



**Christophe Demarey**

*Inria* engineer

with the support of Damien Pollet

# What is Clap?

Clap is a library for implementing command line applications.



```
[~/Documents/travail/rmod/2025-ESUG-Gdansk> ./hello.sh ESUG  
hello, ESUG.  
~/Documents/travail/rmod/2025-ESUG-Gdansk> |
```



# Anatomy of a command



hello

command

# Anatomy of a command

hello 'Esug'

positional



# Anatomy of a command

hello --shout 'Esug'

flag



# Anatomy of a command

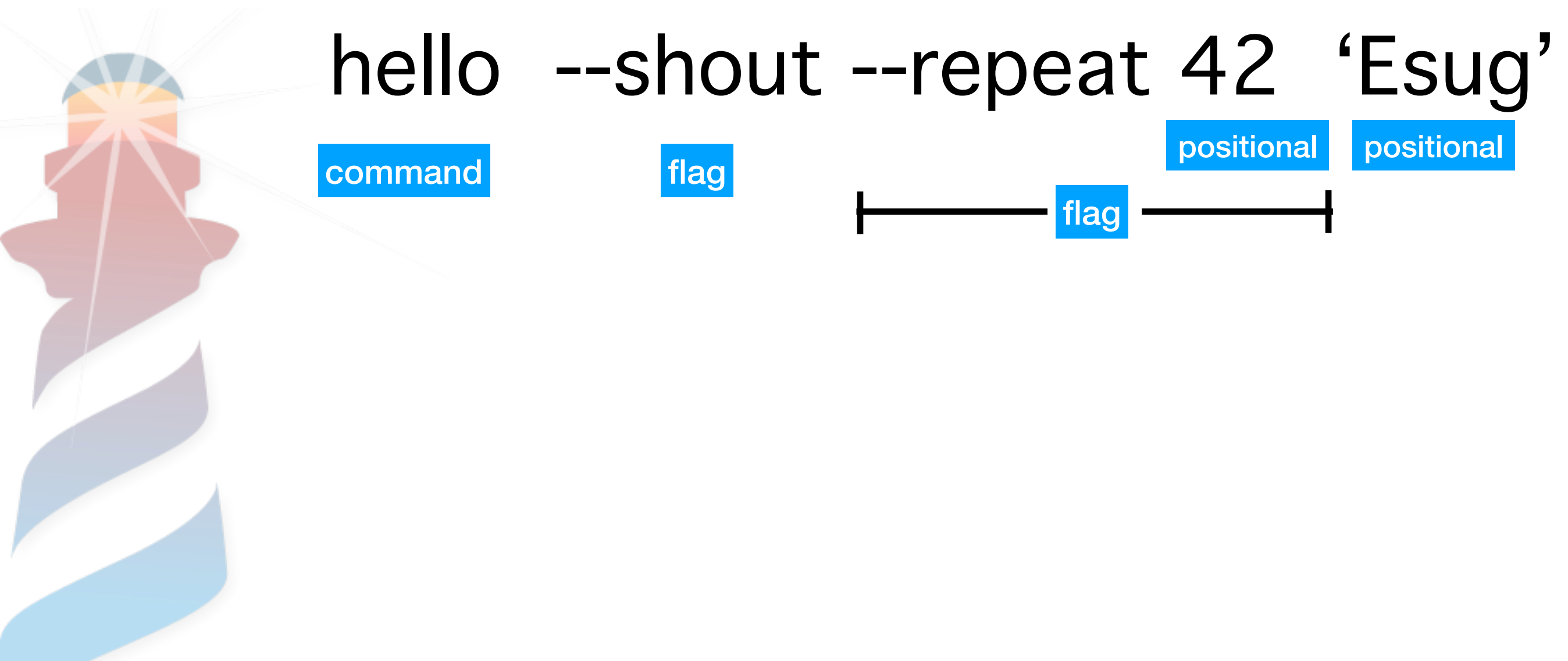


hello --repeat 42 'Esug'

