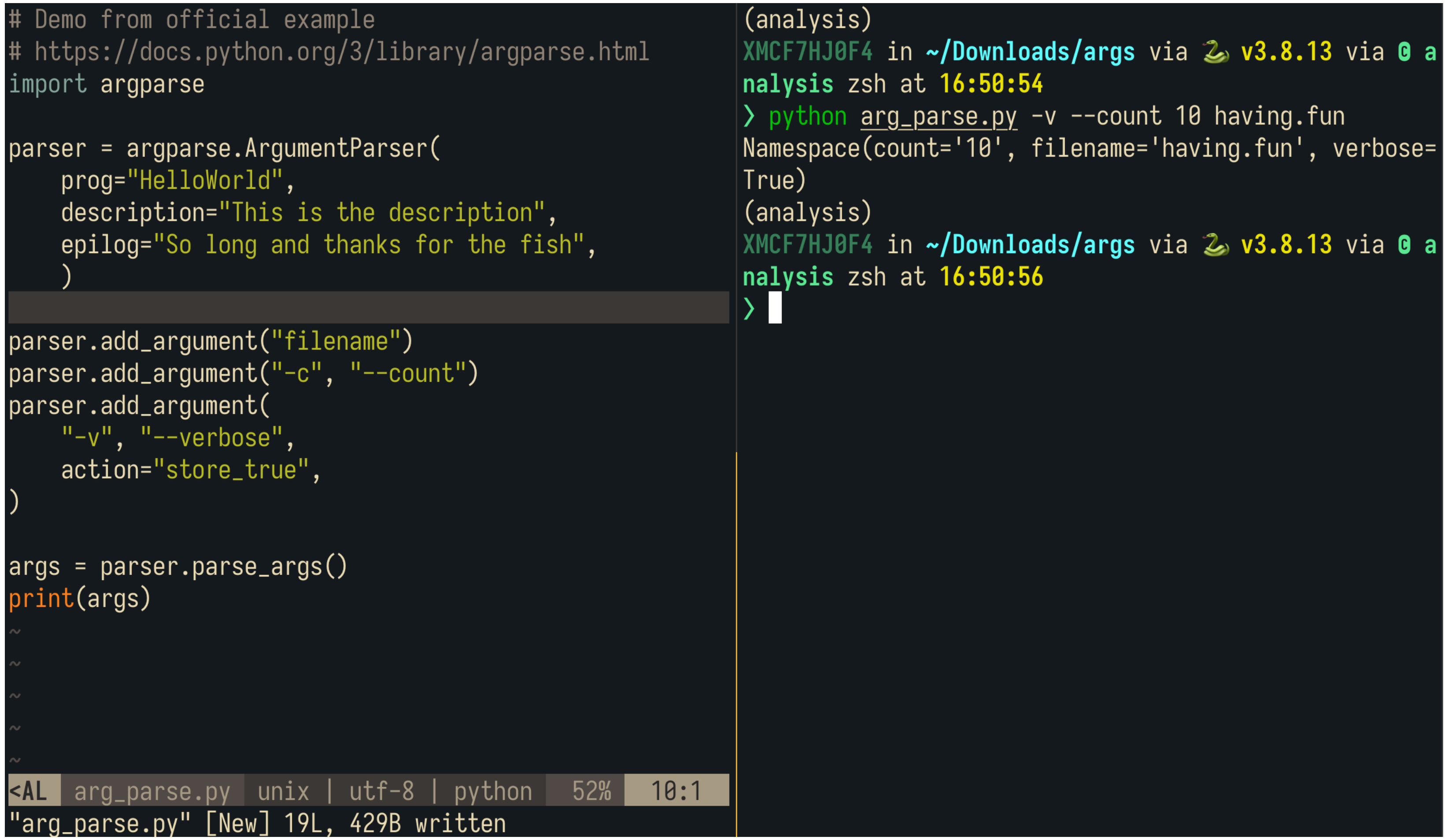
```python
# Demo from official example
# https://docs.python.org/3/library/argparse.html
import argparse

parser = argparse.ArgumentParser(
    prog="HelloWorld",
    description="This is the description",
    epilog="So long and thanks for the fish",
    )

parser.add_argument("filename")
parser.add_argument("-c", "--count")
parser.add_argument(
    "-v", "--verbose",
    action="store_true",
)

args = parser.parse_args()
print(args)
~
~
~
~
~
<AL  arg_parse.py  unix | utf-8 | python   52%   10:1
"arg_parse.py" [New] 19L, 429B written
```

```
(analysis)
XMCF7HJ0F4 in ~/Downloads/args via 🐍 v3.8.13 via 🅒 a
nalysis zsh at 16:48:16
❯ python arg_parse.py --help
usage: HelloWorld [-h] [-c COUNT] [-v] filename

This is the description

positional arguments:
  filename

optional arguments:
  -h, --help            show this help message and
                        exit
  -c COUNT, --count COUNT
  -v, --verbose

So long and thanks for the fish
(analysis)
XMCF7HJ0F4 in ~/Downloads/args via 🐍 v3.8.13 via 🅒 a
nalysis zsh at 16:48:18
❯
```

