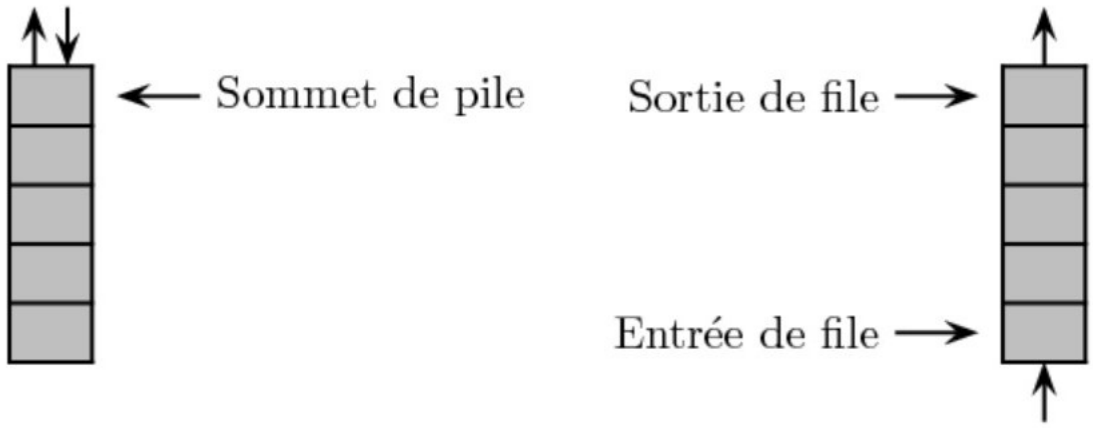
Cours : les Piles et Files

10 novembre 2021



Tout ceci est parfaitement conforme à l'intuition d'une pile d'assiettes, ou de la queue devant un guichet, et suggère fortement des techniques d'implémentations pour les piles et les files.

I. Piles

1. Structure de données

Une pile en python est une structure de données variables qui stoque des données à la suite. Le système d'agglomération des données est le FILO (First In Last Out). On l'implémente en python avec des listes.

Il y a plusieurs types d'opérations sur les Piles que l'on écrira avec des méthodes.

1. La méthode : *__init__* qui est le constructeur. On initie la pile à une liste vide.

```
1 class Pile:
2         def __init__(self):
3                 self.data = []
```

2. La méthode *empiler* pour ajouter un élément dans une liste.

```python
def empiler(self,element):
        self.data.append(element)
```

3. La méthode *hauteur* pour donner la hauteur de la pile.

```python
def hauteur(self):
        return len(self.data)
```

4. La méthode *est_vide* pour donner un booléen pour savoir si la liste est vide.

```python
def est_vide(self):
        return self.hauteur() == 0
```

5. La méthode *depiler* pour enlever un item de la pile et l'obtenir (Il peut y avoir une erreur si la liste est vide. En exercice vous traiterez ce cas).

```python
def depiler(self):
        return self.data.pop()
```

6. La méthode *sommet* pour voir/récupérer le dernier élément de la liste.

```python
def sommet(self):
        return self.data[-1]
```

2. Exercices sur les piles

Pour faire les exercices suivants vous n'utiliserez que les méthodes précédentes. A la fin de chaque méthode vous ajouterez une condition pour vérifier que le nombre de cartes (objets) est respectée.

▪ **Exercice .1.**
Faites une méthode qui retourne l'ordre de la pile.

1. Avec les cartes
2. En pseudo code
3. Sur les ordinateurs en python

▪

▪ **Exercice .2.**
Faites une fonction qui divise une pile en 2 piles (une avec la première moitié des cartes et une avec la seconde moitié des cartes)
1. Avec les cartes
2. En pseudo code
3. Sur les ordinateurs en python

▪

▪ **Exercice .3.**
Faites une fonction qui fusionne deux piles en alternant une carte de chaque pile.
1. Avec les cartes
2. En pseudo code
3. Sur les ordinateurs en python

▪

▪ **Exercice .4.**
Faites une fonction mélange_américain qui fusionne deux piles en alternant une ou deux cartes (le 1 ou 2 choisis aléatoirement) de chaque pile.
1. Avec les cartes
2. En pseudo code
3. Sur les ordinateurs en python

▪

II. Files

1. Structure de données

Une file en python est une structure de données variables qui stoque des données à la suite. Le système d'agglomération des données est le FIFO (First In First Out). On l'implémente en python avec des listes.

