Fiche n°01 : Les bases de la programmation

Rédigée par Pascal Delahaye

Cette fiche contient des élèments de cours à assimiler pour la leçon de la semaine suivante.
Vous devez impérativement :

- Travailler avec un ordinateur afin de vérifier une à une les différentes instructions qui sont présentées.
- Vérifier votre assimilation en effectuant tous les exercices d'entrainement proposés à la fin du poly.
- Prévoir environ 1h30 de travail personnel.

Une évaluation de 10mn sera effectuée pour vérifier la qualité de votre travail d'auto-apprentissage.

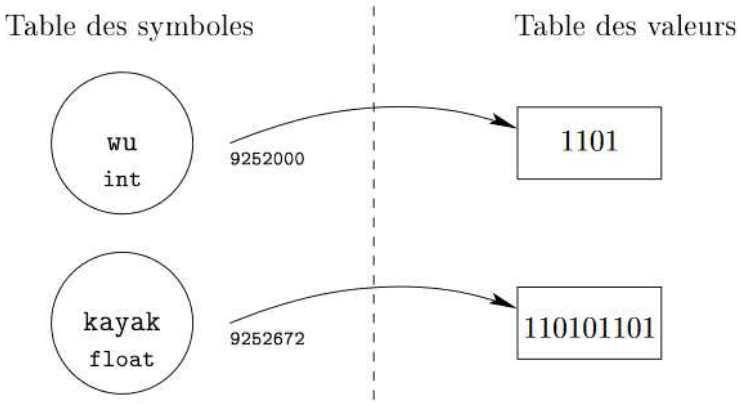
Commencer par importer et installer sur votre ordinateur la distribution ANACONDA (Python 3.6 - 64 bits) présente à l'adresse suivante : `https://www.anaconda.com/download/`

Les programmes se rédigent et s'utilisent dans l'application `spider`.
Le but de cette fiche est de présenter quelques instructions de base du langage Python.

I] L'affectation d'une variable

Une variable en python est un nom (comme par exemple `A, L, kayak, a1...`) auquel est associé un type (entier, flottant, liste...) et une valeur. Ces données sont stockées dans la mémoire de l'ordinateur dans deux espaces :

- La table des symboles où à chaque nom de variable est associé son type et une adresse dans la table des valeurs.
- La table des valeurs qui contient les valeurs (données sous la forme d'un codage binaire) des différentes variables existantes.



L'*affectation* consiste à créer une variable et à lui attribuer un type et une valeur.
L'instruction se présente sous la forme suivante :

Python
```
# Affectation d'une ou plusieurs variables

A = 2.6               # On affecte à la variable A la valeur 2.6 de type "flottant"
A                     # On vérifie que la variable A contient bien la valeur 2.6

A,B,C = 3,[],"toto"   # On affecte à A la valeur 3 de type "entier"
                      # On affecte à B la valeur []  de type "liste vide"
                      # On affecte à C la valeur "toto" de type "chaîne de caractères"

A,B = B,A             # On échange les valeurs des variables A et B (uniquement en Python)
A,B                   # Vérification du contenu des deux variables
C = A                 # La variable C prend la valeur de la variable A
```

Attention : dans l'instruction `A = B` :

1. `A` repésente le nom de la variable qui est créée ou modifiée
2. `B` représente la valeur (entier, flottant, liste, chaîne de caractères...) que l'on attribue à cette variable

Lors de la création d'une variable, l'ordinateur réserve un espace dans sa mémoire et y stocke sous forme "binaire" la valeur dont est affectée la variable. La taille de l'espace mémoire réservé diffère selon le type de la valeur stockée : entier, flottant, liste, chaîne de caractères...

II] Appel d'une fonction ou d'une méthode

1) Deux premières fonctions à connaître

Lorsque l'on souhaite afficher une phrase ou le contenu d'une variable à l'écran, on utilise la fonction `print()`.

Python
```
print("texte que l'on souhaite imprimer")    # Le texte entre guillemets s'affiche à l'écran
print(A)                                     # Le contenu de la variable A s'affiche à l'écran
print("texte",A)                             # Le texte et le contenu de A s'affichent à l'écran
```

Les boucles `for` vues plus tard font souvent varier une variable d'une valeur a à une valeur b. On utilise pour cela la fonction `range()`.

Python
```
L1 = range(10)    # L1 est la liste contenant les entiers de 0 à 9 (Attention : pas à 10)
L1                # Vérification du contenu

L2 = range(1,10)  # L2 est la liste contenant les entiers de 1 à 9 (Attention : pas à 10)
L2                # vérification du contenu de L2
```

2) Interaction avec l'utilisateur

Parfois, le concepteur d'un programme souhaite que les valeurs des arguments puissent être directement demandées à l'utilisateur. Dans ce cas, on utilise la fonction `input()`.

