Plan du chapitre

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 $n! \geqslant 10^{10}$.

Un premier algorithme

$n \longleftarrow 1$

tant que $n! < 10^{10}$ faire

$\quad n \longleftarrow n+1$

Résultat : n

Programme

```python
n=1
while factorial(n)<10**10:
    n=n+1
print(n)
```

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

Algorithme

$n \longleftarrow 1$

$f \longleftarrow 1$

tant que $f < 10^{10}$ faire

$\quad n \longleftarrow n+1$

$\quad f \longleftarrow f \times n$

Résultat : n

Programme

```python
n=1
f=1
while f<10**10:
    n=n+1
    f=f*n
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 ?

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

```
Maintenant, p vaut 8
```

- Avant d'entrer dans la boucle : $\overset{\text{c}}{\boxed{3}}$ $\overset{\text{p}}{\boxed{1}}$
- A la fin de la première itération : $\overset{\text{c}}{\boxed{2}}$ $\overset{\text{p}}{\boxed{2}}$
- A la fin de la deuxième itération : $\overset{\text{c}}{\boxed{1}}$ $\overset{\text{p}}{\boxed{4}}$
- A la fin de la troisième(dernière) itération : $\overset{\text{c}}{\boxed{0}}$ $\overset{\text{p}}{\boxed{8}}$

Exemple 3

Que va afficher le programme suivant ?

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

- Avant d'entrer dans la boucle : $\overset{\text{a}}{\boxed{2020}}$ $\overset{\text{cpt}}{\boxed{0}}$
- A la fin de la première itération : $\overset{\text{a}}{\boxed{1010}}$ $\overset{\text{cpt}}{\boxed{1}}$
- A la fin de la deuxième itération : $\overset{\text{a}}{\boxed{505}}$ $\overset{\text{cpt}}{\boxed{2}}$

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

Exemple 4

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

```
a=2016
u=0
while a>=5:
    a=a-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, chaine 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 $n \geqslant 2$ et qui affiche le plus petit diviseur (différent de 1) de n.

```
n=int(input('Un entier supérieur ou égal 2'))
k=2
while n % k!=0:
    k+=1
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 :

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

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

```
    n=n+1    #ne doivent pas être interchangées.

#En sortie de boucle, n contient l'indice
#du premier terme négatif ou nul de la suite.

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).

```
texte=input('Un texte svp:')
i=0
while i<len(texte) and (texte[i] not in 'aeiouy'):
    i=i+1
if i==len(texte):
    print('Pas de voyelle')
else:
    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?

```
c=int(input("Entrez un entier positif n : "))
p=1
while c>0:
   p=p*2
   c=c-1
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 décroit 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.

```
c=int(input("Entrez un entier positif n : "))
p=1
while c>0:
    p=p*2
    c=c-1
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 $b \geqslant 1$. 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.

```
q=0
r=a
while r>=b:
    q=q+1
    r=r-b
```

La terminaison est presque évidente : comme la variable **r** est entière et diminue à chaque tour de boucle, la condition **r>=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 $0 \leqslant r < b$, ce qui caractérise la division euclidienne de a par b.