TD - 02

Exercices d'applications

Savoirs et compétences :

- ❒ *Alg – C16 : Piles - Algorithmes de manipulation : fonctions «push» et «pop».*

Exercice 1 - Construction d'une pile

On souhaite réaliser les fonctions de base permettant la gestion d'une pile. On souhaite que celle-ci soit implémentée sous la forme d'un tuple. La taille de la pile doit être limitée à 100 données.

Rappel Affectation d'un tuple :

```
>>> t = (1,2,3)
```

Question 1 *Implémenter la fonction* `est_vide` *permettant de vérifier qu'une pile est vide. Les spécifications de la fonction seront les suivantes :*

▪ Python
```
def est_vide(pile):
    """
    Vérifie si la pile est vide.
    Entrée :
     * pile(tuple)
    Sortie :
     * retourne True si la pile est vide, False
     sinon
    """
```

Question 2 *Implémenter la fonction* `est_pleine` *permettant de vérifier qu'une pile est pleine. Les spécifications de la fonction seront les suivantes :*

▪ Python
```
def est_vide(pile,nb):
    """
    Vérifie si la pile est pleine.
    Entrée :
     * pile(tuple)
     * nb(int) : nombre d'éléments maximum dans une pile
    Sortie :
     * retourne True si la pile est pleine, False
     sinon
    """
```

Question 3 *Créer la fonction* `push` *permettant d'ajouter un élément au sommet de la pile. Les spécifications de la fonction devront être indiquées.*

Question 4 *Créer la fonction* `pop` *permettant de supprimer l'élément au sommet de la pile et de le renvoyer. Les spécifications de la fonction devront être indiquées.*

Exercice 2 – Notation polonaise inversée

La notation polonaise inversée permet de réaliser des opérations arithmétiques sans utiliser de parenthèses. On parle aussi d'expressions **postfixées**. Dans cette notation, l'opérateur (+, -, *, /) suit toujours l'opérande (nombres).

Dans cette notation, on doit évaluer une expression composée d'opérateurs et d'opérandes. L'expression est lue de gauche à droite en suivant les règles suivantes :

- si on lit une opérande, on l'empile ;
- si on lit un opérateur, on dépile deux opérandes, on réalise le calcul et on empile le résultat.

▪ **Exemple** Pour calculer $7 \cdot 2$, on évalue l'expression $7\,2\,*$.

Pour calculer $((1+2)\cdot 4)+3$, on évalue l'expression $1\,2\,+\,4\,*\,3\,+$. ▪

R Le principal avantage de cette méthode est la suppression totale des parenthèses. En contrepartie, elle nécessite une petite gymnastique intellectuelle pour les néophytes que nous sommes.

Cette notation est utilisée dans certaines calculatrices HP et dans certains utilitaires (programme calc d'Emacs, description des bibliographies dans LaTeX *etc.*).

Dans cet exercice, la pile sera implémentée sous forme d'une liste d'opérande et d'opérateurs. On disposera donc des méthodes associées à la gestion des listes.

Question 5 *Créer la fonction* `est_nombre` *permettant de savoir si un élément de la pile est un nombre ou non.*

Question 6 *Créer la fonction* `est_operation` *permettant de savoir si un élément de la pile est une opération ou non.*

Question 7 *Créer la fonction* `evaluer` *permettant d'évaluer une expression postfixée. Les spécifications sont les suivantes :*

▪ **Python**

```
def evaluer(exp):
    """
    Évaluer le résultat d'une opération post-fixée.
    Entrée :
     * ex(lst) : liste d'opérateurs et d'opérandes
    Sortie :
     * res(flt) : résultat du calcul de l'expression.
    """
```