■私有变量

# 如果要让内部属性不被外部访问，可以把属性的名称前加上两个下划线\_\_，  
# 在Python中，实例的变量名如果以\_\_开头，就变成了一个私有变量（private），  
# 只有内部可以访问，外部不能访问  
  
  
class Test(object):  
  
 def \_\_init\_\_(self):  
 self.\_\_name = 'Tom'  
 self.age = 25  
  
 # 可以用这个函数将私有变量暴露出去  
 def get\_name(self):  
 return self.\_\_name  
  
 # 也可以用这个函数来让外部可以修改私有变量  
 def set\_name(self, name):  
 self.\_\_name = name  
  
 def pt(self):  
 print(self.\_\_name)  
  
  
a = Test()  
print(a.age)  
# 输出：25  
  
# 在外部无法访问私有变量\_\_name  
print(a.\_\_name)  
# 输出: AttributeError: 'Test' object has no attribute '\_\_name'  
  
# 而内部的pt函数就能访问内部的私有变量\_\_name  
a.pt()  
# 输出： Tom  
  
# 通过访问get\_name获取到私有变量  
print(a.get\_name())  
# 输出： Tom

■继承和多态

class Fat(object):  
  
 def \_\_init\_\_(self):  
 self.name = 'fuqin'  
 self.age = 35  
  
  
class Son(Fat):  
 pass  
  
  
a = Son()  
print(a.name)  
# 输出：fuqin  
  
# 可以在子类里用新的值来覆盖父类里的age属性  
a.age = 38  
  
# 也可以在子类里创建新属性  
a.jineng = 'codding'  
  
print(a.age)  
# 输出：38  
print(a.jineng)  
# 输出：codding  
  
# a的类型是Son同时也是其父类Fat  
print(isinstance(a, Son))  
# 输出： True  
  
print(isinstance(a, Fat))  
# 输出： True

■获取对象信息

三种方法

type(), isinstance(), dir()

class Shi(object):  
  
 def \_\_init\_\_(self):  
 self.name = 'Joe'  
 self.age = 26  
  
 def say\_hello(self):  
 print('Hello', self.name)  
  
  
a = Shi()  
  
# type方法查看一个对象的类型  
print(type(a))  
# 输出： <class '\_\_main\_\_.Shi'>  
  
# isinstance方法判断一个对象是否从属于一个类  
print(isinstance(a, Shi))  
# 输出： True  
  
# dir方法列出对象的所有方法和属性  
print(dir(a))  
# 输出：  
# ['\_\_class\_\_', '\_\_delattr\_\_', '\_\_dict\_\_', '\_\_dir\_\_', '\_\_doc\_\_', '\_\_eq\_\_',  
# '\_\_format\_\_', '\_\_ge\_\_', '\_\_getattribute\_\_', '\_\_gt\_\_', '\_\_hash\_\_', '\_\_init\_\_',  
# '\_\_init\_subclass\_\_', '\_\_le\_\_', '\_\_lt\_\_', '\_\_module\_\_', '\_\_ne\_\_', '\_\_new\_\_',  
# '\_\_reduce\_\_', '\_\_reduce\_ex\_\_', '\_\_repr\_\_', '\_\_setattr\_\_', '\_\_sizeof\_\_',  
# '\_\_str\_\_', '\_\_subclasshook\_\_', '\_\_weakref\_\_', 'age', 'name', 'say\_hello']