UNIVERSITE IBA DER THIAM DE THIES
UFR SET - Département Informatique
COURS : POO 2 / 2022 - 2023
M. DIOUF

Fonction – Procédure – Liste – Dictionnaire

Exercice 1 :
Le but présumé de ce jeu est de jeter des dés en tentant d'atteindre un certain total par jet.
Le but de cet exercice est d'écrire une fonction qui, pour une valeur donnée, renvoie toutes les combinaisons possibles de 3 dés (dés classiques allant de 1 à 6) pouvant donner cette valeur.
Si la solution (1, 2, 3) convient pour la valeur 6 alors la solution (2, 3, 1) ne peut plus convenir (les dés sont interchangeables). Attention, les solutions doivent être uniques.
Exemple : pour la valeur 7 voici les combinaisons [(1, 1, 5), (1, 2, 4), (1, 3, 3), (2, 2, 3)]

Exercice 2 :
Soit des comptes bancaires d'individus définis par la liste :

```
comptes = [ {'nom': 'Diouf', 'prenom': 'Modou', 'epargne': 2500},
            {'nom': 'Sene', 'prenom': 'Pathé', 'epargne': 5000},
            {'nom': 'Ngom', 'prenom': 'Khadim', 'epargne': 10000},
            {'nom': 'Ndiaye', 'prenom': 'Fatou', 'epargne': 1250},
            {'nom': 'Mbengue', 'prenom': 'Alice', 'epargne': 2500},
            {'nom': 'Déme', 'prenom': 'Maty'},
            {'nom': 'Ngom', 'prenom': 'Satou', 'epargne': 4530},
            {'nom': 'Thioune', 'prenom': 'Ameth', 'epargne': 2200}]
```

On considère que les individus qui portent le même 'nom' sont de la même famille.
En cas d'absence de revenu attribué à un individu, nous considérerons que son épargne est nulle (cas de 'Maty Déme').
Écrire une fonction qui retourne le nom de la famille à la plus faible épargne ainsi que le nom de la famille à la plus forte épargne avec le montant de leur épargne respective.
Ici, ('Ndiaye', 1250) et ('Ngom', 10000).

Exercice 3 :
Soit deux listes L1 et L2 permettant de coder deux entiers naturels.
Par exemple : L1 = [9, 4, 0] pour coder le nombre 940 et L2 = [8, 3] pour coder le nombre 83 *.
* On s'interdit les nombres commençant par 0 (à l'exception du chiffre 0).
Ainsi, le nombre 83 sera codé [8, 3], et non [0, 8, 3].
Le but est d'écrire une fonction "somme_2nombres()" qui fait la somme L1 + L2 et de retourner le résultat dans le même format de liste : [1, 0, 2, 3] car 940 + 83 = 1023

```
>>> somme_2nombres([9, 4, 0], [8, 3])
>>> [1, 0, 2, 3]
>>> somme_2nombres([1, 9, 3], [7]))
>>> [2, 0, 0]
```

```
  1
  9  4  0
+    8  3
----------
1  0  2  3
```

Exercice 4 :

Soit une liste d'entiers, le but de l'exercice est d'écrire une fonction "somme_couple()" qui renvoie les couples des indices d'éléments de la liste, de telle sorte que la somme de ces deux éléments est égale à la valeur cible choisie.

Exemple :

```
>>> somme_couple([-2, -1, 2, 1], cible = 0)
>>> [(0, 2), (1, 3)]
```

Explications : soit [-2, -1, 2, 1]

- Les deux nombres aux indices 0 et 2, c-à-d respectivement -2 et 2, donnent la cible = 0 si on les additionne (-2 + 2 = 0) ;
- De même pour le couple (1, 3) où les éléments correspondants aux indices 1 et 3, ç-à-d respectivement -1 et 1, donnent aussi la cible 0 lorsqu'on les additionne (-1 + 1 = 0).

Spécifications :

1. Si la cible ne peut être atteinte avec les entrées, la fonction renvoie une liste vide [] ;
2. Les indices renvoyés dans un couple doivent être différents : (i, j) tels que $i \neq j$;
3. Pour un couple d'indices donné, vous pouvez renvoyer indifféremment : (i, j) ou (j, i), mais pas les deux ;
4. De même, l'ordre des couples dans la liste n'est pas important.

```
>>> somme_couple([2, 9, 5, 3, -1], cible = 6)
>>> []
>>> somme_couple([-2, -2, -1, -1, 1, 2, 2], cible = 0)
>>> [(0, 5), (0, 6), (1, 5), (1, 6), (2, 4), (3, 4)])
```

Exercice 5 :

Dans cet exercice, une liste "sieges" de dimension n représente une rangée de n sièges.
Si sieges[i]=1, le siège à l'emplacement i ($0 \leq i \leq n-1$) est occupé par une personne.
Si sieges[i]=0, le siège à l'emplacement i est libre. Ex. : sieges = [0, 1, 0, 0, 1, 0, 0, 0, 1].
Dans la rangée, au moins une place de siège doit être libre, et au moins un siège est occupé.
Dans le respect de la distanciation physique, il faut trouver un siège libre qui maximise la distance avec le plus proche voisin.

```
                  i = 6
                   |
sieges = [0, 1, 0, 0, 1, 0, 0, 0, 1]
              |<--->|<--->|
                2     2
```

Ici, sieges[6] = 0 est l'emplacement libre qui maximise la distance, car le plus proche voisin est à une distance égale à 2.
Tous les autres emplacements libres donneraient au moins un voisin à une distance inférieure à 2.
Pour respecter la distanciation physique, il faut se placer sur le premier siège libre à gauche, où le plus proche voisin est alors à trois emplacements.
Écrire la fonction max_distance() qui prend une rangée de sièges occupés ou libres en paramètre, et qui renvoie pour un emplacement libre la plus grande distance avec le voisin le plus proche.