8.2 类和对象

目录

CONTENTS

8.2.1 类的定义和使用

8.2.2 构造函数

8.2.3 析构函数

8.2.4 实例属性和类属性

8.2.5 私有成员和公有成员

8.2.6 方法

面向对象程序设计的一个关键性观念是将数据以及对数据的操作封装在一起，组成一个相互依存、不可分割的整体，即对象。

对于相同类型的对象进行分类、抽象后，得出共同的特征而形成了类，面向对象程序设计的关键就是如何合理地定义和组织这些类以及类之间的关系。

8.2.1 类的定义和使用

1. 类定义

Python使用class关键字来定义类，创建类时，用变量形式表示的对象属性称为数据成员或属性（成员变量），用函数形式表示的对象行为称为成员函数（成员方法）。成员属性和成员方法统称为类的成员。语法形式如下：

```
class 类名：
    属性（成员变量）
    属性
    …
    成员函数（成员方法）
```

例：定义一个Person人员类。

```
class Person:
        num=1                          #成员变量
        def SayHello(self):    #成员函数
                print(" Hello!") ;
```

2. 对象定义

对象是类的实例。如果人类是一个类的话，那么某个具体的人就是一个对象。只有定义了具体的对象，并通过“对象名.成员”的方式来访问其中的数据成员或成员方法。

Python创建对象的语法如下：

```
对象名=类名（ ）
```

例：定义Person类的对象p.

```
p=Person()
p.SayHello()      #访问成员函数SayHello()
```

运行结果如下：

```
Hello！
```

8.2.2 构造函数

类可以定义一个特殊的叫做__init__( )的方法。一个类定义了__init__（)方法以后，类实例化时就会自动为新生成的类实例调用__init__（)方法。构造函数一般用于完成对象数据成员设置初值或进行其他必要的初始化工作。如果用户未涉及构造函数，Python将提供一个默认的构造函数。

例：定义一个复数类Complex，构造函数完成对象变量初始化工作。

```
class Complex:
    def __init__(self, realpart, imagpart):
        self.r = realpart
        self.i = imagpart
x = Complex(3.0,-4.5)
print(x.r, x.i)
```

运行结果如下：

```
3.0 -4.5
```

8.2.3 析构函数

Python中类的析构函数是__del__ ，用来释放对象占用的资源，在Python收回对象空间之前自动执行。如果用户未涉及析构函数，Python将提供一个默认的析构函数进行必要的清理工作。

例：

```
class Complex:
    def __init__(self, realpart, imagpart):
        self.r = realpart
        self.i = imagpart
    def __del__(self):
        print("Complex不存在了")
x = Complex(3.0,-4.5)
print(x.r, x.i)
print(x)
del x                #删除x对象变量
```

运行结果如下：

```
3.0 -4.5
<__main__.Complex object at 0x000001A4147A39B0>
Complex不存在了
```

8.2.4 实例属性和类属性

实例属性属于实例(对象)，只能通过对象名访问；类属性属于类可通过类名访问，也可以通过对象名访问，为类的所有实例共享。

例：定义含有实例属性（姓名name， 年龄age）和类属性（人数num）的Person人员类。

```
1  class Person:
2      num=1                        #类属性
3      def __init__(self, str,n):       #构造函数
4          self.name = str          #实例属性
5          self.age=n
6      def SayHello(self):          #成员函数
7          print("Hello!")
8      def PrintName(self):             #成员函数
9          print("姓名：", self.name, "年龄：", self.age)
10     def PrintNum(self):          #成员函数
11         print(Person.num)            #由于是类属性，所以不写self .num
12 #主程序
13 P1= Person("夏敏捷",42)
14 P2= Person("王琳",36)
15 P1.PrintName()
16 P2.PrintName()
17 Person.num=2                     #修改类属性
18 P1.PrintNum()
19 P2.PrintNum()
```

```
姓名： 夏敏捷 年龄： 42
姓名： 王琳 年龄： 36
2
2
```

