## 1.多继承以及MRO顺序

多继承指的是子类继承多个父类，可以通过三种方式访问父类的方法：

**父类名.父类方法(self)**：这种方式容易造成父类方法被调用多次的问题，而且一旦父类名称发生变化，子类调用的地方都需要修改。

**super(指定某个类名, self).父类方法()**：从指定类名的MRO下一级开始调用

**super().父类方法()**：按照MRO顺序查找上级父类的方法。

「方法解析顺序」（Method Resolution Order简称MRO）

可以通过类名.\_\_MRO\_\_属性查找出来当前类的调用顺序，其顺序由C3算法来决定，保证每一个类只调用一次。

单继承用哪种方式调用父类方法都可以，基本上无差别，但是**建议super()**的方式。

## 2.类属性和实例属性的区别

在Python中一切皆对象，类是一个特殊的对象即**类对象**，描述类的属性称为类属性，它属于类。类属性在内存中只有一份，所有实例对象公用。在\_\_init\_\_外部定义。

实例属性用来描述类创建出来的实例对象，需要通过对象来访问，在各自对象的内存中都保存一份。在\_\_init\_\_方法内部定义

## 3.实例方法、类方法、静态方法的区别

这三种方法都是保存在**类的内存**中，调用者不同。

实例方法由对象调用，至少一个self参数。

类方法由类调用，至少一个cls参数，并且需要装饰器@classmethod修饰

静态方法由类调用，不需要参数，需要装饰器@staticmethod修饰

## 4.property属性

### 4.1定义

一种用起来像是使用的实例属性一样的特殊属性，可以对应于某个方法，通过使用property属性，能够简化调用者在获取数据的流程。

property属性的定义和调用要注意以下几点：

定义时，在实例方法的基础上添加 @property 装饰器；并且**仅有一个**self参数

调用时，无需括号，加上就错了。

### 4.2实现property属性的两种方式

#### 4.2.1装饰器

经典类中的属性只有一种访问方式，其对应被 @property 修饰的方法，即只能读取。

新式类中的属性有三种访问方式，并分别对应了三个被@property、@方法名.setter、@方法名.deleter修饰的方法，分别对应读取、修改和删除属性的方法。

#### 4.2.2类属性

当使用类属性的方式创建property属性时，经典类和新式类无区别

property()方法有四个参数

第一个参数是方法名，调用 对象.属性 时自动触发执行方法

第二个参数是方法名，调用 对象.属性 ＝ XXX 时自动触发执行方法

第三个参数是方法名，调用 del 对象.属性 时自动触发执行方法

第四个参数是字符串，调用 对象.属性.\_\_doc\_\_ ，此参数是该属性的描述信息

## 5.常见的魔方属性、方法

\_\_doc\_\_ 表示类的描述信息

\_module\_\_ 表示当前操作的对象在那个模块

\_\_class\_\_ 表示当前操作的对象的类是什么

\_\_new\_\_ 创建对象时为对象分配空间，在初始化方法\_\_init\_\_之前被调用

\_\_init\_\_ 初始化方法，通过类创建对象时，自动触发执行

\_\_del\_\_ 当对象在内存中被释放时，自动触发执行

\_\_call\_\_ 对象后面加括号，触发执行，例如对象()或者类名()()

\_\_dict\_\_ 类或对象中的所有属性

\_\_str\_\_ 在打印对象时，默认输出该方法的返回值(字符串)

## 6.with与“上下文管理器”

任何实现了 \_\_enter\_\_() 和 \_\_exit\_\_() 方法的对象都可称之为**上下文管理器**，上下文管理器对象可以使用 with 关键字。Python 提供了 with 语法用于简化资源操作的后续清除操作，是 try/finally 的替代方法，实现原理建立在上下文管理器之上。

使用的with open(文件名) as f:这种语法可以在文件调用结束或者文件操作异常的时候自动关闭文件