类有三个部分组成

1. 类的名字：类名
2. 类属性：一组数据 （类似函数内的变量）
3. 类的方法：允许进行的操作方法（行为）



对象抽象就是类

类的实例化就是对象

Python定义一个类

Class <类名>: #类名首字母大写

<语句>

Python定义一个类顺便继承父类

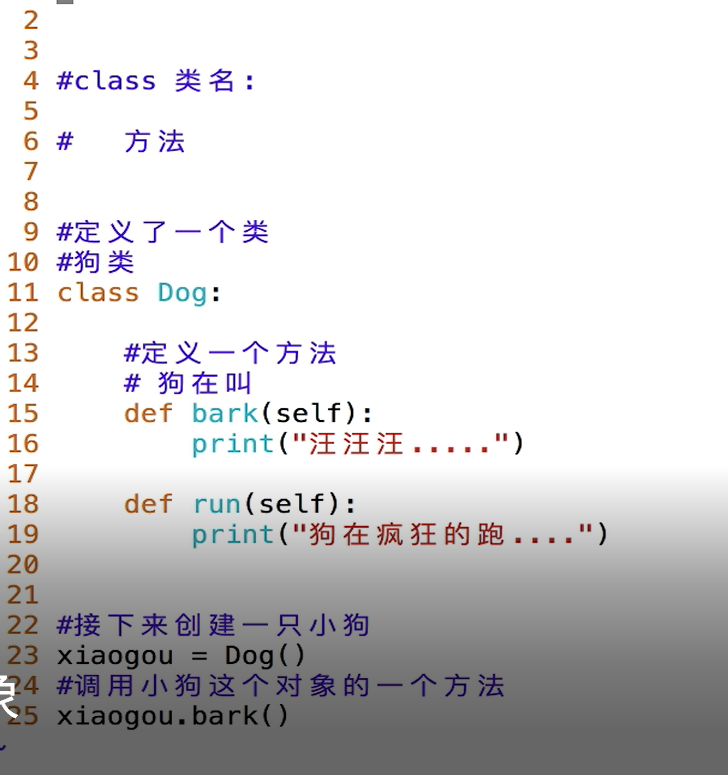
Class 类名():

<语句>

Class <类名>:

Name = 0 #定义属性

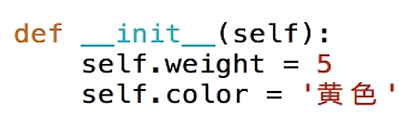
\_weight = 0 #定义私有属性私有属性类在外部无法直接访问



添加属性和获取属性



类里的特殊方法def \_\_init\_\_(self): #左右多是2个下斜杠 用来初始化这个类在这个方法里面加上属性，这个方法是这个类最开始生成对象后默认运行这个方法所以一般在下面上加上属性。



但是当有一个类，多个对象的话怎么办，这时我们需要把固定的属性改成变量，在创建对象时把实参传进去。

class Dog(): #定义类

def \_\_init\_\_(self,newweight,newcolor): #创建方法

self.weight = newweight

self.newcolor = newcolor

dogg = Dog(5,’yellow’) #创建对象并且传入属性 5对应newweigh yellow对应newcolor

self就是对象

如果直接print(对象) 会打印出他的内存地址，如果你想要print(对象) 输出特殊的东西这时你需要一个方法 \_\_str\_\_(self): #两个下划线

class Dog:

\_\_str\_\_(self):

Return “。。。”

这时你print(对象) 直接打印出return的内容

在python中方法名字是\_\_xxx\_\_（self）: 这样的方法都是特殊方法叫魔法方法，你定义对象的时候自动执行

一般调用方法/属性的格式是

对象.方法/属性()

