Tp - Piles et Files

Ce Tp reprend les exemples traités dans le poly de cours.

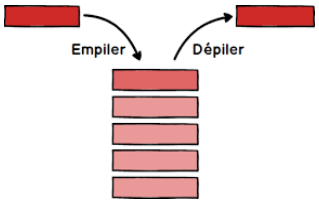
1- METHODE POP() DE LA CLASSE LIST DE PYTHON :

La méthode *pop()* prend en paramètre un entier i. Cette méthode permet de retirer de la liste l'élément dont l'index est i et de le renvoyer. Par défaut l'index i a comme valeur -1 ; il correspond donc au dernier élément de la liste.

⇨ Exécuter dans la console les lignes données ci-contre, et vérifier le bon fonctionnement de cette méthode *pop()*.

```
>>> l=[0,1,2,3,4]

>>> l.pop()
4

>>> l.pop(1)
1

>>> l
[0, 2, 3]
```

2- IMPLEMENTATION PYTHON D'UNE PILE :



⇨ Télécharger le fichier *pile.py* proposé sur nsibranly.fr .

Il contient le script incomplet de la classe *Pile* qui implémente une structure de donnée de type Pile. La méthode *__str__()* qui s'y trouve permet un affichage vertical du contenu de la pile. Le fichier contient aussi un programme principal test.

⇨ Compléter le script de cette classe *Pile* en définissant les méthodes empiler(), depiler(), estVide(), taille(), sommet(), vider() décrites dans le poly de cours.

⇨ Compléter le programme principal pour afficher dans la console les résultats des tests proposés dans le poly de cours. Coller un sreenshot dans un fichier .doc à enregistrer sous le nom *pileFile.doc*

```
class Pile:
    def __init__(self,nom = ""):
        self.nom = nom
        self.contenu = []

    def empiler(self, valeur):
        self.contenu.append(valeur)

    def estVide(self):
        return self.contenu == []

    def depiler(self):
        if not self.estVide():
            return self.contenu.pop()

    def taille(self):
        return len(self.contenu)

    def sommet(self) :
        if not self.estVide():
            return self.contenu[-1]

    def vider(self) :
        """p.contenu = []
           le plus facile, mais en faisant comme cela,
           on remplace l'ancienne liste non vide par une
           nouvelle liste vide.
        """
        for i in range(self.taille()) :
            del(self.contenu[-1])
```

3- UTILISATION DE CETTE CLASSE POUR TRAITER LE SUJET TYPE BAC (2019) :

Les réponses attendues dans ce paragraphe sont des copies d'écran à coller, toujours dans le fichier *pileFile.doc*.

⇨ Compléter la classe Pile en y écrivant le code de la méthode *maxPile()* de la question 3 , page 3 du poly de cours. Tester dans la console cette méthode avec la pile P donnée ci-contre, et avec comme argument le nombre 4. Copier les sreenshots de l'exécution dans la console de `>>> print(P)` et de `>>> P.maxPile(4)`.

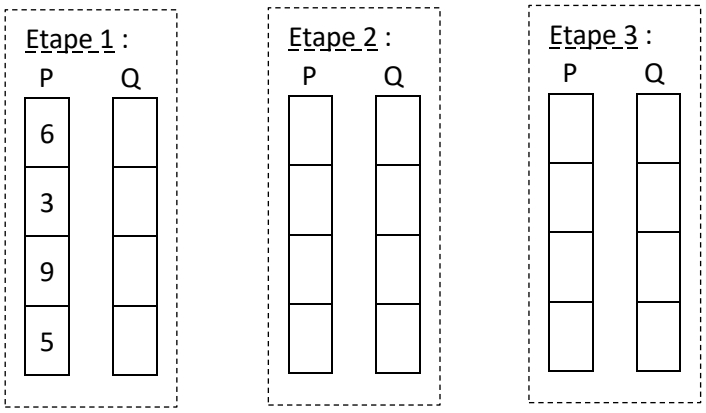
6
3
9
5

```
def maxPile(self,i) :
    Q = Pile('Q')
    j = 1
    while j <= i:
        x = self.depiler()
        Q.empiler(x)
        if j == 1 :
            max = x
            jMax = j
        elif x > max :
            max = x
            jMax = j
        j = j + 1
    while not Q.estVide():
        x = Q.depiler()
        self.empiler(x)
    return jMax
```

```
# Main
if __name__ == '__main__' :
    P = Pile("P")
    l = [5,9,3,6]
    for val in l :
        P.empiler(val)
    print(P)
```

```
>>> (executing file "exerciceBacPile.py")

Etat de la pile P :

| 6 |
| 3 |
| 9 |
| 5 |
-----

>>> P.maxPile(4)
3
```

Que se passe-t-il durant l'exécution de `P.maxPile(4)` ?



