Liste chainée, pile et file

Thème 1 - Partie 3

Liste chainée

• Histoire

Qui sont les concepteurs du principe de liste chaînée ?

. . . . . .

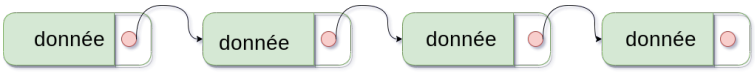
. . . . . .

. . . . . .

• Definition

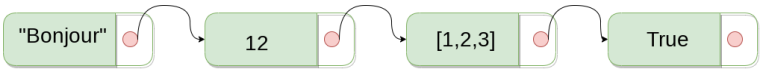
Définition `Liste chainée`

Une liste chainée est une structure permettant d'implémenter une liste, c'est-à-dire une séquence finie de valeurs (de même type ou non). Les éléments sont dits chainés car à chaque élément est associé l'adresse mémoire de l'élément suivant de la liste.



Remarque

Les données contenu dans la chaîne peuvent être de tout type :



Définition `Maillon`

Les éléments d'une liste chainée sont appelés **maillon** - noté `M` dans ce cours - de la liste. Il sont composés :

- d'un contenu utile `M.valeur` (de n'importe quel type),
- d'un pointeur `M.suivant` pointant vers l'élément suivant de la séquence.

Exercice 1 Tête & Queue

Les listes chainée sont composées d'une **`tête et d'une queue`**

1. Parmis ces définitions, laquelle corresond à la notion de **`tête`** d'une liste chainée :

   ☐ le dernier maillon ☐ le premier maillon ☐ l'ensemble de tous les maillons

2. Parmis ces définitions, laquelle corresond à la notion de **`queue`** d'une liste chainée :

   ☐ le dernier maillon ☐ l'ensemble de tous les maillons ☐ l'ensemble de tous les maillons après la tête.

Interface

Il n'existe pas norme mais l'on retrouve fréquement les instructions suivantes :

- `est_vide()` : renvoie vrai si la liste L est vide,
- `taille()` : renvoie le nombre d'éléments de la liste L,
- `get_dernier_maillon()` : renvoie le dernier élément de la liste,
- `get_maillon_indice(i)` : renvoie le maillon d'indice i ,
- `ajouter_debut(m)` : ajoute un maillon `m` au début de la liste L,
- `ajouter_fin(m)` : ajoute un maillon `m` à la fin de la liste L,
- `inserer(v, i)` : insert un maillon `m` à l'indice `i`,
- `supprimer(i)` : supprime le maillon à l'indice `i`,

Implémentation

i Pour implémenter les listes chainées, on implémentera la classe `Maillon`, on considérera que l'instanciation d'un maillon crée une liste chainée dont il est la tête.

Exemple

En suppossant que `Maillon` soit une classe qui implémente un maillon. On considère le code ci-dessous :

```
1 m = Maillon(3, Maillon(4, Maillon(5, Maillon(6, None))))
```

Le code ci-dessus instanciera une liste chainée, nommée `m`, représentée par :

Exercice 2 ★

On considère le code ci-dessous :

```
1 lc = Maillon("N", Maillon("S", Maillon("I", None)))
```

1. Réprésenter la liste chainée instancier par le code ci-dessous :

   .....

   .....

2. Quel est la valeur de la tête de cette liste ?

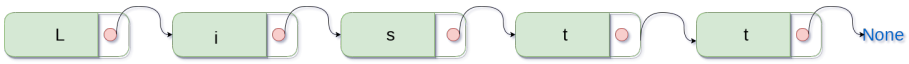
   .....

3. Représenter la queue de cette liste.

   .....

Exercice 3 ★

Quelle instruction permet de créer la liste chainée, nommé `l`, ci-dessous :



. . . . . .

. . . . . .

. . . . . .

• La classe Maillon

Attributs :

- `valeur` : le contenu du maillon
- `suivant` : un objet de type Maillon ou None

Méthodes :

- `__init__` : constructeur qui prend un paramètre `valeur` la valeur du maillon et `suivant` le maillon suivant ou `None`
- `__repr__` : represente le maillon et ses suivants lors de l'exécution de l'instruction `print` sur une
- `taille()` : méthode **récursive** qui renvoie le nombre d'éléments de la chaine,
- `premier_maillon()` : renvoie la valeur du premier de la chaine ,
- `dernier_maillon()` : renvoie la valeur du dernier élément chaine,
- `maillon(i)` : renvoie la valeur du maillon d'indice i ,
- `ajouter_debut(m)` : ajoute un maillon `m` au début de la chaine,
- `ajouter_fin(m)` : ajoute un maillon `m` à la fin de la chaine,
- `inserer( i, v)` : insert un maillon `v` à l'indice `i`,
- `supprimer(i)` : supprime le maillon qui est à l'indice i,

•• Détails des différentes primitives de la classe `Maillon` __

— `__init__` : le constructeur prend en paramètre `valeur` et `suivant` qui par défaut vaut `None`.
Assertion : le type de suivant doit être None ou Maillon

— `__repr__` : represente le maillon et ses suivant lors de l'exécution de l'instruction sur une instance `Maillon`.

