**1.python的函数参数传递**

看两个例子:

a = 1

def fun(a):

a = 2

fun(a)

print a # 1

a = []

def fun(a):

a.append(1)

fun(a)

print a # [1]

所有的变量都可以理解是内存中一个对象的“引用”，或者，也可以看似c中void\*的感觉。

通过id来看引用a的内存地址可以比较理解：

a = 1

def fun(a):

print "func\_in",id(a) # func\_in 41322472

a = 2

print "re-point",id(a), id(2) # re-point 41322448 41322448

print "func\_out",id(a), id(1) # func\_out 41322472 41322472

fun(a)

print a # 1

注：具体的值在不同电脑上运行时可能不同。

可以看到，在执行完a = 2之后，a引用中保存的值，即内存地址发生变化，由原来1对象的所在的地址变成了2这个实体对象的内存地址。

而第2个例子a引用保存的内存值就不会发生变化：

a = []

def fun(a):

print "func\_in",id(a) # func\_in 53629256

a.append(1)

print "func\_out",id(a) # func\_out 53629256

fun(a)

print a # [1]

这里记住的是类型是属于对象的，而不是变量。而对象有两种,“可更改”（mutable）与“不可更改”（immutable）对象。在python中，strings, tuples, 和numbers是不可更改的对象，而 list, dict, set 等则是可以修改的对象。(这就是这个问题的重点)

当一个引用传递给函数的时候,函数自动复制一份引用,这个函数里的引用和外边的引用没有半毛关系了.所以第一个例子里函数把引用指向了一个不可变对象,当函数返回的时候,外面的引用没半毛感觉.而第二个例子就不一样了,函数内的引用指向的是可变对象,对它的操作就和定位了指针地址一样,在内存里进行修改.

2．简述ORM原理