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.

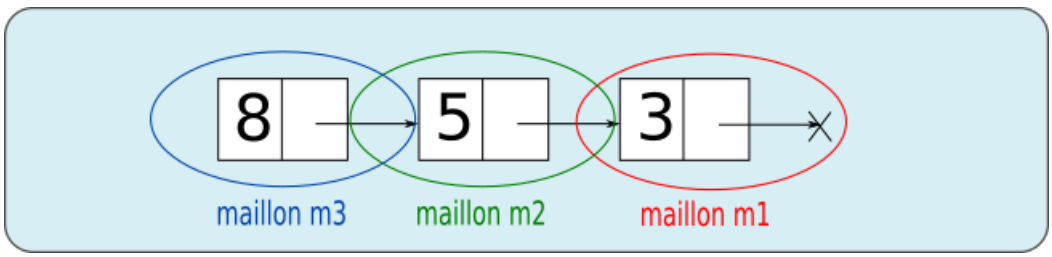
```python
class Maillon:
    def __init__(self, valeur=None, suivant=None):
        self.valeur = valeur
        self.suivant = suivant

    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.



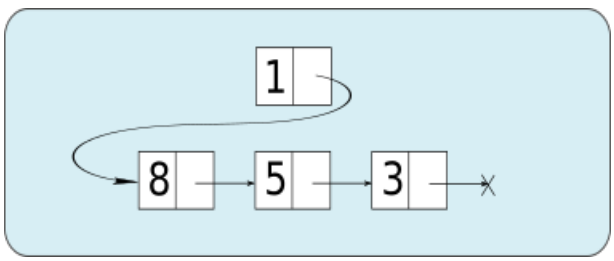
a) Construire un maillon m1 qui contient la valeur 3.

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.

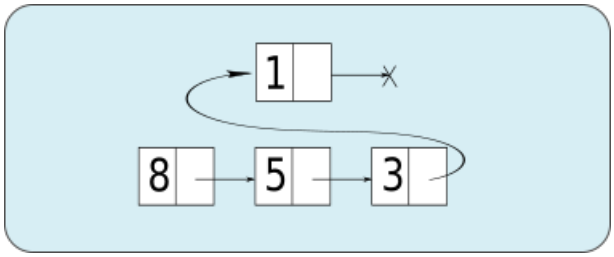
2) Créer un nouveau maillon m contenant la valeur 1. Nous allons relier ce maillon à la chaine de maillons précédente.

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



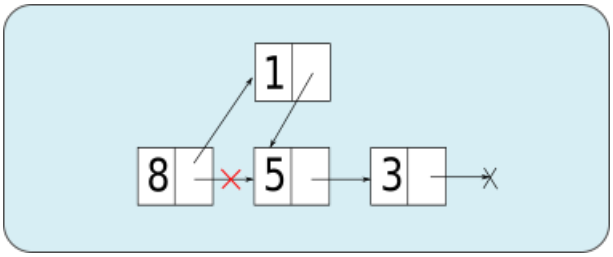
Écrire les instructions pour réaliser cette opération et afficher la chaine.

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) Écrire la fonction creer_liste() sans paramètre qui renvoie une liste chainée vide.

2) Créer une liste vide nommée maliste. Vérifier que maliste est vide avec la bonne méthode.

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

4) Écrire la fonction parcourir qui prend en paramètre une liste chainée. Cette fonction affiche un à un les éléments de la liste chainée et ne renvoie rien.
On suivra l'algorithme ci-dessous :

- Il faut, dans la fonction, définir une variable locale maillon qui prend la valeur du premier maillon de la liste chainée.
- Tant que la variable locale maillon n'est pas vide, on affiche la valeur et on passe au maillon suivant.

5) 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.

6) La fonction get_item prend en paramètre une liste chainée et un nombre entier i. Cette fonction renvoie la valeur de la liste dont l'index (position) est i. Par exemple, si i=0, on renvoie la première valeur.
Quelques remarques :

- On s'assure que l'argument i a des valeurs valides.
- Comme pour la fonction parcourir, il faut une variable locale maillon qui parcourt la liste.

7) Écrire la fonction renverse qui renverse la liste chainée. La première valeur devient la dernière, etc. Elle prend en argument une liste chainée et renvoie une liste chainée avec les éléments en ordre inverse.

8) La fonction ajouter_fin prend en paramètre une liste chainée et un élément à insérer. La fonction renvoie la liste chainée avec l'élément ajouté en fin de liste.
On suivra l'algorithme ci-dessous :

- Il faut parcourir la liste chainée jusqu'au dernier maillon de la chaine.
- On fait pointer ce dernier maillon sur un nouveau maillon qui contient l'élément passé en argument.

9) Si la liste chainée est longue, l'insertion en fin de liste nécessite de la parcourir entièrement pour insérer un nouveau maillon. Ceci n'est donc pas très optimal.

Une solution consiste à ajouter un attribut `last` à la classe `Liste` qui contient le dernier maillon de la liste chainée. Ainsi, quand on ajoute une valeur en fin de liste, il suffit de faire pointer le maillon `last` sur le maillon à insérer et d'actualiser l'attribut `last`.

Apporter les modifications nécessaires à la classe `Liste` et récrire la fonction `ajouter_fin` en utilisant l'attribut `last`.