M1102: Introduction à l'algorithmique et à la programmation

Feuille de TD n°9

Les dictionnaires pour les structures de données hétérogènes

INSTRUCTIONS: à la fin du TD, vous devez déposer sous Celene une archive zip contenant les fichiers Python correspondant à ce TD (fournis avec la feuille). Vous pourrez déposer une version améliorée de votre travail jusqu'à la **VEILLE** du TD suivant.

L'évaluation de la période 2 sera basée sur ces rendus.

Objectifs

Pour continuer vers la programmation orientée objet, l'objectif de cette feuille est de montrer comment on peut utiliser les dictionnaires Python pour représenter des entités. Dans l'exercice sur les serpents de la feuille TD 6, les serpents sont représentés par des triplets de la forme `(nom,longueur,danger)`. Par conséquent, il faut savoir que pour un serpent `s`, `s[0]` représente le nom, `s[1]` représente la longueur et `s[2]` le niveau de danger. Si l'on décide de rajouter une information au serpent comme par exemple son genre, le serpent sera représenté par un tuple qui pourra être de la forme suivante : `(nom,genre,longueur,danger)`. Malheureusement, si le genre est inséré en 2ème position comme ici, `s[1]` ne représente plus la longueur mais le genre ! Il faut donc corriger toutes les fonctions qui utilisaient `s[1]` pour accéder à la longueur. Pour éviter ce genre de désagréments, on peut utiliser les dictionnaires qui vont permettre de donner un nom à chacune des propriétés du serpent. Par exemple le serpent `s=('Python3',0.3,0)` pourra être représenté par le dictionnaire `sd={'nom':'Python3','longueur':0.3,'danger':0}`. Pour accéder au nom du serpent il suffit d'utiliser `sd['nom']`, pour la longueur `sd['longueur']` etc. Évidemment si l'on rajoute l'information du genre, l'accès aux autres informations reste inchangé.
De manière générale une entité (comme vous pouvez en trouver dans un MCD) peut se représenter en Python sous la forme d'un dictionnaire.

Exercice 1 *Développement durable*

Dans le cadre d'une étude sur le développement durable et les moyens de transport on souhaite créer un fichier permettant de stocker des informations sur les individus et leurs moyens de transport. Voici les informations que l'on souhaiterait stocker :

- le nom de la personne
- son âge
- son moyen de transport privilégié

1. Définissez une manière de représenter une personne et implémentez l'API permettant de manipuler ces personnes. Les fonctions de l'API se trouvent dans le fichier `transport.py`. Si votre API est correcte, vous pouvez lire le fichier de personnes `personnes.txt` qui vous retournera une liste de personnes. Les fonctions suivantes travaillent sur des listes de personnes.
2. Écrire une fonction qui affiche une liste de personnes
3. Écrire une fonction qui retourne la moyenne d'âge des utilisateurs d'un moyen de transport passé en paramètre

4. Écrire une fonction qui retourne la liste des moyens de transport utilisés par des personnes d'une liste (on ne veut pas de doublons).

Le script `test_transport.py` permet de passer des tests unitaires sur cette exercice. Ces tests ne peuvent se faire que si vous avez implémenter la fonction Personne.

Exercice 2 *Nouvelles matrices*

Constant Denlechangeman, votre chef de projet, a encore décidé de changer la représentation des matrices. Il veut, cette fois, utiliser un dictionnaire contenant les informations suivantes : le nombre de lignes, le nombre de colonnes et un champ contenant les valeurs (stockées comme vous le souhaitez).
Écrire cette nouvelle implémentation dans le fichier `matriceAPI3.py` et vérifier que vos fonctions des feuilles précédentes sur les matrices marchent toujours.
Attention ! On rappelle que les fonction sur les matrices doivent rendre des résultats corrects et cohérents, que l'on fasse l'import de `matriceAPI1`, `matriceAPI2` ou `matriceAPI3`.
Si vous ne l'avez pas fait implémentez les fonction de la feuille TD8

Exercice 3 *Sudoku*

Constant Denlechangeman, votre chef de projet est un fan de Sudoku ! Il vous demande d'écrire des fonctions qui lui permettent de l'aider à résoudre ses Sudoku. Les règles de ce jeu peuvent être trouvées ici `http://fr.wikipedia.org/wiki/Sudoku`. Les Sudoku seront représentés par des matrices dans lesquels les cases non remplies vaudront `None`. Dans le répertoire `sudoku`, les fichiers `sudoku*.txt` sont des fichiers de matrices contenant des grilles complètes ou non de Sudoku. Le fichier `resultatsSudoku.txt` contient un compte-rendu de ce que devrait retourner vos fonctions sur les différentes grilles fournies. Les fonctions demandées sont les suivantes :

1. Une fonction qui permet de savoir si une grille (même partiellement remplie) est bien un Sudoku ; c'est-à-dire qu'elle respecte les règles du Sudoku.
2. Une fonction qui, à partir d'un Sudoku partiellement rempli et des indices d'une case non remplie, va retourner la liste des valeurs possibles pour cette case.
3. Une fonction qui permet de savoir si une grille partiellement remplie est bloquée (c'est-à-dire qu'elle possède au moins une case non remplie qui ne peut pas être remplie).
4. Une fonction qui retourne la liste des cases non remplies qui n'ont qu'une seule valeur possible.

Le fichier `sudoku.py` contient une liste de fonctions utiles pour implémenter les fonctions demandées.