Chap03-操作符和表达式

- 3.0 概述
- 3.1 运算符
- 3.2 总结

3.0 概述

运算符示例	功能说明
x+y	算术加法，列表、元组、字符串合并
x-y	算术减法，集合差集
x*y	乘法，序列重复
x/y	除法（在Python 3.x中叫做真除法）
x//y	求整商
-x	相反数
x%y	余数（对实数也可以进行余数运算），字符串格式化
x**y	幂运算
x<y; x<=y; x>y; x>=y	大小比较（可以连用），集合的包含关系比较
x==y; x!=y	相等（值）比较，不等（值）比较
x or y	逻辑或（只有x为假才会计算y）
x and y	逻辑与(只有x为真才会计算y)
not x	逻辑非
x in y; x not in y	成员测试运算符
x is y; x is not y	对象实体同一性测试（地址）
|、^、&、<<、>>、~	位运算符
&、|、^	集合交集、并集、对称差集
@	矩阵相乘运算符

3.0 概述

- Python中的除法有两种，“/”和“//”分别表示除法和整除运算，并且Python 2.x和Python 3.x对“/”运算符的解释也略有区别。Python 2.x将“/”解释为普通除法，而Python 3.x将其解释为真除法。例如，在Python 3.5.1中运算结果如下：

```
>>> 3/5
0.6
>>> 3//5
0
>>> 3.0/5
0.6
>>> 3.0//5
0.0
>>> 13//10
1
>>> -13//10
-2
```

3.0 概述

•而上面的表达式在Python 2.7.11中运算结果如下：

```
>>> 3/5
0
>>> 3//5
0
>>> 3.0/5
0.6
>>> 3.0//5
0.0
>>> 13//10
1
>>> -13//10
-2
```

3.1 运算符

- 占位运算符
- 算术运算符
- 赋值运算符
- 比较运算符和逻辑运算符
- 位运算符
- 身份运算符和成员运算符
- 序列运算符和运算符优先级

占位运算符和算术运算符

占位运算符

运算符类似于C语言中sprintf或printf函数中使用的占位符，在字符串中可以给出一些占位符用来表示不同类型的数据，而实际的数据值在字符串之外给出。

占位符	描述	占位符	描述
%d	有符号整型十进制数	%s	字符串
%f或%F	有符号浮点型十进制数		

例如

- s1='%s上次数学成绩%d，本次%d，成绩提高%f ' % ('小明',85,90,5/85)
- s2='%5s上次数学成绩%5d，本次%5d，成绩提高%.2f ' % ('小明',85,90,5/85)
- s3='%5s上次数学成绩%05d，本次%05d，成绩提高%08.2f ' % ('小明',85,90,5/85)

执行完毕后，通过print函数分别输出s1、s2和s3，可得到下面结果：

- 小明上次数学成绩85，本次90，成绩提高0.058824
- 小明上次数学成绩 85，本次 90，成绩提高0.06
- 小明上次数学成绩00085，本次00090，成绩提高00000.06

提示

由于%作为占位符的前缀字符，因此对于有占位符的字符串，表示一个%时需要写成“%%“。例如，执行“print('优秀比例为%.2f%%，良好比例为%.2f%%。' %(5.2,20.35))“，输出结果为：优秀比例为5.20%，良好比例为20.35%。

算术运算符

算术运算是计算机支持的主要运算之一，其运算对象是数值型数据。

运算符	使用方法	功能描述
+（加）	x+y	x与y相加
-（减）	x-y	x与y相减
*（乘）	x*y	x与y相乘
/（除）	x/y	x除以y
//（整除）	x//y	x整除y，返回x/y的整数部分
%（模）	x%y	x整除y的余数，即x-x//y的值
-（负号）	-x	x的负数
+（正号）	+x	c
**（乘方）	x**y	x的y次幂