Exemple :

```
>>> chaine = Maillon(21 , Maillon ( 15, Maillon( 45)))
>>> print(chaine)
21 -> 15 -> 45
```

— `taille()` : méthode **reccursive** qui renvoie le nombre d'éléments de la chaine,

Exemple :

```
>>> chaine = Maillon(21 , Maillon ( 15, Maillon( 45)))
>>> chaine.taille()
3
```

— `dernier_maillon()` : méthode **reccursive** qui renvoie la valeur du dernier maillon chaine,

Exemple :

```
>>> chaine = Maillon(21 , Maillon ( 15, Maillon( 45)))
>>> chaine.dernier_maillon()
45
```

— `maillon_indice(i)` : méthode **reccursive** qui renvoie la valeur du maillon d'indice i de la chaine ,
Assertion : l'indice ne doit pas dépasser la longueur de la chaine -1

Exemple :

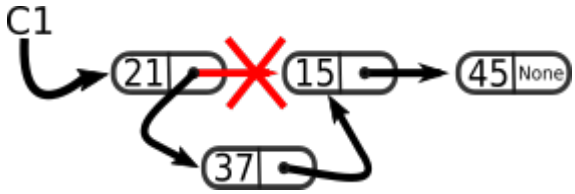
```
>>> chaine = Maillon(21 , Maillon ( 15, Maillon( 45)))
>>> chaine.get_maillon_indice(1)
21
```

— `ajouter_fin(m)` : méthode qui ajoute un maillon m à la fin de la chaine,
Assertion : le type de m doit être Maillon

Exemple :

```
>>> chaine = Maillon(21 , Maillon ( 15, Maillon( 45)))
>>> m = Maillon( 56)
>>> chaine.ajouter_fin(m)
>>> print(chaine)
21 -> 15 -> 45 -> 56
```

— `inserer(v, i)` : méthode qui insère un maillon ayant pour valeur `v` à l'indice `i`,



Assertion : l'indice ne doit pas dépasser la longueur de la chaine
Assertion : l'indice ne doit pas être supérieur à 0

Exemple :

```
>>> chaine = Maillon(21 , Maillon ( 15, Maillon( 45)))
>>> chaine.inserer(1, 6)
>>> print(chaine)
21 -> 15 -> 6 -> 45
```

— `supprimer(i)` : méthode qui supprime le maillon qui est à l'indice i,

Assertion : l'indice ne doit pas dépasser la longueur de la chaine
Assertion : l'indice doit être strictement superieur à 0

Exemple :

```
>>> chaine = Maillon(21 , Maillon ( 15, Maillon( 45)))
>>> chaine.supprimer(1)
>>> print(chaine)
21 -> 45
```

Pile

i Les piles sont extrèmement utiles en informatique et vous les utilisez quotidiennement, parfois même sans vous en rendre compte :

- La fonction annuler (`Ctrl-Z`) de votre traitement de textes par exemple est une pile : Quand vous tapez `Ctrl-Z`, vous annulez la dernière opération effectuée. Quand vous faites une nouvelle opération, celle-ci est mémorisée au sommet de la pile. Vous ne pouvez pas annuler l'avant dernière bopération sauf à annuler la dernière.
- Le bouton retour de votre navigateur internet fonctionne également à l'aide d'une pile. Les pages web consultées lors de votre navigation sur une page sont empilées et le bouton retour permet d'accéder à la dernière page présente sur la pile.
- Certaines calculatrices fonctionnent à l'aide du'une pile pour stocker les arguments des opérations : c'est le cas de beaucoup de calculatrices de la marque HP, dont la première calculatrice scientifique ayant jamais été produite : la HP 35 de 1972.

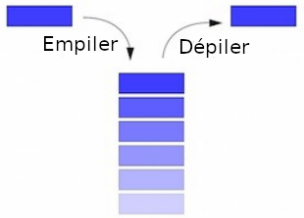
• Définition & représentation

Définition

`Pile`

Une pile est une liste particulière ou l'on accède uniquement au dernier élément que l'on nomme le sommet de la pile.
Le dernier élément entré est le premier à sortir.

On représente en général cette structure sous forme verticale.



Exemple

On peut prendre pour exemple une pile d'assiettes : La dernière assiette rangée sera la première que l'on ressortira. On parle parfois de pile LIFO (Last In First Out : dernier entré, premier sorti).

