

# 数据类型

# 数字类型

- ◎ 整数
- ◎ 浮点数
- ◎ 复数

# 整数 (int)

- ◎ 与数学中整数概念一致
- ◎ 整数可正可负
- ◎ 默认情况下，整数采用十进制。其它进制需要增加相应的引导符号

# 整数不同进制表示

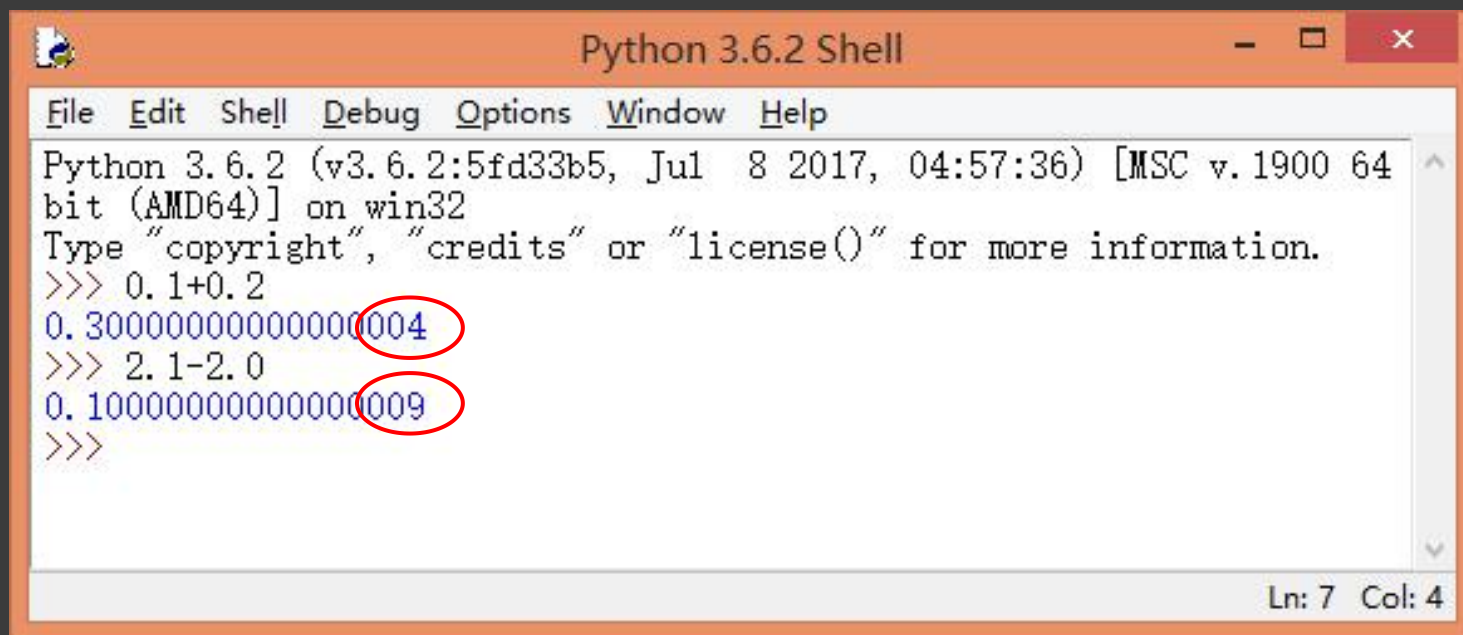
进制	前导符	示例	十进制
◎ 二进制：	0b或0B	0b10	2
◎ 八进制：	0o或0O	0o10	8
◎ 十六进制：	0x 或0X	0x10	16

# 浮点数(float)

- ◎ 与数学中的实数概念一致
- ◎ 取值范围与精度都有限制
- ◎ 表示方式:
  - 小数: 1.23, 3.14, -9.01
  - 科学计数法:  $1.23 \times 10^9$       1.23e9  
                         0.000012      1.2e-5

# 浮点数运算

- ◎ 浮点数运算存在不确定尾数，有误差



```
Python 3.6.2 Shell
File Edit Shell Debug Options Window Help
Python 3.6.2 (v3.6.2:5fd33b5, Jul 8 2017, 04:57:36) [MSC v.1900 64
bit (AMD64)] on win32
Type "copyright", "credits" or "license()" for more information.
>>> 0.1+0.2
0.30000000000000004
>>> 2.1-2.0
0.10000000000000009
>>>
```

The screenshot shows a Python 3.6.2 Shell window. The title bar is "Python 3.6.2 Shell". The menu bar includes "File", "Edit", "Shell", "Debug", "Options", "Window", and "Help". The main text area shows the following output:

```
Python 3.6.2 (v3.6.2:5fd33b5, Jul 8 2017, 04:57:36) [MSC v.1900 64
bit (AMD64)] on win32
Type "copyright", "credits" or "license()" for more information.
>>> 0.1+0.2
0.30000000000000004
>>> 2.1-2.0
0.10000000000000009
>>>
```

The trailing zeros in the results "0.30000000000000004" and "0.10000000000000009" are circled in red, highlighting the floating-point precision error.

