Python程序设计

第七讲 面向对象程序设计
继承与多态

张 华
WHU

继承

- 封装、继承、多态是面向对象编程的三大要素。
- 继承
  - 继承是用来实现代码复用和设计复用的机制，是面向对象程序设计的重要特性之一。
    - 设计一个新类时，如果可以继承一个已有的设计良好的类然后进行二次开发，无疑会大幅度减少开发工作量。
  - 在继承关系中，已有的、设计好的类称为父类或基类，新设计的类称为子类或派生类。
  - 子类和父类之间是一种“is-a”关系，即子类（对象）也是一种父类（对象）。
    - 例如，苹果是一种水果，卡车是一种车，狗是一种动物

继承

继承

- 派生类可以继承父类的公有成员，但是不能继承其私有成员。
- 如果需要在派生类中调用基类的方法，可以使用内置函数super()或者通过“基类名.方法名()”的方式来实现这一目的。
- Python支持多继承，如果父类中有相同的方法名，而在子类中使用时没有指定父类名，则Python解释器将从左向右按顺序进行搜索。
- 通过类的方法mro()或类的属性__mro__可以输出其继承的层次关系。
- object是根类（祖先类）。

继承

继承示例

```
class Animal(object):       #定义基类
    def show(self):
        print('I am an animal.')

class Cat(Animal):  # 派生类，继承了基类的show()方法
    def show(self):
        super(Cat, self).show()    # 或者 Animal.show()
        print('In fact, I am a cat.')

Cat().show()
```

```
I am an animal.
In fact, I am a cat.
```

继承

继承示例

```
class Pet():
    def __init__(self, name):
        self.nickname = name

class Cat(Animal, Pet):  # 多重继承
    def __init__(self, name):
        # 调用基类的构造函数对基类数据成员进行初始化
        super(Cat, self).__init__(name)
        # 或者 Pet.__init__(self, name)

    def show(self):
        super(Cat, self).show()    # 或者 Animal.show()
        print('In fact, I am a cat.')
        print('My nickname is '+self.nickname+'.')

Cat('Tom').show()
```

```
I am an animal.
In fact, I am a cat.
My nickname is Tom.
```

多态

多态

- 所谓多态，是指基类的同一个方法在不同派生类对象中具有不同的表现和行为。
- 派生类继承了基类行为和属性之后，还会增加某些特定的行为和属性，同时还可能会对继承来的某些行为进行一定的改变，这都是多态的表现形式。
- Python大多数运算符可以作用于多种不同类型的操作数，并且对于不同类型的操作数往往有不同的表现，这本身就是多态，是通过特殊方法与运算符重载实现的。

多态

多态示例

```
class Animal(object):          #定义基类
    def show(self):
        print('I am an animal.')

class Cat(Animal):             #派生类，覆盖了基类的show()方法
    def show(self):
        print('I am a cat.')

class Dog(Animal):             #派生类
    def show(self):
        print('I am a dog.')

class Tiger(Animal):           #派生类
    def show(self):
        print('I am a tiger.')

class Test(Animal):            #派生类，没有覆盖基类的show()方法
    pass
```

多态

多态示例

```
x = [item() for item in (Animal, Cat, Dog, Tiger, Test)]
for item in x:          #遍历基类和派生类对象并调用show()方法
    item.show()
```

```
I am an animal.
I am a cat.
I am a dog.
I am a tiger.
I am an animal.
```

抽象方法与抽象类

定义抽象类和抽象方法

- 包含抽象方法的类是抽象类，不能创建抽象类的对象。
- 抽象方法一般在抽象类中定义，并且要求在派生类中必须重新实现，否则不允许派生类创建实例。
- 示例

```
import abc

class Shape(metaclass=abc.ABCMeta):
    @abc.abstractmethod
    def area(self):
        pass

s = Shape()
```

```
s = Shape()
TypeError: Can't instantiate abstract class Shape with
abstract methods area
```