面对对象的3大特性

封装,继承,多态

1 封装顾名思义就是将内容封装到某一个地方，然后再调用封装的内容

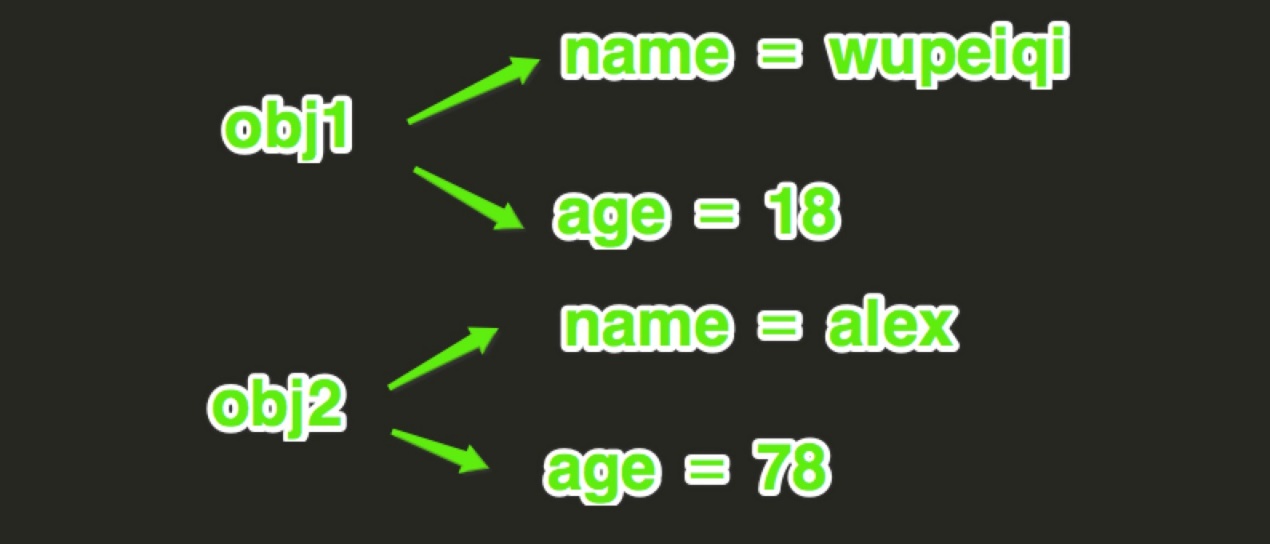


Self是一个形式参数

当执行 obj1 = Foo(‘wupeiqi’, 18 ) 时，self 等于 obj1

当执行 obj2 = Foo(‘alex’, 78 ) 时，self 等于 obj2

所以，内容其实被封装到了对象 obj1 和 obj2 中，每个对象中都有 name 和 age 属性，在内存里类似于下图来保存。



上图展示了对象 obj1 和 obj2 在内存中保存的方式，根据保存格式可以如此调用被封装的内容：对象.属性名

class Foo:

    def \_\_init\_\_(self, name, age):

        self.name = name

        self.age = age

obj1 = Foo('wupeiqi', 18)

print obj1.name    # 直接调用obj1对象的name属性

print obj1.age     # 直接调用obj1对象的age属性

obj2 = Foo('alex', 73)

print obj2.name    # 直接调用obj2对象的name属性

print obj2.age     # 直接调用obj2对象的age属性

2、通过self间接调用被封装的内容

执行类中的方法时，需要通过self间接调用被封装的内容

class Foo:

    def \_\_init\_\_(self, name, age):

        self.name = name

        self.age = age

    def detail(self):

        print self.name

        print self.age

obj1 = Foo('wupeiqi', 18)

obj1.detail()  # Python默认会将obj1传给self参数，即：obj1.detail(obj1)，所以，此时方法内部的 self ＝ obj1，即：self.name 是 wupeiqi ；self.age 是 18

obj2 = Foo('alex', 73)

obj2.detail()  # Python默认会将obj2传给self参数，即：obj1.detail(obj2)，所以，此时方法内部的 self ＝ obj2，即：self.name 是 alex ； self.age 是 78

得到需求先考虑清楚

比如”烤地瓜”

