Exercices

Exercice 1 * On souhaite construire une fonction qui renvoie la note la plus basse dans un tableau. Les notes sont comprises entre 0 et 20.

1. Écrire la fonction **note_mini(tab: list) → int** qui renvoie la valeur minimale du tableau **tab**.
2. Construire par compréhension un tableau de 10 entiers aléatoires entre 0 et 20.
3. Tester alors la fonction.

Exercice 2 * Écrire la fonction **extrema(tab: list) → tuple** qui renvoie les valeurs minimale et maximale du tableau **tab**, sous forme de tuple. On considérera qu'on travaille encore sur des tableaux d'entiers compris entre 0 et 20.

Exercice 3 * Soit **maxi_position(tab: list) → tuple**, la fonction qui renvoie un tuple contenant:

- la valeur maximale du tableau **tab**,
- l'indice de la **première** apparition de cette valeur.

1. **Avant de construire la fonction**, écrire 3 assertions qu'elle doit passer. On choisira des tableaux de tests judicieux.
2. Écrire la fonction.

Exercice 4 * Écrire la fonction **maxi_position_dernier(tab: list) → tuple** qui renvoie la valeur maximale du tableau **tab** et l'indice de la **dernière** apparition de cette valeur, sous forme de tuple.

Exercice 5 ** Soit la fonction **maxi_nb(tab: list) → int** qui renvoie le nombre d'apparitions de la valeur maximale du tableau **tab**.

1. **Avant de construire la fonction**, écrire 3 assertions qu'elle doit passer.
2. Écrire l'algorithme de la fonction.
3. Écrire la fonction.

Exercice 6 *** On peut assimiler une chaîne de caractère à un tuple, c'est à dire une séquence ordonnée et non modifiable. Ainsi on peut repérer un caractère par son indice.

```
mot = "bonjour"
print(mot[3]) # affiche 'j'
```

1. Écrire la fonction **est_voyelle(lettre: str) → bool** qui renvoie **True** si **lettre** est une voyelle, **False** sinon.
2. Écrire la fonction **compter_voyelles(mot: str) → dict** qui renvoie un dictionnaire du décompte des voyelles. Dans la fonction, on utilisera un dictionnaire **voyelles** qui, au départ, associe chaque voyelle à l'entier 0.

```
voyelles = {"a": 0, "e": 0, "i": 0, "o": 0, "u": 0, "y": 0}
```

3. Dans le programme principal, écrire la boucle qui affiche dans la console chaque voyelle suivie de son nombre d'occurrences. Par exemple, on pourra afficher:
   - a: 3
   - e: 5
   - …
4. Écrire la fonction **max_voyelles(voyelles: dict) → list** qui renvoie les voyelles qui comptent le plus d'occurrences dans le dictionnaire renvoyé par la fonction précédente. Par exemple, pour le mot "orangeade" la fonction **compter_voyelles** renvoie le dictionnaire suivant:

```
{'a': 2, 'e': 2, 'i': 0, 'o': 1, 'u': 0, 'y': 0}
```

Ainsi le tableau renvoyé par la fonction **max_voyelles** sera ['a', 'e'].

Exercice 7 ***

1. Construire par compréhension un tableau contenant trois tableaux. Chacun des tableaux contiendra cinq nombres aléatoires entre 0 et 100.
2. Écrire la fonction **somme_ligne(tab: list, lig: int) → int** qui calcule et renvoie la somme des entiers de la ligne passée en paramètre.
3. Écrire la fonction **maximum(tab: list) → int** qui renvoie l'entier le plus grand du tableau.

Exercice 8 ***

1. Construire par compréhension le tableau suivant:

```
tab = [[0, 1, 2], [3, 4, 5], [6, 7, 8]]
```

2. Écrire le programme qui parcourt le tableau et affiche:

```
0 1 2 3 4 5 6 7 8
```

Exercice 9 ***

1. Construire par compréhension un tableau carré, dont le nombre de lignes (et donc de colonnes) est compris en 3 et 6. Ce tableau contiendra des entiers compris entre 0 et 100.
2. Écrire la fonction **recuperer_diagonale(tab: list) → list** qui renvoie les entiers de la diagonale descendante (du haut gauche vers le bas droite).