Fiche 5	LA BOUCLE TANT QUE	TP Algorithmique

Avant de commencer :

A retenir :

Une boucle est une structure itérative, c'est à dire qu'elle permet d'exécuter plusieurs fois de suite une même séquence d'instructions.
Il existe trois types de boucles en algorithmique :

- Tant Que
- Répéter ... jusqu'à
- Pour

En Python, on ne retrouve que deux structures « TantQue » et « Pour ». Nous ne travaillerons donc que sur ces deux là.

Commençons donc, comme le titre l'indique par la boucle « Tant Que ».

Point de cours :

Pseudo-code	Python

...
Tant que *condition*
  instructions
Fin Tant que
...
	...
	while(*condition*) :
	  instructions
	...

Lorsque l'on rencontre l'instruction « **Tant que** » dans l'exécution d'un algorithme, on évalue la condition :
- si elle est vérifiée, on effectue les instructions ;
- si elle n'est pas réalisée, on sort de la boucle et on passe à l'instruction située après le **Fin Tant que**.

Exemple 1

Affichons tous les nombres entiers de 1 à 5 dans l'ordre croissant.
On initialise *i* à 1.
Tant que *i* est inférieur ou égal à 5, on affiche la valeur de *i* et on recommence.
i est un compteur.

Algorithme : 1 à 5 Tant que

Variable numérique : *i*
DEBUT
  i ← 1
  Tant que *i* ≤ 5
  Afficher *i*
  i ← *i* + 1
  Fin Tant que
FIN

Python : 1 à 5 Tant que

i = 1
while *i* <= 5 :
  print(*i*)
  i = *i* + 1

Exemple 2

L'utilisation de la structure « **Tant que** » peut aboutir à une exécution sans fin. Dans les deux cas suivants, la boucle est infinie.

Algorithme : boucle infinie 1
Variables numériques : *i*, *a* **DEBUT** *i* ← 0 *a* ← 0 **Tant que** *i* < 2 *a* ← *a* + 1 **Fin Tant que** **FIN**

Algorithme : boucle infinie 2
Variable numérique : *i* **DEBUT** *i* ← 7 **Tant que** *i* ≠ 20 *i* ← *i* + 3 **Fin Tant que** **FIN**

Dans le premier cas, la valeur de la variable *i* n'est pas modifiée à l'intérieur de la boucle.

Dans le deuxième cas, la valeur de la variable *i* est bien modifiée mais la condition d'exécution de boucle est toujours vérifiée.

CE QU'IL FAUT SAVOIR

La boucle « **Tant que** » est une **boucle non déterministe**, on ne sait pas forcément à l'avance combien d'itérations seront effectuées. Il arrive qu'elle ne soit pas exécutée lorsque la condition n'est pas vérifiée.

Il existe un deuxième type de structure itérative non déterministe, très proche du « **Tant que** » : c'est la structure « **Répéter... jusqu'à** ».

Pseudo-code
... **Repeter** *instructions* **Jusqu'à** *condition* ...

Python
En Python, cette structure n'existe pas mais on peut utiliser une structure équivalente avec le **while**.

La boucle répète l'action jusqu'à ce que la condition soit vérifiée. Avec cette structure, il y a obligatoirement au moins une itération effectuée, contrairement à la boucle « **Tant que** ».

Exemple 3

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

Algorithme : 1 à 5 Répéter... jusqu'à
Variable numérique : *i* **DEBUT** *i* ← 1 **Répéter** **Afficher** *i* *i* ← *i* + 1 **Jusqu'à** *i* > 5 **FIN**

Cet algorithme fournit le même affichage que l'algorithme vu dans l'exemple 1 avec la boucle « **Tant que** ».

Exemple 4 : Contrôle de saisie

Algorithme : contrôle de saisie 1

Variables numériques : *n*
DEBUT
  Répéter
  Afficher « Saisir un entier naturel »
  Saisir *n*
  Jusqu'à PartieEntiere(*n*) = *n*
FIN

Algorithme : contrôle de saisie 2

Variables numériques : *x*
DEBUT
  Répéter
  Afficher « Saisir un nombre strictement positif »
  Saisir *x*
  Si *x* ≤ 0 **alors**
  Afficher « J'ai dit STRICTEMENT POSITIF ! »
  Jusqu'à *x* > 0
FIN

Il s'agit d'un usage très courant de la structure « **Répéter... jusqu'à** ».

Applications :

- Application 1 : pliage d'une feuille

Une feuille de papier ordinaire mesure environ 0,16 mm d'épaisseur. On décide de la plier en deux, puis en 4, puis en 8, etc.

Combien faut-il de pliages pour arriver à une épaisseur supérieure à la hauteur de la Tour Eiffel (324 m) ?

Écrire un algorithme qui permet de répondre à cette question.

- Application 2 : jeu de petits chevaux

Le jeu des petits chevaux est un jeu de société. Pour gagner il faut être le premier à amener ses pions (des chevaux) à la fin du parcours. Le jeu se joue avec un dé à 6 faces. Pour pouvoir sortir un cheval de son écurie, le joueur doit faire un 6.

Écrire un algorithme qui simule les tirages successifs du dé jusqu'à ce que le cheval puisse sortir de l'écurie.

- Application 3 : mot de passe

Pour accéder à un forum, un étudiant doit se connecter à l'aide d'un mot de passe. Tant qu'il ne saisit pas le bon mot de passe, l'ordinateur lui demande de le saisir à nouveau.

1. Écrire un algorithme qui demande à l'étudiant son mot de passe autant de fois que nécessaire pour accéder au forum.

2. Modifier l'algorithme précédent pour que l'étudiant soit limité à trois propositions. Pour cela on peut mettre en place un compteur.