# 《源码探秘 CPython》24. 字符串的intern机制

原创 古明地觉 古明地觉的编程教室 2022-02-08 09:30





## intern 机制

如果字符串的**interned**标识位为1，那么Python虚拟机将为其开启intern机制。那么什么是intern机制呢？

在Python中，某些字符串也可以像小整数对象池里的整数一样，共享给所有变量使用，从而通过避免重复创建来降低内存使用、减少性能开销，这便是intern机制。

Python的做法是在内部维护一个全局字典，所有开启intern机制的字符串均会保存在这里，后续如果需要使用的话，会先尝试在全局字典中获取，从而实现避免重复创建的功能。

```
void
PyUnicode_InternInPlace(PyObject **p)
{
    PyObject *s = *p;
    PyObject *t;
    //类型检测, 必须是PyUnicodeObject
    if (!PyUnicode_CheckExact(s))
        return;
    //检测interned标识位, 判断是否开启intern机制
    if (PyUnicode_CHECK_INTERNED(s))
        return;
    //创建字典, 用于存储开启intern机制的字符串
    if (interned == NULL) {
        interned = PyDict_New();
        if (interned == NULL) {
            PyErr_Clear(); /* Don't leave an exception */
            return;
        }
    }
    Py_ALLOW_RECURSION
    //下面的内容单独分析
    t = PyDict_SetDefault(interned, s, s);
    Py_END_ALLOW_RECURSION
    if (t == NULL) {
        PyErr_Clear();
        return;
    }
    if (t != s) {
        Py_INCREF(t);
        Py_SETREF(*p, t);
        return;
    }
    Py_REFCNT(s) -= 2;
    _PyUnicode_STATE(s).interned = SSTATE_INTERNED_MORTAL;
}
```

首先会进行一系列的检查，包括类型检查、因为intern共享机制只能用在字符串对象上，所以检查传入的对象是否已经被intern机制处理过了。

然后是**PyDict_SetDefault**，这个函数应该能猜出来是做什么的吧，就对应字典的

setdefault方法。当key不存在时，则新增键值对、并返回指定的value；key存在，则直接返回该key对应的value。

然后我们看到该函数在设置key、value的时候，传入的都是 s，这就意味着 s 指向的字符串对象的引用计数被增加了 2。这会造成该字符串的引用计数在Python程序结束前永远不会为0，这也是最后面的**Py_REFCNT(s) -= 2;**、也就是将计数减2的原因。

然后估计很多人都以为Python在创建一个字符串时，会首先检测该字符串是否已经存在，如果有，就不用创建新的，这样可以节省空间。但其实不是这样的，事实上节省内存空间是没错的，可Python并不是在创建PyUnicodeObject的时候就通过intern机制实现了节省空间的目的。

从PyUnicode_FromString中我们可以看到，无论如何一个合法的PyUnicodeObject总是会被创建的，而intern机制也只对PyUnicodeObject起作用。

对于任何一个字符串，Python总是会为它创建对应的PyUnicodeObject，尽管创建出来的对象在intern机制中已经存在了(有另外的PyUnicodeObject也维护了相同的字符串)。

而这正是关键所在，通常Python在运行时创建了一个PyUnicodeObject对象（假设叫temp）之后，基本上都会调用PyUnicode_InternInPlace对temp进行处理。如果维护的值已经有其他的PyUnicodeObject维护了，或者说其他的PyUnicodeObject对象维护了一个与之一模一样的字符数组（字典中已经存在），那么temp的引用计数就会减去1。

然后temp会由于引用计数为0而被销毁，只是昙花一现，然后归于湮灭。

> 所以现在我们就明白了intern机制，并不是说先判断是否存在，如果存在，就不创建。而是先创建，然后发现已经有其他的PyUnicodeObject维护了一个与之相同的字符数组，于是intern机制将引用计数减一，导致引用计数为0，最终被回收。

但是这么做的原因是什么呢？为什么非要创建一个PyUnicodeObject来完成intern操作呢？这是因为PyDictObject必须要求必须以**PyObject ***作为key。

