

# TP 4

## Structures conditionnelles et itératives (2/2)

🐞 **Explication** 🐞 Ce TP 4 - Structures conditionnelles et itératives (2/2) est une application directe des notions découvertes dans les

*Cours 3 - La boucle for, Cours 4 - Structures conditionnelles et Cours 5 - La boucle while.*

Voici les objectifs du TP 4 :

- 1) Consolider l'ensemble des notions déjà vues jusque-là,
- 2) Manipuler les boucles while et for, et savoir coder un compteur.

### I - Boucle for versus boucle while

**Les lignes de code seront toujours écrites dans l'éditeur, les tests dans la console.**

🐞 **Explication** 🐞 Cette première partie permet de s'assurer que les bases du cours sur les boucles for et while sont bien assimilées.

On rappelle que toute boucle for peut être réécrite sous la forme d'une boucle while, mais que l'inverse n'est pas toujours vraie.

# 1

**Exercice**

🕒 Pour chacun des exemples suivants,

- 1) Lire et intuiter le fonctionnement des lignes de code, puis tester-les avec l'éditeur.
- 2) Réécrire les boucles for en boucles while, et vice-versa (si c'est possible).

Lignes de code	Résultats en console	Réécriture en boucle while ou for
<pre> 1 s=0 2 for i in range(5): 3     if i%2==0: 4         s=s+1 5 print(s) </pre>		
<pre> 1 s=0 2 for i in range(5): 3     if i%2==0: 4         s=s+1 5 print(s) </pre>		

<pre> 1 i=0 2 while i&lt;5: 3     i=i+1 4     print(i) </pre>		
<pre> 1 i=0 2 while i&lt;5: 3     i=i+1 4     print(i) </pre>		

**2****Exercice**

- 1) a) À l'aide d'une boucle for, écrire une fonction

zero\_list1(L),

qui prend en paramètre une liste L de nombres, et **renvoie** son premier terme non nul ou 0 si tous les termes sont nuls.

- b) À l'aide d'une boucle while, écrire une fonction

zero\_list2(L),

qui prend en paramètre une liste L de nombres, et **renvoie** son premier terme

non nul ou 0 si tous les termes sont nuls.

- 2) a) À l'aide d'une boucle for, écrire une fonction

signe1(L),

qui prend en paramètre une liste L de nombres, et **teste** si tous les nombres sont de même signe ou pas.

- b) À l'aide d'une boucle while, écrire une fonction

signe2(L),

qui prend en paramètre une liste L de nombres, et **teste** si tous les nombres sont de même signe ou pas.

## II - Résolution d'énigmes avec Python

🔗 **Explication** 🔗 Les exercices suivants ont été conçus à partir d'énigmes du site *Project Euler*. Il s'agit d'énigmes mathématiques pouvant être résolues efficacement avec un ordinateur.

**3****Exercice**

- 1) a) À l'aide d'une boucle for, écrire une fonction

facto1(n),

qui prend en paramètre un entier n, et **renvoie** sa factorielle.

```

1 # voici un exemple d'appel :
2 >>> facto(5)
3 120

```

- b) À l'aide d'une boucle while, écrire une fonction

facto2(n),

qui prend en paramètre un entier n, et **renvoie** sa factorielle.

- 2) Écrire une fonction

somme\_entier(n),

qui prend en paramètre un entier n, et **renvoie** la somme de ses chiffres.

```

1 # voici un exemple d'appel :
2 >>> somme_entier(174)
3 12

```

- 3) **Énigme 1 :**  
Quelle est la somme des chiffres de 100! ?

## 4

## Exercice



**(Suite de Fibonacci)** On considère la suite de Fibonacci i.e. la suite définie par :

$$\begin{cases} F_0 = 1 \text{ et } F_1 = 1 \\ \forall n \in \mathbb{N}, F_{n+2} = F_{n+1} + F_n \end{cases}$$

- 1) a) À l'aide d'une boucle `for`, écrire une fonction

`fibonacci1(n),`

qui prend en paramètre un entier `n`, et **renvoie** le terme d'indice `n` de la suite.

- b) Tester la fonction.

- 2) a) À l'aide d'une boucle `while`, écrire une fonction

`fibonacci2(n),`

qui prend en paramètre un entier `n`, et **renvoie** le terme d'indice `n` de la suite.

- b) Tester la fonction.

### 3) Énigme 2 :

Quelle est la somme de tous les termes de la suite de Fibonacci qui sont pairs et inférieurs ou égaux à quatre millions ?

*Demander au professeur si le résultat est correct !*

### 4) Énigme 3 :

Voici les premiers termes de la suite :

$$F_0 = 1, F_1 = 1, F_2 = 2, F_3 = 3, \\ F_4 = 5, F_5 = 8, F_6 = 13, \dots$$

Le terme d'indice 6 est donc le premier terme de la suite qui contient deux chiffres. Quel est l'indice du premier terme de la suite qui contient 1000 chiffres ?

*Demander au professeur si le résultat est correct !*

## III - Création de petits jeux avec Python

🐍 **Explication** 🐍 Cette partie récréative permet de montrer qu'avec les connaissances actuelles, il est déjà possible de coder des petits jeux sympatiques.

