Langages de script
TP n° 3 : Listes, ensembles et tables de hachage

Les exercices optionnels sont marqués avec ($\star$).

Exercice 1 : Retour sur les listes

Dans les mutations de liste on peut aussi avoir des doubles itérations. Par exemple `[(a,b) for a in [1,2] for b in [3,4]]` produit `[(1,3),(1,4),(2,3),(2,4)]` ou `[x for y in [[1,2],[3,4],[5,6,7]] if len(y) <= 2 for x in y ]` donne `[1,2,3,4]`.

1. Écrire une fonction `couples(i,j)` qui génère une liste de tous les couples d'entiers $\{(x, y) \mid 0 \leqslant x \leqslant i \text{ et } 0 \leqslant y \leqslant j\}$.
2. Une matrice d'entiers est une liste de listes d'entiers. Écrire une fonction `transpose(m)` qui calcule la transposée de la matrice `m`. Par exemple, `transpose([[1,2,3],[4,5,6]])` renvoie `[[1,4],[2,5],[3,6]]`
3. Écrire une fonction `produit(l)` qui calcule le produit cartésien de toutes les listes contenues dans `l`. Par exemple `produit([[4,5],['cd'],[1,2,3]])` donne `[[4,'cd',1],[5,'cd',1],[4,'cd',2],[5,'cd',2],[4,'cd',3],[5,'cd',3]]`

Exercice 2 : Ensembles

1. Écrire une fonction `ens(w)` qui prend en argument une chaîne de caractères `w` et retourne l'ensemble de ses caractères. Par exemple : `ens("caracteres")` retourne `{'a','c','e','s','r','t'}` alors que `ens("")` retourne `set()`. Écrire une version en ajoutant un par un les caractères à un ensemble et une version de la forme `return { code }`. Est-ce que votre fonction marche aussi pour d'autres types de données ?
2. Écrire une fonction `afficher(s)` qui affiche tous les éléments de l'ensemble `s`. Quel est l'ordre dans lequel les éléments sont affichés ? Est-ce que votre fonction marche aussi pour d'autre type de données ?
3. Écrire une fonction `minimum(s)` qui renvoie le minimum d'un ensemble non vide d'entiers.
4. En utilisant les ensembles écrire une fonction `memeslistes(l1,l2)` qui renvoie `True`, si les listes `l1` et `l2` contiennent les mêmes éléments (dans n'importe quel ordre et possiblement dupliqués), `False` sinon.
   Par exemple `memeslistes([1,2,1,3,3],[3,2,2,1])` renvoie `True`.
5. Écrire un *script* python exécutable `comptercaracteres` qui prend en argument un fichier et affiche combien de caractères différents le fichier contient. Par exemple `comptercaracteres truc` affiche `7` si le fichier `truc` contient deux lignes de la forme « `machin` ».
6. Écrire une fonction `multiples(p,n)` qui renvoie l'ensemble de tous les multiples de l'entier p (sans p lui-même) qui sont inférieurs ou égale à l'entier n. Par exemple, `multiple(2,10)` renvoie `{8, 10, 4, 6}`

7. Écrire une fonction `premiers` qui renvoie un ensemble avec tous les nombres premiers inférieur ou égal à l'entier `n` donné en entrée. Utiliser la méthode du crible d'Ératosthène : On crée un ensemble s de tous les entiers de 2 jusqu'à n. On met p à 2. (A) Tant que $p < n$ et $p \notin s$, on incrémente p de 1 (donc p est premier). Ensuite on enlève tous les multiples de p de s, on incrémente p de 1 et on continue avec (A).

Exercice 3 : Table de hachage simple

Une façon d'implémenter les ensembles est d'utiliser des *tables de hachage*. Le but d'une table de hachage est entre autres de disposer d'une structure de donnée permettant de tester en général très rapidement (sans parcourir toute une liste par exemple), si un élément est contenu dans un ensemble. Dans cet exercice on propose d'implémenter une simple méthode de hachage permettant de représenter des ensembles d'entiers et de la comparer avec l'implémentation de Python des ensembles.

Ici, une table de hachage de taille n est juste un tableau (une liste en Python) de taille n contenant des entiers (ou `None`). Une *fonction de hachage* h est une fonction qui associe à une valeur d'un domaine d'entrée un entier entre 0 et $n-1$. Par exemple, si le domaine d'entrée sont les entiers on peut choisir tout simplement $h(k) = k \ mod \ n$. Une table de hachage peut être utilisée par exemple pour stocker un ensemble (contenant des entiers) de taille n au maximum :
Un ensemble vide correspond à un tableau (une liste) `t` où tous les éléments sont `None`. Ajouter un élément k à la table consiste à calculer $p = h(k)$. Ensuite, si le tableau `t` à la position p est vide, on stocke k à cette position. Si le tableau à la position p contient déjà k, on ne fait rien. Sinon, on a un *conflit*. On peut résoudre le conflit en remplaçant p avec $(p+1) \ mod \ n$, $(p+2) \ mod \ n$, etc. (cette méthode s'appelle *linéaire*), jusqu'à ce qu'on trouve une place libre dans le tableau. Une autre méthode (appelée *quadratique*) consiste à remplacer p successivement par $(p+1) \ mod \ n$, $(p+4) \ mod \ n$, $(p+9) \ mod \ n$, etc.
Pour tester si un élément est dans l'ensemble, on essaie de le retrouver comme si on voulait l'ajouter.
Évidemment, si on essaie d'ajouter un élément à une table pleine, la méthode d'ajout ne s'arrête pas.

1. Écrire une fonction `inittable(n)` qui renvoie une table de hachage (liste) de taille `n` contenant des éléments `None`.
2. Écrire une fonction `ajoutertable(x,n,t)` qui ajoute l'élément `x` à la table `t` qui est de taille `n` avec la méthode linéaire. Cette fonction ne renvoie rien !
3. Écrire une fonction `danstable(x,n,t)` qui renvoie `True` si l'élément `x` est dans la table de taille `n`, sinon `False`
4. Écrire une fonction `testhachage(n,l)` qui crée une table de hachage de taille `n` et ensuite effectue les opérations suivantes sur une liste d'entiers `l` :
   - ajouter tous les éléments de `l` à la table
   - tester, si tous les entiers de la liste `l` sont dans la table
5. Écrire une fonction `testset(l)` qui crée un ensemble vide et ensuite effectue les opérations suivantes sur une liste d'entiers `l`
   - ajouter tous les éléments de `l` à l'ensemble

- tester, si tous les entiers de la liste `l` sont dans l'ensemble

6. Écrire une fonction `testmethodes(k,m,n)` qui crée une liste de k entiers choisi au hasard entre 0 et m et qui ensuite calcule et affiche le temps d'exécution de la fonction `testhachage(n,l)` et indépendamment de la fonction `testset(l)` de sorte qu'on puisse comparer les deux. Ensuite faire différents test pour comparer les deux méthodes.

   Indication : Dans le module `random` il y a une fonction `randint` qui produit des entiers au hasard. Dans le module `time` il y a une fonction `process_time` qui donne le temps d'exécution du programme du début jusqu'au point actuel.

7. Ajouter dans les tests la méthode quadratique.

8. ($\star$) Comment pourrait-on enlever un élément de la table ?
   Écrire une fonction `enlever(x,n,t)` qui ôte l'élément `x` à la table `t` de taille `n` avec la méthode linéaire. Cette fonction ne renvoie rien !

9. ($\star$) Modifier vos fonctions pour représenter des ensembles de chaînes de caractères (ou autre types de données) ?