HMIN113 - Système
Initiation aux scripts Système en Python

-

Pierre Pompidor

Le but de ce TP est de vous initier au **langage Python** qui est un langage de plus en plus utilisé pour écrire des **scripts système** mais aussi pour **étendre des logiciels** notamment en bioinformatique, géomatique, physique...

Dans un premier temps, vous allez écrire un script qui affiche "bonjour" dans le terminal, puis l'améliorer pour qu'il salue ses paramètres.
Puis dans un second temps, vous allez écrire le début d'un "vrai" script qui reconnaîtra les options qui lui sont passées.

Ecrivez un script Python qui affiche "Bonjour !"

La première étape est d'écrire un "Hello world". Saisissez pour cela le code suivant sous votre éditeur préféré (fichier `bonjour_v1.py`) :

```
#!/usr/bin/env python3
print("Bonjour !")
```

Pour l'exécuter **à partir du terminal** n'oubliez pas de lui donner le droit d'exécution, puis :

— si vous avez rajouté le répertoire courant (`.`) dans votre `PATH` : `bonjour_v1.py`
— sinon (quel déhonneur) : `./bonjour_v1.py`

Modifiez ce script pour qu'il affiche : "Bonjour [le premier paramètre du script] !"

Maintenant, passons un paramètre au script : `bonjour_v2.py VotrePrenom`
Ce paramètre sera stocké dans un tableau de paramètres appelé `argv` (accessible à condition que le module `sys` soit inclus), et dont le premier élément (c'est à dire le premier vrai paramètre passé au script) est appelé `sys.argv[1]` (`sys.argv[0]` étant le nom du script).

```
#!/usr/bin/env python3
import sys
print("Bonjour ", sys.argv[1], "!")
```

Maintenant essayez d'afficher la valeur de tous les paramètres su script :

Pour contrôler la présence de paramètres :
→ testez le nombre d'éléments d'une liste par la fonction `len(liste)`

Pour faire afficher une liste quelconque de paramètres :
→ mise en place d'une boucle automatique par

```
for parametre in sys.argv :
    # instructions à exécuter ...
```

(la variable `parametre` étant instanciée par chaque élément de la liste `sys.argv`).

Et puis commençont les choses sérieuses :
écrivez le script analyse_options.py qui affiche deux messages différents suivant les options qui lui sont passées (a ou p).

(Attention cela suppose un début de maîtrise de l'application d'expressions régulières en Python (voir plus bas)...)

Exemples d'appel du script : `analyse_options.py -a`
ou `analyse_options.py -p`
ou `analyse_options.py -ap`
ou ...

Algorithme :

```
Si la chaîne de lettres options est constituée des lettres 'a' ou/et 'p' :
       Si la chaîne de lettres options contient l'option 'a' :
          Affichez ''Option a !''
       FinSi
       Si la chaîne de lettres options contient l'option 'p' :
          Affichez ''Option p !'
       FinSi
Finsi
Sinon Affichez un message d'alerte : les options utilisées sont incorrectes
```

Mise en place d'expressions régulières en Python :

Le fonction `search` du module `re` (qui doit être inclus) va contrôler la présence du motif dans la chaîne de caractères qui lui est passée en argument (via une variable).

```
resultat = re.search("motif à chercher", variable)
if resultat :
    # instructions à exécuter
    # si le motif a été retrouvé dans la variable
```