Exercices listes chaînées
Correction

Christophe Viroulaud

Terminale - NSI

Archi 04

Sommaire

Exercice 1 - 1

```
1  class Maillon:
2      """
3      Crée un maillon de la liste chaînée
4      """
5
6      def __init__(self, val: int, suiv: object) ->
       None:
7          self.valeur = val
8          self.suivant = suiv
```

Exercice 1 - 2


```
1  class Liste:
2      """
3      Crée une liste chaînée
4      """
5
6      def __init__(self):
7          self.tete: Maillon = None
```

Exercice 1 - 3

```
1 def ajoute(self, val: int) -> None:
2     self.tete = Maillon(val, self.tete)
```

Exercice 1 - 4

```
1 lst = Liste()
2 lst.ajoute(8)
3 lst.ajoute(5)
4 lst.ajoute(3)
5 lst.ajoute(9)
6 lst.ajoute(10)
```

Exercice 1 - 5

```python
def dernier(self) -> int:
    # gestion d'erreur
    if self.tete is None:
        return -1

    en_cours = self.tete
    while en_cours.suivant is not None:
        en_cours = en_cours.suivant
    return en_cours.valeur
```

Exercice 1 - 6

```python
def dernier_aux(self, m: Maillon) -> int:
    if m.suivant is None:
        return m.valeur
    else:
        return self.dernier_aux(m.suivant)

def dernier_rec(self) -> int:
    # gestion d'erreur
    if self.tete is None:
        return -1

    return self.dernier_aux(self.tete)
```

