Séquentialité avec Turtle (Programmation 02)

I – Vocabulaire

1.1 Notion de VARIABLE

Pour l'instant, vous pouvez voir les variables comme des cartons permettant d'y ranger des choses.

La syntaxe Python permettant de créer une variable a utilise l'opérateur **=**.

```
>>> a = 50
```

Sa structure est assez simple en apparence :

- A GAUCHE : le nom de la variable voulue (ce nom ne peut pas contenir d'espace). Ici **a**.
- AU MILIEU : l'opérateur **=**
- A DROITE : la valeur qu'on veut associer à notre variable. Ici **50**.

On peut ensuite réutiliser sa valeur associée en utilisant juste son nom :

```
>>> a * 2
100
```

```
>>> a
50
```

1.2 - Notion de FONCTION

Les fonctions sont des moyens de répondre à des problèmes spécifiques.

- on leur donne des ENTRÉES : les données sur lesquelles on veut travailler et
- elles répondent en renvoyant une réponse en SORTIE.

ENTREE(S) ⇒ Fonction ⇒ SORTIE

Exemple avec la fonction VALEUR_ABSOLUE qui attend une entrée (un nombre) et qui renvoie la valeur absolue du nombre.

-12 ⇒ Fonction VALEUR ABSOLUE ⇒ 12
12 ⇒ Fonction VALEUR ABSOLUE ⇒ 12

D'autres exemples

5; 10; 50; 0; 20 ⇒ Fonction MAXIMUM ⇒ 50
5; 10; 50; 0; 20 ⇒ Fonction MINIMUM ⇒ 0
5; 10; 50; 0; 20 ⇒ Fonction SOMME ⇒ 85

1.3 - Syntaxe de l'appel d'une fonction

La syntaxe d'utilisation des fonctions en Python est la suivante :

```
nom_fonction(a, b, c, d...)
```

1. on écrit le nom de la fonction
2. on place des deux parenthèses ouverture-fermeture () juste à la suite du nom de la fonction
3. on place les ENTRÉES dans les parenthèses, en les séparant par des virgules au besoin

Python intégre plusieurs **fonctions natives** : des fonctions présentes de base dans Python.

Dans le cadre de ce site, les fonctions natives sont toujours écrites en vert.

Fonction VALEUR ABSOLUE : cette fonction se nomme **abs()** en Python.

```
>>> abs(-12)
12
```

```
>>> abs(12)
12
```

Fonction MAXIMUM : cette fonction se nomme **max()** en Python.

```
>>> max(10, 0, 50, 40)
50
```

II – Module Turtle

La **séquentialité** est une notion fondamentale : vous devez impérativement comprendre que lorsqu'on fournit un programme comportant plusieurs lignes à un ordinateur, il ne les exécute pas au hasard. Non, il les exécute ligne par ligne, en commençant par la première et en passant à la suivante uniquement après avoir correctement exécuté la ligne en cours.

Pour travailler sur la séquentialité, le mode interactif de la console est merveilleux : on ne peut rentrer qu'une instruction à la fois.

IMPORTANT : Distinction majuscule - minuscule

Attention aux minuscules et aux minuscules. Python distingue les deux. Cela veut dire que pour l'interpréteur Python, **`turtle`** ou **`Turtle`**, ce n'est pas la même chose.

`>>> import turtle as trt`	Importation du module turtle sous l'alias trt (plus court)
`>>> crayon = trt.Turtle()`	Création d'un crayon (attention à la majuscule)
`>>> crayon.forward(50)`	Avance de 50 pixels selon l'angle actuel
`>>> crayon.right(90)`	Tourne à droite de 90°
`>>> crayon.left(70)`	Tourne à gauche de 70°
`>>> crayon.setheading(45)`	Place la tortue selon un angle de 45° quelque soit l'angle actuel.
`>>> crayon.pencolor("red")`	Le trait de la tortue sera rouge à partir de maintenant
`>>> crayon.penup()`	Lève le crayon (il n'écrit plus)
`>>> crayon.pendown()`	Abaisse le crayon (il écrit)
`>>> crayon.clear()`	Efface le dessin effectué par ce crayon
`>>> crayon.home()`	Déplace le crayon en x = 0 et y = 0
`>>> crayon.goto(50,100)`	Déplace le crayon en x = 50 et y = 100
`>>> crayon.setx(100)`	Déplace horizontalement le crayon jusqu'à x = 100
`>>> crayon.sety(150)`	Déplace verticalement le crayon jusqu'à y = 100
`>>> crayon.circle(75)`	Trace un cercle de rayon `75 (360° donc)`
`>>> crayon.circle(75,90)`	Trace un quart de cercle de rayon 75 (car 90° = 360/4)
`>>> crayon.dot(5, "red")`	Trace un point rouge de 5 pixels de rayon.
`>>> crayon.write("bonjour")`	Ecrit bonjour à l'endroit où se trouve la tortue.
`>>> crayon.pensize(5)`	Fixe à 5 pixels la largeur du trait
`>>> crayon.fillcolor("yellow")`	Fixe la couleur de remplissage à jaune
`>>> crayon.begin_fill()`	On commence à surveiller la forme qu'on dessine
`>>> crayon.circle(50)`	... on dessine ...
`>>> crayon.end_fill()`	Fin de la surveillance et remplissage de la forme obtenue

Fonction et méthode

Avec l'apparition des parenthèses entre lesquelles on transmet (ou pas) des données, on voit que **`forward()`**, **`pencolor()`** ou encore **`penup()`** sont des fonctions mais dont la syntaxe d'utilisation est un peu étrange.

Ses fonctions particulières qui agissent sur l'objet placé devant elles se nomment des méthodes.

Quelle est cette syntaxe ? On donne le nom d'un objet sur lequel agir, on place un point et on donne ensuite le nom de la méthode et les données sur lesquelles la méthode devra travailler :

`objet.methode(données)`