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(self, 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\_child = Node(value)

break

else:

break

def pre\_order(self, current):

if current is not None:

print(current.data)

self.pre\_order(current.left\_child)

self.pre\_order(current.right\_child)

def in\_order(self, current):

if current is not None:

self.in\_order(current.left\_child)

print(current.data)

self.in\_order(current.right\_child)

tree = BST()

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

for i in arr:

tree.create\_BST(i)

print('previous order')

tree.pre\_order(tree.root)

print('in order')

tree.in\_order(tree.root)