À faire individuellement ou par petits groupes de deux ou trois.

Exercice 1. Fonctions – Retours multiples

En Python, il est possible de retourner plusieurs valeurs avec une seule instruction **`return`**. Ces multiples valeurs doivent simplement être séparées par des virgules. Par exemple, la fonction suivante calcule $a+b$ et $a-b$ et retourne ces deux valeurs.

```python
def add_sub(a: int, b: int) -> int:
    return a+b, a-b
```

Ces deux valeurs peuvent ensuite être stockées dans deux variables distinctes, en une seule instruction. Il faut, ici aussi, séparer les deux variables par des virgules.

```python
add, sub = add_sub(2, 3)
print(add)  # Affiche 5
print(sub)  # Affiche -1
```

a) Définissez une fonction **`min_max(l: List[int]) -> int`** prend une liste en entrée et retourne le minimum et le maximum de cette liste.

b) Complétez le programme suivant en remplissant le corps de la fonction **`create_list()`** qui crée une liste aléatoire de taille 10. Vous pourrez ensuite tester la fonction **`min_max()`** comme indiqué dans la suite du programme.

```python
from typing import List
from random import randint

def create_list() -> List[int]:
    ...

l = create_list()
print(l)
low, high = min_max(l)
print(low, high)
```

Exercice 2. Divisibilité par 3

Voici une façon d'interprêter la divisibilité par 3:

Soit un entier positif $n = d_k...d_0$, où $d_0...d_k$ sont les chiffres qui le composent, et soit $S = \sum_{i=0}^{k} d_i$.

Si $S < 10$, alors n est divisible par 3 si et seulement si $S \in \{3, 6, 9\}$.
Sinon, alors on pose $n = S$ et on recalcule S pour cette nouvelle valeur de n.

a) Définissez une fonction **`sum_of_digits(n: int) -> int`** qui prend en entrée un entier **`n`** et qui retourne la somme de ses chiffres.
Astuce : Vous pouvez soit utiliser des divisions successives par 10, soit convertir l'entier en une chaîne de caractères pour récupérer les chiffres «caractère après caractère».

b) Définissez une fonction **`disible_by_3(n: int) -> bool`** qui prend en entrée un entier **`n`** et qui détermine si cet entier est divisible ou non par 3.
Astuce : Il se peut que la fonction **`sum_of_digits()`** doive être appelée successivement.
Rappel : L'entier **`n`** est divisible par 3 si la valeur retournée par la fonction **`sum_of_digits()`** est égale à 3, 6, ou 9.

Vous pourrez tester vos fonctions avec le programme suivant :

```python
for i in range(50):
    if divisible_by_3(i):
        print(f"{i} est divisible par 3")
    else:
        print(f"{i} n'est pas divisible par 3")
```

Exercice 3. Liste de listes

Partie papier-crayon

Une liste est donc une structure de données qui peut contenir plusieurs valeurs. En poussant cette définition un peu plus loin, on peut en déduire qu'une liste peut contenir d'autres listes.

Observez le programme suivant:

```
matrix: List[List[int]] = []
rows: int = 2
columns: int = 3
for i in range(rows):
    row: List[int] = []
    for j in range(columns):
        row.append(i+j)
    matrix.append(row)
print(matrix)
```

Ce programme contient des *boucles imbriquées*. Il peut être difficile de suivre mentalement l'exécution d'un tel programme. Une fois encore, le *code tracing* et un tableau de suivi des variables peut être utile.

a) Quelles sont les valeurs prises successivement par les variables **`i`** et **`j`** ? Cela permettra de déterminer l'état «initial» puis l'état «final».

b) Pour ce *code tracing*, nous allons suivre les valeurs des variables: **`matrix`**, **`row`**, **`i`**, et **`j`**.

	`matrix`	**`row`**
i = 0, j = 0	[]	[0]
i = 0, j = 1	[]	[0, 1]
...		
...		
...		
i = ?, j = ?	?	?

Table 1: Exemple de tableau de suivi de variables

Partie sur ordinateur

a) Puisque la fonction **`print()`** permet d'afficher le contenu d'une liste, utilisez-la pour vérifier votre travail dans la partie papier-crayon.

b) Complétez la fonction **`pretty_print()`** qui prend en entrée une liste de listes (ou une matrice) et qui l'affiche de façon plus lisible comme proposé ci-après.
Astuce 1 : Nous pouvons afficher «en dur» les crochets ouvrants et fermants.
Astuce 2 : Pour chaque ligne à afficher, nous pouvons créer une variable de type **`str`** qui contient les valeurs à afficher séparées par des espaces.

```
def pretty_print(m: List[List[int]]) -> None:
    ...

matrix: List[List[int]] = [[1, 2, 3], [4, 5, 6]]
pretty_print(matrix)
'''
Affiche le résultat suivant:
[
  1 2 3
  4 5 6
]
'''
```