python进阶——

面向对象编程

本章要点

- 面向对象概述
- 类和对象
- 继承
- 多态

面向对象概述

- 面向对象三大基本特征：
  - 封装：对象和类
    - 类是具有相同属性和行为的一组对象的集合
    - 对象是系统中用来描述客观事物的一个实体
  - 继承
    - 在现有类的基础上通过添加属性或方法来对现有类进行扩展。
  - 多态
    - 相同的操作或函数、过程可作用于多种类型的对象上并获得不同的结果

类的定义

- Python中定义类的语法格式如下：

class 类名：

[类变量]

[def __init__(self,paramers):]

[def 函数名(self,...)]

```
class fruit:  # 通过关键字class定义一个类
    name = 'apple'    # 定义类变量
    price=6.7
    # 定义了一个实例方法，方法至少有一个参数self
    def printName(self):
        print(self.name)
        print(self.price)
```

对象的创建


```
class fruit:
    name='apple'
    price=6.7
    def printName(self):
        print(self.name)
        print(self.price)
if __name__=='__main__':
    myfruit=fruit()
    myfruit.printName()
    print('fruit.name={0},fruit.price={1}'.format(fruit.name,fruit.price))
    myfruit.name='pear'
    myfruit.price=3.5
    myfruit.printName()
    print('fruit.name={0},fruit.price={1}'.format(fruit.name,fruit.price))
```

类变量

运行结果：

```
apple
6.7
fruit.name=apple,fruit.price=6.7
pear
3.5
fruit.name=apple,fruit.price=6.7
```

实例变量及封装

- 实例变量：实例化之后，每个实例单独拥有的变量
- 实例变量的定义如下：

self.变量名

- 调用实例变量有如下两种方式：

（1）在类外通过对象直接调用

（2）在类内通过self间接调用

```
class Point:
    def __init__(self,x,y):
        self.x=x
        self.y=y
p1=Point(0,0)
p1.x=2
p1.y=3
```

方法

实例方法

- 一般都以self作为第一个参数
- 必须和具体的对象实例进行绑定才能访问，即必须由对象调用
- 执行自动将调用该方法的对象赋值给self

```
class Foo:
    animal='兔子'
    def __init__(self,feature):
        self.feature=feature
    def print(self):
        print('调用了实例方法')
        print('{0}的特征是：{1}'.format(self.animal,self.feature))
    @classmethod
    def enemy(cls,name):
        print('调用了类方法')
        enemyName=name
        print('{0}的天敌是：{1}'.format(cls.animal,enemyName))
    @staticmethod
    def eat(name):
        print('调用了静态方法')
        eatName = name
        print('{0}的食物是：{1}'.format(Foo.animal, eatName))
```

方法

类方法

- 必须以cls作为第一个参数，cls表示类本身
- 定义时使用@classmethod装饰器
- 可以通过类名或实例对象名来调用
- 执行类方法时，自动将调用该方法的类赋值给cls参数。

```
class Foo:
    animal='兔子'
    def __init__(self,feature):
        self.feature=feature
    def print(self):
        print('调用了实例方法')
        print('{0}的特征是：{1}'.format(self.animal,self.feature))
    @classmethod
    def enemy(cls,name):
        print('调用了类方法')
        enemyName=name
        print('{0}的天敌是：{1}'.format(cls.animal,enemyName))
    @staticmethod
    def eat(name):
        print('调用了静态方法')
        eatName = name
        print('{0}的食物是：{1}'.format(Foo.animal, eatName))
```

方法

静态方法

- 不需要默认的任何参数，跟一般的普通函数类似
- 方法内不能使用任何实例变量
- 定义时使用@staticmethod装饰器
- 可以通过类名或实例对象名来调用

```
class Foo:
    animal='兔子'
    def __init__(self,feature):
        self.feature=feature
    def print(self):
        print('调用了实例方法')
        print('{0}的特征是：{1}'.format(self.animal,self.feature))
    @classmethod
    def enemy(cls,name):
        print('调用了类方法')
        enemyName=name
        print('{0}的天敌是：{1}'.format(cls.animal,enemyName))
    @staticmethod
    def eat(name):
        print('调用了静态方法')
        eatName = name
        print('{0}的食物是：{1}'.format(Foo.animal, eatName))
```

任务3-1 简单的购物车管理

- **任务描述**：编写一个python程序，设计一个类来管理用户的购物车
- **任务分析**：
  - 设计一个购物车类：
    - 定义构造方法
    - 定义私有实例变量__myCart：存放购物信息的列表对象
    - 定义实例方法：添加商品、修改商品数量、清空购物车、保存购物车信息、打开购物车等

- 在定义一个类的时候，也可以继承某个现有的类，新类被称为子类或派生类，而被继承的类则被称为父类或基类
- Python中定义类的继承的语法格式如下：

```
class 类名(父类名):
    [类变量]
    [def __init__(self,paramers):]
    [def 函数名(self,...)]
```

- Python中多态的方式有：

（1）通过继承机制，子类覆盖父类的同名方法，这样通过子类对象调用时，调用的是子类的覆盖方法

（2）定义类实例方法的时候，尽量把变量视作父类类型，这样，所有子类类型都可以正常被接收

（3）不同类具有的相同方法

预习

- GUI编程

THANK YOU