### 自定义一个Person类，其中有字段age，并提供相应的set get 方法，age业务逻辑上不可以使用<= 0 的数字

需求：如果使用setAge 为age赋值时，如果设置的是非法的数据，那么抛出一个自定义的编译期异常（AgeException），然后测试，调用setAge方法，并进行try except 处理。

class AgeException(Exception):  
 def \_\_init\_\_(self,msg):  
 self.msg=msg

from day13.task.demo5 import AgeException  
  
class Person:  
 \_\_age=None  
  
 def setAge(self,age):  
 self.\_\_age=age  
 def getAge(self):  
 return self.\_\_age  
  
 def compare(self,n):  
 if self.\_\_age>n.getAge():  
 print(**"设置合理！"**)  
 else:  
 raise AgeException(**"设置非法！"**)

from day13.task.Person import Person  
from day13.task.demo5 import AgeException  
  
p=Person()  
p.setAge(20)  
  
p1=Person()  
p1.setAge(0)  
  
  
try:  
 p.compare(p1)  
except AgeException as e:  
 print(e)

## 使用键盘录入模拟从银行取钱，假设余额是3000元，如果录入的数据大于余额，通过给出提示：金额不足异常。

class bank(Exception):  
 def \_\_init\_\_(self,msg):  
 self.\_\_msg=msg  
  
def Qvqian():  
 p=int(input(**"请输入取款金额："**))  
 i=3000  
 if p<=i:  
 print(**"取款成功！您的余额为："**,i-p)  
 else:  
 raise bank(**"余额不足异常！"**)  
Qvqian()