TD: piles et parenthésage:

Le but de cet exercice est d'écrire un programme en Python capable de vérifier si une expression est bien parenthésée ou pas. Les trois types de parenthésage qui seront pris en compte sont les parenthèses ( ), les crochets [ ] et les accolades { }.

Votre programme doit par exemple retourner True pour les expressions suivantes:

- $[a + (b + c)]$
- $[((5 * 3) + (2 * 10))/2]$

et False pour les expressions suivantes:

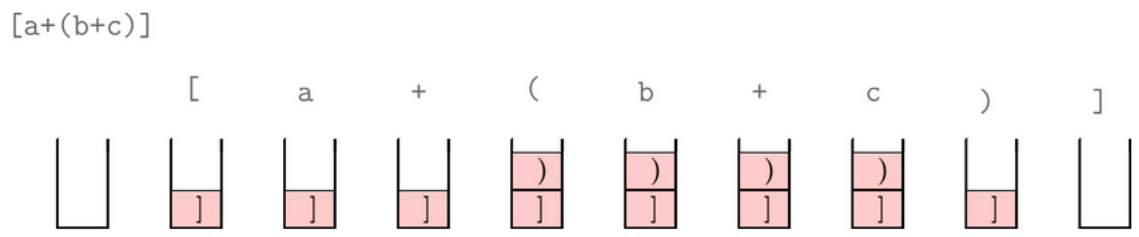
- $[a + (b + c)$
- $]a + (b + c)]$

La façon la plus simple de réaliser un tel programme est d'utiliser une pile. Au départ la pile est vide. Ensuite, on lit l'expression, caractère par caractère:

- Si le caractère est ni une parenthèse, ni un crochet, ni une accolade, on ne fait rien.
- Si le caractère lu est un symbole ouvrant, alors on empile le caractère fermant correspondant.
- Si le caractère lu est un symbole fermant:
  - Si la pile est vide, on ne fait rien,
  - Sinon, si le caractère lu est identique au sommet de la pile, on dépile ce caractère.

Nous pouvons constater que lorsque le parenthésage est correct, la pile sera vide après que tous les caractères sont lus.

Dans l'exemple suivant vous pouvez observer l'évolution de la pile à chaque lecture d'un nouveau caractère de l'expression $[a + (b + c)]$.



1. Justifier que la pile peut être modélisée par une liste lorsqu'on utilise les méthodes suivantes. Préciser alors comment accéder à la valeur au sommet de la pile.

```
>>> L.append('e')
>>> print(L)
['a', 'b', 'c', 'e']
```

```
>>> L = ['a', 'b', 'c', 'd']
>>> L.pop()
'd'
>>> print(L)
['a', 'b', 'c']
```

2. Écrire une fonction ouvrante(c) qui prend en entrée un caractère c et qui renvoie True si c est un symbole ouvrant.

3. Écrire une fonction fermante(c) qui prend en entrée un caractère c. Si ce caractère est un symbole ouvrant, alors la fonction doit renvoyer le symbole fermant correspondant. Dans le cas contraire elle doit renvoyer None.

4. Écrire et tester la fonction qui vérifie le parenthésage.