

# 第2章 数据类型

# Outline

- 2.1 **Python**的对象模型
- 2.2 Python变量
- 2.3 数据类型
  - 2.3.1 Boolean类型
  - 2.3.2 数字
  - 2.3.3 字符串

## 2.1 Python的对象模型

对象是python语言中最基本的概念，在python中处理的每样东西都是对象。

python中有许多内置对象可供编程者使用，内置对象可直接使用，如数字、字符串、列表、del等；非内置对象需要导入模块才能使用，如正弦函数sin(x)，随机数产生函数random( )等。

# 2.1 Python的对象模型

- 常用内置对象

对象类型	示例
数字	1234, 3.14, 3+4j
字符串	'swfu', "I'm student", ""Python ""
列表	[1, 2, 3]
字典	{1:'food' ,2:'taste', 3:'import'}
元组	(2, -5, 6)
文件	f=open('data.dat', 'r')
集合	set('abc'), {'a', 'b', 'c'}
布尔型	True, False
空类型	None
异常	Exception, ValueError, TypeError
文件	f = open('data.txt','rb')
编程单元类型	函数、模块、类

## 2.2 Python变量

**变量名可以包括字母、数字和下划线，但是数字不能作为开头字符**

- ◆ 例如，test1是有效变量名，而1test则是无效变量名

**系统关键字不能做变量名使用**

- ◆ 例如，and、break等都是系统关键字，不能作为变量名使用

**Python的变量名区分大小写**

- ◆ 例如，test和Test是两个不同的变量

提示 Python 3.x默认使用UTF-8编码，变量名中允许包含中文，如“测试”是一个有效的变量名。

## 2.2 Python变量

在Python中，不需要事先声明变量名及其类型，直接赋值即可创建各种类型的对象变量。例如语句

```
>>> x = 3
```

创建了整型变量x，并赋值为3，再例如语句

```
>>> x = 'Hello world.'
```

创建了字符串变量x，并赋值为'Hello world.'。这一点适用于Python任意类型的对象。

## 2.2 Python变量

- 虽然不需要在使用之前显式地声明变量及其类型，但是Python仍属于强类型语言。编程支持的是动态类型，Python解释器会根据变量的用途来推断其类型。Python还支持运算符重载，这允许用户为自定义类型定义运算符的行为。Python是一种动态类型语言，这意味着变量的类型可以在程序运行过程中发生变化。Python的变量名与类型是解耦的，这允许变量指向不同类型的对象。Python的变量名与类型是解耦的，这允许变量指向不同类型的对象。

```
>>> x = 3
>>> print(type(x))
<class 'int'>
>>> x = 'Hello world.'
>>> print(type(x))
<class 'str'>
>>> x = [1, 2, 3]
>>> print(type(x))
<class 'list'>
>>> isinstance(3, int)
True
>>> isinstance('Hello world', str)
True
```

## 2.2 Python变量

- 内置函数`type()`用来返回变量类型，内置函数`isinstance()`用来测试对象是否为指定类型的实例。代码中首先创建了整型变量`x`，然后又分别创建了字符串和列表类型的变量`x`。当创建了字符串类型的变量`x`之后，之前创建的整型变量`x`自动失效，创建列表对象`x`之后，之前创建的字符串变量`x`自动失效。可以将该模型理解为“状态机”，在显式修改其类型或删除之前，变量将一直保持上次的类型。



## 2.2 Python变量

- 在大多数情况下，如果变量出现在赋值运算符或复合赋值运算符（例如+=、\*=等等）的左边则表示创建变量或修改变量的值，否则表示引用该变量的值，这一点同样适用于使用下标来访问列表、字典等可变序列以及其他自定义对象中元素的情况。例如下面的代码：

```
>>> x = 3 #创建整型变量
>>> print(x**2)
9
>>> x += 6 #修改变量值
>>> print(x) #读取变量值并输出显示
9
>>> x = [1, 2, 3] #创建列表对象
>>> print(x)
[1, 2, 3]
>>> x[1] = 5 #修改列表元素值
>>> print(x) #输出显示整个列表
[1, 5, 3]
>>> print(x[2]) #输出显示列表指定元素
3
```

## 2.2 Python变量

- 字符串和元组属于不可变序列，这意味着不能通过下标的方式来修改其中的元素值，例如下面的代码试图修改元组中元素的值时抛出异常。

```
>>> x = (1, 2, 3)
```

```
>>> print(x)
```

```
(1, 2, 3)
```

```
>>> x[1] = 5
```

```
Traceback (most recent call last):
```

```
File "<pyshell#7>", line 1, in <module>
```

```
    x[1] = 5
```

```
TypeError: 'tuple' object does not support item assignment
```

