Sujet 2

Exercice 1 :

```
1  def indices_maxi(tab):
2      ind_max = []
3      val_max = tab[0]
4      for i in range (len(tab)):
5          if tab[i] > val_max :
6              val_max = tab[i]
7      for i in range (len(tab)):
8          if tab[i] == val_max :
9              ind_max.append(i)
10     return (val_max, ind_max)
```

Ligne 1 :
On définit une fonction qui prend en entrée une liste "tab"

Lignes 2 :
On initialise une liste vide qui sera utilisée pour stocker les indices des valeurs maximales.

Ligne 3 :
On initialise "val_max" avec la première valeur de la liste "tab".

Ligne 4 :
On créer une boucle de la longueur de la liste "tab".

Ligne 5 :
On vérifie si l'élément actuel de la liste "tab" à l'index "i" est supérieur à "val_max".

Ligne 6 :
Si la condition précédente est vraie, "val_max" prend la valeur de l'élément actuel de "tab" à l'index "i".

Ligne 7 :
Voir ligne 4.

Ligne 8 :
Cette instruction vérifie si l'élément actuel de la liste "tab" à l'index "i" est égal à "val_max".

Ligne 9 :
Si la condition précédente est vraie, l'index "i" est ajouté à la liste "ind_max".

Ligne 10 :
La fonction retourne un tuple contenant la valeur maximale de la liste "tab" et la liste des indices où cette valeur maximale apparaît.

Exercice 2 :

```
1  def positif(pile):
2      pile_1 = list(pile)
3      pile_2 = []
4      while pile_1 != []:
5          x = pile_1.pop()
6          if x >= 0:
7              pile_2.append(x)
8      while pile_2 != []:
9          x = pile_2.pop()
10         if x >= 0:
11             pile_1.append(x)
12     return pile_1
```

Ligne 1 :

On définit une fonction qui prend en entrée une pile (où les éléments sont ajoutés ou retirés en utilisant le principe Last In, First Out).

Ligne 2 :

On crée une copie de la pile initiale. "pile_1" sera utilisée pour effectuer des opérations sans modifier la pile d'origine.

Ligne 3 :

On initialise une liste vide "pile_2". Cette liste sera utilisée pour stocker les valeurs positives de la pile.

Ligne 4 :

La boucle "while" s'exécute tant que la pile "pile_1" n'est pas vide.

Ligne 5 :

On prend le dernier élément de la pile "pile_1" et le stocke dans la variable "x", tout en le retirant de "pile_1".

Ligne 6 :

On vérifie si la valeur "x" est supérieure ou égale à zéro.

Ligne 7 :

Si la valeur "x" est positive, elle est ajoutée à la pile "pile_2".

Ligne 8 :

La boucle "while" s'exécute tant que la pile "pile_2" n'est pas vide.

Ligne 9 :

On prend le dernier élément de la pile "pile_2" et on stocke dans la variable "x", tout en le retirant de "pile_2".

Ligne 10 :
On vérifie si la valeur "x" est supérieure ou égale à zéro.

Ligne 11 :
Si la valeur "x" est positive, elle est ajoutée à la pile "pile_1".

Ligne 12 :
La fonction retourne la pile "pile_1", qui contient maintenant tous les éléments positifs dans leur ordre d'origine et les éléments non positifs ont été retirés.