Structure de données: Pile &File

Programmation en Python–1ère année–

MPSI2/MPSI3

mazza8azzouz@gmail.com

27 janvier 2025

Plan

La structure pile

Définition

- Une pile (stack en anglais) est une structure dynamique dans laquelle l'insertion au la suppression d'un élément s'effectue toujours à partir de la même extrémité de cette structure.
- Cette extrémité est appelée le sommet de la pile

.

Le mécanisme LIFO (last in, first out)

- Une pile permet de modéliser un système régi par le mécanisme ≪ dernier arrivé premier sorti ≫ ; on dit souvent LIFO (last in, first out)
- L'action consistant à ajouter un nouvelle élément au sommet de la pile s'appelle empiler ; celle consistant à retirer l'élément situé au sommet de la pile s'appelle dépiler

.

Figure – Pile d'd'assiettes

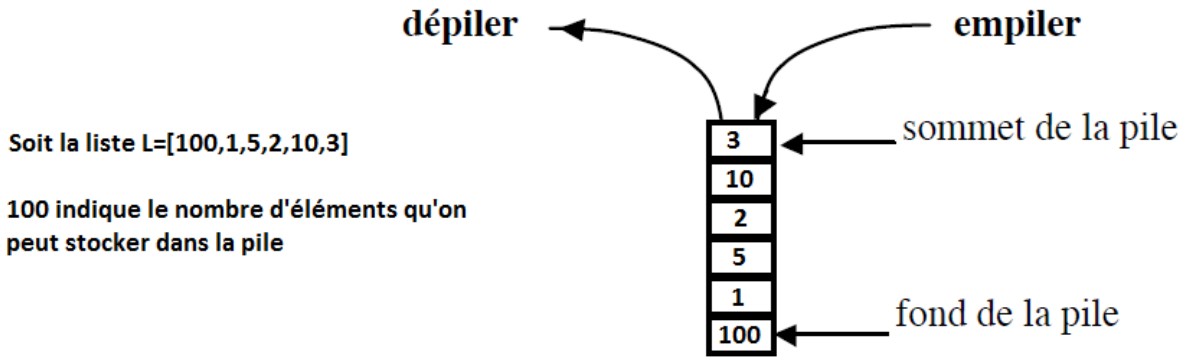
Les applications d'une pile

De nombreuse applications s'appuient sur l'utilisation d'une pile, on peut citer :

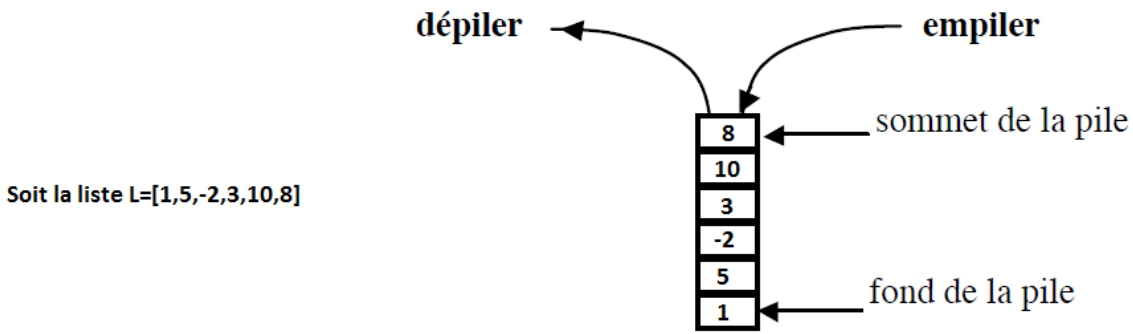
- Dans un navigateur web, une pile sert à mémoriser les pages Web visitées. L'adresse de chaque nouvelle page visitée est empilée et l'utilisateur dépile l'adresse de la page précédente en cliquant le bouton ≪ Afficher la page précédente ≫.
- L'évaluation des expressions mathématiques en notation post-fixée (ou polonaise inverse) utilise une pile.
- La fonction ≪ Annuler la frappe ≫ (en anglais ≪ Undo ≫) d'un traitement de texte mémorise les modifications apportées au texte dans une pile.
- Vérification de parenthèsage d'une chaine de caractères ;
- La récursivité (une fonction qui fait appel à elle même) ;
- etc.

