argparse vs docopt vs click vs fire

C'est CLI qui gagne ?

Soirée des communautés rennaises

Jean-Luc Tromparent

Luc Sorel-Giffo

Command Line Interface

- une IHM qui remonte aux années 60
- une IHM de choix pour du tooling IT
- avantages :
  - rapidité d'usage (mémoire musculaire)
  - composabilité (scripting)
  - simplicité de développement

Définition d'une grammaire

```
$ gliss
Usage:
  gliss list
  gliss create <title> [--description=<description>] [--priority=<priority>]
  gliss show <issue-id>
  gliss close <issue-id>
  gliss comment <issue-id> <comment>
  gliss assign <issue-id> <assignee-name>
```

- respect des conventions (POSIX…)
- des arguments et des options logiques et intuitifs
- auto apprentissage avec l'aide intégrée

gliss - another gitlab CLI

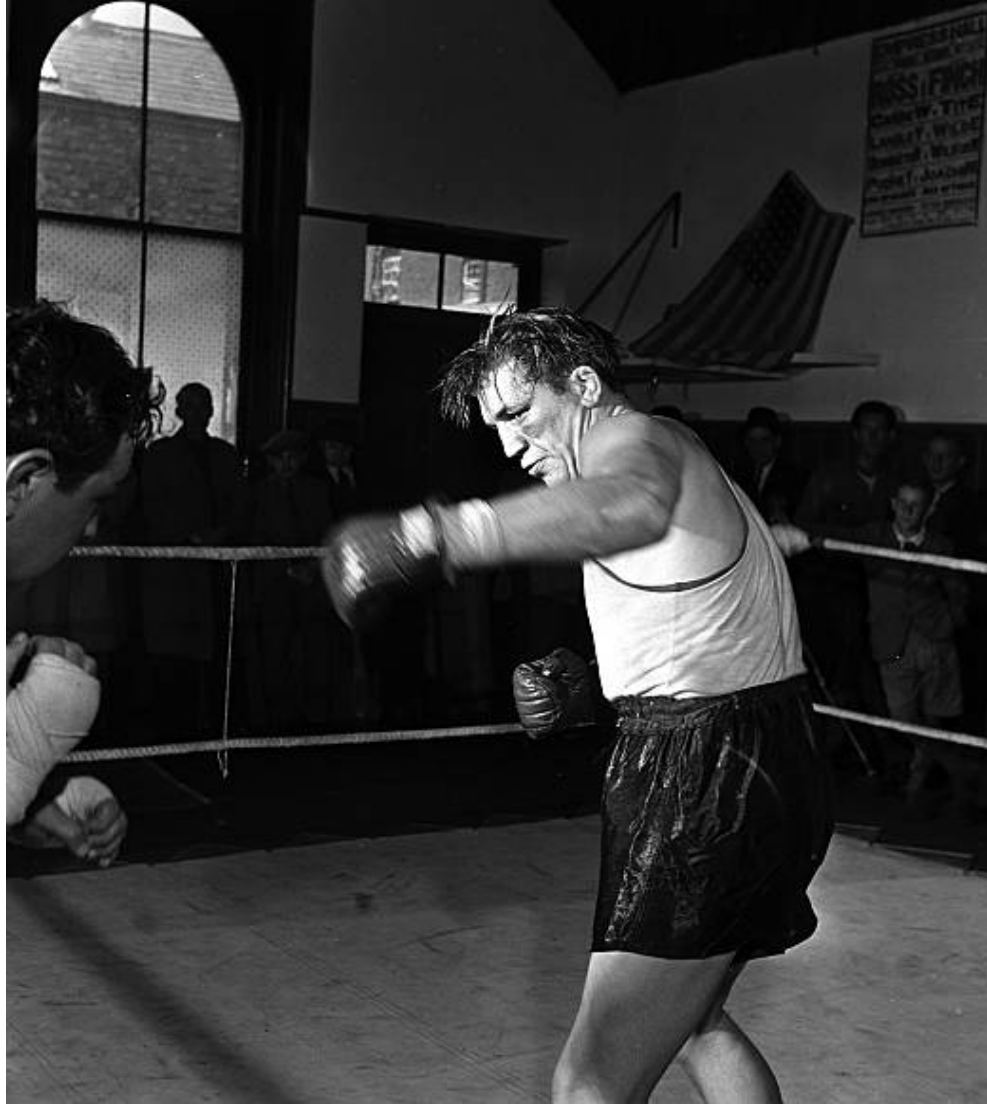
```
$ gliss --help
gliss - Command Line Interface to interact with GitLab Issues.

Usage:
  gliss list
  gliss create <title> [--description=<description>] [--priority=<priority>]
  gliss show <issue-id>
  gliss close <issue-id>
  gliss comment <issue-id> <comment>
  gliss assign <issue-id> <assignee-name>

Options:
  -h --help                          Show this screen.
  -d --description=<description>     Description of the issue.
  -p --priority=<priority>           Priority of the issue.
                                     Valid values are "Low", "Medium", "High".
```

