第四章：Python 对象

注意：

们现在来学习 Python 语言的核心部分。首先我们来了解什么是 Python 对象，然后讨论最常用的内建类型，接下来我们讨论标准类型运算符和内建函数，之后给出对标准类型的不同分类方式。这有助于我们更好的理解他们如何工作。最后我们提一提 Python 目前还不支持的类型（这对那些有其他高级语言经验的人会有所帮助）。

这里主要讨论的是数据类型的那个对象，而不是面对对象的知识。更确切的说是在讲数据类型。

目录

[4.1 Python 对象 1](#_Toc350440794)

[4.2 标准类型 2](#_Toc350440795)

[4.3 其他内建类型 3](#_Toc350440796)

[4.4 内部类型 5](#_Toc350440797)

[4.5 标准类型运算符 6](#_Toc350440798)

[4.6标准类型内建函数 7](#_Toc350440799)

[4.7 类型工厂函数 8](#_Toc350440800)

[4.8 标准类型的分类 8](#_Toc350440801)

[4.9 不支持的类型 9](#_Toc350440802)

## 4.1 Python 对象

Python 使用对象模型来存储数据。构造任何类型的值都是一个对象。

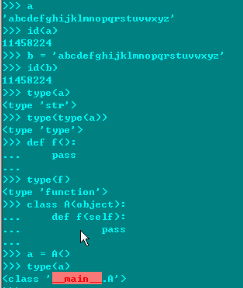
1、所有的 Python 对像都拥有三个特性：身份，类型和值。

python对象有三个特征: 身份和类型是只读的, 如果对象支持不支持更新操作, 那么值也就是只读的.

身份: 唯一的身份标识, 可以使用内建函数id()得到, 可以看作是对象的内存地址…

类型: 对象的类型决定该对象保存什么类型的值, 可以进行什么操作, 遵循什么规则., 可以使用内建函数type()查看python对象的类型.

值: 对象表示的数据项



2、对象属性

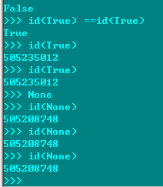
某些 Python 对象有属性、值或相关联的可执行代码，比如方法（method）。Python 用点（.）

标记法来访问属性。

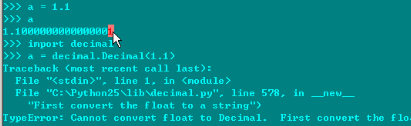
## 4.2 标准类型

标准数据类型: 数字, 整形, 布尔型, 长整型, 浮点型, 复数型, 字符串, 列表, 元组, 字典.

布尔的True、false都是唯一的。就是我们这个东西从解释器启动开始始终就一个对象。

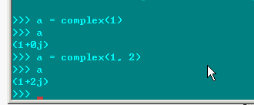


关于精度的问题：



Decimal的参数是字符串类型的。所以报错。

复数类型：



字符串, 列表, 元组, 字典.：他们都是序列对象，即他们都是一个有顺序的，就相当于集合，但是不是数学上的集合，比如支持切片遍历等等。

字符串：str，接受一个对象转化成为string。

列表：list（）是他的工厂函数。（注意：这些函数的理解直接help以及dir，自己查）。Python中没有数组，但是列表的功能很强大的。列表是可变的，有顺序的，是集合类型的，元组是不可变的。

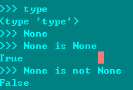
为什么提供元组呢？处于安全性上的考虑，动态语言太灵活了，那与控制，所以为=一些东西为了不让动。

字典：作为键是有有要求的，可被哈希化的才行。到后边讲解。

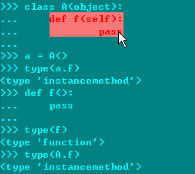
## 4.3 其他内建类型

其他内建类型: 类型（type）, None（唯一的内建对象）, 文件（file）, 集合/固定集合（set）, 函数/方法（function/ instancemethod）, 模块, 类。

