

# TDA : Pile

**Sébastien Jean**

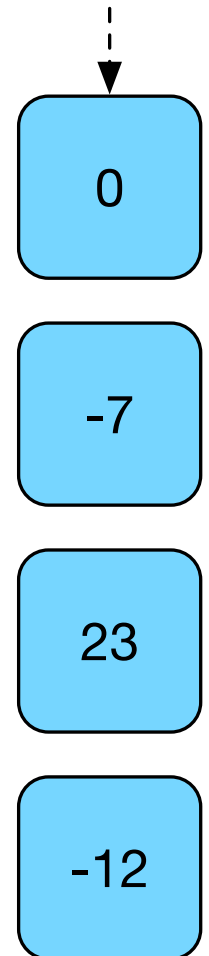
IUT de Valence  
Département Informatique

v1.0, 26 novembre 2025

# Type de données *Pile*

- Type de données **linéaire** et **dynamique**, similaire à un vecteur mais accès LIFO (*Last In First Out*), notion de **sommet**
- Possibilité d'**obtenir la taille**, de **savoir si la pile est vide**, de **voir l'élément en sommet de pile**, d'**ajouter ou de retirer un élément** (en sommet de pile)

Sommet



# TDA *Pile*

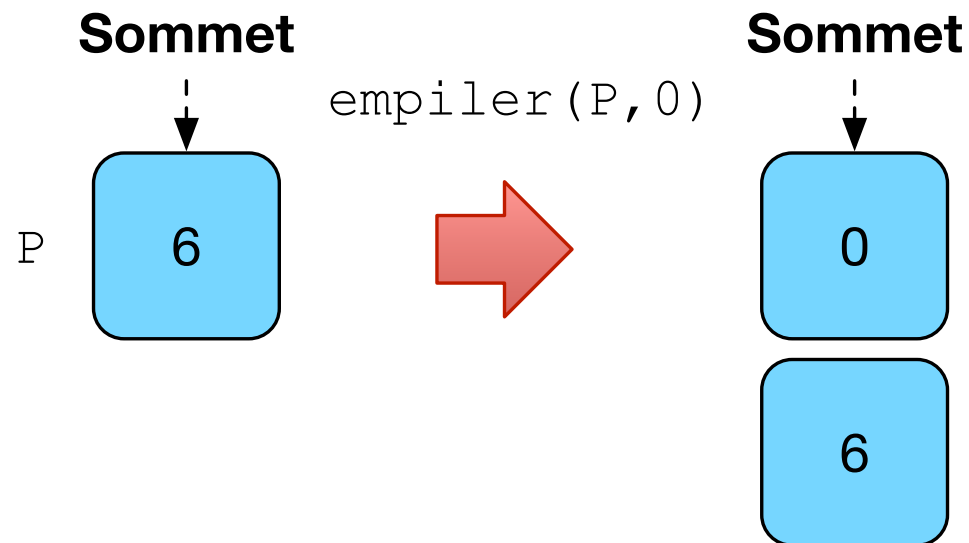
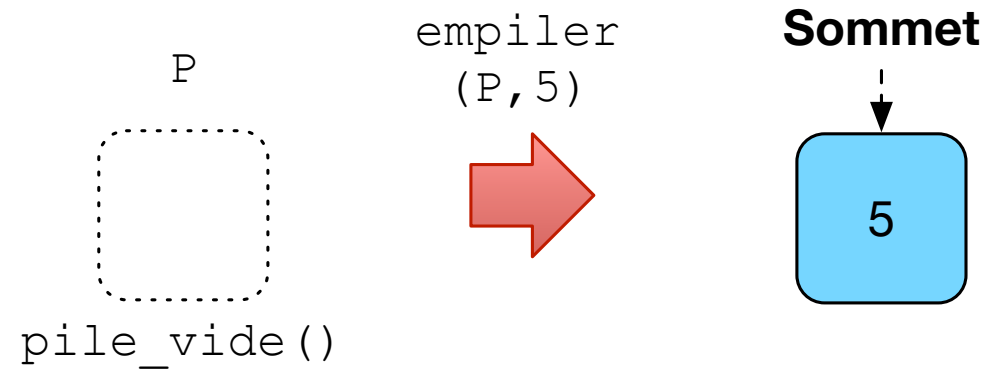
- Nom : *Pile* (de *T*)
- Dépendances : Booléen, Entier, *T* (type des éléments)
- Opérations :
  - Constructeurs :
    - *pile\_vide* :  $\rightarrow \text{Pile}$
  - Transformateurs :
    - *empiler* :  $\text{Pile} \times T \rightarrow \text{Pile}$
    - *dépiler* :  $\text{Pile} \rightarrow \text{Pile}$
  - Observateurs :
    - *taille* :  $\text{Pile} \rightarrow \text{Entier}$
    - *est\_vide* :  $\text{Pile} \rightarrow \text{Booléen}$
    - *voir\_sommet* :  $\text{Pile} \rightarrow T$
- *A suivre ...*

