链滴

# python 是如何进行参数传递的?

作者：zyjImmortal
原文链接：https://ld246.com/article/1589204243229
来源网站：链滴

在分析python的参数传递是如何进行的之前，我们需要先来了解一下，python变量和赋值的基本原，这样有助于我们更好的理解参数传递。

# python变量以及赋值

- 数值

从几行代码开始

```
In [1]: a = 1

In [2]: b = a

In [3]: a = a + 1
```

我们先将1赋值给a，也就是a指向了1这个对象，**在python中一切皆对象**。接着b=a，则表示让b也向了1这个对象，python中一个对象是可以被多个引用所指向。最后执行的a=a+1,这里需要注意一，python的数据类型中如int、str等不可变类型，执a=a+1这种操作，并不是把a指向的对象的值增加，而是生成一个新的对象2，并让a指向2这个对象，原来的对象还存在于内存中。那这里的话还是会指向，我们来分别看一下a和b的值：

```
In [4]: a
Out[4]: 2

In [5]: b
Out[5]: 1
```

通过这个例子你可以看到，这里的 a 和 b，开始只是两个指向同一个对象的变量而已，或者你也可以它们想象成同一个对象的两个名字。简单的赋值 b = a，并不表示重新创建了新对象，只是让同一个象被多个变量指向或引用。同时，指向同一个对象，也并不意味着两个变量就被绑定到了一起。如果给其中一个变量重新赋值，并不会影响其他变量的值。

- 列表

还有一个列表的例子，再来瞅瞅：

```
In [6]: l1 = [3,4,5,6]

In [7]: l2 = l1

In [10]: l1.append(7)

In [11]: l1
Out[11]: [3, 4, 5, 6, 7]

In [12]: l2
Out[12]: [3, 4, 5, 6, 7]
```

代码中，我们让l1和l2这两个变量都指向了[3,4,5,6]这个对象，我们知道列表是一种可变的数据结构所以append操作并不会产生新的对象，只是在末尾添加了一个元素，变成了[3, 4, 5, 6, 7]，由于 l1 和 l2 同时指向这个列表，所以列表的变化会同时反映在 l1 和 l2 这两个变量上，那么，l1 和 l2 的值就时变为了[3, 4, 5, 6, 7]。

- 对象删除

python中变量是可以删除的，但是对象是没办法删除的

```
In [22]: a = [1,4,5]

In [23]: del a
```

del语句删除a这个变量，就无法通过a访问[1,4,5]，但是这个对象在存在中还是存在的，python的垃回收机制发现引用为0的时候就会把它回收掉。

- 总结

  - 变量的赋值，只是表示让变量指向了某个对象，并不表示拷贝对象给变量；而一个对象，可以多个变量所指向
  - 可变对象（列表，字典，集合等等）的改变，会影响所有指向该对象的变量
  - 对于不可变对象（字符串、整型、元组等等），所有指向该对象的变量的值总是一样的，也不改变。但是通过某些操作（+= 等等）更新不可变对象的值时，会返回一个新的对象
  - 变量可以被删除，但是对象无法被删除

# python函数是如何进行参数传递的

python的参数传递是赋值传递或者说是引用传递，python里一切皆对象，所以参数传递时，只是让变量与原变量指向了同一个对象，下面我们来看个例子：

```
In [28]: def func(b):
    ...:     b = 2

In [29]: a = 1

In [30]: func(a)
```

```
In [31]: a
Out[31]: 1
```

这里的参数传递，使变量 a 和 b 同时指向了 1 这个对象。但当我们执行到 b = 2 时，系统会重新创一个值为 2 的新对象，并让 b 指向它；而 a 仍然指向 1 这个对象。所以，a 的值不变，仍然为 1。

如何改变a的值呢?

我们可以在函数中将b返回

```
def func(b):
    b = 2
    return b
a = 1
a = func(a)
a
2
```

上面的例子我们的是int类型的，下面我们看一下列表的例子：

```
def func(l2):
    l2.append(77)

l1 = [12,3,6]
func(l1)
l1
[12,3,6,77]
```

这里 l1 和 l2 先是同时指向值为[1, 2, 3]的列表。不过，由于列表可变，执行 append() 函数，对其末加入新元素 4 时，变量 l1 和 l2 的值也都随之改变了.

那大家看一下面的例子，结果是什么呢?

```
def func(l2):
    l2 = l2 + [4]

l1 = [12,3,6]
func(l1)
l1
[12,3,6]
```

可以看到，l1没有发生变化，原因是l2 + [4]这个操作表示创建了一个末尾加入元素 4的新列表，并让 l 指向这个新的对象，l1还是指向原有的对象。

# 总结

今天，我们讨论了 Python 的变量及其赋值的基本原理，并且解释了 Python 中参数是如何传递的。其他语言不同的是，Python 中参数的传递既不是值传递，也不是引用传递，而是赋值传递，或者是对象的引用传递。需要注意的是，这里的赋值或对象的引用传递，不是指向一个具体的内存地址，而指向一个具体的对象。

**本文链接：https://immortalp.com/articles/2020/05/11/1589204241941.html**