# 12 Exercices

**Exo 12.1**

**1.** Lecture

**a.** Écrire la fonction `affiche`, de paramètre un tableau, affichant les valeurs de ce tableau :

```
>>> note = [10, 6, 16, 20, 8]
>>> affiche(note)
10
6
16
20
8
```

**b.** Modifier votre fonction pour qu'elle n'affiche que les notes au moins égales à 10

```
>>> note = [10, 6, 16, 20, 8]
>>> affiche(note)
10
16
20
```

**2.** Écriture

**a.** Écrire la fonction `plus_un`, de paramètre un tableau, augmentant de 1 chacune de ses valeurs sans dépasser 20 :

```
>>> note = [10, 6, 16, 20, 8]
>>> plus_un(note)
>>> note
[11, 7, 17, 20, 9]
```

**b.** Modifier votre fonction pour qu'elle ne modifie pas le tableau de départ, mais en renvoie un nouveau :

```
>>> note = [10, 6, 16, 20, 8]
>>> plus_un(note)
[11, 7, 17, 20, 9]
>>> note
[10, 6, 16, 20, 8]
```

**3.** Drapeaux

**a.** Écrire la fonction `est_dans` de paramètres une valeur et un tableau et renvoyant `True` ou `False` selon si la valeur est dans le tableau ou non.

```
>>> note = [10, 6, 16, 20, 8]
>>> est_dans(16, note)
True
>>> est_dans(15, note)
False
```

**b.** Écrire la fonction `positionne` de paramètre un tableau et renvoyant un triplets de booléens indiquant si certaines des valeurs sont, respectivement, inférieures ou égales à 10, strictement comprises entre 10 et 15, et supérieures ou égales à 15.

```
>>> note = [10, 6, 16, 20, 8]
>>> positionne(note)
(True, False, True)
```

**4.** Accumulateurs

**a.** Écrire la fonction `nombre_de_plus_grand` de paramètres un tableau et une valeur et renvoyant le nombre valeurs du tableau supérieures égales à cette valeur.

```
>>> note = [10, 6, 16, 20, 8]
>>> nombre_de_plus_grand(note, 10)
3
```

**b.** Écrire la fonction `somme` qui renvoie la somme des valeurs d'un tableau.

```
>>> note = [10, 6, 16, 20, 8]
>>> somme(note)
60
```

Modifier cette fonction en la fonction `moyenne`, renvoyant la moyenne des valeurs.

**c.** Écrire la fonction `maximum` de paramètres un tableau renvoyant sa plus grande valeur de ce tableau.

```
>>> note = [10, 6, 16, 20, 8]
>>> maximum(note)
20
```

**d.** Écrire la fonction `list_to_str` de paramètre un tableau, qui renvoie une chaine de caractère contenant chaque valeur du tableau séparées par `' - '`.

```
>>> note = [10, 6, 16, 20, 8]
>>> list_to_str(note)
'10 - 6 - 16 - 20 - 8'
```

**Exo 12.2** YAM'S.

**1.** Écrire la fonction qui renvoie le nombre de 6 dans un tableau d'entiers.

```
>>> lancer = [3, 6, 2, 3, 3]
>>> nb_six(lancer)
1
```

**2.** La modifier pour que « 6 » soit une entrée.

```
>>> lancer = [3, 6, 2, 3, 3]
>>> nombre(lancer, 6)
1
```

**3.** Écrire la fonction `brelan` qui renvoie la somme de trois résultats identiques du tableau s'il y en a, ou `0` sinon.

```
>>> lancer = [3, 6, 2, 3, 3]
>>> brelan(lancer)
9
>>> lancer = [3, 6, 2, 3, 2]
>>> brelan(lancer)
0
```

**Exo 12.3** MASTER MIND.

1. Écrire la fonction prenant en paramètres deux tableaux de même taille et renvoyant le nombre de valeurs communes aux mêmes indices.

```
>>> tab1 = [1,6,3,3,2]
>>> tab2 = [1,3,2,3,1]
>>> bonne_place(tab1, tab2)
2
```

2. La compléter pour qu'elle renvoie aussi, parmi les autres valeurs, le nombre de valeurs identiques, mais pas aux mêmes indices.

```
>>> tab1 = [1,6,3,3,2]
>>> tab2 = [1,3,2,3,1]
>>> bonne_place(tab1, tab2)
(2, 2)
```

**Exo 12.4** MORPION.

1. Écrire la fonction qui renvoie le nombre de `'X'` dans un tableau représentant un alignement possible de trois valeurs parmi `'X'`, `'O'` et `'_'` d'une grille de morpion.

```
>>> alignement = ['X', '_', 'X']
>>> nb_X(alignement)
2
```

2. La modifier pour que « `'X'` » soit une entrée.

```
>>> alignement = ['O', 'X', 'X']
>>> nombre(alignement, 'X')
2
>>> nombre(alignement, 'O')
1
```

3. On modélise une grille de morpion par le tableau de ses trois lignes, et ses lignes par des tableau de `'X'`, `'O'` et `'_'`
Écrire la fonction `liste_des_alignements` qui renvoie le tableau de tous les alignements possibles :

```
>>> grille = [ ['X', '_', 'O'],
...            ['O', 'X', 'X'],
...            {'_', '_', '_'],
...          ]
>>> liste_des_alignements(grille)
[['X', '_', 'O'], ['O', 'X', 'X'],
↪  ['_', '_', '_'], ['X', 'O', '_'],
↪  ['_', 'X', '_'], ['O', 'X', '_'],
↪  ['X', 'X', '_'], ['_', 'X', 'O']]
```

**Exo 12.5** DÉMINEUR.

On modélise la grille de jeu par un tableau de ses lignes, chaque ligne étant modélisée par un tableau dont les valeurs sont `'*'` (une bombe) ou `' '` (rien).

1. Écrire la fonction prenant en paramètres le tableau d'une grille de jeu, un numéro de ligne et un numéro de colonne et renvoyant un booléen indiquant s'il y a une bombe ou non à cette emplacement.

```
>>> grille = [ [' ', '*', ' ', ' ', ' '],
...            [' ', ' ', '*', ' ', ' '],
...            [' ', ' ', ' ', ' ', ' '],
...            [' ', ' ', ' ', ' ', '*'],
...            ['*', ' ', ' ', ' ', ' '],
...          ]
>>> bombe(grille, 0, 1)
True
```

2. Écrire la fonction prenant en paramètres le tableau d'une grille de jeu, un numéro de ligne et un numéro de colonne et renvoyant le nombre de bombes autour de la case correspondante dans la grille.

```
>>> grille = [ [' ', '*', ' ', ' ', ' '],
...            [' ', ' ', '*', ' ', ' '],
...            [' ', ' ', ' ', ' ', ' '],
...            [' ', ' ', ' ', ' ', '*'],
...            ['*', ' ', ' ', ' ', ' '],
...          ]
>>> bombe(grille, 0, 2)
2
```

**Exo 12.6** AUTRES SUJETS - JEU DE LA VIE - I.A. PIERRE/FEUILLE/CISEAUX - ...