

# Structure de données 1

# PILES

## TYPE ABSTRAIT:

- interface
- implementation fonctionnelle
- implementation en POO
- exemple d'utilisation

FONCTIONNEL	POO
<pre>def Pile():     return []  def est_vide(pile):     return pile == []  def depile(pile):     assert pile != [], 'impossible de depiler : pile vide'     return pile.pop()  def empile(pile,a):     pile.append(a)  def sommet(pile):     assert pile != [], 'pile vide'     return pile[-1]</pre>	<pre>class Pile:     def __init__(self):         self.lst = []      def est_vide(self):         return self.lst == []      def depile(self):         return self.lst. ..      def empile(self, a):         ..      def sommet(self):         return ..</pre>
<pre># Créer pile P avec les lettres 'p', 'i', '1', 'e' &gt;&gt;&gt; P = Pile() &gt;&gt;&gt; for c in list('pile'):     .. empile(P,c) &gt;&gt;&gt; P ['p', 'i', '1', 'e']</pre>	<pre># Créer pile P &gt;&gt;&gt; P = Pile() &gt;&gt;&gt; for c in list('pile'):     .. P.empile(c)  &gt;&gt;&gt; P.lst ['p', 'i', '1', 'e']</pre>
<pre># depiler &gt;&gt;&gt; depile(P) 'e' &gt;&gt;&gt; P ['p', 'i', '1']</pre>	<pre># depiler &gt;&gt;&gt; P.depile() 'e' &gt;&gt;&gt; P.lst ['p', 'i', '1']</pre>
<pre># sommet &gt;&gt;&gt; sommet(P) '1'</pre>	<pre># sommet &gt;&gt;&gt; P.sommet() '1'</pre>

## Application

### 1. implémentation à l'aide de fonctions:

Donner l'état de la pile après chaque étape (numérotées) :

```
>>> P = Pile()
>>> empile(P,5) # (1)
>>> empile(P,8) # (2)
>>> empile(P,3) # (3)
>>> depile(P) # (4)
>>> empile(P,12) # (5)
>>> depile(P) # (6)
>>> depile(P) # (7)
```

### 2. Implémentation à l'aide d'une classe:

idem avec les instructions:

```
>>> p = Pile() # (1)
>>> p.empile(10) # (2)
>>> p.empile(20) # (3)
>>> print(p.depile()) # (4)
>>> print(p.sommet()) # (5)
```