Structure de données 1 PILES

TYPE ABSTRAIT:
- interface
- implementation fonctionnelle
- implementation en POO
- exemple d'utilisation

FONCTIONNEL	POO

`def Pile():`
`  return []`
 	
`def est_vide(pile):`
`  return pile == []`
 	
`def depile(pile):`
`  assert pile != [], 'impossible de depiler : pile vide'`
`  return pile.pop()`
 	
`def empile(a,pile):`
`  pile.append(a)`
 	
`def sommet(pile):`
`  assert pile != [], 'pile vide'`
`  return pile[-1]`
	`class Pile:`
	`  def __init__(self):`
	`  self.lst = []`
	
	`  def est_vide(self):`
	`  return self.lst = []`
	
	`  def depile(self):`
	`  self.lst. ..`
	
	`  def empile(self, a):`
	`  ..`
	
	`  def sommet(self):`
	`  return ..`

`# Créer pile P avec les lettres 'p', 'i', 'l', 'e'`
`>>> P = Pile()`
`>>> for c in list('pile'):`
`  .. empile(P,c)`
`>>> P`
`['p', 'i', 'l', 'e']`
	`# Créer pile P`
	`>>> P = Pile()`
	`>>> for c in list('pile'):`
	`  .. P.empile(c)`
	
	`>>> P.lst`
	`['p', 'i', 'l', 'e']`

`# depiler`
`>>> depile(P)`
`'e'`
`>>> P`
`['p', 'i', 'l']`
	`# depiler`
	`>>> P.depile()`
	`'e'`
	`>>> P.lst`
	`['p', 'i', 'l']`

`# sommet`
`>>> sommet(P)`
`'l'`
	`# sommet`
	`>>> P.sommet()`
	`'l'`

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)
```