Fiche 6	LA BOUCLE POUR	TP Algorithmique

Point de cours :

Quand le nombre d'itérations est connu dès le départ, il s'agit d'une **boucle déterministe**, il est alors conseillé d'utiliser la structure « **Pour** ».

Pseudo-code

```
...
Pour <variable> allant de <première valeur> à <dernière valeur> (par pas de <pas>)
   Instructions
Fin Pour
...
```

La boucle « **Pour** » fait varier la valeur du compteur <variable> entre <première valeur> et <dernière valeur>. Le pas est optionnel et vaut 1 par défaut.

En Python, pour afficher toutes les valeurs de <première valeur> à <dernière valeur>, on peut écrire :

Python

```
...
for i in range (<première valeur>,<dernière valeur> + 1):
   instructions
...
```

Exemple 5

Affichons tous les nombres entiers de 1 à 5 dans l'ordre croissant.

Algorithme : 1 à 5 Pour

```
Variable numérique : i
DEBUT
   Pour i allant de 1 à 5
      Afficher i
   Fin Pour
FIN
```

Python version 1

```
for i in range (1,6) :
   print(i)
```

Python version 2

```
# Par défaut la <première valeur> est 0
for i in range (5) :
   print(i + 1)
```

Exemple 6 : Affichons des étoiles

Algorithme : étoile1

Variable numérique : *i*
DEBUT
  Pour *i* **allant de** 1 **à** 4
  Afficher « * »
  Fin Pour
FIN

Affichage obtenu
* * * *

Algorithme : étoile2

Variables numériques : *i*, *j*
DEBUT
  Pour *j* **allant de** 1 **à** 4
  Pour *i* **allant de** 1 **à** 4
  Afficher « * »
  Fin Pour
  Retour à la ligne
  Fin Pour
FIN

Affichage obtenu
* * * * * * * * * * * * * * * *

Algorithme : étoile3

Variables numériques : *i*, *j*
DEBUT
  Pour *j* **allant de** 1 **à** 4
  Pour *i* **allant de** 1 **à** *j*
  Afficher « * »
  Fin Pour
  Retour à la ligne
  Fin Pour
FIN

Affichage obtenu
* * * * * * * * * *

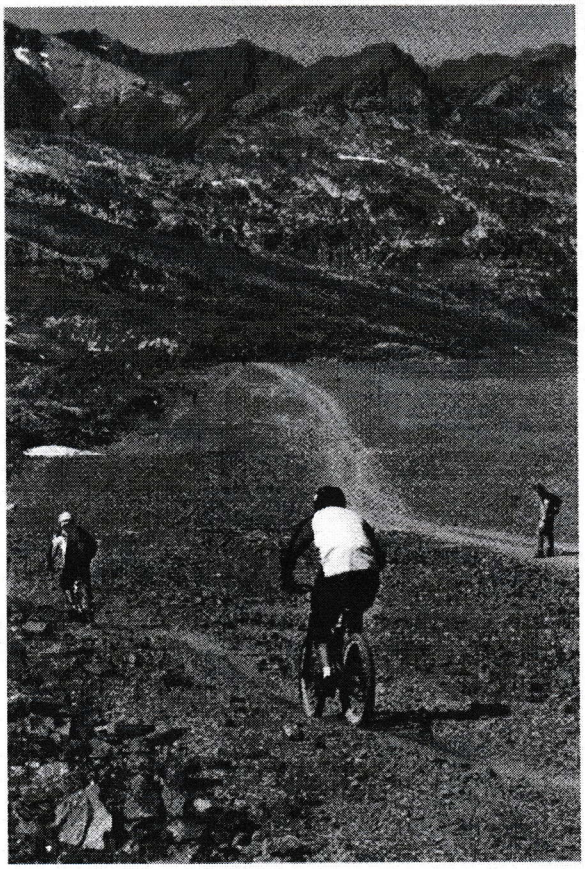
Application 6 : entraînement

Un cycliste souhaite s'entraîner pour une compétition. Il prépare un programme d'entraînement de 3 semaines. Le premier jour, il parcourt 30 km puis il décide d'augmenter la distance parcourue de 10 km chaque jour.

1. Écrire un programme qui calcule et affiche le nombre de kilomètres parcourus le dixième jour d'entraînement. De même pour le dernier jour d'entraînement.

2. Modifier ce programme pour qu'il calcule le nombre total de kilomètres parcourus durant ce programme d'entraînement.

3. Modifier ce programme pour qu'il calcule et affiche le nombre de kilomètres parcourus semaine par semaine.



Application 7 : compte à rebours

Écrire un programme demandant à l'utilisateur de saisir un nombre entier positif *n* et qui affiche tous les nombres entiers dans l'ordre décroissant de *n* à 0.

Indication Python

 # Avec la structure **for...in range...**, *par défaut, le pas vaut 1.*
On peut le modifier, par exemple, la commande :
***for** i **in range**(1,10,2) :*
    ***print**(i)*
affiche les nombres entiers entre 1 et 9 par pas de 2, c'est-à-dire 1, 3, 5, 7, 9.

