BTS SIO – 1

ALGORITHMIQUE APPLIQUÉE – Partie 2

Table des matières

I. Instructions conditionnelles

En algorithmique, on peut vouloir effectuer ou non une instruction selon qu'une condition est vrai ou fausse. On parle d'**instruction conditionnelle**. Le tableau suivant on présente la structure en pseudo code et en Python.

Pseudo-code	Python

Si <condition> **Alors**
<instructions>
SinonSi <condition> **Alors**
<instructions>
Sinon
<instructions>
FinSi
	`if <condition> :`
	`  <instructions>`
	`elif <condition> :`
	`  <instructions>`
	`else :`
	`  <instructions>`

L'algorithme suivant demande à l'utilisateur un nombre entier puis affiche sa parité.

```
Variable : n(entier)
Début
        Afficher "Saisir un nombre entier"
        Saisir n
        Si n%2==0 Alors
                Afficher "Ce nombre est pair "
        Sinon
                Afficher "Ce nombre est impair"
        FinSi
Fin
```

Implémentation en Python :

```
n=int(input("Saisir un nombre entier:"))
if n%2==0:
    print("Ce nombre est pair.")
else:
    print("Ce nombre est impair")
```

Remarques :

- Attention à ne pas oublier les « **:** » qui traduisent le « Alors » !!!
- On peut imbriquer des instructions conditionnelles les unes dans les autres.

II. Instruction itératives

Pour exécuter plusieurs fois de suite un bloc d'instruction, on utilise une boucle.

Si le nombre d'exécution est connu à l'avance, on va privilégier une boucle « Pour », sinon une boucle « TantQue ».

II.1. La boucle Pour (for)

Activité

Exécuter les programmes Python suivants. En déduire la définition de `range(0,10)` et le rôle de la structure <`for nom-var in L :`> pour une liste L.

Programme1 :

```
for k in range(0,10):
    print(k)
```

Programme2 :

```
for k in range(0,11,2):
    print(k)
```

Programme3:

```
for k in [2,3,7,9]:
    print(k)
```

`range(0,10)` : ……

<`for nom-var in L :`> : ……

En pseudo-code	En Python	Résultat
Pour var **dans** liste **Faire** <instructions> **FinPour**	`for var in liste :` `  <instructions>`	*La variable var prend toutes les valeurs des éléments de la liste, et pour chacunes d'elles, exécute le bloc instructions*
Pour var **de** n à p **Faire** <instructions> **FinPour**	`for var in range(n,p+1) :` `  <instructions>`	*La variable var prend toutes les valeurs entières comprises entre n et p+1, et pour chacunes d'elles, exécute le bloc instructions*

Remarques :

- `in range(3)` : la variable prendra toutes les valeurs comprises entre 0 et 2 (inclus).
- `in range(2,11,2)` : la variable prendra toutes les valeurs comprises entre 0 et 10 avec un pas de 2, c'est-à-dire : 2, 4, 6, 8 et 10.

Conseil :

Lorsque l'on écrit un algorithme ou un programme non trivial, il est bon d'écrire des commentaires à côté de certaines instructions. On utilise pour cela « # » en Python on fait de même en pseudo-code.

Les commentaires n'ont aucun effet lors de l'exécution. Il constitue simplement une aide dans la lecture ou la relecture d'un programme ou d'un algorithme.

L'algorithme suivant calcule la somme des entiers de 1 à 100 :

```
Variable : S(entier)
Début
        S←0   #initialisation
        Pour k de 1 à 100 Faire
                S←S+k
        FinPour
        Afficher S
Fin
```

Implémentation en Python :

```
S=0 #initialisation
for k in range(1,101): #Attention range(1,101) et non range(1,100)
    S=S+k
print(S)
```

II.2. La boucle TantQue (while)

Activité
Exécuter le programme Python suivant et en déduire le rôle de la structure « `while <condition> :` »

```
n=0
while n<10:
    n=n+3
    print(n)
```

En pseudo-code	En Python	Résultat
TantQue <condition> **Faire** <instructions> **FinTantQue**	`while <condition>:` `  <instructions>`	Tant que la condition `<condition>` est vrai, on effectue le bloc d'instructions `<instructions>`