positional

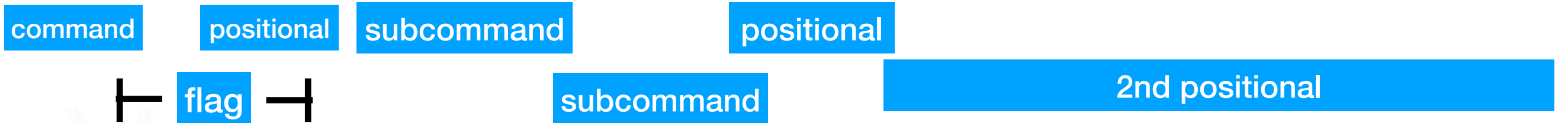
flag

# Anatomy of a command



# Anatomy of a command

git -C dir remote add foo git@mygitserver.com





# Simple example

hello --shout 'Esug'

command

flag

positional



# Simple command description




```
(ClapCommandSpec id: #hello)
  description: 'Provides greetings';
  commandClass: self;
  addHelp;
  addFlag: #shout description: 'Greet loudly';
  addPositional: #who spec: [ :positional |
    positional
      description: 'Recipient of the greetings';
      multiple: true;
      defaultValue: [ :arg :app |
        { app defaultRecipient } ] ];
  yourself
```

# A typical Clap command class

## Class Definition

- inherits from Clap Application



```
ClapApplication << #EsugGitCommand  
  slots: {};  
  package: 'A-ESUG-Clap-Demo'
```

# A typical Clap command class

## Class side

- A method returning the command specification
- If root command, annotated with pragma `<commandline>`
- `commandClass`: the clap command class that inherits from `ClapApplication`: `self`
- Instance of the `commandClass` is returned by `ClapContext command`

# A typical Clap command class

## Instance side

- Must define `execute` method
- Define accessors for flags, positional

`execute`

`self sayHello`

`recipients`

`^ self positional: #who`

`isShouting`

`^ self hasFlag: #shout`



# How to run my app?

Create a shell script to wrap the run of Pharo: **myapp.sh**

```
#!/usr/bin/env bash
```

**Find the real path of the current script**

```
# some magic to find out the real location of this script  
dealing with symlinks
```

```
DIR=`readlink "$0"` || DIR="$0";
```

```
DIR=`dirname "$DIR"`;
```

```
"$DIR"/MyApp.app/MacOs/Pharo --headless "$DIR"/MyApp.app/  
Resources/MyApp.image --no-default-preferences clap myapp "$@"
```

# How to run my app?

Create a shell script to wrap the run of Pharo: **myapp.sh**

```
#!/usr/bin/env bash
```

```
# some magic to find out the real location of this script  
dealing with symlinks
```

```
DIR=`readlink "$0"` || DIR="$0";
```

```
DIR=`dirname "$DIR"`;
```

**Do not start with Pharo UI but command-line**

```
"$DIR"/MyApp.app/MacOs/Pharo --headless "$DIR"/MyApp.app/  
Resources/MyApp.image --no-default-preferences clap myapp "$@"
```

# How to run my app?

Create a shell script to wrap the run of Pharo: **myapp.sh**

```
#!/usr/bin/env bash
```

```
# some magic to find out the real location of this script  
dealing with symlinks
```

```
DIR=`readlink "$0"` || DIR="$0";
```


```
DIR=`dirname "$DIR"`;
```

```
"$DIR"/MyApp.app/MacOs/Pharo --headless "$DIR"/MyApp.app/  
Resources/MyApp.image --no-default-preferences clap myapp "$@"
```

