INFO-H-100 - Programmation
TP 14 — Programmation fonctionnelle.

Lorsque l'exercice demande d'écrire une fonction, écrivez la fonction demandée et testez-la avec plusieurs valeurs pertinentes.

N'utilisez pas de boucles mais les fonctions `map`, `reduce` et `filter`.

Ex. 1. Ecrire une fonction qui renvoie une liste contenant la valeur absolue de chaque élément d'une liste.

```
>>> map_abs([-1,2,-4,0])
[1,2,4,0]
```

Ex. 2. Ecrire une fonction qui renvoie une liste composée de chaque élément d'une liste divisé par 100.

```
>>> list_divide100(range(0,500))
[0.0, 0.01, 0.02, ... 4.99]
```

Ex. 3. Ecrire une fonction qui renvoie une liste composée de chaque élément d'une liste divisé par la taille de la liste. Utilisez un `lambda`.

```
>>> list_divide(range(0,100))
[0.0, 0.01, 0.02, 0.03 ... 0.99]
```

Ex. 4. Ecrire une fonction qui renvoie une liste composée des entiers pairs d'une liste d'entiers. Ecrire une version sans `lambda` et une version avec un `lambda`.

```
>>> list_even([4,1,2,8,7])
[4,2,8]
```

Ex. 5. Ecrire une fonction qui renvoie une liste composée des entiers divisibles par n d'une liste d'entiers. Utilisez un `lambda`.

```
>>> divisibles(4,[4,1,2,8,7])
[4,8]
```

Ex. 6. Ecrire une fonction qui renvoie le produit des éléments d'une liste.

```
>>> list_product(range(1,5))
24
```

Ex. 7. Ecrire une fonction qui renvoie le maximum d'une liste d'entiers (ne pas utiliser la fonction `max`).

```
>>> list_max([2,7,1,8,1])
8
```

Ex. 8. Il est permis de passer plusieurs itérables (listes, tuples, chaînes, ...) à la fonction `map`. Par exemple :

```
>>> map(lambda x,y:x+y, [1,2,3,4], [4,3,2,1])
[5,5,5,5]
```

Ecrire une fonction qui renvoie le produit scalaire de deux vecteurs représentés par des tuples.

```
>>> produit_scalaire((3,4),(5,6))
39
```

Ex. 9. Ecrire une fonction qui renvoie une liste composée du sinus de chaque élément d'une liste.

```
>>> map_sinus(list_divide100(range(0,500)))
[0.0, 0.009999833334166664, 0.01999866669333308, ..., -0.9617129034267934]
```

Ex. 10. Soit un dictionnaire d'occurrences `(char,int)` comme ci-dessous.

```
{'A':10,'T':3,'C':5,'G':2}
```

Ecrire une fonction qui renvoie une liste composée des lettre qui ont au moins x occurrences dans ce dictionnaire. Utilisez un `lambda`.

```
>>> filter_occurrences(5,dico)
['A','C']
```

Ex. 11. Ecrire une fonction qui renvoie le nombre de palindromes contenu dans une liste.

```
>>> count_palindromes(['radar','python','sos','socrate']):
2
```

Ex. 12. Ecrire une fonction qui renvoie la factorielle de n en utilisant un `reduce`, un `lambda` et un `range`.

Ex. 13. Ecrire une fonction qui calcule l'écart-type d'une liste. Pour calculer l'écart-type, utiliser la formule 1.

$$\begin{aligned} s_1 &\leftarrow \sum_{0 \leq i < n} (A_i^2) \\ s_2 &\leftarrow \frac{1}{n} \left(\sum_{0 \leq i < n} A_i \right)^2 \\ v &\leftarrow \frac{s_1 - s_2}{n} \\ s &\leftarrow \sqrt{v} \end{aligned} \tag{1}$$

Ex. 14. Ecrire une fonction qui met en majuscule le premier mot de chaque phrase d'un texte. Un texte est représenté par une liste de mots. Le dernier mot de chaque phrase est terminé par un point.

```
>>> cap(["see", "spot", "run.", "run", "spot", "run."])
['See', 'spot', 'run.', 'Run', 'spot', 'run.']
```

Ex. 15. Sans utiliser de boucle, écrire une fonction qui reçoit une matrice (sous la forme d'une liste de listes d'entiers) et qui renvoie le nombre d'entiers pairs qu'il y a dans la matrice.

