

1INDF chapitre 7 : boucle 'répéter X fois'

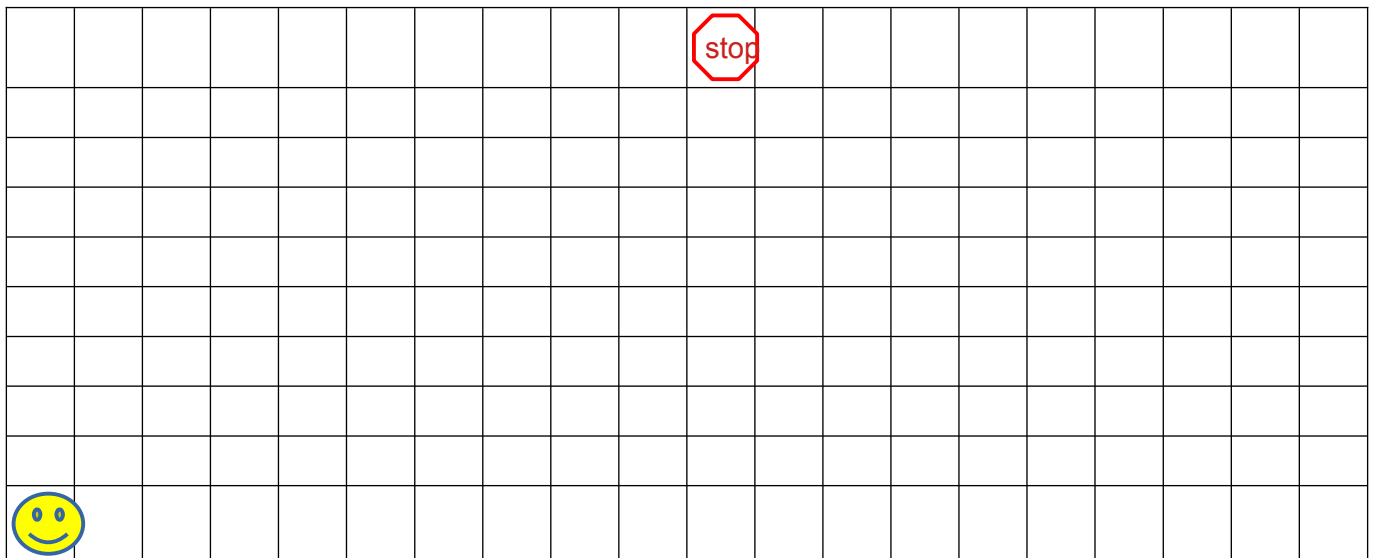
Nom:	Groupe:	Date:
------	---------	-------

Une boucle est une structure de contrôle destinée à exécuter une séquence d'instructions (« bloc ») plusieurs fois de suite.

- La boucle « tant que » (« while » en anglais) répète l'exécution d'une séquence d'instructions tant que la condition d'un test reste vraie ;
- la boucle « répéter X fois » répète l'exécution d'une séquence d'instructions un certain nombre de fois. Ce nombre de fois doit être connu à l'avance.

Introduction

Dans cet exercice,  doit arriver à , mais cette fois il n'y a pas de chemin,  peut se déplacer sur n'importe quelle case. Les actions possibles sont : « avancer d'un pas vers l'est », « avancer d'un pas vers l'ouest », « avancer d'un pas vers le nord » et « avancer d'un pas vers le sud ».



Questions :

- a) écrivez le pseudo-code correspondant, contenant l'instruction « répéter X fois » afin de permettre à  d'arriver à  avec 4 instructions au total ;
- b) sur une feuille à part, faites le logigramme correspondant.

Réponse :

```
répéter 10 fois :  
    avancer d'un pas vers l'est  
répéter 9 fois :  
    avancer d'un pas vers le nord
```

A retenir

En informatique, le fait de « répéter X fois » une instruction se traduit par une boucle « for ».

