Python : fiche ressource 1

1 - Notion de variable

Un ordinateur n'est rien d'autre qu'une calculatrice surpuissante. En Anglais, un ordinateur se dit « computer », ce qui signifie littéralement « calculateur », ou « calculatrice ». La souris et le clavier entrent une information. Cette information est stockée quelque part afin que l'ordinateur calcule ce qu'il doit faire ensuite. La zone de la mémoire où cette information est stockée est appelée **variable**. Nous définissons cette variable par un **nom**, l'ordinateur la définit par une **zone** particulière de sa **mémoire**.
En Python, on déclare une variable grâce au signe « = ».

```
ma_variable=2
```

Par cette action, nous demandons à l'ordinateur d'enregistrer dans une zone précise de sa mémoire le nombre entier 2. L'ordinateur retient donc cette zone, et sait que lorsque l'on écrira « ma_variable », il devra aller chercher la valeur correspondante à cette endroit.

2 - Les différentes opérations

Lorsque les variables sont des nombres, il est possible de les additionner, soustraire, multiplier ou diviser, comme des nombres mathématiques. Chaque opération a un symbole qui lui est propre.

Opération mathématique	**Équivalent Python**
Addition	+
Soustraction	-
Multiplication	*
Division	/
Quotient de la division euclidienne	//
Reste de la division euclidienne	%
Puissance	**

3 - Les différents objets

Il existe en Python plusieurs types d'objets différents. Les types les plus basiques sont résumés dans ce tableau.

Type de variable	Syntaxe	Intérêt	Erreur récurrente
Nombre entier	4	Se repérer dans une liste ou une boucle	confusion nombre entier - nombre flottant
Nombre flottant	4.2	Similaire à un nombre décimal	Idem
Chaîne de caractères	'exemple' ou "exemple"	Traiter un texte	Un espace compte comme 1 caractère
Liste	[1,4,8,2] ['c','exemple']	Impossible à résumer dans une si petite case	Copier avec « = » plutôt que deepcopy()
Fonction	def fonction(argument) : … return(resultat)	Transforme un argument en lui appliquant un bloc d'instructions	Oubli de « : » variable - Indentations non respectées

4 - Les différentes fonctions

a) Définition d'une fonction

Une **fonction** Python est similaire à une fonction en Mathématiques. Elle prend un objet (antécédent) et le transforme en un autre (image). En Python, l'antécédent est appelé **argument**. Lorsque l'on définit une fonction, la syntaxe est toujours la même :

```
def fonction_random(argument):
```

On utilise « **def** » pour indiquer à l'ordinateur que l'objet qui suit est une fonction que l'on définit. Cette fonction est ensuite stockée quelque part sous le nom « fonction_random ».

b) Bloc d'opérations

Ce qui suit cette ligne de définition est le bloc d'opérations que l'argument va subir. Ce bloc doit être **indenté**, c'est-à-dire qu'il doit être décalé de 4 espaces vers la droite par rapport à « def ».

```
def fonction_random(argument):
    resultat=argument+2
```

c) Retourner un résultat

Enfin, il faut dire à la fonction de nous renvoyer le résultat. Pour cela, on utilise « return(...) ». Là encore, il faut que cette partie soit indentée par rapport) « def ».

```
def fonction_random(argument):
    resultat=argument+2
    return(resultat)
```

d) Appliquer une fonction

Pour appliquer une fonction, il suffit d'écrire sur une ligne le nom de la fonction, et l'argument auquel on veut que la fonction s'applique. Le résultat s'affiche alors sous la cellule que l'on exécute.

```
[3]: fonction_random(4)

[3]: 6
```

e) Une analogie pour mieux comprendre

Une fonction Python peut être vue comme une opération hospitalière. En amont, il faut dire au **patient** (l'argument) le **nom** de l'opération qu'il va subir (c'est à ça que sert le « def »). On lui décrit ensuite les **détails de l'opération** (bloc d'opérations), et enfin on lui indique qu'il retournera à sa chambre et qu'on le réveillera (« return »).

Lorsque l'on applique la fonction, ça y est, c'est le jour de l'opération. La chirurgienne indique le nom du **patient** qu'elle veut opérer (argument) et **quelle opération** elle va faire (fonction). Le patient est alors opéré par la chirurgienne (l'ordinateur applique le

bloc d'opérations). On le fait **retourner** à sa chambre après l'opération pour qu'il puisse voir sa famille (l'utilisateur).

L'intérêt d'une fonction est qu'elle peut être utilisée plusieurs fois, pour divers arguments. C'est pareil pour l'opération à l'hôpital, que la chirurgienne peut faire d'autres fois avec d'autres patients.

5 - Les différentes boucles

Les boucles sont essentielles pour répéter un grand nombre d'opérations successives. Si vous faites plusieurs fois le même calcul, cela signifie que vous auriez pu écrire une unique boucle et laisser l'ordinateur faire tous les calculs à votre place.

a) Boucle « for »

Cette boucle a toujours un début et une fin. Cela signifie que si on ne change rien à ce qui est écrit après « for », alors on fait toujours le même nombre de fois la boucle. Sa syntaxe exacte sera détaillée dans une autre fiche.

b) Boucle « while »

Cette boucle peut être traduite par « tant que ». Elle est donc suivie d'une condition. Tant que cette condition reste vraie, alors la boucle est effectuée. Cela signifie donc qu'il n'y a potentiellement pas de fin à cette boucle, donc que l'ordinateur ne finit jamais d'exécuter notre ordre… Cette boucle est donc plus compliquée à manipuler que la boucle « for ». Sa syntaxe exacte sera détaillée dans une autre fiche.