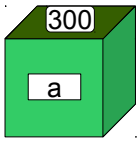
TP1 Python : Affectations, entrée, sortie

Dans les programmes, on doit souvent stocker des données pour les utiliser plus tard. On utilise pour ça des variables. Supposons que l'on veuille stocker la valeur 300, on peut utiliser une variable nommée par exemple « a », cette variable peut être vue comme une boîte :



En langage naturel, on peut écrire :	a prend la valeur 300
ou :	Affecter à a la valeur 300
En pseudo-code, on écrira ceci :	300 → a (on met 300 dans la boîte « a »)
ou ceci :	a ← 300
En Python, on écrira cela :	a = 300

Remarques :

- Le signe = n'a pas le même sens en informatique et en mathématiques.
  Ainsi, en informatique, l'instruction « 300 = a » ne veut rien dire (on ne peut pas mettre a dans 300 !).
  De même, en informatique, « a = b » ne veut pas dire la même chose que « b = a » !
- L'instruction « a ← 300 » est une affectation (on affecte la valeur 300 à la variable « a »).

Exercice 1 : affectations

Traduisez en Python les algorithmes suivants tout en les entrant dans la console de Thonny et donnez la valeur finale de r.

1°) Algorithme écrit en langage naturel :

Affecter à m la valeur 2
Affecter à n la valeur m – 1
Affecter à p la valeur n + 4
Affecter à q la valeur 3 p
Affecter à r la valeur p – m

2°) Algorithme écrit en « pseudo-code » :

```
p ← 1
r ← p + 3
p ← p – 1
r ← p + 1
```

Remarques :

- le pseudo-code est une façon d'écrire des algorithmes ;
- un tableau de valeurs des variables permet de vérifier qu'un programme fait ce qu'on attend de lui.

Exercice 2 : affectations, (re)découverte de deux règles

Pour chaque question, tapez les instructions suivantes en mode console et dîtes ce que vous remarquez :

1°)
```
>>> a = 4
>>> a = 1
>>> a
```

2°)
```
>>> a = 4
>>> b = 7
>>> a = b
>>> a
>>> b
```

3°)
```
>>> a = 4
>>> b = 7
>>> b = a
>>> a
>>> b
```

Deux propriétés de l'affectation

Écrasement : quand on met une valeur dans une variable, toute valeur existant déjà dans la variable est effacée (écrasée).

Par exemple, à la suite des deux instructions :

a ← 2
a ← 3

la variable « a » contient 3 et pas 5.

Si on veut ajouter 3 au contenu précédent de « a », on doit faire :

a ← a + 3

Copie : l'affectation entre variables copie le contenu d'une variable sans l'effacer.

Par exemple, à la suite des trois instructions :

a ← 2
b ← 7
a ← b

le contenu de « b » est copié dans « a ». Ainsi les deux variables *a* et *b* contiennent la valeur 7.

Remarque : comme dit précédemment, « a = b » ne veut pas dire la même chose que « b = a ».
« a = b » veut dire : on met la valeur de b dans a ; « b = a » veut dire : on met la valeur de a dans b.

Sorties

En mode console, pour afficher la valeur d'une variable, il suffit de taper son nom (exemples : exercices 1 et 2). En mode éditeur, on doit utiliser la fonction :

print(…)

Par exemple, pour afficher la valeur d'une variable nommée *dist*, on taperait :

print(dist)

(il faut que la variable *dist* ait été créée avant…).

Pour afficher un texte, on utilise des guillemets :

print("Ce programme calcule des distances.")

