DS 03/01/22 - Correction

Devoir n°5 : Les Piles et les Files	**Thème 1 : Structures de données**
	EVALUATION

	Exercice n°1	Exercice n°2
Barème	8 pts	8 pts

Exercice n°1 : Sujet BAC 2021

Cet exercice traite des notions de piles et de programmation orientée objet.

On crée une classe Pile qui modélise la structure d'une pile d'entiers.

Le constructeur de la classe initialise une pile vide.

La définition de cette classe sans l'implémentation de ses méthodes est donnée ci-dessous.

```
class Pile:
    def __init__(self):
        """Initialise la pile comme une pile vide."""

    def est_vide(self):
        """Renvoie True si la liste est vide, False sinon."""

    def empiler(self, e):
        """Ajoute l'élément e sur le sommet de la pile, ne renvoie rien."

    def depiler(self):
        """Retire l'élément au sommet de la pile et le renvoie."""
```

```
    def nb_elements(self):
        """Renvoie le nombre d'éléments de la pile. """

    def afficher(self):
        """Affiche de gauche à droite les éléments de la pile, du fond
        de la pile vers son sommet. Le sommet est alors l'élément
        affiché le plus à droite. Les éléments sont séparés par une
        virgule. Si la pile est vide la méthode affiche « pile
        vide »."""
```

Seules les méthodes de la classe ci-dessus doivent être utilisées pour manipuler les objets Pile.

1.a

Enoncé

Écrire une suite d'instructions permettant de créer une instance de la classe Pile affectée à une variable `pile1` contenant les éléments 7, 5 et 2 insérés dans cet ordre.
Ainsi, à l'issue de ces instructions, l'instruction `pile1.afficher()` produit l'affichage : 7, 5, 2.

Solution

```
pile1 = Pile()
pile1.empiler(7)
pile1.empiler(5)
pile1.empiler(2)
```

1.b

Enoncé

Donner l'affichage produit après l'exécution des instructions suivantes.

```
element1 = pile1.depiler()
pile1.empiler(5)
pile1.empiler(element1)
pile1.afficher()
```

Solution

7, 5, 5, 2

2.

Enoncé

On donne la fonction mystere suivante :

```
def mystere(pile, element):
    pile2 = Pile()
    nb_elements = pile.nb_elements()
    for i in range(nb_elements):
        elem = pile.depiler()
        pile2.empiler(elem)
        if elem == element:
            return pile2
    return pile2
```

a. Dans chacun des quatre cas suivants, quel est l'affichage obtenu dans la console ?

- Cas n°1

```
>>>pile.afficher()
7, 5, 2, 3
>>>mystere(pile, 2).afficher()
```

- Cas n°2

```
>>>pile.afficher()
7, 5, 2, 3
>>>mystere(pile, 9).afficher()
```

- Cas n°3

```
>>>pile.afficher()
7, 5, 2, 3
>>>mystere(pile, 3).afficher()
```

- Cas n°4

```
>>>pile.est_vide()
True
>>>mystere(pile, 3).afficher()
```

b. Expliquer ce que permet d'obtenir la fonction `mystere`.

Solution

2.a

- cas n°1 : 3, 2
- cas n°2 : 3, 2, 5, 7
- cas n°3 : 3
- cas n°4 : pile vide

2b La fonction mystere renvoie une pile qui contiendra tous les éléments de la pile passée en paramètre (pile) à condition qu'ils soient situés au-dessus de l'élément passé en paramètre (element). L'élément element sera lui aussi présent dans la pile renvoyée par la fonction.

3.

Enoncé

Écrire une fonction `etendre(pile1, pile2)` qui prend en arguments deux objets Pile appelés pile1 et pile2 et qui modifie pile1 en lui ajoutant les éléments de pile2 rangés dans l'ordre inverse. Cette fonction ne renvoie rien.
On donne ci-dessous les résultats attendus pour certaines instructions.

```
>>>pile1.afficher()
7, 5, 2, 3
>>>pile2.afficher()
1, 3, 4
>>>etendre(pile1, pile2)
>>>pile1.afficher()
7, 5, 2, 3, 4, 3, 1
>>>pile2.est_vide()
True
```

Solution

```
def etendre(pile1, pile2):
    while not pile2.est_vide():
        x = pile2.depiler()
        pile1.empiler(x)
```

4.

