06. 万恶之源-再谈编码

本节主要内容:

1. is和==的区别
2. 编码的问题

一. is和==的区别

1. id()

通过id()我们可以查看到一个变量表示的值在内存中的地址.

```
s = 'alex'
print(id(s))   # 4326667072
s = "alex"
print(id(s))   # 4326667072


lst = [1, 2, 4]
print(id(lst))      # 4326685768

lst1 = [1, 2, 4]
print(id(lst1))      # 4326684360

# 我们发现. 字符串的数据地址是一样的. 而 列表的数据地址是不一样的.

tup = (1, 2)
tup1 = (1, 2)
print(id(tup))
print(id(tup1))

print(id("哈哈哈呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵"*100))
print(id("哈哈哈呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵"*100))

s1 = "00000000000000000000000000000000000000000哈哈呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵哈哈哈呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵"
s2 = "00000000000000000000000000000000000000000哈哈呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵
```

```
呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵哈哈哈呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵"
print(id(s1))
print(id(s2))
print(s1 is s2)

a1 = str("alexalexalex"+"abcdef1111abcdefgabcdef1111abcdefgabcdef1111abcdefgabcdef1111abcdefgabcdef1111abcdefgabcdef1111abcdefgabcdef1111abcdefgabcdef1111abcdefgabcdef1111abcdefg")
print(id(a1))
a2 = str("alexalexalex"+"abcdef1111abcdefgabcdef1111abcdefgabcdef1111abcdefgabcdef1111abcdefgabcdef1111abcdefgabcdef1111abcdefgabcdef1111abcdefgabcdef1111abcdefgabcdef1111abcdefg")
print(id(a2))

s1 = "@1 2 "
s2 = "@1 2 "
print(id(s1))
print(id(s2))     # 结果一致, 但是在终端中是不一致的. 所以在python中,命令行代码和py文件中的代码运行的效果可能是不一样的
```

小数据池(常量池): 把我们使用过的值存储在小数据池中.供其他的变量使用.
小数据池给数字和字符串使用, 其他数据类型不存在.
对于数字: -5~256是会被加到小数据池中的. 每次使用都是同一个对象.
对于字符串:

1. 如果是纯文字信息和下划线. 那么这个对象会被添加到小数据池
2. 如果是带有特殊字符的. 那么不会被添加到小数据池. 每次都是新的
3. 如果是单一字母*n的情况. 'a'*20, 在20个单位内是可以的. 超过20个单位就不会添加到小数据池中

注意(一般情况下): 在py文件中. 如果你只是单纯的定义一个字符串. 那么一般情况下都是会被添加到小数据池中的. 我们可以这样认为: 在使用字符串的时候, python会帮我们把字符串进行缓存, 在下次使用的时候直接指向这个字符串即可. 可以节省很多内存.

这个问题千万不要纠结. 因为官方没有给出一个完美的结论和定论.所以只能是自己摸索.

以下内容摘自官网中关于id()的描述

> Return the “identity” of an object. This is an integer which is guaranteed to be unique and constant for this object during its lifetime. Two objects with non-overlapping lifetimes may have the same `id()` value.
>
> **CPython implementation detail:** This is the address of the object in memory.

说了这么多. 这个id()和is有什么关系呢. 注意. is比较的就是id()计算出来的结果. 由于id是帮我们查看某数据(对象) 的内存地址. 那么is比较的就是数据(对象)的内存地址.
最终我们通过is可以查看两个变量使用的是否是同一个对象.

== 双等表示的是判断是否相等, 注意. 这个双等比较的是具体的值.而不是内存地址

```
s1 = "哈哈"
s2 = "哈哈"
print(s1 == s2)  # True
print(s1 is s2)  # True 原因是有小数据池的存在 导致两个变量指向的是同一个对象

l1 = [1, 2, 3]
l2 = [1, 2, 3]
print(l1 == l2)  # True, 值是一样的
print(l1 is l2)  # False, 值是假的
```

总结:
is 比较的是地址
== 比较的是值

二. 编码的补充

1. python2中默认使用的是ASCII码. 所以不支持中文. 如果需要在Python2中更改编码. 需要在文件的开始编写:

```
# -*- encoding:utf-8 -*-
```

2. python3中: 内存中使用的是unicode码.

编码回顾:

1. ASCII : 最早的编码. 里面有英文大写字母, 小写字母, 数字, 一些特殊字符. 没有中文, 8个01代码, 8个bit, 1个byte

2. GBK: 中文国标码, 里面包含了ASCII编码和中文常用编码. 16个bit, 2个byte
3. UNICODE: 万国码, 里面包含了全世界所有国家文字的编码. 32个bit, 4个byte, 包含了ASCII
4. UTF-8: 可变长度的万国码. 是unicode的一种实现. 最小字符占8位

    1.英文: 8bit 1byte
    2.欧洲文字:16bit 2byte
    3.中文:24bit 3byte

综上, 除了ASCII码以外, 其他信息不能直接转换.

在python3的内存中. 在程序运行阶段. 使用的是unicode编码. 因为unicode是万国码. 什么内容都可以进行显示. 那么在数据传输和存储的时候由于unicode比较浪费空间和资源. 需要把unicode转存成UTF-8或者GBK进行存储. 怎么转换呢. 在python中可以把文字信息进行编码. 编码之后的内容就可以进行传输了. 编码之后的数据是bytes类型的数据.其实啊. 还是原来的数据只是经过编码之后表现形式发生了改变而已.

bytes的表现形式:

1. 英文 b'alex' 英文的表现形式和字符串没什么两样
2. 中文 b'\xe4\xb8\xad' 这是一个汉字的UTF-8的bytes表现形式

字符串在传输时转化成bytes=> encode(字符集)来完成

```
s = "alex"
print(s.encode("utf-8"))    # 将字符串编码成UTF-8
print(s.encode("GBK"))  # 将字符串编码成GBK
结果:
b'alex'
b'alex'


s = "中"
print(s.encode("UTF-8"))    # 中文编码成UTF-8
print(s.encode("GBK"))  # 中文编码成GBK

结果:
b'\xe4\xb8\xad'
b'\xd6\xd0'
```

记住: 英文编码之后的结果和源字符串一致. 中文编码之后的结果根据编码的不同. 编码结果也不同. 我们能看到. 一个中文的UTF-8编码是3个字节. 一个GBK的中文编码是2个字节. 编码之后的类型就是bytes类型. 在网络传输和存储的时候我们python是保存和存储的bytes

类型. 那么在对方接收的时候. 也是接收的bytes类型的数据. 我们可以使用decode()来进行解码操作. 把bytes类型的数据还原回我们熟悉的字符串:

```
s = "我叫李嘉诚"
print(s.encode("utf-8"))    #
b'\xe6\x88\x91\xe5\x8f\xab\xe6\x9d\x8e\xe5\x98\x89\xe8\xaf\x9a'

print(b'\xe6\x88\x91\xe5\x8f\xab\xe6\x9d\x8e\xe5\x98\x89\xe8\xaf\x9a'.decod
e("utf-8"))  # 解码
```

编码和解码的时候都需要制定编码格式.

```
s = "我是文字"
bs = s.encode("GBK")    # 我们这样可以获取到GBK的文字
# 把GBK转换成UTF-8
# 首先要把GBK转换成unicode. 也就是需要解码
s = bs.decode("GBK")  # 解码
# 然后需要进行重新编码成UTF-8
bss = s.encode("UTF-8") # 重新编码
print(bss)
```