Corrigé – Les parcours par valeurs et par indice

Exercice 1

1. 163
2. La fonction convertit une chaîne de caractères représentait un nombre binaire en nombre décimal.

Exercice 2

1.
```
def longueur(ch):
    nb_carac = 0
    for carac in ch:
        nb_carac = nb_carac + 1
    return nb_carac
```

2.
```
def sans_dernier(ch):
    ch_sans_der = ""
    for carac in ch:
        if longueur(ch_sans_der) < longueur(ch) - 1:
            ch_sans_der = ch_sans_der + carac
    return ch_sans_der
```

3.
```
def caractere(ch, idx):
    for idx_courant in range(longueur(ch)):
        if idx_courant == idx:
            return ch[idx_courant]
```

Exercice 3

4

Exercice 4

1.
```
def longueur(lst):
    nb_elements = 0
    for elem in lst:
        nb_elements = nb_elements + 1
    return nb_elements
```

2.
```
def somme(lst):
    somme = 0
    for element in lst:
        somme = somme + element
    return somme
```

3.
```
def maxi(lst):
    max_element = 0
    for element in lst:
        if element > max_element:
            max_element = element
    return max_element
```

4.
```
def appartient(lst, el):
    for element in lst:
        if el == element:
            return True
    return False
```

Exercice 5

```
f = open ("mystere.txt", "r")
chaine = ""
for line in f:
    carac = chr(int(line))
    chaine = chaine + carac

print(chaine)
```

Exercice 6

1. 0 à n-1
2. La chaîne a une longueur de 4 caractères et les indices vont de 0 à 3 (4 valeurs différentes). L'indice 4 provoque donc une erreur puisqu'il n'existe pas ici.

Exercice 7

```
def extension(fichier):
    extension = ""
    for idx in range(len(fichier) - 1, len(fichier) - 4, -1):
        extension = fichier[idx] + extension
    return extension
```

Exercice 8

1. 0 à n-1
2. La liste a une longueur de 4 éléments et les indices vont de 0 à 3 (4 valeurs différentes). L'indice 4 provoque donc une erreur puisqu'il n'existe pas ici.

(TSVP)

Exercice 9

```python
# IMPORTS
from math import sqrt

# FONCTIONS
# 1)
def distance(a, b):
    xa = a[0]
    ya = a[1]
    xb = b[0]
    yb = b[1]

    return sqrt((xb - xa)**2 + (yb - ya)**2)

# 2)
def milieu(a, b):
    xa, ya = a[0], a[1]
    xb, yb = b[0], b[1]

    return [(xa + xb)/2, (ya + yb)/2]

# SCRIPT
# tests
assert distance([4, 3], [4, 9]) == 6.0
assert distance([4, 3], [10, 3]) == 6.0
assert distance([4, 4], [4, 4]) == 0.0

assert milieu([2, 3], [4, 6]) == [3, 4.5]
```

Exercice 10

```python
# FONCTIONS
# 1)
def niveau_de_gris(rvb):
    r, v, b = rvb
    moyenne = (r + v + b) / 3
    return round(moyenne, 1)

# 2)
def negatif(rvb):
    r, v, b = rvb
    return [255 - r, 255 - v, 255 - b]

# SCRIPT
# tests
assert niveau_de_gris([2, 2, 2]) == 2.0
assert niveau_de_gris([1, 2, 3]) == 2.0
assert negatif([255, 255, 255]) == [0, 0, 0]
assert negatif([0, 0, 0]) == [255, 255, 255]
```

Exercice 11

```
# FONCTIONS
def est_palindromique(lst):
    for idx in range(len(lst)):
        if lst[idx] != lst[len(lst)-1-idx]:
            return False
    return True

# SCRIPT
# tests
assert est_palindromique(["ab", "cd", "ef", "cd", "ab"])
assert not est_palindromique(["bb", "cd", "ef", "cd", "ab"])
```

Exercice 12

```
# FONCTIONS
def slice(chaine, debut, fin):
    sous_chaine = ""
    for idx in range(len(chaine)):
        if idx >= debut and idx < fin:
            sous_chaine = sous_chaine + chaine[idx]
    return sous_chaine

# SCRIPT
# tests
assert slice("tension", 2, 5) == "nsi"
```

Exercice 13

```
# FONCTIONS
def indice_element(lst, val):
    for idx in range(len(lst)):
        if lst[idx] == val:
            return idx
    return -1

# SCRIPT
# tests
assert indice_element([1, 2, 4, 6, 4, 3], 4) == 2
assert indice_element([1, 2, 4, 6, 4, 3], 5) == -1
```

Exercice 14

```
# FONCTIONS
def indice_max(lst):
    maxi = 0
    indice_maxi = 0
    for indice in range(len(lst)):
        if lst[indice] > maxi:
```

```
            maxi = lst[indice]
            indice_maxi = indice
    return indice_maxi

# SCRIPT
# tests
assert indice_max([1, 2, 4, 6, 4, 3]) == 3
assert indice_max([1, 2, 4, 6, 4, 3, 6]) == 3
```