Basic steps

- create an ArgumentParser
- Add arguments
- Parse args
- Use parsed args as configuration

python argparse



```
# Demo from official example
# https://docs.python.org/3/library/argparse.html
import argparse

parser = argparse.ArgumentParser(
    prog="HelloWorld",
    description="This is the description",
    epilog="So long and thanks for the fish",
    )

parser.add_argument("filename")
parser.add_argument("-c", "--count")
parser.add_argument(
    "-v", "--verbose",
    action="store_true",
)

args = parser.parse_args()
print(args)
~
~
~
~
~
<AL  arg_parse.py  unix | utf-8 | python   52%   10:1
"arg_parse.py" [New] 19L, 429B written
```

```
(analysis)
XMCF7HJ0F4 in ~/Downloads/args via 🐍 v3.8.13 via 🅒 a
nalysis zsh at 16:50:54
❯ python arg_parse.py -v --count 10 having.fun
Namespace(count='10', filename='having.fun', verbose=
True)
(analysis)
XMCF7HJ0F4 in ~/Downloads/args via 🐍 v3.8.13 via 🅒 a
nalysis zsh at 16:50:56
❯ 
```

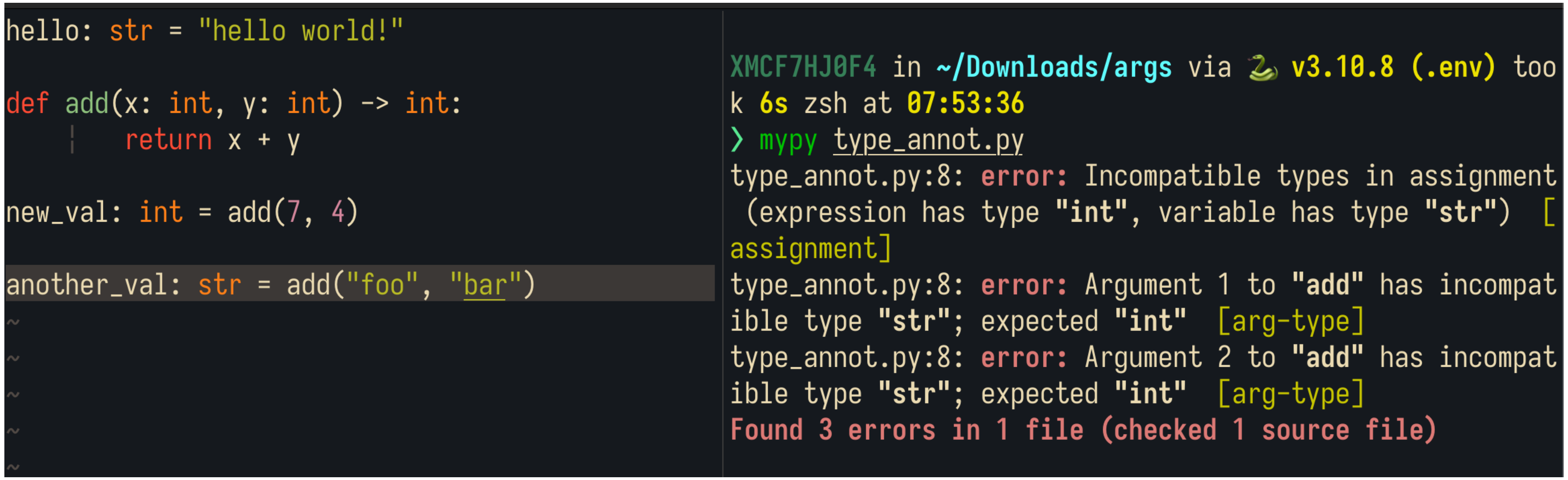
basic steps

- create an ArgumentParser
- Add arguments
- Parse args
- Use parsed args as configuration

- It can be onerous to create lots for arguments
- What if a user specified the wrong argument?
  python add.py --number HELLO_WORLD
- What could be better?