先分析烤地瓜的属性

熟的程度cookedlevel （属性） 0-3是生的超过3是半生不熟，5已经考好了，超过8就已经成炭了

但是你怎么表达考的熟的程度了？ 不要说用3啊5啊8 啊表示这样别人看不懂这时我们再用一个属性

Cookstring：这是字符串表示，地瓜熟的程度

烤地瓜没有配料怎么行这时我们需要配料

condiments：这是地瓜的配料表

。。。。。。。

。。。。。。。

。。。。。。。

这时我们再定义方法

cook（） 代表烤地瓜的时间传入的参数是时间

addcondiments() 代表给地瓜添加配料

当然我们上面写的大部分都是经典类也就是 class xxx: 输入类名后这直接打上冒号，

而python3有新式类也就是 class xxx(object):差别差不多

私有属性

在定义一个属性的时候我们可以通过类外边传递参数访问属性，这时我们也可以做到外面不能修改和访问属性，

首先在定义类后会定义\_\_init\_\_方法，这个方法主要做初始化

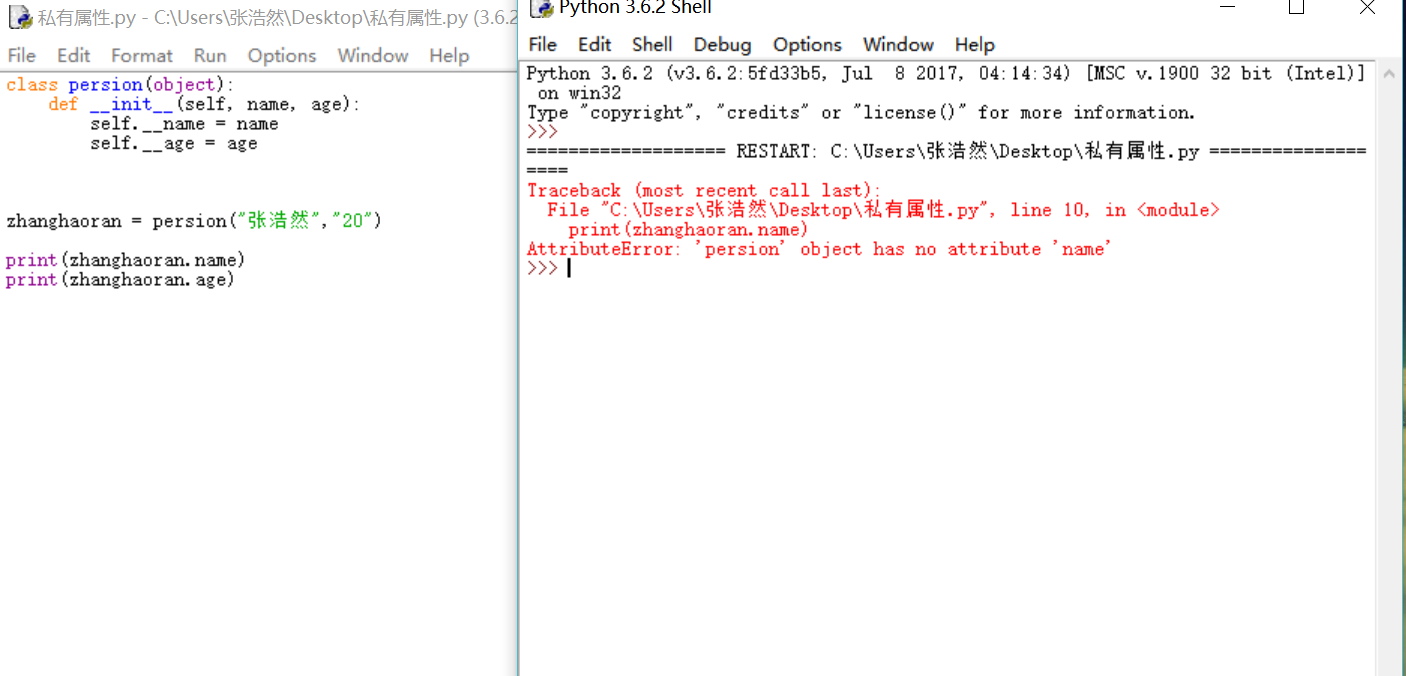
然后再这个方法中定义属性

class Persion(object):

