**变量赋值**

Python 中的变量赋值不需要类型声明。

每个变量在内存中创建，都包括变量的标识，名称和数据这些信息。

每个变量在使用前都必须赋值，变量赋值以后该变量才会被创建。

等号（=）用来给变量赋值。

等号（=）运算符左边是一个变量名,等号（=）运算符右边是存储在变量中的值。例如：

#!/usr/bin/python

# -\*- coding: UTF-8 -\*-

counter = 100 # 赋值整型变量

miles = 1000.0 # 浮点型

name = "John" # 字符串

print counter

print miles

print name

[**运行实例 »**](http://www.runoob.com/try/runcode.php?filename=demo_variable&type=python) [下载](http://static.runoob.com/code/python/demo_variable.py)

以上实例中，100，1000.0和"John"分别赋值给counter，miles，name变量。

执行以上程序会输出如下结果：

100

1000.0

John

**多个变量赋值**

Python允许你同时为多个变量赋值。例如：

a = b = c = 1

以上实例，创建一个整型对象，值为1，三个变量被分配到相同的内存空间上。

您也可以为多个对象指定多个变量。例如：

a, b, c = 1, 2, "john"

以上实例，两个整型对象1和2的分配给变量a和b，字符串对象"john"分配给变量c。

**标准数据类型**

在内存中存储的数据可以有多种类型。

例如，一个人的年龄可以用数字来存储，他的名字可以用字符来存储。

Python有一些标准类型用于定义操作上，他们和为他们每个人的存储方法可能。

Python有五个标准的数据类型：

* Numbers（数字）
* String（字符串）
* List（列表）
* Tuple（元组）
* Dictionary（字典）

**Python数字**

数字数据类型用于存储数值。

他们是不可改变的数据类型，这意味着改变数字数据类型会分配一个新的对象。

当你指定一个值时，Number对象就会被创建：

var1 = 1  
var2 = 10

您也可以使用del语句删除一些对象的引用。

del语句的语法是：

del var1[,var2[,var3[....,varN]]]]

您可以通过使用del语句删除单个或多个对象的引用。例如：

del var  
del var\_a, var\_b

Python支持四种不同的数字类型：

* int（有符号整型）
* long（长整型[也可以代表八进制和十六进制]）
* float（浮点型）
* complex（复数）

**实例**

一些数值类型的实例：

|  |  |  |  |
| --- | --- | --- | --- |
| **int** | **long** | **float** | **complex** |
| 10 | 51924361L | 0.0 | 3.14j |
| 100 | -0x19323L | 15.20 | 45.j |
| -786 | 0122L | -21.9 | 9.322e-36j |
| 080 | 0xDEFABCECBDAECBFBAEl | 32.3+e18 | .876j |
| -0490 | 535633629843L | -90. | -.6545+0J |
| -0x260 | -052318172735L | -32.54e100 | 3e+26J |
| 0x69 | -4721885298529L | 70.2-E12 | 4.53e-7j |

* 长整型也可以使用小写"L"，但是还是建议您使用大写"L"，避免与数字"1"混淆。Python使用"L"来显示长整型。
* Python还支持复数，复数由实数部分和虚数部分构成，可以用a + bj,或者complex(a,b)表示， 复数的实部a和虚部b都是浮点型

**Python字符串**

字符串或串(String)是由数字、字母、下划线组成的一串字符。

一般记为 :

s="a1a2···an"(n>=0)

它是编程语言中表示文本的数据类型。

python的字串列表有2种取值顺序:

* 从左到右索引默认0开始的，最大范围是字符串长度少1
* 从右到左索引默认-1开始的，最大范围是字符串开头

如果你的实要取得一段子串的话，可以用到变量[头下标:尾下标]，就可以截取相应的字符串，其中下标是从0开始算起，可以是正数或负数，下标可以为空表示取到头或尾。

比如:

s = 'ilovepython'

**s[1:5]**的结果是love。

当使用以冒号分隔的字符串，python返回一个新的对象，结果包含了以这对偏移标识的连续的内容，左边的开始是包含了下边界。

上面的结果包含了s[1]的值l，而取到的最大范围不包括上边界，就是s[5]的值p。

加号（+）是字符串连接运算符，星号（\*）是重复操作。如下实例：

**实例(Python 2.0+)**

#!/usr/bin/python # -\*- coding: UTF-8 -\*- str = 'Hello World!' print str # 输出完整字符串 print str[0] # 输出字符串中的第一个字符 print str[2:5] # 输出字符串中第三个至第五个之间的字符串 print str[2:] # 输出从第三个字符开始的字符串 print str \* 2 # 输出字符串两次 print str + "TEST" # 输出连接的字符串

以上实例输出结果：

Hello World!

H

llo

llo World!

Hello World!Hello World!

