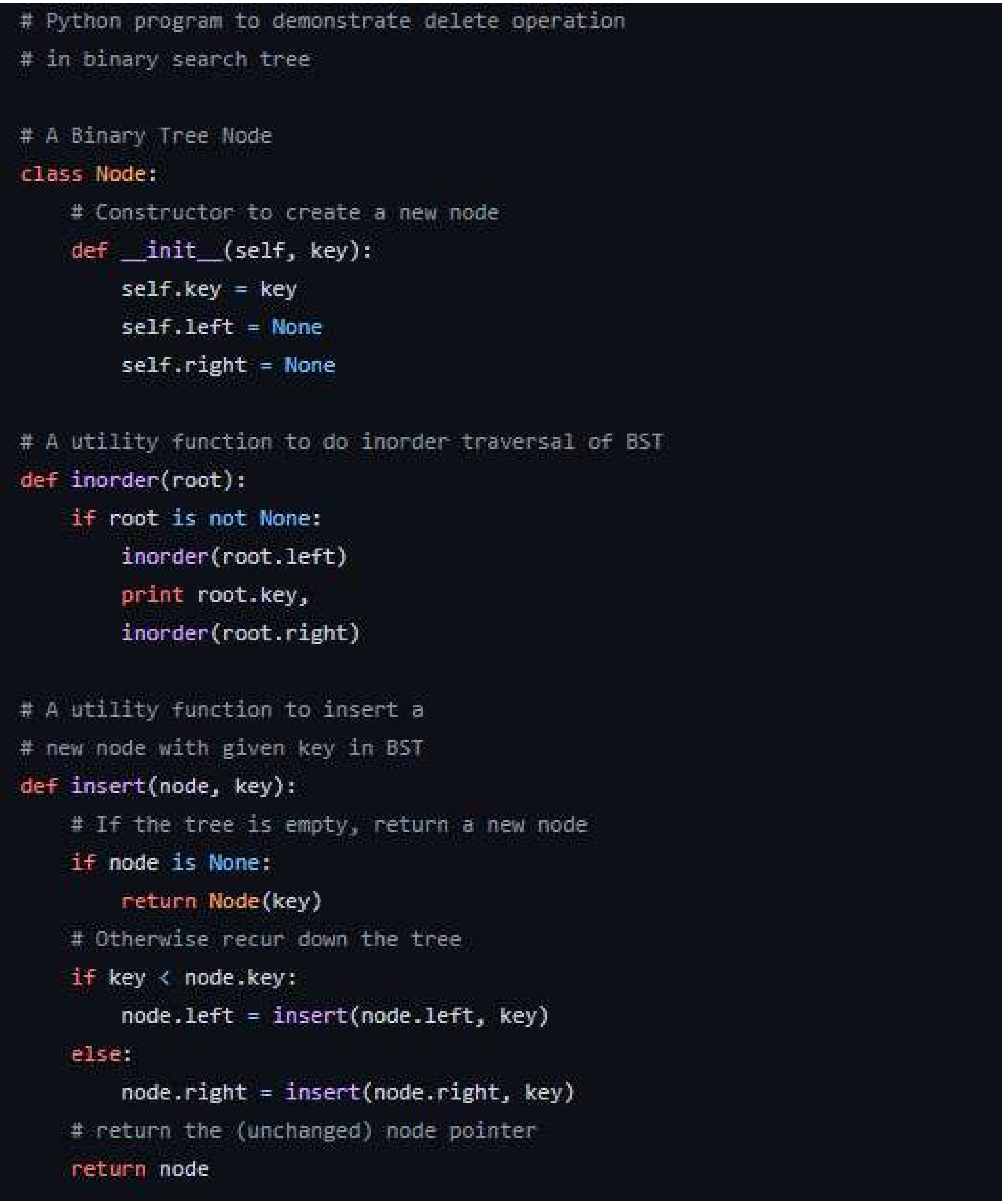
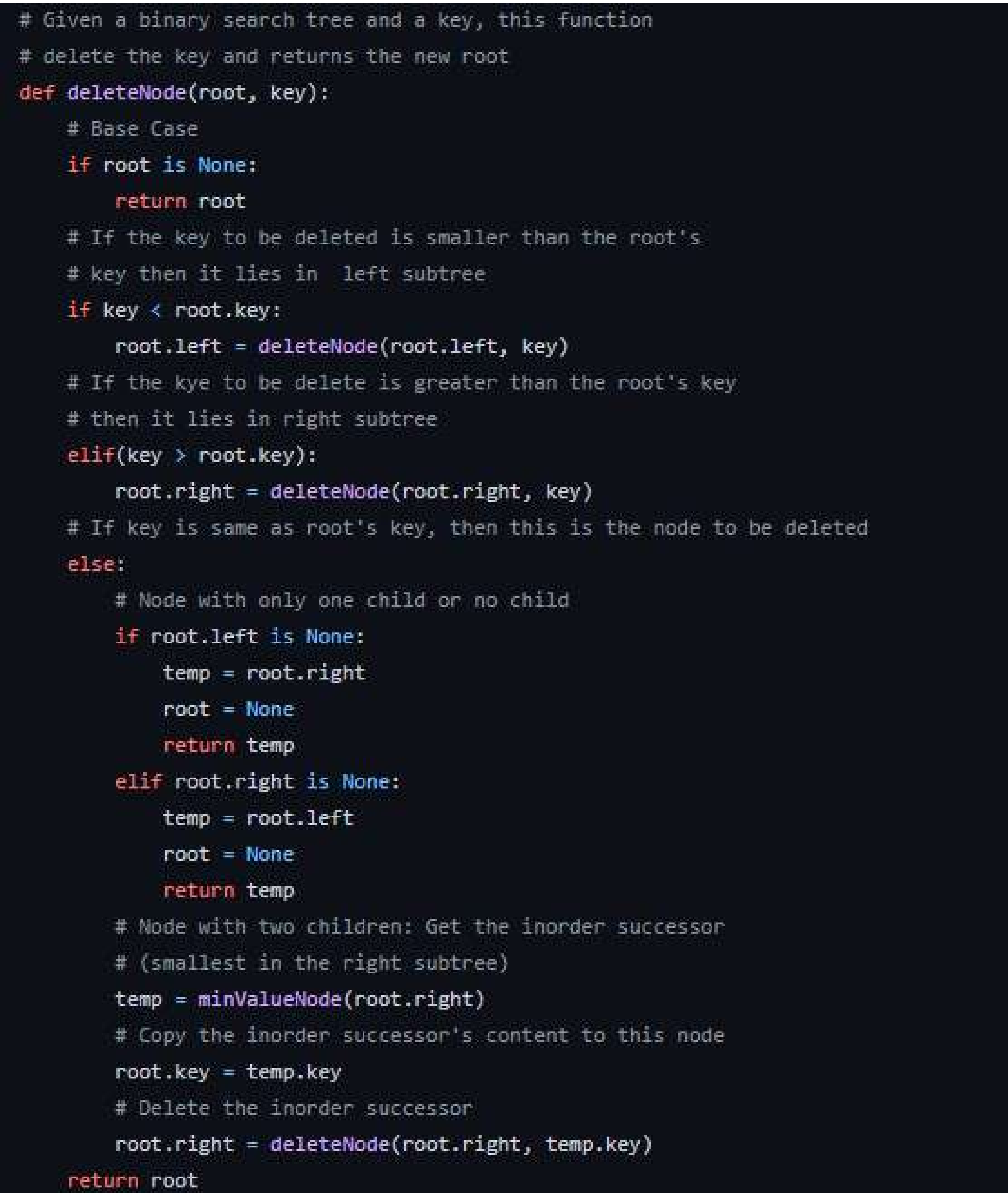
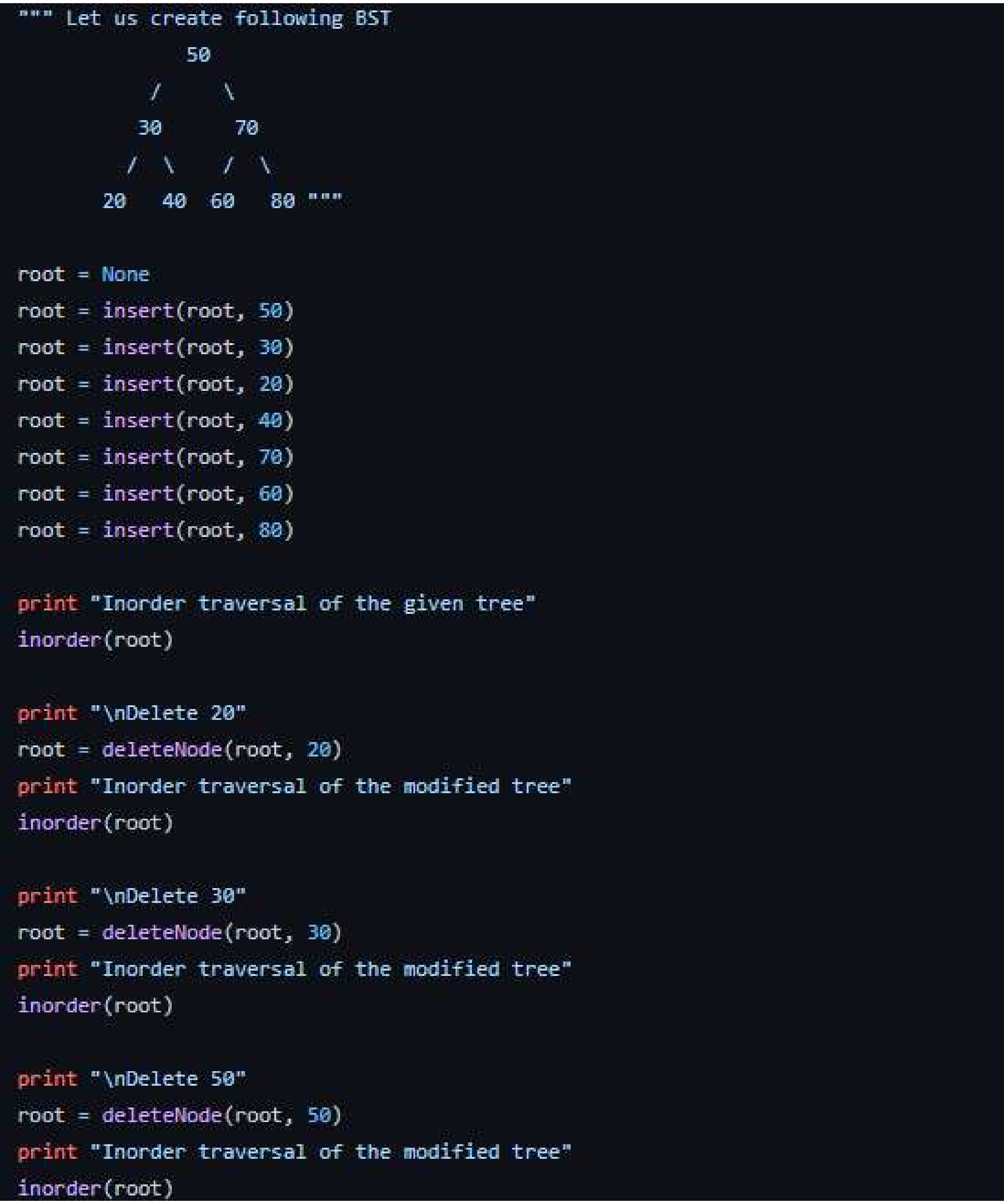


```python
# Python program to demonstrate delete operation
# in binary search tree

# A Binary Tree Node
class Node:
    # Constructor to create a new node
    def __init__(self, key):
        self.key = key
        self.left = None
        self.right = None

# A utility function to do inorder traversal of BST
def inorder(root):
    if root is not None:
        inorder(root.left)
        print root.key,
        inorder(root.right)

# A utility function to insert a
# new node with given key in BST
def insert(node, key):
    # If the tree is empty, return a new node
    if node is None:
        return Node(key)
    # Otherwise recur down the tree
    if key < node.key:
        node.left = insert(node.left, key)
    else:
        node.right = insert(node.right, key)
    # return the (unchanged) node pointer
    return node
```



```python
