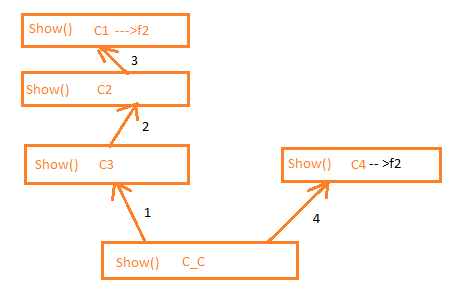


***用例1***: 子类C\_C继承2个父类，以Class C\_C(C3,C4) 申明，而C3是C2子类，C2是C1的子类。现在假设从C\_C实例对象中调用方法f2. f2 方法在C1以及C4中有所定义。C1的f2的方法体为输出C1: Hello World. C4的f2方法体为输出C4: Hello World.

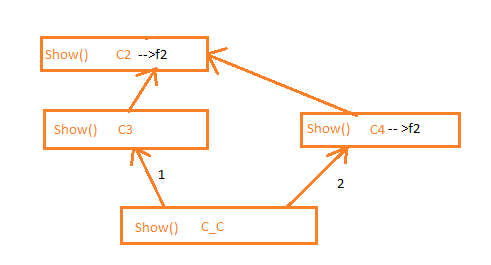
分析：首先C\_C的继承顺序是C3然后C4, 从而C3的方法具有比C4更优先的调用顺序。而Python首先没有在C\_C中找到f2方法时，会在具有优先级比较高的C3中查找f2, C3也没有则Python会**深度遍历**，再去C3的父类中找，一直找到没有更高层的父类。所以程序输出C1:Hello World.



***用例2：***基于用例1，现在在每个Class当中都定义了一个Show()，这个Show()方法打印出属于Message from Class名。即是C1输出Message from C1, C2输出Message from C2, C3输出Message from C3…..

在C1的f2函数中现在除了输出C1:Hello World以后， 还会调用self.Show()函数。如果我们创建一个C\_C的实例，并且调用f2函数以后，会输出什么？

分析：首先原理和用例1一样，f2的函数将会以深度优先的顺序去调用C1中的f2，而不是C4中的f2函数。 然后f2函数中将调用self.Show()函数， 重点在于当Python在类定义中遇到self关键字的时候，都会从调用的实例最底层中开始查找，所以首先查找C\_C中式否有Show()函数，结果找到，所以输出C1:Hello World, Message from C\_C. 基于这个原理，如果C\_C中没有Show()函数，那么这一次查找Show()函数将会到C3当中。所以如果C\_C中没有Show()函数，则会输出C1:Hello World, Message from C3.



***用例3：***现在C\_C继承了C3和C4类，C3继承C2, C4继承C2. f2函数只存在于类C2和类C4中。如果我们建立一个C\_C的实例对象，调用f2函数，那么将输出什么？

分析：首先Python会在C\_C中去寻找f2函数，然后找不到会沿着具有优先级的父类C3中寻找。C3 还没有时，依照用例1，他将继续深度优先遍历，但是由于C4和C3具有相同的父类，Python将会找寻C4，然后再找寻 C2. **拥有共同的父类将改变寻找的顺序**。 所以输出C4:Hello World.