Algo Python - Exercices type - Variables, boucles for - Corrigé

Remarque : Lorsque ce n'est pas précisé, le langage de prédilection est Python.

I Introduction - Syntaxe de base

Exercice I — Écrire un algorithme qui affiche précisément *Hello World !*

Correction On donne la solution en langage Python et langage TI (calculatrice).

Python	TI
print("Hello World !")	Disp "Hello World !"

□

Exercice II — Corrigez les erreurs dans le programme ci-dessous, afin qu'il affiche Bonjour.

```
print(Bonjour)
```

Correction Il faut mettre *Bonjour* entre guillemets. Donc le programme correct est : *print("Bonjour")* □

Exercice III — Les deux algorithmes suivant sont-ils identiques ?

print("Bonjour")	print("Bonjour !")

Correction Non. L'un renvoie *Bonjour* (sans point d'exclamation) et l'autre *Bonjour !* (avec un point d'exclamation). □

Exercice IV — Écrire un programme qui affiche exactement le texte qui suit :

```
Bonjour
Je m'appelle Hélène
Je suis une fille
Comme les autres
```

Correction

```
print("Bonjour")
print("Je m'appelle Hélène")
print("Je suis une fille")
print("Comme les autres")
```

Remarque : on peut aussi utiliser un seul print. Dans ce cas, il faut pouvoir dire à l'ordinateur de sauter une ligne entre chaque phrase. Le caractère pour "sauter une ligne" est \n (antislash n). Donc le code donnerait :

print("Bonjour\nJe m'appelle Hélène\nJe suis une fille\nComme les autres")

C'est illisible, mais l'ordinateur comprend. A l'avenir, on essaiera d'avoir des références musicales plus évoluées. □

Exercice V — (*, un peu délicat) Les deux programmes suivant vont-ils afficher la même chose ?

Algo 1	Algo 2
print("Un") print("Deux") print("Trois") print("Soleil")	print("Un ", end = "") print("Deux ", end = "") print("Trois " , end = "") print() print("Soleil")

Correction Le premier algo affiche :

```
Un
Deux
Trois
Soleil
```

(on saute une ligne après chaque instruction *print(...)*).

En revanche, dans le second, on a des instructions du type print("Un ", end =""). Cela veut dire que l'on va terminer la phrase par un vide, et pas par un saut de ligne. Donc on ne revient pas à la ligne : on affiche *Un Deux Trois*

Ensuite, l'instruction *print()* affiche quand à elle un texte vide. Mais ce texte vide est directement suivi d'un saut de ligne (pour éviter le saut de ligne, il faut ajouter un end="" dans le print, ce qui n'est pas fait ici). En d'autre terme, on retourne à la ligne.

L'algorithme 2 en entier affiche donc :

```
Un Deux Trois
Soleil
```

□

II Variable

Exercice VI — Que fais l'algorithme suivant ?

```
monAge = 25
print("J'ai", monAge, " !")
```

Correction Il affiche la phrase *J'ai 25 ans !* □

Exercice VII — La distance entre la Terre et la Lune est d'environ 55 758 000 km. Que fait l'algorithme suivant ?

```
distance = 55758000
print(distance)
print(2*distance)
```

Correction Il affiche la distance Terre-Lune, puis deux fois la distance, c'est à dire : 55758000 / 111516000 ($2 \times 55758000 = 111516000$). □

Exercice VIII — (important)

1. Parmi les noms de variables suivants, lesquels sont autorisés, et lesquels sont interdits ?
   maVariable ; distanceMaisonLycee ; 123Soleil ; Soleil123 ; J'ai_faim, J_ai_faim ; Trop !Bien
2. maVariable, MaVariable et mavariable représente-t-elle la même variable ?

Correction
1. 123Soleil n'est pas valide (on ne peut pas commencer par un chiffre). J'ai_faim n'est pas valide (il y a une apostrophe). Trop !Bien n'est pas valide (il y a un point d'exclamation).
   Les autres sont des noms valides.
2. Non, les majuscules ont une importance dans le nom des variables. □

Exercice IX — (important) Qu'affiche le programme suivant ?

```
contenance = 60
print("Départ :", contenance)
contenance = 100
print("Remplissage :", contenance)
contenance = contenance - 15
print("Consommation :", contenance)
```

Correction Il affiche :

```
60
100
85
```

□

Exercice X — (important) Une cour de récréation en forme de carrée a été mesurée avec quatre bâtons de longueurs respectives 17m, 7m et 2m. La longueur du côté de la cour est égale à 5 fois le premier bâton plus 2 fois le second plus 1 fois le troisième plus 2 fois le quatrième.

Écrire un programme qui affiche deux lignes : la première doit contenir la surface de la cour (en m^2), et la seconde ligne doit contenir son périmètre (en mètre).

Correction

```
longueur = 17 * 5 + 7 * 2 + 5 * 1 + 2 * 2
print(longueur * longueur)
print(longueur * 4)
```

La surface d'un carré vaut longueur fois longueur et le périmètre 4 fois la longueur. □

Exercice XI — (extrêmement important) Parmi les programmes suivants, lesquels sont valides ? Pour les non valides, dire ce qui ne va pas ; pour les programmes valides, préciser ce qu'affichent les algorithmes valides.

Algo 1
```
nombreDeFrere = 2
nombreDeSoeur = 0
print(nombreDeFrere + nombreDeSoeur)
```

Algo 2
```
nombreDeFrere = 2
print(nombreDeSoeur)
```

Algo 3
```
monAge = 15
monAge = 16
print(monAge)
```

Algo 4
```
age = 15
age = age+1
print(age)
```

Algo 5
```
1 = 1
print(1)
```

Algo 6
```
monAge = 15
monAge-1 = 15
print(monAge)
```

Algo 7
```
monAge = tonAge + 1
tonAge = 15
print(monAge)
```

Correction 1. L'algo 1 est valide et affiche 2 (le résultat de 2+0).

