



2 years ago



## 1/10

```
# -*- coding:utf-8 -*-  
  
a = 0o101  
print("a="+str(a))  
  
b=64  
print('b='+str(b))  
c=-237  
print('c='+str(c))  
d=0x80  
print('d='+str(d))  
e=-0x92  
print('e='+str(e))
```

上面的代码输出结果为：

问题	输出	调试控制台	终端
	a=65		
	b=64		
	c=-237		
	d=128		
	e=-146		

长整型是整型的超集，可以表示无限大的整数（实际上只受限于机器的虚拟内存大小）。长整型和标准整型，目前已经基本统一，当数学运算遇到整型异常的情况，在Python2.2以后的版本，会自动转换为长整型。继续添加测试代码：

问题	输出	调试控制台	终端
	a=65		
	b=64		
	c=-237		
	d=128		
	e=-146		
	longint=99920027994400699944002799920001		

### 1.2.2 布尔型和布尔对象

每一个Python对象都天生具有布尔值（True或False），进而可用于布尔测试（如用在if、while中）。

- None
- False (布尔型)
- 0 (整型0)
- 0L (长整型0)
- 0.0 (浮点型0)
- 0.0+0.0j (复数0)
- "" (空字符串)
- [] (空列表)
- () (空元组)
- {} (空字典)
- 用户自定义的 类实例, 该类定义了方法 **nonzero()** 或 **len()**, 并且这些方法返回0或False

```
#基本测试
print(bool(1))
print(bool(True))
print(bool('0'))
print(bool([]))
print(bool((1,)))
```

3/10



```
问题  输出  调试控制台  终端

e=-146
longint=99920027994400699944002799920001
True
True
True
False
True
>
```

下面我们看看bool类型作为只有0和1取值的特殊整型的特性。

```
#使用bool数
foo = 42
bar = foo<42

print(bar)
print(bar+10)
print('%s' %bar)
print('%d' %bar)
```

运行结果如下：

```
False
10
False
0
```

再来验证下没有\_nonzero\_()方法的对象，默认是True。

```
#无_nozero_() class C:pass
```

```
c=C() print(bool(c))
```

运行结果如下：



```
问题  输出  调试控制台  终端

False
True
False
10
False
0
无_nozero_():True
```

### 1.2.3 双精度浮点型

Python里的浮点型数字都是双精度，类似C语言的double类型。可以用十进制或者科学计数法表示。下面我们看一些典型的浮点型数字。

添加测试代码：

```
# 双精度浮点
print(0.0)
print(-777.)
print(-5.555567119)
print(96e3 * 1.0)
print(-1.609E-19)
```

运行结果如下：



```
问题  输出  调试控制台  终端

0
无_nozero_():True
0.0
-777.0
-5.555567119
96000.0
-1.609e-19
>
```

### 1.2.4 复数

在Python中，有关复数的概念如下：

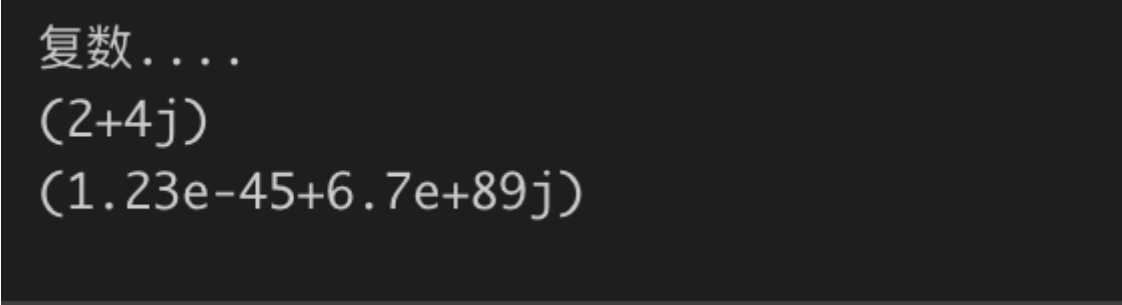
- 虚数不能单独存在，它们总是和一个值为0.0的实数部分一起来构成一个复数。
- 复数由实数部分和虚数部分组成。
- 表示虚数的语法：real+imagj。
- 实数部分和虚数部分都是浮点型。
- 虚数部分必须有后缀j或J。

复数可以用使用函数 `complex(real, imag)` 或者是带有后缀j的浮点数来指定。

下面添加代码测试复数：

```
print(complex(2, 4))
print(1.23e-045+6.7e+089j)
```

运行结果如下：



```
复数....
(2+4j)
(1.23e-45+6.7e+89j)
```

### 1.2.5 十进制浮点型

十进制浮点通常称为decimal类型，主要应用于金融计算。双精度浮点型使用的是底和指数的表示方法，在小数表示上精度有限，会导致计算不准确，decimal采用十进制表示方法，看上去可以表示任意精度。

下面我们看一下十进制浮点的例子。

```
# 十进制浮点
# 十进制浮点
from decimal import *

print("十进制浮点....")
dec=Decimal('.1')
print(dec)
print(Decimal(.1))
print(dec +Decimal(.1))
```

