第三章 Python变量和数据类型

一、Python中的数据类型

计算机顾名思义就是可以做数学计算的机器，因此计算机程序理算当然地可以处理各种数值。但是，计算机能处理的不止数值，还可以处理文本，图形，音频，视频等，不同的数据需要定义不同的数据类型。在Python中，能够直接处理的数据类型有以下几种：

1.整数

1）Python可以处理任意大小的整数，也包括负整数；

2）表示方法：和数学上的写法一模一样，例如：1,，100，-8080，0等等；

3）计算机由于是二进制，所以，有时候用十六进制表示整数比较方便，十六进制用0x前缀和0 9，a～f表示，例如：0xff00，0xa5b4c3d2等等

2.浮点数

1）浮点数也就是小数，之所以称为浮点数，是因为按照科学记数法表示时，一个浮点数的小数点位置是可以变的，比如，1.23×10^9和12.3×10^8是相等的。

2）数学写法：1.23,3.14,-9.01等等

3）对于很大或者很小的浮点数要用科学记数法表示：把10用e代替

1.23×10^9：1.23e9或者12.3e8；

0.000012可以写成：1.2e-5

4）整数和浮点数在计算机内部存储的方式是不同的，整数运算永远是精确到的（包括除法），而浮点数运算则可能是有四舍五入的误差。

3.字符串

1）字符串是以”和””括起来的任意文本，比如’abc’,”xyz”等等。

2）注意：”和””本身只是一种表示方式，不是字符串的一部分，因此，字符串’abc’只有a，b，c这3个字符。

4.布尔值

1）布尔值和布尔代数的表示完全一致，一个布尔值只有True、False两种值，要么是True，要么是False，在Python中，可以直接用True、False表示布尔值（请注意大小写），也可以通过布尔运算计算出来。

2）布尔值可以用and、or、not运算；

and运算：与运算，只要所有都为True，and运算结果才是True；

or运算：或运算，只要其中有一个为True，or运算结果就是True；

not运算：非运算，它是一个单目运算符，把True变成False，False变成True。

5.空值

1）空值是Python里一个特殊的值，用None表示；

2）None不能理解为0，因为0是有意义的，而None是一个特殊的空值。

3）此外，Python还提供了列表、字典等多种数据类型，还允许创建自定义数据类型，后面会写到。

练习1：计算十进制整数 45678 和十六进制整数 0x12fd2 之和。

```
print  45678 + 0x12fd2
```

练习2：请用字符串表示出Learn Python in imooc。

```
print "Leran Python in imooc"
```

练习3：请计算以下表达式的布尔值（注意==表示判断是否相等）：100<99;0xff==255;

```
print "Leran Python in imooc"
```

二、Python之print语句

1.print语句可以向屏幕上输出指定的文字。比如输出'hello,world'，用代码实现如下：

```
print 'hello,world'
```

注意：

1）当我们在Python交互式环境下编写代码时，¿¿¿是Python解释器的提示符，不是代码的一部分；

2）当我们在文本编辑器中编写代码时，千万不要自己添加¿¿¿。

2.print语句也可以跟上多个字符串，用逗号","隔开，就可以连成一串输出：

```
print 'The quick brown fox','jumps over','the lazy dog'
```

3.print语句会依次打印每个字符串，遇到逗号","会输出一个空格，因此，输出的字符串是这样的：

```
The quick brown fox jumps over the lazy dog
```

4.print也可以打印整数，或者计算结果：

```
print 300
300  #运行结果
print 100+200
300  #运行结果
```

因此，我们可以把计算100+200，Python解释器自动计算出结果300，但是，'100+200='是字符串而非数学公式，Python把它视为字符串，打印如下：

```
print '100 + 200 =',100+200
100 + 200 = 300  #运行结果
```

5.练习：请用两种方式打印出 hello, python：

```
print 'hello, python'
print 'hello,','python'
hello, python  #运行结果
```

三、Python的注释

任何时候，我们都可以给程序加上注释。注释是用来说明代码的，给自己看也给别人看，而程序运行的时候，Python解释器会直接忽略掉注释，所以，有没有注释不影响程序的执行结果，但是影响到别人能不能看懂你的代码。

1.Python的注释以#开头，后面的文字直接到行尾都算注释

```
# 这一行全部都是注释
print 'hello' #这也是注释
```

2.注释还有一个巧妙的用途，就是一些代码我们不想运行，但是有不想删除，就可以用注释暂时屏蔽掉：

```
# 暂时不想运行下面一行代码
# print 'hello, pyhton'
```

四、Python中什么是变量

1.在Python中，变量的概念基本上和初中代数的方程变量是一致的。

2.例如，对于方程式y=x*x，x就是变量。当x=2时，计算结果就是4，当x=5时，计算结果是25。

3.只是在计算机程序中，变量不仅可以是数字，还可以是任意数据类型。

4.在Python程序中，变量是用一个变量名表示，变量名必须大小写英文、数字和下划线的组合，且不能以数字开头，比如：

```
a=1;
#变量a是一个整数
t_007='T007'
#变量t_007是一个字符串
```