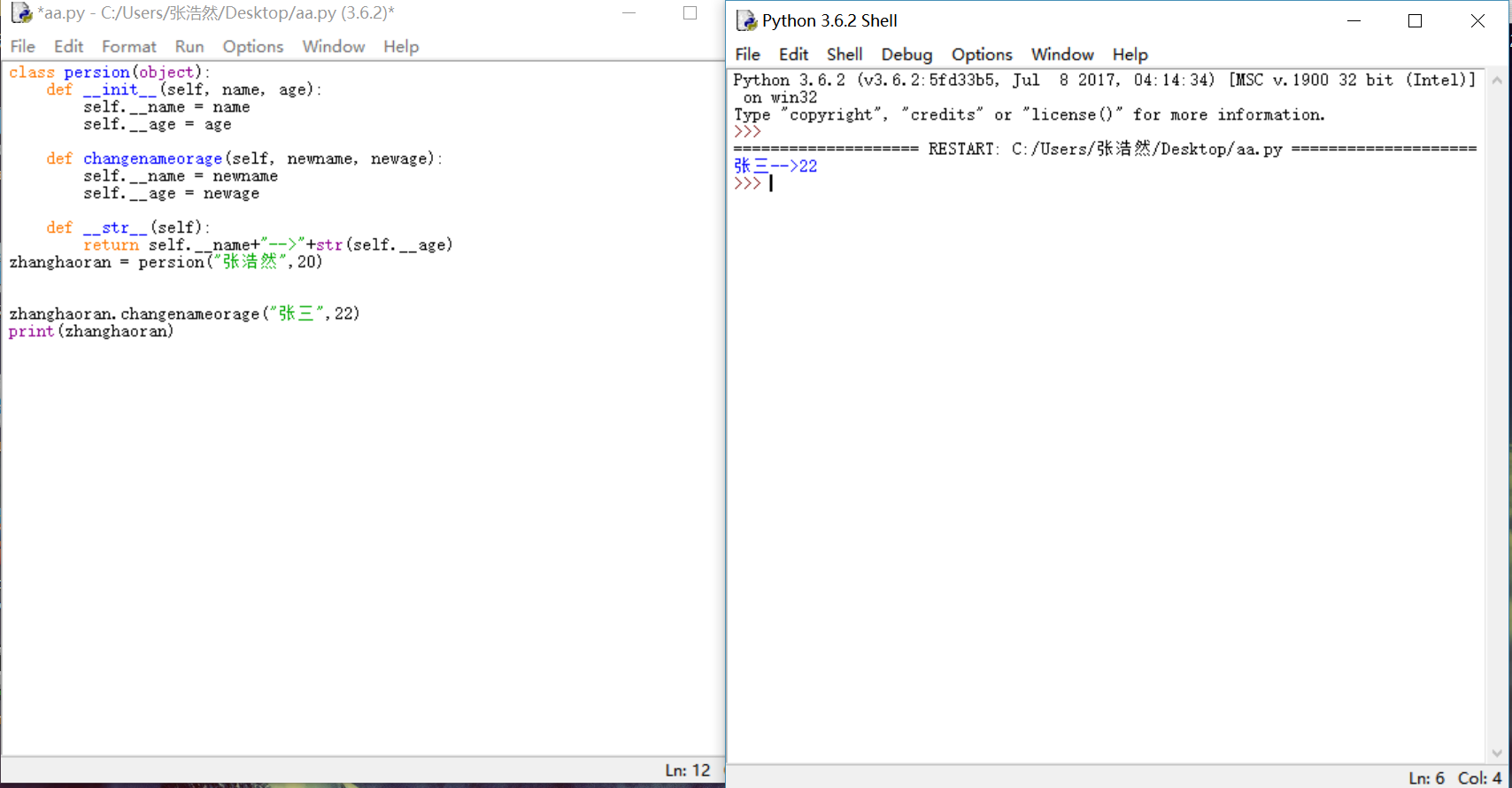
def \_\_init\_\_(self, high, weigh):

self.\_\_high = high

self.\_\_weigh =weigh #定义一个私有属性在属性的前面加上\_\_ 两个下横杠 这时我们从外面不能去更改他，如果我们要更改只能从内部更改，最好的方法是定义一个方法专门用来更改私有属性