---

然后关于PyUnicodeObject对象的intern机制，还有一点需要注意。实际上，被intern机制处理过后的字符串分为两类，一类处于**SSTATE_INTERNED_IMMORTAL**状态，另一类处于**SSTATE_INTERNED_MORTAL**状态，这两种状态的区别在unicode_dealloc中可以清晰的看到，SSTATE_INTERNED_IMMORTAL状态的PyUnicodeObject是永远不会被销毁的，它与Python解释器共存亡。

PyUnicode_InternInPlace只能创建SSTATE_INTERNED_MORTAL的PyUnicodeObject对象，如果想创建SSTATE_INTERNED_IMMORTAL对象，必须通过另外的接口来强制改变PyUnicodeObject的intern状态。

```
void
PyUnicode_InternImmortal(PyObject **p)
{
    PyUnicode_InternInPlace(p);
    if (PyUnicode_CHECK_INTERNED(*p) != SSTATE_INTERNED_IMMORTAL) {
        _PyUnicode_STATE(*p).interned = SSTATE_INTERNED_IMMORTAL;
        Py_INCREF(*p);
    }
}
```

**但是问题来了，什么样的字符才会开启intern机制呢?**

在Python3.8中，如果一个字符串的所有字符都位于0 ~ 127之间，那么会开启intern机制。

```
>>> a = "abc" * 1000
>>> b = "abc" * 1000
```

```
#之前的话是不超过20个字符
#但是在Python3.8中这个限制被扩大了很多
>>> a is b
True
>>>
>>> a = "abc" * 2000
>>> b = "abc" * 2000
#显然3 * 2000,6000个字符是不会开启intern机制的
#所以长度限制是多少，有兴趣可以自己试一下
>>> a is b
False
>>>
```

在Python3.8中，如果一个字符串只有一个字符，并且位于0~255之间，那么会开启intern机制。

```
>>> a = chr(255) * 2
>>> b = chr(255) * 2
#不位于0~127之间，所以不是ASCII字符
#因此没有开启intern机制
>>> a is b
False
>>>
>>> a = chr(255)
>>> b = chr(255)
# 但如果只有一个字符的话，则会开启
>>> a is b
True
>>> # 另外，空字符串也会开启
>>> a = ""
>>> b = ""
>>> a is b
True
```

实际上，存储单个字符这种方式有点类似于bytes对象中的缓存池。是的，正如整数有小整数对象池、bytes对象有字符缓存池一样，字符串也有其对应的PyUnicodeObject缓存池。

在Python的整数对象中，小整数的对象池是在Python初始化的时候被创建的，而字符串对象体系中的缓存池则是以静态变量的形式存在的。在Python初始化完成之后，缓存池的所有PyUnicodeObject指针都为空。

当创建一个PyUnicodeObject对象时，如果字符串只有一个字符，且位于0~255。那么会先对该字符串进行intern操作，再将intern的结果缓存到池子里。同样当再次创建PyUnicodeObject对象时，检测是不是只有一个字符，然后检查字符是不是存在于缓存池中，如果存在，直接返回。

**另外我们也可以通过sys.intern强制开启字符串的intern机制。**

## 小结

以上就是字符串相关的内容，正如之前所说，字符串没有想象中的那么简单。而在CPython里面，字符串的源码有一万六千多行，显然我们没办法一步一步地全部分析完，有兴趣的可以自己深入研究一下。

其实能把字符串的存储搞明白，已经是前进了一大步了。

收录于合集 #CPython 97

上一篇
《源码探秘 CPython》25. 列表是怎么实现的？解密列表的数据结构

下一篇
《源码探秘 CPython》23. 字符串的相关操作

文章已于2022-02-08修改

喜欢此内容的人还喜欢

python-字符串编码问题怎么破
一位代码

MySQL查询小工具（一）json格式的字符串字段中，替换json数组中对象的某个属性值
Seven的代码实验室


python 7天进阶之路-对象和json转换
缪斯之子