**Run image without preferences, triggering clap and run your root command: my app**



# How to run my app?



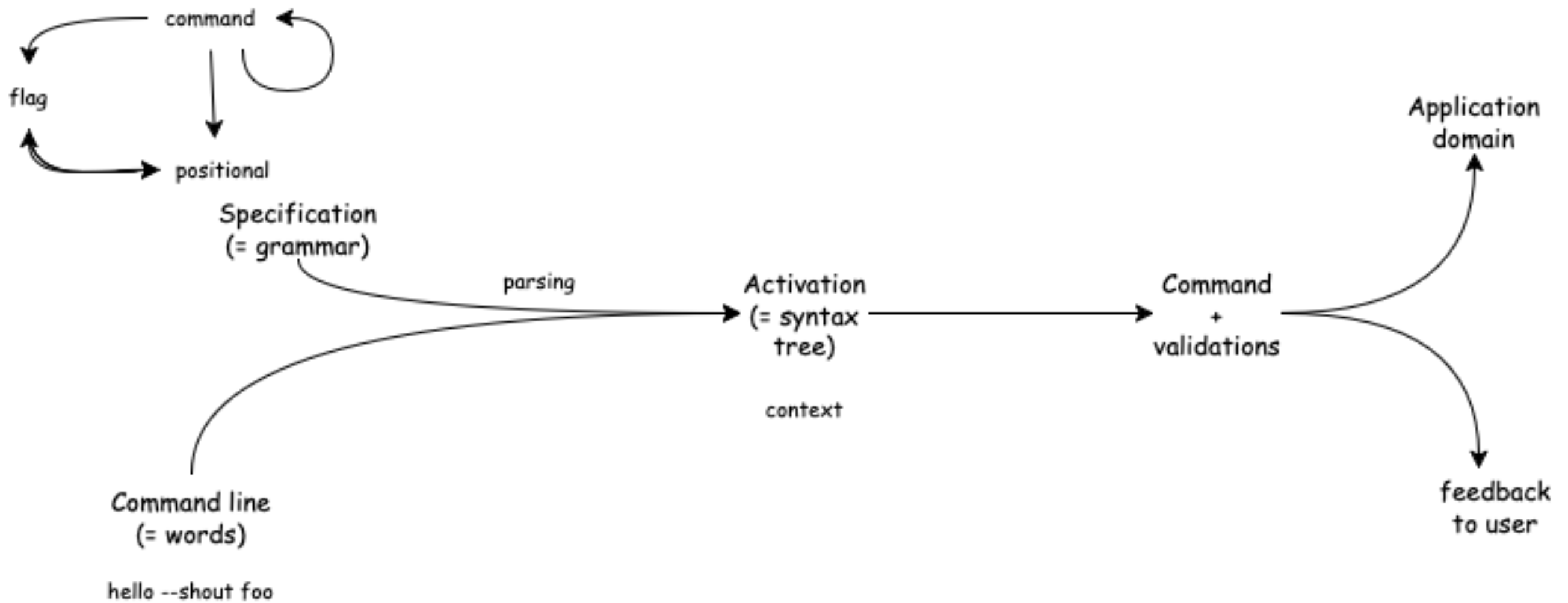
```
2025-ESUG-Gdansk — ~/Documents/travail/rmod/2025-ESUG-Gdansk
[~/Documents/travail/rmod/2025-ESUG-Gdansk> ./hello.sh --help
Provides greetings

Usage: hello [--help] [--whisper] [--shout] [--language <language-value>] [<who>]

Parameters:
  <who>          Recipient of the greetings

Options:
  --help          Prints this documentation
  --whisper       Greet discretely
  --shout         Greet loudly
  --language <language-value>
                  Select language of greeting
[~/Documents/travail/rmod/2025-ESUG-Gdansk> ./hello.sh
hello, world.
[~/Documents/travail/rmod/2025-ESUG-Gdansk> ./hello.sh ESUG
hello, ESUG.
~/Documents/travail/rmod/2025-ESUG-Gdansk> |
```

# Clap flow



# Two kinds of commands

## Root command

- Must be defined with the `<commandline>` pragma
- Callable directly from command-line