(side note) Type annotation

Type annotation



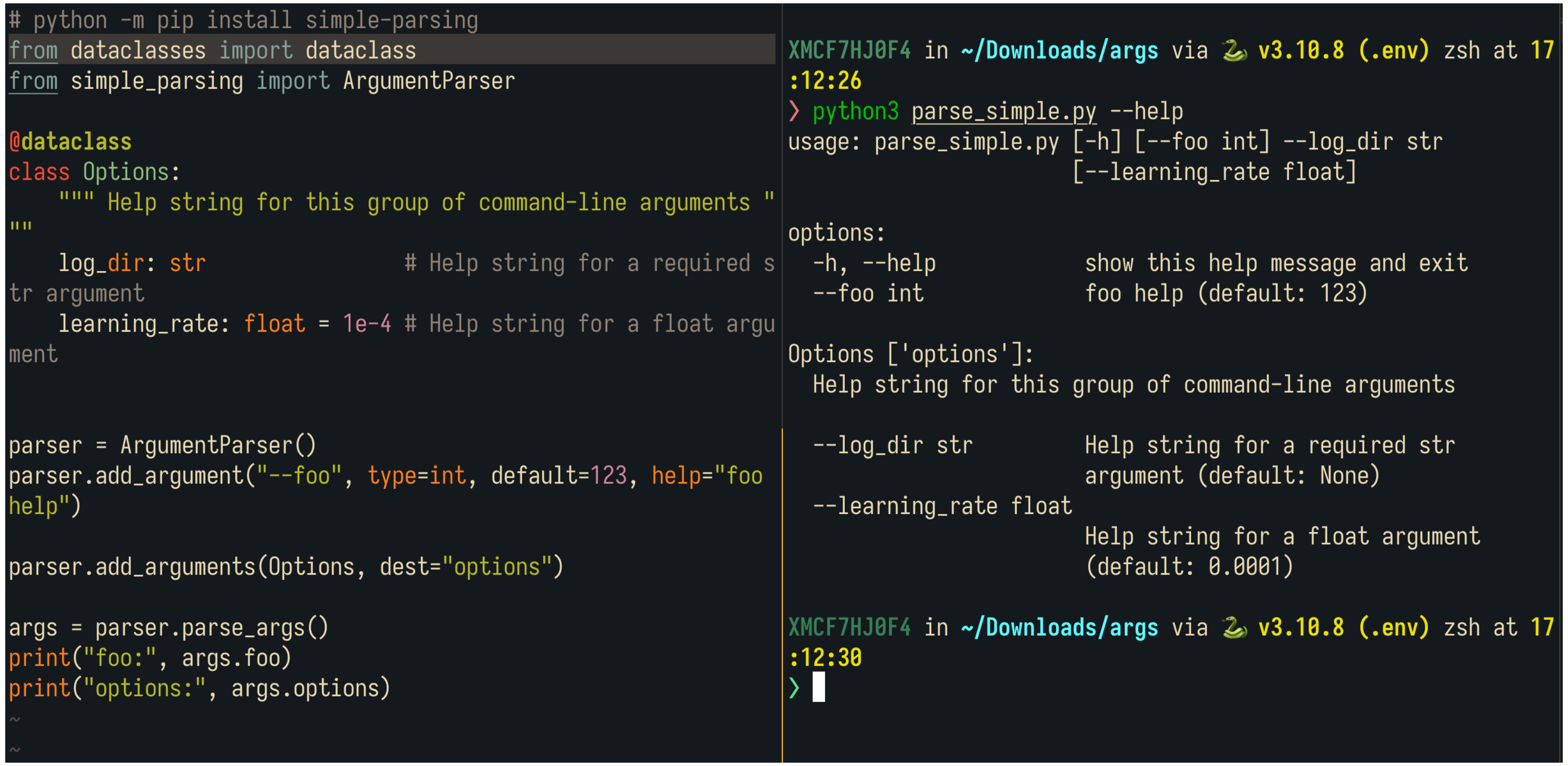
```
hello: str = "hello world!"

def add(x: int, y: int) -> int:
    return x + y

new_val: int = add(7, 4)

another_val: str = add("foo", "bar")
~
~
~
~
~
```

```
XMCF7HJ0F4 in ~/Downloads/args via 🐍 v3.10.8 (.env) too
k 6s zsh at 07:53:36
❯ mypy type_annot.py
type_annot.py:8: error: Incompatible types in assignment
 (expression has type "int", variable has type "str")  [
assignment]
type_annot.py:8: error: Argument 1 to "add" has incompat
ible type "str"; expected "int"  [arg-type]
type_annot.py:8: error: Argument 2 to "add" has incompat
ible type "str"; expected "int"  [arg-type]
Found 3 errors in 1 file (checked 1 source file)
```

