LE PARADIGME FONCTIONNEL - EXERCICES

Exercice 0.a : Réécrire le code suivant avec un opérateur ternaire en 2 lignes.

```
age = int(input())
resultat = None

if age >= 18 :
      resultat = "Tu peux voter."
else :
      resultat = "Tu ne peux pas voter."
print(resultat)
```

Exercice 0.b : Qu'affiche le code suivant ?

```
a = 3
print(("zero" if a == 0 else "positive") if a >= 0 else "negative")
```

Exercices 1 : Fonctions anonymes

En utilisant la notation `lambda`, écrire une fonction anonyme qui :

1. Retourne `True` si un entier passé en paramètre est pair, `False` sinon.
2. Retourne `True` si le paramètre est un entier, `False` sinon.
3. Retourne la somme de deux entiers passés en paramètre.
4. Retourne `True` si l'entier passé en paramètre représente une année bissextile, `False` sinon.

Exercices 2 : Trouve

Écrire une fonction `trouve(propriete, l)` qui reçoit en arguments une fonction `propriete(x) -> bool` et une liste `l` et renvoie le premier élément `x` de `l` tel que `propriete(x)` vaut `True`. Si aucun élément de `l` ne satisfait `propriete`, alors la fonction renvoie `None`.

Exercices 3 : Filtre

Écrire une fonction `filtre(propriete, l)` qui reçoit en arguments une fonction `propriete(x) -> bool` et une liste `l` et renvoie une liste contenant tous les éléments `x` de `l` tel que `propriete(x)` vaut `True`. Si aucun élément de `l` ne satisfait `propriete`, alors la fonction renvoie une liste vide.

Exercices 4 : Applique

Écrire une fonction `applique(fonction, l)` qui reçoit en arguments une fonction `fonction` et une liste `l` et renvoie une nouvelle liste, de même taille, où la fonction `fonction` a été appliquée à chaque élément de `l`.

Exercice 5 : Tri

```
liste = ['Libellule', 'Paille', 'Cool', 'Ada Lovelace', 'Parallèle']
```

En tulisant la fonction `sorted`, trier `liste` dans l'ordre croissant du nombre de lettre `'l'` (minusucle ou majuscule) :

1. En utlisant une fonction nommé
2. En utilisant une fonction lambda. (On pourra utiliser la fonction len sur une liste en comprehension).

Exercices 6 : Double

Écrire une fonction `double(f)` qui reçoit une fonction `f(x)` en argument et renvoie une fonction qui applique deux fois de suite la fonction `f` à son argument.
Sans exécuter le code, que vaut `double(double(lambda x : x*x))(2)` ?

Exercices 7 : Composition

Écrire une fonction `compose(f, g)` qui reçoit en arguments deux fonctions `f` et `g` et renvoie leur composition, c'est à dire la fonction h telle que, pour tout x, h(x) est égale à f(g(x)).

Exercice 8 : Ensembles comme des fonctions

Une manière simple de représenter un ensemble d'entiers S est de définir la fonction caractéristique de S, nommée `car_S`, telle que `car_S(x)` renvoie `True` si et seulement si $x \in S$. Par exemple, l'ensemble S_1 constitué de tous les entiers strictement positifs sera représenté par la fonction `car_S1` suivante :

```
def car_S1(x): return x > 0
```

Ainsi, pour savoir si l'entier 4 est dans S_1 , il suffit d'appeler la fonction `car_S1(4)`, qui renvoie `True`, tandis que `car_S1(-3)` renvoie `False`.

Question 1 : Donner les fonctions caractéristiques des ensembles S_2 , S_3 , S_4 et S_5 suivants :
• S_2 est l'ensemble de tous les entiers (positifs ou nuls) qui sont pairs ;
• S_3 est l'ensemble contenant uniquement les entiers 0, 3, 7 et 9 ;
• S_4 est l'ensemble des entiers entre 10 et 1000 (inclus) ;
• S_5 est l'ensemble vide.

