Python 9 – Fonctions et Variables

I - Fonctions, paramètres, arguments

1-RAPPELS

→ On déclare avec **def** et on répond avec **return**.

→ Pour savoir où les variables sont stockées, il faut comparer l'appel et la déclaration :

```
01 def multiplier(a,b) :    → déclaration
02    return a*b
03
04 x = multiplier(5, 10)      → appel
05 y  = multiplier(2, 8)      → appel
```

En comparant L04 et L01, on voit que 5 sera stockée dans a et 10 dans b.

→ Séquences des lignes suivies par l'interpréteur sur l'exemple :

01 (déc.) - 04 (appel a=5 b=10) – 01 – 02 – 04(retour x = 50)
05 (appel a=2 b=8) – 01 – 02 – 05(retour y=16)

2 - VOCABULAIRE

→ Les informations envoyées à la fonction lors d'un Appel se nomment les **Arguments**. (exemple 5, 10, 2, 8).

→ Les variables prévues pour stocker les arguments (**a, b**) sont nommées les **Paramètres** de la fonction. Elles se trouvent dans le prototype de la fonction (celle avec def).

II - Documenter une fonction

A - Commenter une fonction avec # permet d'expliquer le fonctionnement du code à un autre développeur qui voudrait la modifier.

B - Documenter une fonction veut dire qu'on explique comment l'utiliser :
→ comment elle se nomme
→ ce qu'elle fait
→ ce qu'elle attend en entrée (paramètres)
→ ce qu'elle renvoie en sortie (la réponse).
La documentation DOCSTRING est un string commençant par """ et finissant par """ qu'on place juste sous le prototype de la fonction.

C - DOCUMENTATION rapide : prototype + phrase

```
1    def addition(nbr1:int, nbr2:int) -> int:
 2       """Renvoie la somme de nbr1 et nbr2"""
3        return nbr1 + nbr2
```

D - DOCUMENTATION LONGUE : prototype sans type + plusieurs phrases

Elle permet de fournir plus d'informations mais elle est plus lourde à lire.

Aucune codification particulière, il suffit d'être clair et cohérent sur l'ensemble des documentations fournies. Voir le cours pour des exemples.

E – help()

La fonction **help(nom_fonction)** permet de récupérer la documentation.

III - Fonction sans retour

Une fonction qui ne renvoie rien se nomme une procédure.

En Python, les procédures n'existent pas vraiment : toute fonction renvoie au moins **None**, un mot-clé signifiant « J'ai fini mais je n'ai rien à donner ».

```
def addition(a:int, b:int) -> None:
    print(a+b)
    return None
    x = addition(5, 10)
```

Ici, **x** contiendra **None** : La console affiche 15 mais la fonction ne répond rien. C'est comme si vous aviez placé return None après la dernière ligne de votre déclaration.

ATTENTION : print et return sont DIFFERENTS.

→ **`print()`** est une fonction qui provoque un affichage et ne renvoie rien (donc None)

→ **return** est un mot-clé qui renvoie ce qu'on demande à l'endroit où l'appel à la fonction a été provoqué.