Enoncé

Écrire une fonction supprime_toutes_occurences(pile, element) qui prend en arguments un objet Pile appelé pile et un élément element et supprime tous les éléments element de pile.
On donne ci-dessous les résultats attendus pour certaines instructions.

```
>>>pile.afficher()
7, 5, 2, 3, 5
>>>supprime_toutes_occurences (pile, 5)
>>>pile.afficher()
7, 2, 3
```

Solution

```
def supprime_toutes_occurences(pile, element):
    p2 = Pile()
    while not pile.est_vide():
        x = pile.depiler()
        if x != element:
```

```
            p2.empiler(x)
    while not p2.est_vide():
        x = p2.depiler()
        pile.empiler(x)
```

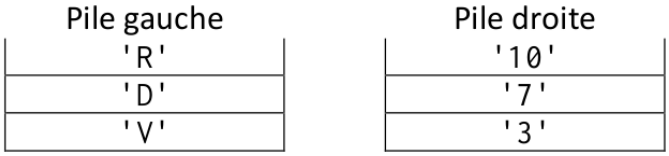
Exercice n°2 : Sujet BAC 2021

On cherche à obtenir un mélange d'une liste comportant un nombre pair d'éléments. Dans cet exercice, on notera N le nombre d'éléments de la liste à mélanger.

La méthode de mélange utilisée dans cette partie est inspirée d'un mélange de jeux de cartes :

- On sépare la liste en deux piles :
  - à gauche, la première pile contient les N/2 premiers éléments de la liste ;
  - à droite, la deuxième pile contient les N/2 derniers éléments de la liste.
- On crée une liste vide.
- On prend alors le sommet de la pile de gauche et on le met en début de liste.
- On prend ensuite le sommet de la pile de droite que l'on ajoute à la liste et ainsi de suite jusqu'à ce que les piles soient vides.

Par exemple, si on applique cette méthode de mélange à la liste `['V','D','R','3','7','10']`, on obtient pour le partage de la liste en 2 piles :



La nouvelle liste à la fin du mélange sera donc `['R','10','D','7','V','3']`

Question 1

Enoncé

Que devient la liste `['7','8','9','10','V','D','R','A']` si on lui applique cette méthode de mélange ?

Solution

Avec cette méthode de mélange, on obtient :
`['10', 'A', '9', 'R', '8', 'D', '7', 'V']`

On considère que l'on dispose de la structure de données de type pile, munie des seules instructions suivantes :

```
p = Pile() : crée une pile vide nommée p
p.est_vide() : renvoie Vrai si la liste est vide, Faux sinon
p.empiler(e) : ajoute l'élément e dans la pile
e = p.depiler() : retire le dernier élément ajouté dans la pile et le reto
p2 = p.copier() : renvoie une copie de la pile p sans modifier la pile p e
```

Question 2

Enoncé

Recopier et compléter le code de la fonction suivante qui transforme une liste en pile.

```
def liste_vers_pile(L):
    '''prend en paramètre une liste et renvoie une pile'''
    N = len(L)
    p_temp = Pile()
    for i in range(N):
        ...
    return ...
```

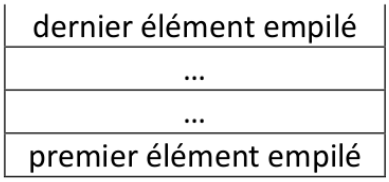
Solution

```
def liste_vers_pile(L):
    N = len(L)
    p_temp = Pile()
    for i in range(N):
        p_temp.empiler(L[i])
    return p_temp
```

Question 3

Enoncé

On considère la fonction suivante qui partage une liste en deux piles. Lors de sa mise au point et pour aider au débuggage, des appels à la fonction affichage_pile ont été insérés. La fonction affichage_pile(p) affiche la pile p à l'écran verticalement sous la forme suivante :




```
def partage(L):
    N = len(L)
    p_gauche = Pile()
    p_droite = Pile()
    for i in range(N/2):
        p_gauche.empile(L[i])
    for i in range(N/2,N):
        p_droite.empile(L[i])
    affichage_pile(p_gauche)
    affichage_pile(p_droite)
    return p_gauche, p_droite
```

Quels affichages obtient-on à l'écran lors de l'exécution de l'instruction :

```
partage([1,2,3,4,5,6])
```

?

