Introduction

En algorithmique, il existe deux structures de bases très fréquemment utilisés et qu'il est nécessaire de connaître et maîtriser.

Les tests if qui permettent de **distinguer et traiter plusieurs cas de figure**. En français, se teste se traduit par

```
Si condition est vrai
Alors
executer instructions 1
Sinon
executer instructions 2
```

Exemple 1 :

```
Si motDePasse est juste
Alors
entrez
Sinon
rester dehors
```

Les boucles qui permettent d'exécuter les mêmes instructions un grand nombre de fois.

Il existe deux types de boucles : les boucles inconditionnelles appelés **boucle FOR** et les boucles conditionnelles **boucle WHILE**.

Le structure générale de ces boucles sont :

Boucle FOR (pour)	Boucle While (tant que)
Pour i allant de p à n	tant que condition est vrai
faire instructions	faire instructions
fin du pour	fin du tant que

Exemple 2 : le même programme écrit avec ces deux boucles

pour i allant de 1 à 50	i=1
ecrire i	tant que i<=50
fin du pour	ecrire i
	i=i+1
	fin du tant que

Que font ces algorithmes ? il affiche tous les entiers de 1 à 50.

La différence ? en général on utilise la boucle **for** lorsque l'on sait à l'avance combien d'instructions vont être exécutées.

La boucle **while** est plus générale. Elle peut-être utilisée dans tous les cas, surtout si on ne sait pas à l'avance combien d'instructions vont être nécessaire avant de réaliser une condition recherchée.

✘✘✘ Attention ! deux problèmes se posent en général

- la **correction :** est-ce que la boucle réalise ce qu'on lui demande ?
- la **terminaison** surtout avec une **boucle while** : est-ce que la boucle se termine ou bien tourne-t-elle indéfiniment ?

  Par exemple, dans la boucle tant que de l'exemple 2, si on omet l'instruction **i=i+1** alors la boucle ne va jamais s'arrêter (pourquoi ?) et va épuiser votre batterie.

Le test if en Python

Ecrivons le programme de l'exemple 1 permettant de valider un mot de passe. Supposons que le mot de passe soit **Einstein**.

En python, le test if s'écrit

```
if mp==Einstein:
    entrer=True
else:
    entrer=False
```

Le retour à la ligne à la fin signifie la fin du test Si.

On peut également faire un test if avec « deux if ».

Par exemple si le progamme de la fonction $f(x) = \begin{cases} -1 \text{ si } x < 0 \\ 0 \text{ si } 0 \leqslant x < 1 \\ 1 \text{ si } x \geqslant 1 \end{cases}$ s'écrira

```
if x<0:
    y=-1
elseif x<1:
    y=0
else:
    y=1
```

La variable y représente ici $f(x)$.

La boucle For en Python

Le programme correspondant à l'**exemple 2** va s'écrire en Python de la manière suivante :

```
for i in range(1,51):
    print(i)
```

La fin de la boucle est le retour à la verticale du **for**. Noter l'espace avant l'instruction print(i) qui s'appelle **indentation** et qui doit être respectée pour toutes les instructions à l'**intérieur** d'une boucle for ou while.

Exemple important : programmer une suite récurrente.

On se propose de calculer les 2000 premiers termes de la suite $(u_n)_{n\in\mathbb{N}}$ définie par $u_0 = 2$ et $u_{n+1} = \sqrt{u_n} + 1$.

Une possibilité est :

```
import numpy as np
u=2
for i in range(1,2001):
        u=np.sqrt(u)+1
print(u)
```

☞ **S'exercer** **Quelle différence d'exécution donne le programme suivant ?**

```
import numpy as np
u=2
for i in range(1,2001):
        u=np.sqrt(u)+1
        print(u)
```

La boucle while

Un exemple typique : on sait que la suite des inverses d'entiers définie par $u_n = \dfrac{1}{n}$ pour $n \in \mathbb{N}^*$ et positive, strictement décroissante et tend vers 0 lorsque n tend vers $+\infty$: $\lim\limits_{n\to+\infty} \dfrac{1}{n} = 0$.

On sait alors qu'il existe un entier n_0 à partir duquel u_n sera inférieur à un petit nombre disons $\epsilon = 0,0003$.

La **boucle while** va nous permettre de déterminer ce nombre. Notez qu'ici on ne sait pas à l'avance combien de calculs seront nécessaires. Cet algorithme est un algorithme de recherche de seuil.

Voici une proposition de programme :

```
n=1
epsilon=0.0003
while 1/n>=epsilon:
    n=n+1
print(n)
```

A comparer à

```
n=1
epsilon=0.0003
while 1/n>=epsilon:
    n=n+1
    print(n)
```

Quelques exercices d'applications

Exercice 1

1. Programmer la suite u définie par $u_1 = 2$ et $u_{n+1} = 3u_n + 2$.
2. Montrer que cette suite est à termes positives et strictement croissante.
3. Déterminer à l'aide d'un programme le plus petit entier n tel que $u_n > 1000$.

Exercice 2

Pour $n \geqslant 1$, on pose $h_n = 1 + \dfrac{1}{2} + \dfrac{1}{3} + \ldots + \dfrac{1}{n} = \sum\limits_{i=1}^{n} \dfrac{1}{i}$.

1. Montrer que la suite $(h_n)_{n\geqslant 1}$ est à termes positives et strictement croissante.
   On admet que cette suite n'est pas majorée.
2. Déterminer à l'aide d'un programme h_{50}.
3. Déterminer à l'aide d'un programme le plus petit entier n tel que $h_n > 1000$.