**什么是回调函数？**

当程序运行是，一般情况下，应用程序会时常通过API调用库里所预先备好的函数。但是有些库函数却要求应用先传给它一个函数，好在合适的时候调用，以完成目标任务。这个被传入的、后又被调用的函数就称为回调函数（callback function）。

例如：

有一家旅馆提供叫醒服务，但是要求旅客自己决定叫醒的方法。可以是打客房电话，也可以是派服务员去敲门，睡得死怕耽误事的，还可以要求往自己头上浇盆水。这里，“叫醒”这个行为是旅馆提供的，相当于库函数，但是叫醒的方式是由旅客决定并告诉旅馆的，也就是回调函数。而旅客告诉旅馆怎么叫醒自己的动作，也就是把回调函数传入库函数的动作，称为登记回调函数（to register a callback function）

# 举例  
def jiaoXing(fangShi,name): # fangShi,name 都是形参，大家可以理解成空变量  
 *"""  
 酒店提供叫醒服务，但是叫醒的方式由客户自己去决定  
 :参数1 fangShi: 这里接受一个函数作为实参，默认这个函数有返回值  
 :参数2 name: 这里接受一个名字字符串作为实参* ***:return****:  
 """* f = fangShi(name) # 调用回调函数，得到函数返回值  
 print(f) # 打印函数返回值  
  
# 调用叫醒服务函数，传了一个叫醒方式的回调函数，并且传了被叫醒对象变量值  
# 普通写法：  
def fs1(name):  
 return f"打电话叫醒{name}"  
  
def fs2(name):  
 return f"敲门叫醒{name}"  
  
def fs3(name):  
 return f"翻窗叫醒{name}"  
  
# jiaoXing(fs1,"张三")  
# jiaoXing(fs2,"李四")  
# jiaoXing(fs3,"王五")  
  
# 匿名函数写法：21：05 - 21：15 大家可以选择思考一下，如果将上面的代码用lamdba简写  
  
jiaoXing(lambda name:f"打电话叫醒{name}","赵7")  
jiaoXing(lambda name:f"敲门叫醒{name}","老八")  
jiaoXing(lambda name:f"翻窗叫醒{name}","周9")

**回调机制的优势**

回调机制提供了非常大的灵活性。有的同学可定想回调似乎只是函数间的调用，可以发现两者之间的一个关键的不同：

在回调中，我们利用某种方式，把回调函数像参数一样传入中间函数。可以这么理解，在传入一个回调函数之前，中间函数是不完整的。换句话说，程序可以在运行时，通过登记不同的回调函数，来决定、改变中间函数的行为。这就比简单的函数调用要灵活太多了。

# 回调函数1：生成一个2\*形式的偶数  
def double(x):  
 return x \* 2  
  
# 回调函数2：生成一个4\*形式的偶数  
def quadruple(x):  
 return x \* 4  
  
# 中间函数： 接受一个生成偶数的函数作为参数，返回一个奇数  
def getOddNumber(k, getEvenNumber):  
 return 1 + getEvenNumber(k)  
  
  
# 起始函数，这里是程序的主函数  
def main():  
 k = 1  
 # 当需要生成一个2k+1形式的奇数时  
 i = getOddNumber(k, double)  
 print(i)  
 # 当需要一个4k+1形式的奇数时  
 i = getOddNumber(k, quadruple)  
 print(i)  
 # 当需要一个8k+1形式的奇数时  
 i = getOddNumber(k, lambda x: x \* 8)  
 print(i)  
  
  
if \_\_name\_\_ == "\_\_main\_\_":  
 main()