

TP 3

Structures conditionnelles et itératives (1/2)

📌 **Explication** 📌 Ce TP 3 - Structures conditionnelles et itératives (1/2) est une application directe des notions découvertes dans le Cours 3 - La boucle For. Il permet aussi de s'initier à la notion de structures conditionnelles que l'on découvrira dans l'exercice 1 de ce présent TP.

Voici les objectifs du TP 3 :

- 1) Découvrir et se familiariser avec les structures conditionnelles,
- 2) Se perfectionner sur l'écriture de fonctions,
- 3) Manipuler la boucle For et savoir mettre en place un compteur.

I - Structures conditionnelles

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

📌 **Explication** 📌 Nous allons découvrir et apprendre à travers l'exercice 1 ci-bas ce que sont les structures conditionnelles.

1

Exercice

🕒 Lire et intuiter le fonctionnement des lignes de code suivantes, puis tester-les avec l'éditeur :
Rappelons que les lignes en verte sont des explications et que vous n'avez donc pas à les saisir !


Lignes de code à saisir dans l'éditeur	Résultats obtenus dans la console après compilation
<pre> 1 a=6 2 b=4 3 # découverte du mot-clé if 4 if a>b : 5 print(a, "est supérieur strict à", b)</pre>	
<pre> 1 a=-2 2 b=4 3 # découverte du mot-clé if 4 if a>b : 5 print(a, "est supérieur strict à", b)</pre>	

<pre> 1 a=-2 2 b=4 3 if a>b : 4 print(a,"est supérieur strict à",b) 5 # découverte du mot-clé elif 6 elif a<b : 7 print(a,"est inférieur strict à",b) </pre>	
<pre> 1 a=4 2 b=4 3 if a>b : 4 print(a,"est supérieur strict à",b) 5 # découverte du mot-clé elif 6 elif a<b : 7 print(a,"est inférieur strict à",b) </pre>	
<pre> 1 a=7 2 b=4 3 # découverte du mot-clé and 4 if a>b and a>0 : 5 print(a,"est positif et supérieur strict à",b) </pre>	
<pre> 1 a=7 2 # découverte du mot-clé or 3 if a>0 or a<0 : 4 print(a,"est différent de 0") </pre>	
<pre> 1 a=7 2 # découverte du mot-clé not 3 if not(a==0) : 4 print(a,"est différent de 0") </pre>	
<pre> 1 a=9 #a,b entiers ici 2 b=4 3 # découverte du mot clé or 4 if a==b+1 or a==b-1 : 5 print(" ",a,"-",b," =",1) 6 # on peut mettre plusieurs elif 7 elif a==b+2 or a==b-2 : 8 print(" ",a,"-",b," =",2) 9 elif a==b+3 or a==b-3 : 10 print(" ",a,"-",b," =",3) 11 # découverte du mot-clé else 12 else : # tous les autres cas 13 print("l'écart entre",a,"et",b,"est supérieur à 4") </pre>	

✖ **ATTENTION!** ✖ Il va falloir systématiquement penser à :

- terminer chaque ligne commençant par un mot-clé (if, elif, else) par deux points « : », sans quoi vous aurez un message d'erreur `SyntaxError`,
- indenter après chaque ligne commençant par un mot clé (if, elif, else), sans quoi vous aurez un message d'erreur `IndentationError`.

2**Exercice**


 Écrire une fonction

`vabs(x),`

qui prend en paramètre un réel `x`, et **renvoie** la valeur absolue de ce réel.

On n'utilisera pas les fonctions natives `abs()` ou `sqrt()` déjà définies sur Python.


3**Exercice**

 Écrire une fonction

`intervalle(a,b,c)`

qui **teste** si un nombre donné `c` est dans l'intervalle `[a,b]`, autrement dit renvoie le booléen `True` ou `False` suivant les cas.

4**Exercice**



 Écrire une fonction

`max2(a,b),`

qui prend en paramètres deux réels `a` et `b`, et **renvoie** le maximum des deux réels.

On n'utilisera pas la fonction native `max()` déjà définie sur Python.

