1、下列程序结果是\_\_\_\_\_\_.

class A:  
  
 def \_\_init\_\_(self):  
 print("初始化")  
  
 def \_\_del\_\_(self):  
 print("销毁")  
  
 def \_\_str\_\_(self):  
 return "类A"  
a = A()  
print(a)

**结果：**

初始化

类A

销毁

2. 下列程序执行结果\_\_\_\_\_\_.

class Animal(object):  
  
 def shut(self):  
 print("动物在叫")  
class Dog(Animal):  
  
 def shut(self):  
 super().shut()  
 print("汪汪汪")  
dog = Dog()  
dog.shut()

**结果：**

**动物在叫**

**汪汪汪**

3. 创建猫类：

类名：Cat

属性：无

描述：创建一个Cat类，通过Cat类创建一个对象cat，执行print(cat)输出“喵？喵？喵？”.

**class Cat(object):  
 def \_\_str\_\_(self):  
 return "喵？喵？喵？"  
cat = Cat()  
print(cat)**

4.创建计算器类：

类名：Calculator

属性：number\_1（数字一）、number\_2（数字二）

方法：

def \_\_init\_\_(self,number\_1,number\_2):

# 类的初始化方法

def add(self) # 返回数字一加数字二的值

def sub(self) # 返回数字一减去数字二的值

def div(self) # 返回数字一除以数字二的值

def mul(self) # 返回数字一乘以数字二的值

描述：创建计算器类，通过计算器类创建一个计算器对象，在创建对象时需要传入数字一和数字二，分别调用计算器的四种方法.

**结果：**

**class** Calculator(object):  
 **def** \_\_init\_\_(self, number\_1, number\_2):  
 self.num\_1 = number\_1  
 self.num\_2 = number\_2  
 **def** add(self):  
 **return** self.num\_1 + self.num\_2  
 **def** sub(self):  
 **return** self.num\_1 - self.num\_2  
 **def** div(self):  
 **return** self.num\_1 / self.num\_2  
 **def** mul(self):  
 **return** self.num\_1 \* self.num\_2  
  
cal = Calculator(6, 2)  
print(cal.add())  
print(cal.sub())  
print(cal.div())  
print(cal.mul())

5.创建Cat和Dog类分别继承Animal类，分别重写shut和eat方法，创建Cat类对象cat和Dog类对象dog，分别调用cat和dog的shut和eat方法

class Animal:  
  
 def shut(self):

# 打印叫声  
 pass  
  
 def eat(self):

# 打印爱吃的食物  
 pass

**结果：**

**class** Animal(object):  
 **def** shut(self):  
 print(**"动物的叫声！"**)  
 **def** eat(self):  
 print(**"爱吃的食物！"**)  
  
  
**class** Cat(Animal):  
 **def** shut(self):  
 print(**"喵！喵！喵！"**)  
 **def** eat(self):  
 print(**"喵喵爱吃鱼！"**)  
  
  
**class** Dog(Animal):  
 **def** shut(self):  
 print(**"汪汪汪！"**)  
 **def** eat(self):  
 print(**"汪汪爱吃骨头！"**)  
  
cat = Cat()  
cat.shut()  
cat.eat()  
  
dog = Dog()  
dog.shut()  
dog.eat()

1. 面向对象三大特性是\_\_封装\_\_\_\_、\_\_\_继承\_\_\_、\_\_\_多态\_\_\_.