### 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。

from personexception.PException import AgeException  
class Person:  
 \_\_name = None  
 \_\_age = None  
  
 def setName(self,name):  
 self.\_\_name = name  
 def getName(self):  
 return self.\_\_name  
  
 def setAge(self,age):  
 if age<=0:  
 raise AgeException(**"数据非法！！！"**)  
 else:  
 self.\_\_age = age  
 def getAge(self):  
 return self.\_\_age

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

from personexception.person import Person  
from personexception.PException import AgeException  
  
p=Person()  
p.setName(**"曹东雪"**)  
  
try:  
 p.setAge(0)  
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。

from bankexception.BException import BalanceException  
class Bank:  
 \_\_balance = None  
  
 def setBalance(self,balance):  
 self.\_\_balance = balance  
 def getBalance(self):  
 return self.\_\_balance  
  
 def takeOut(self):  
 b = 3000  
 self.\_\_balance=int(input(**"请输入取款金额："**))  
 if self.\_\_balance>b:  
 raise BalanceException(**"余额不足！！！"**)  
 else:  
 print(**"取款成功！！！"**,**"现在余额为"**,b-self.getBalance(),**"元！"**)

class BalanceException(Exception):  
 def \_\_init\_\_(self,smg):  
 self.smg = smg

from bankexception.BException import BalanceException  
from bankexception.bank import Bank  
n=Bank()  
try:  
 n.takeOut()  
except BalanceException as e:  
 print(e)