Hello World!TEST

**Python列表**

List（列表） 是 Python 中使用最频繁的数据类型。

列表可以完成大多数集合类的数据结构实现。它支持字符，数字，字符串甚至可以包含列表（所谓嵌套）。

列表用[ ]标识。是python最通用的复合数据类型。看这段代码就明白。

列表中的值得分割也可以用到变量[头下标:尾下标]，就可以截取相应的列表，从左到右索引默认0开始的，从右到左索引默认-1开始，下标可以为空表示取到头或尾。

加号（+）是列表连接运算符，星号（\*）是重复操作。如下实例：

**实例(Python 2.0+)**

#!/usr/bin/python # -\*- coding: UTF-8 -\*- list = [ 'runoob', 786 , 2.23, 'john', 70.2 ] tinylist = [123, 'john'] print list # 输出完整列表 print list[0] # 输出列表的第一个元素 print list[1:3] # 输出第二个至第三个的元素 print list[2:] # 输出从第三个开始至列表末尾的所有元素 print tinylist \* 2 # 输出列表两次 print list + tinylist # 打印组合的列表

以上实例输出结果：

['runoob', 786, 2.23, 'john', 70.2]

runoob

[786, 2.23]

[2.23, 'john', 70.2]

[123, 'john', 123, 'john']

['runoob', 786, 2.23, 'john', 70.2, 123, 'john']

**Python元组**

元组是另一个数据类型，类似于List（列表）。

元组用"()"标识。内部元素用逗号隔开。但是元组不能二次赋值，相当于只读列表。

**实例(Python 2.0+)**

#!/usr/bin/python # -\*- coding: UTF-8 -\*- tuple = ( 'runoob', 786 , 2.23, 'john', 70.2 ) tinytuple = (123, 'john') print tuple # 输出完整元组 print tuple[0] # 输出元组的第一个元素 print tuple[1:3] # 输出第二个至第三个的元素 print tuple[2:] # 输出从第三个开始至列表末尾的所有元素 print tinytuple \* 2 # 输出元组两次 print tuple + tinytuple # 打印组合的元组

以上实例输出结果：

('runoob', 786, 2.23, 'john', 70.2)

runoob

(786, 2.23)

(2.23, 'john', 70.2)

(123, 'john', 123, 'john')

('runoob', 786, 2.23, 'john', 70.2, 123, 'john')

以下是元组无效的，因为元组是不允许更新的。而列表是允许更新的：

**实例(Python 2.0+)**

#!/usr/bin/python # -\*- coding: UTF-8 -\*- tuple = ( 'runoob', 786 , 2.23, 'john', 70.2 ) list = [ 'runoob', 786 , 2.23, 'john', 70.2 ] tuple[2] = 1000 # 元组中是非法应用 list[2] = 1000 # 列表中是合法应用

**Python元字典**

字典(dictionary)是除列表以外python之中最灵活的内置数据结构类型。列表是有序的对象结合，字典是无序的对象集合。

两者之间的区别在于：字典当中的元素是通过键来存取的，而不是通过偏移存取。

字典用"{ }"标识。字典由索引(key)和它对应的值value组成。

**实例(Python 2.0+)**

#!/usr/bin/python # -\*- coding: UTF-8 -\*- dict = {} dict['one'] = "This is one" dict[2] = "This is two" tinydict = {'name': 'john','code':6734, 'dept': 'sales'} print dict['one'] # 输出键为'one' 的值 print dict[2] # 输出键为 2 的值 print tinydict # 输出完整的字典 print tinydict.keys() # 输出所有键 print tinydict.values() # 输出所有值

输出结果为：

This is one

This is two

{'dept': 'sales', 'code': 6734, 'name': 'john'}

['dept', 'code', 'name']

['sales', 6734, 'john']

**Python数据类型转换**

有时候，我们需要对数据内置的类型进行转换，数据类型的转换，你只需要将数据类型作为函数名即可。

以下几个内置的函数可以执行数据类型之间的转换。这些函数返回一个新的对象，表示转换的值。

|  |  |
| --- | --- |
| **函数** | **描述** |
| int(x [,base]) | 将x转换为一个整数 |
| long(x [,base] ) | 将x转换为一个长整数 |
| float(x) | 将x转换到一个浮点数 |
| complex(real [,imag]) | 创建一个复数 |
| str(x) | 将对象 x 转换为字符串 |
| repr(x) | 将对象 x 转换为表达式字符串 |
| eval(str) | 用来计算在字符串中的有效Python表达式,并返回一个对象 |
| tuple(s) | 将序列 s 转换为一个元组 |
| list(s) | 将序列 s 转换为一个列表 |
| set(s) | 转换为可变集合 |
| dict(d) | 创建一个字典。d 必须是一个序列 (key,value)元组。 |
| frozenset(s) | 转换为不可变集合 |
| chr(x) | 将一个整数转换为一个字符 |
| unichr(x) | 将一个整数转换为Unicode字符 |
| ord(x) | 将一个字符转换为它的整数值 |
| hex(x) | 将一个整数转换为一个十六进制字符串 |
| oct(x) | 将一个整数转换为一个八进制字符串 |