

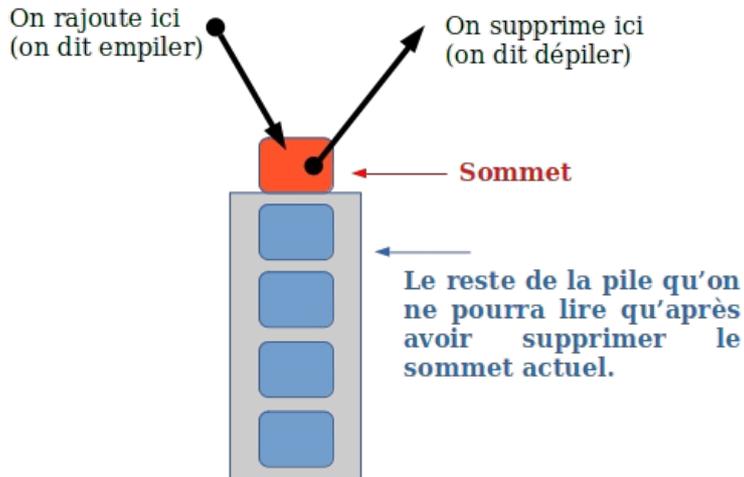
RÉSUMÉ 15 - TYPE ABSTRAIT PILE LIFO

Lien vers l'activité : [type abstrait Pile LIFO](#)
www.infoforall.fr - Dernière modif. : 25 10 2020



1-TYPE ABSTRAIT PILE / STACK : ORGANISATION ET INTERFACE

C'est une **séquence ordonnée et finie** d'éléments respectant le principe **LIFO**.
LIFO veut dire **Last In First Out**. Le dernier élément arrivé sera le premier à repartir.



FONCTIONS D'INTERFACE MINIMALE

On retrouve les fonctions de la liste mais limitées à l'action sur le SOMMET. On en trouve des versions mutables ou non mutables. Ici, c'est non mutable puisqu'on renvoie des Piles après chaque modification.

- `nouvellePile()` -> `Pile`
- `estVide(p:Pile)` -> `bool`
- `empiler(elt:Elt, p:Pile)` -> `Pile` : on rajoute au SOMMET
- `depiler(p:Pile)` -> `Pile` : on supprime le SOMMET
- `lireSommet(p:Pile)` -> `Elt`

2-IMPLÉMENTATIONS D'UNE PILE AVEC PYTHON

IMPLÉMENTATION SOUS FORME D'UN TABLEAU OU D'UNE LISTE CHAÎNÉE

```
1  def nouvellePile() :  
2      return []  
3  
4  def estVide(pile) :  
5      return pile == []  
6  
7  def empiler(x, pile) :  
8      pile.append(x) # rajoute un élément à la fin du tableau  
9  
10 def depiler(pile) :  
11     if not estVide(pile) :  
12         return pile.pop() # supprime et renvoie le dernier élément du tableau  
13  
14 def lireSommet(pile) :  
15     if not estVide(pile) :  
16         return pile[-1] # ou return pile[len(pile)-1]
```

On peut utiliser un **TABLEAU** ou une **LISTE CHAÎNÉE** puisqu'au final une **PILE** est un peu un cas particulier et limité de la **LISTE**. On peut utiliser le type natif `list` si on fait du Python et qu'on ne veut pas tout avoir à refaire. On utilise alors un tableau dynamique.

IMPLÉMENTATION SOUS FORME DE TUPLE (SOMMET, QUEUE)

Nous avons vu que le coût en insertion / suppression de la tête était constant pour cette implémentation. Or, une pile est bien une telle liste dont la tête se nomme le sommet.

```
1 class Elt :
2     '''Juste pour pouvoir noter Elt dans les prototypes'''
3
4 class Pile :
5     '''Juste pour pouvoir noter Pile dans les prototypes'''
6
7     '''Implémentation de type abstrait Pile en utilisant des tuples (sommet, autre_pile)'''
8
9     def nouvellePile() -> Pile :
10        '''Renvoie une liste vide'''
11        return ()
12
13    def pileVide(p:Pile) -> bool :
14        '''Renvoie True si la pile est vide'''
15        return p == ()
16
17    def empiler(elt:Elt, p:Pile) -> Pile :
18        '''Renvoie une nouvelle pile où x est le sommet et pile la suite de notre nouvelle pile.'''
19        return (elt, p)
20
21    def lireSommet(p:Pile) -> Elt :
22        '''Renvoie la valeur stockée au sommet, sans la supprimer de la pile'''
23        if not pileVide(p) :
24            return p[0]
25
26    def depiler(p:Pile) -> Pile :
27        '''Renvoie une nouvelle pile où on a supprimé l'ancien sommet'''
28        if pileVide(p) : # la liste envoyée est ()
29            return nouvellePile()
30        else :
31            return p[1]
```

www.infoforall.fr