Bonus : écrire une fonction qui reçoit une matrice et un prédicat, et qui renvoie le nombre d'entiers qui satisfont le prédicat. (Un prédicat est une fonction qui renvoie un booléen.)

INFO-H-100 - Programmation
TP 14 — Programmation fonctionnelle.
Corrections

Solution de l'exercice 1:

```
def map_abs(ls):
    return list(map(abs,ls))
```

Solution de l'exercice 2:

```
def divide100(x):
    return float(x)/100

def list_divide100(ls):
    return list(map(divide100, ls))
```

Solution de l'exercice 3:

```
def list_divide(ls):
    return list(map(lambda x:float(x)/len(ls),ls))
```

Solution de l'exercice 4:

```
def is_even(n):
    return n%2==0

def list_even(ls):
    return list(filter(is_even,ls))
```

Version avec `lambda` :

```
def list_even(ls):
    return list(filter(lambda x:x%2==0,ls))
```

Solution de l'exercice 5:

```
def divisibles(n,ls):
    return list(filter(lambda x:x%n==0, ls))
```

Solution de l'exercice 6:

```
import functools
def list_product(ls):
    return functools.reduce(lambda x,y:x*y,ls)
```

Solution de l'exercice 7:

```
def max2(a, b):
    if a > b:
        return a
    else:
        return b

def list_max(ls):
    return functools.reduce(max2, ls)
```

Version avec `lambda` :

```
def list_max(ls):
    return functools.reduce(lambda x,y: x if x > y else y, ls)
```

Solution de l'exercice 8:

```
def product(a,b):
    return a*b

def scalar_product(v1,v2):
        return sum(map(product,v1,v2))
```

Version avec `lambda` :

```
def scalar_product(v1,v2):
    return sum(map(lambda x,y:x*y,v1,v2))
```

Solution de l'exercice 9:

```
def divide100(x):
    return float(x)/100

def list_divide100(ls):
    return list(map(divide100, ls))

def map_sinus(ls):
    import math
    return list(map(math.sin,ls))
```

Solution de l'exercice 10:

```
def filter_occurrences(n,dico):
    return list(filter(lambda x:dico[x]>=n, dico))
```

Solution de l'exercice 11:

```
def is_palindrome(word):
    return word == word[::-1]

def count_palindromes(ls):
    return len(list(filter(is_palindrome, ls)))
```

Solution de l'exercice 12:

```
def factorial(n):
    """n > 0"""
    return functools.reduce(lambda x, y : x * y, range(1, n + 1))
```

Solution de l'exercice 13:

```
import math

def square(a):
    return float(a**2)

def std(ls):
    n = len(ls)
    s1 = sum(map(square,ls))     # s1 = sum(map(lambda a: a**2, ls))      (sans la fonction square)
    s2 = square(sum(ls))/n       # s2 = float(sum(ls)**2)/n               (sans la fonction square)
    v = (s1 - s2)/n
    return math.sqrt(v)
```

Solution de l'exercice 14:

```
def cap(ls):
    return list(map(chose,ls,['.']+ls[:-1]))

def chose(elem1,elem2):
    if elem2[-1] == '.':
        return elem1.capitalize()
    else:
        return elem1
```

Solution de l'exercice 15:

```python
def count_even(mat):
    return bonus(mat, lambda x: x%2==0)

def bonus(mat, pred):
    return sum(map(lambda ligne: len(list(filter(pred, ligne))), mat))
```