|  |
| --- |
| 变量和字符编码 |
| **作者：李昂**  **归档：python之路day1**  **日期：2018/2/5** |
| **快捷键：**  Ctrl + 1 标题1  Ctrl + 2 标题2  Ctrl + 3 标题3  Ctrl + 4 实例  Ctrl + 5 程序代码  Ctrl + 6 正文  Ctrl + q 正文注释 |
| **格式说明：**  蓝色字体：注释  黄色背景：了解  绿色背景：重要  红色背景：特别重要 |

目 录

[变量和字符编码 I](#_Toc503194172)

[第1章 变量定义规则 1-3](#_Toc503194173)

[第2章 变量类型 2-4](#_Toc503194174)

[2.1 整数 2-4](#_Toc503194175)

[2.2 浮点数 2-4](#_Toc503194176)

[2.3 字符串 2-4](#_Toc503194177)

[2.4 转义字符 2-4](#_Toc503194178)

[2.4.1 打印多行(’’’ ’’’) 2-5](#_Toc503194179)

[2.5 布尔值 2-5](#_Toc503194180)

[2.6 空值 2-7](#_Toc503194181)

[2.7 变量 2-7](#_Toc503194182)

[2.7.2 数据类型 2-8](#_Toc503194183)

[2.7.3 变量 2-13](#_Toc503194184)

[第3章 字符编码 3-14](#_Toc503194185)

# 变量定义规则

* 变量名只能是 字母、数字或下划线的任意组合
* 变量名的第一个字符不能是数字
* 以下关键字不能声明为变量名  
  ['and', 'as', 'assert', 'break', 'class', 'continue', 'def', 'del', 'elif', 'else', 'except', 'exec', 'finally', 'for', 'from', 'global', 'if', 'import', 'in', 'is', 'lambda', 'not', 'or', 'pass', 'print', 'raise', 'return', 'try', 'while', 'with', 'yield']

*# Author：Li Ang  
# encoding=utf8*name = **"你好 ，世界"**

key = “value”

#等号前后要有空格

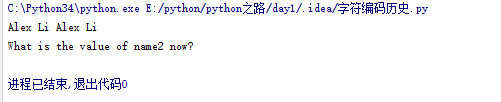
多个变量定义：

*# Author：Li Ang  
# encoding=utf8*name = **"你好 ，世界"**name = **"Alex Li"**

name2 = name

**#变量的传递**  
print(name,name2)  
  
name = **"Jack"**print(**"What is the value of name2 now?"**)

运行结果如下：



# 变量类型

## 整数

Python可以处理任意大小的整数，当然包括负整数，在程序中的表示方法和数学上的写法一模一样，例如：1，100，-8080，0，等等。

计算机由于使用二进制，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。

## 浮点数

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变的，比如，1.23x109和12.3x108是完全相等的。浮点数可以用数学写法，如1.23，3.14，-9.01，等等。但是对于很大或很小的浮点数，就必须用科学计数法表示，把10用e替代，1.23x109就是1.23e9，或者12.3e8，0.000012可以写成1.2e-5，等等。

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的（除法难道也是精确的？是的！），而浮点数运算则可能会有四舍五入的误差。

## 字符串

字符串是以单引号'或双引号"括起来的任意文本，比如'abc'，"xyz"等等。请注意，''或""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。如果'本身也是一个字符，那就可以用""括起来，比如"I'm OK"包含的字符是I，'，m，空格，O，K这6个字符。

## 转义字符

如果字符串内部既包含'又包含"怎么办？可以用转义字符\来标识，比如：

'I\'m \"OK\"!'

表示的字符串内容是：

I'm "OK"!

转义字符\可以转义很多字符，比如\n表示换行，\t表示制表符，字符\本身也要转义，所以\\表示的字符就是\，可以在Python的交互式命令行用print()打印字符串看看：

#单引号和双引号完全相同

>>> **print**('I\'m ok.')

I'm ok.

>>> print ("1\n2\t\t3\n")

1

2 3

>>>

### 打印多行(’’’ ’’’)

如果字符串内部有很多换行，用\n写在一行里不好阅读，为了简化，Python允许用'''...'''的格式表示多行内容，可以自己试试：

>>> print('''line1

... line2

... line3''')

line1

line2

line3

上面是在交互式命令行内输入，注意在输入多行内容时，提示符由>>>变为...，提示你可以接着上一行输入，注意...是提示符，不是代码的一部分：

## 布尔值

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：

>>> True

True

>>> False

False

>>> 3 > 5

False

>>> 3 < 5

True

>>> 3 != 5

True

>>> 3 == 5

False

>>> 3 = 3

File "<stdin>", line 1

SyntaxError: can't assign to literal

>>> 3 == 3

True

>>>

布尔值可以用and、or和not运算。

and运算是与运算，只有所有都为True，and运算结果才是True：

>>> True **and** True

True

>>> True **and** False

False

>>> False **and** False

False

>>> 5 > 3 **and** 3 > 1

True

or运算是或运算，只要其中有一个为True，or运算结果就是True：

>>> True **or** True

True

>>> True **or** False

True

>>> False **or** False

False

>>> 5 > 3 **or** 1 > 3

True

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

>>> **not** True

False

>>> **not** False

True

>>> **not** 1 > 2

True

布尔值经常用在条件判断中，比如：

# Author：Li Ang  
# encoding=utf8  
age = int(input("Age:"))  
print (type(age))  
**if** age > 18:  
 print ("OK")  
**else**:  
 print ("No")