Exercice 1 - 7

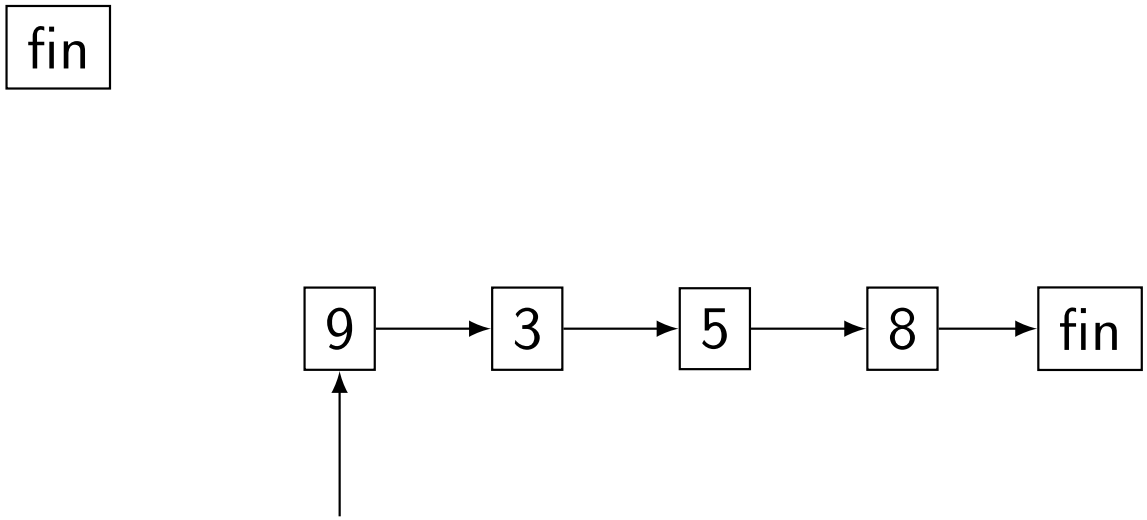

FIGURE 1 – Tant que le `Maillon` en cours n'est pas `None`

Exercice 1 - 7

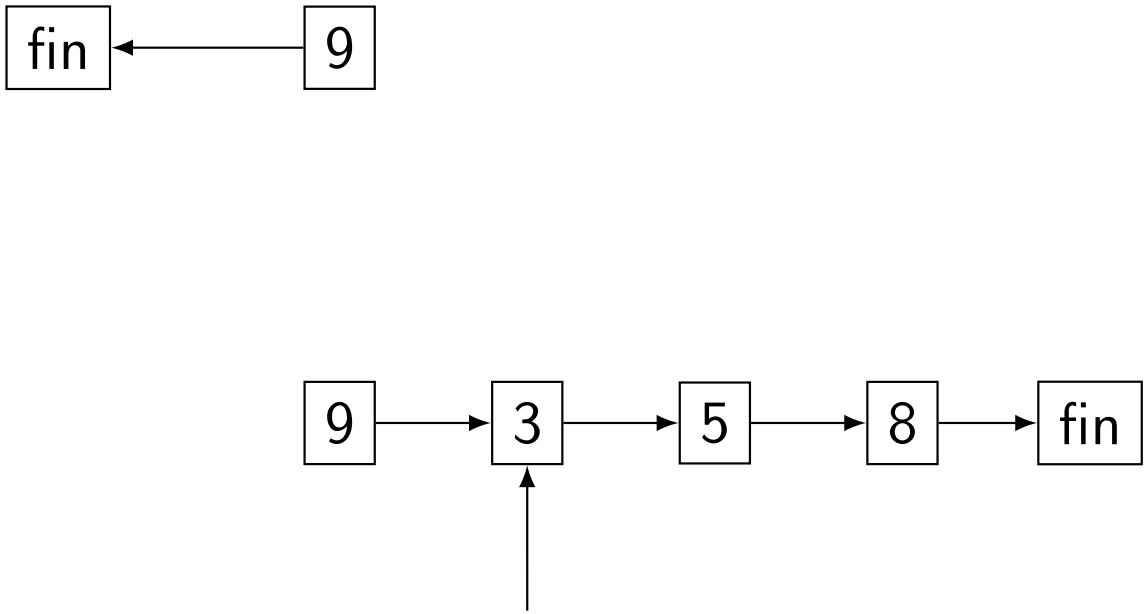

FIGURE 2 – Créer un nouveau **`Maillon`** et l'attacher au précédent.

Exercice 1 - 8

```
1  def renverser(self) -> None:
2      res = None
3      en_cours = self.tete
4      while en_cours is not None:
5          # crée maillon et l'attache au précé
       dant
6          res = Maillon(en_cours.valeur, res)
7          # va voir le maillon suivant
8          en_cours = en_cours.suivant
9      self.tete = res
```

Exercice 1 - 9

```
1  def dupliquer(self):
2      en_cours = self.tete
3      while en_cours is not None:
4          doublon = Maillon(en_cours.valeur,
       en_cours.suivant)
5          en_cours.suivant = doublon
6          en_cours = doublon.suivant
```

Sommaire

Exercice 2 - 1

```
1 from liste import Liste, Maillon
```

Exercice 2 - 2

```
1  l1 = Liste()
2  l1.ajoute(8)
3  l1.ajoute(5)
4  l1.ajoute(3)
5  l1.ajoute(9)
6  l1.ajoute(10)
7
8  l2 = Liste()
9  l2.ajoute(4)
10 l2.ajoute(0)
11 l2.ajoute(2)
```

Exercice 2 - 3

```
1  def concatener(l1: Liste, l2: Liste) -> Liste:
2      def concatener_rec(tete1: Maillon, tete2: Maillon) -> Maillon:
3          """
4          fonction interne pour additionner 2 listes
5          """
6          if tete1 is None:
7              return tete2
8          else:
9              return Maillon(tete1.valeur, concatener_rec(tete1.suivant, tete2))
10
11     res = Liste()
12     res.tete = concatener_rec(l1.tete, l2.tete)
13     return res
```

Exercice 2 - 4

La liste 2 n'est pas recopiée dans la nouvelle liste. Une modification du contenu de l'une d'elle modifie l'autre : c'est un **effet de bord**.

Exercice 2 - 5

Le complexité dépend de la taille de la première liste, qui est entièrement copiée.

Sommaire

Exercice 3 - 1

```
1  def longueur(lst: tuple) -> int:
2      if len(lst) == 0:
3          return 0
4      else:
5          return 1+longueur(lst[1])
```

Exercice 3 - 2

```python
def afficher(lst: tuple) -> str:
    if len(lst) == 0:
        return "fin"
    else:
        return lst[0] + " - " + afficher(lst[1])
```