Devoir 4 – Les listes

Exercice 1 : 4 points

Cet exercice est un QCM. Pour chaque question, une seule des réponses proposées est correcte. Une mauvaise réponse, l'absence de réponse ou choisir plusieurs réponses ne rapporte, ni n'enlève aucun point. Écrire sur votre copie le numéro de la question ainsi que la réponse choisie.

1. On considère le tableau `t` suivant :

```
t = [[1, 2, 3], [2, 3, 4], [3, 4, 5], [4, 5, 6]
```

Quelle est la valeur de `t[1][2]` ?

(a) `1` (b) `2` (c) `3` (d) `4`

2. On considère le tableau `tableau` suivant :

```
tableau = [[1, 2, 3], [4, 5, 6], [7, 8, 9]]
```

Laquelle des expressions suivantes vaut `7` ?

(a) `tableau[3, 1]` (b) `tableau[3][1]` (c) `tableau[2, 0]` (d) `tableau[2][0]`

3. Qu'affiche la suite d'instructions ci-dessous ?

```
mat = [[6, 7], [8, 9], [10, 11]
for i in range(len(mat)):
    print(mat[i])
```

(a) `6 8 10`
(b) `7 9 11`
(c) `[6, 7] [8, 9] [10, 11]`
(d) `6 7 8 9 10 11`

4. Que contient la variable `image` après les instructions ci-dessous ?

```
image = [[0, 0, 0, 0], [0, 0, 0, 0], [0, 0, 0, 0], [0, 0, 0, 0]]
for i in range(4):
    for j in range(4):
        if (i+j) == 3:
            image[i][j] = 1
```

(a) `[[0, 0, 0, 0], [0, 0, 0, 0], [0, 0, 0, 0], [1, 1, 1, 1]]`
(b) `[[0, 0, 0, 1], [0, 0, 0, 1], [0, 0, 0, 1], [0, 0, 0, 1]]`
(c) `[[0, 0, 0, 1], [0, 0, 1, 0], [0, 1, 0, 0], [1, 0, 0, 0]]`
(d) `[[0, 0, 0, 1], [0, 0, 1, 1], [0, 1, 1, 1], [1, 1, 1, 1]]`

Exercice 2 : 3 points

Écrire des instructions permettant d'obtenir la liste de listes ci-contre. Il est entendu qu'on ne saisira pas la liste « élément par élément ».

```
liste = [
         [3, 4, 5, 6, 7],
         [6, 8, 10, 12, 14],
         [9, 12, 15, 18, 21],
         [12, 16, 20, 24, 28],
         [15, 20, 25, 30, 35],
         [18, 24, 30, 36, 42]
        ]
```

Exercice 3 : 4 points

Le code ci-contre définit une fonction `retourne` qui prend en paramètre une chaîne de caractères et renvoie la chaîne de caractères « inverse ». Par exemple, l'appel `retourne("Vive Zorglub !")` renvoie la chaîne de caractères `"! bulgroZ eviV"`. Cependant, le professeur de NSI, distrait, a tout mélangé et les différentes indentations ont été perdues !

```
return inverse
for i in range(len(chaine)):
def retourne(chaine):
pos = len(chaine) - i - 1
inverse = inverse + chaine[pos]
inverse = ""
```

1. Réécrire correctement la fonction `retourne`.
2. Proposer une autre façon d'écrire cette fonction `retourne` sans utiliser la variable `pos`.

Exercice 4 : 4 points

Chez le dentiste, la bouche grande ouverte, lorsqu'on essaie de parler, il ne reste que les voyelles. Le but de cet exercice est d'écrire une fonction `dentiste` qui prend en paramètre une chaîne de caractères et qui renvoie la chaîne de caractères obtenue en ne gardant que les voyelles. Par exemple, l'appel `dentiste("J'ai mal !")` renvoie la chaîne de caractères `"aia"`.

On rappelle que les voyelles sont les lettres a, e, i, o, u et y.

1. On suppose qu'une fonction `est_voyelle` a déjà été écrite. Cette fonction renvoie `True` si la lettre (chaîne de caractères) donnée en paramètre est une voyelle, et `False` sinon.
   Écrire la fonction `dentiste`.
2. Écrire la fonction `est_voyelle`.

Exercice 5 : 5 points

Voici les premières lignes du triangle de Pascal :

```
1
1   1
1   2   1
1   3   3   1
1   4   6   4   1
```

Chaque ligne de ce triangle commence et se termine par un 1, et chaque nombre (à part le premier et le dernier) est égal à la somme du nombre situé au-dessus et du nombre situé « au-dessus et à gauche ».
La ligne suivante de ce triangle est donc :

```
1   5   10   10   5   1
```

1. On décide de représenter chaque ligne de ce triangle par une liste d'entiers en Python. On représente par exemple la cinquième ligne du triangle par la liste `[1, 4, 6, 4, 1]`.
   Écrire une fonction `ligne_suivante` qui prend en paramètre une liste représentant une ligne du triangle de Pascal et qui renvoie la liste représentant la ligne suivante du triangle. Par exemple, `ligne_suivante([1, 3, 3, 1])` renvoie la liste `[1, 4, 6, 4, 1]`.
2. On représente maintenant les premières lignes du triangle de Pascal par une liste de listes :

   `[[1], [1, 1], [1, 2, 1], [1, 3, 3, 1], [1, 4, 6, 4, 1]]`

   Compléter la fonction `triangle` ci-dessous afin qu'elle renvoie les `n` premières lignes du triangle de Pascal, `n` étant un entier naturel non nul donné en paramètre de la fonction.
   Par exemple, `triangle(3)` renvoie `[[1], [1, 1], [1, 2, 1]]`.

```
1  def triangle(n):
2      if n == 1:
3          ...
4      elif n == 2:
5          ...
6      else:
7          ...
8          ...
9          ...
10         ...
```

Remarque. – Il n'y a pas nécessairement exactement 4 lignes à ajouter dans le `else`*.*