## 2.2 Python变量

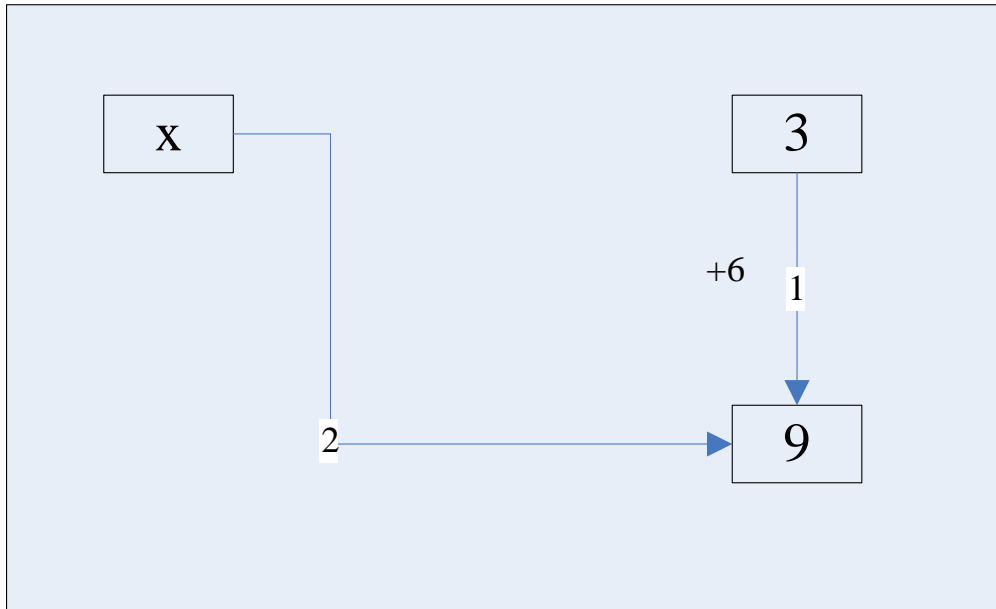
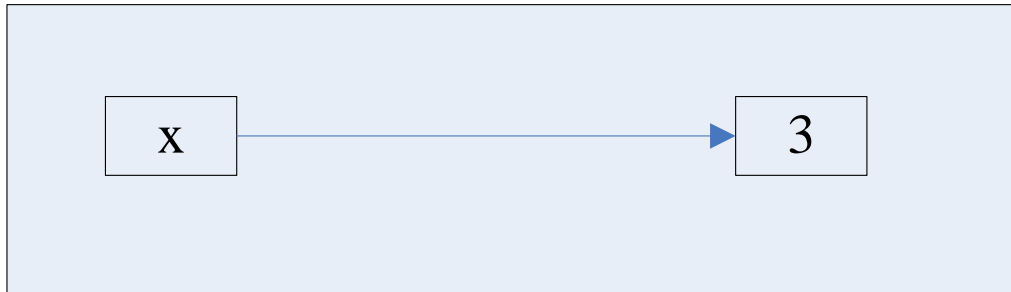
■ 在Python中，允许多个变量指向同一个值，例如：

```
>>> x = 3
>>> id(x)
1786684560
>>> y = x
>>> id(y)
1786684560
```

■ 然而，需要注意的是，继续上面的示例代码，当为其中一个变量的修改值以后，其内存地址将会变化，但这并不影响另一个变量，例如接着上面的代码再继续执行下面的代码：

```
>>> x += 6
>>> id(x)
1786684752
>>> y
3
>>> id(y)
1786684560
```

## 2.2 Python变量



## 2.2 Python变量

- Python采用的是基于值的内存管理方式，如果为不同变量赋值为相同值，这个值在内存中只有一份，多个变量指向同一块内存地址，前面的几段代码也说明了这个特点。再例如下面的代码：

```
>>> x = 3
>>> id(x)
10417624
>>> y = 3
>>> id(y)
10417624
>>> y = 5
>>> id(y)
10417600
>>> id(x)
10417624
```

## 2.2 Python变量

- Python具有自动内存管理功能，对于没有任何变量指向的值，Python自动将其删除。Python会跟踪所有的值，并自动删除不再有变量指向的值。因此，Python程序员一般情况下不需要太多考虑内存管理的问题。尽管如此，显式使用del命令删除不需要的值或显式关闭不再需要访问的资源，仍是一个好的习惯，同时也是一个优秀程序员的基本素养之一。