• Interface

•• Opérations

Les piles ne peuvent être modifiées que par 2 opérations :

- **empiler** un nouvel élément (ajouter un élément au sommet de la pile)
- **dépiler** le dernier élément (on l'enlève de la pile)

Il n'est pas possible d'accéder directement au n-ième élément d'une pile : Il faudrait dépiler les n-1 premiers éléments et de ce fait, détruire partiellement notre pile.

•• Pimitives

Les quelques primitives des piles les plus courantes sont :

- creer : créer une pile vide
- empiler un nouvel élément (ajouter un élément au sommet de la pile)
- dépiler le dernier élément saisi
- de consulter la valeur de l'élément au sommet de la pile sans le dépiler
- de tester si la pile est vide
- de connaître le nombre d'éléments dans la pile.

Exercice 4

On contruite deux piles avec les instructions suivantes :

```
1  créer pile vide, nommée p1
2  créer pile vide, nommée p2
3  empiler(1) sur p1
4  empiler(2) sur p1
5  empiler(3) sur p1
6  empiler(4) sur p1
7  empiler(5) sur p1
8  empiler(depiler(p1)) sur p2
9  empiler(depiler(p1)) sur p2
10 empiler(depiler(p1)) sur p2
```

Réprésenter les piles p1 et p2 à la fin de ces instructions

.....

.....

.....

.....

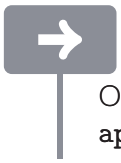
.....

.....

.....

Implémentation

- `__init__()` : Un constructeur qui crée une pile vide
- `est_vide()` qui renvoie `True` si la pile est vide et `False` sinon
- `depile()` qui dépile le dernier élément saisi renvoie la valeur dépilée ou None si la pile est vide
- `empile(v)` qui prend en paramètre une valeur `v` et empile cette valeur ne renvoie rien
- `sommet` qui permet de recuperer le sommet de la pile et le laissant au sommet de la pile



Remarque

On utilisera un tableau pour représenter une pile ainsi que les méthodes `append` et `pop`

File

i Les files sont extrèmement utiles en informatique et vous les utilisez quotidiennement, parfois même sans vous en rendre compte :

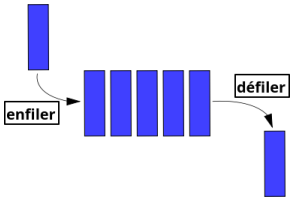
- les serveurs d'impression, qui traitent ainsi les requêtes dans l'ordre dans lequel elles arrivent, et les insèrent dans une file d'attente (dite aussi queue ou spool) ;
- les serveurs web traitent aussi les requêtes dans l'ordre dans lequel elles arrivent ;
- Certains ordonnanceurs dans un système d'exploitation, qui doivent accorder du temps-machine à chaque tâche, sans en privilégier aucune ;

• Définition & représentation

Définition

`File`

Une file est une liste particulière ou l'on accède uniquement à l'élément le plus ancien (qui a été placé en premier).



Exemple

On peut prendre pour exemple la file d'attente à la cantine : Le premier arrivé sera le premier servit.

On parle FIFO (First In Last Ou)

• Interface

•• Opérations

Les files ne peuvent être modifiées que par 2 opérations :

- **enfiler** un nouvel élément (ajouter un élément en queue de la file)
- **défiler** le premier élément (on l'enlève la tête de la file)

Il n'est pas possible d'accéder directement au n-ième élément d'une pile : Il faudrait dépiler les n-1 premiers éléments et de ce fait, détruire partiellement notre pile.

•• Pimitives

Les quelques primitives des files les plus courantes sont :

- creer : créer une file vide
- empiler un nouvel élément (ajouter un élément au sommet de la file)
- dépiler le dernier élément saisi

— de consulter la valeur de l'élément au sommet de la file sans le dépiler
— de tester si la file est vide
— de connaître le nombre d'éléments dans la file.

Exercice 5

On contruite deux files avec les instructions suivantes :

```
créer file vide, nommée f1
créer file vide, nommée f2
enfiler(1) sur p1
enfiler(2) sur p1
enfiler(3) sur p1
enfiler(4) sur p1
enfiler(5) sur p1
enfiler(defiler(p1)) sur p2
enfiler(defiler(p1)) sur p2
enfiler(defiler(p1)) sur p2
```

Réprésenter les piles p1 et p2 à la fin de ces instructions

.....

.....

.....

.....

.....

.....

.....

• Implémentation

- `__init__()` : Un constructeur qui crée une file vide
- `est_vide()` qui renvoie `True` si la file est vide et `False` sinon
- `enfile(v)` qui prend en paramètre une valeur `v` et enfile cette valeur en queue de file et ne renvoie rien
- `defile()` qui défile le dernier élément saisi renvoie la valeur dépilée ou `None` si la file est vide
- `tete` qui permet de recuperer le sommet de la file et le laissant au sommet de la file