- Pré-conditions

- `dépiler(p) → est_vide(p) = FAUX`
- `voir_sommet(p) → est_vide(p) = FAUX`

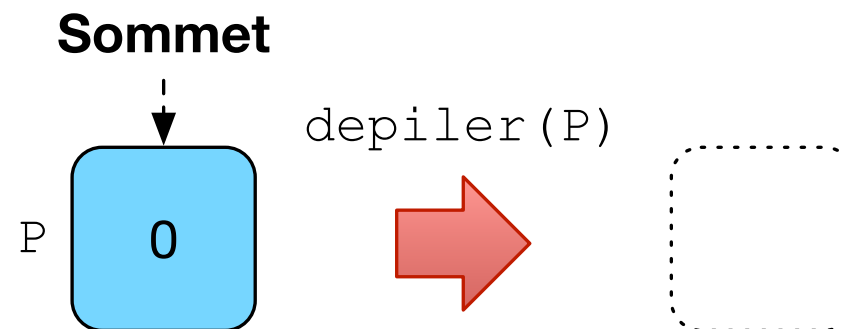
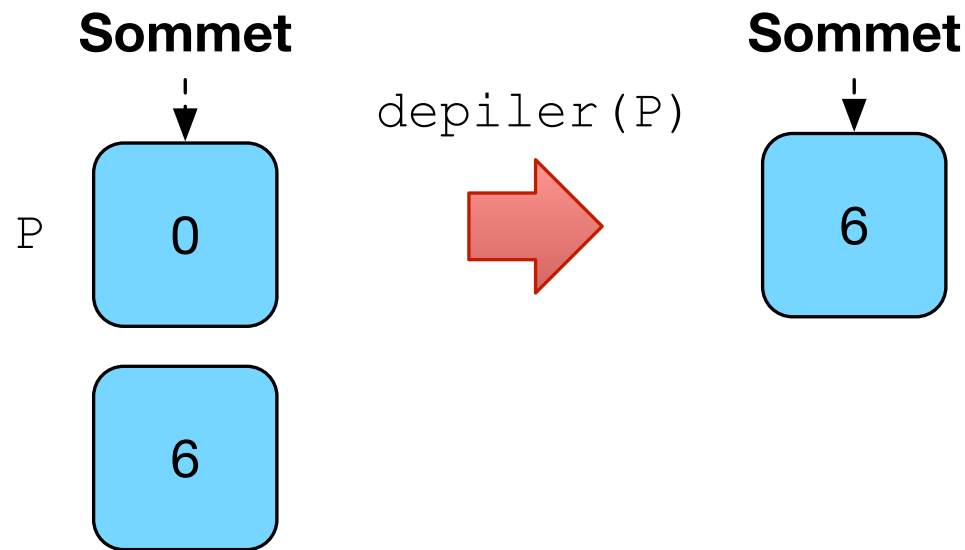
# Exemples d'opérations/situations

