Implémentation d’un arbre binaire

Les arbres binaires peuvent être implémentés de la même manière que les listes chaînées. Dans celles-ci chaque maillon posséde au plus un maillon suivant. Pour l’arbre, chaque nœud posséde au plus deux nœuds suivants, un gauche et un droit. Dans ce TP, la structure de donnée sera implémentée à l’aide de la programmation objet.

1. Implémenter une classe **Node** représentant un nœud et possédant les 3 attributs suivants :

- **value** : valeur de l’étiquette du nœud, de type quelconque, pas de valeur par défaut.
- **left** : nœud suivant gauche, de type **Node**, **None** par défaut.
- **right** : nœud suivant droit, de type **Node**, **None** par défaut.

Soit l’arbre binaire instancié par le code ci-dessous.

```
noeud_A = Node("A", None, None)
noeud_E = Node("E", None, None)
noeud_I = Node("I", None, None)
noeud_K = Node("K", None, None)
noeud_L = Node("L", None, None)
noeud_J = Node("J", None, noeud_K)
noeud_B = Node("B", noeud_A, None)
noeud_D = Node("D", noeud_J, noeud_E)
noeud_H = Node("H", noeud_L, noeud_I)
noeud_C = Node("C", noeud_B, noeud_D)
noeud_G = Node("G", None, noeud_H)
noeud_F = Node("F", noeud_C, noeud_G)
arbre = noeud_F
```

2. Le représenter sur papier. Identifier sa racine et ses feuilles. Justifier.

Soit l’arbre binaire instancié par le code ci-dessous.

```
arbre = Node('manchot')
arbre.left = Node('poule')
arbre.left.left = Node('chat')
arbre.left.left.left = Node('rat')
arbre.left.left.right = Node('chien')
arbre.left.right = Node('pigeon')
arbre.right = Node('canari')
arbre.right.left = Node('léopard')
arbre.right.right = Node('vache')
arbre.right.right.right = Node('gazelle')
```

3. Le représenter sur papier. Identifier sa racine et ses feuilles. Justifier.

Soit l’arbre binaire instancié par le code ci-dessous.

arbre = Node(7, Node(11, None, Node(2, None, None)), Node(1, Node(9, None, None), Node(13, None, None)))

4. Le représenter sur papier. Identifier sa racine et ses feuilles. Justifier.

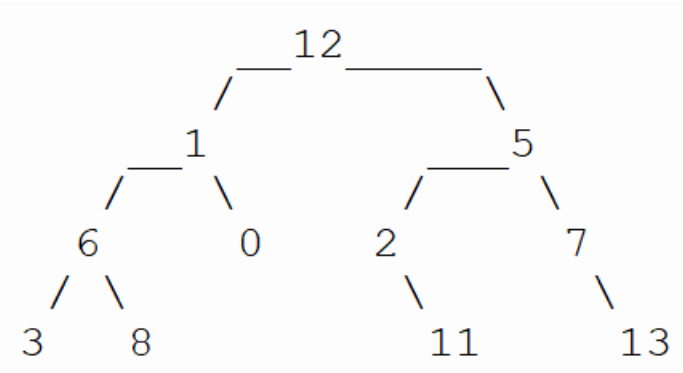
Le module **Python binarytree** implémente les arbres binaire exactement de la même manière que ci-dessus. De plus il possède de nombreuses fonctions pour les étudier et les afficher.

5. Importer sa classe **Node** avec **from binarytree import Node**.

6. Utiliser la fonction **print** pour afficher les trois arbres précédents après les avoirs codés et ainsi vérifier vos réponses.

7. Quel est le résultat classique de la fonction **print** sur un objet ? Pourquoi le résultat est-il différent avec un objet du type **Node** du module **binarytree** ?

8. Implémenter l’arbre ci-dessous avec les trois méthodes présentées précédemment. Vérifier.



9. Tester si l’arbre de la question **8** est un arbre équilibré grâce à son attribut **is_balanced**. Construire en Python un arbre permettant de vérifier le cas contraire et recopier vos exemples.

10. Tester si l’arbre de la question **8** est un arbre (presque) complet grâce à son attribut **is_complete** . Construire en Python un arbre permettant de vérifier le cas contraire et recopier vos exemples.

11. Tester si l’arbre de la question **8** est un arbre parfait grâce à son attribut **is_perfect** . Construire en Python un arbre permettant de vérifier le cas contraire et recopier vos exemples.

12. Tester si l’arbre de la question **8** est un arbre strict grâce à son attribut **is_strict** . Construire en Python un arbre permettant de vérifier le cas contraire et recopier vos exemples.