Exercices : Listes

Exercice 01 Qu'affichent les programmes suivants ?

```
1  # Programme 1
2
3  L = [67, 88, 98, 15, 83]
4
5  print(L[2])
```

Réponse : .....

```
1  # Programme 2
2
3  L = [1, 2, 3, 4]
4
5  print(L[1] * L[-1])
```

Réponse : .....

```
1  # Programme 3
2
3  L = ["salut", "bonjour", "bonsoir", "ciao"]
4
5  print(len(L[1]))
```

Réponse : .....

```
1  # Programme 4
2
3  L = [6, 8, 3, 9, 7, 1]
4
5  print(max(L) * sum(L))
```

Réponse : .....

Exercice 02 Que contient la liste `L` à la fin des programmes suivants ?

```
1  # Programme 1
2
3  L = []
4
5  for k in range(4):
6      L.append(k**2)
```

Réponse : .....

```
1  # Programme 2
2
3  L = []
4  a = 1
5
6  for i in range(1, 6):
7      L.append(a)
8      a += a
```

Réponse : .....

```
1  # Programme 3
2
3  L = [3]
4
5  while sum(L) < 30:
6      L.append(2*max(L)+1)
```

Réponse : .....

```
1  # Programme 4
2
3  L = []
4  x = 8
5
6  while x >= 0:
7      L.append(x**2-1)
8      x = x-2
```

Réponse : .....

Exercice 03 Même exercice :

```
# Programme 1

L = [5, 8, 3, 2, 6]
N = len(L)

for i in range(N):
    L[-i] = L[i]
```

Réponse : .....

```
# Programme 2

L = [2, 2]

while max(L) < 100:
    L.append(L[-1] * L[-2])
```

Réponse : .....

Exercice 04 On considère la fonction suivante :

```
def fonction(liste):
  n = 0
  p = 0

  for e in liste:
      if e >= 0:
          p += 1
      else:
          n += 1

  return n*p
```

1. Que retourne la fonction si `liste = [1,-4,7]` ?

   .....

2. Quel est le rôle de cette fonction ?

   .....

3. La fonction peut-elle retourner 4 ? Si oui, donner un tel exemple pour `liste` .

   .....

Exercice 05 Quel est le rôle des fonctions suivantes ?

```
def mystere1(liste):
    n = 0

    for e in liste:
        if e > 0:
            n += 1

    return n
```

Réponse : .....

```
def mystere2(liste, el):
    c = 0

    for i in range(len(liste)):
        if liste[i] == el:
            c += 1

    return c
```

Réponse : .....

Exercice 06 Écrire une fonction `somme(liste1, liste2)` qui retourne une liste formée par la somme terme à terme des éléments de `liste1` et `liste2`.

Si `liste1` et `liste2` ne sont de même longueur, la fonction doit retourner `None`.

```
>>> somme([1,2,3], [4,5,6])
[5,7,9]

>>> somme([2,4,6,8,10], [1,3,5,7,9])
[3,7,11,15,19]

>>> somme([1,2,3], [4,5,6,7])
None
```