面向对象_05_面向对象封装案例 II

面向对象封装案例 II

— 黑马程序员《Python 入门教程完整版》笔记

目标

- 士兵突击案例
- 身份运算符

封装

1. **封装** 是面向对象编程的一大特点
2. 面向对象编程的 **第一步** —— 将 **属性** 和 **方法** **封装** 到一个抽象的 **类** 中
3. **外界** 使用 **类** 创建 **对象**，然后 **让对象调用方法**
4. **对象方法的细节** 都被 **封装** 在 **类的内部**

> 一个对象的 **属性** 可以是 **另外一个类创建的对象**

01. 士兵突击

需求

1. **士兵** **许三多** 有一把 AK47
2. **士兵** 可以 **开火**
3. **枪** 能够 **发射** 子弹
4. **枪** 装填 **装填子弹** —— **增加子弹数量**

Soldier
name gun
__init__(self): fire(self):

Gun
model bullet_count
__init__(self, model): add_bullet(self, count): shoot(self):

1.1 开发枪类

`shoot` 方法需求

- 1> 判断是否有子弹，没有子弹无法射击
- 2> 使用 `print` 提示射击，并且输出子弹数量

```
class Gun:

    def __init__(self, model):

        # 枪的型号
        self.model = model
        # 子弹数量
        self.bullet_count = 0

    def add_bullet(self, count):

        self.bullet_count += count

    def shoot(self):

        # 判断是否还有子弹
        if self.bullet_count <= 0:
            print("没有子弹了...")

            return

        # 发射一颗子弹
        self.bullet_count -= 1

        print("%s 发射子弹[%d]..." % (self.model, self.bullet_count))

# 创建枪对象
ak47 = Gun("ak47")
ak47.add_bullet(50)
ak47.shoot()
```

1.2 开发士兵类

> 假设：每一个新兵 都 **没有枪**

定义没有初始值的属性

在定义属性时，如果 **不知道设置什么初始值**，可以设置为 `None`

- `None` **关键字** 表示 **什么都没有**
- 表示一个 **空对象，没有方法和属性，是一个特殊的常量**
- 可以将 `None` 赋值给任何一个变量

`fire` 方法需求

- 1> 判断是否有枪，没有枪没法冲锋
- 2> 喊一声口号
- 3> 装填子弹
- 4> 射击

```
class Soldier:

    def __init__(self, name):

        # 姓名
        self.name = name
        # 枪，士兵初始没有枪 None 关键字表示什么都没有
        self.gun = None

    def fire(self):

        # 1. 判断士兵是否有枪
        if self.gun is None:
            print("[%s] 还没有枪..." % self.name)

            return

        # 2. 高喊口号
        print("冲啊...[%s]" % self.name)

        # 3. 让枪装填子弹
        self.gun.add_bullet(50)

        # 4. 让枪发射子弹
        self.gun.shoot()
```

小结

1. 创建了一个 **士兵类**，使用到 `__init__` 内置方法
2. 在定义属性时，如果 **不知道设置什么初始值**，可以设置为 `None`
3. 在 **封装的** 方法内部，还可以让 **自己的 使用其他类创建的对象属性** 调用已经 **封装好的方法**

02. 身份运算符

身份运算符用于 **比较** 两个对象的 **内存地址** 是否一致 —— **是否是对同一个对象的引用**

- 在 `Python` 中针对 `None` 比较时，建议使用 `is` 判断

运算符	描述	实例
is	is 是判断两个标识符是不是引用同一个对象	x is y，类似 id (x) == id (y)

is not	is not 是判断两个标识符是不是引用不同对象	x is not y，类似 id (a) != id (b)

is 与 == 区别：

`is` 用于判断 **两个变量 引用对象是否为同一个**
`==` 用于判断 **引用变量的值** 是否相等

```
>>> a = [1, 2, 3]
>>> b = [1, 2, 3]
>>> b is a
False
>>> b == a
True
```