Nom :	Prénom :	Classe :

NSI 1re — Boucle non bornée (boucle while)

Objectifs :
- Savoir identifier et utiliser une boucle non bornée.
- Savoir créer une boucle bornée.

Introduction

Nous avons déjà parlé des boucles bornées[1] — également appelées *boucles for* — une structure qui permet de répéter une série d'instructions **un nombre de fois déterminé** à l'avance.

Dans ce cours, nous allons aborder un autre type de boucle : la boucle non bornée — aussi appelée *boucle while*.
Une **boucle non bornée** est une structure qui permet de répéter une série d'instructions **tant qu'une affirmation** est **vraie**.

Prenons tout de suite un exemple de boucle non bornée dans la vie de tous les jours :

> Tant que je n'ai pas retrouvé mes clés :
>
> → Je cherche dans mes poches.
> → Je cherche sous les sièges de la voiture.
> → Je cherche dans mon sac.
> → Je cherche sur mon bureau.

Dans cet exemple, nous avons une **affirmation** « *je n'ai pas retrouvé mes clés* »
arrêtons-nous un instant sur cette affirmation, qui pourrait être réécrite en « *clés trouvées == Faux* »

Et tant que cette **affirmation** est **vraie**, le programme exécute les instructions contenue dans la boucle : je cherche dans mes poches, je cherche sous les sièges de la voiture, etc.

Voici à présent comment nous pourrions transcrire cet exemple en langage Python :

```
1  cles_trouvees = False
2  while cles_trouvees == False:
3      print('Je cherche dans mes poches')
4      print('Je cherche sous les sièges')
5      print('Je cherche dans mon sac')
6      print('Je cherche sur mon bureau')
```

Notons qu'une boucle non bornée s'écrit avec `while` suivi d'une affirmation, ici `cles_trouvees == False`

Tant que cette affirmation est vraie (donc ici que `cles_trouvees` vaut `False`),
alors le bloc d'instructions contenu dans la boucle est exécuté.

Observons également ici que le bloc d'instructions sera exécuté un nombre infini de fois…

1 Voir le cours « Boucle bornée (boucle for) »
ainsi que le cours « Boucle bornée (boucle for) #2 : utilisations de range() ».

Spécificités d'une boucle non bornée

Généralement, une boucle non bornée, possède :
- Une **affirmation/condition de départ** : ce qui permet à la boucle d'effectuer au moins une **itération**.
- Une **condition d'arrêt** : ce qui permet à la boucle de se terminer.

S'il n'existe pas de condition d'arrêt, alors c'est une boucle qui ne se termine jamais,
on parle alors d'une **boucle infinie** !

Ce serait mentir que de dire que nous n'avons jamais été la victime d'une boucle infinie dans un de ses programmes ! La boucle infinie est le *bug numéro un*, lié à une boucle non bornée !

➤ Exercice 1 : enquêter

Observez chacune des boucles non bornées ci-dessous, puis déterminez-en :
- La **condition de départ** : est-elle présente, oui ou non ? Justifiez.
- La **condition d'arrêt** : est-elle présente, oui ou non ? Justifiez.

N°	Code Python	Condition de départ	Condition d'arrêt

1	`age = 10`
	`while age < 18:`
	`  print('mineur(e)')`
	`  age = age + 1`
		La boucle se lance car age vaut 10 et la condition de départ est age < 18	Condition d'arrêt valide car age finit par atteindre 18, ce qui stoppe la boucle.

2	`clients = 0`
	`while clients < 50:`
	`  print('boutique fermée')`
		La boucle se lance car clients vaut 0 et la condition de départ est clients < 50	Condition d'arrêt non valide car la valeur de clients ne change pas dans le bloc d'instruction de la boucle.

3	`objets = 3`
	`while objets == 0:`
	`  print('panier vide')`
		La boucle ne se lance pas car objets vaut 3 et la condition de départ est objets == 0	Condition d'arrêt non valide car la valeur de objets ne change pas dans le bloc d'instruction de la boucle.