5.在Python中，等号=是赋值语句，可以把任意数据类型赋值给变量，同一个变量可以反复赋值，而且可以是不同类型的变量，例如：

```
a=123;  #a是整数
print a
a='imooc' #a变为字符串
print a
```

这种变量本身类型不固定的语言称之为动态语句，与之对应的是静态语句。

6.静态语言在定义变量时必须指定变量类型，如果赋值的时候类型不匹配，就会报错。例如：Java是静态语言，赋值语句如下：

```
int a=123;  //a是整数类型变量
a="imooc";  //错误：不能把字符串赋给整型变量
```

和静态语言相比，动态语言更灵活，就是这个原因。

7.请不要把赋值语句的等号等同于数学的等号。比如如下的代码：

```
x=10
x=x+2
```

如果从数学上理解x=x+2，是不成立的，在程序中，赋值语句先计算右侧的表达式想x+2，得到12，再赋给变量x。由于x之前的值是10，重新赋值后，x的值变为12。

8.最后，理解变量在计算机内存中的表示也非常重要。当我们写：a='ABC'时，Python解释器干了两件事情：

（1）在内存中创建了一个'ABC'的字符串；

（2）在内存中创建了一个名为a的变量，并把它指向'ABC'。

9.也可以把一个变量a赋值给另一个变量b，这个操作实际上是把变量b指向变量a所指向的数据，例如下面的代码：

```
a='ABC'
b=a
a='XYZ'
print b
```

10.练习：等差数列可以定义为每一项与它的前一项的差等于一个常数，可以用变量 x1 表示等差数列的第一项，用 d 表示公差，请计算数列1 4 7 10 13 16 19 ...前100项的和。

```
x1=1
d=3
n=100
x100=x1+(n-1)*d     #第100项
s=(x1+x100)*100/2   #运用前n项和的公式
print s
```

五、Python中定义字符串

前面我们讲了什么是字符串，字符串可以用”或者””括起来表示。

1.字符串本身包含’：可以用””括起来表示

```
"I'm OK"
```

2.字符串本身包含”：可以用”括起来表示

```
'Learn "Python" in immoc'
```

3.字符串本身既包含’又包含”：使用转义字符

```
'Bob said \"I\'m OK\".'
#'Bob said "I'm OK".'
# \ 是一个普通字符，不代表字符串的起始地址
```

注意：转义字符 不计入字符串的内容中。

常用的转义字符还有：

```
# \n 表示换行
# \t 表示一个制表符
# \\ 表示\字符本身
```

4.练习：请将下面两行内容用Python的字符串表示并打印出来： Python was started in 1989 by ”Guido”.Python is free and easy to learn.

```
s1 = 'Python was started in 1989 by "Guido".'
print s1
s2='Python is free and easy to learn.'
print s2
```

六、Python中raw字符串与多行字符串

如果字符串中有很多转义字符，对每一个字符都进行转义比较麻烦，为了避免这种情况，我们可以在字符串前面加个前缀r，表示这是个raw字符串，里面的字符就不需要转义了，例如：

```
r'\(~_~)/ \(~_~)/'
```

但是r'...'表示法不能表示多行字符串，也不能表示包含'和 "的字符串（为什么？）

如果要表示多行字符串，可以用"'..."'表示：

```
'''Line 1
Line 2
Line 3'''
```

上面这个字符串的表示方法和下面的是完全一样的：

```
'Line 1\Line 2\Line 3'
```

还可以在多行字符串前面添加r，把这个多行字符串也变成一个raw字符串：

```
r'''Python is created by "Guido".
It is free and easy to learn.
Let's start learn Python in imooc!'''
```

练习：请把下面的字符串用r"'..."'的形式改写，并用print打印出来：

```
'\"To be, or not to be\": that is the question.\nWhether it\'s nobler in the mind to suffer.
```

```
print r'''"To be, or not to be":that is the question.
Whether it's nobler in the mind to suffer.'''
```

七、Python中的Unicode字符串

因为计算机只能处理数字，如果要处理文本，就必须先把文本转换为数字才能处理。最早的计算机在设计时采用8个比特（bit）作为一个字节（byte），所以，一个字节能表示的最大的整数就是255（二进制11111111=十进制255），0 - 255被用来表示大小写英文字母、数字和一些符号，这个编码表被称为ASCII编码，比如大写字母 A 的编码是65，小写字母 z 的编码是122。

如果要表示中文，显然一个字节是不够的，至少需要两个字节，而且还不能和ASCII编码冲突，所以，中国制定了GB2312编码，用来把中文编进去。

类似的，日文和韩文等其他语言也有这个问题。为了统一所有文字的编码，Unicode应运而生。Unicode把所有语言都统一到一套编码里，这样就不会再有乱码问题了。

Unicode通常用两个字节表示一个字符，原有的英文编码从单字节变成双字节，只需要把高字节全部填为0就可以。

因为Python的诞生比Unicode标准发布的时间还要早，所以最早的Python只支持ASCII编码，普通的字符串'ABC'在Python内部都是ASCII编码的。