Une boucle « for » permet de répéter une instruction un nombre X de fois, où X est défini par le programmeur. La boucle « for » nécessite donc de savoir d'avance combien de fois une instruction doit être exécutée.

La commande « range »

La commande « `range(n)` » génère une séquence de *n* nombres. Le premier nombre généré est 0, le deuxième est 1, le troisième est 2, et le dernier est *n*-1.

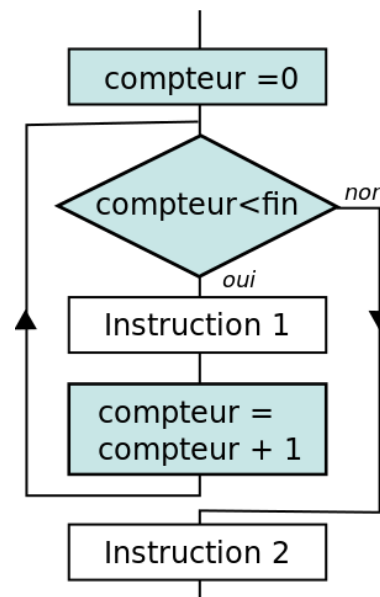
Exemple : `range(10)` produit les nombres de 0, 1, 2, 3, ... , 9, mais pas le nombre 10.

La commande « `range(n)` » permet de définir le point de départ, le point d'arrivée et la taille des sauts, la syntaxe étant : `range(start, stop, step)`.

Exemple : `range(11, 21, 3)` produit les nombres 11, 14, 17, 20.

Le logigramme de droite montre le fonctionnement d'une boucle « répéter X fois » :

1. `compteur` est mis à 0 ;
2. tant que le compteur n'a pas atteint « fin » (« fin » est le nombre de fois où la boucle doit se répéter, donc `fin = X`) :
 - i. l'instruction 1 est exécutée ;
 - ii. `compteur` est incrémenté de 1.
3. lorsque `compteur == fin`, la boucle se termine et l'instruction 2 est exécutée.



Le pseudo-code Python correspondant à l'exemple 😊 doit arriver à  devient donc:

1. `for x in range(10)`
2. avancer d'un pas vers l'est
3. `for x in range(9)`
4. avancer d'un pas vers le nord

Exemple de programme en Python avec une boucle « for » et la commande « range » :

```
for i in range(8):  
    print(i)  
  
print("fin")
```

1I NDF chapitre 7, TP6: boucles "for"

Nom:	Groupe:	Date:
------	---------	-------

Exercice 1 :

- sur une feuille à part, prédiriez l'affichage des programmes ci-dessous ;
- écrivez puis exécutez les programmes avec Thonny pour vérifier vos réponses ;

Programme 1	Programme 2
<pre>for i in range(8): print(i)</pre>	<pre>for j in range(55, 99, 8): print(j)</pre>
Programme 3	Programme 4
<pre>for k in range(99, 55, 8): print(k)</pre>	<pre>for m in range(99, 55, -8): print(m)</pre>

Exercice 2 : écrivez un programme "TP7_exo2_NDF.py" affichant à l'écran...

- 12 fois votre prénom, en colonne :

```
prénom  
prénom  
prénom  
...
```

- 12 fois votre prénom et le numéro de la ligne, en colonne :

```
0. prénom  
1. prénom  
...
```

- 12 fois votre prénom et le numéro de la ligne sur une ligne, grâce à la syntaxe

```
print(<texte>, end=" ") :
```

```
0. prénom, 1. prénom, 2. prénom, 3. prénom, ...
```

- les nombres de 5 à 20 sur une ligne :

```
5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20
```

- les nombres de 17 à 53 en ligne, en allant de 3 en 3 :

```
1 4 7 10 13 16 19 22 25 28
```

- les 20 premiers multiples de 7 → *vous devez utiliser l'opérateur de multiplication*

```
7 14 21 28 35 42 49 56 63 70 77 84 91 98 105 112 119 126 133 140
```

- les 17 premières puissances de 2

- un compte à rebours, de 10 à 1;