使用decimal类型，首先要引入decimal模块，然后通过Decimal类来初始化一个Decimal对象。

运行结果如下：

```
十进制浮点....  
0.1  
0.10000000000000000055511151231257827021181583404541015625  
0.20000000000000000055511151231
```

## 1.2.6 操作符

在Python中同时支持不同数值类型的数字进行混合运算，数字类型不一致怎么做运算？这个时候就涉及到强制类型转换问题。这种操作不是随意进行的，它遵循以下基本规则：

首先，如果两个操作数都是同一种数据类型，没有必要进行类型转换。仅当两个操作数类型不一致时，Python才会去检查一个操作数是否可以转换为另一类型的操作数。如果可以，转换它并返回转换结果。

由于某些转换是不可能的，比如果将一个复数转换为非复数类型，将一个浮点数转换为整数等等，因此转换过程必须遵守几个规则。要将一个整数转换为浮点数，只要在整数后面加个.0就可以了。要将一个非复数转换为复数，则只需要要加上一个“0j”的虚数部分。

这些类型转换的基本原则是：整数转换为浮点数，非复数转换为复数。在 Python 语言参考中这样描述`coerce()`方法：

如果有一个操作数是复数，另一个操作数被转换为复数。

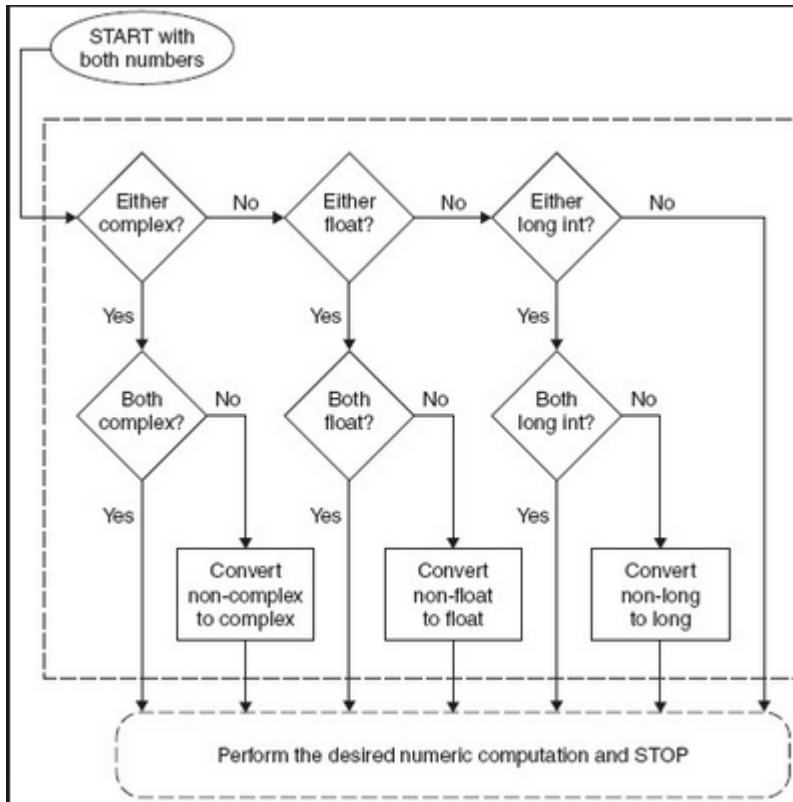
否则，如果有一个操作数是浮点数，另一个操作数被转换为浮点数。

否则，如果有一个操作数是长整数，则另一个操作数被转换为长整数；

否则，两者必然都是普通整数，无须类型转换。

数字类型之间的转换是自动进行的，程序员无须自己编码处理类型转换。Python 提供了 `coerce()` 内建函数来帮助你实现这种转换。

转换流程图如下图所示：



### 1.2.7 转换工厂

函数 `int()`, `long()`, `float()` 和 `complex()` 用来将其它数值类型转换为相应的数值类型。从 Python2.3 开始, Python 的标准数据类型添加了一个新成员: 布尔 (Boolean) 类型。从此 `true` 和 `false` 现在有了常量值即 `True` 和 `False` (不再是 1 和 0)。

下面继续添加代码进行测试。

```
print("转换工厂.....")
print(int(4.222222))
print(float(4))
print(complex(4))
```

结果如下:

```
转换工厂.....
4
4.0
(4+0j)
```

### 1.2.8 进制转换



添加测试代码：

运行结果如下：

### 1.2.9 ASII 转换

添加测试代码：

运行结果如下：

### 1.2.10

9/10

本节留给大家的练习题目也很简单：

1. 将文章中所有代码手动敲打一遍
2. 扩展阅读,请自行查阅资料了解Python常用的数学函数：

- `ceil(x)`
- `floor(x)`
- `fabs(x)`
- `factorial (x)`
- `hypot(x,y)`
- `sqrt(xx+yy)`
- `pow(x,y)`
- `sqrt(x)`
- `log(x)`
- `log10(x)`
- `trunc(x)`
- `isnan (x)`
- `degree (x)`
- `radians(x)`

下一节，我们继续学习Python中常用的几种数据结构。

欢迎到关注微信订阅号，交流学习中的问题和心得

