Algorithmique et prise main de Python

1 Variables

1.1 Type de variables

- Chaînes de caractères : des mots ou des phrases, notés entre guillemets.
- Réels
- Entiers (positifs ou négatifs)
- Booléen : peut prendre les valeurs VRAI ou FAUX.

1.2 Questions de cours

a) À quoi sert une variable ?
b) Comment affecter une valeur à une variable a ?
c) Comment afficher la valeur d'une variable a ?
d) Comment demander de saisir un nombre et le stocker dans une variable a ?
e) Quel est le symbole pour l'exposant (puissance) en Python ? par exemple a^5.
f) À quoi correspondent les opérations `//` et `%` en Python ?
g) Quelle est la différence entre les opérations `/` et `//` en Python ?

Exercice 1. On donne l'algorithme suivant écrit en pseudo code :

```
1  A = 5
2  B = 2
3  C = A
4  A = A + 1
5  B = B + C
```

Compléter le tableau ci-dessous, appelé « tableau de l'état des variables », avec le contenu des variables à chaque ligne. *On notera* `NI` *pour une variable non initialisée.*

Étape N° ligne	Variables		
	A	B	C
1			
2			
3			
4			
5			

1.3 Un exemple : échange de variables

```
Algo Mystere
Variables: a, b, c réels
Traitement:
    Saisir a, b
    c=a
    a=b
    b=c
Fin
```

1) Que fait l'algorithme `Mystere` ?
2) Dresser le tableau de l'état des variables du programme ci-dessus.

2 Structure conditionnelle : Si/Sinon

Une **structure alternative** (ou **test**) consiste, selon qu'une certaine condition est vérifiée ou non, à effectuer un certain traitement ou un autre. On la rédige de la manière suivante :

```
Si condition alors
   traitement 1     #indentation obligatoire
Sinon
   traitement 2     #le sinon peut être omis
Fin si
```

En Python, on écrira :

```
if condition :
   traitement 1
else:
   traitement 2
```

Exemple

Algorithme qui selon deux nombres donnés affiche le plus grand :

```
Algo Max
Variables: a, b réels

Traitement:
        Saisir a, b
        Si a>b alors:
                    Afficher : a
        Sinon:
                    Afficher : b
    Fin si
Fin
```

En Python :

```
a=float(input("Entrer a"))
b=float(input("Entrer b"))
if a>b:
        print(a)
else:
        print(b)
```

Exercice 2.

```
Algo Moyenne
Variables: a, b, c, X, Y, Z , m réels
Traitement:
        Saisir a, b, c, X, Y, Z
    On affecte ((aX+bY+cZ)/(a+b+c)) à m
    Si m>=10
            alors afficher : "réussite à l'examen"
    Sinon afficher : "échec à l'examen"
    Fin si
Fin
```

Voici un algorithme qui permet déterminer si un étudiant a eu la moyenne à un examen.

a) Que représente les nombres a, b, c ?
b) Quel message est affiché si a=2, b=1, c=4, X=11, Y=12, Z=8.

c) Écrire le programme correspondant en Python.

Exercice 3. Écrire un programme qui demande la moyenne m obtenue au bac, puis qui indique la mention obtenue.

(Rappels : Si $m \geqslant 16$: mention TB ; si $16 > m \geqslant 14$: mention B ; si $14 > m \geqslant 12$: mention AB ; si $12 > m \geqslant 10$: pas de mention ; si $m < 10$: échec.)

3 Boucle pour

La boucle « Pour » permet de faire répéter un traitement un **nombre de fois connu à l'avance**. On indique alors une valeur de départ, une valeur d'arrivée ou à ne pas dépasser et une valeur d'**incrémentation** (la manière de passer d'une valeur à la suivante.

On la rédige de la manière suivante :

```
Pour i allant de (VI) à (VF) par pas de (P) faire
  traitement  #indentation obligatoire
Fin Pour
```

avec VI désignant la valeur initiale, VF la valeur finale et P le pas (ou l'incrément) et où i est appelé **indice de boucle** (on utilise parfois les lettres n, j ou k pour les indices.

Le pas d'incrémentation peut être facultatif quand on avance de 1 en 1.
En Python, on écrira :

```
for i in range(debut, fin+1, pas):
    traitement
 #fin for
```

i prendra les valeurs de debut jusqu'à `fin` en avant de pas en pas.

Si le pas vaut 1 on peut ne pas l'écrire, dans le range.

3.1 Exemple

```
Algo Table
Variables: i,a entiers
Traitement:
        Saisir a
    Pour i  allant de 1 à 10 faire
                Afficher(i*a)
    Fin Pour
```

Cet algorithme affiche la table de multiplication de a.

