

Exercice 1 : Listes chaînées - nouvelles fonctionnalités

Reprendre la classe *Liste* vue en cours et ajouter les méthodes ci-après :

1. `__str__` s'appelle directement : `print(nom_objet)`.

```

1  def afficher_rec(self, maillon: object)->str:
2      """
3      crée la chaîne d'affichage pour __str__
4      """
5      if maillon is None:
6          return "\n" # renvoi à la ligne
7      else:
8          return str(maillon.valeur)+" "+self.afficher_rec(
9              maillon.suivant)
10
11 def __str__(self):
12     return self.afficher_rec(self.tete)

```

2. Version impérative.

```

1  def renverser(self)->None:
2      res = None
3      maillon_en_cours = self.tete
4      while maillon_en_cours is not None:
5          res = Maillon(maillon_en_cours.valeur, res)
6          maillon_en_cours = maillon_en_cours.suivant
7      # la liste est retournée on récupère la tête
8      self.tete = res

```

3. S'appuyer sur le schéma

```

1  def concatener_rec(self, tete1: object, tete2: object)->
2      object:
3      """
4      méthode interne pour additionner 2 listes
5      """
6      if tete1 is None:
7          return tete2
8      else:
9          return Maillon(tete1.valeur, self.concatener_rec(
10              tete1.suivant, tete2))
11
12 def concatener(self, l: object)->object:
13     """
14     appel principal de la méthode récursive pour
15     additionner 2 listes
16     """
17     res = Liste()
18     res.tete = self.concatener_rec(self.tete, l.tete)
19     return res

```

4. La complexité dépend de la taille de la première liste mais pas du tout de celle de la seconde.

Exercice 2 :

```
1  def inserer_rec(self, val: int, lst: object)->object:
2      """
3      méthode interne pour recréer la liste avec l'élément insér
4      é
5      """
6      if lst is None or val <= lst.valeur:
7          return Maillon(val, lst)
8      else:
9          return Maillon(lst.valeur, self.inserer_rec(val, lst.
10             suivant))
11
12 def inserer(self, val: int)->None:
13     """
14     appel principal pour l'insertion dans une liste triée
15     """
16     self.tete = self.inserer_rec(val, self.tete)
```