TP : Listes chainées en POO

La liste chainée est composée de maillons. Chaque maillon de la liste contient une valeur et pointe sur le maillon suivant. La liste chainée pointe sur le premier maillon de la chaine de maillons.

1 Le maillon

On donne l'implémentation du maillon en programmation orientée objet.

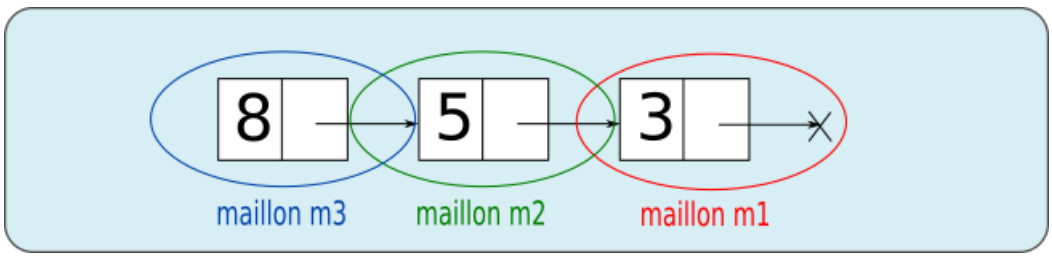
```python
class Maillon:
    def __init__(self, valeur=None, suivant=None):
        self.valeur = valeur
        self.suivant = suivant

    def __repr__(self):
        if self.suivant is None:
            return "["+str(self.valeur)+"]->" +"[]"
        else:
            return "["+str(self.valeur)+"]->" +str(self.suivant)
```

La classe Maillon définit 2 attributs et 2 méthodes :

— Les attributs valeur et suivant sont initialisés à la construction ou à None en l'absence de valeur.
— Les méthodes pour construire l'objet et pour l'afficher.

1) Les maillons ci-dessous sont reliés entre eux. Ils forment une chaine.



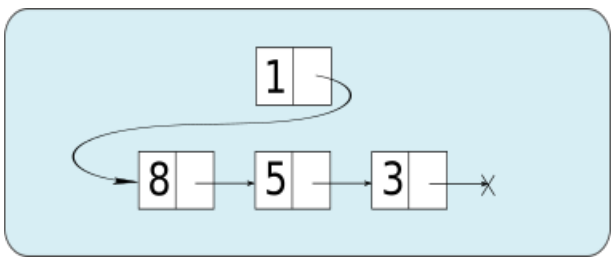
a) Construire un maillon m1 qui contient la valeur 3.

b) Construire le maillon m2 qui contient la valeur 5 et qui pointe sur le maillon m1.

c) Construire le maillon m3 qui contient la valeur 8 et pointe sur le maillon m2.

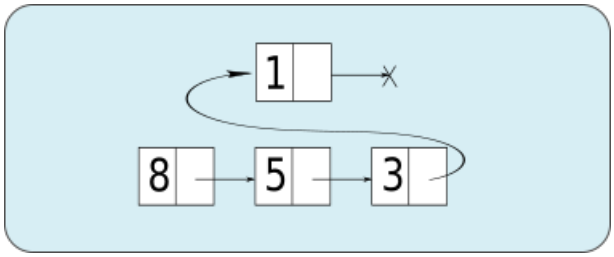
2) Créer un nouveau maillon m contenant la valeur 1. Nous allons relier ce maillon à la chaine de maillons précédente.

a) Le maillon m est placé au début de la chaine.



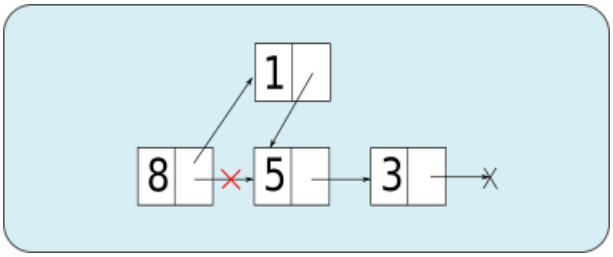
Écrire les instructions pour réaliser cette opération et afficher la chaine.

b) Le maillon m est placé à la fin de la chaine.



Écrire les instructions pour réaliser cette opération et afficher la chaine.

3) Le maillon m est placé dans la chaine entre la première et la deuxième valeur.



— Le maillon m pointe sur le second maillon de la chaine
— Le premier maillon m3 pointe sur le maillon m et plus sur le maillon m2.

a) Écrire les instructions pour réaliser cette opération et afficher la chaine.

b) Recommencer en plaçant le maillon m entre le deuxième et le troisième maillon de la chaine.

2 La liste chainée

Dans la partie précédente, nous avons vu qu'il est possible de lier des maillons sous forme de chaine.

On va créer un nouvel objet Liste qui a un attribut unique maillon. Cet attribut a pour valeur None si la liste est vide et un maillon si elle n'est pas vide.

La liste contient plusieurs méthodes répondant à l'interface de liste chainée.

