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:
        """
        crée la chaîne d'affichage pour __str__
        """
        if maillon is None:
            return "\n" # renvoi àla ligne
        else:
            return str(maillon.valeur)+" "+self.afficher_rec(
                maillon.suivant)

    def __str__(self):
        return self.afficher_rec(self.tete)
```

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 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
```

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

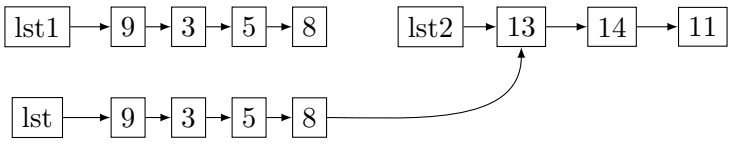

FIGURE 1 – Fonctionnement de **concatener_rec**

Christophe Viroulaud