Boucles non bornées

A Définition d'une boucle non bornée

- Dans un certain nombre de cas, il est indispensable de répéter des instructions sans savoir à l'avance combien de fois on les répète. La boucle **for** est alors inefficace. On utilise un autre type de boucle : les **boucles non bornées**. Les instructions sont alors répétées **tant** qu'une condition est vérifiée. On parle ainsi de **boucle tant que** ou de **boucle while**.
- La boucle s'arrête quand la condition n'est plus vérifiée.

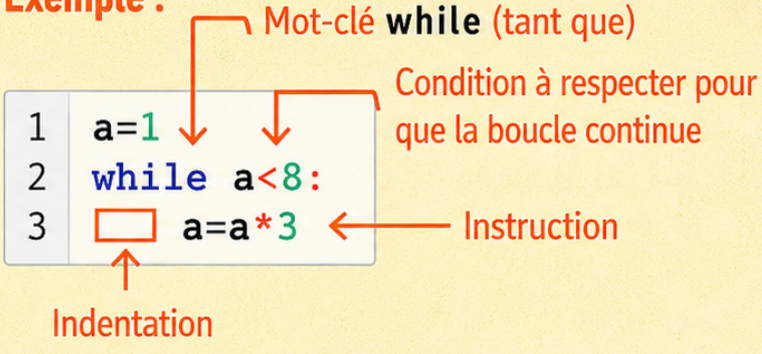
Boucles non bornées

B Boucle « while »

- Pour répéter un ensemble d'instructions tant qu'une condition est vérifiée, la syntaxe est la suivante :

```
while condition:
    instruction1
    instruction2
    …
```

Exemple :




```
1  a=1
2  while a<8:
3      a=a*3
```

Ligne 2 : le mot-clé **while** indique la création d'une boucle while et précède la condition : tant que a < 8.

Ligne 3 : l'instruction à répéter, a=a*3, est **indentée** (décalée vers la droite).

Ce programme multiplie la valeur de la variable a par 3 tant qu'elle est inférieure à 8. La variable a vaut donc successivement 1, 3 et 9 avant l'arrêt de la boucle.

Valeur de *a* ligne 2	Condition	Tour de boucle	Action ligne 3	Valeur de *a* ligne 3
1	1 < 8 Vrai	1	affectation a=1*3	3
3	3 < 8 Vrai	2	affectation a=3*3	9
9	9 < 8 Faux	–	–	9

Boucles non bornées

C Utilisation du compteur

Il est parfois utile de compter le nombre de répétitions effectuées dans une boucle **while**. Dans ce cas, on peut définir un compteur. Ce compteur est initialisé à la valeur 0 et augmente de 1 à chaque tour de boucle.

Exemple :

```
1  a=1
2  compteur=0
3  while a<100:
4      a=a*3
5      compteur=compteur+1
```

Ligne 2, on initialise le compteur à la valeur 0.

Ligne 5, le compteur augmente de 1 à chaque tour de boucle.

À la fin de l'exécution de ce programme, le compteur est égal à 5.

Boucles non bornées

QCM

Pour chaque proposition, cocher la bonne réponse.

1 Combien de tours de boucle sont effectués dans le programme suivant ?

```
1 x=0
2 while x<3:
3     x=x+2
```

a. ☐ 0
b. ☐ 1
c. ☐ 2

2 Quelles sont les valeurs affichées dans le programme suivant ?

```
1 a=2
2 while a!=8:
3     print(a)
4     a=a*2
```

a. ☐ 2, 4, 8
b. ☐ 1, 2, 3
c. ☐ 2, 4

3 Quelle condition faut-il écrire pour que la boucle s'arrête quand x ≥ 9 ?

```
1 x=1
2 while .....:
3     x=x+3
```

a. ☐ x > 9
b. ☐ x < 9
c. ☐ x > =9

4 Quelle est la valeur affichée ?

```
1 x=1
2 while x<3:
3     x=4*x
4 print(x)
```

a. ☐ 1 b. ☐ 4 c. ☐ 16

5 À quelle condition cette boucle s'arrête-t-elle ?

```
while a==5:
```

a. ☐ si *a* est égal à 5.
b. ☐ si *a* est différent de 5.
c. ☐ si *a* est nul.