# Given a binary search tree and a key, this function
# delete the key and returns the new root
def deleteNode(root, key):
    # Base Case
    if root is None:
        return root
    # If the key to be deleted is smaller than the root's
    # key then it lies in  left subtree
    if key < root.key:
        root.left = deleteNode(root.left, key)
    # If the kye to be delete is greater than the root's key
    # then it lies in right subtree
    elif(key > root.key):
        root.right = deleteNode(root.right, key)
    # If key is same as root's key, then this is the node to be deleted
    else:
        # Node with only one child or no child
        if root.left is None:
            temp = root.right
            root = None
            return temp
        elif root.right is None:
            temp = root.left
            root = None
            return temp
        # Node with two children: Get the inorder successor
        # (smallest in the right subtree)
        temp = minValueNode(root.right)
        # Copy the inorder successor's content to this node
        root.key = temp.key
        # Delete the inorder successor
        root.right = deleteNode(root.right, temp.key)
    return root
```



```python
""" Let us create following BST
              50
           /     \
          30      70
         /  \    /  \
       20   40  60   80 """

root = None
root = insert(root, 50)
root = insert(root, 30)
root = insert(root, 20)
root = insert(root, 40)
root = insert(root, 70)
root = insert(root, 60)
root = insert(root, 80)

print "Inorder traversal of the given tree"
inorder(root)

print "\nDelete 20"
root = deleteNode(root, 20)
print "Inorder traversal of the modified tree"
inorder(root)

print "\nDelete 30"
root = deleteNode(root, 30)
print "Inorder traversal of the modified tree"
inorder(root)

print "\nDelete 50"
root = deleteNode(root, 50)
print "Inorder traversal of the modified tree"
inorder(root)
```