- la table de multiplication de n'importe quel nombre, par exemple :

```
1 x 7 = 7  
2 x 7 = 14  
3 x 7 = 21  
...
```

Exercice 3 : refaites le programme « TP3_exo1_devinette » pour qu'il effectue les tâches suivantes :

Version 1, à sauvegarder sous « TP6_devinette_v1_NDF » :

- demander à l'utilisateur d'entrer un nombre entre 1 et 100 ;
- à l'aide d'une structure « si, sinon si, sinon », afficher « Bravo ! » lorsque l'utilisateur entre le bon nombre, ou les indices « trop petit » ou « trop grand », selon le cas ;
- donner 4 essais à l'utilisateur en utilisant une boucle « for ». *Dans cette première version du programme, l'utilisateur doit entrer quatre nombres, même s'il trouve la réponse dès le premier essai.*

Version 2 : à sauvegarder sous « TP6_devinette_v2_NDF » :

- le nombre d'essais est assigné à la variable `nb_essais` ;
- à chaque essai, le programme affiche le nombre d'essais restants, et annonce « Dernier essai ! » lorsqu'il ne reste qu'un seul essai.

Affichage de « TP6_devinette_v1_NDF » :	Affichage de « TP6_devinette_v2_NDF » :
<pre>***** Devine mon nombre secret ! ***** Entrez un nombre au clavier: 44 Trop petit ! Entrez un nombre au clavier: 88 Bravo ! Tu as deviné ! Entrez un nombre au clavier: 33 Trop petit ! Entrez un nombre au clavier: 99 Trop grand !</pre>	<pre>***** Devine mon nombre secret ! ***** Il vous reste 4 essais. Entrez un nombre au clavier: 44 Trop petit ! Il vous reste 3 essais. Entrez un nombre au clavier: 99 Trop grand ! Il vous reste 2 essais. Entrez un nombre au clavier: 77 Trop petit ! Dernier essai !! Entrez un nombre au clavier: 88 Bravo ! Vous avez deviné !</pre>

Version3 (pour les curieux) : utilisez la commande « break » pour terminer le programme dès que l'utilisateur a deviné le nombre. **IMPORTANT** : *cette commande n'est pas recommandée, un programme contenant un « break » est considéré comme étant un mauvais programme.*

Exercice 4 : Elisa reçoit 5,25 chf chaque semaine comme argent de poche, qu'elle met sagement dans sa tirelire. Ecrivez un programme "TP6_exo4_NDF" qui effectue les tâches suivantes:

- initialiser le montant d'argent dans la tirelire à 0 au départ ;
- afficher combien d'argent Elisa possède à chaque semaine, pendant 8 semaines, de la façon suivante:

```
Après 1 semaine(s), la tirelire contient 5.25 chf.
Après 2 semaine(s), la tirelire contient 10.5 chf.
...
```

- c) Elisa se permet d'aller au cinéma toutes les trois semaines, ce qui lui coûte 14,90 chf.
Ajoutez des lignes de commande afin d'obtenir l'affichage suivant :

```
Après 1 semaine(s), la tirelire contient 5.25 chf.  
Après 2 semaine(s), la tirelire contient 10.5 chf.  
Après 3 semaine(s), la tirelire contient 15.75 chf.  
Elisa va au cinéma !  
La tirelire contient maintennt 0.8499999999999996 chf.  
Après 4 semaine(s), la tirelire contient 6.1 chf.  
Après 5 semaine(s), la tirelire contient 11.35 chf.  
Après 6 semaine(s), la tirelire contient 16.6 chf.  
Elisa va au cinéma !  
La tirelire contient maintennt 1.7000000000000001 chf.  
Après 7 semaine(s), la tirelire contient 6.9500000000000001 chf.  
Après 8 semaine(s), la tirelire contient 12.2000000000000001 chf.
```

Note : Python est très peu précis avec les nombres décimaux, c'est pourquoi il est préférable d'utiliser des nombres entiers tant que faire se peut.