4	`age = 0`
	`while age < 18:`
	`  age = int(input('Âge ? '))`
	`print('majeur(e)')`
		La boucle se lance car age vaut 0 et la condition de départ est age < 18	Condition d'arrêt valide, dans la mesure où on estime que l'utilisatrice / utilisateur finira par saisir une valeur > 18

➤ Exercice 2 : déduire

Quel est le risque principal des programmes précédents n'ayant pas de condition d'arrêt valable ?

Le risque principal est d'avoir un programme qui ne s'arrête jamais, à cause d'une boucle dite infinie.

➤ Exercice 3 : débugger

Le programme suivant contient une boucle infinie.
Écrivez en dessous votre proposition de **modification** de son **bloc d'instructions** permettant de le "réparer".

```
1  a = 0
2  while a < 10:
3      print(a)
```

```
a = 0
while a < 10 :
   print(a)
   a = a + 1        # ou encore : a += 1
```

➤ Exercice 4 : conversion d'une boucle bornée en boucle non bornée

Il est possible de s'amuser à convertir une boucle bornée (boucle for) en boucle non bornée (boucle while).

Par exemple :

Python, en "mode" boucle bornée	Python, en "mode" boucle non bornée

`for i in range(5):`
`    print(i)`
	`i = 0`
	`while i < 5:`
	`    print(i)`
	`    i = i + 1`

Comme nous le voyons, le programme contient bien plus de lignes en "mode" boucle non bornée.
C'est que l'intérêt d'une boucle non bornée est ailleurs, nous y reviendrons juste après.
En attendant, entraînons-nous à effectuer quelques autres conversions :

Python, en "mode" boucle bornée	Python, en "mode" boucle non bornée (à compléter)

`for age in range(1, 10):`
`    print(age)`
	`age = 1`
	`while age < 10 :`
	`   print( age )`
	`   age = age + 1        # ou age += 1`

`for e in range(3, -1, -1):`
`    print(e)`
`print('Partez!')`
	`e = 3`
	`while e > -1 :`
	`   print( e )`
	`   e = e - 1        # ou e -= 1`
	`print( 'Partez!' )`

Intérêt d'une boucle non bornée

Une boucle non bornée est utile dès que le nombre de répétitions n'est pas connu à l'avance ou dépend de données externes (saisie utilisateur, arrivée d'un événement, convergence d'un calcul, etc.), alors qu'une boucle bornée est adaptée quand on sait dès le départ combien de fois répéter le bloc d'instructions.

Ainsi, la boucle non bornée est plus adaptée aux situations d'attente : lecture jusqu'à la fin d'un fichier, demandes successives à l'utilisatrice ou utilisateur, boucle principale d'un jeu, etc.

Prenons un exemple :

Nous souhaitons créer un programme dans lequel l'utilisatrice/utilisateur doit presser la touche `x` pour quitter. En d'autres termes, tant que la touche `x` n'est pas pressée, on lui demande une saisie (avec input).

Voici une implémentation Python possible :

```
1  reponse = False
2  while reponse != 'x':
3      reponse = input('? ')
```

Dans ce programme, tant que la variable `reponse` ne contient pas `'x'` (ligne 2),
alors on stocke une saisie utilisatrice/utilisateur dans `reponse` (ligne 3).

➤ Exercice 5 : créer une boucle non bornée

Créez un programme dans lequel on demande à l'utilisatrice/utilisateur un nombre positif.
Tant que ce nombre n'est pas positif (supérieur à 0), le programme pose la même question.

```
nbre = -1

while nbre < 0 :
   nbre = int( input('Saisir un nombre positif svp ? ') )
```

Pour aller plus loin

Il est possible de récupérer un nombre entier aléatoire en Python, grâce au code suivant :

```
# Import de la fonction randint()
# depuis le module «random» de Python
from random import randint

# Entier aléatoire de 1 (inclus) à 6 (inclus)
a = randint(1, 6)
```

Sur une feuille à part, créez le programme dans lequel il fau
et 6 (inclus) choisi par l'ordinateur. Tant que l'utilisatrice/uti
programme lui demande un essai.

```
# Une correction possible :

from random import randint
ordi = randint(1, 6)
humain = 0
while humain != ordi:
   humain = int(input('Nbre entre 1 et 6 svp ? '))
print('Bravo!')
```