T NSI	Structures de données	*TP_POO : Parcourir les arbres*

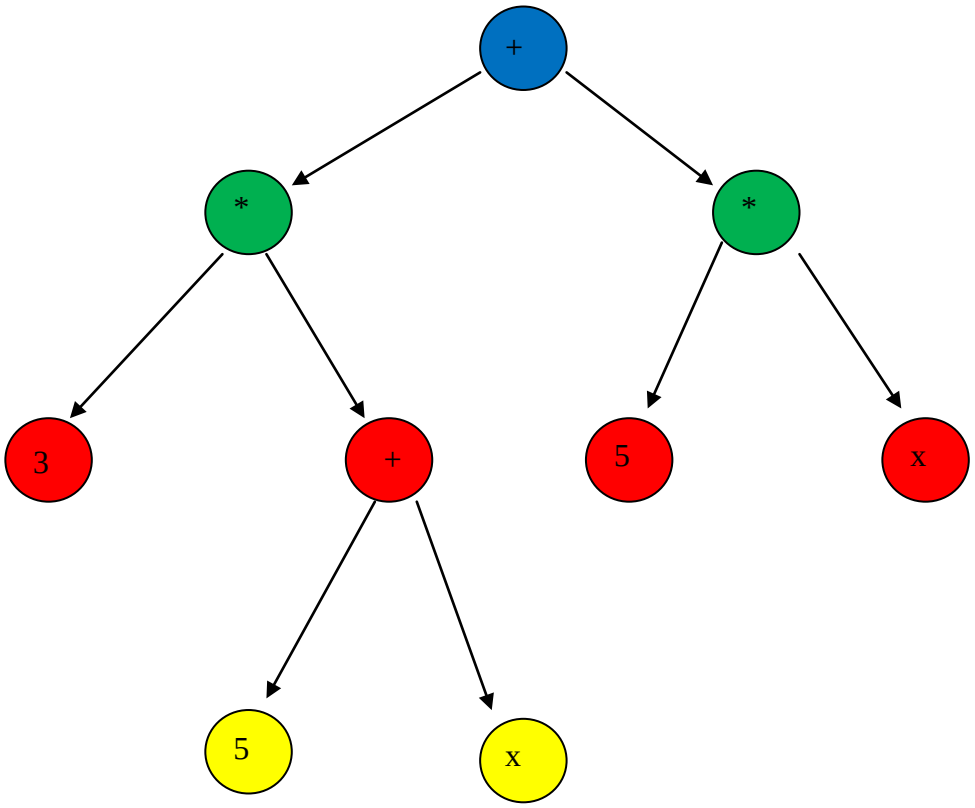
Objectifs *: Parcours sur les arbres*

I. Expression arithmétique

Une expression arithmétique, peut être représentée par un arbre dont les nœuds internes portent les opérateurs et les feuilles des symboles de variables, ou des constantes.

Ouvrir le fichier EvaluerExpression.py

Q1.Dessiner l'arbre1
Q2.Créer l'arbre2 :



Parcours des arbres : Dans un parcours, tous les nœuds de l'arbre sont visites.

- Dans un parcours prefixe (preorder traversal), chaque nœud est visité avant que ses enfants soient visités.
- Dans un parcours infixe , chaque nœud est visité après la visite de son fils gauche (et avant la visite de son fils droit).
- Dans un parcours postfixe (postorder traversal), chaque nœud est visité après que ses enfants sont visités.

Q3.Ecrire le parcours Préfixe de l'arbre 2.
Q4.Pour le parcours infixe, on ajoute la convention suivante : on ajoute une parenthèse ouvrante à chaque fois qu'on entre dans un sous-arbre et on ajoute une parenthèse fermante lorsqu'on quitte ce sous- arbre.

Q5.Ecrire le parcours postfixe de l'arbre 2 (notation polonaise).

Q6.Ajouter une méthode postfixe.
Q7.Ajouter une méthode prefixe.

Q8. Ecrire une méthode evaluation qui prend l'arbre d'une expression ne contenant que des constantes en argument et retourne la valeur de celle-ci.

II. Le tri du bijoutier :

On dispose d'une liste de nombres. Par exemple, la liste 7, 9, 3, 5, 4, 1, 8.

On associe à chaque élément n de la liste un nœud v (initialisation : père[v]=NIL, fils_gauche[v]=NIL, fils_droit[v]=NIL, clef[v]=n).

1. Dresser l'arbre obtenu en appliquant l'algorithme Arbre_Insérer aux éléments de la liste (dans l'ordre de la liste) en partant d'un arbre vide pour le premier élément, chaque appel à l'algorithme modifiant l'arbre.

Pseudo-code

```
Arbre_Insérer ( Arbre T, noeud z )

        y : = NIL
        x := racine[T]
        TantQue x distinct de NIL faire
                    y:=x
                    Si clef[z]< clef[x]
                    alors x:=fils_gauche[x]
                    sinon x:= fils_droit[x]
                    FinSi
        FinTantQue

        père [ z ] : = y
        Si y=NIL
        alors racine [T]: = z
        sinon
                Si clef[z]< clef[y]
                alors fils_gauche[y]:=z
                sinon fils_droit[y]:=z
        FinSi
```

2. L'un des parcours postfixe, infixe, préfixe de la liste trie la liste. Lequel ?