Solution

Il y a des erreurs dans l'énoncé pour la fonction partage, voici la fonction partage corrigée :

```
def partage(L):
    N = len(L)

    p_gauche = Pile()
    p_droite = Pile()
    for i in range(N//2):
        p_gauche.empiler(L[i])
```

```
    for i in range(N//2,N):
        p_droite.empiler(L[i])
        affichage_pile(p_gauche)
        affichage_pile(p_droite)
    return p_gauche, p_droite
```

On obtient l'affichage suivant :

3
2
1

et

6
5
4

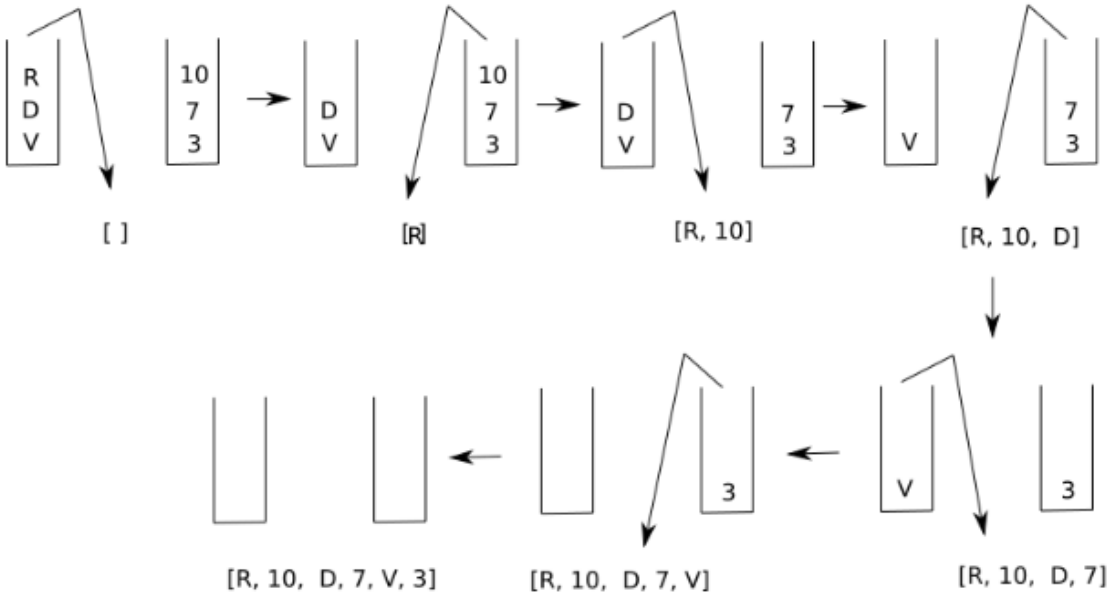
Question 4

Enoncé

4.a Dans un cas général et en vous appuyant sur une séquence de schémas, expliquer en quelques lignes comment fusionner deux piles p_gauche et p_droite pour former une liste L en alternant un à un les éléments de la pile p_gauche et de la pile p_droite.
4.b. Écrire une fonction fusion(p1,p2) qui renvoie une liste construite à partir des deux piles p1 et p2.

Solution

4.a



4.b

```
def fusion(p1, p2):
    L = []
    while not p1.est_vide():
        L.append(p1.depiler())
        L.append(p2.depiler())
    return L
```

Question 5

Enoncé

Compléter la dernière ligne du code de la fonction affichage_pile pour qu'elle fonctionne de manière récursive.

```
def affichage_pile(p):
    p_temp = p.copier()
    if p_temp.est_vide():
        print('____')
    else:
        elt = p_temp.depiler()
        print('| ', elt, ' |'))
        ...             # ligne à compléter
```

Solution

Attention dans le sujet il y a une parenthèse en trop au niveau de 2e print

```
def affichage_pile(p):
    p_temp = p.copier()
    if p_temp.est_vide():
        print('______')
    else:
        elt = p_temp.depiler()
        print('| ',elt,' |')
    affichage_pile(p_temp)
```