访问拒绝



访问成功

\_\_del\_\_()方法

\_\_del\_\_()方法和\_\_init\_\_()方法相反 init方法是在对象初始化的时候调用del在类被删除的时候调用但是什么时候类会删除对象了？是在程序运行完后python解释器发现程序没有了会kill掉对象释放空间，就是在这个kill对象的过程中kill到最后看见还有一个del方法这时候执行这个方法，或者使用del这个函数删除对象

继承

继承分单继承和多继承、

继承就是一个类继承另一个类的方法和属性（不包括私有属性）

格式

Class 子类名(父类名):

在上面我们用了新式类 class 类名(object): 意思为继承object类，什么事object类了？ Object类是python默认的类所有的类直接或者间接的继承了他

名词（基类，派生类）

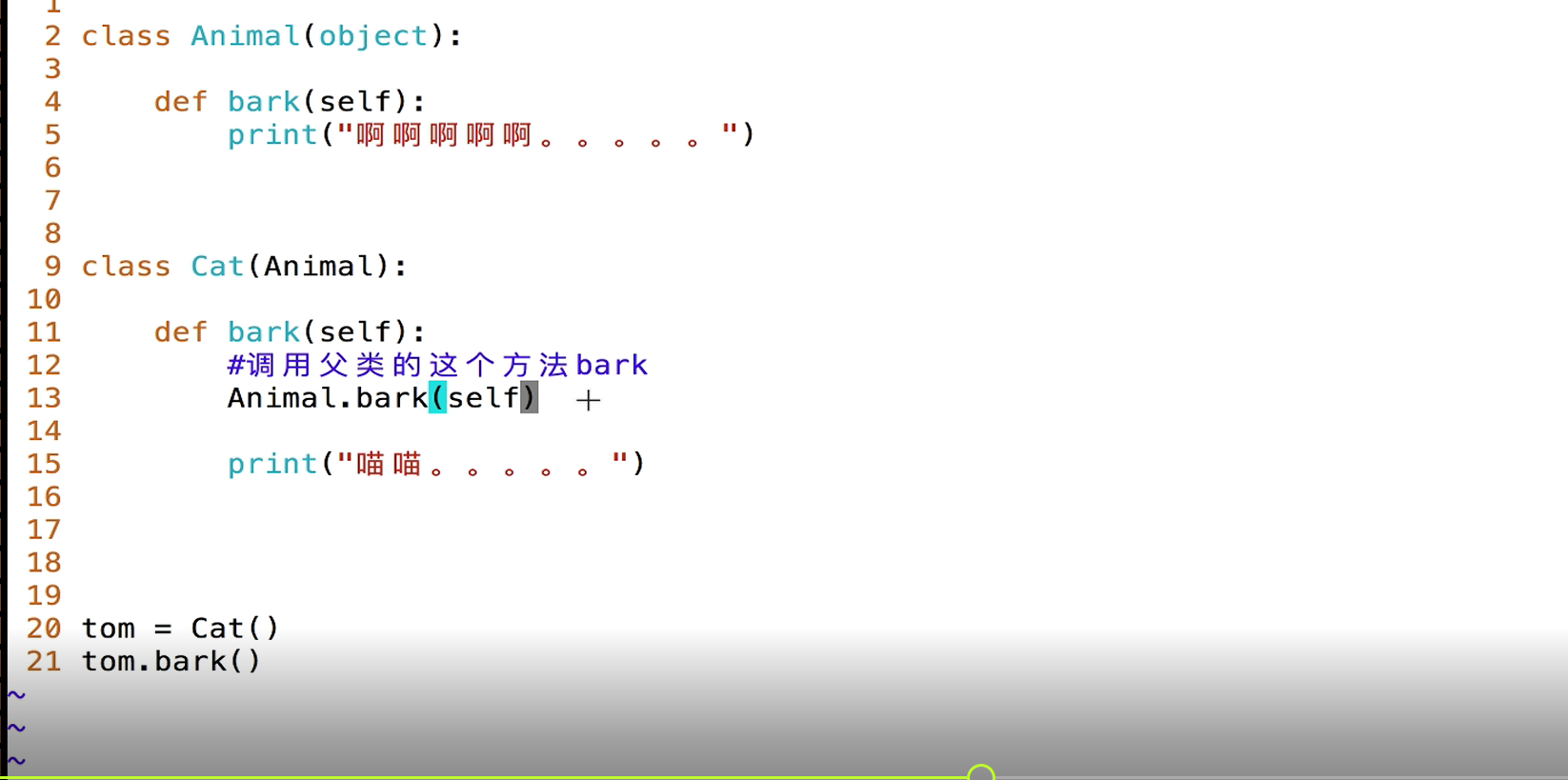
基类和派生类只是相对来说的，一个父类对于子类而言就是基类 一个子类对于父类而言是派生类