Le vétéran du circuit : **argparse**

- inclus dans la librairie standard depuis python 2.7
- recommandé dans la documentation python (vs getopt et optparse)



gliss-argparse.py

```
import argparse

parser = argparse.ArgumentParser(description='Command Line Interface to interact with GitLab Issues.')

subparsers = parser.add_subparsers(title='Commands', dest='command')

# List issues
list_parser = subparsers.add_parser('list', help='List all issues')

# Create new issue
create_parser = subparsers.add_parser('create', help='Create a new issue')
create_parser.add_argument('title', help='Title of the issue')
create_parser.add_argument('-d', '--description', help='Description of the issue')
create_parser.add_argument('-p', '--priority', choices=['Low', 'Medium', 'High'],
                           default='Low', help='Priority of the issue')

# Show issue details
show_parser = subparsers.add_parser('show', help='Show details of an issue')
show_parser.add_argument('issue_id', type=int, help='ID of the issue')

...

args = parser.parse_args()
print(vars(args))
```

Génération du --help à partir des valeurs des paramètres *help*

```
$ python gliss-argparse.py --help
usage: gliss-argparse.py [-h] {list,create,show,close,comment,assign} ...

Command Line Interface to interact with GitLab Issues.

options:
  -h, --help            show this help message and exit

Commands:
  {list,create,show,close,comment,assign}
    list                List all issues
    create              Create a new issue
    show                Show details of an issue
    close               Close an issue
    comment             Comment on an issue
    assign              Assign an issue to a user
```

Aide détaillée pour chaque commande

```
$ python gliss-argparse.py create --help
usage: gliss-argparse.py create [-h] [-d DESCRIPTION] [-p {Low,Medium,High}]
                                title

positional arguments:
  title                 Title of the issue

optional arguments:
  -h, --help            show this help message and exit
  -d DESCRIPTION, --description DESCRIPTION
                        Description of the issue
  -p {Low,Medium,High}, --priority {Low,Medium,High}
                        Priority of the issue
```

Valeur par défaut

```
$ python gliss-argparse.py create "implement create issue"
{'command': 'create', 'title': 'implement create issue', 'description': None, 'priority': 'Low'}
```

Validation des arguments

```
$ python gliss-argparse.py create "implement create issue" --priority=critical
usage: gliss-argparse.py create [-h] [--description DESCRIPTION] [--priority {Low,Medium,High}] title
gliss-argparse.py create: error: argument --priority: invalid choice: 'critical' (choose from 'Low', 'Medium', 'High')
```

Typage

```
$ python gliss-argparse.py show 1234
{'command': 'show', 'issue_id': 1234}
```

L’ancienne gloire : **docopt**

- proposé à pycon UK 2012

  http://youtu.be/pXhcPJK5cMc
- 7.8k ⭐
- Used by: 86k repos
- approche recommandée par la PEP 257

> *The docstring of a script (a stand-alone program) should be usable as its “usage” message, printed when the script is invoked with incorrect or missing arguments (or perhaps with a “-h” option, for “help”).*

gliss-docopt.py

```
"""gliss - Command Line Interface to interact with GitLab Issues.

Usage:
  gliss list
  gliss create <title> [--description=<description>] [--priority=<priority>]
  gliss show <issue-id>
  gliss close <issue-id>
  gliss comment <issue-id> <comment>
  gliss assign <issue-id> <assignee-name>

Options:
  -h --help                          Show this screen.
  -d --description=<description>     Description of the issue.
  -p --priority=<priority>           Priority of the issue.
                                     Valid values are "Low", "Medium", "High".
"""

import docopt

opts = docopt.docopt(__doc__)
print(opts)
```

Tout le --help mais rien d'autre que le --help

```
$ python gliss-docopt.py --help
gliss - Command Line Interface to interact with GitLab Issues.

Usage:
  gliss list
  gliss create <title> [--description=<description>] [--priority=<priority>]
  gliss show <issue-id>
  gliss close <issue-id>
  gliss comment <issue-id> <comment>
  gliss assign <issue-id> <assignee-name>

Options:
  -h --help                          Show this screen.
  -d --description=<description>     Description of the issue.
  -p --priority=<priority>           Priority of the issue [default: Low].
                                     Valid values are "Low", "Medium", "High".
```

Pas d'aide spécifique par commande

```
$ python gliss-docopt.py create --help
gliss - Command Line Interface to interact with GitLab Issues.

Usage:
  gliss list
  gliss create <title> [--description=<description>] [--priority=<priority>]
  gliss show <issue-id>
  gliss close <issue-id>
  gliss comment <issue-id> <comment>
  gliss assign <issue-id> <assignee-name>

Options:
  -h --help                          Show this screen.
  -d --description=<description>     Description of the issue.
  -p --priority=<priority>           Priority of the issue [default: Low].
                                     Valid values are "Low", "Medium", "High".
```

Les valeurs par défaut sont supportés

```
$ python gliss-docopt.py create "implement create issue"
{'--description': None,
 '--priority': 'Low',
 '<assignee-name>': None,
 '<comment>': None,
 '<issue-id>': None,
 '<title>': 'implement create issue',
 'assign': False,
 'close': False,
 'comment': False,
 'create': True,
 'list': False,
 'show': False}
```

Mais pas de validation des valeurs

```
$ python gliss-docopt.py create toto --priority critical
{'--description': None,
 '--priority': 'critical',
 '<assignee-name>': None,
 '<comment>': None,
 '<issue-id>': None,
 '<title>': 'toto',
 'assign': False,
 'close': False,
 'comment': False,
 'create': True,
 'list': False,
 'show': False}
```

comment ca va doc(opt) ?