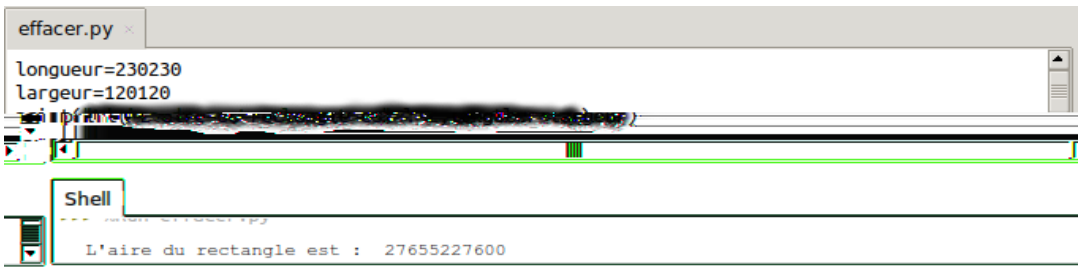
Pour mixer les deux, on sépare avec des virgules :

print("La distance calculée est :",dist)

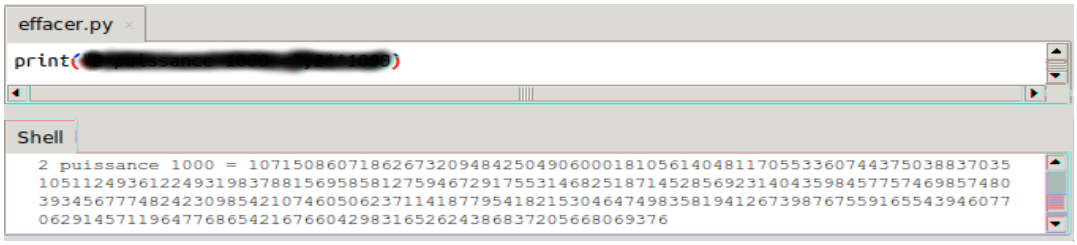
Exercice 3 : sorties

Tapez dans Thonny, en les complétant, les algorithmes suivants de façon à ce qu'ils affichent ce qui est écrit dans la console quand on les exécute (F5 ou) ; on n'utilisera pas de calculatrice ici.

1°)



2°)



3°) Même consigne ; vous pourrez changer le nom à la première ligne seulement et le reste du programme devra s'adapter…

Types de données

L'informatique est la science de la manipulation des informations, ou des données.
Celles-ci peuvent se présenter sous différents aspects, entre autres :

- des nombres entiers, tels que : 4 ; 236 ; – 21 etc.
- des nombres réels, tels que : 3.14 ; – 1.732 etc.
- du texte , par exemple : "Bonjour tout le monde !"
- des tableaux ou listes contenant des données, par exemple :
  jours = ["Samedi","Dimanche","Lundi","Mardi","Mercredi","Jeudi","Vendredi"]

Il est parfois nécessaire de passer d'un type à l'autre, on appelle cela un *trans-typage*.

Exercice 4 : types de données

1°) Tapez dans la console et observez bien les réponses de Python :

```
>>> a = 30
>>> type(a)
>>> b = "test !"
>>> type(b)
```

La commande type(…) demande le type de donnée d'une variable.
Pour Python, les entiers sont de type *int* (integer = entier) et les textes de type *str* (string = chaîne).

2°) Comment Python appelle-t-il les nombres à virgules ?

3°) Tapez dans la console et observez bien les réponses de Python :

```
>>> nb1 = 10
>>> nb2 = 20
>>> nb1+nb2
>>> chn1 =  "encore un "
>>> chn2 = "test"
>>> chn1+chn2
>>> chn2+nb1
```

On ne peut pas ajouter un texte et un nombre. Si on veut obtenir le texte "test10", il faut convertir le nombre 10 en le texte "10". Ceci se fait à l'aide d'une fonction de trans-typage, ici str(…).
Il existe également les commandes int(…) et float(…) pour convertir en entier ou en réel.