如果你的子类是想访问父类中的私有属性的话，你可以在父类中定义一个方法，在方法中访问自己的私有属性然后，子类继承父类，在继承过程中子类也继承了父类的所有方法包括访问私有属性的方法，这是直接调用即可

如果在一个方法名字前面加了两个下划线\_\_那么这个方法就成了私有方法（和私有属性一样在外部不能访问不能调用包括子类）

子类重新写父类的方法

首先子类发现父类的方法不适合我子类我们可以重新写这个方法，哼简单就直接在子类里面定义相同的方法名字然后写入自己的类容即可，也可以在子类中调用父类的方法



多继承

就是继承多个类，一样如果有私有方法或者属性，照样不可以继承

class A(object):

def …(self):

…

class B(object):

def …(self):

…

class C(A, B): #C作为子类继承了2个父类（除了私有类不可以继承）

pass #占位，不写入方法和属性

如果子类继承的两个父类有相同的方法名默认继承第一个类的方法名（ class C(A, B): 定义C子类的时候定义的第一个父类 ），如果第一个类没有需要调用的就会跳到第二个类调用方法，如果第二个方法也没有。。直接报错，因为你使用的方法没有出现在你继承的两个父类中（除非你还有一个爷类，，爷类有这个方法就调用爷类的这个方法，爷类就是你没有直接继承他，而是你的2个父类，或者一个父类继承的类也就是广度遍历）

广度遍历（python3默认的）

首先找自己继承的类调用他们的方法，如果这个第一个父类没有，就会找第二个父类，如果第二个父类也没有就找第3个。。。。。，如果自己继承的父类都没有就找爷爷类（间接继承的类）爷爷类的顺序是第一个父类的父类，他没有就找第二个父类的父类以此类推，只到顶还没有就报错。

