



Python语言

第三周 Python运算符&字符串

陈世峰 岭南师范学院信息工程学院
chenshifeng@lingnan.edu.cn/13234075348



本章内容

- 1、运算符
- 2、字符串的格式化
- 3、字符串常用操作
- 4、字符串处理函数

表达式和运算符



表达式和运算符

你所编写的大多数语句都包含了**表达式 (Expressions)**。一个表达式的简单例子便是 $2+3$ 。表达式可以拆分成**运算符 (Operators)** 与**操作数 (Operands)**。

运算符 (Operators) 是进行某些操作，并且可以用诸如 $+$ 等符号或特殊关键词加以表达的功能。运算符需要一些数据来进行操作，这些数据就被称作 操作数 (Operands)。在上面的例子中 2 和 3 就是操作数。

Python支持**算术运算符、赋值运算符、位运算符、比较运算符、逻辑运算符、字符串运算符、成员运算符和身份运算符**等基本运算符。

算术运算符

运算符	具体描述	例 子
+	相加运算	1+2的结果是3
-	相减运算	100-1的结果是99
*	乘法运算	2*2的结果是4
/	除法运算（如不能整除，Python 3默认的是提供17位数字的精度）	4/2的结果是2.0
%	求模运算	10%3的结果是1
**	幂运算。x**y返回x的y次幂	2**3的结果是8
//	整除运算，即返回商的整数部分	9//2的结果4

```
x=3
y=4
z=-5
a='a'
b='b'

print(x+y)
print(x-y)
print(a+b)
print(a-b)
print(a*3)
print(x*y)
print(x**y)
print(x/y)
print(x%y)
print(x//y)
print(z//y)
```

运行结果如下：

```
7
-1
ab
出错
aaa
12
81
0.75
3
0
-2
```

赋值运算符

运算符	具体描述	例 子
=	直接赋值	$x = 3$;将3赋值到变量x中
+=	加法赋值	$x += 3$;等同于 $x = x + 3$;
-=	减法赋值	$x -= 3$;等同于 $x = x - 3$;
*=	乘法赋值	$x *= 3$;等同于 $x = x * 3$;
/=	除法赋值	$x /= 3$;等同于 $x = x / 3$;
%=	取模赋值	$x \% = 3$;等同于 $x = x \% 3$;
**=	幂赋值	$x ** = 3$;等同于 $x = x ** 3$;
//=	整除赋值	$x //= 3$;等同于 $x = x // 3$;

```
x = 3
```

```
x += 3
```

```
print(x)
```

```
x -= 3
```

```
print(x)
```

```
x *= 3
```

```
print(x)
```

```
x /= 3
```

```
print(x)
```

运行结果如下：

6

3

9

3.0

位运算符

位运算符	具体描述
&	按位与运算，运算符查看两个表达式的二进制表示法的值，并执行按位“与”操作。只要两个表达式的某位都为 1，则结果的该位为 1；否则，结果的该位为 0（同真为真）
	按位或运算，运算符查看两个表达式的二进制表示法的值，并执行按位“或”操作。只要两个表达式的某位有一个为 1，则结果的该位为 1；否则，结果的该位为 0（同假为假）
^	按位异或运算。异或的运算法则为：0异或0=0，1异或0=1，0异或1=1，1异或1=0
~	按位非运算。0取非运算的结果为1；1取非运算的结果为0
<<	位左移运算，即所有位向左移
>>	位右移运算，即所有位向右移

```
x = 8
```

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

```
print(a)
```

```
b = x << 3
```

```
print(b)
```

```
c = x >> 5
```

```
print(c)
```

