Exercice 1:

Correction:

```
1  def RechercheMin(tab):
2      m = tab[0]
3      i = 0
4      index = 0
5      for v in tab:
6          if v < m:
7              m = v
8              index = i
9          i = i + 1
10     return index
```

Description :

On initialise M au premier élément, si celui-ci est le plus petit, la valeur sera donc égale à lui et non à 0.

On initialise I à zéro, car cela correspondra au premier élément que l'on vérifiera (au sein de la liste).

Index est tout d'abord égal à 0. Il sera modifié durant l'exécution du programme.

```
m = tab[0]
i = 0
index = 0
```

```
for v in tab:
    if v < m:
        m = v
        index = i
    i = i + 1
```

Pour chaque valeur (v) au sein de la liste (tab), si la valeur est supérieure à m (première valeur enregistrée), cette valeur v devient la valeur m, et index prendra donc la valeur de i.

i est une valeur qui s'incrémente, c'est à dire qu'elle augmente durant l'exécution du programme.

Une fois la liste entièrement parcourue, l'indice conservé (dans index) est donc renvoyé.

```
return index
```

Remarques :

L'une des premières choses auxquelles il faut faire attention, c'est qu'il faut renvoyer l'indice et non la valeur minimale.

Ce code peut être optimisé en passant par un 'for i in range' mais cette méthode permet de ne pas se tromper.

Exercice 2:

Correction:

```
1  def separe(tab):
2      i = 0
3      j = len(tab) -1
4      while i < j:
5          if tab[i] == 0:
6              i += 1
7          else:
8              tab[i],tab[j] = tab[j],tab[i]
9              j -= 1
10     return tab
```

Explication:
on initialise i a 0
on fait parcourir j dans la liste tab et on le soustrait à -1 pour éviter de provoquer une erreur

```
def separe(tab):
    i = 0
    j = len(tab) -1
    while i < j:
```

si l'indice i de tab est égal à 0,
on ajoute +1 à i tant que i est inférieur à j et que l'indice tab[i] est égal à 0

```
while i < j:
    if tab[i] == 0:
        i += 1
```

sinon on inverse tab[i] et tab[j] de tab et
on retire -1 à j car si on ajoute + 1 à j alors qu'on a jouter +1 à i cela va provoquer une erreur
et on renvoie tab

```
        else:
            tab[i],tab[j] = tab[j],tab[i]
            j -= 1
    return tab
```

Résulat des exemples:

```
>>> separe([1, 0, 1, 0, 1, 0, 1, 0])
[0, 0, 0, 0, 1, 1, 1, 1]
```

```
>>> separe([1, 0, 0, 0, 1, 1, 0, 1, 1, 0, 1, 0, 1, 1, 1, 0])
[0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1]
```