

Exercice type bac sur les piles

La classe Pile utilisée dans cet exercice est implémentée en utilisant des listes Python et propose quatre éléments d'interface :

- Un constructeur qui permet de créer une pile vide, représentée par [] ;
- La méthode `est_vide()` qui renvoie True si l'objet est une pile ne contenant aucun élément, et False sinon ;
- La méthode `empiler` qui prend un objet quelconque en paramètre et ajoute cet objet au sommet de la pile. Dans la représentation de la pile dans la console, cet objet apparaît à droite des autres éléments de la pile ;
- La méthode `depiler` qui renvoie l'objet présent au sommet de la pile et le retire de la pile.

Exemples :

```
>>> mapile = Pile()
>>> mapile.empiler(2)
>>> mapile
[2]
>>> mapile.empiler(3)
>>> mapile.empiler(50)
>>> mapile
[2, 3, 50]
>>> mapile.depiler()
50
>>> mapile
[2, 3]
```

1. La méthode `est_triee` ci-contre renvoie :

- True si, en dépilant tous les éléments, ils sont traités dans l'ordre croissant
- False sinon.

Compléter les points de suspension des lignes 6 et 8

```
1 def est_triee(self):
2     if not self.est_vide():
3         e1=self.depiler()
4         while not self.est_vide():
5             e2=self.depiler()
6             if e1 .... e2 :
7                 return False
8             e1=.....
9         return True
```

2. On crée dans la console la pile A représentée par [1, 2, 3, 4]

a. Donner la valeur renvoyée par l'appel `A.est_triee()` :

b. Donner le contenu de la pile A après l'exécution de cette instruction :

3. On souhaite maintenant écrire le code d'une méthode `depileMax` d'une pile non vide ne contenant que des nombres entiers et renvoyant le plus grand élément de cette pile en le retirant de la pile. Après l'exécution de `p.depileMax()`, le nombre d'éléments de la pile `p` diminue donc de 1.

```
1 def depileMax(self):
2     assert not self.est_vide(), "Pile Vide"
3     q=Pile()
4     maxi=self.depiler()
5     while not self.est_vide() :
6         elt=self.depiler()
7         if maxi<elt:
8             q.empiler(maxi)
9             maxi=.....
10        else:
11            .....
12        while not q.est_vide():
13            self.empiler(q.depiler())
14        return maxi
```

Compléter les lignes 9 et 11

4. On crée la pile B représentée par [9, -7, 8, 12, 4] et on effectue l'appel B.depileMax().
- a. Donner le contenu des piles B et q à la fin de chaque itération de la boucle while de la ligne 5.

b. Donner le contenu des piles B et q avant l'exécution de la ligne 14.

c. Donner un exemple de pile qui montre que l'ordre des éléments restants n'est pas préservé après l'exécution de depileMax.

5. On donne le code de la méthode traiter() :

```
1  def traiter(self):
2      q = Pile()
3      while not self.est_vide():
4          q.empiler(self.depileMax())
5      while not q.est_vide():
6          self.empiler(q.depiler())
```

- a. Donner les contenus successifs des piles B et q
- avant la ligne 3,
 - avant la ligne 5,
 - à la fin de l'exécution de la fonction traiter lorsque la fonction traiter est appliquée sur la pile B contenant [1, 6, 4, 3, 7, 2].
- -
 -

b. Expliquer le traitement effectué par cette méthode.