## Sub command

- Callable only from its parent command
- May have sub commands



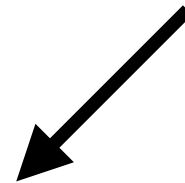
# Demo



© Inria / Photo C. Morel

# Testing the command from Pharo

Get activation = match the command specification  
against the provided command-line (words)



testHelloFrench

```
context := ClapHelloCommandLineExample hello  
activateWith: #('hello' '--language' 'fr').
```

```
self assert: context exitStatus equals: 0.
```


```
self
```

```
assert: self outputString  
equals: 'bonjour, tout le monde.' , self lineEnding
```



# Testing the command from Pharo

Assert command is successful



```
testHelloFrench
```

```
context := ClapHelloCommandLineExample hello  
activateWith: #('hello' '--language' 'fr').
```

```
self assert: context exitStatus equals: 0.
```

```
self
```

```
assert: self outputString  
equals: 'bonjour, tout le monde.' , self lineEnding
```



# Testing the command from Pharo

testHelloFrench

```
context := ClapHelloCommandLineExample hello  
activateWith: #('hello' '--language' 'fr').
```

```
self assert: context exitStatus equals: 0.
```

```
self
```

```
assert: self outputString  
equals: 'bonjour, tout le monde.' , self lineEnding
```

outputString

^ context stdio stdout contents utf8Decoded



# Command-line design

- Split command-line logic from business logic
- Ease maintenance, reusability
- Offer a clear help
- Clear explanation when an error occurs
- Well thought API (like a web service)





# General structure

- Flat structure (ex: curl)

`mycli init`

`mycli build`

`mycli deploy`

- Hierarchical structure (ex: git, pharo-launcher)


`mycli user add`

`mycli user delete`

`mycli config set`

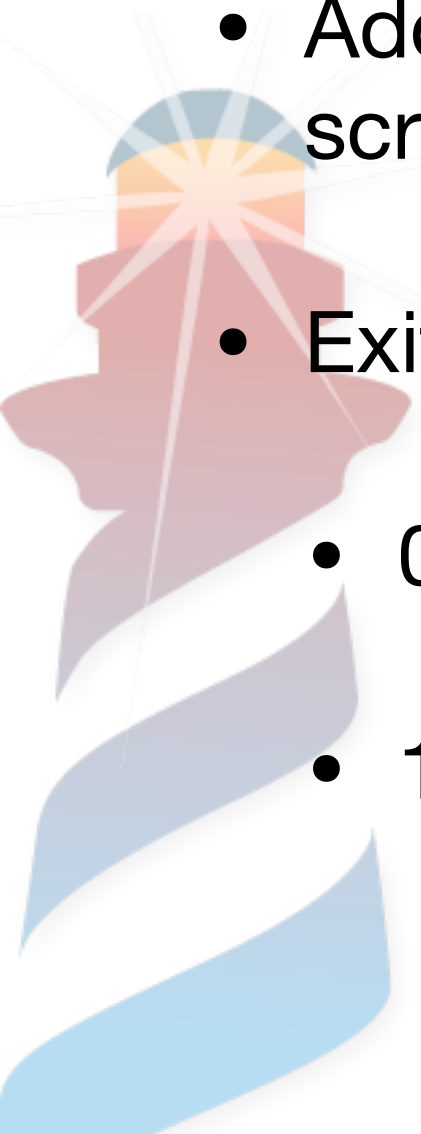


# Positionals Vs Flags

- 
- Positionals: mandatory and unambiguous  
`mycli rename old.txt new.txt`
  - Flags (with positional): optional and with a default value  
`mycli convert file.txt --format pdf`

# Output

- Default: verbose and human readable
- Add a `--json` flag (or equivalent) to use the command in scripts
- Exit with the right return code
  - 0: success
  - 1 or more : error



# Thanks!



**[pharo-contributions/clap-st](#)**