Exercice 1 : Dans une version simplifiée, nous pourrions utiliser les structures proposées par Python pour implémenter les piles et files.

1. Créer une classe *Pile* qui possédera un attribut *donnees*. Cet attribut sera une liste Python vide.
2. Implémenter les méthodes d'une pile.
3. Implémenter une classe *File* sur le même modèle.

Exercice 2 : L'historique d'un navigateur web conserve la page récemment visitée. De plus, le bouton *retour arrière* permet de revenir à la page précédemment explorée.

1. Quelle structure de données permet d'implémenter le comportement de l'historique de navigation ?
2. Le fichier *historique.csv* contient une liste d'adresses web récemment visitées. À l'aide de la bibliothèque *csv* créer la classe **Historique** adaptation de la structure de données évoquée dans la première question. Créer une instance de *Historique* et le remplir avec les données du fichier *csv*.
3. Écrire une méthode **retour() → str** qui renvoie l'adresse du dernier site visité.
4. Écrire une méthode **nouvelle_adresse(lien : str) → None** qui stocke l'adresse *lien* dans l'historique. La bibliothèque *datetime* et sa méthode *now* permettra de créer la date.

Exercice 3 : Il est possible de créer une *file* avec deux *piles* en appliquant l'algorithme suivant :

— Enfiler un élément dans la pile gauche.
— Défiler un élément dans la pile droite. Si la pile est vide, dépiler la pile gauche dans la pile droite.

1. Vérifier sur le papier le fonctionnement de la file.
2. Implémenter cette classe **File**. Cette file contiendra des entiers.

Exercice 4 : Flavius Josèphe était un historien juif du premier siècle. Il participa aux révoltes contre les Romains et fut assiégé avec quarante de ses compatriotes dans la forteresse de Jotapata en 67. Les extrémistes du groupe persuadèrent l'ensemble de se tuer pour ne pas tomber aux mains des Romains. Ne partageant pas ce point de vue mais n'osant s'opposer au groupe, Josèphe proposa que l'on se mette en cercle et que chaque troisième personne soit tuée, la dernière devant se suicider.
En utilisant une *file* trouver la position à choisir parmi les 41 soldats pour être le dernier éliminé.

Exercice 5 : Un EDI se charge de vérifier si le code écrit par le programmeur est correctement parenthésée, c'est à dire si à chaque parenthèse ouvrante correspond une parenthèse fermante.

— Quelle structure de données va-t-on utiliser ?
— Écrire la fonction **bien_parenthesee(code : str) → bool** qui renvoie *True* si la ligne de code *code* est correctement parenthésée.

Exercice 6 : L'écriture polonaise inverse des expressions arithmétiques place l'opérateur après ses opérandes. Cette notation ne nécessite aucune parenthèse ni aucune règle de priorité. Ainsi l'expression polonaise inverse

$$1\ 2\ 3\ \times\ +\ 4\ \times$$

désigne l'expression traditionnelle

$$(1 + 2 \times 3) \times 4$$

En utilisant une pile pour stocker les résultats intermédiaires, il est facile de calculer l'expression :

— Si on a un nombre on le place sur la pile.
— Si on a un opérateur on récupère les deux nombres du sommet de la pile, on leur applique l'opérateur et on replace le résultat sur la pile.

Écrire la fonction **polonaise(chaine : str) → int** qui calcule l'expression *chaine*. Les éléments de la chaîne seront séparés par un espace. On n'utilisera que l'addition et la multiplication.

Christophe Viroulaud