Variables (Programmation 04)

1 Notion de VARIABLE (rappel)

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

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

Une fois la variable en mémoire, on peut utiliser sa valeur associée en utilisant son nom :

```
>>> a * 2          >>> a
100                50
```

2 Affichage VARIABLE (Thonny)

Menu VIEW – VARIABLES ou
Menu AFFICHAGE – VARIABLES.

3 Expression, instruction, affectation ?

Expression : ensemble de valeurs associées à des opérateurs. Une expression **ne modifie pas l'état** de la mémoire.

Evaluation : action d'évaluer une expression. Une évaluation **ne modifie pas l'état** de la mémoire.

Instruction : une instruction **modifie l'état** de la mémoire.

Affectation (opérateur =) : l'instruction la plus fondamentale puisqu'elle **modifie l'état** de la mémoire en modifiant juste une variable.

```
>>> "bon" * 2              → Evaluation
>>> crayon.forward(10)     → Instruction
>>> a = "bon" * 2          → Affectation
```

4 Sens d'exécution de l'affectation

Une **affectation** se fait toujours **de la droite vers la gauche**.

1. On évalue l'expression située à droite.
2. On affecte le résultat à gauche.

5 Version naïve de la boîte

Vous pouvez voir les variables comme le nom d'une boîte contenant quelque chose. Lorsqu'on tape ce nom, Python vous renvoie le contenu de la « boîte ».

6 - Incrémentation

Instruction augmentant la valeur associée à la variable **en utilisant sa valeur actuelle**.

```
>>> a = 10
>>> a = a + 1
```

Python évalue a+1 qui donne 11. Il affecte 11 à **a**.

7 - Pas de retroaction

Si une variable est définie à un certain moment à partir du contenu d'une autre variable, la modification ultérieure de cette variable n'aura aucune conséquence.

```
>>> a = 10
>>> b = a
>>> a = 20
>>> b
10
```

8 - Permutation de deux contenus

Nécessite une troisième variable temporaire en NSI.

```
>>> g = 7
>>> d = 55
>>> temp = d
>>> d = g
>>> g = temp
>>> g
55
>>> d
7
```

Méthode qui fonctionne mais à éviter en cours :

```
>>> g, d = d, g
```

9 - Affectation multiple (à éviter)

```
>>> a, b, c = 10
```
: a, b et c référencent 10.

III - Remarques importantes

4 règles sur les noms de variables :

+ Un nom de variable commence par une minuscule.

+ Le nom doit être explicite

+ Le caractère Espace n'est pas autorisé

+ Attention aux mots déjà réservés

Pas d'espace dans un nom de variable

Python est sensible à la casse

toto et **toTO** sont deux variables différentes

Deux manières d'écrire une variable

nb_paquets ou **nbPaquets**

Annexe : Mots réservés

False	class	finally	is	return
None	continue	for	lambda	try
True	def	from	nonlocal	while
and	del	global	not	with
as	elif	if	or	yield
assert	else	import	pass	break
except	in	raise		