Question 2 : En utilisant cette représentation, on peut encoder les opérations habituelles sur les ensembles comme des fonctions d'ordre supérieur. Par exemple, l'opération d'union $S_1 \cup S_2$ de deux ensembles S_1 et S_2 peut être définie par la fonction `union(car_S1, car_S2)` qui renvoie la fonction caractéristique de l'union des ensembles S_1 et S_2 représentés respectivement par les fonctions `car_s1` et `car_S2`. Écrire la fonction `union` en Python.

Question 3 : De la même manière, donner la fonction `inter` qui réalise l'intersection de deux ensembles.

Question 4 : Donner la fonction `ajout` telle que `ajout(x, car_S)` renvoie la fonction caractéristique de l'ensemble dans lequel `x` a été ajouté à l'ensemble dont la fonction caractéristique est `car_S`.

Question 5 : En utilisant la fonction précédente, donner la fonction ensemble telle que `ensemble(l)` convertit une liste `l` en une fonction caractéristique de l'ensemble qui contient les valeurs de `l`.

Exercice 9 : Hors programme - `map`, `reduce`, `filter`.

Pour écrire les fonctions suivantes, utilisez les fonctions `map`, `reduce` et `filter`

Question 1 : `carre(l : list)` qui renvoie la liste des carrés des éléments d'une liste.

```
print(carre([1, 2, 3, 4]))
```

```
[1, 4, 9, 16]
```

Question 2 : `maximum(l : list)` qui renvoie le plus grand élément d'une liste non vide.

```
print(maximum([5, -2, 7, 4]))
```

```
7
```

Question 3 : `passe-bande(l : list)` qui ne renvoie que une liste des éléments compris entre les valeurs 5 et 10 incluses de la liste.

```
print(passe-bande([5, -2, 7, 4]))
```

```
[5, 7]
```

Question 4 : `fst(l : list)` qui renvoie une liste constituée des éléments gauche d'une liste de paires (tuples avec 2 valeurs).

```
print(fst([(1, 2), (-1, 2), (6, 5), (4, -3)]))
```

```
[1, -1, 6, 4]
```

Question 5 : `permute(l : list[int])` qui remplace les 0 par des 1 (et réciproquement) dans une liste `l` (constituée uniquement de 0 et de 1).

```
print(permute([0, 1, 0, 1, 0, 0, 1, 1]))
```

```
[1, 0, 1, 0, 1, 1, 0, 0]
```

Question 6 : `compte0(l : list[int])` qui compte le nombre de 0 dans la liste `l`.

```
print(compte0([0, 1, 0, 1, 0, 0, 1, 1]))
```

```
4
```

Question 7 : `pgs(v,l : list)` qui renvoie la longueur de la plus grande séquence de `v` dans la liste `l`.

```
print(pgs(1, [0, 1, 1, 0, 0, 2, 2, 2, 1, 1, 1, 0, 0, 2, 1]))
```

```
3
```

Question 8 : `somme_des_carres(l : list[int])` qui renvoie la somme des carrés des éléments d'une liste.

```
print(somme_des_carres([1, 2, 3, 4]))
```

```
30
```

Question 9 : `somme_des_carres_positif(l : list[int])` qui renvoie la somme des carrés des éléments positifs d'une liste.

```
print(somme_des_carres_positif([2, -2, 4, -5, -6]))
```

```
20
```

Question 10 : `moyenne_des_notes_positives(l : list[int])` qui renvoie la moyen des notes positive d'une liste.

```
print(moyenne_des_notes_positives([12, -8, 10, -5, 17]))
```

```
13.0
```

Exercice 10 : Hors programme - Curryfication

On dispose de la fonction `distance_euclidienne` suivante qui calcule la distance euclidienne entre deux points de l'espace.

```
from math import sqrt

def distance_euclidienne(xa, ya, za, xb, yb, zb):
    return sqrt( (xb - xa)**2 + (yb - ya)**2 + (zb-za)**2)

print(distance_euclidienne(1, 1, 5, 4, 5, 5))
```

Curryfiez la fonction pour que les appels suivants fonctionnent :

Question 1 :

```
print( distance_euclidienne_c1 (1)(1)(5)(4)(5)(5) )
```

Question 2 :

```
print( distance_euclidienne_c2 (1, 1, 5)(4, 5, 5) )
```