Implémentation d'une pile avec une liste

- En python, il existe deux façons pour implémenter une pile avec une liste :
  1. Soit on utilise une liste de taille finie pour réaliser une pile ;



  2. Soit on utilise une liste de taille non finie pour réaliser une pile ;.



Les primitives

- Afin de manipuler une pile, on doit programmer un ensemble des fonctions de gestion d'une pile (primitives). Voici les primitives communément utilisées :
  1. PileVide() : Crée une pile vide ;
  2. EstVide(P) : renvoie vrai si la pile P est vide, faux sinon ;
  3. Taille(P) : renvoie la taille de la pile P ;
  4. SommetPile(P) : renvoie l'élément sommet de la pile P ;
  5. Empiler(P,v) : ajoute au sommet de la pile P l'élément v ;
  6. Depiler(P) : supprime de la pile le sommet.

Remarque

Les primitives peuvent être utilisées dans les deux cas d'implémentation d'une pile : soit avec une liste limitée ou soit avec une liste illimitée

Les primitives d'une pile **à capacité illimitée**

- La fonction PileVide() : permet de créer une pile vide, pour cela, elle retourne une liste vide :

```
def PileVide() :
    return []
```

- La fonction EstVide() : permet de tester si une pile est vide : :

```
def EstVide(P) :
    if len(P)==0 :
        return True
    else :
        return False
```

- La fonction Taille() : permet de retrouner le nombre des éléments d'une pile :

```
def Taille(P) :
    return len(P)
```

Les primitives d'une pile **à capacité illimitée**

- La fonction Empiler(P,v) : permet d'empiler un élément v dans une pile P :

```
def Empiler(P,v) :
    P.append(v)
    return P
```

- La fonction Depiler(P) : permet de supprimer le dernier élément empilé dans la pile

```
def Depiler(P) :
    if len(P)==0 :
        print("Erreur : pile vide")
    else :
        P.pop()
    return P
```

- La fonction SommetPile(P) : permet de retrouner le dernier élément empilé dans la pile P

```
def SommetPile(P) :
    if len(P)==0 :
        print("pile vide")
        return None
    else :
        return P[len(P)-1]
```

Implémentation d'une pile à **capacité limitée**

- La fonction **creer_pile(c)** crée une liste vide de longueur c+1

```
def creer_pile(c) : # c'est la capacité de la pile :
    pile = (c+1) * [None]
    pile[0] = 0
    return pile
```

→ Le premier élément pile[0] contient le nombre d'élément de la pile, initialement 0.

- Avec cet implémentation il est alors facile de retourner la taille de la pile :

```
def taille(p) : :
    return p[0]
```

- Pour empiler un élément il faut prendre garde que la taille de la pile n'excède pas sa capacité et incrémenter le compteur de la taille p[0] :

```
def empiler(p,e) :
    taille = p[0]
    assert taille < len(p)-1
    p[taille+1] = e
    p[0]=p[0] + 1
    return p
```

Implémentation d'une pile à **capacité limitée**

- De même pour le dépilement :

```
def depiler(p) :
    taille = p[0]
    assert taille > 0 # erreur si pile vide
    e = p[taille]
    p[taille] = None # l'élément est supprimé
    p[0]=p[0]-1 # Décrémentation de la taille
    return e
```

- Pour tester si la pile est vide :

```
def est_vide(p) :
    if p[0]==0 :
        return True
    else :
        return False
```

- Pour lire le dernier élément ; s'assurer que la pile est non vide

```
def top(p) :
    taille = p[0]
    assert taille > 0
    p[taille+1] = e
```

La structure file

Définition

- Une file (queue en anglais) est une structure de données dans laquelle l'insertion se fait à la fin et la suppression d'un élément s'effectue à partir de début de cette structure.
- Le fonctionnement ressemble à une file d'attente : les premières personnes à arriver sont les premières personnes à sortir de la file.

.

Le mécanisme FIFO (first in, first out)

- Une file permet de modéliser un système régi par le mécanisme "premier arrivé premier sorti" ; on dit souvent FIFO (first in, first out)
- L'action consistant à ajouter un nouvelle élément s'appelle enfiler ; celle consistant à retirer l'élément situé au début de la file s'appelle défiler

.