- Annotate data types to a dynamic language, without forcing python to become other static typed language
- Type checking does not come by default, but is supported by type checking tools like mypy

Simple parsing

Simple parsing

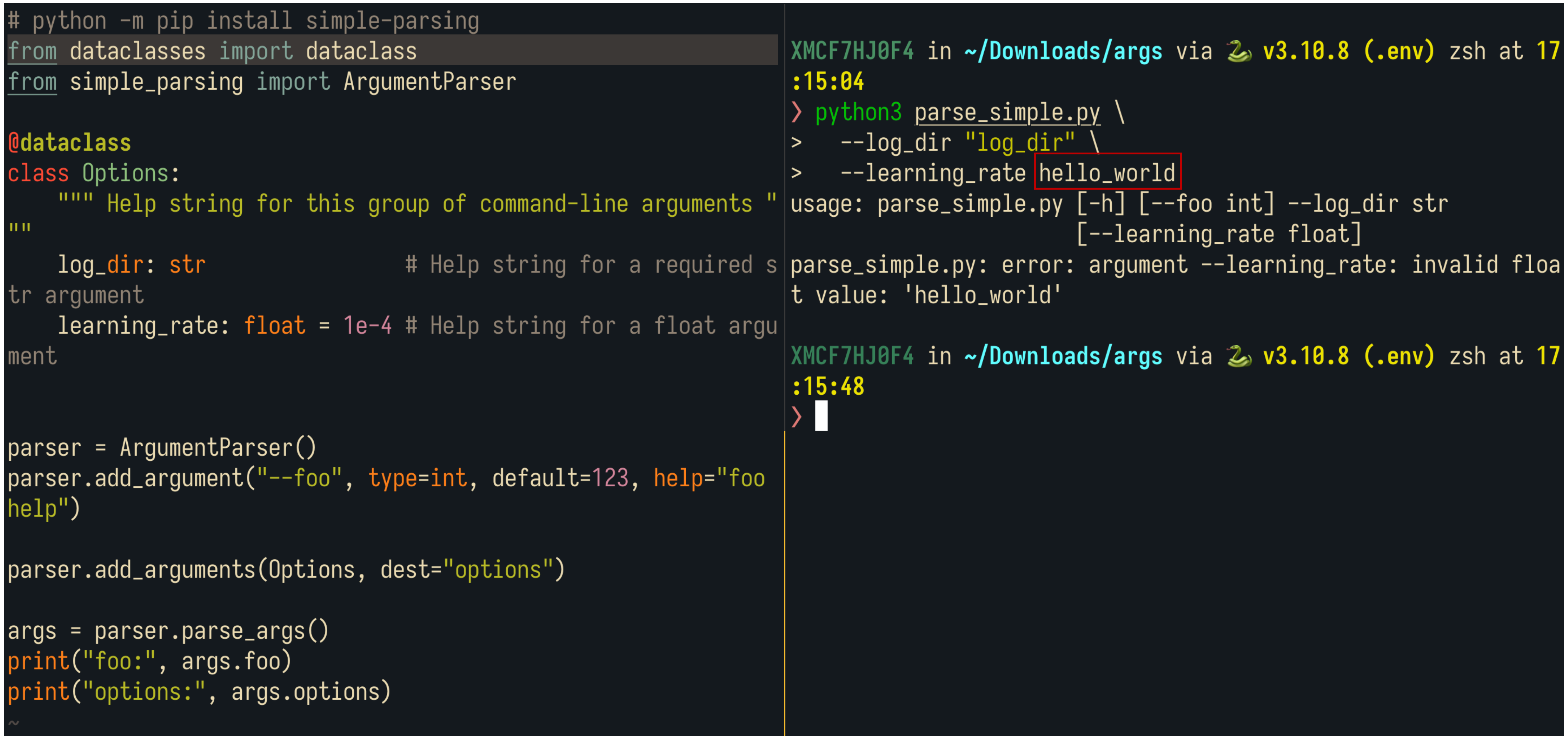


```
# python -m pip install simple-parsing
from dataclasses import dataclass
from simple_parsing import ArgumentParser

@dataclass
class Options:
    """ Help string for this group of command-line arguments "
""
    log_dir: str                # Help string for a required s
tr argument
    learning_rate: float = 1e-4 # Help string for a float argu
ment


parser = ArgumentParser()
parser.add_argument("--foo", type=int, default=123, help="foo
help")

parser.add_arguments(Options, dest="options")

args = parser.parse_args()
print("foo:", args.foo)
print("options:", args.options)
~
~
```

```
XMCF7HJ0F4 in ~/Downloads/args via 🐍 v3.10.8 (.env) zsh at 17
:12:26
❯ python3 parse_simple.py --help
usage: parse_simple.py [-h] [--foo int] --log_dir str
                       [--learning_rate float]

options:
  -h, --help            show this help message and exit
  --foo int             foo help (default: 123)

Options ['options']:
  Help string for this group of command-line arguments

  --log_dir str         Help string for a required str
                        argument (default: None)
  --learning_rate float
                        Help string for a float argument
                        (default: 0.0001)

XMCF7HJ0F4 in ~/Downloads/args via 🐍 v3.10.8 (.env) zsh at 17
:12:30
❯ 
```

