Exercices parcours séquentiel Correction

Christophe Viroulaud

Première - NSI

Algo 02

1. Exercice 1
2. Exercice 2
3. Exercice 3
4. Exercice 4
5. Exercice 5
6. Exercice 6
7. Exercice 7
8. Exercice 8
9. Exercice 9

Exercice 1

```python
def note_mini(tab: list) -> int:
    # les valeurs sont entre 0 et 20
    mini_prov = 21
    for val in tab:
        if val < mini_prov:
            mini_prov = val
    return mini_prov
```

Code 1 – Parcours par valeur

```python
def note_mini(tab: list) -> int:
    # les valeurs sont entre 0 et 20
    mini_prov = 21
    for i in range(len(tab)):
        if tab[i] < mini_prov:
            mini_prov = tab[i]
    return mini_prov
```

Code 2 – Parcours par indice

```
entiers = [randint(0, 20) for i in range(10)]
print(entiers)
print(note_mini(entiers))
```

Code 3 – Programme principal

```
[14, 3, 15, 9, 2, 5, 16, 3, 2, 2]
2
```

Code 4 – Une sortie possible

Exercice 2

```python
def extrema(tab: list) -> tuple:
    """
    Renvoie le mini et le maxi de tab

    Args:
        tab (list):
    Returns:
        tuple: (mini, maxi)
    """
    mini = 20
    maxi = 0
    for val in tab:
        if val < mini:
            mini = val
        if val > maxi:
            maxi = val
    return (mini, maxi)
```

```
entiers = [randint(0, 20) for i in range(10)]
print(entiers)
print(extrema(entiers))
```

Code 5 – Programme principal

```
[14, 3, 15, 9, 2, 5, 16, 3, 2, 2]
(2, 16)
```

Code 6 – Une sortie possible

```
assert extrema( [8, 2, 19, 14] ) == (2, 19)
assert extrema( [10, 10, 10, 10] ) == (10, 10)
```

Code 7 – Assertions possibles pour tester la fonction

Exercice 3

```
assert maxi_position( [3, 4, 5] ) == (2, 5)
assert maxi_position( [13, 42, 5] ) == (1, 42)
assert maxi_position( [3, 3, 3, 3] ) == (0, 3)
assert maxi_position( [] ) == (0, 0)
```

Remarque

Il n'y a pas de consigne spécifique dans le cas où le tableau est vide.

```python
def maxi_position(tab: list) -> tuple:
    """
    renvoie le max et sa première position dans le
   tableau
    """
    # On peut affecter plusieurs variables sur 1 ligne
    indice, val_max = 0, 0
    for i in range(len(tab)):
        if tab[i] > val_max:
            val_max = tab[i]
            indice = i
    return (indice, val_max)
```

Exercice 4

```python
def maxi_position_dernier(tab: list) -> tuple:
    """
    renvoie le max et sa dernière position dans le tableau
    """
    # On peut affecter plusieurs variables sur 1 ligne
    indice, val_max = 0, 0
    for i in range(len(tab)):
        # il suffit juste de modifier la comparaison
        if tab[i] >= val_max:
            val_max = tab[i]
            indice = i
    return (indice, val_max)
```

Exercice 5

```python
assert maxi_nb( [3, 4, 5] ) == 1
assert maxi_nb( [13, 13, 42, 5, 42, 42] ) == 3
assert maxi_nb( [3, 3, 3, 3] ) == 4
assert maxi_nb( [] ) == 0
```

- Initialiser le maximum à 0
- Initialiser le nombre d'occurrences à 0
- Pour chaque entier du tableau :
  - Si l'entier est égal au maximum :
    - incrémenter le nombre d'occurrences
  - Sinon si l'entier est supérieur au maximum :
    - mettre à jour le maximum
    - incrémenter le nombre d'occurrences

```python
def maxi_nb(tab: list) -> int:
    """
    renvoie le nombre d'occurrences du
  maximum
    """
    nb, val_max = 0, 0
    for val in tab:
        if val == val_max:
            nb = nb + 1
        elif val > val_max:
            # réinitialisation du max
            val_max = val
            nb = 1
    return nb
```

```
maxi_nb ([13, 13, 42, 5, 42, 42])
```

Initialisation :

- `nb = 0`
- `val_max = 0`

itération 1 :

- `val = 13`
- `nb = 1`
- `val_max = 13`

itération 2 :

- `val = 13`
- `nb = 2`
- `val_max = 13`

itération 3 :