En Python :

```
a=int(input("entrer a"))
for i in range (1,11):
        print(i*a)
```

Exercice 4.
Que font les instructions python suivantes ?

```
for i in range(1, 12, 1):
    print(i)
#fin for

for i in range(1, 12):
    print(i)
#fin for

for i in range(0, 13, 3):
    print(i)
#fin for

for i in range(6):
    print(i)
#fin for

for i in range(16,6,-1):
    print(i)
#fin for

for i in range(19,-7,-4):
    print(i)
#fin for
```

Exercice 5. Étant donné un entier strictement positif n, écrire l'algorithme puis le programme Python permettant de calculer la somme $1 + 2 + 3 + \cdots + n$.

4 Boucle tant que

La boucle « Tant que » permet de répéter un traitement un **certain nombre de fois non connu à l'avance**. Tant qu'une certaine condition est réalisée, on continue d'entrer dans la boucle.

On la rédige de la manière suivante :

```
Tant que condition faire
   traitement     #indentation obligatoire
Fin Tant que
```

En Python, on écrira :

```
while condition :
   traitement
#fin while
```

Exemple

```
Algo Tq
Variables :        x nombre réel
Traitement :
                Saisir x
                Tant que x>5 faire
                        x prend la valeur x/2 + 1
                Fin Tant que
                Afficher x
Fin
```

Si on rentre $x = 18$, x prend ensuite les valeurs 10, 6, 4. Il affiche la valeur 4.

En Python :

```
x=float(input("donner x"))
while x>5:
        x=1/2*x+1
print(x)
```

Exercice 6. Écrire un programme qui demande la valeur d'un entier N, puis qui affiche tous les multiples de 7 inférieurs ou égaux à N.

Exercice 7. On dispose de N œufs. Trouver un programme donnant le nombre NB de boîtes de 12 que l'on peut remplir, et le nombre Reste d'œufs qui resteront. On fera seulement des additions et des soustractions.

5 Listes

Une liste est une collection d'éléments ordonnés, séparés par des virgules, l'ensemble étant enfermé dans des crochets.

Exemple. `L=[3,51,-12,9]`
La numérotation commence à zéro : `L[0]` vaut 3. `L[1]` vaut 51.

Quelques fonctions :
`len` : taille de la liste.
Dans notre exemple, `len(L)=4`.
`append` : ajoute un élément à la liste.
`L.append(8)` signifie que la liste `L` devient `L=[3,51,-12,9,8]`.

Exemples

Taper les commandes suivantes dans la console :

```
L=8*[1]
M=list(range(3,21,4))
P=[23,2,4,1,7,8,-3]
```

Trouver le maximum de P :

```
m=P[0]
n=len(P)
for i in range(n):
        if P[i]>m:
                m=P[i]
print("le maximum est ", m)
```

Exercice 8. Une liste contient `N` nombres.

a) Compter ses éléments non nuls.
b) Rechercher la place p du plus petit élément.

Exercice 9. Écrivez un programme qui analyse un par un tous les éléments d'une liste de nombres (par exemple celle de l'exercice précédent) pour générer deux nouvelles listes. L'une contiendra seulement les nombres pairs de la liste initiale, et l'autre les nombres impairs.

6 Chaînes de caractères

Une donnée de type « chaîne de caractères » est une suite de caractères quelconques. Une constante chaîne de caractères est indiquée en écrivant les caractères en question soit entre apostrophes, soit entre guillemets : `'Bonjour'` et `"Bonjour"` sont deux écritures correctes de la même chaîne.

Exemple

```
ch1="Bonjour"
ch2="Monsieur"
```

a) Que fait `ch1+ch2` ?
b) Que fait `ch1[2:]` ?
c) Que fait `ch2[:3]` ?

Exercice 10. Écrivez un script qui détermine si une chaîne contient ou non le caractère « n ».
Exercice 11. Écrivez un script qui recopie une chaîne (dans une nouvelle variable) en l'inversant. Ainsi, par exemple, « zorglub » deviendra « bulgroz ».

AIDE-MÉMOIRE PYTHON

Instructions de base

- On ne déclare pas les variables. Pas d'instructions de début et de fin ; c'est l'indentation qui joue ce rôle.
- Précéder les commentaires d'un #
- Pour afficher à l'écran, utiliser :
  `print("bonjour",variable,"bonsoir")`
  si `variable=alex`, l'affichage sera *bonjour alex bonsoir*)
- `input()` # saisie au clavier d'une chaîne de caractères à transformer en `int\verb` ou en `float`.
- On peut regrouper les instructions avec `N=int(input())`
- Les virgules des décimaux se notent avec des points.
- `a//b` affiche le quotient de *a* par *b*.
  `a%b` affiche le reste de la division euclidienne de *a* par *b*.
- Pour générer des nombres aléatoires on peut faire un import de la librairie `random` (`import random`) et utiliser `random.random()` (un nombre entre 0 et 1) ou `random.randint(1,5)` par exemple pour tirer un nombre entier entre 1 et 5.