抽象方法与抽象类

练习3

- 创建一个图形类族，包括下面的类
  - Shape抽象类，表示封闭的平面图形
    - 抽象方法area，计算图形的面积
  - Circle类，Shape的子类，表示圆
    - 实现继承的抽象方法
  - Rect类，Shape的子类，表示矩形
    - 实现继承的抽象方法
- 定义计算形状面积的函数calcArea
  - 计算任意类型Shape对象的面积
- 定义测试函数
  - 创建若干圆和矩形，调用calcArea计算它们的面积

抽象方法与抽象类

练习3

- 增加图形库的功能
  - 增加Point类表示点
  - 在Shape抽象类中增加一个Point类型的position实例变量，表示位置
  - 在Shape抽象类中增加修改位置、绘制图形、移动图形、改变大小等方法
  - 修改Circle类和Rect类
  - 定义更多的Shape子类
- 基于图形库创建一个脚本动画程序
  - 创建若干图形，构成一张卡通画
  - 移动图中图形的位置，改变它们的大小
  - 实现设计的动画效果

特殊方法与运算符重载

- Python类有特殊方法，其中比较常见的是构造函数和析构函数，除此之外，Python还支持大量的特殊方法。
- 运算符重载就是通过重写特殊方法实现的。

特殊方法与运算符重载

方法	功能说明
__new__()	类的静态方法，用于确定是否要创建对象
__init__()	构造方法，创建对象时自动调用
__del__()	析构方法，释放对象时自动调用
__add__()	+
__sub__()	-
__mul__()	*
__truediv__()	/
__floordiv__()	//
__mod__()	%
__pow__()	**
__eq__()、__ne__()、 __lt__()、__le__()、 __gt__()、__ge__()	==、!=、 <、<=、 >、>=
__lshift__()、__rshift__()	<<、>>
__and__()、__or__()、 __invert__()、__xor__()	&、|、 ~、^

特殊方法与运算符重载

方法	功能说明
__iadd__()、__isub__()	+=、-=，很多其他运算符也有与之对应的复合赋值运算符
__pos__()	一元运算符+，正号
__neg__()	一元运算符-，负号
__contains__ ()	与成员测试运算符in对应
__radd__()、__rsub__	反射加法、反射减法，一般与普通加法和减法具有相同的功能但操作数的位置或顺序相反，很多其他运算符也有与之对应的反射运算符
__abs__()	与内置函数abs()对应
__bool__()	与内置函数bool()对应，要求该方法必须返回True或False
__bytes__()	与内置函数bytes()对应
__complex__()	与内置函数complex()对应，要求该方法必须返回复数
__dir__()	与内置函数dir()对应
__divmod__()	与内置函数divmod()对应
__float__()	与内置函数float()对应，要求该该方法必须返回实数
__hash__()	与内置函数hash()对应
__int__()	与内置函数int()对应，要求该方法必须返回整数

特殊方法与运算符重载

方法	功能说明
__len__()	与内置函数len()对应
__next__()	与内置函数next()对应
__reduce__()	提供对reduce()函数的支持
__reversed__()	与内置函数reversed()对应
__round__()	对内置函数round()对应
__str__()	与内置函数str()对应，要求该方法必须返回str类型的数据
__repr__()	打印、转换，要求该方法必须返回str类型的数据
__getitem__()	按照索引获取值
__setitem__()	按照索引赋值
__delattr__()	删除对象的指定属性
__getattr__()	获取对象指定属性的值，对应成员访问运算符“.”

特殊方法与运算符重载

方法	功能说明
__getattribute__()	获取对象指定属性的值，如果同时定义了该方法与__getattr__()，那么__getattr__()将不会被调用，除非在__getattribute__()中显式调用__getattr__()或者抛出AttributeError异常
__setattr__()	设置对象指定属性的值
__base__	该类的基类
__class__	返回对象所属的类
__dict__	对象所包含的属性与值的字典
__subclasses__()	返回该类的所有子类
__call__()	包含该特殊方法的类的实例可以像函数一样调用
__get__()	定义了这三个特殊方法中任何一个的类称作描述符（descriptor），描述符对象一般作为其他类的属性来使用，这三个方法分别在获取属性、修改属性值或删除属性时被调用
__set__()	
__delete__()	