

Plan du chapitre

I	Un exemple introductif	1
II	Les boucles « Tant que »	1
II.1	Syntaxe d'une boucle while	1
II.2	Exemples	2
II.3	Comment choisir entre une boucle for et une boucle while	3
III	Preuve de terminaison et de correction	4



I Un exemple introductif

Une boucle **while** permet de répéter l'exécution d'un bloc d'instructions tant qu'une certaine condition est vérifiée.

Exemple 1

Déterminer le plus petit entier n tel que $n! \geq 10^{10}$.

Un premier algorithme

$n \leftarrow 1$
tant que $n! < 10^{10}$ **faire**
 $n \leftarrow n + 1$

Résultat : n

Programme

```
1 n=1
2 while factorial(n) < 10**10:
3     n=n+1
4 print(n)
```

Et si on s'interdit l'usage de la fonction factorielle?

Algorithme

$n \leftarrow 1$
 $f \leftarrow 1$
tant que $f < 10^{10}$ **faire**
 $n \leftarrow n + 1$
 $f \leftarrow f \times n$

Résultat : n

Programme

```
1 n=1
2 f=1
3 while f < 10**10:
4     n=n+1
5     f=f*n
6 print(n)
```

II Les boucles « Tant que »

II.1 Syntaxe d'une boucle while

```
while condition :
    <instr 1>
    ...
    <instr n>
<ensuite>
```

- *condition* désigne un booléen.
- A l'exécution, *condition* est évalué, et si sa valeur est **True**, le bloc d'instructions est exécuté.
- Ce bloc est marqué par l'indentation.
- L'exécution du bloc est ensuite répétée jusqu'à ce que *condition* prenne la valeur **False**.

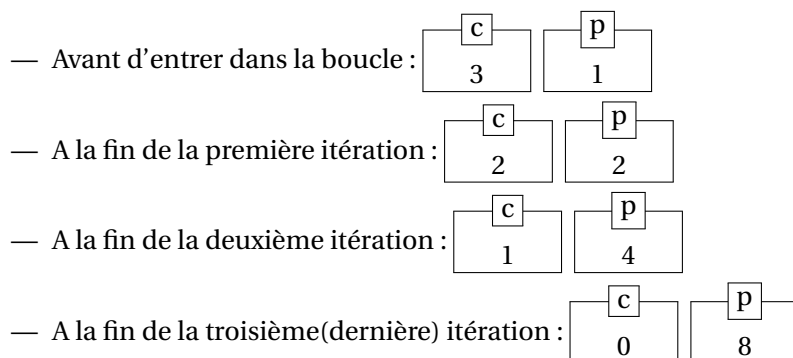
II.2 Exemples

Exemple 2

Que va afficher le programme suivant?

```
1 c,p=3,1
2 while c>0:
3     p=p*2
4     c=c-1
5 print("Maintenant, p vaut", p)
```

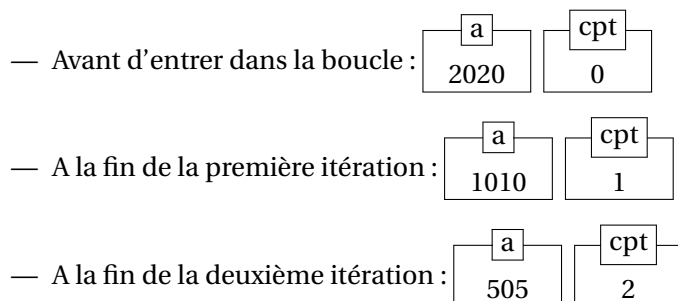
```
1 Maintenant, p vaut 8
```



Exemple 3

Que va afficher le programme suivant?

```
1 a, cpt=2020,0
2 while a%2==0:
3     cpt=cpt+1
4     a=a/2
5 print("Maintenant a vaut", a, "et cpt vaut", cpt)
```



```
1 Maintenant a vaut 505 et cpt vaut 2
```

Exemple 4

Que valent a et u à la sortie de la boucle suivante?

```

1 a=2016
2 u=0
3 while a>=5:
4     a=a-5
5     u=u+1

```

Réponse : à la sortie de la boucle, a et u contiennent respectivement le reste et le quotient de la division euclidienne de 2016 par 5, c'est-à-dire respectivement 1 et 403.

II.3 Comment choisir entre une boucle for et une boucle while

- Si on connaît à l'avance le nombre d'itérations à effectuer, ou plus généralement si on veut parcourir un objet itérable (range, liste, chaîne de caractères...), on choisit une boucle **for**.
- A l'inverse, si la décision d'arrêter la boucle ne peut s'exprimer que par un test, c'est la boucle **while** qu'il faut choisir.

