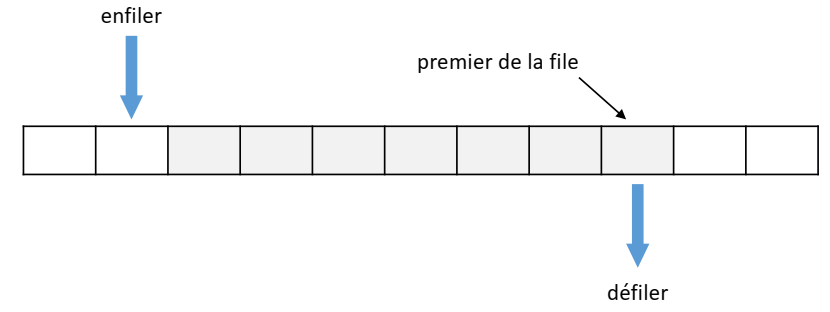
****La file est une structure de données qui donne accès uniquement au plus ancien élément inséré : le premier de file. On décrit souvent ce comportement par « premier entré, premier sorti » FIFO en anglais (First In First Out)

**Représentation** **:**

# TAD

**Opérations :**

* Constructeur : **File()**

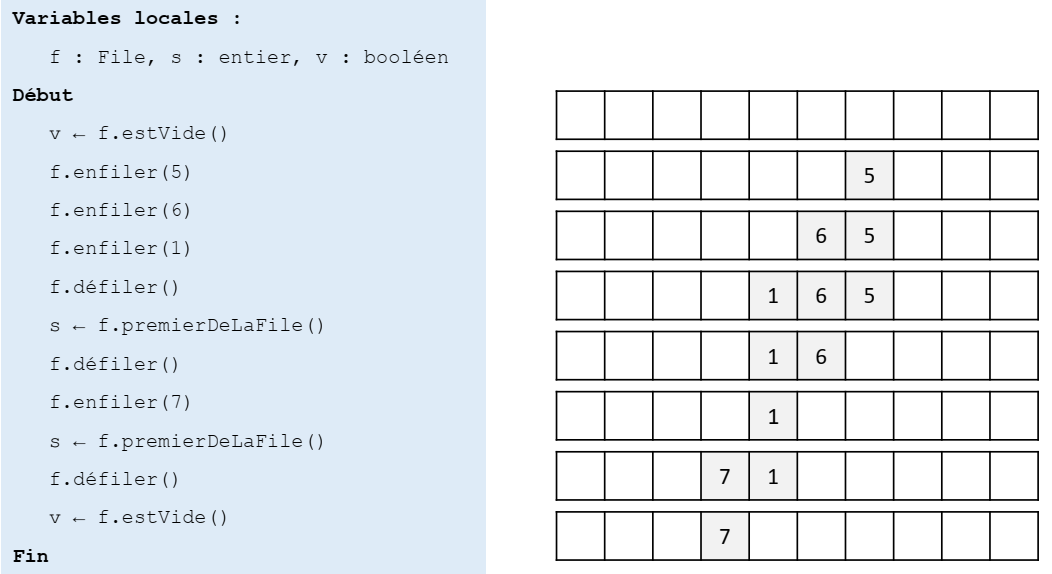
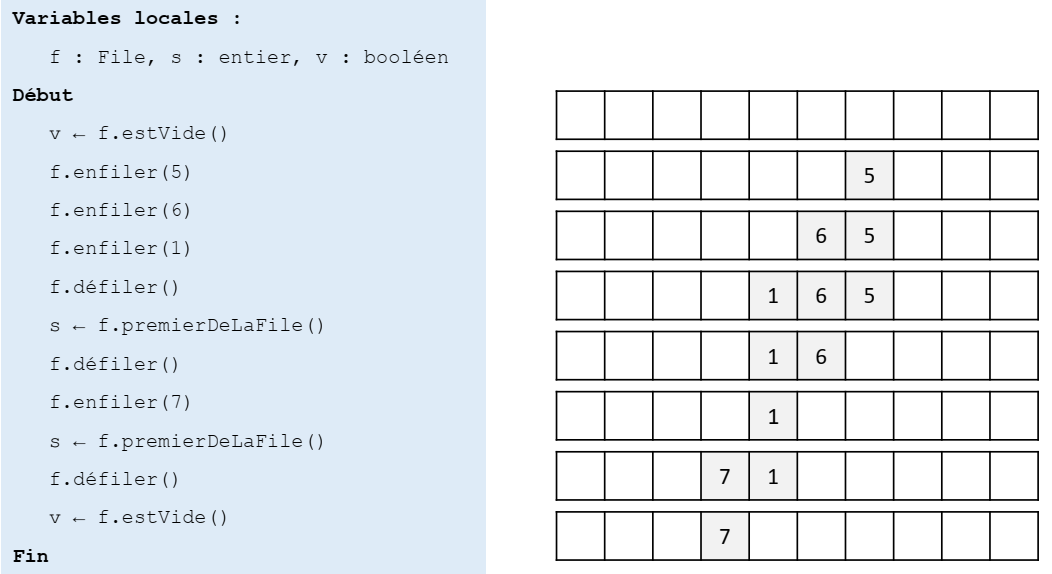
*Postconditions : La file est une file vide*