6 Pour quelle valeur de *a* la boucle while fait-elle exactement 3 tours ?

```
while a<4:
    a=a+1
```

a. ☐ 0 b. ☐ 1 c. ☐ 2

Boucles non bornées

Correction

Question 1 : Combien de tours de boucle sont effectués dans le programme suivant ?

```
1  x=0
2  while x<3:
3      x=x+2
```

Explications :

1. Au départ, `x=0`.
2. Comme `0<3`, la boucle s'exécute une première fois.
3. Après le 1er tour, `x` devient `2`.
4. Comme `2<3`, la boucle s'exécute une deuxième fois.
5. Après le 2e tour, `x` devient `4`.
6. Comme `4<3` est faux, la boucle s'arrête.

Conclusion : la boucle effectue 2 tours.

Bonne réponse : c. 2

Boucles non bornées

EXERCICE 1 • *Comprendre un programme*

On considère le programme suivant :

```
1  x=2
2  while x<5:
3      x=3*x/2
```

1. Quand la boucle **while** va-t-elle s'arrêter ?

.....

.....

2. Quelle instruction est répétée à chaque tour ?

.....

3. Compléter le tableau suivant donnant les valeurs prises successivement par x dans la condition du **while** et la valeur de x après le **while** (V : Vrai, F : Faux).

Valeur de x	_	2	3	.....	.....
Condition $x < 5$		V	V	.....	.....
Valeur finale de x	2	3	.....	.....	_
Tour n°	_	1	.....	.....	_

Boucles non bornées

EXERCICE 2 • *Comprendre un programme*

On considère le programme suivant :

```
1  mot="Hello"
2  while len(mot)<11:
3      mot=mot+mot
4  print(mot)
```

1. Combien de tours de boucle sont réalisés ?

.....

2. Quelle est la valeur de la variable mot à la fin du programme ?

.....

Boucles non bornées

EXERCICE 3 • *Corriger un programme*

Lilou a écrit le programme suivant pour afficher tous les nombres impairs inférieurs à 50.

```
1  u=1
2  while u<50:
3      print(u+2)
```

1. Le programme ne s'arrête pas, expliquer pourquoi.

.....

.....

.....

.....

2. Corriger le programme de Lilou :

```
1  .....
2  .....
3  .....
4  .....
```

Boucles non bornées

Correction

Exercice 3 : Corriger un programme

```
1  u=1
2  while u<50:
3      print(u+2)
```

1. Le programme ne s'arrête pas, expliquer pourquoi.

La variable u n'est jamais modifiée dans la boucle.
Elle reste donc égale à 1. La condition u<50 reste toujours vraie : la boucle ne s'arrête jamais.
Comme u vaut toujours 1, le programme affiche toujours 3.

Conclusion : c'est une boucle infinie.

2. Corriger le programme de Lilou :

```
1  u=1
2  while u<50:
3      print(u)
4      u=u+2
```

Cette fois, on affiche d'abord la valeur de u,
puis on augmente u de 2 à chaque tour.
Le programme affiche donc 1, 3, 5, ..., 49 puis s'arrête.

Programme corrigé correctement.

Boucles non bornées

EXERCICE 4 • *Compléter un programme*

SNT
Info embarquée

Compléter le programme suivant qui demande le code d'accès tant que le code est faux et affiche « accès autorisé » lorsque le bon code est entré.

```
1  code="fab513"
2  rep=""
3  while .....
4      rep=input("Entrez le code.")
5  print.....
```

Boucles non bornées

EXERCICE 4 • *Compléter un programme*

SNT
Info embarquée

Compléter le programme suivant qui demande le code d'accès tant que le code est faux et affiche « accès autorisé » lorsque le bon code est entré.

```
1  code="fab513"
2  rep=""
3  while rep!=code:
4      rep=input("Entrez le code.")
5  print("accès autorisé")
```

Explications :

- On initialise `rep` à la chaîne vide `""` pour que la condition `rep!=code` soit vraie au départ.
- Tant que le code saisi (`rep`) est différent du bon code (`code`), on redemande le code.
- Quand le bon code est entré, la boucle s'arrête et le message « accès autorisé » s'affiche.