Python在后来添加了对Unicode的支持，以Unicode表示的字符串用u'...'表示，比如：

```
print u'中文'
中文
```

注意：不加u，中文就不能正常显示

Unicode字符串除了多了一个 u 之外，与普通字符串没啥区别，转义字符和多行表示法仍然有效：

转义：

```
u'中文\n日文\n韩文'
```

多行：

```
u'''第一行
第二行'''
```

raw+多行：

```
ur'''Python的Unicode字符串支持"中文",
"日文",
"韩文"等多种语言'''
```

如果中文字符串在Python环境下遇到 UnicodeDecodeError，这是因为.py文件保存的格式有问题。可以在第一行添加注释

```
# -*- coding: utf-8 -*-
```

目的是告诉Python解释器，用UTF-8编码读取源代码。然后用Notepad++另存为... 并选择UTF-8格式保存。

练习：用多行Unicode字符串表示下面的唐诗并打印：

```
# -*- coding: utf-8 -*-
print '''静夜思
床前明月光，
疑似地上霜。
举头望明月，
低头思故乡。'''
```

八、Python中整点数和浮点数

Python支持对整数和浮点数直接进行四则混合运算，运算规则和数学上的四则运算规则完全一致。

基本的运算

```
1 + 2 + 3  #==>6
4 * 5 - 6  #==>14
7.5 / 8 + 2.1  #==>3.0375
```

使用括号可以提升优先级，这和数学运算完全一致，注意只能使用小括号，但是括号可以嵌套很多层：

```
(1 + 2) * 3    # ==> 9
(2.2 + 3.3) / (1.5 * (9 - 0.3))    #    ==>0.421455938697318O7
```

和数学运算不同的地方是，Python的整数运算结果仍然是整数，浮点数运算结果仍然是浮点数：

```
1 + 2    # ==> 整数 3
1.0 + 2.0    # ==> 浮点数 3.0
```

但是整数和浮点数混合运算的结果就变成浮点数了：

```
1 + 2.0    # ==> 浮点数 3.0
```

区分整数运算和浮点数运算的原因：整数运算的结果永远是精确的，而浮点数运算的结果不一定精确，因为计算机内存再大，也无法精确表示出无限循环小数，比如 0.1 换成二进制表示就是无限循环小数。

整数除不尽的时候：

```
11 / 4    # ==> 2
```

Python的整数除法，即使除不尽，结果仍然是整数，余数直接被扔掉。不过，Python提供了一个求余的运算

```
11 % 4    # ==> 3
```

计算 11 / 4 的精确结果：按照“整数和浮点数混合运算的结果是浮点数”的法则，把两个数中的一个变成浮点数再运算就没问题了。

```
11.0 / 4    # ==> 2.75
```

练习：请计算 2.5 + 10 / 4 ,并解释计算结果为什么不是期望的 5.0 ?修复上述运算，使得计算结果为5.0

```
#print 2.5 + 10 / 4
print 2.5 + 10 / 4.0
```

九、Python中的布尔类型

Python支持布尔类型的数据，布尔类型只有True和False两种值；

布尔类型有以下几种运算：

（1）与运算and：只有两个布尔值都为True时，计算结果才为True：

```
True and True    # ==> True
True and False    # ==> False
False and True    # ==> False
False and False    # ==> False
```

（2）或运算or：只要有一个布尔值为True，计算结果就是True：

```
True or True    # ==> True
True or False    # ==> True
False or True    # ==> True
False or False    # ==> False
```

（3）非运算not：把True变为False，或者把False变为True：

```
not True    # ==> False
not False    # ==> True
```

布尔运算在计算机中用来做条件判断，根据计算结果为True或者False，计算机可以自动执行不同的后续代码。

在Python中，布尔类型还可以与其他数据类型做 and、or和not运算，请看下面的代码：

```
a = True
print a and 'a=T' or 'a=F'
```

计算结果不是布尔类型，而是字符串 'a=T'，这是为什么呢?

因为Python把0，空字符串""和None看成False，其他数值和非空字符串都看成True，所以：

```
True and 'a=T' #计算结果是 'a=T'
#继续计算 'a=T' or 'a=F' 计算结果还是 'a=T'
```

短路计算：涉及到 and 和 or 运算的一条重要法则

（1）在计算 a and b 时，如果 a 是 False，则根据与运算法则，整个结果必定为 False，因此返回 a；如果 a 是 True，则整个计算结果必定取决与 b，因此返回 b。（2）在计算 a or b 时，如果 a 是 True，则根据或运算法则，整个计算结果必定为 True，因此返回 a；如果 a 是 False，则整个计算结果必定取决于 b，因此返回 b。

Python解释器在做布尔运算时，只要能提前确定计算结果，它就不会往后算了，直接返回结果。

练习：

```
# -*- coding: utf-8 -*-
a = 'python'
print 'hello,', a or 'world'
b = ''
print 'hello,', b or 'world'
#因为Python把0、空字符串''和None看成False，其他数值和非空字符串都看成True，而且Python解释器
#由于 a = "python" 中 a 不为空值，所以 a or "world" ==> True or True，直接返回第一个 true，
#然而 b = ""刚好相反，b为空值，所以 b or "world" ==> False or True, 返回的是第二True。即返
```