L'algorithme suivant détermine et affiche le plus petit entier n tel que $1{,}01^n > 200$:

```
Variable : n(entier)
Début
        n←1   #initialisation
        TantQue 1,01^n<=200 Faire
                n←n+1
```

```
    FinTantQue
    Afficher n
Fin
```

Implémentation en Python :

```
n=1 #initialisation
while 1.01**n<=200:
    n=n+1
print(n)
```

Résultat: 533

III. Fonctions et procedures

III.1. Fonctions usuelles

Les fonctions mathématiques usuelles (exp, ln, sin, $\sqrt{\ }$, …) sont disponibles en Python en chargeant le module math à l'aide de l'instruction `import math`, ou `from math import sqrt` (pour la racine carrée), ou encore `from math import *` pour importer tout le module.

Exemples :

```
>>> import math
>>> math.sqrt(3)  #dans ce cas on écrit math.fonction
1.7320508075688772
```

```
>>> from math import *
>>> sqrt(3)
1.7320508075688772
```

```
>>> from math import cos
>>> cos(30)
0.15425144988758405
```

III.2. Création de fonctions et procédures

Activité :

1. Dans l'éditeur de texte de Thonny rentrez le programme suivant et exécutez-le.

```
def f(x):
    return (x*x)
def g(x):
    print(x*x)
```

2. Dans la console, saisir et valider f(2), f(5), g(2) et g(5).
3. Saisir dans la console a=f(3), puis appeler a, et ensuite b=g(3) et appelé b. Que constatez-vous ?

Remarque :
Pour les fonctions, on privilégiera l'utilisation de « `return` » plutôt que « `print` ».

Activité :
Exécuter le programme Python suivant, puis dans la console, saisir et valider mystere(3), mystere(4) et mystere(7).
Quel est le rôle de cette fonction ?

```
def mystere(n):
    if n%2==0:
        return('oui')
    else:
        return('non')
```

Pseudo-code	Python

Fonction nom_fonction(paramètres,types) :
Début
<instructions>
Retourner(données de sortie)
FinFonction
	`def nom_fonction(paramètres) :`
	`  <instructions>`
	`  return(données de sortie)`

La fonction suivante retourne à tout entier n, la liste de ses chiffres :

Fonction liste_chiffres(n :entier) :liste
Variables locales : k(entier), L(liste), ch(chaîne)
Début
    ch←str(n) #chaîne de caractères associée à n
    L←[] #initialisation (liste vide)
    Pour k de 0 à longueur(ch)-1 **Faire**
        Ajouter int(ch[k]) à L #int(ch[k]) renvoie le chiffre associé à ch[k]
    FinPour
    Retourner(L)
FinFonction

Implémentation en Python :

```
def liste_chiffres(n):
    ch,L=str(n),[]
    for k in range(len(ch)):
        L.append(int(ch[k]))
    return(L)
```

Activité :

1. Exécuter le programme Python suivant :

```
a=6
def f(x):
    a=20
    b=2*x
    return(b+1)
```

2. Dans la console, appeler la valeur f(4), puis celle de a puis b. Expliquez le résultat…

Explications :

Lorsqu'une variable est définie dans une fonction, elle est créée lors de l'appel de la fonction, puis détruit lors de la sortie de celle-ci.

La variable b n'a donc une existence que lors de l'appel de la fonction f.

Quant à la variable a, l'instruction « a=20 » se comporte comme la déclaration d'une nouvelle variable, détruite après la sortie de f lorsqu'on appelle la variable, on obtient donc 6.

Une variable définie dans une fonction est appelée ***variable locale****, elle est détruite après la sortie de la fonction. Tout autre variable est appelée* ***variable globale****.*

TD

Exercice 1 :
Que vaut la variable a1 après l'exécution de l'algorithme suivant ?

Variables : a1 (réel), a2, a3 (entiers)
Début
    a1←0.1
    a2←5
    a3←100
    a1←a1+a2/a3
Fin

Exercice 2 :
Que valent les variables x et n après exécution de l'algorithme suivant ?