- `val = 42`
- `nb = 1`
- `val_max = 42`

itération 4 :

- `val = 5`
- `nb = 1`
- `val_max = 42`

itération 5 :

- `val = 42`
- `nb = 2`
- `val_max = 42`

itération 6 :

- `val = 42`
- `nb = 3`
- `val_max = 42`

Exercice 6

```python
def est_voyelle(lettre: str) -> bool:
    """
    vérifie si lettre est une voyelle
    """
    voyelles = ("a", "e", "i", "o", "u", "y")
    for v in voyelles:
        if lettre == v:
            return True
    return False
```

```python
def compter_voyelles(mot: str) -> dict:
    """
    compte le nombre de chaque voyelles de mot
    """
    voyelles = {"a": 0, "e": 0, "i": 0, "o": 0, "u": 0, "y": 0}
    for lettre in mot:
        # utilise la fonction précédente
        if est_voyelle(lettre):
            voyelles[lettre] = voyelles[lettre] + 1
    return voyelles
```

Code 8 – version 1

```python
def compter_voyelles(mot: str)->dict:
    """
    compte le nombre de chaque voyelles de mot
    """
    voyelles = {"a": 0, "e": 0, "i": 0, "o": 0, "u": 0, "y": 0}
    for lettre in mot:
        # compare la lettre aux voyelles
        if lettre in voyelles.keys():
            voyelles[lettre] = voyelles[lettre] + 1
    return voyelles
```

Code 9 – méthode alternative

```python
voyelles = compter_voyelles("orangeade")

for lettre, nb in voyelles.items():
    print(lettre, ":", nb)
```

Code 10 – Affichage du dictionnaire dans le programme principal

```python
def max_voyelles(voyelles: dict) -> list:
    """
    parcourt le dict voyelles et renvoie
    celle qui a la plus grande valeur
    """
    maxi = 0
    lettres_maxi = []
    for lettre, nb in voyelles.items():
        if nb > maxi:
            maxi = nb
            # réinitialise le tableau avec la nouvelle
    lettre max
            lettres_maxi = [lettre]
        elif nb == maxi:
            # ajoute la lettre aux autres lettres max
            lettres_maxi.append(lettre)

    return lettres_maxi
```

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

Exemple d'exécution :

- lettre 'o' :
  - `maxi` $= 1$
  - `lettres_maxi` = ['o']
- lettre 'i' :
  - `maxi` $= 1$
  - `lettres_maxi` = ['o']
- lettre 'a' :
  - `maxi` $= 2$
  - `lettres_maxi` = ['a']
- lettre 'u' :
  - `maxi` $= 2$
  - `lettres_maxi` = ['a']
- lettre 'e' :
  - `maxi` $= 2$
  - `lettres_maxi` = ['a', 'e']

Exercice 7

```
tab = [ [randint(0, 100) for i in range(5)]
                                for j in range(3)]
```

```python
def somme_ligne(tab: list, lig: int) -> int:
    somme = 0
    for val in tab[lig]:
        somme = somme + val
    return somme
```

Code 11 – version 1

```python
def somme_ligne(tab: list, lig: int) -> int:
    somme = 0
    for col in range(len(tab[lig])):
        somme = somme + tab[lig][col]
    return somme
```

Code 12 – version 2

```python
def maximum(tab: list) -> int:
    maxi = 0
    for ligne in tab:
        for colonne in ligne:
            if colonne > maxi:
                maxi = colonne
    return maxi
```

Code 13 – version 1

```python
def maximum(tab: list) -> int:
    maxi = 0
    for lig in range(len(tab)):
        for col in range(len(tab[lig])):
            if tab[lig][col] > maxi:
                maxi = tab[lig][col]
    return maxi
```

Code 14 – version 2

Exercice 8

```python
tab = [[i + j for i in range(3)]
                    for j in range(0, 7, 3)]
```

```python
for ligne in tab:
    for colonne in ligne:
        print(colonne, end=" ")
```

Code 15 – version 1

```python
for i in range(len(tab)):
    for j in range(len(tab[0])):
        print(tab[i][j], end=" ")
```

Code 16 – version 2

Exercice 9

```
lignes = randint(3, 6)
tab = [[randint(0, 100) for i in range(lignes)]
                        for j in range(lignes)]
```

```python
def recuperer_diagonale(tab: list) -> list:
    diagonale = []
    for lig in range(len(tab)):
        diagonale.append(tab[lig][lig])
    return diagonale
```