运行结果如下：

```
1
```

```
64
```

```
0
```

比较运算符

运算符	具体描述
==	等于运算符（两个=）。例如 <code>a == b</code> ，若a等于b，则返回True；否则返回False
!=	不等运算符。例如 <code>a != b</code> ，若a不等于b，则返回True；否则返回False
<	小于运算符
>	大于运算符
<=	小于等于运算符
>=	大于等于运算符

```
x = 2
y = 2
z = 3
a='str'
b='stR'
```

```
print(id(x) == id(y))
print(x <= z)
print(a == b)
print(type(a)==type(b))
```

```
True
True
False
True
```

逻辑运算符

运算符	具体描述
and	逻辑与运算符。例如 <code>a and b</code> ，当a和b都为True时等于True；否则等于False
or	逻辑或运算符。例如 <code>a or b</code> ，当a和b至少有一个为True时等于True；否则等于False
not	逻辑非运算符。例如 <code>not a</code> ，当a等于True时，表达式等于False；否则等于True

短路运算（惰性求值）

- 当 x 是 `False` 时， $x = \text{False}$; $y = \text{True}$; $x \text{ and } y$ 将返回 `False`。在这一情境中，Python 将不会计算 y ，因为它已经了解 `and` 表达式的左侧是 `False`，这意味着整个表达式都将是 `False` 而不会是别的值。这种情况被称作 **短路计算**（Short-circuit Evaluation）。
- $x = \text{True}$; $y = \text{False}$; $x \text{ or } y$ 将返回 `True`。在这里短路计算同样适用。

例：

`0 and 5`

`'this is string' and 4 or True`

`'this is string' and 4 and True`

`3>5 and a>3`

`3>5 or a>3`

输出结果如下：

`0`

`4`

`True`

`False`

出错

成员运算符

in	成员运算符，如果字符串中包含给定的字符则返回True
not in	成员运算符，如果字符串中未包含给定的字符则返回True
r或者R	指定原始字符串。原始字符串是指所有的字符串都是直接按照字面的意思来使用，没有转义字符、特殊字符或不能打印的字符。原始字符串字符串的第一个引号前加上字母"r"或"R"

```
print(r"hello\nworld!")
```

```
hello\nworld!
```

```
print( 'abc' in 'abcdefg')
```

```
True
```

运行结果如下：

身份运算符

同一性测试运算符（identity comparison）is用来测试两个对象是否是同一个，如果是则返回True，否则返回False。如果两个对象是同一个，二者具有相同的内存地址。

```
>>> 3 is 3
```

```
True
```

```
>>> x = [300, 300, 300]
```

```
>>> x[0] is x[1]          #基于值的内存管理，同一个值在内存中只有一份
```

```
True
```

```
>>> x = [1, 2, 3]
```

```
>>> y = [1, 2, 3]
```

```
>>> x is y                #上面形式创建的x和y不是同一个列表对象
```

```
False
```


条件表达式

条件表达式的形式如下：

<表达式1> if <表达式2> else <表达式3>

计算顺序：先计算<表达式2>的值，如果这个值为True，计算<表达式1>，否则计算<表达式3>。如果条件表达式写在赋值语句里，如：

y = <表达式1> if <表达式2> else <表达式3>

```
>>>5 if 5>6 else 6
```

```
>>>1 if 1>2 else 2 if 5>6 else 3
```

条件表达式的结合性是从右至左。

$$2a \div 2a = ?$$

$$2a \div 2 \times a = ?$$

运算符的优先级

- 如果你有一个诸如 $2 + 3 * 4$ 的表达式，是优先完成加法还是优先完成乘法呢？基础数学知识会告诉我们应该先完成乘法。这意味着乘法运算符的优先级要高于加法运算符
- 教材中给出 Python 中从最低优先级到最高优先级的优先级列表。这意味着，在给定的表达式中，Python 将优先计算列表中位置靠后的那些优先级较高的运算符与表达式。
- 在日常工作中，强烈建议你最好使用圆括号操作符来对运算符与操作数进行分组，以更加明确地指定优先级。这也能使得程序更加可读。