Variables : x (liste), n (entier)
Début
    x←[1,2]
    Ajouter 3 à x
    n←x[0]+x[1]+x[2]
Fin

Exercice 3 :
Écrire un algorithme demandant à l'utilisateur un nombre entier de trois chiffres, puis affichant la somme des chiffres de ce nombre. Implémenter cet algorithme en Python.

Exercice 4 :
Écrire un algorithme calculant puis affichant

$$\sum_{k=1}^{50} k^2 = 1^2 + 2^2 + \cdots + 50^2$$

Implémentez en Python puis exécuter le programme.

Exercice 5 :
Écrire un algorithme qui génère et affiche la liste des diviseurs d'un nombre entier naturel rentrer par l'utilisateur. Implémenter et exécuter cet algorithme en Python avec le nombre 646.

Exercice 6 :
Quel est le rôle de l'algorithme suivant ?

Variables : k, compt (entiers), texte (chaîne)
Début
    Afficher "Saisir un texte : "

```
        Saisir texte
        k←0
        compt←0
        TantQue k<=longueur(texte)-1 Faire
                Si texte[k]== 'e ' Alors
                compt←compt+1
                FinSi
                k←k+1
        FinTantQue
        Afficher compt
Fin
```

Exercice 7 (en Python):
Écrire un algorithme demandant à l'utilisateur un pseudo d'au moins quatre caractères, jusqu'à ce qu'il soit correct, puis affichant « Bonjour<pseudo> !>.

Exercice 8 (en Python):
Écrire en python une procédure `comparaison` qui affiche le résultat de la comparaison de deux réels a et b. Par exemple, il devra résulter de l'appel de `comparaison(8,19)` l'affichage « 8 < 19 » et de l'appel de **comparaison(12,12)** l'affichage « 12 = 12 ».

Exercice 9 (en Python):
Écrire un programme qui, après la saisie d'un entier naturel, renvoie l'écriture à l'envers de cet entier.
Par exemple, si on rentre le nombre 12 345, le nombre doit renvoyer 54 321.
Ensuite, le programme doit dire si le nombre saisi est un palindrome, c'est-à-dire un nombre se lisant de la même façon dans les deux sens (par exemple, 56 765 est un palindrome).

Exercice 10 (en Python) :
Si on additionne le nombre 548 et son écriture à l'envers 845, on obtient 1393.
Un autre nombre à trois chiffres, plus petit que 548, renvoie aussi 1393. Écrire un programme avec une boucle « while » qui recherche ce nombre.

Exercice 12 (en Python) :
Le programme suivant doit servir à convertir un nombre entier de seconde au format heures, minutes, secondes.
a. Quelles sont les variables utilisées dans ce programme ? De quel type sont-elles ?
b. Quelle variable n'est pas modifié au cours de ce programme ?
c. Compléter les lignes sur lesquels les instructions manquent. Tester votre programme

```
nbsec=int(input('Entrer le nombre total de secondes:'))
```

```
s=nbsec
h=s//3600
s=...
m=...
s=...
print(nbsec,'secondes=',h,'h',m,'min',s,'sec')
```

Exercice 13 (en Python):

Dans cet exercice, on attribue à chaque lettre une valeur égal à son rang dans l'alphabet : A=1, B=2, …, Z=26.

On appelle « poids » d'un mot écrit en majuscules la somme obtenue en multipliant la valeur de chaque lettre par son rendant le mot.

Prenons le mot BRAVO : B=2, R=18, A=1, V=22 et O=15.

Ainsi son poids est $2 \times 1 + 18 \times 2 + 1 \times 3 + 22 \times 4 + 15 \times 5 = 204$.

Écrire un algorithme d'une fonction nommé poids, dont le paramètre est un mot écrit en majuscules, qui renvoie le poids du mot.

Aide : vous pourrez utiliser la fonction `ord()`*, qui renvoie le code ASCII d'un caractère sachant que les codes acquis des lettres majuscules vont de 65, pour un 90, pour Z. Par exemple* `ord('G')=71`*.*

Exercice 14 (en Python)*:

Écrire un algorithme de jeux de hasard dans la règle est la suivante :

on tire au hasard un nombre entier de trois chiffres que l'utilisateur doit deviner.

