Exercice 1.1

Ecrire une fonction, la plus courte possible, qui retourne l'alphabet sous forme de dictionnaire, où la clé est la lettre, et la valeur est sa position dans l'alphabet.

Par exemple :

```
{'A': 1, 'B': 2, 'C': 3, 'D': 4, ...}
```

Exercice 1.2

Créer les deux listes suivantes :

```
liste1 = [26, 7, 14, 56, 32, 71, 2, 6, 7]
```

```
liste2 = [7, 6, 2, 36, 47, 26, 14, 45, 87]
```

Implémenter plusieurs méthodes différentes afin de déterminer l'intersection de cette liste (les valeurs présentes dans les deux listes). Créer une nouvelle fonction par méthode.

Afin d'éviter les doublons, utiliser des `set`. Le résultat doit être le suivant :

```
{2, 6, 7, 14, 26}
```

Exercice 1.3

Dans la fonction suivante :

```
def fonction_liste_defaut(nombre, liste=[]):
    liste.append(nombre)
    return liste
```

La référence de la `liste` utilisée par défaut est toujours la même lorsque l'on appelle la fonction avec un seul paramètre.

Modifier la fonction pour faire en sorte qu'une nouvelle liste soit créée à chaque appel avec un seul paramètre (le comportement ne doit pas être modifié si on passe une liste en deuxième paramètre).

Exercice 1.4

Ecrire une version la plus courte possible de cette fonction, qui retourne une liste de sommes des ranges de chaque nombre pair, jusqu'à un maximum donné :

```
def liste_de_sommes1(maxi):
    nombres = range(maxi + 1)
    sommes = []
    for nombre in nombres:
        if nombre % 2 == 0:
            somme = 0
            for i in range(nombre + 1):
                somme += i
            sommes.append(somme)
    return sommes
```

Par exemple pour `liste_de_sommes1(10)`, la fonction retourne :

```
[0, 3, 10, 21, 36, 55]
```

Correspondant à `[0, 1 + 2, 1 + 2 + 3 + 4, 1 + 2 + 3 + 4 + 5 + 6, ...]`.

Ecrire plusieurs fonctions si plusieurs méthodes de simplification sont trouvées.

Exercice 1.5

Créer une fonction générateur qui reproduit exactement le même comportement que `range()` (avec 1, 2 ou 3 paramètres). Il ne faut pas utiliser de `list`.

Exercice 1.6

Le fichier *gros_texte.txt* contient une centaine de paragraphes.

Créer un générateur qui prend en paramètre un nombre de lettres et qui renvoie les mots dont la taille correspond à ce nombre de lettres.

Implémenter une fonction qui fait la même chose mais avec une liste (donc pas de `yield`).

Exercice 1.7

Créer un décorateur qui permet de contrôler le type (et par conséquent le nombre) des valeurs passées en paramètre d'une fonction. Faire des essais avec 2 ou 3 fonctions.