'''

1.了解什么是对象和类

2.面向对象的三个主要特征：继承、封装和多态

3.创建自己的类

4.如何为类添加私有方法

5.掌握如何继承一个或多个类（多继承）

6.如何检测类之间的继承关系

类：拥有共同特征的同一类事物的总称或抽象

Bird

对象：类的具体化，从类创建对象也称为类的实例化

继承：

Sparrow ->Bird

Pegeon ->Bird

多态：

Bird -> Sparrow

Bird -> Pigeon

1.类是代码块，用冒号(:)结尾

2.如果类是空的，必须加pass语句

3.类方法和函数的定义方式类似，如果是实例方法，第一个参数必须是self；

self表示当前类的实例，self可以是任何变量名，这里用self是一种习惯；

'''

class MyClass: #空类必须加pass

pass

class Person:

def setName(self,name): #self:类的实例

self.name = name #定义name变量，并初始化

self.name2= name+'hello'

def getName(self):

return self.name

def getName2(self):

return self.name2

def greet(self):

print("Hello,I'm {}".format(self.name))

person1 = Person()

person1.setName("Leon")

print(person1.getName())

print(person1.getName2())

person1.greet()

8.2类方式的私有化

#为类的方法名加上两个下划线(\_\_)，表示私有方法

class Person:

def method1(self):

print("method1")

#\_ClassName\_\_methodName

def \_\_method2(self):

print("method2")

p = Person()

p.method1()

p.\_Person\_\_method2()

print("=="\*12)

import inspect

methods = inspect.getmembers(p,predicate=inspect.ismethod)

for item in methods:

print(item)

8.3类代码块

class MyClass:

print('MyClass') #不创建实例，也会执行

count=0

def counter(self):

self.count +=1

my=MyClass()

my.counter()

my.counter()

print(my.count)

8.4类的继承

class ParentClass:

def method1(self):

print("method1")

class ChildClass(ParentClass):

def method2(self):

print("method2")

child = ChildClass()

child.method1()

child.method2()

8.5检测类的继承关系，issubclass()

'''

A B

issubclass(childclass,parentclass)

isinstance(obj,class)

classname.\_\_bases\_\_

'''

class MyParentClass:

def method(self):

return 50

class ParentClass(MyParentClass):

def method1(self):

print("method1")

class ChildClass(ParentClass):

def method2(self):

print("method2")

class MyClass:

def method(self):

return 10

print(issubclass(ChildClass,ParentClass))

print(issubclass(ChildClass,MyParentClass))

print(issubclass(MyClass,MyParentClass))

#输出类的基类

print(ChildClass.\_\_bases\_\_) #直接父类

print(ParentClass.\_\_bases\_\_)

print(MyClass.\_\_bases\_\_) #从根类object继承

child = ChildClass()

print(isinstance(child,ChildClass)) #检测实例的类类型

print(isinstance(child,ParentClass)) #True,超类

parent=ParentClass()

print(isinstance(parent,MyParentClass))

print(isinstance(parent,ChildClass))

8.6类的多继承

class Calculator:

def calculate(self,expression):

self.value = eval(expression)

def printResult(self):

print('result:{}'.format(self.value))

class MyPrint:

def printResult(self):

print("计算结果：{}".format(self.value))

class NewCalculator(Calculator,MyPrint): #按顺序有限调用前面类的同名方法

pass

calc=NewCalculator()

calc.calculate('1+2\*3')

calc.printResult()

class NewCalculator1(MyPrint,Calculator):

pass

calc1=NewCalculator1()

calc1.calculate('1+2\*3+5')

calc1.printResult()

print(NewCalculator.\_\_bases\_\_)

8.7接口

#接口, hasattr,getattr, setattr

class MyClass:

def method1(self):

print("method1")

def method2(self):

print("---method2")

def default(self):

print('default')

my=MyClass()

if hasattr(my,'method1'):

my.method1()

else:

print("method1方法不存在")

print("="\*10)

method=getattr(my,'method2',my.default())

method()

def fun():

print("fun")

method2=getattr(my,'method23',fun)

method2()

#setter，动态往对象添加方法

def process():

print('动态添加的process方法')

setattr(my,'hello',process)

my.hello()