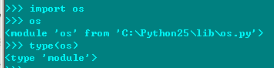
None（唯一的内建对象）



函数/方法（function/instancemethod）



模块



类

类和类型怎么理解呢：到后面再看。类型即是type。类就是你定义的一个类呗。

这些是当你做 Python 开发时可能会用到的一些数据类型。我们在这里讨论 Type 和 None

类型的使用，除此之外的其他类型将在其他章节中讨论。

2对象和 type 类型对象

在本章我们要讨论所有的 Python的数据类型，虽然看上去把类型本身也当成对象有点特别，我们还是要在这里提一提。你一定还记得，对象的一系列固有行为和特性（比如支持哪些运算，具有哪些方法）必须事先定义好。从这个角度看，类型正是保存这些信息的最佳位置。描述一种类型所需要的信息不可能用一个字符串来搞定，所以类型不能是一个简单的字符串，这些信息不能也不应该和数据保存在一起， 所以我们将类型定义成对象。

2、下面我们来正式介绍内建函数 type()。通过调用 type()函数你能够得到特定对象的类型

信息：

>>> type(42)

<type 'int'>

我们仔细研究一下这个例子，请注意看 type 函数有趣的返回值。我们得到一个简洁的输出结果<type 'int'>。不过你应当意识到它并不是一个简简单单的告诉你 42 是个整数这样的字符串。您看到的<type 'int'>实际上是一个类型对象，碰巧它输出了一个字符串来告诉你它是个 int 型对象。

现在你该问自己了，那么类型对象的类型是什么？来， 我们试验一下：

>>> type(type(42))

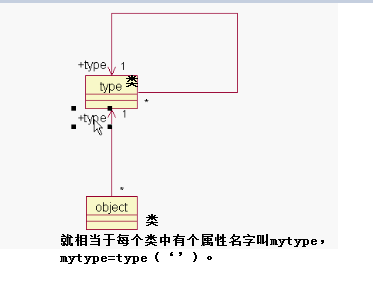
<type 'type'>

没错，所有类型对象的类型都是 type，它也是所有 Python 类型的根和所有 Python 标准类的默认元类（metaclass）。你现在有点搞不明白，没关系，随着我们逐步深入的学习类和类型，你就会慢慢理解。

随着 Python 2.2 中类型和类的统一，类型对象在面向对象编程和日常对象使用中扮演着

更加重要的角色。从现在起， 类就是类型，实例是对应类型的对象。

type(type(1)) 可以看出类型对象本身也是对象, 它的类型是type。可以如下理解，其实不是的。先这样帮助你理解类和类型的概念。



3、None的类型是NoneType, NoneType只有一个值, 就是None, 不支持任何运算也没有任何内建方法, 布尔值总返回False

每个对象天生都有布尔值, 以下对象的布尔值是False

a)None

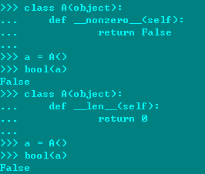
b)False(bool类型)

c)所有值为0的数

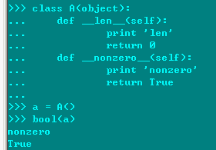
d)‘’ 空字符串

e)[] | () | {} 空列表, 空元组, 空字典

用户创建的实例如果定义了nonzero(\_\_nonzero\_\_())或length(\_\_len\_\_())且值为0, 那么返回的布尔值也是False



调用顺序：先调用的是nonzero(\_\_nonzero\_\_())这个方法



当要获取一个对象的布尔值的时候, 首先会调用\_\_nonzero\_\_(必须返回布尔类型或者int类型.)方法, 如果实现了该方法, 就返回该方法返回的布尔值, 如果没有实现该方法, 继续调用\_\_len\_\_方法(该方法返回值必须是大于等于0的), 如果实现了\_\_len\_\_方法, 就根据其返回值返回布尔值.

3、问题：如何理解python编程中类.类型.对象的实例等等

回答：

类就是一类事物的概括，比如人。

类型是数据类型，有内置的如字符串，数字，复数等，还有其他自定义的，就是自己写的类。

对象和实例是一个东西，是类里具体的东西，如男人，女人，其他人等，这里男人，女人还可以是一类，如再分年老的男人，年轻的男人等。

只要记住：类是一类事物的统称，实例（或对象）是具体的东西。

仅供参考。

举例：

class Person:

'''人的基本属性：名字，年龄，性别等'''

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

self.name = name

self.age = age

self.sex = sex

class Man(Person):

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

super(Man, self).\_\_init\_\_(name, age, 'male')

class Woman(Person):

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

super(Woman, self).\_\_init\_\_(name, age, 'female')