深度遍历（python2默认的）

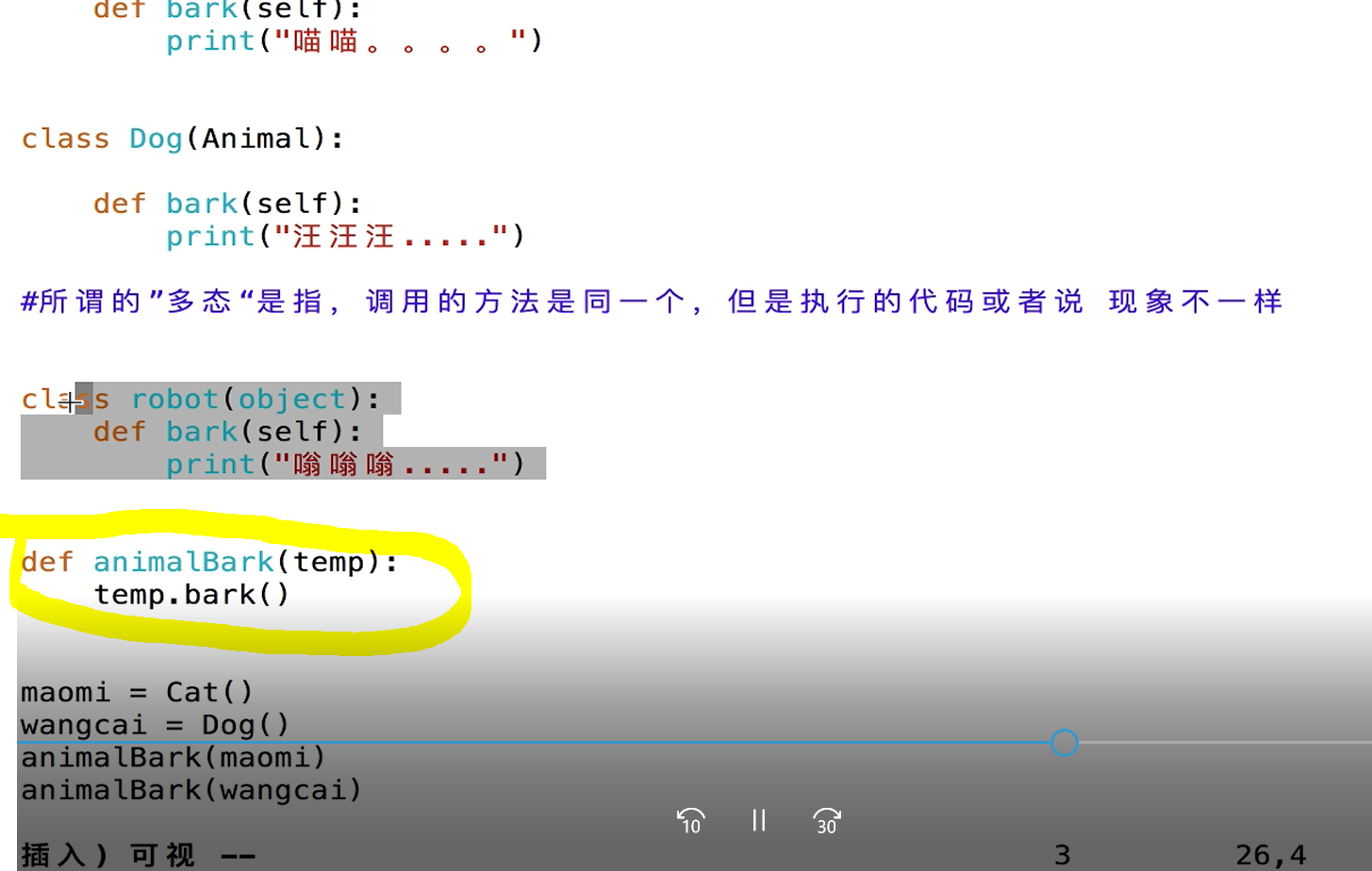
也就是说继承多个父类而父类又继承多个父类，这时调用方法的查找方式是先找第一个父类，然后再找这个父类的父类直到这个父类的分支都没有，再找另外一个父类的分支以此类推。

如果想查看遍历的顺序可以使用方法\_\_mro\_\_ print(c.\_\_mro\_\_) #只有python3里面有。

多态

指调用的方法是同一个但是执行的代码不是一样的

我们只需要定义一个函数，函数传入对象，内容就是方法即可



类属性和实例属性

类属性就是在创建类对象（类）时def上一级并且在类里面的属性下面aaa就是类属性，他像一个公有属性一样，对类对象通用

Class Dog（object）：

aaa =1

def \_\_init\_\_(self):