Ln: 7 Col: 4

# 复数(complex)

- 与数学中的复数概念一致
- 由实部和虚部组成，虚部用j表示。
- 例如：

$$2 + 3j$$

实部 虚部

real 方法取实部

imag 方法取虚部

complex() 函数可用于创建一个值为  $\text{real} + \text{imag} * j$  的复数。

# 算术运算符

运算符	说明	示例	运算结果
+	加法	5+10	15
-	减法	100-5	95
*	乘法	8*9	72
/	浮点数除法	100/5	20.0
//	整数除法	100//5	20
%	模（求余）	9%4	1
**	幂	2**3	8



# 数学库(math)

函数或常数	功能
e	自然常数
pi	圆周率
log(x[,base])	返回以base为底的对数，缺省为e
pow(x,y)	返回x的y次方
sqrt(x)	返回x的平方根
fabs(x)	返回x的绝对值
round(x[,n])	返回浮点数x的四舍五入值，n代表舍入到小数点后的位数
divmod(x,y)	返回x和y的商和余数

# 数学库(math)

使用格式:

- ④ `>>>import math`

- ④ `>>>math.pi`

`3.141592653589793`

`>>>math.sqrt(9)`

`3.0`

# 字符串

- ◎ 字符串是以引号括起来的任意文本
- ◎ 是一个有序序列
- ◎ 表示形式：
  - 单引号: `'abc'`
  - 双引号: `"hello"`
  - 三引号: `"""hello  
world"""`

# 转义字符

<code>\</code>	反斜杠符号
<code>\'</code>	单引号
<code>\"</code>	双引号
<code>\a</code>	响铃
<code>\b</code>	退格(Backspace)
<code>\n</code>	换行
<code>\v</code>	纵向制表符
<code>\t</code>	横向制表符
<code>\r</code>	回车
<code>\f</code>	换页
<code>\ooo</code>	最多三位 八进制, 例如: <code>\12</code> 代表换行
<code>\xyy</code>	十六进制数, yy代表的字符, 例如: <code>\x0a</code> 代表换行

# 字符串运算符

运算符	功能	示例
+	连接字符串	<pre>&gt;&gt;&gt;'hello'+'world' 'helloworld'</pre>
*	复制字符串	<pre>&gt;&gt;&gt;'ab'*3 'ababab'</pre>

# 字符串切片

- 字符串是一个有序序列，可以是正向递增也可以是反向递减：

0	1	2	3	4	5	6	7
a	b	c	d	e	f	g	h
-8	-7	-6	-5	-4	-3	-2	-1

- 索引：在[ ]中给出序号
- 切片：在[ ]中给出切片序号范围

# 布尔值

- ◎ 布尔值：True、False
- ◎ 逻辑运算和关系运算的结果是布尔值

# 关系运算符

运算符	表达式	含义	实例	结果
==	x==y	x等于y	"ABCD"== "ABCDEF"	False
!=	x!=y	x不等于y	"ABCD"!= "abcd"	True
>	x>y	x大于y	"ABC"> "ABD"	False
>=	x>=y	x大于等于y	123>=23	True
<	x<y	x小于y	"ABC"< "DEF"	True
<=	x<=y	x大于等于y	"123"<= "23"	True



# 关系运算符实例

- `>>> 1<3<5` #等价于`1<3 and 3<5`
- `True`
- `>>> 3<5>2`
- `True`
- `>>> 1>6<8`
- `False`
- `>>> import math` #sqrt是math 模块下的函数, 导入math 模块
- `>>> 1>6<math.sqrt(9)`
- `False`
- `>>> 'Hello'> 'world'` # `ascii('H') = 72 < ascii('w') = 119`
- `False`
- `>>> 'Hello'> 3` #字符串和数字不能比较
- `TypeError: unorderable types: str()>int()`

# 逻辑运算符

逻辑量1	逻辑量2	结果
False	False	False
False	True	False
True	False	False
True	True	True
and 运算		

逻辑量1	逻辑量2	结果
False	False	False
False	True	True
True	False	True
True	True	True
or 运算		

逻辑量	结果
False	True
True	False
not 运算	

# 逻辑运算符实例

- `>>> 3>5 and a>3` #注意, 此时并没有定义变量a
- `False`
- `>>> 3>5 or a>3` #3>5的值为False, 所以需要计算后面的表达式
- `NameError: name 'a' is not defined`
- `>>> 3<5 or a>3` #3<5的值为True, 不需要计算后面的表达式
- `True`
- `>>> 3 and 5` #最后一个计算的表达式的值作为整个表达式的值
- `5`
- `>>> 3 and 5>2`
- `True`
- `>>> not 3`
- `False`
- `>>> not 0`
- `True`

# 运算符的优先级和结合性

优先级	运算符	描述	结合性
1	$+x, -x$	正, 负	
2	$x**y$	幂	从右向左
3	$x*y, x/y, x\%y$	乘, 除, 取模	从左向右
4	$x+y, x-y$	加, 减	从左向右
5	$x<y, x\leq y, x==y, x!=y, x\geq y, x>y$	比较	从左向右
6	<code>not x</code>	逻辑否	从左向右
7	<code>x and y</code>	逻辑与	从左向右
8	<code>x or y</code>	逻辑或	从左向右

# 优先级和结合性实例

- ◎ `>>> 3+5 * 4` #先乘后加
- ◎ 23
- ◎ `>>> 5 * 3/2` #从左向右
- ◎ 7.5
- ◎ `>>> 2**3**2` #从右向左
- ◎ 512
- ◎ `>>> 3<5 or a>3` #从左向右
- ◎ True