## 4.4 内部类型

z 代码

编译过的python的源代码片段, 是可执行对象. 通过调用内建函数compile()可以得到代码对象. 代码对象可以被exec命令或eval()内建函数来执行. 代码是用户自定义函数的核心, 运行期获得上下文.. \_\_code\_\_是函数的一个属性, 函数除了代码这个属性, 还有函数名, 文档字符串, 默认参数, 全局命名空间等必须的属性

z 帧

帧对象: 用于跟踪记录对象

z 跟踪记录

跟踪记录: 用于异常发生时, 程序访问跟踪记录对象处理程序.

z 切片

切片: 序列类型的对象可以使用切片操作。



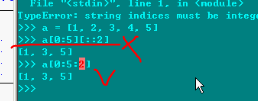
从第0个到第6个，第六个不取。



负值是从后向前数的



前面的默认是0，后面的默认是他自己的长度



第三个是步长

切片sequence[::number] number为负数, 倒序显示字符串, 正数则正序显示字符串, 数值代表显示字符的step值.

多维切片 sequence[start1: end1, start2: end2]

省略切片 sequence[…, start1: end1]

切片对象 使用slice([start, ]stop[, step])可以创建一个切片对象

z 省略

省略对象: 用于扩展切片语法, 起记号作用..在切片语法中表示省略号, 省略对象有一个唯一的名字Ellipsis, 布尔值始终是True

z Xrange

Xrange: 调用xrange()生成Xrange对象, 类似内建函数range, 用于节省内存使用或range无法完成的超大数据集场合

我们一直用的range().xrange（）做了优化的，占用的内存更小了。性能上有更多的优势。

## 4.5 标准类型运算符

1、对象值比较

数字根据大小比较

字符串根据字符先后顺序比较

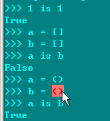
list | tuple | dict顺序按照其中元素(dict按照键)比较

链式比较相当于多个比较使用and连接

自定义类型对象的比较是引用值比较, 也就是id(object\_name)的比较

2、对象身份比较

is / is not用来比较两个别名是否引用同一个对象



因为元组是不可变对象，值相同的时候，在内存中只有一个。列表是可变对象。

整数对象和字符串对象是不可变对象…



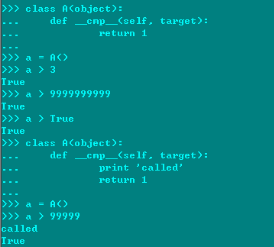
3、布尔类型运算符: and, or, not

## 4.6标准类型内建函数

除了这些运算符， 我们刚才也看到， Python 提供了一些内建函数用于这些基本对象类型：

cmp(), repr(), str(), type(), 和等同于 repr()函数的单反引号(``) 运算符。

1、cmp(obj1, obj2) 比较obj1和obj2, 根据比较结果返回整数i, i<0 🡪 obj1<obj2, i>0 🡪 obj1>obj2, i==0 🡪 obj1 == obj2….自定义类型中通过定义方法\_\_cmp\_\_(target)来实现….使用比较运算符, 或直接调用cmp时该方法被调用



说明：Return 1（1就是true）就是我始终都比你大，谁跟我比，我比谁大。

我是一个自定义的类型，但我没实现那个方法\_\_cmp\_\_（），怎么比较，比较的是id值。

2、type(obj) 得到一个对象的类型, 返回相应的type对象

有什么用呢？

3、str(obj) 返回对象适合可读性好的字符串表示…在自定义类中使用\_\_str\_\_(self)返回一个字符串, 调用str(obj)时会被隐式调用

4、repr(obj) 返回一个对象的字符串表示, repr返回的字符串通常可以被用于使用eval动态创建一个对象. 通常 obj == eval(repr(obj))是成立的

1. ``, 反单引号, `obj`和repr(obj)做的事情是一样的.
2. isinstance(obj, (type[, type, …])) 判断obj是不是第二个参数tuple中的列举的类型的实例

7、types模块提供一些已知的类型

* 1. 由于每一种类型都只有一个类型对象, 所以, 可以使用引用比较代替值比较以提升性能: 用if type(num) is types.IntType 替代if type(num) == types.IntType
  2. from-import: 只引入某个模块的一部分属性: 比如from types import IntType 这样做可以有效的减少查询次数
  3. python的operator模块中有绝大多数运算符对应的同功能的函数可供使用.