#先判断age > 18的值为True 执行print ("OK") 为False 执行print ("No")

## 空值

空值是Python里一个特殊的值，用None表示。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。

## 变量

变量的概念基本上和初中代数的方程变量是一致的，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。

变量在程序中就是用一个变量名表示了，变量名必须是大小写英文、数字和\_的组合，且不能用数字开头，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。

Answer = **True**

变量Answer是一个布尔值True。

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：

窗体顶端

# -\*- coding: utf-8 -\*-



123   
ABC

#### **数据类型和变量**

阅读: 1035528

### 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，计算机程序理所当然地可以处理各种数值。但是，计算机能处理的远不止数值，还可以处理文本、图形、音频、视频、网页等各种各样的数据，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：

#### **整数**

python 2.0 中有长整型概念

* 长整形
* 整形

32位机器上整数的位数为-2\*\*31~2\*\*31-1

64位机器上整数的位数为-2\*\*63~2\*\*63-1

python2 中当整数的位数超过限定后，python会自动的将其转换为长整形，

python没有限制长整数数值的大小，但实际上计算机内存有限，我们使用的长整数数值不可能无限大。

Python 2.7.13 (v2.7.13:a06454b1afa1, Dec 17 2016, 20:53:40) [MSC v.1500 64 bit (AMD64)] on win32

Type "help", "copyright", "credits" or "license" for more information.

>>> type(2\*\*30)

<type 'int'>

>>> type(2\*\*31)

<type 'long'>

>>>

在python3 中没有了长整形概念，同一为整形。

Python 3.4.4 (v3.4.4:737efcadf5a6, Dec 20 2015, 20:20:57) [MSC v.1600 64 bit (AMD64)] on win32

Type "help", "copyright", "credits" or "license" for more information.

>>> type(2\*\*64)

<class 'int'>

>>> type(2\*\*31)

<class 'int'>

>>>

Python可以处理任意大小的整数，当然包括负整数，在程序中的表示方法和数学上的写法一模一样，例如：1，100，-8080，0，等等。

计算机由于使用二进制，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。

#### **浮点数**

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变的，比如，1.23x109和12.3x108是完全相等的。浮点数可以用数学写法，如1.23，3.14，-9.01，等等。但是对于很大或很小的浮点数，就必须用科学计数法 表示，把10用e替代，1.23x109就是1.23e9，或者12.3e8，0.000012可以写成1.2e-5，等等。

>>> 2.13E3

2130.0

>>> 21.3E2

2130.0

>>>

>>> 2.13E4

21300.0

>>> 2.13 \* 10\*\*4

21300.0

>>>

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的（除法难道也是精确的？是的！），而浮点数运算则可能会有四舍五入的误差。

#### **字符串**

字符串是以单引号'或双引号"括起来的任意文本，比如'abc'，"xyz"等等。请注意，''或""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。如果'本身也是一个字符，那就可以用""括起来，比如"I'm OK"包含的字符是I，'，m，空格，O，K这6个字符。

如果字符串内部既包含'又包含"怎么办？可以用转义字符\来标识，比如：

'I\'m \"OK\"!'

表示的字符串内容是：

I'm "OK"!

转义字符\可以转义很多字符，比如\n表示换行，\t表示制表符，字符\本身也要转义，所以\\表示的字符就是\，可以在Python的交互式命令行用print()打印字符串看看：

>>> **print**('I\'m ok.')

I'm ok.

>>> print('I\'m learning\nPython.')

I'm learning

Python.

>>> print('\\\n\\')

\

\

如果字符串里面有很多字符都需要转义，就需要加很多\，为了简化，Python还允许用r''表示''内部的字符串默认不转义，可以自己试试：

>>> print('\\\t\\')

\ \

>>> print(r'\\\t\\')

\\\t\\

如果字符串内部有很多换行，用\n写在一行里不好阅读，为了简化，Python允许用'''...'''的格式表示多行内容，可以自己试试：

>>> print('''line1

... line2

... line3''')

line1

line2

line3

上面是在交互式命令行内输入，注意在输入多行内容时，提示符由>>>变为...，提示你可以接着上一行输入，注意...是提示符，不是代码的一部分：

┌────────────────────────────────────────────────────────┐

│Command Prompt - python \_ □ x │

├────────────────────────────────────────────────────────┤

│>>> print('''line1 │

│... line2 │

│... line3''') │

│line1 │

│line2 │

│line3 │

│ │

│>>> \_ │

│ │

│ │

│ │

└────────────────────────────────────────────────────────┘

当输入完结束符``` 和括号)后，执行该语句并打印结果。

如果写成程序并存为.py文件，就是：

print('''line1

line2

line3''')

