TP 2

Listes et fonctions

✌ **Explication** ✌ Ce *TP 2 - Listes et fonctions* est une application directe des notions découvertes dans le *Cours 2 - Listes et fonctions*.

> Voici les deux objectifs du TP 2 :
>
> **1)** Se familiariser avec la notion de listes et découvrir les commandes classiques relatives aux listes,
>
> **2)** Se familiariser avec la notion de fonctions, et être capable d'écrire des fonctions simples et de manière esthétique (si possible).

Ce *TP 2 - Listes et fonctions* fait partie intégrante du cours. Il faudra le finir chez soi (si ce n'est pas fait lors de la séance), puis apprendre les quelques commandes classiques sur lesquelles vous serez évalués en début de séance du TP 3.

I - Les listes

Dans cette partie I, nous travaillerons uniquement dans la console !

✌ **Explication** ✌ **(Rappel de cours)** Une liste est définie en donnant ses éléments, séparés par des virgules, le tout encadré par des crochets.

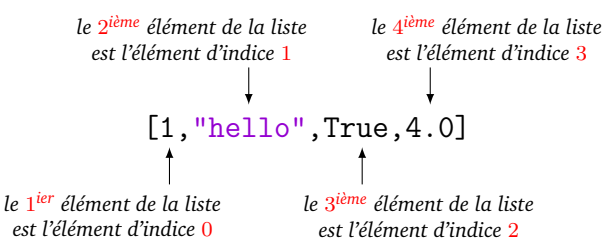
Exemple

```
# on effectue l'affectation suivante :
>>> L=[1,"hello",True,4.0]

# rappelons qu'une affectation est une instruction et que rien n'apparaît à l'écran !

# il faut taper la commande suivante pour accéder au contenu de la variable L
>>> L
[1,"hello",True,4.0]
```

La variable `L` contient maintenant la liste suivante constituée de 4 éléments (qui ne sont pas du même type sur cet exemple !) :



✖ **ATTENTION !** ✖ Ne pas confondre la position d'un élément et son indice !
Dans l'exemple précédent, la chaîne de caractère `"hello"` est le $2^{ième}$ élément de la liste, mais correspond à l'élément d'indice 1 de la liste.

☞ *Cliquez ici pour accéder au corrigé*

✌ **Explication** ✌ Soit i un entier naturel, `y` une donnée quelconque, `L` et `M` deux listes. Voici les commandes à lire, comprendre puis apprendre par coeur :

Commandes	*À quoi sert la commande ?*
`>>> L[i]`	accéder à l'élément d'indice `i` de la liste
`>>> L[-i]`	accéder au $i^{ième}$ élément de la liste en partant de la fin
`>>> len(L)`	accéder à la longueur de la liste
`>>> L[i:j]`	extraire la sous-liste constituée des éléments d'indices `i` jusqu'à `j-1`
`>>> L[:j]`	idem que `L[0:j]`
`>>> L[i:]`	idem que `L[i:len(L)-1]`
`>>> L+M`	concaténer les listes `L` et `M`
`>>> 3*L`	idem que `L+L+L`
`>>> L[i]=y`	remplacer l'élément d'indice `i` par la donnée `y`
`>>> L.append(y)`	ajouter la donnée `y` en fin de liste
`>>> L.sort()`	trier la liste `L` par ordre croissant (si c'est possible !)

✖ **ATTENTION !** ✖ Signalons que les méthodes `.append()` et `.sort()` modifient la liste `L` initiale et ne sont pas précédées d'une affectation (**Exemple** On ne fait jamais `L=L.append(15)`).

1 *Exercice*

🕑 Compléter, puis vérifier avec la console :

```
>>> L=[1,[-2,5,4],"hello",2,-9.0,False]

..... # qu'obtenez-vous ?

>>> L

..... # qu'obtenez-vous ?

>>> type(L)

..... # qu'obtenez-vous ?

>>> len(L)

..... # qu'obtenez-vous ?

>>> type(len(L))

..... # qu'obtenez-vous ?

>>> L[0]

..... # qu'obtenez-vous ?

>>> L[3]

..... # qu'obtenez-vous ?

>>> type(L[len(L)-1])

..... # qu'obtenez-vous ?

>>> L[-1]==L[len(L)-1]

..... # qu'obtenez-vous ?

>>> L[1][len(L[1])-1]

..... # qu'obtenez-vous ?

>>> L[-2]

..... # qu'obtenez-vous ?

>>> L[-4]

..... # qu'obtenez-vous ?

>>> L[2:4]

..... # qu'obtenez-vous ?

>>> L[:3]

..... # qu'obtenez-vous ?

>>> L[4:]

..... # qu'obtenez-vous ?

>>> L[0]="bonjour"

..... # qu'obtenez-vous ?

>>> L

..... # qu'obtenez-vous ?
```