优先级	运算符及操作数形式	意义描述
0	[...], (...), {...}	创建列表、元组和字典
1	s[i], s[i:j]	索引、切片
2	s.attr	属性
3	f(...)	函数调用
4	+a, -a, ~a	一元运算符
5	a**b	乘方（从右至左运算）
6	a*b, a/b, a//b, a%b	乘法、除法、截取除法、取余数
7	a+b, a-b	加法、减法
8	a<<b, a>>b	左移、右移
9	a&b	按位与
10	a^b	按位异或
11	a b	按位或
12	a<b, a<=b, a>b, a>=b, a==b, a!=b	小于、小于等于、大于、大于等于、等于、不等于
13	a is b, a is not b	身份检查
14	a in s, a not in s	序列成员检查
15	not a	逻辑非
16	a and b	逻辑与
17	a or b	逻辑或
18	a if b else c	条件表达式运算符

- 当优先级相同时,按运算符的结合性
- 结合性是指运算的计算是从左开始还是从右开始, Python的运算符绝大多数是从左开始, 只两个特例, **乘方 (**) 和条件表达式运算**从右开始。

字符串格式化



字符串格式化

- 用%操作符格式化字符串
- format()方法

(1)使用%进行格式化

语法规则:

"%[-][+][0][m][.n]格式字符" %x

解释:

第一个%: 格式标志, 表示格式开始

-: 指定左对齐输出

+: 对正数加正号

0: 指定空位填0

m: 指定最小宽度

n: 指定精度

格式字符: 指定类型

%: 格式运算符

x: 待转换的表达式

[]中的为可选

1	%c	格式化字符及其ASCII码
2	%s	格式化字符串
3	%d	格式化整数
4	%u	格式化无符号整型
5	%o	格式化无符号八进制数
6	%x	格式化无符号十六进制数
7	%X	格式化无符号十六进制数 (大写)
8	%f	格式化浮点数字, 可指定小数点后的精度
9	%e	用科学计数法格式化浮点数
10	%E	作用同%e, 用科学计数法格式化浮点数
11	%g	%f和%e的简写
12	%G	%f 和 %E 的简写
13	%p	用十六进制数格式化变量的地址


```
>>> "%d %d"%(12,12.3)
```

```
'12 12'
```

显示十进制数

```
>>> "%6d %6d"%(12,12.3)
```

```
' 12  12'
```

设定十进制数显示宽度

```
>>> "%10s is %-3d years old"("Rose",18)
```

```
'  Rose is 18  years old'
```

显示字符串和整数，分别设置宽度

```
>>> print('%f' % 1.11)
```

```
1.110000
```

默认保留6位小数

```
>>> print('%.1f' % 1.11)
```

```
1.1
```

取1位小数

```
>>> print('%e' % 1.11)
```

```
1.110000e+00
```

默认6位小数，用科学计数法

```
>>> print('%.3e' % 1.11)
```

```
1.110e+00
```

取3位小数，用科学计数法

```
>>> print('%g' % 1111.1111)
```

```
1111.11
```

默认6位有效数字

>>> print('%.7g' % 1111.1111)	
1111.111	# 取7位有效数字
>>> print('%.2g' % 1111.1111)	
1.1e+03	# 取2位有效数字, 自动转换为科学计数法
>>> print('%s' % 'hello world')	
hello world	# 字符串输出
>>> print('%20s' % 'hello world')	
hello world	# 右对齐, 取20位, 不够则补位
>>> print('%-20s' % 'hello world')	
hello world	# 左对齐, 取20位, 不够则补位
>>> print('%.2s' % 'hello world')	
he	# 取2位
>>> print('%10.2s' % 'hello world')	
he	# 右对齐, 取2位
>>> print('%-10.2s' % 'hello world')	
he	# 左对齐, 取2位

(2)使用字符串的format方法进行格式化

字符串format()方法的基本使用格式是：

<模板字符串>.format(<逗号分隔的参数>)

