Cours 6 – Piles

I Introduction

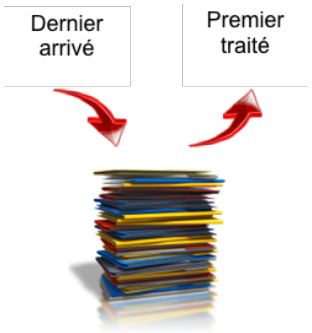
En informatique, il existe différentes façons de stocker des données. On a pour le moment principalement utilisé des tableaux (listes ou arrrays Python) et des bases de données relationnelles (pour les requêtes SQL) mais d'autres structures sont possibles (listes chaînées, graphes, arbres, piles...). Chaque type de structure répond à des besoins spécifiques, et le choix d'une structure influence l'efficacité du programme.

Dans un tableau, les données sont stockée dans une table de taille fixée (`np.array`) ou de taille variable (list). On peut accéder aux données par leur indice : $t[0]$, $t[2]$...

Valeur	45	154	58	78	31	5	74
Index	0	1	2	3	4	5	6

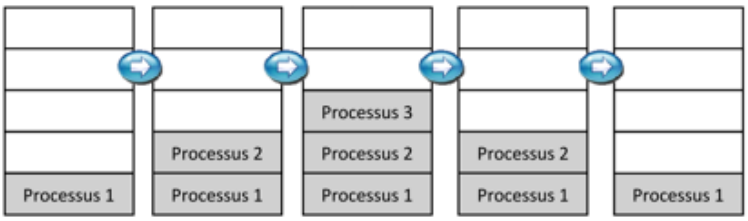
Dans une structure de pile, les données sont empilées comme dans une pile d'assiettes ou de dossiers : on pose les éléments les uns au dessus des autres, et on ne peut accéder qu'au dernier élément empilé.

On parle de structure "LIFO" (Last In First Out)



I.1 Exemples d'applications

1) Gestion de processus par un système d'exploitation



2) Page précédente / Annuler la saisie

Remarque : On ne considère dans les prochaines parties que des piles à capacité illimitée, ie des piles sans limite de taille.

II Primitives de piles

Il existe un certain nombre d'opérations de base sur les piles. Les opérations suivantes sont essentielles : elle sont parfois appelées primitives :

— Créer une pile vide

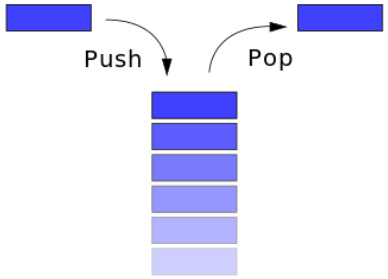
— Empiler un élément : PUSH

  — Un élément x est placé au sommet de la pile

— Dépiler un élément : POP

  — On retire le dernier élément de la liste (qui est donc modifiée) et on le renvoie

— Tester si la pile est vide



III Implémentation des primitives de piles en Python

On va maintenant voir comment on peut implémenter les opérations sur les piles en Python.

Pour représenter les piles, on va utiliser des listes Python. On va en particulier utiliser les deux méthodes suivantes :

— `L.append(x)` qui rajoute un élément `x` à la fin d'une liste `L`

— `L.pop()` qui enlève le dernier élément de la liste `L` et le renvoie.

1) Ecrire une fonction `creer_pile()` qui renvoie une pile vide (une liste vide Python)

2) Ecrire une fonction `push(x,p)` qui rajoute un élément `x` au sommet de la pile `p`

3) Ecrire une fonction `pop(p)` qui enlève le dernier élément de la pile `p` et le renvoie

4) Ecrire une fonction `est_vide(p)` qui teste si une pile est vide

IV Exercices

On utilise maintenant uniquement les opérations sur les piles définies précédemment. Toute autre opération sur les listes est prohibée.

