Filtrer une pile

Filtrer une pile

On met en œuvre les piles à l'aide de la classe `Pile` suivante :

```
class Pile:
    def __init__(self):
        self.valeurs = []

    def est_vide(self):
        """Détermine si la pile est vide"""
        return len(self.valeurs) == 0

    def empile(self, x):
        """Empile x dans la pile"""
        self.valeurs.append(x)

    def depile(self):
        """Dépile une valeur et la renvoie
        Lève une erreur si la pile est vide
        """
        if self.est_vide():
            raise ValueError("La pile est vide")
        return self.valeurs.pop()
```

Toutes les opérations sur les piles se feront par l'intermédiaire des méthodes de cette classe. Il est donc **interdit** d'accéder ou de modifier directement l'attribut `valeurs` d'une `Pile`.

On considère une pile ne contenant que des nombres entiers de signe quelconque.

On souhaite filtrer les valeurs de cette pile afin d'obtenir deux nouvelles piles :

- la pile des valeurs positives ou nulles,
- la pile des valeurs strictement négatives.

Dans chacune des piles résultantes, les valeurs seront dans le même ordre que dans la pile initiale.

Schématiquement :

```
Pile initiale           Pile des        Pile des
    -9                  positifs        négatifs
    -4
    6
    2
    -5                                    -9
    -3          ==>       6               -4
    2                     2               -5
    -1                    2               -3
    1                     1               -1
  =====                 =====           ======
```

Objectif

On demande d'écrire deux fonctions :

- `renverse` prend en paramètre une pile et renvoie une nouvelle pile contenant les mêmes valeurs, mais renversées (les valeurs initialement en haut de la pile sont désormais en bas et réciproquement);
- `filtre` prend en paramètre une pile et renvoie le couple formé de la pile des nombres positifs et de celle des nombres strictement négatifs.

Ces deux fonctions peuvent modifier la pile reçue en paramètre. Il n'est pas demandé de la "reconstruire" après le traitement.

Exemples

```
>>> pile_1 = Pile()
>>> for x in [1, 2, 3]:
        pile_1.empile(x)
>>> pile_1.valeurs
[1, 2, 3]
>>> envers_1 = renverse(pile_1)
>>> envers_1.valeurs
[3, 2, 1]

>>> pile_2 = Pile()
>>> for x in [1, -1, 2, -3, -5, 2, 6, -4, -9]:
        pile_2.empile(x)
>>> positifs, negatifs = filtre(pile_2)
>>> positifs.valeurs
[1, 2, 2, 6]
>>> negatifs.valeurs
[-1, -3, -5, -4, -9]

>>> pile_3 = Pile()
```

```
>>> pile_3.empile(3)
>>> positifs, negatifs = filtre(pile_3)
>>> positifs.valeurs
[3]
>>> negatifs.valeurs
[]
```