

argparse turns the words typed after your script's name into an object of typed values, and writes the help and error messages for you. Each section starts with a fresh `p = argparse.ArgumentParser()`; passing a list to `p.parse_args([...])` lets you try it without a terminal.

THE SKELETON

```
p = argparse.ArgumentParser(
    description="Resize images.")
the text shown at the top of -h

p.add_argument("path")
positional: required, matched by order

p.add_argument("out", nargs="?",
    default="out.png")
a positional you may leave out

p.add_argument("--size", type=int,
    default=800)
optional: starts with --

args = p.parse_args(["cat.png"])
vars(args)
→ {'path': 'cat.png', 'out': 'out.png',
  'size': 800}
vars() shows every value as a dict

args.path                → 'cat.png'
values are attributes

p.parse_args(["a", "--size", "9"]).size
→ 9
type=int converted the text

p.parse_args(["a", "--size=9"]).size → 9
--size=9 works too

args = p.parse_args()
in a real script: reads sys.argv[1:]
```

ON/OFF FLAGS

```
p.add_argument("-v", "--verbose",
    action="store_true")
False unless the flag is given

p.parse_args(["-v"]).verbose → True
-v and --verbose both work

p.parse_args([]).verbose → False

p.add_argument("--dry-run",
    action="store_true")

p.parse_args(["--dry-run"]).dry_run
→ True
dashes become underscores

p.add_argument("--no-cache",
    dest="cache", action="store_false")
dest= picks the attribute name

p.parse_args([]).cache → True
store_false: True unless the flag is given

p.add_argument("--color", default=True,
    action=BooleanOptionalAction)
gives --color and --no-color (from argparse)

p.parse_args(["--no-color"]).color
→ False
```

VALUES

```
p.add_argument("--ratio", type=float)
p.parse_args(["--ratio", "0.5"]).ratio
→ 0.5

p.parse_args([]).ratio → None
an optional with no default is None

p.add_argument("--out", type=Path)
type= takes any function of one string

p.parse_args(["--out", "res"]).out.name
→ 'res'

p.add_argument("--fmt", default="json",
    choices=["json", "csv"])
anything else is refused

p.parse_args(["--fmt", "csv"]).fmt
→ 'csv'

try:
    p.parse_args(["--fmt", "xml"])
except SystemExit as e:
    code = e.code
code
→ 2
bad input: message on stderr, then exit code 2

p.add_argument("--token", required=True)
a flag that must be given
```

SEVERAL VALUES

```
nargs: "?" zero or one, "*" zero or more, "+" one or more

p.add_argument("-p", nargs=2, type=int)
exactly 2, e.g. a point

p.parse_args(["-p", "8", "6"]).p → [8, 6]

p.add_argument("--tag", action="append")
repeat the flag to collect values

p.parse_args(["--tag", "a",
    "--tag", "b"]).tag
→ ['a', 'b']

p.add_argument("-q", action="count",
    default=0)

p.parse_args(["-qqq"]).q → 3
count: -q is 1, -qqq is 3

p.add_argument("files", nargs="+")
one or more, as a list

p.parse_args(["a.txt", "b.txt"]).files
→ ['a.txt', 'b.txt']
```

CHECK A VALUE YOURSELF

```
def positive(text):
    n = int(text)
    if n > 0:
        return n
    raise ArgumentError("need > 0")
raise ArgumentError for a clean message

p.add_argument("--jobs", type=positive)

p.parse_args(["--jobs", "4"]).jobs → 4

tool --jobs 0
error: argument --jobs: need > 0
```

SUBCOMMANDS (GIT STYLE)

```
sub = p.add_subparsers(dest="cmd",
    required=True)

dest stores which one ran

add = sub.add_parser("add")
add.add_argument("item")

rm = sub.add_parser("rm")
rm.add_argument("item")
rm.add_argument("--force",
    action="store_true")

a = p.parse_args(["rm", "x", "--force"])
a.cmd, a.item, a.force
→ ('rm', 'x', True)
each command has its own arguments

add.set_defaults(func=do_add)
then call args.func(args)
```

HELP AND USAGE

```
g = p.add_mutually_exclusive_group()
g.add_argument("--json",
    action="store_true")
g.add_argument("--csv",
    action="store_true")
one or the other, never both

p.format_usage()
→ 'usage: tool [-h] [--json | --csv]\n'
written for you from the arguments

p.add_argument("--out", metavar="FILE",
    help="where to save")
help text, and the name shown in -h

p.add_argument("--version",
    action="version", version="1.0")
prints 1.0 and exits

-h, --help
added for you: prints the help and exits

ArgumentParser(epilog="e.g. tool a.png")
text at the end of the help

p.parse_known_args(["--json", "--x"])[1]
→ ['--x']
unknown words come back instead of an error
```

WATCH OUT

```
p.add_argument("--count")

p.parse_args(["--count", "3"]).count
→ '3'
text, not a number: add type=int

p.add_argument("--debug", type=bool)

p.parse_args(["--debug", "False"]).debug
→ True
bool("False") is True: use action="store_true"

p.add_argument("-I", action="append",
    default=["."])

p.parse_args(["-I", "src"]).I
→ ['.', 'src']
append adds to the default, it doesn't replace it

args.dry-run
that's args.dry minus run: write args.dry_run

p.parse_args() # in a notebook or test
reads the wrong argv: pass a list
```