IV.1 Recherche du dernier élément empilé

Ecrire une fonction `sommet(p)` qui renvoie le dernier élément empilé dans une pile `p` (supposée non vide). A la fin de l'exécution, `p` doit être dans le même état qu'au début.

```
In [6]: p=creer_pile()
        push(1,p)
        push(2,p)
        push(3,p)
        print(p)
        x=sommet(p)
        print(x)
        print(p)
```

```
Out [6]: [1, 2, 3]
         3
         [1, 2, 3]
```

IV.2 Dupliquer l'élément de tête

Ecrire une fonction `dup(p)` qui rajoute à une pile p (supposée non vide) une nouvelle fois son dernier élément. (Par exemple [1,2,3] devient [1,2,3,3]). Cette fonction doit modifier `p` mais ne renvoie rien.

```
In [8]: p=creer_pile()
        push(1,p)
        push(2,p)
        push(3,p)
        print(p)
        dup(p)
        print(p)
```

```
Out [8]: [1, 2, 3]
         [1, 2, 3, 3]
```

IV.3 Échanger les deux derniers éléments empilés

Écrire une fonction `swap(p)` qui échange les deux derniers éléments d'une pile p (supposée contenir au moins deux éléments). Cette fonction doit modifier `p` mais ne renvoie rien.

```
In [10]: p=creer_pile()
         push(1,p)
         push(2,p)
         push(3,p)
         print(p)
         swap(p)
         print(p)
```

```
Out [10]: [1, 2, 3]
          [1, 3, 2]
```

IV.4 Renversement/Taille d'une pile

1) Écrire une fonction `inversion1(p)` qui prend en argument une pile `p` et qui renvoie la pile obtenue en inversant l'ordre des éléments de `p`. (on s'autorise à vider la pile `p`).

```
In [14]: p=creer_pile()
         push(1,p)
         push(2,p)
         push(3,p)
         print(p)
         p2=inversion1(p)
         print(p2)
         print(p)
```

```
Out[14]: [1, 2, 3]
         [3, 2, 1]
         []
```

2) Écrire une fonction `taille(p)` qui renvoie la taille d'une pile p. A la fin de l'exécution, `p` doit être revenue dans son état initial. (On pourra utiliser une seconde pile). Évidemment, on n'utilisera pas `len(p)`

```
In [12]: p=creer_pile()
         push(1,p)
         push(2,p)
         push(3,p)
         print(p)
         print(taille(p))
         print(p)
```

```
Out [12]: [1, 2, 3]
          3
          [1, 2, 3]
```

3) Ecrire maintenant une fonction `inversion2(p)` qui renvoie la même liste que `inversion1(p)`, mais cette fois-ci, `p` doit revenir à son état initial

```
In [16]: p=creer_pile()
         push(1,p)
         push(2,p)
         push(3,p)
         print(p)
         p2=inversion2(p)
         print(p2)
         print(p)
```

```
Out [16]: [1, 2, 3]
          [3, 2, 1]
          [1, 2, 3]
```

V Piles à capacité finie

Si on connaît le nombre maximal N d'éléments qui sera stocké dans la pile, on peut réaliser une pile avec un tableau de taille $N + 1$.

0	1 … n	$n+1$ … N
n	éléments	place disponible

Le premier élément du tableau sert à stocker le nombre n d'éléments empilés. Ces éléments sont empilés dans les cases 1 à n. Le reste du tableau sert de mémoire disponible.

	0	1	2	3	…
`p = creer_pile(10)`	0	…			
`empiler(p, A)`	1	A	…		
`empiler(p, B)`	2	A	B	…	
`empiler(p, C)`	3	A	B	C	…
`depiler(p)`	2	A	B	C	…

Les éléments colorés sont ceux qui sont réellement dans la pile. Les autres, par exemple C à la dernière ligne, ne sont plus accessibles et seront écrasés lorsqu'on en empilera de nouveaux.

On peut encore définir les fonctions `creer_pile(N)`, `push(x,p)`, `pop(p)` et `est_vide(p)`. Il faut alors bien penser à modifier $p[0]$.

(Pour simplifier, on ne vérifiera pas s'il est bien possible d'empiler ou de dépiler. En effet, on ne peut pas dépiler si la pile est vide, ni empiler si la pile est pleine)