Tant qu'il ne l'a pas trouvé, on lui indique le nombre de chiffres correct. Une fois trouvé, on affiche son nombre d'essais.

Implémenter en python c'est algorithme en apportant au préalable le module `Random` puis la fonction `randint` (écrire en début de programme l'instruction `import Random` puis l'instruction `from Random import randint`). Le choix d'un nombre entier de trois chiffres au hasard peut s'obtenir avec `randint(100,999)`.

Correction TD :

Exercice 1 :

Que vaut la variable a1 après l'exécution de l'algorithme suivant ?

a1=0.15

Exercice 2 :

Que valent les variables x et n après exécution de l'algorithme suivant ?

x vaut [1,2,3] et n vaut 6.

Exercice 3 :

Écrire un algorithme demandant à l'utilisateur un nombre entier de trois chiffres, puis affichant la somme des chiffres de ce nombre. Implémenter cet algorithme en Python.

Variables : n(chaine), somme(entier)
Début :
    Afficher "Entrer un nombre entier de 3 chiffres : "
    Saisir n
    Somme ←int(n[0]+n[1]+n[2])
    Afficher "La somme des chiffres du nombre ",n, "est ",somme, ". "
Fin

En Python :

```
n=input("Entrer un nombre entier de 3 chiffres : ")
somme=int(n[0])+int(n[1])+int(n[2])
print("La somme des chiffres du nombre ",n, "est ",somme, ". ")
```

Exercice 4 :

Écrire un algorithme calculant puis affichant

$$\sum_{k=1}^{50} k^2 = 1^2 + 2^2 + \cdots + 50^2$$

Implémentez en Python puis exécuter le programme.

Variables : s,k (entiers)
Début :
    s ←0   #initialisation
    Pour k de 1 à 50 **Faire**
        s ←s+k^2
    FinPour
    Afficher s
Fin

En Python :

```
s=0
for k in range(1,51):
    s=s+k**2
print(s)
```

Résultat: 42 925.

Exercice 5 :

Écrire un algorithme qui génère et affiche la liste des diviseurs d'un nombre entier naturel rentrer par l'utilisateur. Implémenter et exécuter cet algorithme en Python avec le nombre 646.

Variables : n,k (entiers), L (liste)
Début :
    L ←[] #initialisation :liste vide
    Afficher "Saisir un nombre entier : "
    Saisir n
    Pour k de 1 à n **Faire**
        Si n%k==0 **Alors**
            Ajouter k à L
        FinSi
    FinPour
    Afficher L
Fin

En Python :

```
L=[] #initialisation de la liste vide
n=int(input("Saisir un nombre entier:"))
for k in range(1,n+1):
      if n%k==0:
          L.append(k)
print(L)
```

Résultat : [1, 2, 17, 19, 34, 38, 323, 646]

Exercice 6 :

Quel est le rôle de l'algorithme suivant ?
Déterminer le nombre de caractère « e »présents dans le texte saisi par l'utilisateur

Exercice 7 (en Python):

Écrire un algorithme demandant à l'utilisateur un pseudo d'au moins quatre caractères, jusqu'à ce qu'il soit correct, puis affichant « Bonjour<pseudo> !>.

```
pseudo=input("Saisir un pseudo d'au moins 4 caractères:")
while len(pseudo)<4:
    pseudo=input("pseudo incorrect, veuillez saisir un pseudo d'au mins 4
caractères:")
print("Bonjour",pseudo,"!")
```

Exercice 8 (en Python):

Écrire en python une procédure `comparaison` qui affiche le résultat de la comparaison de deux réels a et b. Par exemple, il devra résulter de l'appel de `comparaison(8,19)` l'affichage « 8 < 19 » et de l'appel de **comparaison(12,12)** l'affichage « 12 = 12 ».

```
def comparaison(a,b):
    if a<b:
        print(a,"<",b)
    elif a==b:
        print(a,"=",b)
    else:
        print(a,">",b)
```

Exercice 9 (en Python):

Écrire un programme qui, après la saisie d'un entier naturel, renvoie l'écriture à l'envers de cet entier.