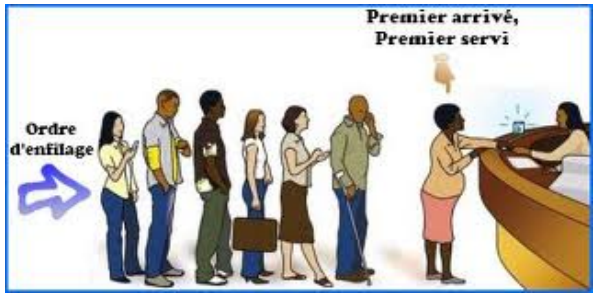


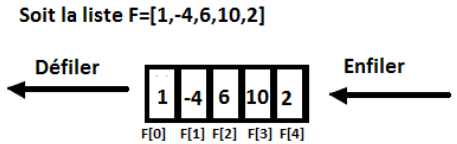
Figure – file d'attente

Les applications d'une file

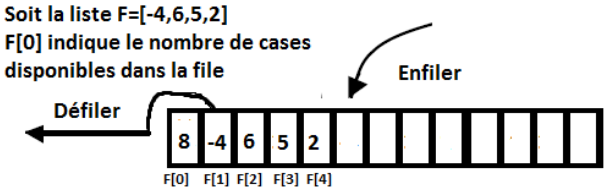
- En général, on utilise des files pour mémoriser temporairement des transactions qui doivent attendre pour être traitées ;
- Les serveurs d'impression, qui doivent traiter les requêtes dans l'ordre dans lequel elles arrivent, et les insèrent dans une file d'attente (ou une queue) ;
- Certains moteurs multitâches, dans un système d'exploitation, qui doivent accorder du temps-machine à chaque tâche, sans en privilégier aucune ;
- Un algorithme de parcours en largeur utilise une file pour mémoriser les noeuds visités ;
- On utilise aussi des files our créer toutes sortes de mémoires tampons (en anglais buffers).
- etc.

Implémentation d'une file avec une liste

- En python, il existe deux façons pour implémenter une file avec une liste :
  1. Soit on utilise une liste de taille finie pour réaliser une file ;



  2. Soit on utilise une liste de taille non finie pour réaliser une file.



La collection deque

- Le type liste n'est pas adéquat pour représenter une file, pour cela, sera mieux d'utiliser le type deque de module collections.
  >>>from collections import deque
  >>>F=deque([1,3,10])
- La fonction popleft permet de défiler et la fonction append permet d'enfiler.

.

Les primitives

- Pour résoudre un problème donné qui repose sur la structure file, il est nécessaire de programmer un ensemble des primitives pour la gestion d'une file :
  1. FileVide() : renvoie une liste vide ;
  2. EstVide() : renvoie vrai si la file est vide, faux sinon ;
  3. PremierElement(F) : renvoie le premier élément de la file F ;
  4. Enfiler(P,v) : ajoute à la fin de la file F l'élément v ;
  5. Defiler(F) : supprime de la file F le premier élément.

Remarque

Les primitives peuvent être utilisées dans les deux cas d'implémentation d'une file : soit avec une liste limitée ou soit avec une liste illimitée

Les primitives d'une file **à capacité illimitée**

- La fonction FileVide() : permet de créer une file vide, pour cela, elle retourne une liste vide :

```
def FileVide() :
    return []
```

- La fonction EstVide() : permet de tester si une file est vide : :

```
def EstVide(F) :
    if len(F)==0 :
        return True
    else :
        return False
```

- La fonction Taille() : permet de retrouner le nombre des éléments d'une file :

```
def Taille(F) :
    return len(F)
```

Les primitives d'une file **à capacité illimitée**

- La fonction Enfiler(F,v) : permet d'enfiler un élément v dans une file F :

```
def Enfiler(F,v) :
    F.append(v)
    return F
```

- La fonction Defiler(F) : permet de supprimer le premier élément de la file F

```
def Depiler(F) :
    if len(F)==0 :
        print("Erreur : file vide")
    else :
        F.pop(0)#F.remove(F[0])
    return F
```

- La fonction PremierElement(F) : permet de retrouner le premier élément enfilié dans la file F

```
def PremierElement(F) :
    if len(F)==0 :
        print("file vide")
        return None
    else :
        return F[0]
```