Boucles non bornées

EXERCICE 5 • *Comprendre et modifier un programme*

1. Combien de tours sont effectués dans le programme suivant ? Pourquoi ?

```
1  x=5
2  while x<3:
3      x=x-0.5
```

2. Que vaut la variable *x* à la fin de l'exécution du programme ?

3. Modifier ce programme de manière à ce que la boucle s'arrête quand *x* est inférieur à 3.

```
1
2
3
4
```

Boucles non bornées

Correction

Exercice 5 : Comprendre et modifier un programme

1. Combien de tours sont effectués dans le programme suivant ? Pourquoi ?

```
1  x=5
2  while x<3:
3      x=x-0.5
```

Au départ, x vaut 5.
La condition x<3 est fausse dès le début,
donc la boucle `while` ne s'exécute jamais.

Réponse : 0 tour.

2. Que vaut la variable x à la fin de l'exécution du programme ?

Comme la boucle ne s'exécute pas, la variable x ne change pas.

Réponse : x=5.

3. Modifier ce programme de manière à ce que la boucle s'arrête quand x est inférieur à 3.

```
1  x=5
2  while x>=3:
3      x=x-0.5
```

Cette fois, la boucle continue tant que
x est supérieur ou égal à 3.
Dès que x devient inférieur à 3, elle s'arrête.

Condition correcte à écrire : x>=3

Boucles non bornées

EXERCICE 6 • *Comprendre et corriger un programme*

Ayman a écrit le programme suivant pour afficher les cubes d'entiers inférieurs à 1 000 :

```
1  x=1
2  while x**3>1000:
3      print(x**3)
4      x=x+1
```

1. Qu'affiche ce programme ?

.....

2. Corriger le programme d'Ayman :

```
1  x=1
2  while .....:
3      print(x**3)
4      x=x+1
```

Boucles non bornées

Correction

Exercice 6 : Comprendre et corriger un programme

Rappel : le programme d'Ayman (faux)

```
1  x=1
2  while x**3>1000:
3      print(x**3)
4      x=x+1
```

1. Qu'affiche ce programme ?

Au départ, `x=1`, donc `x**3=1`. La condition `x**3>1000` est fausse.
La boucle `while` ne s'exécute donc jamais.

Réponse : ce programme n'affiche rien.

2. Corriger le programme d'Ayman

```
1  x=1
2  while x**3<1000:
3      print(x**3)
4      x=x+1
```

Cette fois, la boucle continue tant que le cube de `x` est inférieur à 1000.
Le programme affiche donc les cubes 1, 8, 27, ..., 729.

Condition correcte à écrire : `x**3<1000`

Boucles non bornées

EXERCICE 7 • *Comprendre un programme*

On considère la fonction `f` suivante :

```
1  def f(n):
2      while n>4:
3          n=n-5
4      return n
```

1. Que renvoient $f(0)$, $f(1)$, $f(5)$, $f(6)$?

.....

.....

2. Donner l'ensemble des valeurs de sortie possible pour f pour un paramètre n entier positif.

.....

Boucles non bornées

Correction

Exercice 7 : Comprendre un programme

Programme

```
1  def f(n):
2      while n>4:
3          n=n-5
4      return n
```

1. Que renvoient $f(0)$, $f(1)$, $f(5)$, $f(6)$?

La fonction enlève 5 tant que n est supérieur à 4.

$f(0) = 0$
$f(1) = 1$
$f(5) = 0$ (5 devient 0)
$f(6) = 1$ (6 devient 1)

2. Ensemble des valeurs de sortie possibles

La fonction soustrait 5 autant de fois que nécessaire.
Elle renvoie donc le reste de la division de n par 5.