- Use python dataclass to predefine options and arguments
- A cleaner and more readable approach
- Options are typed

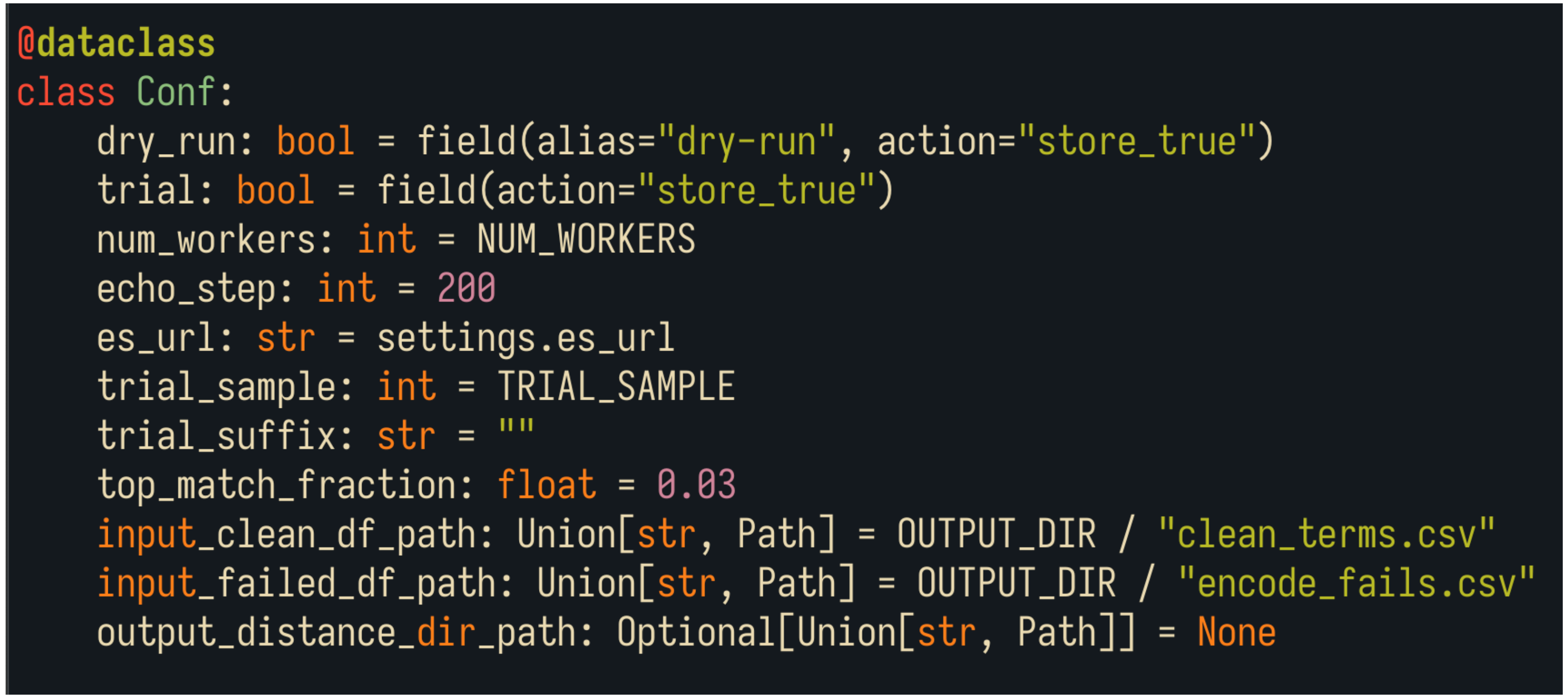
Simple parsing



```
# python -m pip install simple-parsing
from dataclasses import dataclass
from simple_parsing import ArgumentParser

@dataclass
class Options:
    """ Help string for this group of command-line arguments "
""
    log_dir: str                # Help string for a required s
tr argument
    learning_rate: float = 1e-4 # Help string for a float argu
ment


parser = ArgumentParser()
parser.add_argument("--foo", type=int, default=123, help="foo
help")

parser.add_arguments(Options, dest="options")

args = parser.parse_args()
print("foo:", args.foo)
print("options:", args.options)
~
```

```
XMCF7HJ0F4 in ~/Downloads/args via 🐍 v3.10.8 (.env) zsh at 17
:15:04
❯ python3 parse_simple.py \
>   --log_dir "log_dir" \