8、类型工厂函数: int(), long(), float(), complex(), str(), Unicode(), basestring(), list(), tuple(), type(), dict(), bool(), set(), frozenset(), object(), classmethod(), staticmethod(), super(), property(), file()

## 4.7 类型工厂函数

Python 2.2 统一了类型和类， 所有的内建类型现在也都是类， 在这基础之上， 原来的

所谓内建转换函数象 int(), type(), list() 等等， 现在都成了工厂函数。 也就是说虽然他

们看上去有点象函数， 实质上他们是类。当你调用它们时， 实际上是生成了该类型的一个实例，就象工厂生产货物一样。

下面这些大家熟悉的工厂函数在老的 Python 版里被称为内建函数：

z int(), long(), float(), complex()

z str(), unicode(), basestring()

z list(), tuple()

z type()

以前没有工厂函数的其他类型，现在也都有了工厂函数。除此之外，那些支持新风格的类

的全新的数据类型，也添加了相应的工厂函数。下面列出了这些工厂函数：

z dict()

z bool()

z set(), frozenset()

z object()

z classmethod()

z staticmethod()

z super()

z property()

z file()

## 4.8 标准类型的分类

1、标准类型是”基本内建数据对象原始类型”

基本: 是python的标准或核心

内建: python默认提供

数据: 用于一般数据存储

对象: 对象是数据和功能的默认抽象

原始: 这些类型提供的是最底层的粒度数据存储

类型: 本身就是数据类型

2、们对类型进行分类的三种方式。

1）按存储模型进行分类:

标量/原子类型: 数值, 字符串等可以存储单一字面对象的类型.

容器类型: 列表, 元素, 字典等可以存储多个字面变量的类型, python中的容器类型都可以存储不同类型的元素

2）按更新模型进行分类: 对象创建之后, 值不可以改变, 注意: 这里是对象, 而不是变量

可变类型: 列表, 字典

不可变类型: 数字, 字符串, 元组

3）按访问模型进行分类: 访问对象的方式

直接访问: 数值

顺序访问: 列表, 元组, 字符串等可以按照索引访问的类型

映射访问: 字典

我们提供了一个汇总表（表 4.9）。表中列出了所有的标准类型， 我们使用的三个模型，

以及每种类型归入的分类。

表 4.9 标准类型分类

数据类型 存储模型 更新模型 访问模型

数字 Scalar 不可更改 直接访问

字符串 Scalar 不可更改 顺序访问

列表 Container 可更改 顺序访问

元组 Container 不可更改 顺序访问

字典 Container 可更改 映射访问

## 4.9 不支持的类型

不支持的类型: char, byte, pointer。

在我们深入了解各个标准类型之前，我们在本章的结束列出 Python 目前还不支持的数据类

型。

char 或 byte Python

没有 char 或 byte 类型来保存单一字符或 8 比特整数。你可以使用长度为 1 的字符串表示字符或 8 比特整数。

指针

Python 替你管理内存，因此没有必要访问指针。在 Python 中你可以使用 id()函数得到一

个对象的身份号， 这是最接近于指针的地址。因为你不能控制这个值，所以其实没有太大意义。实在 Python 中， 一切都是指针。

int vs short vs long

Python 的普通整数相当于标准整数类型，不需要类似 C 语言中的 int, short, long 这三种整数类型。事实上 Python 的整数实现等同于 C 语言的长整数。 由于 Python 的整型与长整型密切融合， 用户几乎不需要担心什么。 你仅需要使用一种类型， 就是 Python 的整型。即便数值超出整型的表达范围， 比如两个很大的数相乘， Python 会自动的返回一个长整数给你而不会报错。

float VS double

C 语言有单精度和双精度两种浮点类型。 Python 的浮点类型实际上是 C 语言的双精度浮

点类型。 Python 认为同时支持两种浮点类型的好处与支持两种浮点类型带来的开销不成比例，所以 Python 决定不支持单精度浮点数。对那些宁愿放弃更大的取值范围而需要更高精确度的用户来说， Python 还有一种十进制浮点数类型 Decimal， 不过你必须导入 decimal 模块才可以使用它。浮点数总是不精确的。Decimals 则拥有任意的精度。在处理金钱这类确定的值时，Decimal 类型就很有用。 在处理重量，长度或其它度量单位的场合， float 足够用了。