Devoir à la maison n° 1

Consignes

Votre DM sera rendu sous forme d'un seul fichier Python impérativement nommé de la façon suivante :

`<VOTRE_NOM>_<Votre_prénom>_TNSI_DM1.py`

Les parties entre <> étant à remplacer. Exemple : `HOLZER_Marie_TNSI_DM1.py`

Vous me l'enverrez via la messagerie de l'ENT.

Vous devez suivre les conventions rappelées dans le document Mémo Python, notamment concernant les commentaires `# IMPORTS` / `# FONCTIONS` / `# SCRIPT`, et les conventions de nommage des variables et fonctions.

Vous êtes priés de nommer vos fonctions comme demandé dans l'énoncé et de bien nommer vos variables pour rendre votre programme clair. N'hésitez pas à faire des `print` pertinents pour afficher vos résultats de manière lisible.

> **Votre programme ne doit pas générer d'erreurs à l'exécution !**
> S'il y a une erreur à l'exécution, vous perdez la moitié des points.

Exercice 1

Dans cet exercice aucune « pythonnade » (*slices*, méthodes de la class `str`) n'est autorisée.

1. Écrire une fonction `all_but_last(s)` qui prend en argument une chaîne de caractères et renvoie la même chaîne privée de son dernier caractère.
   Exemple : `assert all_but_last("abcde")=="abcd"`
2. Écrire une fonction `no_whitespace(s)` qui prend en argument une chaîne de caractères et renvoie la même chaîne mais où tous les espaces ont été supprimées.
   Exemple : `assert no_whitespace("a b c d e")=="abcde"`
3. Écrire une fonction `reverse(s)` qui prend en argument une chaîne de caractères et renvoie la même chaîne écrite à l'envers. Exemple : `assert reverse("abcde")=="edcba"`

Exercice 2

Le langage Python fournit une méthode `split` pour les chaînes de caractères : elle prend en argument un caractère `sep`, et renvoie la liste des « mots » contenus dans la chaîne, en utilisant `sep` comme délimiteur. On donne des exemples ci-dessous.

```
>>> s = "Que j'aime à faire apprendre ce nombre utile aux sages"
>>> s.split(" ")
['Que', "j'aime", 'à', 'faire', 'apprendre', 'ce', 'nombre', 'utile', 'aux', 'sages']
```

```
>>> s.split("e")
['Qu', " j'aim", ' à fair', ' appr', 'ndr', ' c', ' nombr', ' util', ' aux sag', 's']
```

1. Écrire une fonction `explose(s, sep)` qui renvoie la même liste que l'appel `s.split(sep)`.
   Les assertions ci-dessous devront être vérifiées.

   ```
   assert explose("abc de f", " ")==["abc", "de", "f"]
   assert explose("a bc def   g h ", " ")==["a", "bc", "def", "g", "h"]
   ```

2. Modifier cette fonction pour autoriser l'utilisation de plusieurs délimiteurs. Exemple :

   ```
   s = "Hey, you - what are you doing here!?"
   assert explose(s, " ,-!?")==["Hey", "you", "what", "are", "you", "doing", "here"]
   ```

Exercice 3

Écrire une fonction récursive `appartient(valeur, lst, indice)` prenant en paramètres une valeur `valeur`, un tableau (une liste en Python) `lst` et un entier `indice` renvoyant `True` si `valeur` apparaît dans `lst` entre l'indice `indice` (inclus) et `len(ldt)` (exclu), et `False` sinon.

On supposera que `indice` est toujours compris entre `0` et `len(lst)`.

Voici un jeu de test qui ne doit donc pas générer d'erreurs dans votre code :

```
lst = [5, 12, 49, 7]

assert not appartient(5, lst, 1)
assert appartient(5, lst, 0)
assert appartient(7, lst, 3)
```