>   --learning_rate hello_world
usage: parse_simple.py [-h] [--foo int] --log_dir str
                       [--learning_rate float]
parse_simple.py: error: argument --learning_rate: invalid floa
t value: 'hello_world'

XMCF7HJ0F4 in ~/Downloads/args via 🐍 v3.10.8 (.env) zsh at 17
:15:48
❯
```

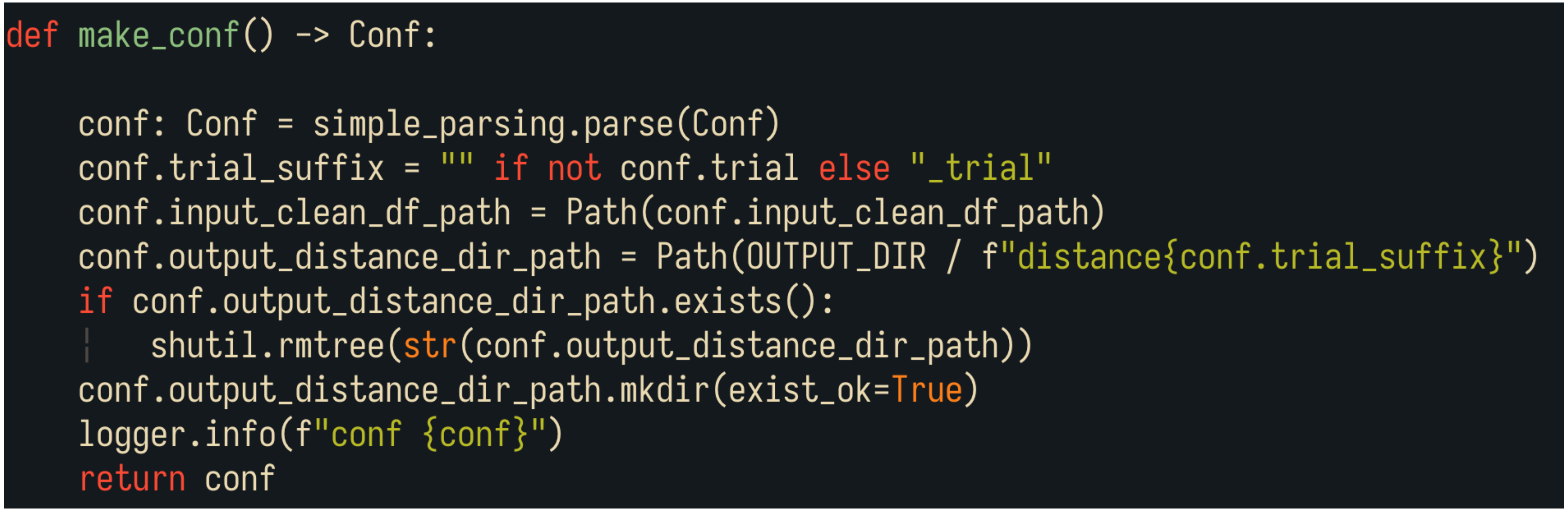
Simple parsing



```python
@dataclass
class Conf:
    dry_run: bool = field(alias="dry-run", action="store_true")
    trial: bool = field(action="store_true")
    num_workers: int = NUM_WORKERS
    echo_step: int = 200
    es_url: str = settings.es_url
    trial_sample: int = TRIAL_SAMPLE
    trial_suffix: str = ""
    top_match_fraction: float = 0.03
    input_clean_df_path: Union[str, Path] = OUTPUT_DIR / "clean_terms.csv"
    input_failed_df_path: Union[str, Path] = OUTPUT_DIR / "encode_fails.csv"
    output_distance_dir_path: Optional[Union[str, Path]] = None
```