Structure de contrôle

- `if(N>1)` : # aller à la ligne et décaler, revenir au même niveau pour le `else` : et de nouveau à la ligne en décalant.
  `elif` : correspond au sinon si (contraction de `else if`)
- On peut combiner plusieurs conditions avec le `and`, le `or` et le `not`.

Boucles

- `while(N>1)` : #aller à la ligne et décaler
- `for i in range(5)` : #aller à la ligne et décaler ; l'indice `i` prendra les valeurs 0,1,2,3 et 4.
- `for i in range(3,10,2)` l'indice `i` prendra les valeurs 3,5,7 et 9.
- `for i in range(4,-1,-1)` l'indice `i` prendra les valeurs 4,3,2,1 et 0.

Listes

En Python, une liste est notée entre `[ ]` et les éléments sont séparés par des virgules.

- Pour créer une liste, voici 3 méthodes :
  - Si on veut saisir une liste complète connue, taper directement `Liste=[1,3,5,7]`.
  - Pour remplir une liste remplie de 0, on utilise `Liste=[0]*4` ou `Liste=[0 for i in range(n)]`.
  - On crée une liste vide en début de programme `Liste=[ ]`. Ensuite, on la remplit en utilisant la fonction append :

```
Liste=[ ]
for i in range(0,10):
        Liste.append(i)
#Crée la liste [0,1,2,3,4,5,6,7,8,9]
```

- On ne peut pas entrer directement une valeur saisie au clavier avec `.append`. Il faut faire

```
val=int(input())
Liste.append(val)
```

- Pour afficher une liste, faire *print*(*Tab*) ou utiliser une boucle pour avoir un élément par ligne.
- La fonction *len* donne la longueur de la liste c'est-à-dire le nombre d'éléments qu'elle contient.

```
Liste=[1,3,5,7,9]
a=len(Liste)
print(a) #donnera 5
```

- Les éléments d'une liste de n éléments sont numérotés à partir de 0 jusqu'à n-1.
- Le contenu de la case *i* est noté `Liste[i]`.
  `Liste[3]` contient la 4ième valeur. Dans l'exemple précédent, elle vaut 7.

Matrices

Une matrice est représentée en Python par une liste de listes : Pour représenter la matrice

$$\text{Mat} = \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{pmatrix},$$

on tape dans l'éditeur : `Mat=[[1,2,3],[4,5,6]]`.
`Mat[1]` contient `[4,5,6]` car c'est la ligne numéro 1 et que Python numérote à partir de 0.
`Mat[0][2]` contient 3 c'est l'élément de la première ligne (`i=0`), 3ième colonne (`j=2`).

- Nombre de lignes : `n=len(Mat)` # donne à *n* la valeur 2.
- Nombre de colonnes : `p=len(Mat[0])` # donne à *p* la valeur 3.
- L'élément `i, j` de la matrice est noté `Mat[i][j]`.
- Pour construire une matrice de taille `n*p` définie, préremplie de 0 faire :

```
M=[[0 for j in range (p)]for i in range(n)]
```

  Cela crée *n* listes de *p* éléments chacune.
  Il faut toujours initialiser une matrice avant de la remplir.
- Si on veut avoir deux matrices identiques au départ pour en modifier une et garder l'autre, il faut copier les éléments un à un à l'aide d'une double boucle.
  On veut copier `Mat` dans la matrice *M* :

```
M=[[0 for j in range (3)]for i in range(2)]
for i in range(2):
    for j in range(3):
       M[i][j]=Mat[i][j]
```

- `print(Mat)` affiche `[[1,2,3],[4,5,6]]`.
-

```
for i in range(2):
 print(Mat[i])
```

affiche

```
[1,2,3]
[4,5,6]
```

-

```
for i in range(2):
  for j in range(3):
    print(Mat[i])
```

affiche tous les éléments les uns en dessous des autres.

Fonctions et procédures

On utilise pour les deux la structure
`def nomfonction(param1,param2):`
puis on va à la ligne et on décale.

Pour retourner le résultat de la fonction, on utilise l'instruction

`return res`

(`res` étant le nom de la variable retournée).