- porté dans 23 langages
- le projet n'est plus maintenu depuis 2014 😢
- il existe un fork par la communauté jazzband : docopt-ng

Le favori des bookmakers : **click**

- 13.6k ⭐
- Used by: 1M repos
- créé par Armin Ronacher (flask, jinja, etc.)
- Typer par @tiangolo (fastapi)

gliss-click.py

```
import click

@click.group()
def gliss():
    """Command Line Interface to interact with GitLab Issues."""
    pass

@gliss.command()
def list():
    """List all issues."""
    click.echo("Listing all issues.")

@gliss.command()
@click.argument('title')
@click.option('--description', '-d', help='Description of the issue.')
@click.option('--priority', '-p', type=click.Choice(['Low', 'Medium', 'High']), default='Low',
              help='Priority of the issue.')
def create(title, description, priority):
    """Create a new issue."""
    click.echo(f"Creating issue: {title} {description} {priority}")

...
```

Génération du --help

```
$ python gliss-click.py --help
Usage: gliss-click.py [OPTIONS] COMMAND [ARGS]...

  Command Line Interface to interact with GitLab Issues.

Options:
  --help  Show this message and exit.

Commands:
  assign   Assign an issue to an assignee.
  close    Close an issue.
  comment  Comment on an issue.
  create   Create a new issue.
  list     List all issues.
  show     Show details of an issue.
```

Aide détaillée sur une commande

```
$ python gliss-click.py create --help
Usage: gliss-click.py create [OPTIONS] TITLE

  Create a new issue.

Options:
  -d, --description TEXT          Description of the issue.
  -p, --priority [Low|Medium|High]
                                  Priority of the issue.
  --help                          Show this message and exit.
```

L’outsider : **fire**

- 24.2k ⭐
- Used by: 18k repos
- porté par google

gliss-fire.py

```python
from fire import Fire

class Gliss:
    """Command Line Interface to interact with GitLab Issues."""

    def list(self):
        """List all issues."""
        print("Listing all issues.")

    def create(self, title, description=None, priority=None):
        """Create a new issue."""
        print(f"Creating issue: {title}")
        if description:
            print(f"Description: {description}")
        if priority:
            print(f"Priority: {priority}")

    (...)

if __name__ == '__main__':
    Fire(Gliss)
```

```
$ python gliss-fire.py
NAME
    gliss-fire.py - Command Line Interface to interact with GitLab Issues.

SYNOPSIS
    gliss-fire.py COMMAND

DESCRIPTION
    Command Line Interface to interact with GitLab Issues.

COMMANDS
    COMMAND is one of the following:

     assign
       Assign an issue to an assignee.

     close
       Close an issue.

     comment
       Comment on an issue.

     create
       Create a new issue.

     list
       List all issues.
```

```
$ python gliss-fire.py create --help
NAME
    gliss-fire.py create - Create a new issue.

SYNOPSIS
    gliss-fire.py create TITLE <flags>

DESCRIPTION
    Create a new issue.

POSITIONAL ARGUMENTS
    TITLE

FLAGS
    -d, --description=DESCRIPTION
        Type: Optional[]
        Default: None
    -p, --priority=PRIORITY
        Type: Optional[]
        Default: None

NOTES
    You can also use flags syntax for POSITIONAL ARGUMENTS
```

Conclusion : une diversité d’approches

	argparse	**docopt**	**click**	**fire**
approche	API design de CLI	regexp sur docstring	décorateur sur code métier	inspection du code métier
design	orienté-CLI (-> nestable parser)	orienté-CLI (-> aide)	orienté-métier	orienté-métier (nested objects)
typage des valeurs	API du parser	str ou bool	API des décorateurs	heuristique interne (annotations ignorées 😕)
verbosité et **couplage**	parser verbeux +/- couplé aux fonctions métier	verbosité fonction de la docstring (syntaxe élaborée)	couplage des fonctions métier aux décorateurs empilés	0 boilerplate, le code est métier-first (avec des hacks pour la CLI)
documentation	champs *description* ou *help*	la docstring source	champs *help* et docstring de fonction	docstring et signature de fonctions / méthodes

Merci !

https://tinyurl.com/slack-pythonrennes

https://jiel.github.io/c-est-cli-qui-gagne