

# Corrigé sujet 04 - Année : 2022

[Sujet 04 - 2022 ↴](#)

## 1. Exercice 1

### Script Python

```

1 def recherche(liste):
2     consecutifs = []
3     for i in range(len(liste)-1):
4         if liste[i+1]==liste[i]+1:
5             consecutifs.append((liste[i], liste[i+1]))
6     return consecutifs

```

### Commentaires

- La condition `liste[i+1]==liste[i]+1` permet de tester que deux éléments consécutifs de la liste sont deux entiers qui se suivent.
- On peut utiliser les définitions de liste par compréhension :

### Script Python

```

def recherche(liste):
    return [(liste[i], liste[i+1]) for i in range(len(liste)-1) if
    liste[i+1]==liste[i]+1]

```

## 2. Exercice 2

### Script Python

```

1 def propager(M, i, j, val):
2     if M[i][j]== val: #(1)
3         return
4
5     M[i][j]=val
6
7     # l'élément en haut fait partie de la composante
8     if ((i-1) >= 0 and M[i-1][j] == 1): #(2)
9         propager(M, i-1, j, val)
10

```

```

11     # l'élément en bas fait partie de la composante
12     if ((i+1) < len(M) and M[i+1][j] == 1): #(2)
13         propager(M, i+1, j, val)
14
15     # l'élément à gauche fait partie de la composante
16     if ((j-1) >= 0 and M[i][j-1] == 1): #(2)
17         propager(M, i, j-1, val)
18
19     # l'élément à droite fait partie de la composante
20     if ((j+1) < len(M) and M[i][j+1] == 1): #(2)
21         propager(M, i, j+1, val)

```

1. C'est la condition d'arrêt de la récursivité, on ne relance pas la propagation sur les cases voisines
2. On relance la propagation à partir de la case voisine si celle-ci est dans la grille (première partie de la condition) et aussi dans la même composante (seconde partie de la condition)
3. On relance la propagation à partir de la case voisine si celle-ci est dans la grille (première partie de la condition) et aussi dans la même composante (seconde partie de la condition)
4. On relance la propagation à partir de la case voisine si celle-ci est dans la grille (première partie de la condition) et aussi dans la même composante (seconde partie de la condition)
5. On relance la propagation à partir de la case voisine si celle-ci est dans la grille (première partie de la condition) et aussi dans la même composante (seconde partie de la condition)

### Commentaire

Le `return` ligne 3 (équivalent à un `return None`) permet de mettre fin à la récursivité. On peut faire autrement et éviter d'utiliser `return` d'autant plus que cette fonction modifie une liste en place mais ne renvoie pas de valeur.