# Evidence 1

class Node:

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

self.\_\_data = str(data)

xd

self.\_\_leftPtr = int(-1)

self.\_\_rightPtr = int(-1)

H2 computing 2019

https://emojipedia-us.s3.dualstack.us-west-1.amazonaws.com/thumbs/120/apple/198/ok-hand-sign_1f44c.pnghttps://emojipedia-us.s3.dualstack.us-west-1.amazonaws.com/thumbs/120/apple/198/ok-hand-sign_1f44c.pnghttps://emojipedia-us.s3.dualstack.us-west-1.amazonaws.com/thumbs/120/apple/198/ok-hand-sign_1f44c.png def setData(self, s):

self.\_\_data = str(s)

def setLeftPtr(self, x):

self.\_\_leftPtr = int(x)

def setRightPtr(self, y):

self.\_\_rightPtr = int(y)

def getData(self):

return self.\_\_data

def getLeftPtr(self):

return self.\_\_leftPtr

def getRightPtr(self):

return self.\_\_rightPtr

class Tree:

'''NOTE: This is a binary search tree that assumes that there is no limit to the number of nodes that can be added to the tree.'''

def \_\_init\_\_(self): # constructor() / CreateNewTree method

self.\_\_tree = []

self.\_\_root = -1

def add(self, newItem):

newNode = Node(newItem)

if self.\_\_root == -1:

self.\_\_root = 0

self.\_\_tree.append(newNode)

else:

previous = self.\_\_root

current = self.\_\_root

while current != -1:

if newItem < self.\_\_tree[current].getData():

previous = current

current = self.\_\_tree[current].getLeftPtr()

else:

previous = current

current = self.\_\_tree[current].getRightPtr()

self.\_\_tree.append(newNode) # new index of node is (length of new tree - 1)

if newItem < self.\_\_tree[previous].getData():

self.\_\_tree[previous].setLeftPtr(len(self.\_\_tree) - 1)

else:

self.\_\_tree[previous].setRightPtr(len(self.\_\_tree) - 1)

def \_\_str\_\_(self): # print() method

output = "Root of tree: {0}\n".format(self.\_\_root)

output += "-" \* 63

output += "\n"

output += "| {0:^10} | {1:^20} | {2:^10} | {3:^10} |\n".format("Index", "Data", "Left Ptr", "Right Ptr")

output += "-" \* 63

output += "\n"

for i in range(len(self.\_\_tree)):

current = self.\_\_tree[i] # current nodde

output += "| {0:^10} | {1:^20} | {2:^10} | {3:^10} |\n".format(i, current.getData(), current.getLeftPtr(), current.getRightPtr())

output += "-" \* 63

output += "\n\n"

return output

# Evidence 2

def main():

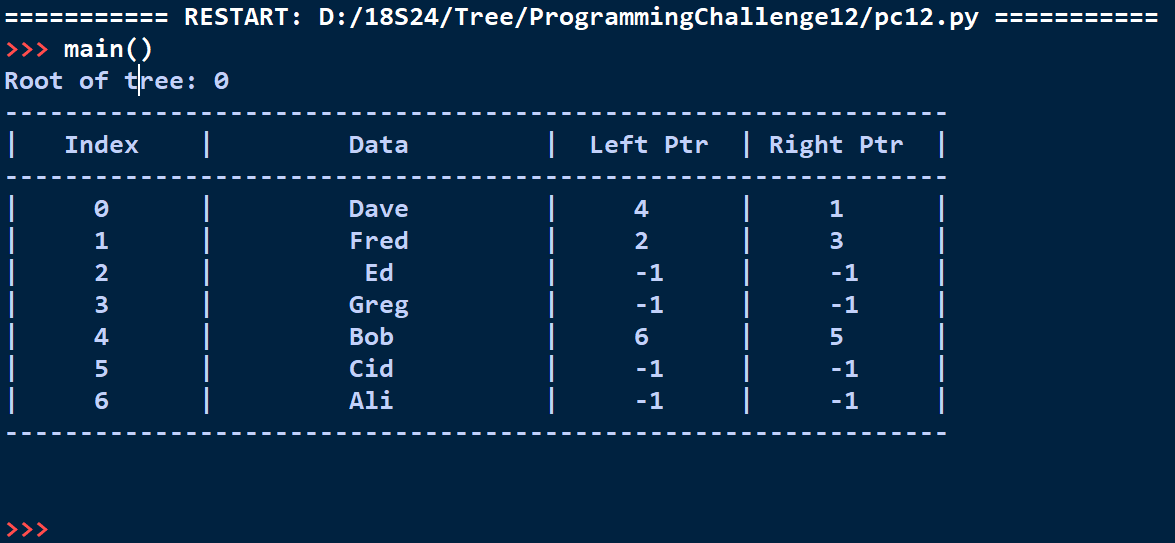
peopleTree = Tree()

peopleList = ["Dave", "Fred", "Ed", "Greg", "Bob", "Cid", "Ali"]

for person in peopleList:

peopleTree.add(person)

print(peopleTree)



# Evidence 3

# within class Tree()

def inOrderTraversal(self, index):

if index != -1:

self.inOrderTraversal(self.\_\_tree[index].getLeftPtr())

print(self.\_\_tree[index].getData())

self.inOrderTraversal(self.\_\_tree[index].getRightPtr())

def getRoot(self):

return self.\_\_root

# end of class Tree

def main():

peopleTree = Tree()

peopleList = ["Dave", "Fred", "Ed", "Greg", "Bob", "Cid", "Ali"]

for person in peopleList:

peopleTree.add(person)

print(peopleTree)

peopleTree.inOrderTraversal(peopleTree.getRoot())