## 2.2 Python变量

- 最后，在定义变量名的时候，需要注意以下问题：
- 变量名必须以字母或下划线开头，但以下划线开头的变量在Python中有特殊含义，本书后面第6章会详细讲解；
- 变量名中不能有空格以及标点符号（括号、引号、逗号、斜线、反斜线、冒号、句号、问号等等）；
- 不能使用关键字作变量名，可以导入keyword模块后使用`print(keyword.kwlist)`查看所有Python关键字；
- 不建议使用系统内置的模块名、类型名或函数名以及已导入的模块名及其成员名作变量名，这将会改变其类型和含义，可以通过`dir(__builtins__)`查看所有内置模块、类型和函数；
- 变量名对英文字母的大小写敏感，例如student和Student是不同的变量。

## 2.3 数据类型

一种编程语言所支持的数据类型决定了该编程语言所能保存的数据

### Python语言常用的内置数据类型

◆ Number（数字）、String（字符串）、List（列表）、  
Tuple（元组）、Set（集合）、Dictionary（字典）

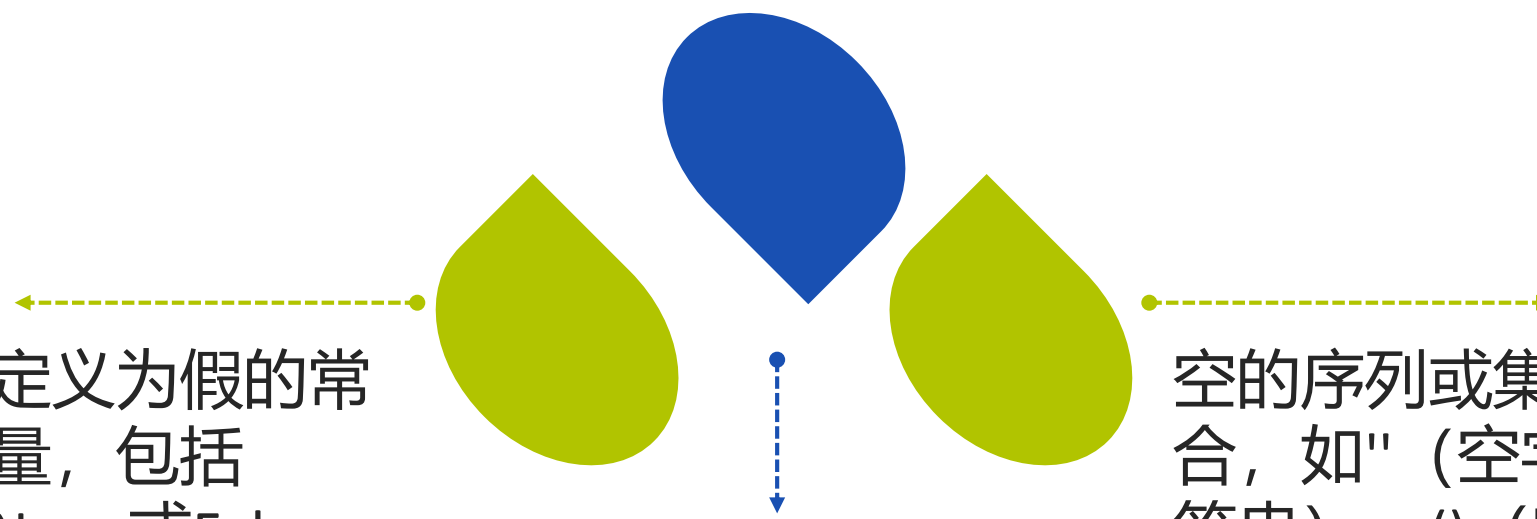
### Python中有3种不同的数字类型

◆ int（整型）、float（浮点型）、complex（复数类型）。



## 2.3.1 Boolean类型

使用bool函数可以将其他类型的数据转为Boolean类型，当给bool函数传入下列参数时其将会返回False：



定义为假的常量，包括None或False

任意值为0的数值，如0、0.0、0j等

空的序列或集合，如"（空字符串）、()（空元组）、[]（空列表）等

### 2.3.2 数字

- 数字是python中最常用的对象，属于不可变对象。
- 可以表示任意大小的数字。

```
>>> a=999999999999999999999999999999999999
```

>>> a\*a

[illegible]

```
>>> a**3
```

[illegible]

Python的IDEL交互界面可以当做简便计算器来使用。

## 2.3.2 数字

- 十进制整数如，0、-1、9、123
- 十六进制整数，需要16个数字0、1、2、3、4、5、6、7、8、9、a、b、c、d、e、f来表示整数，必须以0x开头，如0x10、0xfa、0xabcdef
- 八进制整数，只需要8个数字0、1、2、3、4、5、6、7来表示整数，必须以0o开头，如0o35、0o11
- 二进制整数、只需要2个数字0、1来表示整数，必须以0b开头如，0b101、0b100

