301 Algo-Prog TD3
Sous-programme, liste, chaîne de caractères

1 Fonctions simples sur les listes

Pour tous ces exercices, on suppose que les éléments de liste sont des entiers.

1. Écrivez la fonction *create_liste*() qui permet de créer une liste en demandant à l'utilisateur de saisir des éléments. On supposera que l'utilisateur entre une valeur "spéciale" ("stop", par exemple) pour terminer la création. Cette fonction retourne la liste créée.

2. Écrivez la fonction $take(n, liste\_entier)$ qui retourne une liste contenant les n premiers éléments de liste_entier ($n <= len(list\_entier)$).

3. Écrivez la fonction $drop(n, liste\_entier)$ qui retourne une liste contenant liste_entier privée de ses n premiers éléments ($n <= len(list\_entier)$).

4. Écrivez la fonction $somme(liste)$ qui retourne la somme des éléments de liste (cette liste peut être vide).

5. Écrivez la fonction $pairs\_impairs(liste\_entier)$ qui retourne deux listes : la liste des valeurs paires et la liste des valeurs impaires de liste_entier.

6. Écrivez la fonction $split\_at(n, liste)$ qui divise liste en deux parties en la coupant à l'indice n (attention aux cas particuliers). Cette fonction retourne les deux parties.

7. Écrivez la fonction $to\_str(liste)$ qui retourne une chaîne de caractères contenant les éléments de liste séparés par une virgule.

Vous pouvez tester vos fonctions en écrivant votre propre programme (prenez le cas où la liste est vide) ou avec le programme principal suivant :

```
if __name__ == '__main__':
    assert take(5, list(range(1, 11))) == [1, 2, 3, 4, 5]
    assert drop(5, list(range(1, 11))) == [6, 7, 8, 9, 10]
    assert somme([]) == 0
    assert somme([1, 2, 3]) == 6
    assert pairs_impairs(list(range(1, 5))) == ([2, 4], [1, 3])
    assert split_at(3,list(range(1, 5))) == ([1, 2, 3], [4])
    assert split_at(0, list(range(1, 5))) == ([], [1, 2, 3, 4])
    assert to_str([1,2,3]) == "1,2,3"
```

2 Compteur de mots

1. Écrivez une fonction qui prend en paramètre une chaîne de caractères et qui retourne le nombre de mots qu'elle contient. (On considère seulement les chaînes de caractères sans ponctuation, dont les mots sont séparés uniquement par des espaces. Par exemple, "Bonjour, tu vas bien" contient 4 mots.)

2. Écrivez un programme principal qui demande à l'utilisateur de saisir une chaîne et qui affiche ensuite le nombre de mots contenus dans la chaîne.

3. Et si on prend en compte les caractères de ponctuation ?