class Node:

def \_\_init\_\_(self, data):

self.data = data

self.left\_child = None

self.right\_child = None

class BST:

def \_\_init\_\_(self):

self.root = None

def create\_BST(value):

if self.root == None:

self.root = Node(value)

else:

current = self.root

while True:

if value < current.data:

if current.left\_child:

current = current.left\_child

else:

current.left\_child = Node(value)

break

elif value > current.data:

if current.right\_child:

current = current.right\_child

else:

# current.right\_chid = Node(value)

current.right\_child = Node(value)

break

else:

break

def pre\_order(current):

if not current:

return

else:

print(current.data)

pre\_order(current.left\_child)

pre\_order(current.right\_child)

# in\_order traverse and post\_order traverse can be implemented similarly

tree = BST()

arr = [8,3,1,6,4,7,10,14,13]

for i in arr:

tree.create\_BST(i)

print()

tree.pre\_order(tree.root)