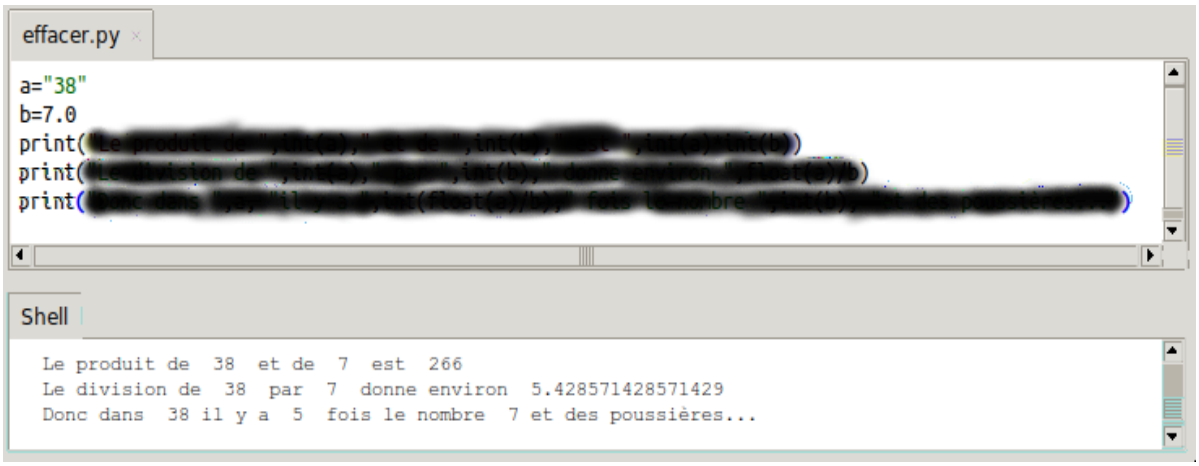
3°) Tapez donc dans la console :

```
>>> chn2+str(nb1)
```

et testez également les instructions suivantes :

```
>>> str(nb1)
>>> float(nb1)
>>> int(nb1)
>>> int(5.25)
```

4°) Tapez et complétez, dans l'éditeur de Thonny, l'algorithme suivant.
Les nombres 38 et 7 ne doivent pas apparaître dans les lignes cachées.



On constate que la conversion du type entier vers le type réel est automatique (exemple : 4 + 7.02).

Entrées

Que se passe-t-il lorsqu'un utilisateur veut retirer de l'argent à un distributeur bancaire ?

- le distributeur affiche une zone d'entrée de code et attend que l'utilisateur entre son code ;
- le distributeur traite ensuite cette information (vérification du code puis actions adaptées).

De même, pour qu'un utilisateur puisse entrer des données dans un algorithme existant, on utilise la commande Saisir (ou Lire ou Demander) en langage naturel.

Cette commande se traduit en langage Python en

input(…)

Par exemple, la commande :

votre_code = input("Quel est votre code ?")

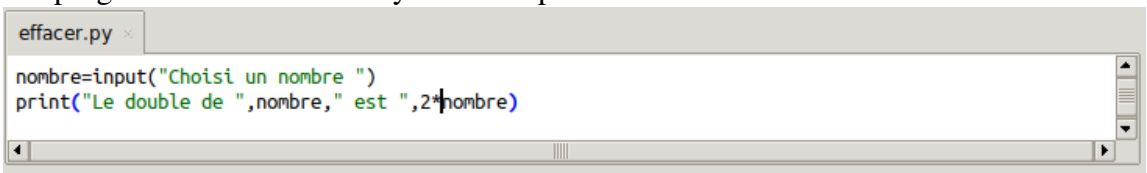
affichera le texte « Quel est votre code ? », attendra la réponse de l'utilisateur et placera la réponse dans la variable votre_code (qu'on pourra utiliser plus tard, pour vérification par exemple).

Exercice 5 : entrées

1°) Écrivez un programme en Python qui demande à un utilisateur son prénom (par exemple : Toto) et affiche le message :

Bonjour *prénom_choisi* (par exemple : Bonjour Toto)

2°) Testez le programme suivant et essayez de le réparer…



La commande input(…) renvoie automatiquement du texte. Il sera ainsi parfois nécessaire de convertir ce texte en entier ou en réel, comme dans l'exemple ci-dessous :

```
larg = input("Largeur du rectangle :")
larg = float(larg)
```

ou directement :

```
larg = float(input("Largeur du rectangle :"))
```

Exercice 6 : quelques petits programmes en Python

Créez les programmes suivants dans Python (mode éditeur) et lancez-les pour vérifier qu'ils fonctionnent. Imaginez que ces programmes sont destinés à un autre utilisateur, qui n'a pas accès à vos programmes. Demandez à chaque fois à votre professeur de vérifier (de valider) votre programme.