□ 位置参数匹配

如果占位符{}为空（没有表示顺序的序号），按照参数出现的先后次序匹配。如果占位符{}指定参数的序号，按照序号对应参数替换。

□ 使用键值对的关键字参数匹配

format()方法中的参数用键值对形式表示时，在模板字符串中用“键”来表示。

□ 使用序列的索引作为参数匹配

"{ }：计算机{ }的CPU占用率为{ }%。".format("2016-12-31","PYTHON",10)

↑
0

↑
1

↑
2

字符串中槽{}的顺序

↑
0

↑
1

↑
2

format() 中参数的顺序

利用format函数进行字符串的格式化操作

```
>>>"{} {}".format("hello", "world") # 不设置指定位置，按默认顺序  
'hello world'
```

```
>>>"{0} {1}".format("hello", "world") # 设置指定位置  
'hello world'
```

```
>>>"{1} {0} {1}".format("hello", "world") # 设置指定位置  
'world hello world'
```

format()方法的格式控制

- format()方法中模板字符串的槽除了包括参数序号，还可以包括格式控制信息。此时，槽的内部样式如下：{<参数序号>:<格式控制标记>}
- 其中，格式控制标记用来控制参数显示时的格式。格式控制标记包括：<填充><对齐><宽度>,<.精度><类型>6个字段，这些字段都是可选的，可以组合使用，这里按照使用方式逐一介绍。

:	<填充>	<对齐>	<宽度>	,	<.精度>	<类型>
引导符号	用于填充的单个字符	< 左对齐 > 右对齐 ^ 居中对齐	槽的设定输出宽度	数字的千位分隔符 适用于整数和浮点数	浮点数小数部分的精度 或字符串的最大输出长度	整数类型 b, c, d, o, x, X 浮点数类型 e, E, f, %

常见的格式化字符串用法举例

对于浮点数 '0.333' 保留小数点(.)后三位

```
print('{0:.3f}'.format(1.0/3))
```

使用下划线填充文本，并保持文字处于中间位置

使用 (^) 定义 '___hello___'字符串长度为 11

```
print('{0:_^11}'.format('hello'))
```

基于关键词输出 'Swaroop wrote A Byte of Python'

```
print('{name} wrote {book}').
```

```
format(name='Swaroop', book='A Byte of Python'))
```

字符串的操作符



字符串的操作符

操作符	描述
+	连接字符串
*	重复输出字符串
[i]	切片操作。通过索引获取字符串中字符，i是字符的索引
[:]	切片操作。截取字符串中的一部分
in	如果字符串中包含给定的字符返回 True
not in	如果字符串中不包含给定的字符返回 True
r/R	原始字符串。原始字符串用来替代转义符表示的特殊字符，在原字符串的第一个引号前加上字母 r (R)，与普通字符串操作相同。
b	返回二进制字符串，在原字符串的第一个引号前加上字母b,可用于写二进制文件，例如 b"123"。
%	格式化字符串操作符

字符串的索引

■字符串是一个字符序列：字符串最左端位置标记为0，依次增加。字符串中的编号叫做“索引”

H	e	l	l	o		W	o	r	l	d
0	1	2	3	4	5	6	7	8	9	10
-11	-10	-9	-8	-7	-6	-5	-4	-3	-2	-1

■Python中字符串索引从0开始，一个长度为L的字符串最后一个字符的位置是L-1

■Python同时允许使用负数从字符串右边末尾向左边进行反向索引，最右侧索引值是-1

■ 单个索引辅助访问字符串中的特定位置

格式为 <string> [<索引>]

```
>>> a='Hello World'
```

```
>>> a[5]
```

```
''
```

```
>>> a[-5]
```

```
'W'
```

