二叉树的代码构建：

class TreeNode(object):

#The min struct of Tree

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

self.left=left

self.right=right

self.data = data

#the min cell

#define the root Node

#distinguish the node is empty

class BTree(TreeNode):

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

self.root = root

def is\_empty(self):

if self.root==0:

return True

else:

return False

def preorder(self,treenode):

#root left right

print treenode.data

if treenode.left !=None:

self.preorder(treenode.left)

if treenode.right !=None:

self.preorder(treenode.right)

def inorder(self,treenode):

if treenode.left !=None:

self.inorder(treenode.left)

print treenode.data

if treenode.right !=None:

self.inorder(treenode.right)

#left right root

def postorder(self,treenode):

if treenode.left !=None:

self.postorder(treenode.left)

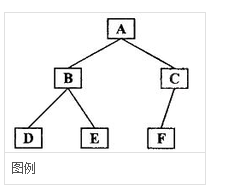
if treenode.right !=None:

self.postorder(treenode.right)

print treenode.data

例子：

要构建这样的一个二叉树



构建代码：

nod3=TreeNode(left=None,right=None,data="D")

nod4=TreeNode(left=None,right=None,data="E")

nod5=TreeNode(left=None,right=None,data="F")

nod1=TreeNode(left=nod3,right=nod4,data="B")

nod2=TreeNode(left=nod5,right=None,data="C")

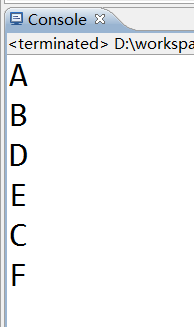
root = TreeNode(left=nod1,right=nod2,data="A")

rtt=BTree(root)

二叉树构建完成

使用preorder方法就能实现先序遍历

rtt.preorder(rtt.root)

运行的结果：

中序遍历、后序遍历不再测试，与读者自行测试。