☞ *Cliquez ici pour accéder au corrigé*

2 *Exercice*

Compléter, puis vérifier avec la console :

```
>>> L=[-5,4,2,17,0,4]

..................  # qu'obtenez-vous ?

>>> L.append(15)

..................  # qu'obtenez-vous ?

>>> L.sort()

..................  # qu'obtenez-vous ?

>>> L

..................  # qu'obtenez-vous ?

>>> L,M=[1,2,3],[4,5,6]

..................  # qu'obtenez-vous ?

>>> L

..................  # qu'obtenez-vous ?

>>> M

..................  # qu'obtenez-vous ?

>>> 3*L

..................  # qu'obtenez-vous ?

>>> L+M

..................  # qu'obtenez-vous ?

>>> L[15]

..................  # qu'obtenez-vous ?

>>> L+"hello"

..................  # qu'obtenez-vous ?

>>> L+5

..................  # qu'obtenez-vous ?
```

3 *Exercice*

(Copie de liste : la mauvaise méthode) Compléter, puis vérifier avec la console :

```
>>> L=[1,2,3,4]

..................  # qu'obtenez-vous ?

>>> M=L

..................  # qu'obtenez-vous ?

>>> M

..................  # qu'obtenez-vous ?


>>> M[0]="hello"

..................  # qu'obtenez-vous ?

>>> M

..................  # qu'obtenez-vous ?

>>> L

..................  # qu'obtenez-vous ?
```

4 *Exercice*

(Copie de liste : la bonne méthode) Compléter, puis vérifier avec la console :

```
>>> L=[1,2,3,4]

..................  # qu'obtenez-vous ?

# on utilise la méthode .copy()
>>> M=L.copy()

..................  # qu'obtenez-vous ?

>>> M

..................  # qu'obtenez-vous ?

>>> M[0]="hello"

..................  # qu'obtenez-vous ?

>>> M

..................  # qu'obtenez-vous ?

>>> L

..................  # qu'obtenez-vous ?
```

Explication **(À savoir)** Pour copier une liste `L` dans la variable `M` et que la nouvelle liste `M` ainsi créée soit indépendante de `L` (i.e. que les modifications effectuées sur l'une n'impactent pas l'autre), il suffit d'utiliser la méthode `.copy()`.

Cliquez ici pour accéder au corrigé

II - Les fonctions

Dans cette partie II, nous écrirons le code des fonctions dans l'éditeur !

✌ **Explication** ✌ Rappelons que l'on définit une fonction de la manière suivante :

- on débute avec le mot clé `def` suivi du nom de la fonction avec d'éventuels paramètres que l'on met entre parenthèses, et on termine cette première ligne avec deux points « : »,
  - ↪ le nom d'une fonction doit être explicite, sans majuscule, ni accents et espaces,
  - ↪ les noms des paramètres suivent la même règle tout en étant les plus courts possibles.
- les (éventuelles) instructions seront indentées,
- on termine avec `return`.

Exemple Dans l'ÉDITEUR, on définit une fonction `hypotenuse(a,b)` qui prend en paramètres deux réels `a` et `b` et renvoie la longueur de l'hypoténuse du triangle rectangle dont les côtés de l'angle droit ont pour longueur `a` et `b`.

```
# pour utiliser la fonction racine carrée
from math import *

def hypotenuse(a,b):
    c=sqrt(a**2+b**2)
    return c
```

Ensuite dans la CONSOLE, on peut tester notre fonction :

```
>>> hypotenuse(3,4)
5.0
```

Exemple (Fonction sans paramètre)
La fonction `age()` suivante n'a pas de paramètre en entrée et est définie dans l'ÉDITEUR par les lignes de code suivantes :

```
def age():
    a=input("Quelle est ton année de naissance ? ")
    print("Tu as ", 2018-int(a), "ans.")
    return None
```

Recopier les lignes de code précédentes dans l'éditeur, puis tester ensuite la fonction age() dans la console :

```
>>> age()
```

5 Exercice

Écrire une fonction

`trinome(x,a,b,c)`

qui prend en paramètres des réels `x,a,b,c` et renvoie en sortie le réel défini par ax^2+bx+c. Tester votre fonction !