Programme 1 :

La machine demande à un utilisateur son année de naissance (exemple : l'utilisateur choisit 2003) et l'année actuelle (exemple : 2018) et affiche le message :

C'était il y a … années (exemple : C'était il y a 15 années)

Programme 2 :

La machine demande à un utilisateur un nombre entier et affiche le message :

2 ^… = …

(exemple : l' utilisateur entre 3 et la machine répond 2^3 = 8)

Programme 3 :

La machine demande à un utilisateur la longueur du côté d'un carré et affiche le périmètre et l'aire de ce carré.

Exercice 7 : traduction en pseudo-code puis en Python

Voici un algorithme :

- étant donnés deux nombres entiers ;
- multiplier le premier nombre par 2 ;
- multiplier le second nombre par 3 ;
- ajouter les deux résultats précédents ;
- afficher le dernier résultat.

1°) Que donne cet algorithme quand on y entre les nombres 4 et 7 ?

2°) Traduisez l'algorithme en pseudo-code.

3°) Traduisez l'algorithme en Python, exécutez-le (vérifiez la réponse à la question 1°) pour voir si votre programme fonctionne bien).

Exercice 8 : Le magicien

Un magicien demande à un spectateur :

- de penser à un nombre entier ;
- de le multiplier par 5 ;
- d'ajouter 7 au résultat ;
- de multiplier par 4 le résultat ;
- d'ajouter 6 au résultat ;
- de multiplier par 5 le résultat ;
- d'annoncer le résultat final obtenu.

1°) Le spectateur pense au nombre 4, quel nombre annonce-t-il ? 2°) Le magicien trouve à chaque fois le nombre choisi au départ par le spectateur ! Soit il est très fort en calcul mental, soit il a un truc de magicien… Créez un programme Python qui : – demande un nombre entier ; – effectue les opérations demandées par le magicien ; – affiche le résultat final (celui que le spectateur annonce). et entrez ce programme dans Thonny.	3°) A l'aide de ce programme, vérifiez la réponse à la question 1°). Relancez plusieurs fois votre programme, choisissez d'autres valeurs de départ et cherchez un lien entre le nombre choisi par le spectateur et celui qu'il annonce. 4°) On appelle x le nombre choisi par le spectateur. Écrivez en fonction de x le résultat qu'il annonce. Prouvez alors la remarque faîte au 3°).

Exercice 9 : où Python apprend des nouveaux mots

1°) Dans la plupart des langages informatiques (et Geogebra), la racine carrée se note sqrt(…) (pour square root). Par exemple, $\sqrt{3}$ s'écrit sqrt(3).
Essayez de taper ceci dans la console Python :

```
>>> a=sqrt(3)
```

Python vous dira que le nom sqrt n'est pas défini ! Cela veut dire que la fonction sqrt() ne fait pas partie du vocabulaire de base de Python.

On peut enrichir le vocabulaire de Python en ajoutant au début du programme l'instruction :

```
from math import sqrt
```

qui veut dire : aller chercher dans la bibliothèque « math » la signification de la fonction sqrt.
(on peut récupérer toutes les fonctions de la bibliothèque math en tapant : from math import *)

Retapez dans la console Python : `>>> a=sqrt(3)`

2°) Faîtes un programme qui demande les deux côtés de l'angle droit dans un triangle rectangle et affiche la longueur approchée de l'hypoténuse.

Exercice 10 (bonus)

1°) Écrivez un algorithme qui résout des équations du type :
$2x+3=0$, $5x+2=0$, $-3x+4=0$, $-5x-1=0$, etc.

2°) Écrivez un algorithme qui résout des équations du type :
$2x+3=3x+1$, $2x-5=-3x+1$, $-4x+2=-5x-2$, etc.