#Aim: Write Python program for deleting an element (assuming data is given) from binary tree.

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

# Given a non-empty binary search tree, return the node

# with minum key value found in that tree. Note that the

# entire tree does not need to be searched

def minValueNode( node):

current = node

# loop down to find the leftmost leaf

while(current.left is not None):

current = current.left

return current

# 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

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)

ch=input("Do you want to delete anything(y/n) :")

while ch=="y":

num=int(input("Enter number to delete:"))

root = deleteNode(root, num)

print ("\nDeleted ",num)

if root is None:

ch='n'

print ("\nTree is empty now...\n")

else:

print("tree after deletion of element is")

inorder(root)

ch=input("Do you want to delete anything(y/n) :")