self.AAA = 2

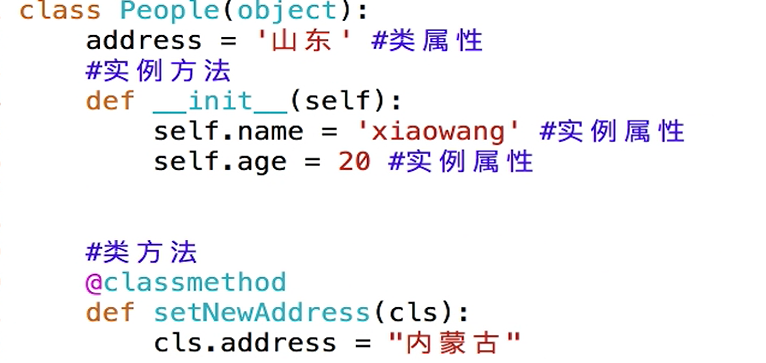
实例属性就是在\_\_init\_\_方法里面的属性专门对对象使用的属性也就是上面的AAA

当实例对象（就是对象）查找属性的时候先查找实例属性实例属性没有，就查找类属性

在def 定义方法前加上@classmethod

这是类方法不是实例方法

类方法顾名思义就是类用的方法



类方法里面的cls和self一样

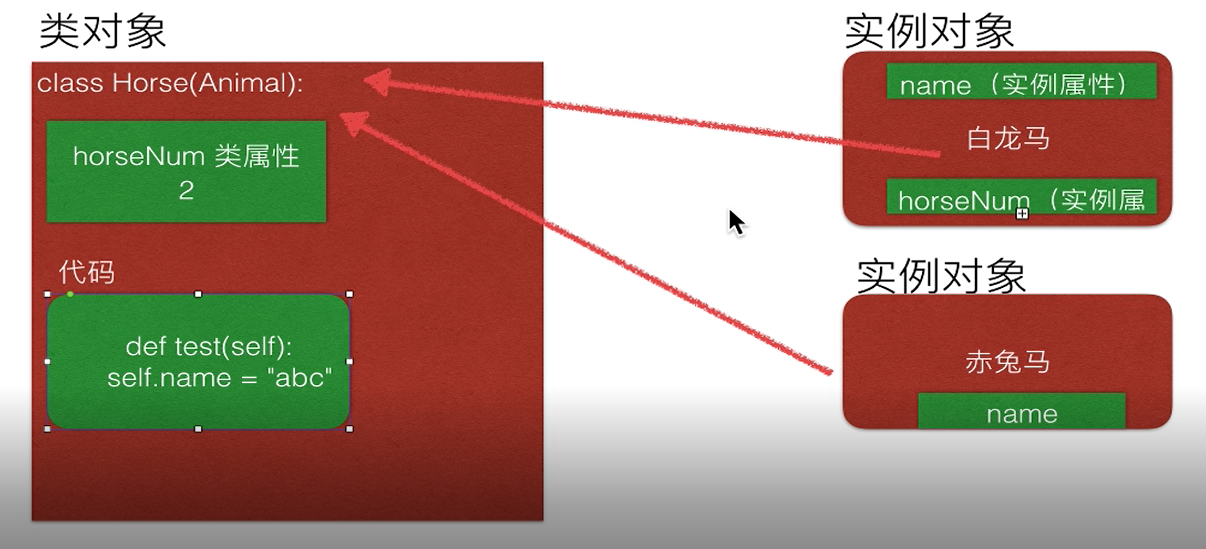
调用也是 类对象.类方法()/类属性: (以前写的都是实例方法是 实例对象.实例方法/类属性)

还有一个重要的规则

实例对象可以访问类方法/实例方法/类属性（不能更改因为类属性在类对象的内存空间中不在实例对象内存空间中一更改的话那么就相当于增加一个实例属性）/实例属性

#注如果一个 类对象.对象属性 = xxx 这个对象属性不存在那么会自动的增加这个属性，这个属性就是对象属性

类对象只可以访问 类方法/类属性

内存拓扑图 梳理对象都可以调用类对象的代码也就是类方法，但是不可以调用类属性。