Application 8 : jeu de dés

Un jeu consiste à lancer 10 dés non truqués à 6 faces et à faire la somme des chiffres obtenus. Si la somme est paire, l'utilisateur gagne 1 point, si la somme est impaire, l'utilisateur perd 2 points. Le jeu affiche la cagnotte du joueur sachant que le joueur démarre la partie avec 10 points.

1. Écrire un algorithme qui simule une partie.

2. Comment modifier cet algorithme pour qu'il simule 5 parties successives ?

3. Comment modifier cet algorithme pour que les parties continuent tant que le joueur dispose d'une cagnotte strictement supérieure à 0 ? (On pourra afficher le détail des parties effectuées.)

DEVOIR MAISON VACANCES TOUSSAINT : « Le nombre secret »

Le nombre secret est un jeu au cours duquel le joueur doit deviner quel est le nombre secret choisi par le maître du jeu.

Ce nombre est un nombre entier compris entre **1** et **10 000**.

À chaque proposition du joueur, le maître du jeu indique si le nombre secret a été trouvé ou bien s'il est plus grand ou plus petit.

A. Première approche

On cherche à concevoir et implémenter des algorithmes **où la machine joue le rôle de maître du jeu.**

1. Travail par écrit

On considère l'algorithme A1 suivant :

Algorithme A1

```
DEBUT
1.  nombresecret ← hasard(1,10 000)
2.  Saisir proposition
3.  Si (proposition > nombresecret) alors
4.     Afficher « Le nombre secret est plus petit »
5.  Sinon
6.     Si (proposition < nombresecret) alors
7.        Afficher « Le nombre secret est plus grand »
8.     Sinon
9.        Afficher « Bravo ! Vous avez trouvé le nombre secret ! »
10.    Fin Si
11. Fin Si
FIN
```

a. Lister les variables utilisées par l'algorithme A1 et leur type.

b. Quel est le rôle de la ligne 1 ?

c. Quelle est la condition qui doit être vérifiée pour effectuer l'instruction ligne 9 ?

d. Quel est le rôle de cet algorithme ?

e. Écrire un nouvel algorithme A2 qui, à partir d'un nombre secret entre **1** et **10 000**, demande au joueur de faire des propositions **tant qu'il n'a pas trouvé** et en indiquant, à chaque fois, si le nombre secret est plus petit ou plus grand. L'algorithme devra indiquer à la fin en combien d'essais le nombre secret a été trouvé (on pourra mettre en place un compteur).

f. Modifier l'algorithme A2 en un algorithme A3 pour qu'il impose, en plus, que le nombre secret soit trouvé en **15** essais maximum.

2. Travail sur poste informatique

Implémenter et tester les algorithmes A1, A2 et A3.

Indication Python

La commande **randint(a,b)** *choisit un nombre entier au hasard entre a et b*

Pour pouvoir utiliser cette commande, vous devez saisir dans l'éditeur :

« from random import* »

B. Pour les braves ! Le nombre secret, le retour !

On cherche à concevoir et implémenter un algorithme où l'utilisateur est le maître du jeu. C'est l'utilisateur qui choisit le nombre secret, qu'il ne communique jamais à la machine. Elle doit trouver le nombre secret en **15** essais maximum.

1. Travail par écrit

a. Par quel nombre la machine devrait-elle commencer pour être efficace ?

b. Si cette première proposition est trop petite par rapport au nombre secret, quel va être le deuxième nombre proposé ? Et si elle est trop grande ?

Pour trouver le nombre secret, il est judicieux de procéder par dichotomie, c'est-à-dire en prenant le milieu de l'intervalle où est caché le nombre secret. Par exemple, si on sait qu'il est compris entre 250 *et* 375, *la proposition sera :* Partie Entiere$\left(\frac{250+375}{2}\right) = 312$ *etc.*

On considère l'algorithme B1 suivant :

Algorithme B1

```
DEBUT
  Saisir min_possible
  Saisir max_possible
  proposition ← PartieEntiere((min_possible + max_possible) /2)
  Afficher proposition
FIN
```

c. Lister les variables de l'algorithme B1 et leur type.

d. Quel est le rôle de cet algorithme ?

e. Écrire un algorithme B2 qui affiche la proposition de la machine, puis où l'utilisateur dit si le nombre secret est plus petit, plus grand ou égal à cette proposition. La machine doit ensuite, selon la réponse de l'utilisateur, lui faire une nouvelle proposition.

f. Écrire un algorithme B3 qui demande à la machine des propositions tant qu'elle n'a pas trouvé le nombre secret.

2. Travail sur poste informatique

a. Implémenter et tester les algorithmes B1, B2 et B3.

b. Modifier le programme B3 pour qu'il impose à l'ordinateur de trouver en 15 essais maximum.

Remarque

Grâce à la méthode par dichotomie, la machine trouvera à coup sûr le nombre secret en 14 essais maximum ($2^{14} = 16\,384$).