- Use case from my recent code

Simple parsing

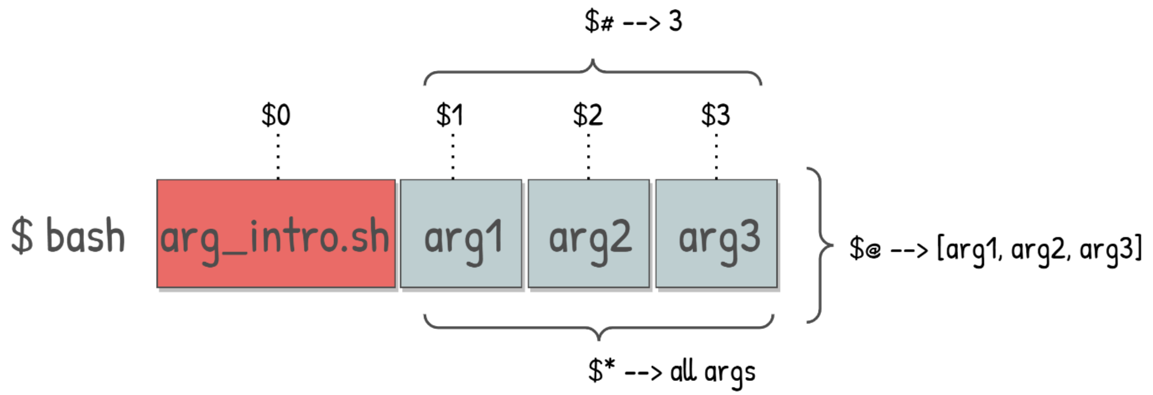


```python
def make_conf() -> Conf:

    conf: Conf = simple_parsing.parse(Conf)
    conf.trial_suffix = "" if not conf.trial else "_trial"
    conf.input_clean_df_path = Path(conf.input_clean_df_path)
    conf.output_distance_dir_path = Path(OUTPUT_DIR / f"distance{conf.trial_suffix}")
    if conf.output_distance_dir_path.exists():
        shutil.rmtree(str(conf.output_distance_dir_path))
    conf.output_distance_dir_path.mkdir(exist_ok=True)
    logger.info(f"conf {conf}")
    return conf
```

- Conf is now a typed dataclass -- Errors with undefined properties will be picked up by the type checker

Command line arguments in other languages

- R
  - default: commandArgs function
  - 3rd library: argparse https://cran.r-project.org/web/packages/argparse/vignettes/argparse.html
- Bash
  - $1, $2, $3, etc.
  - https://stackabuse.com/how-to-parse-command-line-arguments-in-bash/



Summary

- Command line arguments as interface to users
- How to do that
- 3rd party library support