TP N°7: Les ensembles

Un *ensemble* en Python est une collection modifiable d'objets *sans répétition* et *sans ordre* (donc sans numérotage). Il appartient au au type `set`. Il est créé par la séquence de ses éléments (valeurs), encadrées par { et }, ou en utilisant le constructeur `set` appliqué sur un itérable.

- Les valeurs des ensembles peuvent être de types différents.
- Les ensembles ne sont pas *indexées*.
- Les ensembles ne sont *mutables*, on peut ajouter et supprimer des éléments, mais on ne peut pas écrie s[i] = valeur, puisque un ensemble n'est pas une séquence.
- Les emplois basiques sont le test d'appartenance et l'élimination des entrée dupliquées.

Création

```
s1 = set() # ensemble vide
s2 = {"a" , 2.5 , 5} # créer un ensemble avec des valeurs de types hétérogènes
s3 = set([1,2,3]) # créer un ensemble à partir d'un itérable (ensemble)
s4 = set(range(0,1000,200)) # créer un ensemble à partir d'un itérable (intervalle)
s5 = {x*x for x in range(20) if x % 3 == 0} # créer un ensemble en compréhension
s6 = set('aeiouy') # {'o', 'i', 'e', 'a', 'y', 'u'}
```

Remarques :

- L'ensemble vide se note `set()` et non { } qui crée un *dictionnaire*

Exercice 1

Écrire un programme qui, à partir d'une chaîne de caractères `s`, crée un ensemble `lettres` formée des lettres utilisées dans la chaîne `s`.

Fonctions et méthodes associées aux ensembles

```
s = {2, 0, 5, 4} # ensemble initial
s.add(8) # Permet d'ajouter un élément à un ensemble.
s.remove(5) # Supprimer un élément donné d'un ensemble
s.pop() # Permet d'obtenir un élément (lequel ?) d'un ensemble et de le supprimer en même temps.
s.clear() # Vider un ensemble
len(s) # renvoie le nombre d'éléments (cardinal) d'un ensemble
2 in s # Tester avec in et not in si un élément appartient ou non à un ensemble.
s == {1,2} # Comparer le contenu de deux ensembles
max(s), min(s), sum(s) # Retourner le maximum, le minimum et la somme des éléments d'un ensemble
```

Exercice 2

1. Soit `L` une liste. Supprimez, à l'aide d'une seule instruction simple, tous les doublons de `L`. Calculer le nombre d'éléments distincts de `L`.
2. Reprendre la question précédente où cette fois `L` est une liste de chaînes de caractères ASCII (pas de lettres accentuées). On considère qu'une chaîne *« doublonne»* avec une autre si elle sont identiques sans tenir compte de la casse (par exemple 'Everest' et 'eVereSt' sont identiques). Supprimez tous les doublons de `L`.

Opérations ensembensembles

Les diverses opérations ensembensembles, définies dans le cours de mathématiques, sont réalisables avec des `set`. Supposons, par exemple, que l'on ait défini les deux ensembles `S1` et `S2` suivants :

```
>>> S1 = set([1, 2, 3])
>>> S2 = set([0, 1])
```

Union `S1.union(S2)` ou bien `S1 | S2`

Intersection `S1.intersection(S2)` ou bien `S1 & S2`

Différence `S1.difference(S2)` ou bien `S1 - S2`

Différence symétrique `S1.symmetric_difference(S2)` ou bien `S1 ^ S2`

Inclusion `S1.issubset(S2)` ou bien `S2.issuperset(S1)` : tester si S1 est inclus dans S2

Exercice 3

1. Définir deux ensembles (sets), `X = {'a', 'b', 'c', 'd'}` et `Y = {'s', 'b', 'd'}`
2. Affichez les résultats suivants:
   - Les ensembles initiaux ;
   - Le test d'appartenance de l'élément $'c'$ à X ;
   - Le test d'appartenance de l'élément $'a'$ à Y ;
   - Les ensembles $X - Y$ et $Y - X$;
   - L'ensemble $X \cup Y$ (union) ;
   - L'ensemble $X \cap Y$ (intersection).

Exercice 4

1. Écrire un programme python qui permet d'utiliser un ensemble pour vérifier si un mot contient des voyelles.
2. Soient `E` et `F` deux ensembles. Écrire un programme python qui calcul le produit cartésien de `E` et `F`, noté $E \times F$, formé de l'ensemble des couples (x,y) où $x \in E$ et $y \in F$.

Itération sur les ensembles

```
for x in  S: # On peut itérer directement sur une ensemble  : x représente un élément de S
    print(x)

for i in range(len(S)): # i varie de 0 à len(S) - 1
    print(S[i]) # TypeError: 'set' object does not support indexing
```

Remarque : Un ensemble n'est pas une séquence. On peut pas accéder à ses éléments par indices.

Exercice 5

1. Écrire un programme python qui permet de saisir un ensemble de n nombres et affiche le produit de ses éléments.
2. Écrire un programme python qui permet d'énumérer un ensemble de n nombres et crée la liste de sous ensembles énumérés.
3. Calculer le nombre de listes à k éléments dans d'un ensemble à n éléments.
   Rappelons que les listes sont ordonnées : `[1,2,3]` $\neq$ `[1,3,2]`
4. Écrire un programme python qui à partir d'une liste d'entiers `L`, crée une liste de deux ensembles, le premier sous-ensemble contenant les éléments pairs de L et le second ses éléments impairs.
   Exemple : `[1, 2, 4, 7, 3, 2, 1]` $\rightarrow$ `[ {4, 2}, {7, 3, 1} ]`