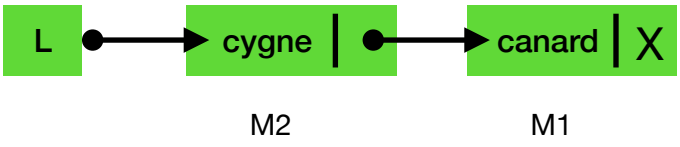
Cours 19 novembre — Résumé de la séance

1) Retour sur les listes chainées — correction de l'exercice du cours — 20 minutes

Exercice :
On considère la liste d'opérations ci-dessous :

```
L = ListeC()
M1, M2 = Maillon(), Maillon()
M2.suivant = M1
M1.valeur = 'canard'
M2.valeur = 'cygne'
L.tete = M2
```

a) Dessinez la liste chainée obtenue.



Pour construire le schéma, on lit :
Le suivant du maillon 2 est le maillon 1.
La tête de la liste est le maillon 2.

b) Quelle instruction devez-vous utiliser pour afficher "cygne" ? **L.tete.valeur** car cygne est la valeur du maillon en tête de liste.
pour afficher le Maillon M1 ? **L.tete.suivant** car le maillon M1 est le maillon suivant la tête.
pour afficher "canard" ? Il faut demander la valeur du maillon suivant la tête soit :
L.tete.suivant.valeur

c) Quelles instructions devez-vous écrire pour ajouter un troisième maillon dont la valeur est "poule d'eau" à votre liste chainée L ?

```
M3 = Maillon()
M1.suivant = M3
M3.valeur = "poule d'eau"
```

Bizarrement, nous n'avons rien à ajouter à la liste car on vient de faire un lien entre le maillon n°1 et le maillon n°3. Toutefois, dans de nombreux cas, nous ne connaitrons pas le maillon final. Dans ce cas, il faudra utiliser :

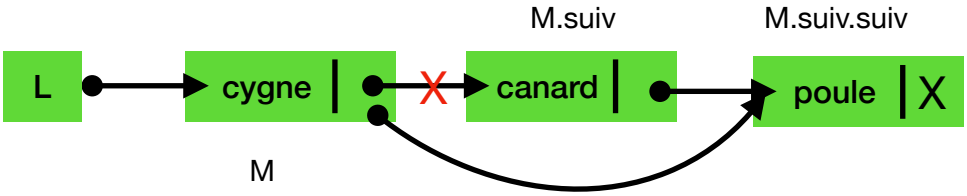
```
M3 = Maillon()
M = L.tete   # M.valeur est cygne

M3.valeur = "poule d'eau"
M = M.suivant  # M.valeur est canard, M.suivant vaut None pour
l'instant
# None est la fin de la liste : on insère notre poule d'eau ici..
M.suivant = M3
```

d) Finalement, vous souhaitez supprimer le 'canard'. Comment faire ?

On va "rerouter" la liste et court-circuiter le canard qui est le suivant de la tête. On appelle M le maillon en tête. On va donc dire que le suivant du maillon M devient le suivant du suivant.

```
M = L.tete
M.suivant = M.suivant.suivant
```



2) TP autour de la notion de liste chainée

Je vous conseille chaudement de vous munir d'une feuille de papier et d'un crayon afin de pouvoir faire un dessin comme ci-dessus. Lorsque l'on connait le nom des maillons, les choses sont simples mais lorsque l'on ne les connait pas, c'est plus complexe !

L'algorithme de base est celui de la taille de la liste chainée. On ne peut pas utiliser **len** car Python ne saura pas ce qu'est la longueur de notre objet L, créé en POO.

Dans l'algo présenté sur mon site web, il y a plusieurs étapes :

```
def tailleListe(L):
    """ Renvoie le nombre de Maillons de la liste L """
```

Il faut évacuer le problème de L.tete. En effet, la tête n'est présente que pour le premier maillon. Cela se prête assez mal à la réalisation d'une boucle où l'on répète plusieurs fois la même opération.

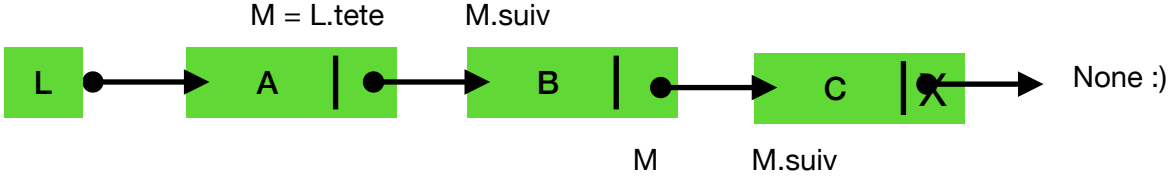
Ainsi, la tete est notre premier maillon :

```
    m = L.tete
```

On compte le nombre de Maillon en vérifiant à chaque étape si le maillon actuel est **None** :

```
   while m is not None:
        m = m.suivant   # le maillon à étudier est le suivant du
maillon actuel.
```

On se décale ensuite afin de considérer le Maillon suivant et voir si celui-ci est **None** ou non.



Les autres algorithmes sont relativement similaires. Je vous suggère d'essayer de les programmer, en vous ramenant toujours à l'algorithme pour la taille.

Les méthodes, intégrées à la classe ListeC, sont détaillées dans le programme Python.