Interface en ligne de commande (CLI)

UE12 P24 - Python

CLI vs *GUI*

2 façons d'interagir avec un programme :

- Command-Line Interface (CLI) : constant dans le temps, précis, facilement automatisable
- Graphical User Interface (GUI) : plus riche, plus simple d'utilisation

Shells

En général le premier programme qu'on lance dans un terminal est un shell, c'est l'interface utilisateur du système d'exploitation :

- Windows : PowerShell
- Unix : bash, sh, zsh, fish…

Le travail principal d'un shell est de permettre de lancer d'autres programmes. Il aide avec l'auto-complétion (Tab), l'historique (↑, ↓, ctrl + R), les variables…

CLI

Trouver de l’aide :

Windows :

```
1 help ls
```

Unix :

```
1 man ls
2 info ls
3 help cd
```

Partout :

```
1 git --help
2 git -h
```

CLI et paramètres

Les paramètres de lignes de commande ont souvent 2 formes :

- une forme explicite avec `--`, par exemple `--help`
- une forme raccourcie avec `-`, par exemple `-h`

CLI et paramètres - Documentation

- des crochets `[...]` signifient qu'un paramètre est optionnel
- un `|` signifie que 2 formes du paramètre sont possibles

Par exemple `git [-v | --version] [-h | --help]` signifie que ces commandes sont valables :

- `git -v`
- `git --version`
- `git -h`
- `git --version -h`

CLI et paramètres

Lorsque vous tapez

```
1 git --help
```

ou

```
1 git commit -m "Some helpful commit message"
```

ou

```
1 echo $PATH # Sous PowerShell : echo $env:PATH
```

votre shell :

- substitue les variables
- découpe la ligne
- lance le bon programme avec ses paramètres

CLI et paramètres - Exemple 1

```
1 git --help
```

Votre shell :

- découpe cette ligne en 2 (`git` et `--help`)
- trouve le programme `git` (à l'aide de la variable `$PATH`)
- lance le programme `git` avec le paramètre `--help`

Ce que fait le paramètre `--help` aura été défini au préalable par les développeurs de `git`.

CLI et paramètres - Exemple 2

```
1  git commit -m "Some helpful commit message"
```

Votre shell :

- découpe cette ligne en 4 (`git`, `commit`, `-m`, `Some helpful commit message`)
- trouve le programme `git` (à l'aide de la variable `$PATH`)
- lance le programme `git` avec ses 3 paramètres

CLI et paramètres - Exemple 3

Windows :

```
1  echo $env:PATH
```

Unix :

```
1  echo $PATH
```

Votre shell :

- substitue la variable PATH par sa valeur
- découpe cette ligne en 2 (`echo`, la valeur de PATH)
- trouve le programme `echo`
- lance le programme `echo` (dont le but est d'afficher son paramètre, comme `print` en Python) avec son paramètre

CLI et paramètres

Lorsque votre programme est lancé, le système d'exploitation va lui transmettre tous les paramètres spécifiés par l'utilisateur.

C'est ensuite à vous de décider de ce que vous faites de ces paramètres. Par exemple si vous voulez supporter le paramètre `--help`, il faut écrire du code qui vérifie s'il est présent dans la liste de paramètres envoyée, puis qui imprime de l'aide le cas échéant.

CLI et paramètres - Tests

En Python, les paramètres envoyés au programme sont stockés dans `sys.argv`. Vous pouvez-donc faire des tests avec ce programme :

print-params.py

```python
import sys

print(sys.argv)
```

> ⚠ **Warning**
>
> N'utilisez jamais `sys.argv` en dehors de tests. Dans un projet réel, il est beaucoup plus simple d'utiliser la classe `ArgumentParser` du module `argparse`

Variables d’environnement

Votre shell est un programme avec des variables, exactement comme votre programme Python.

Vous pouvez lister ses variables d’environnement :

Windows :

```
1 ls env:
```

Vous pouvez les modifier dans vos paramètres

Unix :

```
1 printenv
```

Vous pouvez les modifier dans vos fichiers `~/.bashrc` et `~/.profile` par exemple

Variable d’environnement PATH

La plus célèbre et la seule que vous devez connaître est la variable `PATH`.

Son rôle est de recenser tous les dossiers où se trouvent des programmes.

Lorsque vous tapez une commande (`git`, `python`, `code`…), votre shell va parcourir ces dossiers un par un jusqu’à trouver le programme.

Variable d’environnement PATH

Lorsque vous installez un nouveau programme dans un nouveau dossier, il faut donc modifier votre variable PATH. Sinon votre shell ne le trouvera pas.

> **Note**
>
> Dans Linux, tous les programmes sont installés dans des dossiers standards (comme `/usr/bin`), il est donc plus rare de modifier PATH.

|

Un shell devient très puissant grâce à l'opérateur pipe (|), qui permet d'enchaîner les commandes :

Windows

```
1  ls env: | Select -First 10
```

Unix

```
1  printenv | head -n 10
```

Ici, les sorties des commandes `ls env:` et `printenv` sont envoyées en entrée des commandes `Select -First 10` et `head -n 10`

- Exemple

Par exemple pour séparer la variable PATH, ne garder que les éléments qui contiennent “windows”, puis trier le résultat :

Windows

```
1  $env:PATH -split ';' | Select-String windows | Sort
```

Unix

```
1  echo $PATH | tr ':' '\n' | grep -i Windows | sort
```

> **Tip**
>
> La commande `grep` est très utile : elle filtre les lignes qui contiennent un motif