Les Listes en Python

1. Définir une liste

On définit simplement une liste en python à l'aides des crochets [et] :

```
>>> a=[]
>>> type(a)
<class 'list'>
>>> a=[1,2]
>>> type(a)
<class 'list'>
>>> a
[1, 2]
```

2. Ajouter un élément

```
>>> a=[1,2,3]
>>> a=a+[4]
>>> a
[1, 2, 3, 4]
>>> a.append(5)
>>> a
[1, 2, 3, 4, 5]
```

3. Accèder à un élément

On accède aux éléments d'une liste par leur indice :

```
>>> a=[1,2,3,4,5]
>>> a[0]
1
>>> a[1]
2
>>> a[len(a)-1]
5
>>> a[-1]
5
```

Les listes sont **itérables** comme les tuples et chaînes de caractères :

```
>>> a=[1,8,3,12]
>>> for i in a:
    print(i)

```

```
1
8
3
12
>>> for i in range(3):
    print(a[i])

1
8
3
```

4. Supprimer un élément ou des éléments

On supprime des éléments via leurs indices. On peut utiliser les méthodes del ou pop pour cela.

```
>>> a=[1,2,3,4,5]
>>> del a[2]
>>> a
[1, 2, 4, 5]
>>> a.pop(0) # renvoie la valeur supprimée
1
>>> a
[2, 4, 5]
>>> a.pop() # supprimer et renvoie la dernière valeur
5
>>> a
[2, 4]
```

La méthode del permet de supprimer plusieurs éléments à la fois :

```
>>> a=[1,2,3,4,5]
>>> del a[1:3]
>>> a
[1, 4, 5]
```

5. Modifier un élément

Les listes en python sont **mutables** ce qui les différencie des tuples et des chaînes de caractères :

```
>>> a=[1,2,3]
>>> a[0]=12
>>> a
[12, 2, 3]
>>> a[0],a[2]=a[2],a[0]
>>> a
[3, 2, 12]
```

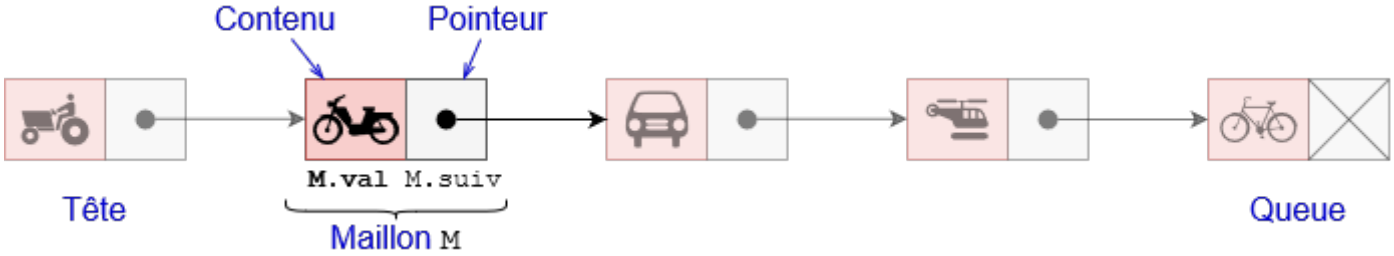
6.Quelques méthodes utiles

```
>>> a=[1,2,3,3,1]
>>> len(a)
5
>>> max(a)
3
>>> min(a)
1
>>> sorted(a) # trie par ordre croissant la liste
[1, 1, 2, 3, 3]
>>> 2 in a
True
>>> 8 in a
False
```

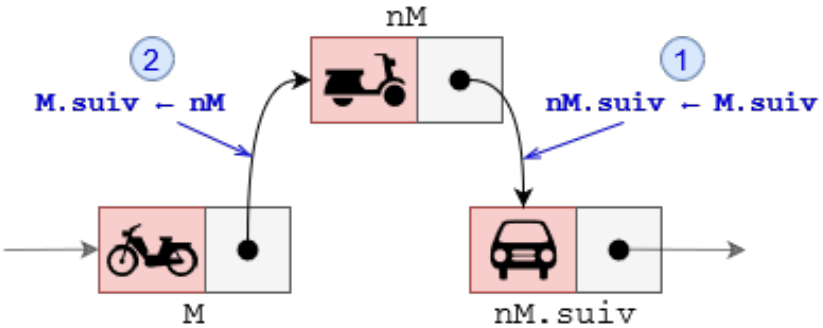
7.Les listes en python ne sont pas des listes mais des tableaux.

a) Principe des listes chainées.