Il y a plusieurs types d'opérations sur les Files que l'on écrira avec des méthodes.
1. La méthode : *__init__* qui est le constructeur. On initie la pile à une liste vide.

```
class File:
        def __init__(self):
                self.data = []
```

2. La méthode *empiler* pour ajouter un élément dans une liste.

```
def enfiler(self,element):
        self.data.append(element)
```

3. La méthode *longueur* pour donner la hauteur de la pile.

```
def longueur(self):
        return len(self.data)
```

4. La méthode *est_vide* pour donner un booléen pour savoir si la liste est vide.

```
def est_vide(self):
        return self.longueur() == 0
```

5. La méthode *defiler* pour enlever un item de la pile et l'obtenir (Il peut y avoir une erreur si la liste est vide. En exercice vous traiterez ce cas).

```
def defiler(self):
        return self.data.pop(0)
```

6. La méthode *dernier* pour avoir le dernier arrivée dans la liste.

```
def dernier(self):
        return self.data[-1]
```

2. Exercice

On va faire un long exercice filé sur les files d'attente. Vous ne pourrez utiliser que les méthodes définies dans l'exercice ou précédemment.

Exercice .5.

Considérons la suite d'instructions suivantes :

```
F ← Creer_file
enfiler 21 à F
enfiler 22 à F
enfiler 23 à F
N ← defiler F
enfiler 24 à F
enfiler 25 à F
N ← defiler F
```

À la fin de cette séquence :

1. Que contient N
2. Que contient la pile F

■

■ Exercice .6.
Dans un supermarché il y a 5 caisses et une file d'attente commune. Dès qu'une caisse est libre, le client en tête de file y est envoyé. Le temps de passage en caisse est aléatoirement compris entre 3 et 10 minutes. Il y a dix clients dans la file d'attente.

1. Réaliser une simulation de leurs passages en caisses et afficher en combien de minutes tous les clients sont passés.
2. Refaire quelques essais avec plus de clients.
3. Réaliser une simulation de leurs passages en caisses et afficher en combien de minutes tous les clients sont passés

■

■ Exercice .7.
Un poste de contrôle est alimenté en pièces toutes les 7 minutes. Ces pièces sont de quatre types qui nécessitent respectivement 4, 6, 8 et 9 minutes pour être contrôlés. Les fréquences des quatre types sont identiques.

1. A l'aide d'une file, simuler le fonctionnement du poste de contrôle pendant 1 heure, puis 80 heures.
2. Calculer le nombre de pièces en attente de contrôle
3. Déterminer la durée de séjour des pièces dans le poste de contrôle. Afficher celle de la dernière
4. Le contrôleur fait une pause de 15 minutes toutes les 4 heures. Déterminer le nombre d'heures effectives de travail

■

3. Exercice

■ Exercice .8.

1. Initialiser une variable μ (vous pouvez écrire mu) et ν (vous pouvez écrire nu). Importer le module random et mettre dans une arrivée,random.expovariate(mu) et dans une variable service random.expovariate(nu)
2. Faire une fonction qui prend en entrée un entier n et qui renvoie une liste de temps d'arrivée en ajoutant à chaque fois une autre variable random.expovariate(mu)
3. Faire une File (qui sera une file d'attente) avec les méthodes vus précédemment. On lui ajoutera comme attribut mu et nu.
4. Faire une classe Individu avec comme attribut un numéro (qui augmente chaque fois de 1) et un temps de service (que l'on définira plus tard) et un temps d'arrivée.
5. Faire une méthode qui prend en entrée un temps de fin t_{fin} qui simule les temps d'arrivée et de service et qui ajoute les individus qui arrivent et enlève les individus qui ont été servis. (Il peut être utile de faire "couler" du temps, faire une variable temps qui s'incrémente et qui permet de gérer les évène-

ments)

■