Entrainement 1

Ecrire un programme qui demande à l'utilisateur un entier $n \geq 2$ et qui affiche le plus petit diviseur (différent de 1) de n .

```

1 n=int(input('Un entier supérieur ou égal 2'))
2 k=2
3 while n % k != 0:
4     k+=1
5 print('Plus petit diviseur (distinct de 1) de ', n, ': ', k)

```

Entrainement 2

Déterminer l'indice du dernier terme strictement positif de la suite définie par récurrence de la manière suivante :

$$u_0 = 2018 \quad \text{et} \quad \forall n \in \mathbb{N}, u_{n+1} = \frac{1}{2}u_n - 3n.$$

```

1 n,u=0,2018 #initialisation des variables n et u
2 while u>0:
3     u=u/2-3*n #Les deux lignes du bloc d'instructions

```

```
4     n=n+1     #ne doivent pas être interchangées.
5
6 #En sortie de boucle, n contient l'indice
7 #du premier terme négatif ou nul de la suite.
8
9 print(n-1)
```

Entrainement 3

Ecrire un programme qui demande à l'utilisateur une chaîne de caractères et qui affiche la position de la première voyelle (s'il y en a au moins une).

```
1 texte=input('Un_texte_svp:')
2 i=0
3 while i<len(texte) and (texte[i] not in 'aeiouy'):
4     i=i+1
5 if i==len(texte):
6     print('Pas_de_voyelle')
7 else:
8     print('Indice_de_la_première_voyelle:',i)
```

Attention à ne pas interchanger les deux booléens `i<len(texte)` et `(texte[i] not in 'aeiouy')` (caractère paresseux du `and`).

III Preuve de terminaison et de correction

Que fait le programme suivant? Comment être sûr que la boucle va s'arrêter?

```
1 c=int(input("Entrez_un_entier_positif_n:"))
2 p=1
3 while c>0:
4     p=p*2
5     c=c-1
6 print(p)
```

Ce programme calcule la valeur de 2^n .

La condition pour que la boucle tourne est que la valeur de `c` soit strictement positive.

Or, la valeur de `c` est un entier, qui décroît d'une unité à chaque itération : ainsi, si la boucle démarre, elle finira forcément par s'arrêter.

Dans la pratique, il est souvent simple de prouver qu'une boucle **while** se termine : la condition de la boucle est en général une inégalité, dont l'une des deux expressions varie à chaque itération.

Il suffit alors simplement de vérifier que cette variable évolue « dans le bon sens », c'est à dire de façon à ce que l'inégalité ne soit pas éternellement vérifiée.

Pour démontrer qu'une boucle fournit le résultat attendu, on cherche *un invariant de boucle*, c'est-à-dire une propriété portant sur le contenu des variables et qui :

- ▶ est vérifiée avant d'entrer dans la boucle,
- ▶ si elle est vérifiée avant une itération de la boucle, est vérifiée après celle-ci,
- ▶ lorsqu'elle est vérifiée en sortie de boucle permet d'en déduire que le programme est correct.

Exemple 5

Montrons par exemple que le programme suivant affiche la valeur de 2^n .

```
1 c=int(input("Entrez_un_entier_positif_n:_"))
2 p=1
3 while c>0:
4     p=p*2
5     c=c-1
6 print(p)
```

La propriété « $p = 2^{n-c}$ » est un invariant de boucle. En effet,

- ▷ cette propriété est vraie avant d'entrer dans la boucle, car alors $p = 1$ et $c = n$.
- ▷ Si cette propriété est vraie à la fin d'une itération, elle est nécessairement vraie à la fin de la suivante, car le corps de la boucle retire 1 à c et multiplie p par 2.

Par conséquent, la propriété « $p = 2^{n-c}$ » est aussi vraie en sortie de boucle, lorsque $c = 0$.

En sortie de boucle, p contient donc bien 2^n .

Exemple 6

On suppose que a et b sont deux entiers naturels, avec $b \geq 1$. Montrer que la boucle suivante termine, et qu'à la fin de cette boucle, q et r contiennent respectivement le quotient et le reste de la division euclidienne de a par b .

```
1 q=0
2 r=a
3 while r>=b:
4     q=q+1
5     r=r-b
```

La terminaison est presque évidente : comme la variable r est entière et diminue à chaque tour de boucle, la condition $r \geq b$ ne peut être vérifiée indéfiniment : la boucle va nécessairement se terminer.

L'invariant de boucle est plus difficile à deviner, mais on peut le vérifier facilement : $a = q \times b + r$.

- ▷ Initialement, cette propriété est vraie, puisque $q = 0$ et $r = a$.
- ▷ Si cette propriété est vraie à la fin d'une itération, elle est vraie à la fin de la suivante, puisque le corps de la boucle retire b à r mais ajoute 1 à q .

A la fin de la boucle, nous avons donc $a = q \times b + r$ et $0 \leq r < b$, ce qui caractérise la division euclidienne de a par b .