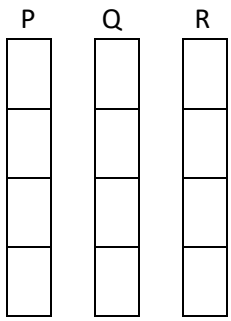
⇨ Compléter la classe en écrivant le code de la méthode *retourner()* de la question 4 , page 3 du poly de cours. Tester dans la console cette méthode avec la pile donnée ci-contre, en ayant comme argument le nombre 3. Copier les sreenshots de l'exécution dans la console de `>>> P.retourner(3)` et de `>>> print(P)` .

```
def retourner(self,j):
    Q = Pile('Q')
    R = Pile('R')
    n = 1
    while n <= j:
        x = self.depiler()
        Q.empiler(x)
        n = n + 1
    while not Q.estVide():
        x = Q.depiler()
        R.empiler(x)
    while not R.estVide():
        x = R.depiler()
        self.empiler(x)
```

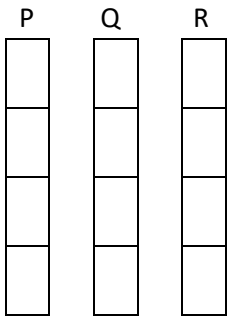
```
>>> (executing file "exerciceBacPile.py")

Etat de la pile P :

| 6 |
| 3 |
| 9 |
| 5 |
-----

>>> P.retourner(3)

>>> print(P)

Etat de la pile P :

| 9 |
| 3 |
| 6 |
| 5 |
-----
```

Compléter ci-dessous le contenu des piles P et Q au cours de l'exécution de `>>> P.retourner(3)` :

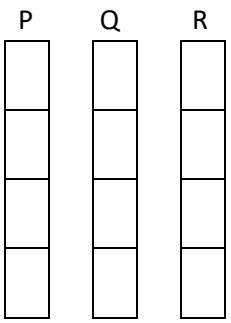
Etape 1 :



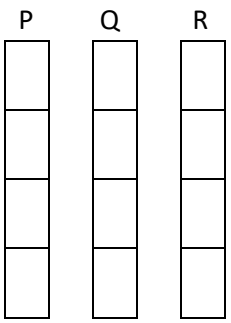
Etape 2 :



Etape 3 :



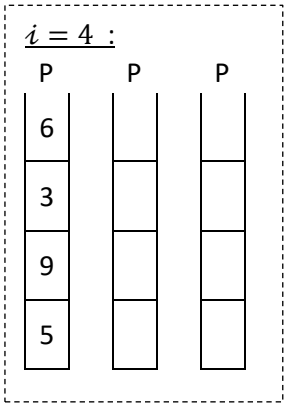
Etape 4 :

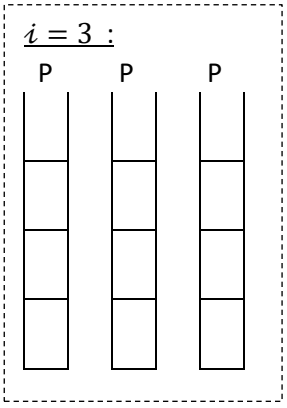


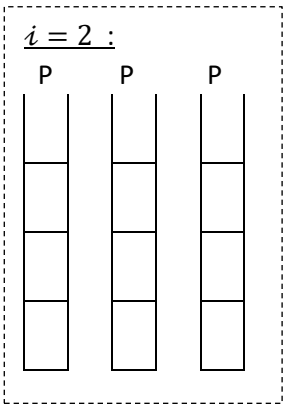
⇨ Compléter encore cette classe en écrivant le code de la méthode *triCrepes()* de la question 5 , page 4 du poly de cours. Tester dans la console cette méthode, toujours avec la pile donnée ci-dessus. Copier les sreenshots de l'exécution dans la console de `>>> P.triCrepes()` et de `>>> print(P)` .

```
def triCrepes(self) :
    i = self.taille()
    while i > 1 :
        j = self.maxPile(i)
        self.retourner(j)
        self.retourner(i)
        i = i - 1
```

```
>>> (executing file "exerciceBacPile.py")

Etat de la pile P :

| 6 |
| 3 |
| 9 |
| 5 |
-----

>>> P.triCrepes()

>>> print(P)

Etat de la pile P :

| 3 |
| 5 |
| 6 |
| 9 |
-----
```


$i = 4$:
P
P
P
6
3
9
5


$i = 3$:
P
P
P


$i = 2$:
P
P
P