Valeurs possibles : {0 ; 1 ; 2 ; 3 ; 4}

Par exemple : $f(5)=0$, $f(6)=1$, $f(7)=2$, $f(8)=3$, $f(9)=4$.

★ **Conclusion : $f(n)$ renvoie le reste de la division de n par 5.**

Boucles non bornées

EXERCICE 8

• *Comprendre et compléter un programme* BRANCHÉ

On considère le programme suivant :

```
1  a=1
2  b=1
3  while a>4 or b<3:
4      a=b+4
5      b=a-5
```

1. Lorsque $a = 1$ et $b = 1$, la condition de la ligne 3 est-elle vérifiée ?

2. Donner les valeurs de a et b à la sortie des 4 premiers tours de boucles.

Boucles non bornées

Correction

Exercice 8 : Comprendre et compléter un programme

```
1  a=1
2  b=1
3  while a>4 or b<3:
4      a=b+4
5      b=a-5
```

1. Lorsque $a=1$ et $b=1$, la condition de la ligne 3 est-elle vérifiée ?

On teste : $a>4$ ou $b<3$.

- $1>4$ est **faux**
- $1<3$ est **vrai**

Réponse : oui, la condition est vérifiée car **faux** ou vrai = vrai.

2. Valeurs de a et de b à la sortie des 4 premiers tours de boucle

Tour	a	b
1	5	0
2	4	−1
3	3	−2
4	2	−3

À chaque tour, on calcule d'abord $a=b+4$, puis $b=a-5$.

3. Quand la condition $a>4$ est-elle vérifiée ?

Elle est vérifiée seulement quand $a=5$, c'est-à-dire au début du 2e tour, après le 1er tour de boucle.

4. Quand la boucle `while` s'arrête-t-elle ?

La boucle s'arrête lorsque la condition $a>4$ or $b<3$ devient fausse. Il faudrait donc avoir à la fois $a \leq 4$ et $b \geq 3$. Or ici, à chaque tour, b diminue de 1 : 1, 0, −1, −2, ... Donc $b<3$ reste toujours vrai.

Conclusion : la boucle ne s'arrête jamais. C'est une boucle infinie.

5. Compléter le programme afin que la boucle finisse avec $a=0$ et $b=-5$

```
1  a=1
2  b=1
3  while a>0 and b>-5:
4      a=b+4
5      b=a-5
6  print(a,b)
```

Boucles non bornées

EXERCICE 9 • *Compléter un programme*

On considère le programme suivant :

```
1  for i in range (10):
2      print(i)
```

Compléter le programme ci-après afin qu'il soit équivalent au précédent mais en n'utilisant qu'une boucle **while**.

```
1  i=0
2  while .....
3      print(i)
4      i=.....
```

Boucles non bornées

Correction

Exercice 9 : Compléter un programme

```
1  for i in range(10):
2      print(i)
```

Ce programme affiche les entiers de 0 à 9.

```
1  i=0
2  while i<10:
3      print(i)
4      i=i+1
```

Explication :

- On commence avec `i=0`.
- Tant que `i<10`, on affiche `i`.
- Après chaque affichage, on augmente `i` de `1`.

Programme complété correctement.

Condition à écrire : `i<10` | Instruction à écrire ligne 4 : `i=i+1`

Boucles non bornées

EXERCICE 10 • *Comprendre un programme*

On considère la fonction `maximum` suivante :

```
1  def maximum(n):
2      s=1
3      while 7*s<=n:
4          s=7*s
5      return s
```

1. Donner maximum(10) et maximum(100).

2. Que fait cette fonction ?

3. Peut-on obtenir un équivalent de la fonction `maximum` en n'utilisant que des boucles `for` ?

Boucles non bornées

Correction

Exercice 10 : Comprendre un programme

```
def maximum(n):
    s=1
    while 7*s<=n:
        s=7*s
    return s
```

1. Donner maximum(10) et maximum(100).

- Pour n=10 : on part de s=1, puis 7*s=7 <= 10, donc s devient 7.
  Ensuite 7*s=49 > 10, donc on s'arrête.
- Pour n=100 : on part de s=1, puis s devient 7, puis s devient 49, et ensuite 7*s=343 > 100, donc on s'arrête.