## 2.3.2 数字-浮点型

浮点型数字使用C语言中的double类型实现，可以用来表示实数

- ◆ 如3.14159、-10.5、3.25e3等
- ◆ 3.25e3是科学记数法的表示方式，其中e表示10，因此，3.25e3实际上表示的浮点数是 $3.25 \times 10^3 = 3250.0$

查看浮点数的取值范围和精度的代码示例

```
import sys #导入sys包  
sys.float_info #查看当前环境中浮点型数字的取值范围和精度
```

## 2.3.2 数字 - 复数类型

复数由实部和虚部组成，每一部分都是一个浮点数，其书写方法如下：

$a+bj$ 或 $a+bj$

其中， $a$ 和 $b$ 是两个数字， $j$ 或 $J$ 是虚部的后缀，即 $a$ 是实部、 $b$ 是虚部

在生成复数时，也可以使用`complex`函数，其语法格式如下：

`complex([real[,imag]])`

其中，`real`为实部值，`imag`为虚部值，返回值为`real+imag*1j`

## 2.3.2 数字 - 复数类型

- Python内置支持复数类型。

```
>>> a = 3+4j
>>> b = 5+6j
>>> c = a+b
>>> c
(8+10j)
>>> c.real #查看复数实部
8.0
>>> c.imag #查看复数虚部
10.0
>>> a.conjugate() #返回共轭复数
(3-4j)
>>> a*b #复数乘法
(-9+38j)
>>> a/b #复数除法
(0.6393442622950819+0.03278688524590165j)
```

## 2.3.2 数字

- Python 3.6.x支持在数字中使用单个下划线作为分隔符，目的是提高数字的可读性。类似在数字中使用逗号作为千位分隔符。
- `>>>1_000_000`
- `>>>1_2_3_4`
- `>>>1_2+3_4j`
- `>>>1_2.3_45`

## 2.3.3 字符串

Python语言中只有用于保存字符串的String类型，而没有用于保存单个字符的数据类型

Python中的字符串可以写在一对单引号中，也可以写在一对双引号或一对三双引号中

三种写法的区别将在后面介绍，目前我们使用一对单引号或一对双引号的写法

对于不包含任何字符的字符串，如"（一对单引号）或""（一对双引号），称为空字符串（或简称为空串）



## 2.3.3 字符串

- 用单引号、双引号或三引号括起来的符号系列称为字符串
- 单引号、双引号、三单引号、三双引号可以互相嵌套，用来表示复杂字符串。
- 'abc'、'123'、'中国'、"Python"
- 字符串属于不可变序列
- 空串表示为"或 ""
- 三引号""或"""表示的字符串可以换行，支持排版较为复杂的字符串；三引号还可以在程序中表示较长的注释。

## 2.3.3 字符串

### 1. 字符串合并

```
>>> a='abc' + '123'      #生成新对象
>>> x= '1234' + 'abcd'   #生成新对象
>>> x
>>> x=x 'edf'
```

### 2. 字符串格式化

```
>>>a = 3.6674
>>>'%7.3f' % a
' 3.667'
>>> "%d:%c"%(65,65)
'65:A'
>>> """My name is %s, and my age is %d"""%( 'nankai',38)
'My name is Dong Fuguo, and my age is 38'
```

## 2.3.3 字符串

### 3. 转义字符

- `\n`: 换行符
- `\t`: 制表符
- `\r`: 回车
- `\'`: 单引号
- `\"`: 双引号
- `\\`: 一个\
- `\ddd`: 3位八进制数对应的字符
- `\xhh`: 2位十六进制数对应的字符

字符串界定符前面加字母r表示原始字符串，其中的特殊字符不进行转义，但字符串的最后一个字符不能是\。

## 2.3.3 字符串-常用转义字符

转义字符	描述	转义字符	描述
\（在行尾时）	续行符	\n	换行
\\	反斜杠符号	\r	回车
\'	单引号	\t	制表符
\"	双引号	\ooo	3位八进制数对应的字符('\141')
\uhhhh	4位16进制表示的Unicode字符 '\u4E2D\u56FD'	\xhh	2位十六进制数对应的字符('\x61')

## 2.3.3 字符串

- 字符串前面加r或者R表示原始字符串，其中的特殊字符不进行转义，但是字符串的最后一个字符不能是\。
- 原始字符串主要用于正则表达式、文件路径或者URL等。
- `>>> print('c:\windows\noodle.exe')`
- `>>> print(r'c:\windows\noodle.exe')`