Pour chacun des exercices, il est demandé de tester votre implémentation, en considérant un maximum de cas possibles. On vous demandera de calculer les temps d'exécution (temps CPU en secondes) de vos programmes, pour cela vous pouvez utiliser le module `time` qui fournit diverses fonctions liées au temps. La méthode `time.time()` du module est utilisée pour obtenir le temps en secondes écoulées. La gestion des secondes intercalaires (leap seconds) dépend de la plateforme utilisée. Vous pouvez donc faire appel à la méthode comme illustré ci-dessous :

```
import time
def sum1(n:int)->int:
    '''
    Retourne la somme des entiers de 0 à n.
    '''
    total = 0
    for i in range(n+1):
        total += i
    return total
start = time.time()
sum1(1000)
end = time.time()
print("Le temps d'exécution est : ", (end-start)*10**3, "ms")
```

Exercice 1 : Écrire une fonction qui prend un entier positif n et qui retourne la somme des chiffres qui le composent. Tester cette fonction et calculer son temps d'execution.

Exercice 2 : Recherche du minimum et du maximum.

1. Écrire deux fonctions `max1` et `min1` qui retournent respectivement l'élément maximal et minimal d'une liste d'entiers. Tester les deux fonctions.
2. Combien de comparaisons les fonctions `max1` et `min1` effectuent-elles en fonction de la taille de la liste en entrée ?
3. En utilisant les fonctions précédentes, écrire une fonction `minmax` qui retourne le couple correspondant au plus petit et au plus grand élément de la liste. Comptez le nombre de comparaisons effectuées par la fonction.
4. Améliorer la fonction précédente pour qu'elle ne fasse pas plus de 150 comparaisons sur des listes contenant 100 éléments.
5. Vérifier le nombre de comparaisons effectuées.
6. Comparer les temps d'exécution en faisant varier la taille de la liste.

Exercice 3 : On s'intéresse au calcul de la division euclidienne entre deux entiers, en utilisant seulement les opérateurs d'addition, de soustraction, de multiplication et de comparaison, entre des nombres entiers. `https://fr.wikipedia.org/wiki/Division_euclidienne#Algorithmes_de_calcul`

1. Écrire une fonction qui implémente la méthode naïve pour la division euclidienne.
2. Écrire une fonction qui implémente la méthode décimale pour la division euclidienne.
3. Écrire une fonction qui implémente la méthode binaire pour la division euclidienne.
4. Comparer les temps d'exécution des differentes fonctions.

Exercice 4 : Un serveur web populaire prend en charge une fonction appelée `no2slash`, dont le but est de réduire plusieurs caractères `'/'` consécutifs en un seul.

Par exemple, la chaîne `'/d1///d2////d3/test.html'` devient `'/d1/d2/d3/test.html'`. L'algorithme `no2slash` consiste à rechercher de manière répétée un caractère `/` et à copier le reste de la chaîne.

```
def no2slash(name):
    l = list(name)
    x = 1
    while x < len(l):
        if (l[x-1] == '/') and (l[x] == '/'):
            for y in range(x+1, len(l)):
                l[y-1] = l[y]
            l = l[:-1]
        else:
            x += 1
    return ''.join(l)
```

Malheureusement, le temps d'exécution de ce code est quadratique par rapport au nombre de caractères `'/'` dans l'entrée. En envoyant plusieurs requêtes simultanées contenant un grand nombre de caractères `'/'`, un pirate peut submerger le serveur créant ainsi une attaque par déni de service (DoS attack).

Écrire une version de la fonction `no2slash()` qui s'exécute en temps linéaire et ne permet pas ce type d'attaque.

Exercice 5 : Écrire une fonction qui prend en paramètres une liste de joueurs numérotés de `1` à `n` et un entier `m`, et qui implémente un jeu où les joueurs forment un cercle, on compte de `1` à `m` de manière répétitive, le joueur à la position `m` est éliminé à chaque tour, et le jeu se termine lorsqu'il ne reste qu'un seul joueur, qui sera le gagnant. ce dernier est retourné par la fonction. Exemple d'exécution :

- Entrée : la liste des joueurs et l'entier `[1, 2, 3, 4, 5], 3`
- Processus :
  - Le joueur 3 est éliminé : `[1, 2, 4, 5]`.
  - Ensuite, le joueur 1 est éliminé : `[2, 4, 5]`.
  - Puis, le joueur 5 est éliminé : `[2, 4]`.
  - Enfin, le joueur 4 est éliminé : `[2]`.
- Sortie : Le joueur 2 est le gagnant.