Il sera nécessaire d'importer le module `random` avec la ligne de code suivante :

```
1 # à saisir dans l'éditeur
2 from random import *
```

```
1 # randint(a,b) permet de générer aléatoirement un entier dans [[a,b]]
2 >>> randint(1,10) # génère aléatoirement un entier entre 1 et 10 compris
3 5
```

## 5

## Exercice



**(Le jeu du juste prix)**

- 1) Écrire une fonction

`juste_prix_1(),`

où l'ordinateur choisit aléatoirement un entier entre 1 et 100 (compris) et demande à l'utilisateur de deviner. À chaque entrée de l'utilisateur, le programme **affiche** si c'est plus ou si c'est moins jusqu'à ce que l'utilisateur gagne.

```
1 # Voici un exemple d'appel :
2 >>> juste_prix_1()
3 # Voilà ce qui est affiché
4 Je choisis un nombre entre 1 et 100.
5 Quel nombre ai-je choisi ? 15
6 # l'utilisateur rentre 15 au clavier
7 Plus !
```

```
8 Quel nombre ai-je choisi ? 75
9 # l'utilisateur rentre 75 au clavier
10 Moins !
11 Quel nombre ai-je choisi ? 35
12 # l'utilisateur rentre 35 au clavier
13 Gagné !
```

- 2) Écrire une fonction

`juste_prix_2(n),`

qui correspond au jeu du juste prix avec `n` essais. À chaque essai raté, l'ordinateur **affiche** le décompte du nombre d'essais restants.

Si l'utilisateur n'a pas deviné au bout des `n` essais, l'ordinateur **affiche** que c'est perdu !

```
1 # Voici un exemple d'appel :
2 >>> juste_prix_2(3)
3 # Voilà ce qui est affiché
```


```

4 Je choisis un nombre entre 1 et 100.
5 Vous avez 3 essai(s) !
6 Quel nombre ai-je choisi ? 15
7 # l'utilisateur rentre 15 au clavier
8 Plus ! Il reste 2 essai(s) !
9 Quel nombre ai-je choisi ? 50
10 # l'utilisateur rentre 50 au clavier
11 Plus ! Il reste 1 essai(s) !
12 Quel nombre ai-je choisi ? 75
13 # l'utilisateur rentre 75 au clavier
14 Moins ! Il reste 0 essai(s). Perdu !

```

## 6

### Exercice

 (Le jeu du Shifumi) Le Shifumi, plus communément connu sous le nom de

*Pierre-Feuille-Ciseaux,*

est un jeu où deux joueurs s'affrontent en choisissant chacun pierre, feuille ou (exclusif) ciseaux. Le gagnant est ensuite déterminé par les règles suivantes :

- la pierre émousse les ciseaux,
- les ciseaux coupent la feuille,
- la feuille enveloppe la pierre.

Le but ici est de coder le jeu du Shifumi, dans le cas où l'un des joueurs est l'utilisateur et l'autre l'ordinateur.

#### 1) Écrire une fonction

`choix_pc(),`

qui **renvoie** le choix aléatoire de l'ordinateur sous la forme d'une chaîne de caractères :

`"pierre", "feuille" ou "ciseaux"`

```

1 # voici un exemple d'appel :
2 >>> choix_pc()
3 "pierre" # une chance sur 3

```

#### 2) Écrire une fonction

`choix_joueur(),`

qui demande à l'utilisateur son choix et le **renvoie** sous la forme d'une chaîne de caractères.

```

1 # voici un exemple d'appel :
2 >>> choix_joueur()
3 pierre(p)/feuille(f)/ciseaux(c)? c
4 # le joueur rentre c au clavier
5 "ciseaux"

```

#### 3) Écrire une fonction

`une_manche(),`

utilisant les fonctions précédemment créées, qui **affiche** le choix du joueur, de l'ordinateur et le résultat de la partie.

```

1 # voici un exemple d'appel de la
  fonction :
2 >>> une_manche()
3 pierre(p)/feuille(f)/ciseaux(c)? c
4 Vous avez joué ciseaux
5 Le PC a joué pierre
6 Vous avez perdu !

```

#### 4) En copiant et modifiant la fonction `une_manche()` précédemment créée, écrire une nouvelle fonction

`une_manche2(),`

qui fait jouer une manche à l'utilisateur et l'ordinateur, et **renvoie** en plus

- **"J"** si l'utilisateur gagne,
- **"E"** si c'est une égalité,
- **"O"** si l'ordinateur gagne,

#### 5) Écrire une fonction

`shifumi(n),`

qui prend en paramètre un entier `n`, et fait jouer une partie de `n` manches entre l'utilisateur et l'ordinateur, pour enfin **afficher** le gagnant, son nombre de victoires et un résumé de la partie.

```

1 # voici un exemple d'appel pour n=3 :
2 >>> shifumi(3)
3 Manche 1
4 pierre(p)/feuille(f)/ciseaux(c)? c
5 Vous avez joué ciseaux
6 Le PC a joué feuille
7 Manche 2
8 pierre(p)/feuille(f)/ciseaux(c)? p
9 Vous avez joué pierre
10 Le PC a joué pierre
11 Manche 3
12 pierre(p)/feuille(f)/ciseaux(c)? f
13 Vous avez joué feuille
14 Le PC a joué pierre
15 Bilan : JEJ
16 2 victoire(s), 1 égalité(s), 0
   défaite(s)
17 Vous avez gagné !

```