Structures abstraites 1
TD : Implémentation d'une file

Rappel : Une structure de données est une organisation logique des données permettant de simplifier ou d'accélérer leur traitement. Une file est une liste particulière dans laquelle le premier élément arrivé est le premier sorti (principe **FIFO**)

La structure de file est un concept abstrait défini par son interface.

Je vous propose donc ici deux implémentations possibles. L'idée principale étant que l'interface (les fonctions de base) pourra être utilisée indépendamment de l'implémentation choisie. Seule l'efficacité des opérations sera modifiée.

1) Préliminaires :

Sans utiliser le cours, rappelez les trois primitives de l'interface d'une File. Relisez le cours si vous ne trouvez pas.

2) Implémentation n°1

Dans cette implémentation, nous utiliserons une simple liste pour représenter la File.
Nous utiliserons donc encore append(x) et pop(0) pour "enfiler" et "défiler".

Voici les primitives :

```
def file():
    """ crée une file vide """
    return []

def est_vide(f):
    """renvoie True si la file est vide
     et False sinon"""
    return f == []

def enfiler(f, x):
    """Ajoute l'élément x à la file f"""
    f.append(x)

def defiler(f):
    """enlève et renvoie le premier élément de la file f"""
    assert not est_vide(f), "File vide"
    return f.pop(0)
```

❐ — Exercice **1** —

- Télécharger le programme TD_File.py depuis bouillotvincent.github.io et tester les instructions suivantes:
- Dans le programme principal, à l'aide d'une boucle while, écrivez un algorithme permettant de faire défiler totalement la file.

```
f = file()
for i in range(5):
    enfiler(f,2*i)
    print(f)
a = defiler(f)
print(a)
print(f)
print(est_vide(f))
```

❐ — Exercice **2** —

- Réaliser les fonctions **taille(p)** et **sommet(p)** qui retourne respectivement la taille de la file et le sommet (= l'élément qui sort en premier) de la file, sans le supprimer. On s'autorise toutes les instructions Python !

3) Implémentation n°2

Nous allons créer une classe File pour implémenter cette structure.
Voici la classe que nous allons utiliser :

```
class File:
    """ classe File :
    création d'une instance File avec à partir d'une liste """
    def __init__(self):
        """Initialisation d'une file vide"""


    def est_vide(self):
        """teste si la file est vide"""
        return

    def defiler(self):
        """défile"""

        return

    def enfiler(self, x):
        """enfile"""
```

❐ — Exercice **3** —

En vous aidant de la première implémentation et de ce que nous avons vu sur les Piles, complétez la classe File dont le prototype a été fourni ci-dessus.

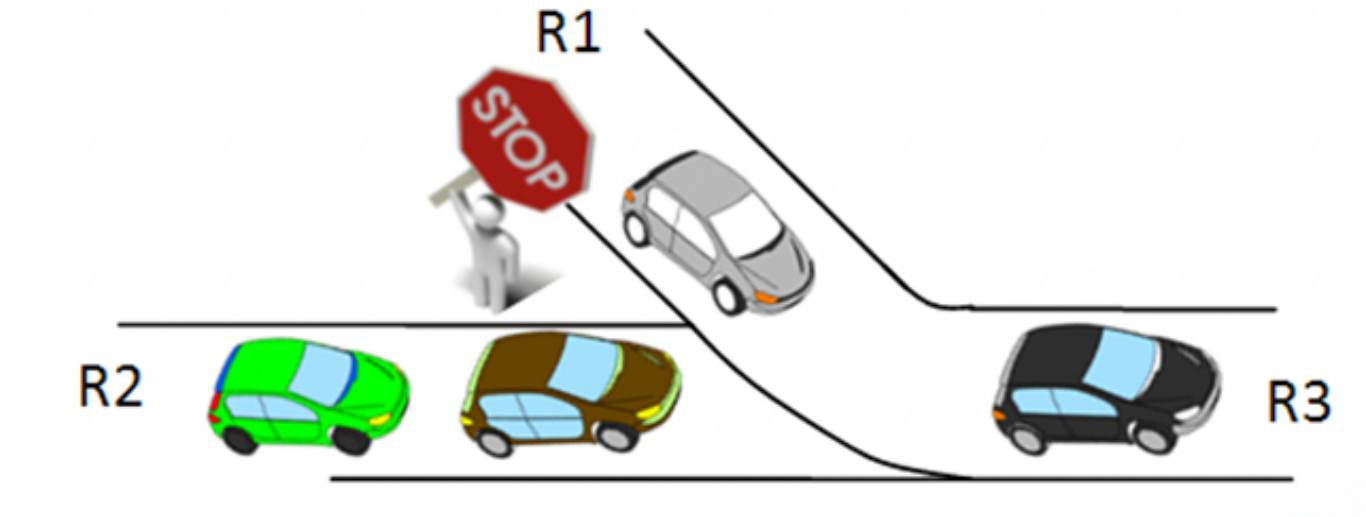
❐ — Exercice **4** —

Dans le fichier TD_File.py, écrire les instructions permettant de :

- ➡ créer une file
- ➡ la remplir avec les entiers 0,2,4,8,16,32
- ➡ la faire afficher
- ➡ "défiler" toute la file en faisant afficher chaque élément récupéré et la file

4) Un exemple d'utilisation : Croisement routier

Pour simuler un croisement routier à sens unique, on utilise 3 files f1, f2 et f3 qui représentent les voitures arrivant sur des routes R1 et R2, et les voitures partant sur la route R3.
La route R2 a un STOP. Les voitures de la file f2 ne peuvent avancer que s'il n'y a aucune voiture sur la route R1, donc dans la file f1.



On souhaite écrire un algorithme qui simule le départ des voitures sur la route R3, modélisée par la file f3.
Dans la file f1, on représentera la présence d'une voiture par le nombre 1 et l'absence de voiture par 0
Dans la file f2, on représentera la présence d'une voiture par le nombre 2 et l'absence de voiture par 0
On n'utilisera que les primitives "enfiler", "defiler", "sommet" et "est_vide".

Exemple :
On testera l'algorithme sur l'exemple suivant :
f1 : tête <–[0, 1, 1, 0, 1]<– queue
f2 : tête <–[0, 2, 2, 2, 0, 2, 0]<– queue
f3 : tête <–[0, 1, 1, 2, 1, 2, 2, 0, 2, 0]<– queue

Question 1
Que doit faire l'algorithme si les deux sommets des files sont à 0?

Question 2
Que doit faire l'algorithme si le sommet de f1 est à 1 et celui de f2 à 2?

Question 3
a) Que doit faire l'algorithme si le sommet de f1 est à 1 et celui de f2 à 0?
b) Que doit faire l'algorithme si le sommet de f1 est à 0 et celui de f2 à 2?

Question 4
Que doit faire l'algorithme si l'une des deux files est vide?

Question 5
Écrire un algorithme qui modélise ce carrefour, on utilisera une fonction `croisement(f1,f2)` qui prend en paramètres deux files f1 et f2 et qui retourne une file f3 contenant la file f3 des voitures sur la route R3.