Par exemple, si on rentre le nombre 12 345, le nombre doit renvoyer 54 321.

Ensuite, le programme doit dire si le nombre saisi est un palindrome, c'est-à-dire un nombre se lisant de la même façon dans les deux sens (par exemple, 56 765 est un palindrome).

```
nb=input("Entrer un nombre: ")
inv=''
for k in nb:
    inv=k+inv
if nb==inv:
    print(int(nb),'est un palindrome')
else:
    print(int(nb),'n est pas un palindrome')
```

Exercice 10 (en Python) :

Si on additionne le nombre 548 et son écriture à l'envers 845, on obtient 1393.

Un autre nombre à trois chiffres, plus petit que 548, renvoie aussi 1393. Écrire un programme avec une boucle « while » qui recherche ce nombre.

```
k=100 #plus petit nombre entier de 3 chiffres
s=101 #100+001
while s!=1393:
    k+=1
    nb=str(k) #nb est la chaîne associé au nombre k
    inv=nb[2]+nb[1]+nb[0] #inv est la chaîne nb à l'envers
    s=int(k)+int(inv)
print(k)
```

Exercice 12 (en Python) :

Le programme suivant doit servir à convertir un nombre entier de seconde au format heures, minutes, secondes.

a. Quelles sont les variables utilisées dans ce programme ? De quel type sont-elles ?

Le programme utilise 4 variables : nbsec, h, m et s toutes de type entier.

b. Quelle variable n'est pas modifié au cours de ce programme ?

La variable nbsec n'est pas modifiée afin de pouvoir l'afficher à la fin de l'exécution du programme.

c. Compléter les lignes sur lesquels les instructions manquent. Tester votre programme

```
nbsec=int(input('Entrer le nombre total de secondes:'))
```

```
s=nbsec
h=s//3600
s=s%3600
m=s//60
s=s%60
print(nbsec,'secondes=',h,'h',m,'min',s,'sec')
```

Exercice 13 (en Python):

Dans cet exercice, on attribue à chaque lettre une valeur égal à son rang dans l'alphabet : A=1, B=2, …, Z=26.

On appelle « poids » d'un mot écrit en majuscules la somme obtenue en multipliant la valeur de chaque lettre par son rang dans le mot.

Prenons le mot BRAVO : B=2, R=18, A=1, V=22 et O=15.

Ainsi son poids est $2 \times 1 + 18 \times 2 + 1 \times 3 + 22 \times 4 + 15 \times 5 = 204$.

Écrire un algorithme d'une fonction nommé poids, dont le paramètre est un mot écrit en majuscules, qui renvoie le poids du mot.

Aide : vous pourrez utiliser la fonction `ord()`*, qui renvoie le code ASCII d'un caractère sachant que les codes acquis des lettres majuscules vont de 65, pour un 90, pour Z. Par exemple* `ord('G')=71`*.*

```
def poids(mot):
    somme=0
    for k in range(len(mot)):
        somme=somme+(k+1)*(ord(mot[k])-64)
    return somme
```

Attention à bien mettre le mot entre guillemets !

Exercice 14 (en Python)*:

Écrire un algorithme de jeux de hasard dans la règle est la suivante : on tire au hasard un nombre entier de trois chiffres que l'utilisateur doit deviner. Tant qu'il ne l'a pas trouvé, on lui indique le nombre de chiffres correct. Une fois trouvé, on affiche son nombre d'essais.

Implémenter en python cet algorithme en important au préalable le module `Random` puis la fonction `randint` (écrire en début de programme l'instruction `import Random` puis l'instruction `from Random import randint`). Le choix d'un nombre entier de trois chiffres au hasard peut s'obtenir avec `randint(100,999)`.

```
from random import randint
nombre=randint(100,999)
nb=str(nombre)
essai=0
rep=input('devine nb: ')
while rep!=nb:
    nb_bonnes_rep=0
    essai=essai+1
    for k in range(3):
        if nb[k]==rep[k]:
            nb_bonnes_rep+=1
    rep=input(str(nb_bonnes_rep) +' chiffre(s) correct(s), recommence: ')
print('gagné en '+str(essai)+' essai(s)!')
```