* Procédure **enfiler (e)**  
  *Postcondition : e est ajouté à la file*
* Procédure **défiler ()**  
  *Précondition : la file n’est pas vide  
  Postcondition : le premier de la file est défilé*

*Résultat : retourne le premier de la file*

* Procédure **vider ()**  
  *Postcondition : la file ne contient plus aucun élément, la file est une file vide*
* Fonction **estVide ()** : renvoie un booléen  
  *Résultat : retourne vrai si la file est vide, faux sinon*
* Fonction **premierDeLaFile ()** : renvoie tout type  
  *Précondition : la file n’est pas vide  
  Résultat : retourne le premier de la file*
* Fonction **longueur ()** : renvoie un entier  
  *Précondition : la file n’est pas vide  
  Résultat : retourne le nombre d’éléments dans la file*

**Exemple d’utilisation :**

**

# Applications des files

Beaucoup de systèmes informatiques reposent sur le principe de files implémentées dans des buffers par exemple :

* Mémorisation de transactions qui doivent attendre pour être traitées :

Les serveurs d'impression, qui doivent traiter les requêtes dans l'ordre dans lequel elles arrivent.

Communication Bluetooth, série, …

* Moteur multitâche dans un système d’exploitation, pour l’allocation du temps processeur. Il doit accorder du temps machine à chaque tâche.
* Gestion des évènements (ex. frappe clavier)
* Un algorithme de parcours en largeur d’un arbre utilise une file pour mémoriser les nœuds visités.

# Implémentation en POO

Deux implémentations possibles, par exemple :

* Par une liste chaînée : premier de la file = tête (ou queue) de liste

enfiler = ajouter en queue (ou tête)

défiler = supprimer la tête (ou queue)

➢ Le coût (complexité) dépend de l’implémentation du type de liste chainée

Simplement chaînée : linéaire O(n)

Doublement chaînée : constant O(1)

* Par un tableau dynamique (list Python) :

- Premier de la file = première case du tableau

enfiler = ajouter en dernière position

défiler = supprimer l’élément en première position

➢ Le coût (complexité) : linéaire O(n) puisque :

linéaire O(n) pour la suppression en 1ère position de la list python

amorti constant en O(1) pour l’ajout en fin de la list python

*Coût amorti signifie que l’on ne prend pas en compte les cas rare et coûteux en opérations et donc en temps comme par exemple quand la liste python est pleine et qu’il faut la recopiée en mémoire dans une liste de taille le double.*

- Premier de la file = dernière case du tableau

enfiler = ajouter en première position

défiler = supprimer l’élément en dernière position

➢ Le coût (complexité) : linéaire O(n) puisque :

linéaire O(n) pour l’ajout en début de la list python

constant en O(1) pour la suppression en fin de list python

*Coût amorti signifie que l’on ne prend pas en compte les cas rare et coûteux en opérations et donc en temps comme par exemple quand la liste python est pleine et qu’il faut la recopiée en mémoire dans une liste de taille le double.*

**Choisissons la list python pour implémenter une file:**

Ouvrir le fichier *File\_depart.py* , puis :

1. Enregistrez sous *File.py*, est implémenter les méthodes manquantes : enfiler, défiler, vider, estVide, premierDeLaFile, longueur, et la surcharge de l’affichage