Institut Préparatoire aux Études
d'Ingénieurs de Tunis

Chapitre I

Les piles et les files en Python

Mme Nesrine Ayed

Plan

- Définition.
- La structure pile.
  - principe.
  - implémentation.
- La structure file.
  - principe.
  - Implémentation.
- Applications.

LA STRUCTURE PILE/FILE

- Structure de donnée.
  - Simple.
  - Dynamique.
  - Linéaire.

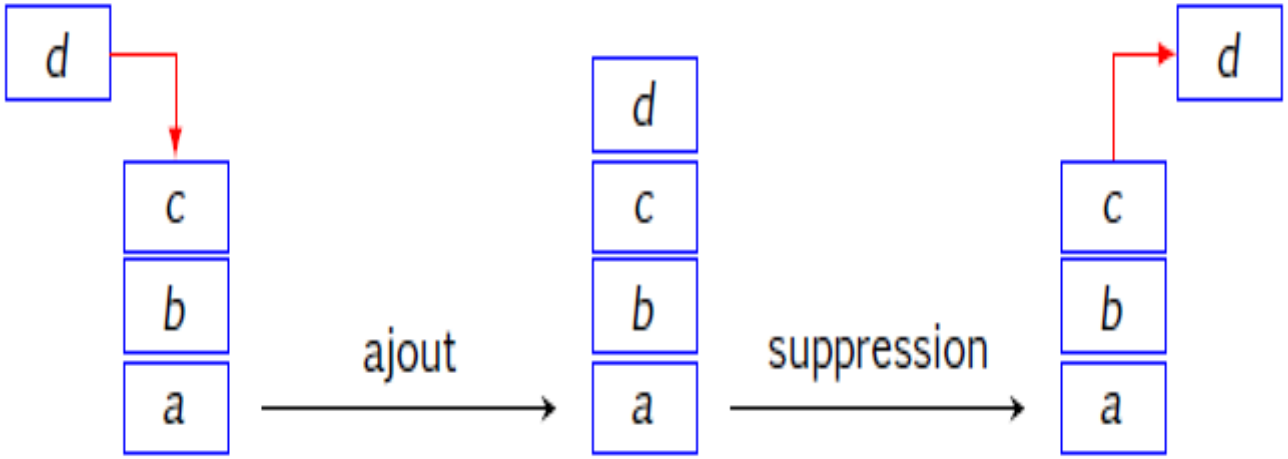
- l'ajout et la suppression ne peuvent se faire qu'aux extrémités.

La structure Pile

- L'ajout et la suppression des éléments se font de la même extrémité (sommet).
- Fondé sur le principe "dernier arrivé premier sorti"
  - LIFO (Last In First Out).
- le principe même de la pile d'assiettes :
  - C'est la dernière assiette posée sur la pile d'assiettes sales qui sera la première lavée.

- Applications:
  - Des retours à la page précédente.
  - La touche "undo" annuler des éditeurs de textes…

La structure Pile



Implémentation de la structure Pile

- La réalisation de la structure pile doit fournir les fonctions de base suivantes:
  - **creer_pile() :** création d'une pile vide.
  - **empiler(p,e) :** empiler un élément e au sommet de une pile p (ajout).
  - **pile_vide(p) :** vérification de l'état d'une pile p.
    - True=pile vide
    - False=pile pleine

Implémentation de la structure Pile

- **desempiler(p) :** désempiler une pile p (supprimer le sommet).
- **sommet_pile(p):** récupérer le sommet d'une pile p.
- **taille_pile(p):** la taille d'une pile p (facultative).

- La création de la structure pile sous python se fera à l'aide de la structure liste.
  - le sommet de la pile est le dernier élément de la liste.

Implémentation de la structure Pile

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

```
def desempiler(p):
    if pile_vide(p)==False:
        p.pop()
```

```
def empiler(p,e):
    p.append(e)
```

```
def sommet(p):
    if pile_vide(P)==False:
        return p[-1]
```

```
def pile_vide(p):
    return len(p)==0
```

```
def taille_pile(p):
    return len(p)
```

