TD: listes chainées:

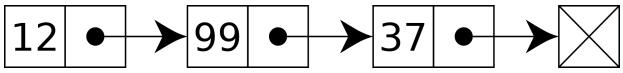

Figure 1: Schématisation d'une liste chainée

Exercice 1: On s'interesse au programme suivant:

```
"""
Une premiere classe pour les maillons de la liste chainee:
"""
class Maillon:
    def __init__(self, valeur = None, suivant = None):
        self.val = valeur
        self.suiv = suivant

    def __str__(self):
        if self.val is not None:
            return str(self.val) + " - " + str(self.suiv)
        else:
            return str(self.val)

"""
Une seconde classe pour la liste chainee:
"""
class Liste_Chaine:
    def __init__(self, premier = None):
        self.prem = Maillon(premier)    #on initialise avec un seul element qui
    contient None

    def ajoute(self, valeur):           #ajout d'un element en fin de liste
        m = Maillon(valeur)             #creation d'un maillon
        if self.prem.val is None:
            self.prem = m               #si la liste est vide, c'est le premier
        else:
            p = self.prem               #sinon, on recherche le dernier element et on
    ajoute m
            while p.suiv is not None:
                p = p.suiv
            p.suiv = m

    def __str__(self):
        return str(self.prem)
```

liste_chainee1.py

1. Expliquer les attributs et les méthodes de la classe Maillon.
2. Expliquer les attributs et les méthodes de la classe Liste_Chaine.
3. A l'aide de l'éditeur, créer une liste chainée contenant les éléments S, A puis T et l'afficher. Expliquer le résultat de l'affichage.

On ajoute les méthodes suivantes:

```
    def tete(self):                          #retourne le premier element de la liste
        if self.prem is not None:
            return self.prem.val

    def queue(self):                         #retourne la liste obtenue apres le
retrait du premier element
        if self.prem.val is None:
            return None
        else:
            liste = Liste_Chaine()
            liste.prem = self.prem.suiv
            return liste
```

liste_chainee2.py

4. Expliquer avec précision le rôle de ces nouvelles méthodes.
5. Comment les tester dans l'éditeur Python?

On ajoute la méthode suivante:

```
    def insere(self,valeur):                 #insere un element en tete de liste
        m = Maillon(valeur)
        if self.prem.val is None:
            self.prem = m
        else:                                #p est le premier element, m le devient
pios p devient le suivant de m
            p = self.prem
            self.prem = m
            m.suiv = p
```

liste_chainee4.py

6. Expliquer le rôle de cette nouvelle méthode.
7. La tester.

Exercice 2: Concaténation:
Compléter la classe Liste_Chaine avec une méthode concat qui permet de concaténer une deuxième liste à la liste initiale. Par exemple:

- si l1 contient 13, 14 et 15
- si l2 contient A, B, C

alors l1.concat(l2) contiendra 13, 14, 15, A , B, C

Exercice 3: Changement de type:

1. Écrire les classes Maillon et Liste_Chainee dans un fichier nommé liste_chaine.py.
2. Écrire dans un second fichier une fonction qui prend en paramètre une liste de type list et qui renvoie une liste chainée. Le module liste_chaine sera importé à l'aide de l'instruction import liste_chaine as lc.
3. Écrire une fonction qui prend en paramètre une liste chainée et qui renvoie une liste de type list.