# code running instructinos

使用Transaction类来定义交易的信息

example:Transaction('A', 'B', 100)表示A向B发送了100个币。

将同一时间产生的相关交易放入列表中，形成交易链。例如A有100，A向B购买一个价值50的物品，则需要这么使用：

tg = [Transaction('A', 'B', 100), Transaction('A', 'A', 50)]

多个交易链需要经过默克尔树进行压缩，将交易列表放入MerkleTree中，将所有的默克尔根放入列表中模拟交易池，如下：

tgs = [MerkleTree(tg1), MerkleTree(tg2), MerkleTree(tg3)]

使用Chain类来模拟产生区块和上链的过程，Chain中输入区块的工作量，如下：

chain = Chain(20) # 代表这个区块链是20个比特的工作量  
  
for i, tg in enumerate(tgs):  
 data = tg  
 chain.add\_to\_pool(data)  
 chain.mine()  
 print\_blocks(chain.blocks[i + 1], is\_first=False)

print\_blocks函数是输出区块的信息，定义如下：

def print\_blocks(block: Block, is\_first=False):  
 print("Nonce:", block.nonce)  
 print("Prev:", block.previous\_hash.hexdigest())  
 print("Hash:", block.hash.hexdigest())  
 print("Data:", block.data)  
 if not is\_first:  
 success("-----------------------------------")  
 block.data.printTree()  
 success("------------------------------------")  
   
 print("")