- Empiler un élément



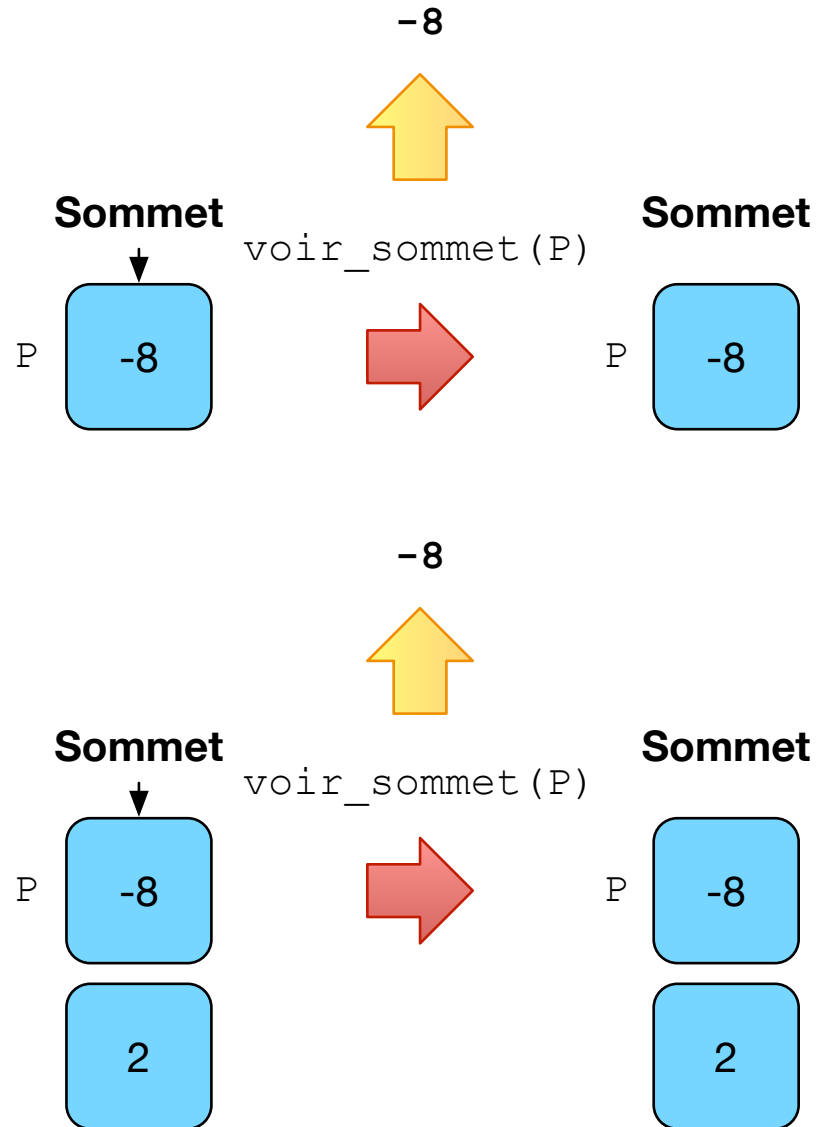
# Exemples d'opérations/situations

- Dépiler un élément



# Exemples d'opérations/situations

- Voir le sommet



- **Axiomes**

- `est_vide(pile_vide()) = VRAI`
- `est_vide(empiler(p, e)) = FAUX`
- `taille(pile_vide()) = 0`
- `taille(empiler(p, e)) = taille(p) + 1`
- `taille(dépiler(p)) = taille(p) - 1`
- `dépiler(empiler(p, e)) = p`
- `voir_sommet(empiler(p, e)) = e`



# Pile et pseudo code

- Dans notre *pseudo code*, on suppose que
  - Le **constructeur** est remplacé par la **déclaration de la variable**
  - Les **transformateurs** sont des **mutateurs**
  - Les **paramètres** sont **passés par copie**

---

```
VARIABLE p : Pile d'entiers  
VARIABLE i : entier
```

```
empiler(p, 3)  
empiler(p, 7)
```

```
i ← voir_sommet(p)  
depiler(p)
```

```
AFFICHER(voir_sommet(p))
```

# Exercice

## Enoncé du problème

On souhaite inverser les éléments d'une pile.

## Spécification du problème

- Donnée d'entrée :  $p$ , pile de  $T$  (la pile à inverser)
- Donnée de sortie :  $r$ , pile de  $T$  (la pile inversée)
- Pré-condition : (aucune)
- Post-condition :  $p$  et  $r$  contiennent les mêmes éléments mais dans l'ordre inverse

## Signature de la fonction

- **inverse\_pile** ( $p$  : pile de  $T$ ) : pile de  $T$

# Exercice

---

```
FONCTION inverse_pile(p : pile de T) : pile de T

    VARIABLE compteur    : entier
    VARIABLE r            : pile de T

    POUR compteur de 1 A taille(p) PAR PAS DE 1
        empiler(r, voir_sommet(p))
        depiler(p)
    FIN POUR

    RETOURNER r

FIN FONCTION
```

---

# Exercice

---

```
FONCTION  inv_pile_rec(p : pile de T,  
                      p_inv : pile de T) : pile de T  
  SI  est_vide(p)  ALORS  
    RETOURNER  p_inv  
  FIN  SI  
  
  empiler(p_inv, voir_sommet(p))  
  depiler(p)  
  
  RETOURNER  inv_pile_rec(p, p_inv)  
FIN  FONCTION
```

---

---

```
FONCTION  inverse_pile(p : pile de T) : pile de T  
  VARIABLE  p_inv : pile de T  
  
  RETOURNER  inv_pile_rec(p, p_inv)  
FIN  FONCTION
```

---

# Fin !

