# Copier une pile

## Copier 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()

    def __eq__(self, autre):
        """Détermine si pile_1 == pile_2"""
        return self.valeurs == autre.valeurs
```

### Objectif

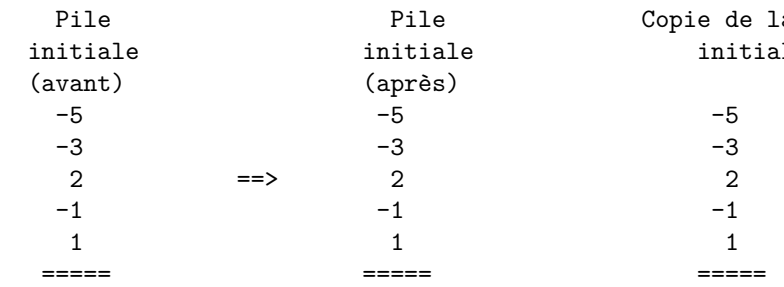
Toutes les opérations sur les piles se feront par l'intermédiaire des méthodes de cette classe. Il est donc **interdit** d'accéder ou de modifier directement l'attribut `valeurs` d'une `Pile`.

On demande d'écrire la fonction `copie` qui prend en paramètre une pile et renvoie une nouvelle pile contenant les mêmes valeurs dans le même ordre que

la pile initiale.

La pile passée en paramètre devra contenir les mêmes valeurs au début et à la fin de l'exécution de la fonction.

Schématiquement :




## Exemples

```
>>> pile = Pile()
>>> temoin = Pile()
>>> for x in [1, 2, 3]:
        pile.empile(x)
        temoin.empile(x)
>>> copie_pile = copie(pile)
>>> pile == temoin
True
>>> pile == copie_pile
True
>>> pile.depile()
3
>>> pile == copie_pile
False
```