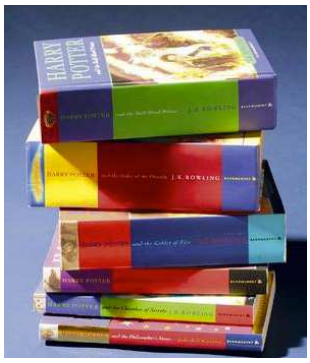
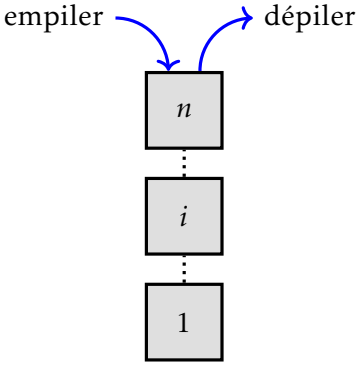
Piles et Files

Les piles (*stacks*)

le premier empilé est le dernier à être dépilé

LIFO (Last In first Out)

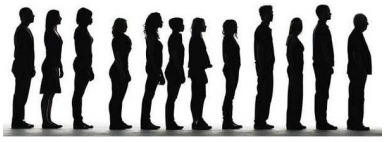


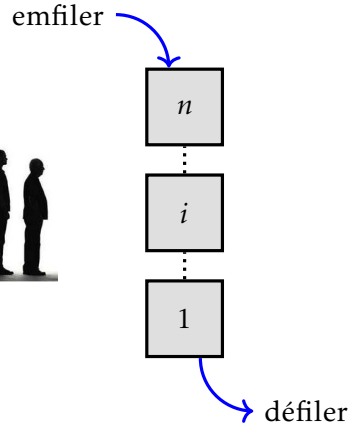


Les files

le premier entré est le premier à sortir

FIFO (First In First Out)





Objectifs

A la fin de la séquence d'enseignement l'élève doit

- distinguer une pile et une file d'une liste **Python**
- pouvoir manipuler les structures de données suivantes :
  - piles (ajout d'un élément, suppression du dernier)
  - files (ajout d'un élément, suppression du dernier)

Table des matières

1 Introduction 2

2 Les piles et les files 2

2.1 Définitions 2

2.2 Principales fonctions associées 3

2.3 Piles et files définies par listes chaînées 3

2.4 Pile définie par un tableau P de n éléments 3

2.5 File définie par un tableau F de n éléments 4

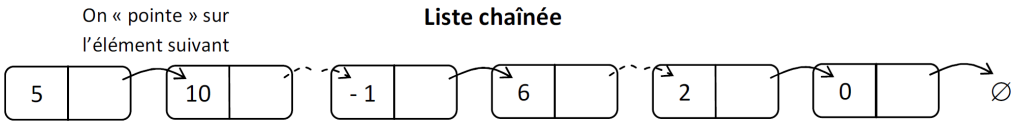
1 Introduction

Les listes (pas au sens python) sont des données composites. Elles rassemblent dans une seule structure un ensemble d'entités plus simples qui peuvent être de types différents. En général, on traite plutôt de listes d'éléments de même type, par exemple des listes de nombres.

La liste est un objet de la catégorie des séquences, lesquelles sont des collections ordonnées d'éléments. Cela signifie simplement que les éléments d'une liste sont toujours disposés dans un certain ordre. Contrairement aux chaînes de caractères, les listes sont des séquences modifiables.

On peut implémenter une liste de deux manières :

- **par une liste chaînée :** Chaque maillon est un objet dans lequel on stocke l'élément et une information pour trouver le suivant (voire aussi le précédent). C'est une structure de données récursive. La liste chaînée n'a pas de taille fixée à l'avance. Seul l'objet « maillon » a une certaine dimension.



- **par un tableau :** Chaque élément de la liste peut être désigné par sa place dans la séquence, à l'aide d'un index. Pour accéder à un élément déterminé, on utilise le nom de la variable qui contient la liste et on lui accole entre deux crochets, l'index numérique qui correspond à sa position dans la chaîne. Le tableau a une dimension fixe. Certains langages (comme Python) permettent de modifier sa taille au cours de son utilisation.

Indices	1	2	...	i	i + 1	...	n - 1	n
Tableau	5	10	...	-1	6	...	2	0

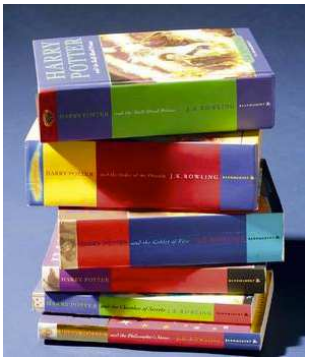
2 Les piles et les files

2.1 Définitions

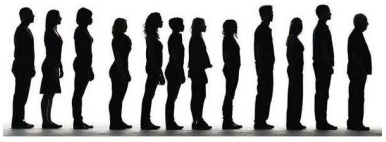
Les piles et les files sont des listes particulières : on accède aux éléments par les extrémités, c'est-à-dire au début ou à la fin.