6 Exercice

1) Écrire une fonction

`aire_triangle(b,h),`

qui prend en paramètres la base `b` et la hauteur correspondante `h` d'un triangle, et renvoie son aire.

2) Que doit-on saisir dans la console pour calculer l'aire du triangle de base 10cm et de hauteur correspondante 5cm ?

7 Exercice

1) a) Écrire une fonction

`conversion1(j,h,m,s),`

qui prend en paramètres un nombre de jours `j`, un nombre d'heures `h`, un nombre de minutes `m` et un nombre de secondes `s`, et renvoie la conversion de cette durée en secondes.

b) À l'aide de votre fonction, convertir la durée suivante en secondes :

3j 10h 55m 34s

c) Quel est le type de la donnée en sortie de la fonction `conversion1` ?

2) a) Écrire une fonction

`conversion2(j,h,m,s),`

qui affiche en sortie dans une phrase le résultat attendu.
Exemple

☞ *Cliquez ici pour accéder au corrigé*

```
>>> conversion2(1,0,0,0)
1j 0h 0m 0sec équivaut à 86400s.
```

b) Tester votre fonction sur la durée suivante :

3j 10h 55m 34s

c) Quel est le type de la donnée en sortie de la fonction `conversion2` ?

8 *Exercice*

1) Écrire une fonction

`echange1(L),`

qui prend en paramètre une liste `L` d'au moins deux éléments, et renvoie en sortie une nouvelle liste semblable à celle saisie en entrée, mais dont les premier et dernier éléments ont été échangés.
Exemple La fonction codée doit passer le test suivant :

```
>>> L=[1,3,4,6]

>>> echange1(L)
[6,3,4,1]

>>> L
[1,3,4,6]
```

2) Écrire une fonction

`echange2(L),`

qui prend en paramètre une liste `L` d'au moins deux éléments, et renvoie en sortie la liste `L` modifiée avec ses premier et dernier éléments échangés.
Exemple La fonction codée doit passer le test suivant :

```
>>> L=[1,3,4,6]

>>> echange2(L)
[6,3,4,1]

>>> L
[6,3,4,1]
```

3) Écrire une fonction

`echange3(L,n),`

qui prend en paramètres une liste `L` et un entier `n`, et renvoie en sortie la liste `L` modifiée avec ses éléments d'indices `n` et `n+1` échangés.
Exemple La fonction codée doit passer le test suivant :

```
>>> L=[1,3,4,6,18,-5]

>>> echange3(L,2)
[1,3,6,4,18,-5]

>>> L
[1,3,6,4,18,-5]
```

9 *Exercice*

Écrire une fonction

`init_listes(a,b,n,p),`

qui prend en paramètres des objets `a` et `b`, des entiers `n` et `p`, et renvoie la liste formée de `n` fois l'objet `a` suivi de `p` fois l'objet `b`.

10 *Exercice*

Écrire une fonction sans argument

`welcome()`

qui demande d'abord l'âge, puis le prénom, pour afficher en sortie une phrase de bienvenue du type :

Bienvenue à Thomas âgé de 15 ans !

si l'utilisateur a répondu `Thomas` et `15` aux deux questions posées.

11 *Exercice*

Écrire une fonction

`reverse(x),`

qui prend en paramètre un réel `x` de deux chiffres et supérieur à 10, et renvoie le réel où les deux chiffres ont été permutés.
Exemple La fonction codée doit passer le test suivant :

```
>>> reverse(17)
71
```

Explication Pour ceux qui ont tout fini, faire les exercices du *Cours 3 - La boucle `For`*.

Cliquez ici pour accéder au corrigé