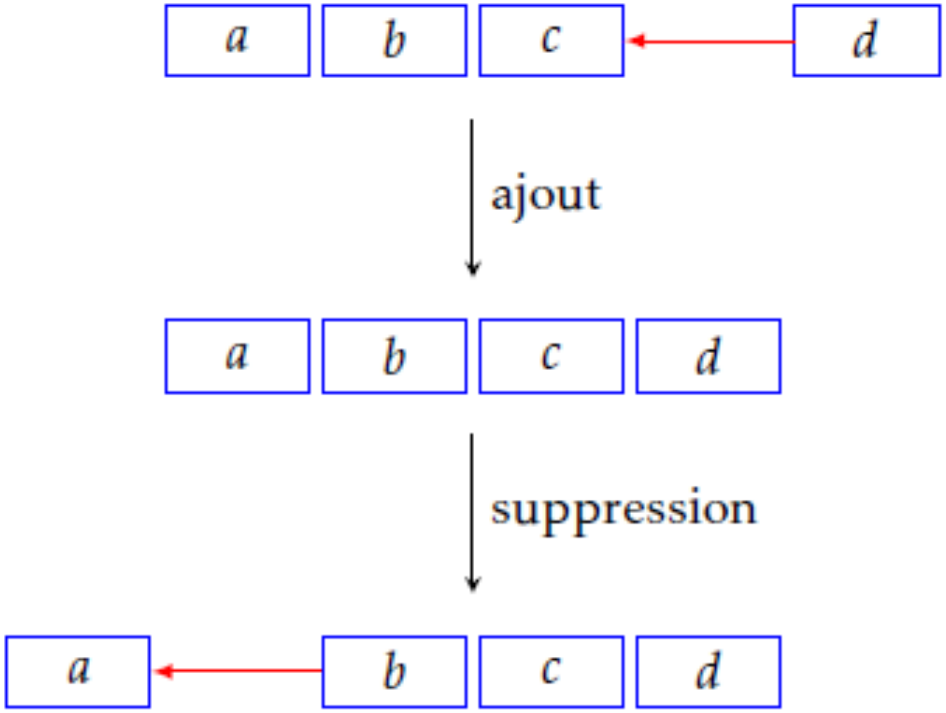
La structure File

- L'ajout et la suppression des éléments se font à partir de deux extrémités différentes.
  - L'ajout se fait en queue.
  - La suppression se fait en tête.
- Fondée sur le principe "premier arrivé premier sorti"
  - FIFO (First In First Out).
- Le principe même de:
  - L'entassement des gobelets dans un distributeur de boisson.
    - on entasse par le haut(queue) et on saisit par le bas (tête).
  - File d'attente.
    - les premières personnes arrivées sont les premières personnes à sortir de la file.

La structure File

- Applications:
  - Mémoriser temporairement des transactions qui doivent attendre pour être traitées.
  - Les serveurs d'impression, qui traitent ainsi les requêtes dans l'ordre dans lequel elles arrivent, et les insèrent dans une file d'attente.

La structure File



Implémentation de la structure File

- La réalisation de la structure file doit fournir les fonctions de base suivantes:
  - creer_file(): création d'une file vide.
  - enfiler(f,e): enfiler un élément e en queue de file f.
  - taille_file(f): la taille d'une file f.
  - defiler(f): défiler un élément (tête) d'une file f et le retourner.
  - file_vide(f): vérification de l'état d'une file f.
    - True=file vide
    - False=file pleine

Implémentation de la structure File

- La création de la structure file sous python se fera à l'aide de la structure liste.
  - La tête de la file est le premier élément de la liste.
  - La queue de la file est le dernier élément de la liste.

Implémentation de la structure File

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

```
def enfiler(f,e):
    f.append(e)
```

```
def taille_file(f):
    return len(f)
```

```
def file_vide(f):
    return taille_file(f)==0
```

```
def defiler(f):
    if file_vide(f)==False:
        s=f.pop(0)
        return s
```

Applications Pile/File:

- Ecrire en python les fonctions suivantes, en se basant impérativement sur les fonctions précédemment définies:

  - **Remplir_pile(n)** qui permet de créer et retourner une pile remplie par n entiers.
  - **Copier_pile(P)** qui retourne une copie de la pile P sans la modifier.
  - **Remplir_file(n)** qui permet de créer et retourner une file remplie par n entiers.
  - **Copier_file(F)** qui retourne une copie de la file F sans la modifier.

Applications Pile/File:

```
from op_pile import *
def Remplir_pile(n):
    p=creer_pile()
    for i in range(n):
        x=int(input("donner un elt de p "))
        empiler(p,x)
    return p
```

Applications Pile/File:

```
def Copier_pile(P):
    p1=creer_pile()
    p2=creer_pile()
    while not pile_vide(P):
        elt=sommet(P)
        desempiler(P)
        empiler(p1,elt)
    while not pile_vide(p1):
        elt=sommet(p1)
        desempiler(p1)
        empiler(P,elt)
        empiler(p2,elt)
    return p2
```

Applications Pile/File:

```
from op_file import *
def Remplir_file(n):
    f=creer_file()
    for i in range(n):
        x=int(input("donner un elt de  f"))
        enfiler(f,x)
    return f
```

Applications Pile/File:

```
def Copier_file_v1(F):
    F1=creer_file()
    F2=creer_file()
    while not file_vide(F):
        s=defiler(F)
        enfiler(F1,s)
        enfiler(F2,s)
    while not file_vide(F1):
        s=defiler(F1)
        enfiler(F,s)
    return F2

def Copier_file_v2(F):
    FF=creer_file()
    for i in range(taille_file(F)):
        s=defiler(F)
        enfiler(FF,s)
        enfiler(F,s)
    return FF
```

FIN