多行字符串'''...'''还可以在前面加上r使用，请自行测试：

窗体顶端

# -\*- coding: utf-8 -\*-

 Run

窗体底端

#### **布尔值**

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：

>>> True

True

>>> False

False

>>> 3 > 2

True

>>> 3 > 5

False

布尔值可以用and、or和not运算。

and运算是与运算，只有所有都为True，and运算结果才是True：

>>> True **and** True

True

>>> True **and** False

False

>>> False **and** False

False

>>> 5 > 3 **and** 3 > 1

True

or运算是或运算，只要其中有一个为True，or运算结果就是True：

>>> True **or** True

True

>>> True **or** False

True

>>> False **or** False

False

>>> 5 > 3 **or** 1 > 3

True

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

>>> **not** True

False

>>> **not** False

True

>>> **not** 1 > 2

True

布尔值经常用在条件判断中，比如：

**if** age >= 18:

**print**('adult')

**else**:

**print**('teenager')

#### **空值**

空值是Python里一个特殊的值，用None表示。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。

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，我们后面会继续讲到。

### 变量

变量的概念基本上和初中代数的方程变量是一致的，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。

变量在程序中就是用一个变量名表示了，变量名必须是大小写英文、数字和\_的组合，且不能用数字开头，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。

Answer = **True**

变量Answer是一个布尔值True。

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：

窗体顶端

# -\*- coding: utf-8 -\*-

 Run

123   
ABC

窗体底端

这种变量本身类型不固定的语言称之为动态语言，与之对应的是静态语言。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（// 表示注释）：

**int** a = 123; *// a是整数类型变量*

a = "ABC"; *// 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。

请不要把赋值语句的等号等同于数学的等号。比如下面的代码：

x = 10

x = x + 2

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

窗体底端

# 字符编码

python解释器在加载 .py 文件中的代码时，会对内容进行编码（默认ascill）

ASCII（American Standard Code for Information Interchange，美国标准信息交换代码）是基于拉丁字母的一套电脑编码系统，主要用于显示现代英语和其他西欧语言，其最多只能用 8 位来表示（一个字节），即：2\*\*8 = 256-1，所以，ASCII码最多只能表示 255 个符号。



**关于中文**

为了处理汉字，程序员设计了用于简体中文的GB2312和用于繁体中文的big5。

GB2312(1980年)一共收录了7445个字符，包括6763个汉字和682个其它符号。汉字区的内码范围高字节从B0-F7，低字节从A1-FE，占用的码位是72\*94=6768。其中有5个空位是D7FA-D7FE。

GB2312 支持的汉字太少。1995年的汉字扩展规范GBK1.0收录了21886个符号，它分为汉字区和图形符号区。汉字区包括21003个字符。2000年的 GB18030是取代GBK1.0的正式国家标准。该标准收录了27484个汉字，同时还收录了藏文、蒙文、维吾尔文等主要的少数民族文字。现在的PC平台必须支持GB18030，对嵌入式产品暂不作要求。所以手机、MP3一般只支持GB2312。

从ASCII、GB2312、GBK 到GB18030，这些编码方法是向下兼容的，即同一个字符在这些方案中总是有相同的编码，后面的标准支持更多的字符。在这些编码中，英文和中文可以统一地处理。区分中文编码的方法是高字节的最高位不为0。按照程序员的称呼，GB2312、GBK到GB18030都属于双字节字符集 (DBCS)。

有的中文Windows的缺省内码还是GBK，可以通过GB18030升级包升级到GB18030。不过GB18030相对GBK增加的字符，普通人是很难用到的，通常我们还是用GBK指代中文Windows内码。

**显然ASCII码无法将世界上的各种文字和符号全部表示，所以，就需要新出一种可以代表所有字符和符号的编码，即：Unicode**

Unicode（统一码、万国码、单一码）是一种在计算机上使用的字符编码。Unicode 是为了解决传统的字符编码方案的局限而产生的，它为每种语言中的每个字符设定了统一并且唯一的二进制编码，规定虽有的字符和符号最少由 16 位来表示（2个字节），即：2 \*\*16 = 65536，  
注：此处说的的是最少2个字节，可能更多

UTF-8，是对Unicode编码的压缩和优化，他不再使用最少使用2个字节，而是将所有的字符和符号进行分类：ascii码中的内容用1个字节保存、欧洲的字符用2个字节保存，东亚的字符用3个字节保存...

所以，python解释器在加载 .py 文件中的代码时，会对内容进行编码（默认ascill），如果是如下代码的话：

报错：ascii码无法表示中文

|  |  |
| --- | --- |
| 1  2  3 | #!/usr/bin/env python    print "你好，世界" |

改正：应该显示的告诉python解释器，用什么编码来执行源代码，即：

|  |  |
| --- | --- |
| 1  2  3  4 | #!/usr/bin/env python  # -\*- coding: utf-8 -\*-    print "你好，世界" |

**注释：**

　　当行注视：# 被注释内容

　　多行注释：""" 被注释内容 """