Binary Tree

March 16, 2021

[ ]:

```python
class Node(object):
    def __init__(self, value):
        self.value = value
        self.left = None
        self.right = None
```

[ ]:

```python
class BinaryTree(object):
    def __init__(self, root):
        self.root = Node(root)

    def print_tree(self, traversal_type):
        if traversal_type == "preorder":
            return self.preorder_print(self.root, "")
        elif traversal_type == "inorder":
            return self.inorder_print(self.root, "")
        elif traversal_type == "postorder":
            return self.postorder_print(self.root, "")

        else:
            print("Traversal type " + str(traversal_type) + " is not supported.
↪")
            return False

    def preorder_print(self, start, traversal):
        """Root->Left->Right"""
        if start:
            traversal += (str(start.value) + "-")
            traversal = self.preorder_print(start.left, traversal)
            traversal = self.preorder_print(start.right, traversal)
        return traversal

    def inorder_print(self, start, traversal):
        """Left->Root->Right"""
        if start:
            traversal = self.inorder_print(start.left, traversal)
            traversal += (str(start.value) + "-")
            traversal = self.inorder_print(start.right, traversal)
        return traversal

    def postorder_print(self, start, traversal):
        """Left->Right->Root"""
        if start:
            traversal = self.postorder_print(start.left, traversal)
            traversal = self.postorder_print(start.right, traversal)
            traversal += (str(start.value) + "-")
        return traversal
```

```
           1
       /       \
      2          3
     /  \      /   \
    4    5     6   7
```

[ ]:

```
# Set up tree:
tree = BinaryTree(1)
tree.root.left = Node(2)
tree.root.right = Node(3)
tree.root.left.left = Node(4)
tree.root.left.right = Node(5)
tree.root.right.left = Node(6)
tree.root.right.right = Node(7)
```

[ ]:

```
print(tree.print_tree("preorder"))
```

[ ]:

```
print(tree.print_tree("inorder"))
```

[ ]:

```
print(tree.print_tree("postorder"))
```

[ ]: