Tests

Exercice 1
On souhaite tester une fonction qui prend comme argument un nombre entier naturel et qui doit renvoyer `True` si le paramètre est un nombre premier, et `False` sinon.
La fonction à tester est la suivante :

```
def premier(n):
    if n == 1:
        return False
    if n == 2:
        return True
    for d in range(2,n):
        if n%d == 0:
            return False
        return True
```

1. Recopier le code de la fonction `premier` dans un fichier.
2. Dans le même fichier, définir une fonction `test1` qui parcourt la liste de **nombres premiers** `[2,3,5,7,11,13,17,19]` et qui affiche chaque valeur pour laquelle la fonction `premier` ne retourne pas `True` (autrement dit pour laquelle elle retourne `False`). Autrement dit, la fonction doit afficher les valeurs de nombres premiers pour lesquels le résultat n'est pas celui attendu, et uniquement celles-ci. Ainsi, si aucune valeur n'est affichée, le test est réussi. Effectuer alors le test en exécutant la fonction `test1`.

   Remarque Puisque la fonction `premier` retourne un booléen, au lieu d'écrire la condition `premier(n) == False`, on peut écrire la condition `not premier(n)`.

3. Écrire de même une fonction `test2` qui vérifie, cette fois sur la liste des **nombres non premiers** de 0 à 15 inclus que la fonction `premier` retourne bien `False`.
   Là aussi, seuls les nombres pour lesquels le résultat n'est pas celui attendu doivent être affichés et le test ne réussit que si aucune valeur n'est affichée.
   Effectuer alors le test en exécutant la fonction `test2` et observer les erreurs trouvées.
4. Trouver les raisons de ces erreurs et redéfinir la fonction `premier` pour corriger les défauts du code de la fonction.
   Effectuer à nouveau les deux tests pour vérifier que la correction est suffisante et n'a pas créé de nouvelles erreurs.
   Corriger encore et répéter le test en cas de nécessité jusqu'à ce que la fonction soit valide.

Exercice 2
Définir une fonction `test_sorted` qui teste le fonctionnement de la fonction `sorted` de Python, autrement dit qui vérifie que la liste retournée par cette fonction est bien triée, quelle que soit la liste d'entiers donnée en paramètre.
Pour cela, utiliser plusieurs jeux de données sous forme de (grandes) listes de nombres aléatoires en utilisant le module `random` (et par exemple sa fonction `randint`).
Il sera nécessaire d'utiliser une fonction qui vérifie qu'une liste donnée en argument est bien triée. Une telle fonction a été vue récemment, son code pourra donc être recopié.

Exercice 3

Définir une fonction `test_euclide` qui teste le fonctionnement de la fonction suivante qui retourne le reste `r` et le quotient `q` de la division euclidienne de `n` par `d` :

```
def euclide(n,d):
    '''n et d sont de type int
    retourne q,r de type int tels que n == d*q+r et 0<=r<d'''
    r = n
    q = 0
    while r>d:
        r = r-d
        q = q+1
    return q,r
```

Le test devra se faire sur de nombreuses valeurs de `n` et de `d`, prises au choix soit dans des `range`, soit dans des listes, soit aléatoirement.

La fonction de test pourra bien entendu faire appel aux opérations de base en Python, à savoir `n//d` et `n%d`.

Faire en sorte que la fonction de test affiche pour quels jeux de valeurs il y a une erreur (afficher les valeurs de `n`, `d`, celles retournées par `euclide(n,d)` ainsi que les valeurs attendues).

Redéfinir et modifier ensuite la fonction `euclide` et tester à nouveau jusqu'à l'avoir corrigée complètement.