H	e	l	l	o		W	o	r	l	d
0	1	2	3	4	5	6	7	8	9	10
-11	-10	-9	-8	-7	-6	-5	-4	-3	-2	-1

字符串的切片

■可以通过两个索引值确定一个位置范围，返回这个范围的子串

格式： <string>[<start>:<end>]

start和end都是整数型数值，这个子序列从索引start开始直到索引end结束，但不包括end位置。

```
>>> a='Hello World'
```

```
>>> a[1:7]
```

```
'ello W'
```

```
>>> a[:5]
```

```
'Hello'
```

```
>>> a[-8:8]
```

```
'lo Wo'
```

```
>>> a[-5:]
```

```
'World'
```

```
>>> a[6:]
```

```
'World'
```

H	e	l	l	o		W	o	r	l	d
0	1	2	3	4	5	6	7	8	9	10
-11	-10	-9	-8	-7	-6	-5	-4	-3	-2	-1

字符串处理函数



字符串处理函数

内置的字符串处理函数

- 1. 大小写转换函数**
- 2. 查找替换函数**
- 3. 字符判断函数**
- 4. 字符串头尾判断函数**
- 5. 计算函数**
- 6. 字符串拆分与合并**

方法	描述
<code>str.lower()</code>	返回字符串 <code>str</code> 的副本，全部字符小写
<code>str.upper()</code>	返回字符串 <code>str</code> 的副本，全部字符大写
<code>str.islower()</code>	当 <code>str</code> 所有字符都是小写时，返回 <code>True</code> ，否则 <code>False</code>
<code>str.isprintable()</code>	当 <code>str</code> 所有字符都是可打印的，返回 <code>True</code> ，否则 <code>False</code>
<code>str.isnumeric()</code>	当 <code>str</code> 所有字符都是字符时，返回 <code>True</code> ，否则 <code>False</code>
<code>str.isspace()</code>	当 <code>str</code> 所有字符都是空格，返回 <code>True</code> ，否则 <code>False</code>
<code>str.endswith(suffix[,start[,end]])</code>	<code>str[start: end]</code> 以 <code>suffix</code> 结尾返回 <code>True</code> ，否则返回 <code>False</code>
<code>str.startswith(prefix[, start[, end]])</code>	<code>str[start: end]</code> 以 <code>prefix</code> 开始返回 <code>True</code> ，否则返回 <code>False</code>
<code>str.split(sep=None, maxsplit=-1)</code>	返回一个列表，由 <code>str</code> 根据 <code>sep</code> 被分割的部分构成
<code>str.count(sub[,start[,end]])</code>	返回 <code>str[start: end]</code> 中 <code>sub</code> 子串出现的次数
<code>str.replace(old, new[, count])</code>	返回字符串 <code>str</code> 的副本，所有 <code>old</code> 子串被替换为 <code>new</code> ，如果 <code>count</code> 给出，则前 <code>count</code> 次 <code>old</code> 出现被替换
<code>str.center(width[, fillchar])</code>	字符串居中函数，详见函数定义
<code>str.strip([chars])</code>	返回字符串 <code>str</code> 的副本，在其左侧和右侧去掉 <code>chars</code> 中列出的字符
<code>str.zfill(width)</code>	返回字符串 <code>str</code> 的副本，长度为 <code>width</code> ，不足部分在左侧添0
<code>str.format()</code>	返回字符串 <code>str</code> 的一种排版格式，3.6节将详细介绍
<code>str.join(iterable)</code>	返回一个新字符串，由组合数据类型（见第2章） <code>iterable</code> 变量的每个元素组成，元素间用 <code>str</code> 分割

1. 大小写转换函数

函数名
lower()
upper()
capitalize()
swapcase()

```
>>> str1="hi,Python"
```

```
>>> str1.lower()
```

```
'hi,python'
```

```
>>> str1.upper()
```

```
'HI,PYTHON'
```

```
>>> str1.capitalize()
```

```
'Hi,python'
```

```
>>> str1.swapcase()
```

```
'HI,pYTHON'
```

