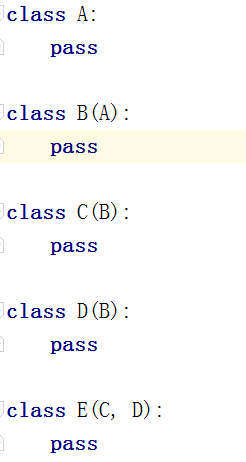
**作业1:**

**写出以下类继承的执行顺序(MRO顺序)及C3推导步骤**



**分析：**B继承于A；C和D继承于B；E继承于C和D。MRO算法采用的是有向无环图的拓扑排序，采用剪枝的方法来分析：

1. 再找到入度为0的点为E，因为没有类继承E，剪去E左右两边，得到 E ->
2. 接下来再找下一个入度为0的点(剪枝后没有类继承C和D了)，C和D的入度为0，根据先取最左原则，剪去C的枝，得到

E ->C ->

1. 再找到入度为0的点为D，则剪去D的枝得到 E ->C ->D ->
2. 继续找到入度为0的点B，则剪去B的枝得到E ->C ->D ->B->
3. 最后得到： E ->C ->D ->B->A的顺序

多继承的mro其原理是c3算法：

MRO是一个有序列表L，在类被创建时计算:

L(E(C,D)) = E+ merge(L(C) ,L(D) ,(CD))

= E+ merge(C(L(B)) ,D(L(B)) ,CD)

= E+C+ merge(L(B) ,D(L(B)) ,D)

= E+C+D+merge(BA,BA)

= E+C+D+B+merge(A,A)

= E+ C+D+B+A

**作业2:**

**实现一个单例模式的类**

**分析：**

当我们实例化一个对象时，是先执行了类的\_\_new\_\_方法（我们没写时，默认调用object.\_\_new\_\_），实例化对象；然后再执行类的\_\_init\_\_方法，对这个对象进行初始化，所有我们可以基于这个，实现单例模式：

**源代码如下：**

# -\*- coding: utf-8 -\*-

import time

import threading

class Singleton(object):

\_instance\_lock = threading.Lock()

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

pass

def \_\_new\_\_(cls, \*args, \*\*kwargs):

if not hasattr(Singleton, "\_instance"):

with Singleton.\_instance\_lock:

if not hasattr(Singleton, "\_instance"):

Singleton.\_instance = object.\_\_new\_\_(cls)

return Singleton.\_instance

obj1 = Singleton()

obj2 = Singleton()

print(obj1,obj2)