变量中记录数据的地址，叫做引用

在python中，函数的实参/返回值都是靠引用来传递的

用id(变量)可以查看变量所在的内存地址

数字类型(int,bool,float,complex)、字符串(str)和元组(tuple)改变则内存地址改变，即内存中的数据不允许被修改，是不可变类型；

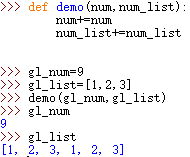
列表(list)、字典(dict)内容改变内存地址不变，即内存中的数据可以被修改，是可变类型

如果在函数中修改全局变量，需要使用global进行声明

Python转有利用元组交换两个变量的值：a, b = b, a

1、在函数内部，针对参数使用赋值语句，不管是可变参数还是不可变参数，都不会影响调用函数时传递的实参变量。只要是针对参数使用赋值语句，会在函数内部修改局部变量的引用，不会影响到外部变量的引用

2、如果使用+=在函数内进行修改，则对于可变类型相当于extend()，对于不可变相当于赋值



缺省参数：

定义函数时，可以给某个参数指定一个默认值，具有默认值的参数叫做缺省参数

带有默认值的缺省参数必须放在参数列表末尾，以下是错误的

def print\_info(name, gender=True, title):

在函数调用时，如果有多个缺省参数，需要指定参数名

print\_info("老王", title="班长")

print\_info("小美", gender=False)

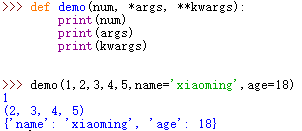
多值参数：

有时可能需要一个函数能够处理的参数个数时不确定的，这个时候，可以使用多指参数

python中有两种多值参数：

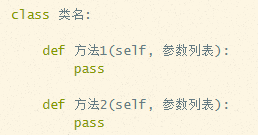
参数名前加一个\*可以接收元组，\*args

参数名前加两个\*可以接收字典，\*\*kwargs



dir(标识符/数据)查看对象内的所有属性和方法

定义只包含方法的类:



方法的定义格式和之前学习过的函数几乎一样，区别在于第一个参数必须是self

创建对象：对象变量 = 类名()

python使用类创建对象之后，对象变量记录的是对象再内存中的地址（十六进制表示内存地址）

%d可以10进制输出数字，%x可以16进制输出数字

在实际开发中，对象的某些属性或方法 可能只希望在对象的内部被使用，而不希望在外部被访问到。

在定义属性或方法时，在属性名或者方法名前增加两个下划线，定义的就是私有属性或方法

class Women：

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

self.name = name

self.\_\_age=18

def \_\_secret(self):

print(“我的年龄是%d” % self.\_\_age)

xiaofang = Women(“小芳”)

#print(xiaofang.\_\_age) --不能直接访问

#xiaofang.\_\_secret() --不能直接调用

继承：

重写：1.覆盖父类方法 2.对父类方法进行扩展

扩展父类方法：super()调用父类封装的方法，用来扩展父类方法

def bark(self): #重写方法

super().bark() #调用父类封装的方法

print(“”) #增加子类其他方法

object是python为所有对象提供的基类，提供一些内置的属性和方法，可以用dir函数查看

python 3.x中定义类时，如果没有指定父类，会默认使用object作为该类的基类；

今后在定义类时，如果没有父类，建议统一继承自object

class 类名(object):

pass

子类对象不能在自己的方法内部直接访问父类的私有属性和私有方法

子类对象可以通过父类的公有方法间接访问到私有属性或私有方法

私有属性、方法是对象的隐私，不对外公开

私有属性、方法通常用于做一些内部的事情

多态:

def \_\_init\_\_(self)相当于构造函数，可以带参数，\_\_init\_\_(self, name)，这种技术可以在创建对象的同时给对象赋初值，即对象初始化

调用对象方法：可以通过self. 访问自己的属性，调用自己的方法

<https://www.cnblogs.com/dianqijiaodengdai/p/6144698.html>

**类属性：给类对象定义的属性，通常用来记录与这个类相关的特征，类属性不会用于记录具体对象的特征。**

**访问类属性的两种方式：类名.类属性；对象.类属性（不推荐）**

**如果使用对象.类属性 = 值赋值语句，只会给对象添加一个属性，而不会影响到类属性的值**

**类方法：**

**@classmethod**

**def 类方法名(cls):**

**pass**

**cls和self类似，调用类方法时，不需要传递cls参数**

**在方法内部，可以通过cls.访问类的属性，也可以通过cls.调用其他的类方法**

**静态方法：如果需要在类中封装一个方法，这个方法既不需要访问实例属性或者调用实例方法，也不需要访问类属性或者调用类方法，这个时候，可以把这个方法封装成一个静态方法。**

**静态方法：**

**class Dog(object): #不继承默认写object**

**dog\_count = 0**

**@staticmethod**

**def run():**

**print(“dor is running”) #不需要访问实例属性也不需要访问类属性的方法**

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

**self.name = name**