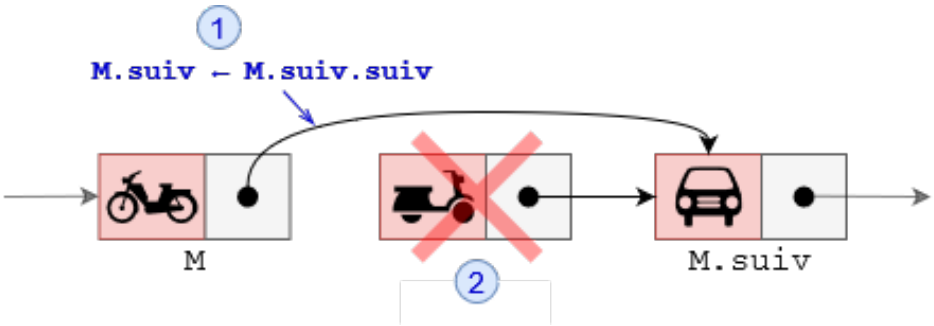
Les listes simplement chainées sont une suite ordonnées de valeurs, chaque maillon à l'exception du dernier possède une valeur et un pointeur vers l'adresse mémoire du prochain maillon.



Pour insérer un maillon, il n'est alors nécessaire d'opérer des modifications que sur deux maillons de la chaîne :

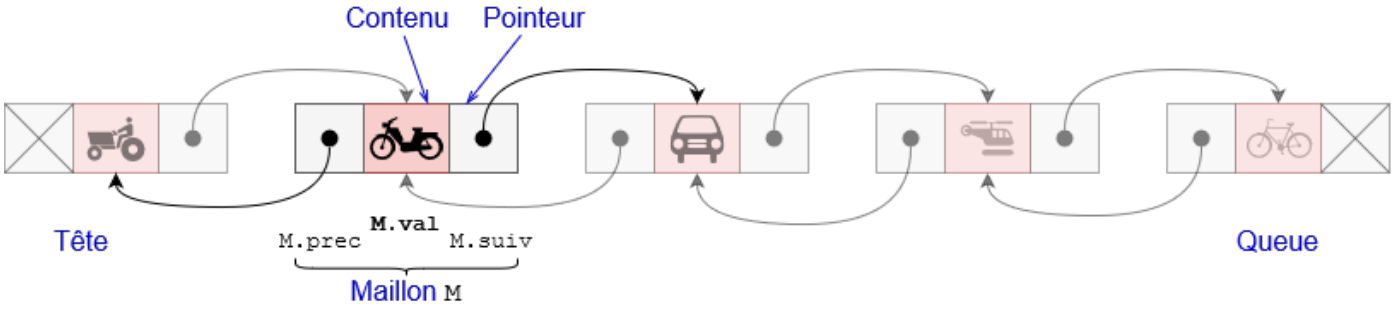


Pour supprimer un maillon, une seule modification est nécessaire.



La chaîne est alors parfaitement définie par son premier maillon.

On peut avoir des listes doublement chainées :



Problèmes d'accès en mémoire aux valeurs d'une liste chainée :

si l'on considère une liste chainée de 100 éléments, pour avoir accès au 50eme maillon , il faut parcourir .....

b) Structure en mémoire des listes en python avec python tutor.

Les listes en python sont en fait des **tableaux**.

L'accès en mémoire des éléments d'un tableau est plus rapide que pour une liste (on n'est pas obligé de parcourir un tableau contrairement à une liste).

L'insertion d'éléments est par contre plus coûteux pour un tableau que pour une liste (on est obligé de copier et de décaler un ensemble de valeurs).

Particularité de l'égalité des listes avec python :

Python 3.6
(known limitations)

```
1  a=[1,2,12,18,7]
2  b=a
3  a[0]="changement"
4  print(a)
5  print(b)
```

Edit this code

Print output (drag lower right corner to resize)

```
['changement', 2, 12, 18, 7]
['changement', 2, 12, 18, 7]
```

Frames

Global frame

a	id1
b	id1

Objects

id1:list

0	1	2	3	4
"changement"	2	12	18	7

8. Génération des listes par compréhension.

```
1  >>> [i for i in range(5)]
2  [0, 1, 2, 3, 4]
3  >>> [i**2 for i in range(5)]
4  [0, 1, 4, 9, 16]
5  >>> [i*j for i in range(5) for j in range(2)]
6  [0, 0, 0, 1, 0, 2, 0, 3, 0, 4]
7
```