Python
```
A = input("Entrer un entier : ")       # la valeur entrée par l'utilisateur est stockée dans A
                                         sous la forme d'une chaîne de caractères
A                                      # vérification de la valeur contenue dans A

A = eval(input("Entrer un entier : ")) # la valeur entrée par l'utilisateur est stockée dans A
                                         sous la forme d'un nombre entier ou flottant
                                         selon la valeur rentrée.
A                                      # vérification de la valeur contenue dans A
```

Un exemple de fonction :

Python
```
def somme():                                          # définition de la fonction "somme"
    a = eval(input("Entrer un premier nombre : "))
    b = eval(input("Entrer un premier nombre : "))
    print("le produit vaut ", a*b)                    # On renvoie le produit de a par b

somme()                                               # utilisation de la fonction somme()
```

Cette fonction `somme()` demande à l'utilisateur de rentrer 2 entiers au clavier et affiche à l'écran une phrase donnant leur produit.

ATTENTION : il est impératif de bien décaler de 4 espaces les instructions se trouvant à l'intérieur de la fonction.

3) Trois méthodes à connaître

Les méthodes sont des procédures qui agissent sur le contenu d'une structure de données.
Elles prennent la forme suivante :

`A.meth(var1,var2,...)` → On applique à la variable A la méthode "meth" avec les arguments "var1,..."

Python
```
L = [1,2]     # on crée une variable contenant la liste [1,2]

L.append(3)   # la valeur "3" est stockée à la fin de la liste L (à droite)
L             # vérification du contenu de L

L.reverse()   # l'ordre des éléments de la liste L sont inversés
L             # vérification du contenu de L

G = L         # commande interdite car ici G et L pointent sur le même espace mémoire
G = L.copy()  # copie le contenu de la liste L dans la variable G : L et G sont indépendantes
```

4) Importation d'une fonction depuis une bibliothèque

Certaines fonctions ne sont accessibles qu'après les avoir importées depuis une bibliothèque.

Python
```
from math import cos, sin, pi     # importation des fonctions "sinus" et "cosinus"
                                    et du nombre "pi" depuis la bibliothèque "math"

cos(pi)                           # vérification de la valeur de cos(pi)
(cos(1))**2 + (sin(1))**2         # vérification d'une formule usuelle


from matplotlib.pylab import plot # importation de la fonction "plot" de la bibliothèque
                                    matplotlib.pylab
plot([1,2,3,4],[1,4,9,16])        # représentation d'une ligne brisée
```

On retiendra que la fonction `plot(X,Y)` de la bibliothèque `matplotlib.pylab` trace une ligne brisée reliant les points dont les abscisses sont contenues dans une liste X et les ordonnées dans une autre liste Y.

5) La bibliothèque random

La bibliothèque `random` est très utile pour la simulation d'expériences aléatoires.
On commencera par retenir les deux fonctions suivantes :

Python
```
from random import randint  # importation de la fonction randint disponible dans la bibli random
n = randint(1,10)           # choisi de façon équiprobable un entier entre 1 et 10

from random import random   # importation de la fonction random
x = random()                # choisi de façon équiprobable un réel entre 0 et 1
y = 5*random()              # choisi de façon équiprobable un réel entre 0 et 5
```

III] Exercices

Exercice : 1

Taper les instructions suivantes :

1. Créer une variable `adj1` prenant la valeur "petit"
2. Créer une variable `adj2` prenant la valeur "vert"
3. A l'aide des deux variables précédentes, afficher à l'écran la phrase "toto est petit et vert"
4. Echanger les valeurs des variables `adj1` et `adj2` et relancer la commande précédente.

Exercice : 2

Taper les instructions suivantes :

1. Demander à l'utilisateur de choisir un nombre a entre 0 et 9 et stocker cette valeur dans une variable A
2. Multiplier cette valeur par 2 puis stocker cette valeur b dans une variable B
3. Afficher à l'écran la phrase : "b est le double de a"
4. Créer une fonction `double()` qui effectue les instructions précédentes.

Exercice : 3

Taper les instructions suivantes :

1. Créer une variable L contenant la liste `[2,3,5,7,9]`
2. Ajouter la valeur 11 en fin de liste puis vérifier le contenu de L
3. Inverser l'ordre des éléments de la liste L puis vérifier le contenu de L.

Exercice : 4

Taper les instructions suivantes :

1. Importer les fonctions array et cos depuis la bibliothèque numpy
2. Importer la fonction plot depuis la bibliothèque matplotlib.pylab
3. Créer une variable X contenant `array([-1,0,1,2,3])`
4. Créer une variable Y contenant `cos(X)`
5. Représenter les points d'abscisses X et d'ordonnées Y

Exercice : 5

Taper les instructions suivantes :

1. Importer les fonctions randint et random la bibliothèque random
2. Créer une variable X contenant la liste de 5 entiers choisis aléatoirement entre 0 et 10
3. Créer une variable Y contenant la liste de 5 entiers choisis aléatoirement entre 0 et 10
4. Créer une variable Z contenant la liste de 5 réels choisis aléatoirement entre 0 et 10
5. Représenter les points d'abscisses X et d'ordonnées Y.
6. Représenter les points d'abscisses X et d'ordonnées Z.