`maximum(10) = 7`

`maximum(100) = 49`

2. Que fait cette fonction ?

Cette fonction renvoie la plus grande puissance de 7 inférieure ou égale à n.

`maximum(n) = la plus grande puissance de 7 <= n`

3. Peut-on obtenir un équivalent de la fonction maximum en n'utilisant que des boucles for ?

Oui.

On peut utiliser une boucle `for` exécutée un nombre suffisant de fois, et garder un test `if` à l'intérieur.

```
def maximum_for(n):
    s=1
    for i in range(n):
        if 7*s<=n:
            s=7*s
    return s
```

Cette version renvoie le même résultat que la fonction initiale.

Conclusion : la fonction maximum renvoie la plus grande puissance de 7 inférieure ou égale à n.

Boucles non bornées

EXERCICE 11 • *Comprendre et tester un programme*

On donne le programme suivant :

```
1  s=0
2  i=1
3  while s<13:
4      s=s+i
5      i=i+1
```

1. BRANCHÉ Que valent *i* et *s* à la fin de l'exécution de ce programme ?

.....

.....

2. BRANCHÉ On intervertit les lignes 4 et 5. Que valent *i* et *s* à la fin de l'exécution de ce programme ?

.....

Boucles non bornées

Correction

Exercice 11 : Comprendre et tester un programme

```
1  s=0
2  i=1
3  while s<13:
4      s=s+i
5      i=i+1
```

1. **Que valent *i* et s à la fin de l'exécution de ce programme ?**

Étape	s	*i*
Départ	s=0	i=1
Après le 1er tour	s=1	i=2
Après le 2e tour	s=3	i=3
Après le 3e tour	s=6	i=4
Après le 4e tour	s=10	i=5
Après le 5e tour	s=15	i=6

La boucle s'arrête car la condition s<13 devient fausse lorsque s=15.

Réponse : à la fin, i=6 et s=15.

2. **On intervertit les lignes 4 et 5. Que valent *i* et s à la fin de l'exécution de ce programme ?**

```
1  s=0
2  i=1
3  while s<13:
4      i=i+1
5      s=s+i
```

Étape	s	*i*
Départ	s=0	i=1
Après le 1er tour	s=2	i=2
Après le 2e tour	s=5	i=3
Après le 3e tour	s=9	i=4
Après le 4e tour	s=14	i=5

La boucle s'arrête car la condition s<13 devient fausse lorsque s=14.

Réponse : à la fin, i=5 et s=14.

Conclusion : l'ordre des instructions dans la boucle change le résultat final.

Boucles non bornées

EXERCICE 12 • *Compléter un programme* — SNT Réseaux sociaux

Le nombre d'abonnés à un groupe d'un réseau social évolue de la manière suivante : en 2018, il y avait 50 000 abonnés et on estime que le nombre d'abonnés est multiplié par 1,3 chaque année depuis.

1. Compléter le programme suivant qui affiche le nombre d'abonnés chaque année tant que le nombre d'abonnés est inférieur à 300 000.

```
1  abonnes=50000
2  while .....
3      abonnes=....*abonnes
4      print(abonnes)
```

Correction

Exercice 12 : Compléter un programme

En 2018, il y a 50 000 abonnés. Le nombre d'abonnés est multiplié par 1,3 chaque année.

```
1  abonnes=50000
2  while abonnes<30000:
3      abonnes=1.3*abonnes
4      print(abonnes)
```

Explications :

- On part de 50 000 abonnés.
- Tant que le nombre d'abonnés est inférieur à 300 000, on recommence la boucle.
- À chaque tour, on multiplie le nombre d'abonnés par 1,3 puis on l'affiche.

Programme complété correctement.

Condition à écrire : `abonnes<300000`

Instruction à écrire : `abonnes=1.3*abonnes`