2. L'algo 2 n'est pas valide, car la variable nombreDeSoeur n'est pas définie !
3. L'algo 3 affiche 16 (la dernière valeur prise par la variable monAge).
4. L'algo 4 affiche aussi 16. L'instruction age = age + 1 veut dire que l'on affecte à la variable age la valeur age+1, où age est la dernière valeur de la variable age : ici 15. A écrire et expliquer en français c'est pénible et ça devient vite incompréhensible.
5. L'algo 5 n'est pas valide. En effet, l'instruction 1=1 veut dire que l'on nomme une variable 1, et on lui assigne la valeur 1. Or le nom d'une variable ne peut pas commencer par un chiffre, donc le programme va renvoyer une erreur.
6. L'algo 6 est non valide. En effet, monAge-1 n'est pas un nom valide de variable (il y a un caractère spécial : le signe moins).
7. L'algo 7 n'est pas valide. En effet, dans la première ligne, l'instruction monAge = tonAge+1 pose problème, car on a pas définit tonAge (on ne le définit qu'à la ligne d'après, mais l'ordinateur lit ligne par ligne). Pour corriger l'algo, il faudrait échanger la 2ème ligne avec la 1ère. L'algo renverrai alors 16. □

III Itération - boucle for

Exercice XII — (basique) Écrire un programme qui affiche 100 fois la phrase *Je dois être sage en cours de maths et écouter le prof.*

Correction
```
for i in range(100) :
    print("Je dois être sage en cours de maths et écouter le prof.")
```
Le fait de savoir faire cet exercice ne vous empêche pas d'être être sage en cours de maths et devoir écouter le prof. □

Exercice XIII — (basique) Écrire un programme qui affiche 32 fois *Bonjour* et une fois *Au revoir !*

Correction
```
for i in range(32) :
    print("Bonjour")
print("Au revoir")
```
Le *print("Au revoir")* n'est pas indenté : il est donc en dehors de la boucle for. □

Exercice XIV — (important) Corrigez les erreurs contenues dans le programme ci-dessous afin qu'il affiche 13 fois de suite le texte 9*8=72.
```
for loop in range(13)
print("9 * 8 = 72)
```

Correction Il manque :

- le deux point à la fin du for.
- l'indentation avant le print(...)
- Le guillemet à fermer après 72

Le code corrigé est :
```
for loop in range(13) :
    print("9 * 8 = 72")
```
□

Exercice XV — (important) Écrire un programme qui affiche tous les nombres entiers de 1 à 100, puis qui affiche la phrase *Ouf ! Fini, mais c'était long.*

Correction On propose deux solutions.

```
compteur = 0
for loop in range(100) :
    compteur = compteur +1
    print(compteur)
print("Ouf ! Fini, mais c'était long")
```

Dans cette algorithme, on a une variable appelée *compteur*, qui compte. Dans la boucle, ajoute 1 à chaque étape au compteur, puis on affiche le compteur. Comme on répète 100 fois la boucle, et que l'on commence à compter à 1, on a bien tous les nombres de 1 à 100.

Voici une deuxième possibilité :

```
for loop in range(100) :
    print(loop+1)
print("Ouf ! Fini, mais c'était long")
```

Ici, on utilise le fait que dans l'instruction for loop in range(100), loop est en fait une variable qui va prendre toutes les valeurs entières entre 0 et 99. Pour pouvoir compter de 1 à 100, il suffit donc d'afficher loop+1 (loop+1 car loop part de 0 et arrive à 100, alors que l'on veut compter de 1 à 100). C'est un point délicat de Python, et sujet à beaucoup d'erreurs bêtes : on commence à compter de 0 ! □

Exercice XVI — (*, un peu plus délicat pour l'instant) Écrire un programme qui compte à l'envers Votre programme devra afficher tous les nombres de 100 jusqu'à 0. Ensuite, il affichera *Décollage !*

Correction On prend une variable compteur, qui comme précédemment va compter. Mais cette fois, on part de 100. On effectue une boucle. A chaque itération de la boucle, on affiche le compteur puis on retranche 1. Donc les valeurs du compteurs vont prendre tous les nombres entre 100 et 0 (à condition d'effectuer 101 fois la boucle, car entre 100 et 0 il y a 101 nombres entiers !).

```
compteur = 100
for loop in range(101) :
    print(compteur)
    compteur = compteur - 1
    print("Décollage !")
```

□

Exercice XVII — (* plus dur, mais important) Ecrire un programme qui affiche :

```
0 1 2 3 4 5 6 7 8 9
0 1 2 3 4 5 6 7 8 9
0 1 2 3 4 5 6 7 8 9
0 1 2 3 4 5 6 7 8 9
0 1 2 3 4 5 6 7 8 9
```

Correction On a 5 lignes : donc on va commencer par une boucle for ligne in range(5).

Puis chaque ligne est identique. Une ligne correspond à l'affichage successif des chiffres 0, 1, ..., 9. Pour l'afficher, on va mettre un *for colonne in range(10)*. Donc *colonne* est une variable qui va de 0 à 9. Pour afficher cette variable, on fait un *print(colonne, end=" ")* (on pense bien au end= " " ; si on met juste print(colonne), on va sauter une ligne entre chaque chiffre !).

Lorsque l'on a fini d'afficher la colonne, on saute une ligne avec un *print()*.

```
for ligne in range(5) :
    for colonne in range(10) :
        print(colonne, end = " ")
    print()
```

□
