#Write a program to implement Huffman Encoding using a greedy strategy.

import heapq

class node:

def \_init\_(self, freq, symbol, left=None, right=None):

self.freq = freq

self.symbol = symbol

self.left = left

self.right = right

self.huff = ''

def \_lt\_(self, nxt):

return self.freq < nxt.freq

def printNodes(node, val=''):

newVal = val + str(node.huff)

if(node.left):

printNodes(node.left, newVal)

if(node.right):

printNodes(node.right, newVal)

if(not node.left and not node.right):

print(f"{node.symbol} -> {newVal}")

chars = ['a', 'b', 'c', 'd', 'e', 'f']

freq = [ 5, 9, 12, 13, 16, 45]

nodes = []

for x in range(len(chars)):

heapq.heappush(nodes, node(freq[x], chars[x]))

while len(nodes) > 1:

left = heapq.heappop(nodes)

right = heapq.heappop(nodes)

left.huff = 0

right.huff = 1

newNode = node(left.freq+right.freq, left.symbol+right.symbol, left, right)

heapq.heappush(nodes, newNode)

printNodes(nodes[0])