Les Structures de Données Python
Cours 3 : Type de données abstrait
types abstraits de données (TAD)
Les piles et les Files

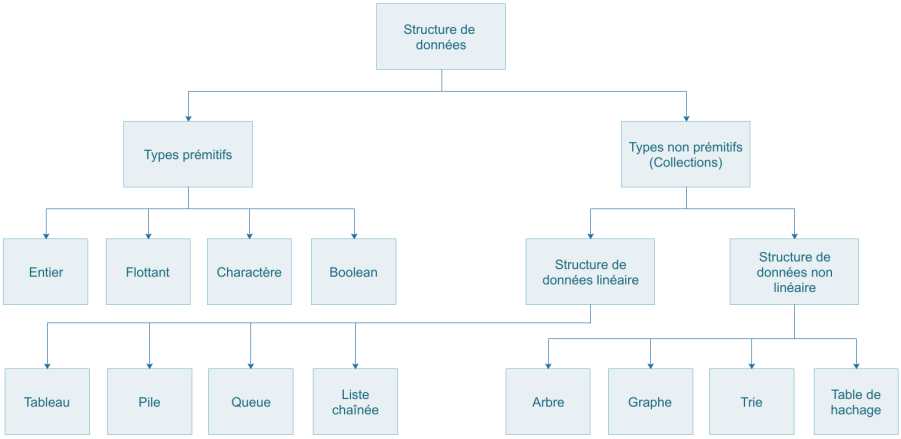
Halim Djerroud

ESIEE
PARIS

révision : 0.1

Plan

Rappel : Les types en python



Type de données abstrait

- ADT : *Abstract Data Type*

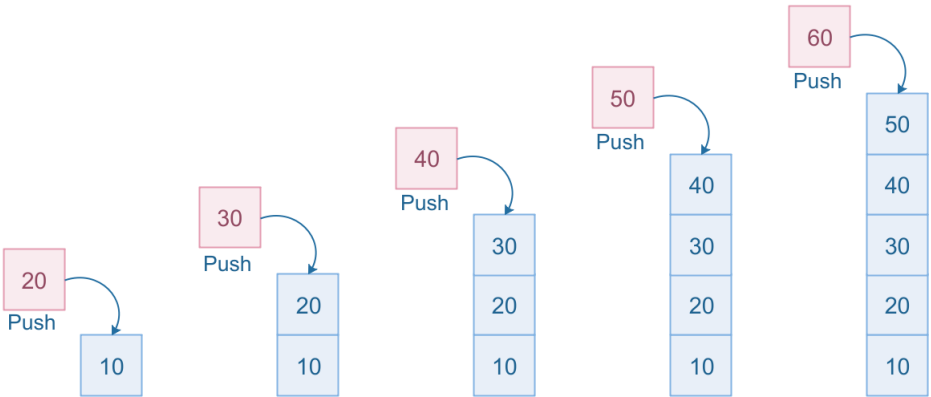
Pile ou *Stack*

- Une pile est une collection d'objets qui sont insérés et supprimés selon le principe du dernier entré, premier sorti (LIFO)
- On peut insérer des objets dans une pile à tout moment, mais ne peut accéder ou supprimer que l'objet inséré le plus récemment qui reste (ce qu'on appelle le « sommet » de la pile).

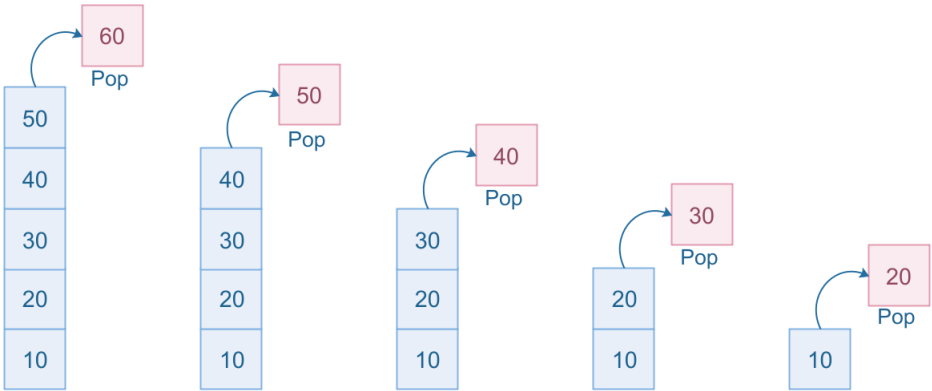
Exemple d'utilisation

- Les navigateurs Web Internet stockent les adresses des sites récemment visités dans une pile. Chaque fois qu'un utilisateur visite un nouveau site, l'adresse de ce site est "poussée" sur la pile des adresses visitées. Le navigateur permet ensuite à l'utilisateur de revenir sur les sites précédemment visités à l'aide du bouton "retour"
- Les éditeurs de texte fournissent généralement un mécanisme "d'annulation" qui annule les opérations d'édition récentes et rétablit les états antérieurs d'un document. Cette opération d'annulation peut être accomplie en gardant les modifications de texte dans une pile.

