Piles : Exercices d'application

Exercice 1 – le bon parenthésage : On souhaite écrire une fonction permettant d'abord de vérifier si une expression est bien ''parenthèsée'', et ensuite de renvoyer la liste (ou la pile) des couples correspondant aux indices de chaque parenthèse ouvrante et fermante qui lui correspond.

Algo1 : avec juste le retour d'un booléen

Algo2 : avec le retour (si bien parenthèsée) d'une liste contenant les indices des couples de parenthèses comme les résultats à droite

Algo3 : on rajoute des crochets [ ] et des accolades {} à l'expression

Expression	Résultat
''	True, []
'()()'	True, [(0, 1), (2, 3)]
'(())()'	True, [(1, 2), (0, 3), (4, 5)]
')('	False, []
'(()'	False, []

Principe de l'algorithme de l'Algo1 :

On crée une ……,
On parcourt l'expression de gauche à droite.
Si on rencontre une parenthèse ouvrante "(" alors on ……,
Si on rencontre une parenthèse fermante ") " alors
Si la pile n'est pas vide on …….
Sinon on retourne ……
A la fin la pile doit être ……..

1. Ecrire son code en Python sur spyder ou jupyter , en utilisant l'une des implémentation de la structure pile que vous avez déjà réalisées, et tester-le sur des expressions.
2. Réaliser les programmes Algo2 et Algo3

Exercice 2 : la notation polonaise inversée (fini les parenthèses !):

Définition : La notation polonaise inverse (NPI) (en anglais RPN pour Reverse Polish Notation), également connue sous le nom de notation post-fixée, permet d'écrire de façon non ambiguë les formules arithmétiques sans utiliser de parenthèses.

Historique : Elle est dérivée de la notation polonaise utilisée pour la première fois en 1924 par le mathématicien polonais Jan Lukasiewicz, la NPI a été inventée par le philosophe et informaticien australien Charles Leonard Hamblin dans le milieu des années 1950, pour permettre les calculs sans faire référence à une quelconque adresse mémoire. À la fin des années 1960, elle a été diffusée dans le public comme interface utilisateur avec les calculatrices de bureau de Hewlett-Packard.

Exemple: Voici une expression algébrique $\left((3+4)-2\right)^3$
En notation in-fixée (celle que nous utilisons la plupart du temps) : ((3 + 4) - 2)^3
En notation post-fixée (NPI) cela s'écrit : 3 4 + 2 - 3 ^

Evaluer (=calculer) le résultat à la main les expressions :

3 4 * 2 - 3 + donne …….. ; 3 4 2 - + 3 * donne …… ; 3 4 2 3 - + * donne ……

L'objectif est d'écrire un programme pour évaluer des expressions écrites en notation polonaise inverse.
Les expressions en NPI seront représentées par des tableaux contenant des entiers et des caractères.
l'expression précédente sera représentée par le tableau : tab=[3,4,'+',2,'*',3,'^']

Principe du programme: Une pile vide est créée, le tableau est parcouru de gauche à droite

- Chaque nombre rencontré est empilé.
- Si l'élément rencontré est un opérateur, on dépile le sommet et le sous-sommet puis on empile le résultat du calcul (sous-sommet "opération" sommet)
- Si l'élément rencontré est une fonction (comme : p), on dépile le sommet et on calcule la valeur de la fonction pour le sommet, puis on empile le résultat

Coder ce programme en Python, et évaluer différentes expressions.
On entrera une chaine de caractères avec des nombres et les opérateurs séparés des espaces et cette chaine sera convertie en tableau avec la commande : tab=chaine.split(sep=" ")