2. 查找替换函数

函数名	功能描述
<code>find(str[,start[,end]])</code>	检测str是否包含在字符串中，如果指定范围start和end，则检查是否包含在指定范围内。如果包含，返回str的索引值，否则返回-1

```
>>> str1="hi,Python!hi,Java!"
>>> str1.find("hi")    #0
>>> str1.rfind("hi")   #10
>>> str1.index("a")
>>> str1.rindex("a")
```


6. 字符串拆分与合并

函数名	功能描述
<code>split(sep, num)</code>	以sep为分隔符分隔字符串，如果num有指定值，则仅截取num个子字符串
<code>join(seq)</code>	以指定字符串作为分隔符，将seq中所有的元素合并为一个新的字符串

```
>>> str1="hi,Python,hi,Java!"
>>> str1.split(",")           #使用逗号做分配符，3个逗号，分隔3次
['hi', 'Python', 'hi', 'Java!']
>>> lst=['hi', 'Python!', 'hi', 'Java!']
>>> s=""
>>> s.join(lst)               #将列表连接为字符串， #'hiPython!hiJava!'
```

```
>>> str1="Hi,Python!"
>>> str1*2    #str1重复显示2次, str1未发生改变
'Hi,Python!Hi,Python!'
>>> id(str1)  #str1在内存中标识
54364264
>>> str1+="Hi,Java!"
>>> id(str1)  #str1连接字符串后, id发生改变
54338768
>>> str1
'Hi,Python!Hi,Java!'
# 字符串切片操作
>>> str1[3:9]
'Python'
```

程序的基本编写方法



IPO程序编写方法



- 输入数据 (Input)
- 处理数据 (Process)
- 输出数据 (Output)

■ 输入数据

输入（Input）是一个程序的开始。程序要处理的数据有多种来源，形成了多种输入方式，包括：文件输入、网络输入、控制台输入、交互界面输出、随机数据输入、内部参数输入等。

■ 处理数据

处理（Process）是程序对输入数据进行计算产生输出结果的过程。计算问题的处理方法统称为“算法”，它是程序最重要的组成部分。可以说，算法是一个程序的灵魂。

■ 输出数据

输出（Output）是程序展示运算成果的方式。程序的输出方式包括：控制台输出、图形输出、文件输出、网络输出、操作系统内部变量输出等。

输入/输出语句



程序的输入和输出

有些时候你的程序会与用户产生交互。举个例子，你会希望获取用户的输入内容，并向用户打印出一些返回的结果。可以分别通过 `input()` 函数与 `print()` 函数来实现这一需求。

input函数

input()函数的一般格式：

```
x = input('提示串')
```

x得到的是一个字符串。

```
>>> x = input('x=')    # 直接输入12.5, x是一个数字的字符串
```

```
>>> x
```

```
'12.5'
```

```
>>> x = input('x=')    # 直接输入abcd, x是字符串'abcd'
```

```
>>> x
```

```
'abcd'
```

```
>>> x = float(input('x='))
```

```
>>> x
```

```
123.77
```

print函数

print()函数的一般格式:

`print(对象1,对象2,...[,sep=' '][,end=' \n'][,file=sys.stdout])`

可以指定输出对象间的分隔符、结束标志符，输出文件。如果缺省这些，分隔符是空格，结束标志符是换行，输出目标是显示器。例如：

```
>>> print(1,2,3,sep="***",end='\n')
```

```
1***2***3
```

```
>>> print(1,2,3)
```

```
1 2 3
```

IPO程序编写方法

■ 圆面积的计算

输入：圆半径radius

处理：计算圆面积 $area = \pi * radius * radius$ （此处， π 取3.1415）

输出：圆面积area

```
radius = input('请输入圆的半径：R = ')
radius = int(radius)
area = 3.1415 * radius * radius
print("圆面积area = ", area)
```

让编程改变世界

Change the world by program