Exercices parcours séquentiel Correction

Christophe Viroulaud

Première - NSI

Algo 02

```
1 les_notes = [randint(0, 20) for _ in range
      (30)]
```

Code 1 – Construction par compréhension

Sommaire

Exercice 1

```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 note in tab:
        if note < mini:
            mini = note
        if note > maxi:
            maxi = note
    return (mini, maxi)
```

Sommaire

Exercice 2

```
1  def maxi_position(tab: list) -> tuple:
2      """
3      renvoie le max et sa première position
       dans le tableau
4      """
5      # On peut affecter plusieurs variables
       sur 1 ligne
6      indice, note_max = 0, 0
7      for i in range(len(tab)):
8          if tab[i] > note_max:
9              note_max = tab[i]
10             indice = i
11     return (indice, note_max)
```

Sommaire

Exercice 3

```
1  def maxi_position_dernier(tab: list) -> tuple:
2      """
3      renvoie le max et sa dernière position dans
       le tableau
4      """
5      # On peut affecter plusieurs variables sur
       1 ligne
6      indice, note_max = 0, 0
7      for i in range(len(tab)):
8          # il suffit juste de modifier la
       comparaison
9          if tab[i] >= note_max:
10             note_max = tab[i]
11             indice = i
12     return (indice, note_max)
```

Sommaire

Exercice 4

```
1  def maxi_nb(tab: list) -> int:
2      """
3      renvoie le nombre d'occurences du maximum
4      """
5      nb, note_max = 0, 0
6      for note in tab:
7          if note > note_max:
8              # réinitialisation du max
9              note_max = note
10             nb = 1
11         elif note == note_max:
12             nb += 1
13     return nb
```

Sommaire

Exercice 5

```
1  def est_present(tab: list, note: int)->bool:
2      """
3      vérifie si note est dans le tableau
4      """
5      for n in tab:
6          if n == note:
7              return True
8      # On est sorti de la boucle sans avoir
       trouvé note
9      return False
```

La fonction dépend de la taille du tableau. La complexité est **linéaire**.

Sommaire

Exercice 6


```
1  def est_voyelle(lettre:str)->bool:
2      """
3      vérifie si lettre est une voyelle
4      """
5      voyelles = ["a", "e", "i", "o", "u", "y"]
6      for v in voyelles:
7          if lettre == v:
8              return True
9      return False
```

```
1  def compter_voyelles(mot: str) -> dict:
2      """
3      compte le nombre de chaque voyelles de mot
4      """
5      voyelles = {"a": 0, "e": 0, "i": 0, "o": 0, "u": 0, "y": 0}
6      for lettre in mot:
7          if est_voyelle(lettre):  # utilise la fonction précé
       dente
8              voyelles[lettre] += 1
9      return voyelles
```

Code 2 – version 1

```
1  def compter_voyelles(mot: str)->dict:
2      """
3      compte le nombre de chaque voyelles de mot
4      """
5      voyelles = {"a": 0, "e": 0, "i": 0, "o": 0, "u": 0, "y": 0}
6      for lettre in mot:
7          # compare la lettre aux voyelles
8          if lettre in voyelles.keys():
9              voyelles[lettre] += 1
0      return voyelles
```

Code 3 – méthode alternative

```
1 voyelles = compter_voyelles("orangeade")
2
3 for lettre, nb in voyelles.items():
4     print(f"{lettre}: {nb}")
```

Code 4 – Affichage du dictionnaire dans le programme principal

```
1  def max_voyelles(voyelles: dict) -> list:
2      """
3      parcourt le dict voyelles et renvoie
4      celle qui a la plus grande valeur
5      """
6      maxi = 0
7      lettres_maxi = []
8      for lettre, nb in voyelles.items():
9          if nb > maxi:
10             maxi = nb
11             # on réinitialise le tableau avec la
       nouvelle lettre max
12             lettres_maxi = [lettre]
13         elif nb == maxi:
14             lettres_maxi.append(lettre)
15     return lettres_maxi
```