5**Exercice**




  Écrire une fonction

`max3(a,b,c),`

qui prend en paramètres trois réels `a`, `b` et `c`, et **renvoie** le maximum des trois réels.

On n'utilisera pas la fonction native `max()` déjà définie sur Python.

6**Exercice**

   Une année est dite bissextile si elle est un multiple de 4, sauf si elle est un multiple de 100. Toutefois, elle est considérée comme bissextile si elle est un multiple de 400.

Écrire une fonction

`bissextile(a),`

qui prend en paramètre un entier `a`, et **renvoie** `True` si l'entier `a` correspond au numéro d'une année bissextile et **renvoie** `False` sinon.

II - Structures itératives

7**Exercice**

  Lire et intuiter le fonctionnement des lignes de code suivantes :

Lignes de code à saisir dans l'éditeur	Résultats obtenus dans la console après compilation
<pre> 1 i=0 2 for k in range(0,4) : 3 if k%2==0 : 4 i=i+1 5 print(i) </pre>	

<pre> 1 i=0 2 for k in range(0,4) : 3 if k%2==0 : 4 i=i+1 5 print(i) </pre>	
<pre> 1 i=0 2 for k in range(0,4) : 3 if k%2==0 : 4 i=i+1 5 else : 6 i=i+2 7 print(i) </pre>	

8**Exercice****1) Écrire une fonction**

`somme_list(L)`

qui prend en paramètre une liste L de réels et **renvoie** la somme de ses éléments.

On n'utilisera pas la fonction native `sum()` déjà définie sur Python.

2) Écrire une fonction

`moyenne_list(L)`

qui prend en paramètre une liste L de réels et **renvoie** la moyenne de ses éléments.

On utilisera obligatoirement la fonction `somme_list(L)` précédemment créée.

9**Exercice****Écrire une fonction**

`fusion_list(L,M)`

qui prend en paramètres deux listes L et M et **renvoie** une nouvelle liste contenant les éléments de L suivis des éléments de M.

On n'utilisera pas l'opérateur de concaténation de listes `+`.

10**Exercice****1) Écrire une fonction**

`max_list(L)`

qui prend en paramètre une liste L de réels et **renvoie** son plus grand élément.

2) Écrire une fonction

`ind_max_list(L)`

qui prend en paramètre une liste L de réels et **renvoie** l'indice de son plus grand élément.

Si le maximum apparaît à plusieurs endroits dans la liste L, alors la fonction doit renvoyer la liste des indices correspondants.

3) Écrire une fonction

`maxi_list(L)`


qui prend en paramètre une liste L de réels et **renvoie** une liste composée de son plus grand élément, et de la liste des indices où se situe ce maximum.

```

1 # la fonction créée doit passer les
   tests suivants :
2 >>> maxi_list([2,5,1,7,9,2,4])
3 [9,[4]]
4
5 >>> maxi_list([2,10,1,7,10,2,4])
6 [10,[1,4]]

```

11**Exercice**

 Écrire une fonction



```
renverse_list(L)
```

qui prend en paramètre une liste L et **renvoie** une liste constituée des mêmes éléments mais dans l'ordre inverse.




On n'utilisera pas la méthode `.reverse()`.

```
1 # la fonction créée doit passer le test
   # suivant :
2 >>> renverse_list([2,4,7,-1])
3 [-1,7,4,2]
```

III - Résolution d'énigmes

 **Explication**  Pour chacune des énigmes suivantes, il s'agit d'écrire une ou des fonctions qui permettent ensuite de les résoudre !

12**Exercice**




   En listant les entiers naturels inférieurs ou égaux à 15 qui sont multiples de 3 ou (inclusif) 5, on obtient :

0, 3, 5, 6, 9, 12 et 15

La somme de ces multiples fait alors 50.

Déterminer la somme des entiers inférieurs ou égaux à 1000 qui sont multiples de 3 ou 5.

13**Exercice**

   $2^{15} = 32768$ et la somme de ses chiffres vaut $3 + 2 + 7 + 6 + 8 = 26$.

Quelle est la somme des chiffres du nombre 2^{1000} ?