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).

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

2. Version impérative.

```
def renverser(self)->None:
    res = None
    maillon_en_cours = self.tete
    while maillon_en_cours is not None:
        res = Maillon(maillon_en_cours.valeur, res)
        maillon_en_cours = maillon_en_cours.suivant
    # la liste est retournée on récupère la tête
    self.tete = res
```

3. S'appuyer sur le schéma

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

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



## Exercice 2:

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