例：为Car类动态增加属性name和成员方法setSpeed().

```
import types                    #导入types模块
class Car:
    price = 100000              #定义类属性price
    def __init__(self, c):
        self.color = c          #定义实例属性color
#主程序
car1 = Car("Red")
car2 = Car("Blue")
print(car1.color, Car.price)
Car.price = 110000              #修改类属性
Car.name = 'QQ'                 #增加类属性
car1.color = "Yellow"           #修改实例属性
print(car2.color, Car.price, Car.name)
print(car1.color, Car.price, Car.name)
def setSpeed(self, s):
    self.speed = s
car1.setSpeed = types.MethodType(setSpeed, Car)    #动态为对象增加成员方法
car1.setSpeed(50)                                  #调用对象的成员方法
print(car1.speed)
```

```
Red 100000
Blue 110000 QQ
Yellow 110000 QQ
50
```

8.2.5 私有成员与公有成员

在定义类的属性时，如果属性名以两个下划线“__”开头则表示是私有属性，否则是公有属性。私有属性在类的外部不能直接访问，需通过调用对象的公有成员方法来访问，或者通过Python支持的特殊方式来访问。Python提供了访问私有属性的特殊方式，可用于程序的测试和调试，对于成员方法也有同样性质。方法如下：　对象名._类名+私有成员

例如：访问Car类私有成员__weight

car1._Car__weight

注意：私有属性是为了数据封装和保密而设的属性，一般只能在类的成员方法（类的内部）中使用访问，虽然Python支持一种特殊的方式来从外部直接访问类的私有成员，但是并不推荐这种做法。公有属性是可以公开使用的，既可以在类的内部进行访问，也可以在外部程序中使用。

例：为Car类定义私有成员。

```
class Car:
    price = 100000                          #定义类属性
    def __init__(self,c,w):
        self.color = c                      #定义公有属性color
        self.__weight= w                    #定义私有属性__weight
#主程序
car1 = Car("Red",10.5)
car2 = Car("Blue",11.8)
print(car1.color)
print(car1. _Car__weight)
print(car1. __weight)                       # AttributeErrorr
```

```
Red
10.5
Traceback (most recent call last):
  File "C:/Users/123/Desktop/1.py", line 19, in <module>
    print(car1. __weight)                       # AttributeErrorr
AttributeError: 'Car' object has no attribute '__weight'
```

在IDLE环境中，在对象或类名后面加上一个圆点“.”,会自动列出其所有公开成员，模块也具有同样的特点。如果在圆点“.”后面再加上一个下划线，则会列出该对象或类的所有成员，包括私有成员。

8.2.6 方法

在类中定义的方法可以粗略分为3大类：公有方法、私有方法、静态方法。其中，公有方法、私有方法都属于对象，私有方法的名字以两个下划线“__”开始，每个对象都有自己的公有方法和私有方法，在这两类方法中可以访问属于类和对象的成员；公有方法通过对象名直接调用，私有方法不能通过对象名直接调用，只能在属于对象的方法中通过“self”调用或在外部通过Python支持的特殊方式来调用。

例：公有方法、私有方法、静态方法的定义和调用。

```
class Fruit:
    price=0
    def __init__(self):
        self.__color='Red'          #定义和设置私有属性color
        self.__city='Kunming'         #定义和设置私有属性city
    def __outputColor(self):        #定义私有方法outputColor
        print(self.__color)         #访问私有属性color
    def __outputCity(self):           #定义私有方法outputCity
        print(self.__city)          #访问私有属性city
    def output(self):               #定义公有方法output
        self.__outputColor( )           #调用私有方法outputColor
        self.__outputCity( )        #调用私有方法outputCity
    @ staticmethod
    def getPrice():               #定义静态方法getPrice
        return Fruit.price
    @ staticmethod
    def setPrice(p):                #定义静态方法setPrice
        Fruit.price=p
#主程序
apple=Fruit()
apple.output()
print(Fruit.getPrice( ))
Fruit.setPrice(9)
print(Fruit.getPrice( ))
```

```
Red
Kunming
0
9
```

总结

总结

类的定义和使用、构造函数、析构函数的定义；成员变量的两种属性；实例属性和类属性；私有成员和公有成员的定义和使用；公有方法、私有方法和静态方法的定义和调用。

THANKS