On distingue :

les piles (*stacks*)
le premier empilé est le dernier à être dépilé
LIFO (Last In first Out)



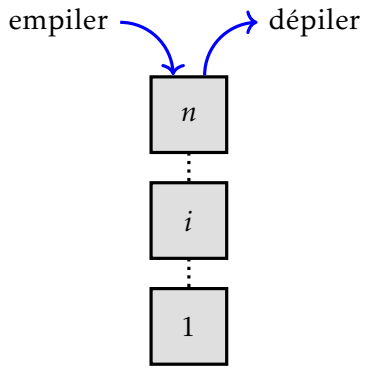
les files
le premier entré est le premier à sortir
FIFO (First In First Out)



2.2 Principales fonctions associées

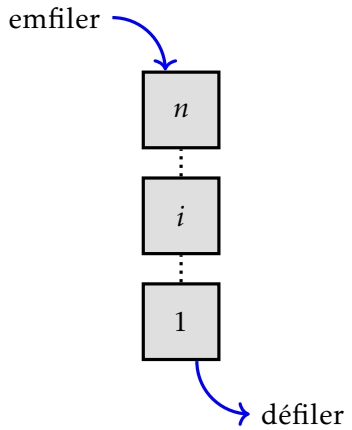
Pour les piles, les principales fonctions associées aux piles sont :

- ajouter un sommer un élément `x`: **empiler** `Pile.append(x)`
- supprimer du sommet le dernier élément de la pile : **dépiler** `Pile.pop()`.
  Remarque : on peut récupérer cet élément en l'affectant à une variable : `x=Pile.pop()`
- tester si la pile est vide : **est_vide** `Pile==[]`
- tester si la pile est plein : **est_pleine** avertissement par `overflow` ou `stackoverflow`



Pour les files, les principales fonctions associées aux piles sont :

- ajouter en queue : **enfiler**
- supprimer en tête : **défiler**
- tester si la pile est vide : **est_vide** `File==[]`
- tester si la pile est plein : **est_pleine** avertissement par `overflow` ou `stackoverflow`



2.3 Piles et files définies par listes chaînées

C'est l'objet du Tp `GRAPHE-Tp-1` pour lequel on utilise une classe `Pile` possédant uniquement les méthodes `empiler`, `depiler`, `est-vide` et`sommet` (retournant la valeur du sommet de la pile).

2.4 Pile définie par un tableau P de n éléments

- l'ordre est déterminé par les indices des éléments du tableau `P[1:n]`
- le tableau possède un attribut `sommet[P]` qui indexe l'élément le plus récemment inséré
- la pile se constitue des éléments `P[1:sommet[P]]` où `P[1]` est l'élément de base de la pile (le plus ancien) et `P[sommet[P]]` est l'élément situé au sommet (le plus récent)
- quand sommet[P] = 0, la pile est vide
- Si **sommet[P]** est supérieur à n, on dit que la pile déborde
  Remarque : Dépiler une pile vide provoque une erreur ; on parle de débordement négatif.

PileVide (P)
1: **si** sommet[P] = 0 **alors**
2: **renvoi:** Vrai
3: **sinon**
4: **renvoi:** Faux
5: **fin si**

Empiler (P,x)
1: sommet[P] ←sommet[P]+1
2: P[sommet[P]] ←x

Dépiler (P)
1: **si** PileVide(P) **alors**
2: "débordement négatif" (Erreur)
3: **sinon**
4: sommet[P] ←sommet[P]-1
5: **renvoi:** P[sommet[P]+1]
6: **fin si**

Pile								sommet[P]	Empiler	Dépiler
	1	2	3	4	5	6	7			
P	15	6	2	9				4	on place en 5	on récupère 9
	1	2	3	4	5	6	7			
P	15	6	2	9	17	3	12	7	débordement !	on récupère 12
	1	2	3	4	5	6	7			
P								0	on place en 1	débordement négatif !

2.5 File définie par un tableau F de n éléments

- la file a un attribut `tête[F]` qui repère sa tête
- la file a un attribut `queue[F]` qui indexe le prochain emplacement pouvant conserver un nouvel élément.
- les éléments de la file sont repérés par `tête[F]`, `tête[F+1]`,...,`queue[F]-1`
- l'emplacement 1 suit l'emplacement n dans un ordre circulaire
- quand `tête[F] = queue[F]`, la file est vide
  Remarque : au départ, on a `tête[F]=queue[F]=1`
- quand `tête[F] = queue[F]+1`, la file est pleine.

Enfiler(F,x)
1: F[queue[F]] ←x
2: **si** queue[F] = longueur[F] **alors** queue[F] ←1
3: **sinon** queue[F] ←queue[F]+1
4: **fin si**

Défiler(F)
1: x ←F[tête[F]]
2: **si** tête[F] = longueur[F] **alors** tête[F] ←1
3: **sinon** tête[F] ←tête[F]+1
4: **fin sirenvoi:** x

File												tête	queue	action précédente
	1	2	3	4	5	6	7	8	9	10	11	12			
P							15	6	2	8	4		7	12	Les éléments 15, 6, 2, 8 et 4 ont été enfilés
	1	2	3	4	5	6	7	8	9	10	11	12			
P	3	5					15	6	2	8	4	17	7	3	Les éléments 17, 3 et 5 ont été enfilés
	1	2	3	4	5	6	7	8	9	10	11	12			
P	3	5					15	6	2	8	4	17	8	3	l'élément 15 à été défilé