Principe de fonctionnement : Empiler



Principe de fonctionnement : Dépiler



Les opérations sur les piles

- Les piles sont les plus simples de toutes les structures de données, mais elles sont aussi parmi les plus importantes.
- Formellement, une pile est un type de données abstrait (ADT) tel qu'une instance `S` supporte les deux méthodes suivantes :
  - `S.push(e)` : Ajoute l'élément `e` au sommet de la pile `S`.
  - `S.pop(e)` : Retire et renvoie l'élément du sommet de pile `S`, une l'erreur se produit si la pile est vide.

Les opérations sur les piles

Pour plus de commodité il existe aussi d'autres méthodes d'accès :

- `S.top()` : Renvoie une référence au sommet de la pile `S`, sans le retirer ; une erreur se produit si la pile est vide.
- `len(S)` : Renvoie le nombre d'éléments dans la pile `S`

Exemple

```
stack = []                  # la pile est vide
stack.append(10)            # la pile contient [10]
stack.append(20)            # la pile contient [10, 20]
stack.append(30)            # la pile contient [10, 20, 30]
result = stack.pop()        # la pile contient [10, 20]

print(result)
print(stack)
```

Pile avec slicing

```
stack = []

stack[len(stack):] = [10]
stack[len(stack):] = [20]
stack[len(stack):] = [30]

result, stack = stack[len(stack) - 1], stack[:len(stack)-1]

print(result)
print(stack)
```

File ou *Queue*

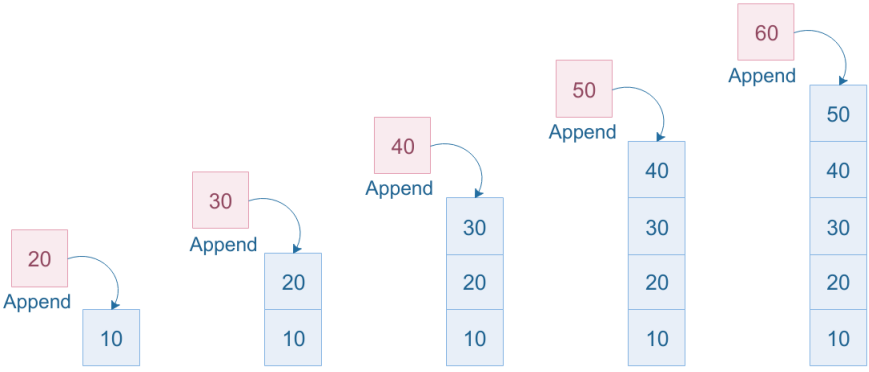
- Une file est une séquence d'éléments
- L'ajout d'un nouvel élément (*enqueue*) se fait à sa fin
- Retrait d'un élément (*dequeue*) à sa tête.
- Une file est une liste de type *FIFO (First-in First-out)*
- Le premier élément qui y a été ajouté sera aussi le premier qui en sortira

Exemple d'utilisation

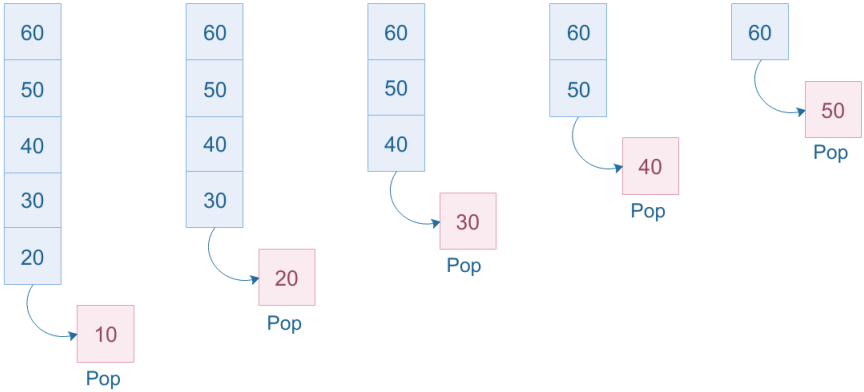
- Exemple :

```
queue = []                      # la file est vide

queue.append(10)                 # la file contient [10]
queue.append(20)                 # la file contient [10, 20]
queue.append(30)                 # la file contient [10, 20, 30]

result = queue.pop(0)            # la file contient [20, 30]

print(result)
print(queue)
```

Principe de fonctionnement : Ajouter (*Enqueue*)



Principe de fonctionnement : Retirer (*Dequeue*)



Utilisation de POP

- Pour retirer le premier élément, on a dû faire l'appel pop(0). La raison pour laquelle on doit spécifier un indice est que la fonction pop est plus générale et permet en fait de retirer n'importe quel élément dans une liste, en précisant son indice.

File avec slicing

```
queue = []

queue[len(queue):] = [10]
queue[len(queue):] = [20]
queue[len(queue):] = [30]

result, queue = queue[0], queue[1:]

print(result)
print(queue)
```