— Le constructeur qui crée une liste vide.
— La méthode est_vide qui renvoie un booléen pour savoir si la liste est vide.
— La méthode tete qui renvoie la tête de la liste soit la première valeur.
— La méthode queue qui renvoie la queue de la liste, c'est à dire la liste sans la tête.
— La méthode inserer qui prend en paramètre un élément à insérer dans la liste et l'ajoute en tête de la liste.

On reprend le fichier contenant la classe Maillon et on crée la classe Liste juste en dessous.

1) Créer le constructeur de la classe Liste permettant de créer une liste chainée vide.

2) Ajouter la méthode est_vide qui renvoie un booléen. Une liste est vide si l'attribut maillon vaut None.

3) Ajouter la méthode tete qui renvoie la tête de la liste, c'est à dire la première valeur de la liste.

4) Ajouter la méthode **queue** qui renvoie une liste constituée des éléments de la liste chainée sans sa tête. Il est donc nécessaire de créer une nouvelle liste chainée queue et lui attribuer le bon maillon.

5) La méthode **inserer** prend en paramètre un élément et l'insère dans la liste. Pour y parvenir, il faut créer un maillon et l'insérer en tête de la liste.
On donne le code à compléter :

```
def inserer(self, element):
    # on crée un nouveau maillon avec la valeur element : [element]->[]
    ...
    # on pointe le maillon sur la liste (attribut maillon) : [element]->[*]->[*]->... comme L->[*]->[*]->...
    ...
    # on pointe la liste sur le nouveau maillon : L->[element]->...
    ...
```

3 Application

1) Créer une liste vide nommée maliste. Vérifier que maliste est vide puis l'afficher avec print.

2) Insérer dans la liste chainée maliste les voyelles a, e, i, o, u de notre alphabet. Afficher la liste.

3) La fonction **longueur_liste** prend en paramètre la liste et renvoie le nombre d'éléments qu'elle contient soit sa longueur. L'algorithme suivant peut vous guider dans son écriture :

on crée la variable longueur à 0
tant que le suivant de liste n'est pas vide :
la longueur augmente de 1
on passe au suivant
on renvoie la longueur

Écrire le code de la fonction en python.

4) La fonction **supprime_tete** prend en paramètre la liste. Pour supprimer la tete, il suffit :

- de vérifier si la liste est non vide. Si oui, alors la queue de liste devient la liste.
- on renvoie la liste

4 Implémenter une pile et une file en POO

1 Pile

On va créer une classe Pile pour implémenter une pile. On rappelle l'interface de la pile :

— crée_pile() qui crée une pile vide ;
— empile(valeur,pile) qui ajoute une valeur au sommet de la pile ;
— dépile(valeur,pile) qui supprime le sommet de la pile en renvoyant sa valeur ;
— pile_vide(pile) qui teste si la pile est vide.

L'implémentation est proche de celle de la liste chainée. Il suffit de la reprendre et de la modifier en respectant l'interface de la pile.

1) Implémentation

a) Créer la classe Pile. Le constructeur de la classe renvoie une pile vide ce qui correspond à la fonction crée_pile(). L'attribut sera nommé **sommet**.

b) Créer les 3 autres méthodes : empile, dépile et pile_vide.

2) Application :

a) Créer une pile p puis empiler plusieurs valeurs. Afficher la pile p.

b) Créer la fonction carres(), qui prend en paramètre une pile et renvoie une nouvelle pile avec les carrés des nombres de la pile passée en paramètre.

c) Créer la pile pcarré contenant les carrés des nombres de la pile p. Afficher les piles p et pcarré.

2 File

On va créer une classe File pour implémenter une file. On rappelle l'interface de la file :

— crée_file() qui crée une file vide ;
— enfile(valeur,file) qui ajoute une valeur à la file ;
— défile(file) qui supprime la tête de la file en renvoyant sa valeur ;
— file_vide(pile) qui teste si la file est vide.

La aussi, l'implémentation est proche de celle de la liste chainée. Il suffit de la reprendre et de la modifier en respectant l'interface de la file.

1) Implémentation

a) Créer la classe File. Le constructeur de la classe renvoie une file vide ce qui correspond à la fonction crée_file(). L'attribut sera nommé **queue**.

b) Créer les 3 autres méthodes : enfile, défile et file_vide.

2) Application :

a) Créer une pile f puis enfiler plusieurs valeurs. Afficher la file f.

b) Créer la fonction impairs(), qui prend en paramètre une file et renvoie une nouvelle file avec les nombres impairs contenus dans la file passée en paramètre.

c) Créer la file fimpairs contenant les nombres impairs de la file p. Afficher les files f et fimpairs.

3 En supplément (difficile)

1) Ajouter les méthodes **hauteur** pour la pile et **longueur** pour la file qui renvoient le nombre d'éléments.

2) Améliorer l'affichage pour la pile et la file, par exemple pour qu'il ressemble à celui des listes en python.

3) L'accès aux valeurs d'une pile ou d'une file n'est pas possible sauf pour le sommet et la tête bien entendu. Comment modifier les classes Pile et File pour que les valeurs ne soient pas accessibles.