例如

```
1. i1,i2=10,3
2. f1,f2=3.2,1.5
3. c1,c2=3+4.1j,5.2+6.3j
4. print(i1+i2) #输出“13”
5. print(c1-c2) #输出“(-2.2-2.2j)”
6. print(f1*f2) #输出“4.800000000000001”
7. print(i1/i2) #输出“3.3333333333333335”
8. print(i1//i2) #输出“3”
9. print(i1%i2) #输出“1”
10. print(-f1) #输出“-3.2”
11. print(+f2) #输出“1.5”
12. print(i1**i2) #输出“1000”
```

提示

十进制小数在转换为二进制时有可能会产生精度损失，所以在第6行和第7行的输出中，结果与实际计算结果之间存在偏差，如f1（3.2）乘以f2（1.5）应该等于4.8，但最后输出的数据与实际计算结果存在0.000000000000001的偏差。

赋值运算符
比较运算符
和逻辑运算符

赋值运算符

赋值运算要求左操作数对象必须是值可以修改的变量。

运算符	使用方法	功能描述
=	y=x	将x的值赋给变量y
+=	y+=x	等价于y=y+x
-=	y-=x	等价于y=y-x
*=	y*=x	等价于y=y*x
/=	y/=x	等价于y=y/x
//=	y//=x	等价于y=y//x
%=	y%=x	等价于y=y%x
**=	y**=x	等价于y=y**x

例如

```
1. i1,i2=10,3 #i1和i2的值分别被赋为10和3
2. i1+=i2 #i1的值被改为13
3. print(i1) #输出“13”
4. c1,c2=3+4.1j,5.2+6.3j #c1和c2的值分别被赋为3+4.1j和
   5.2+6.3j
5. c1-=c2 #c1的值被改为-2.2-2.2j
6. print(c1) #输出“-2.2-2.2j”
7. f1,f2=3.2,1.5 #f1和f2的值分别被赋为3.2和1.5
8. f1*=f2 #f1的值被改为4.8
9. print(f1) #输出“4.8”
10. i1,f1=3,0.5 #i1和f1的值分别被赋为3和0.5
11. i1**=f1 #i1的值被改为1.7320508075688772（即3的0.5次
    幂）
12. print(i1) #输出“1.7320508075688772”
```

比较运算符

- 比较运算的作用是对两个操作数对象的大小关系进行判断。

运算符	使用方法	功能描述
==（等于）	y==x	如果y和x相等，则返回True；否则，返回False
!=（不等于）	y!=x	如果y和x不相等，则返回True；否则，返回False
>（大于）	y>x	如果y大于x，则返回True；否则，返回False
<（小于）	y<x	如果y小于x，则返回True；否则，返回False
>=（大于等于）	y>=x	如果y大于或等于x，则返回True；否则，返回False
<=（小于等于）	y<=x	如果y小于或等于x，则返回True；否则，返回False

例如

1. i1,i2,i3=25,35,25 #i1、i2和i3分别被赋为25、35和60
2. print(i1==i2) #输出“False”
3. print(i1!=i2) #输出“True”
4. print(i1>i3) #输出“False”
5. print(i1<i2) #输出“True”
6. print(i1>=i3) #输出“True”
7. print(i1<=i2) #输出“True”

提示

比较运算返回的结果是布尔值True或False。在执行程序时，程序中的每条语句并不一定是按顺序依次执行。比较运算的主要作用是设置条件，某些语句在满足条件时才会执行一次（即条件语句），而某些语句在满足条件时会重复执行多次（即循环语句）。

逻辑运算符

逻辑运算可以将多个比较运算连接起来形成更复杂的条件判断。

运算符	使用方法	功能描述
and	x and y	如果x和y都为Ture,则返回True;否则，返回Flase
or	x or y	如果x和y都为False,则返回False;否则，返回True
not	not x	如果x为True,则返回False; 如果x为False，返回True

例如：

```
1. n=80,a=100
2. print(n>=0 and n<=a) #输出“True”，判断n是否大于等于0且小于等于a
3. print(n<0 or n>a) #输出“False”，判断n是否小于0或大于a
4. print(not(n>=0 and n<=a)) #输出“False”
```

位运算符

十进制转二进制

除基取余法

用2去除十进制整数，得到商和余数；

如果商不为0，则继续用2除，再得到商和余数，重复该步骤直至商为0；

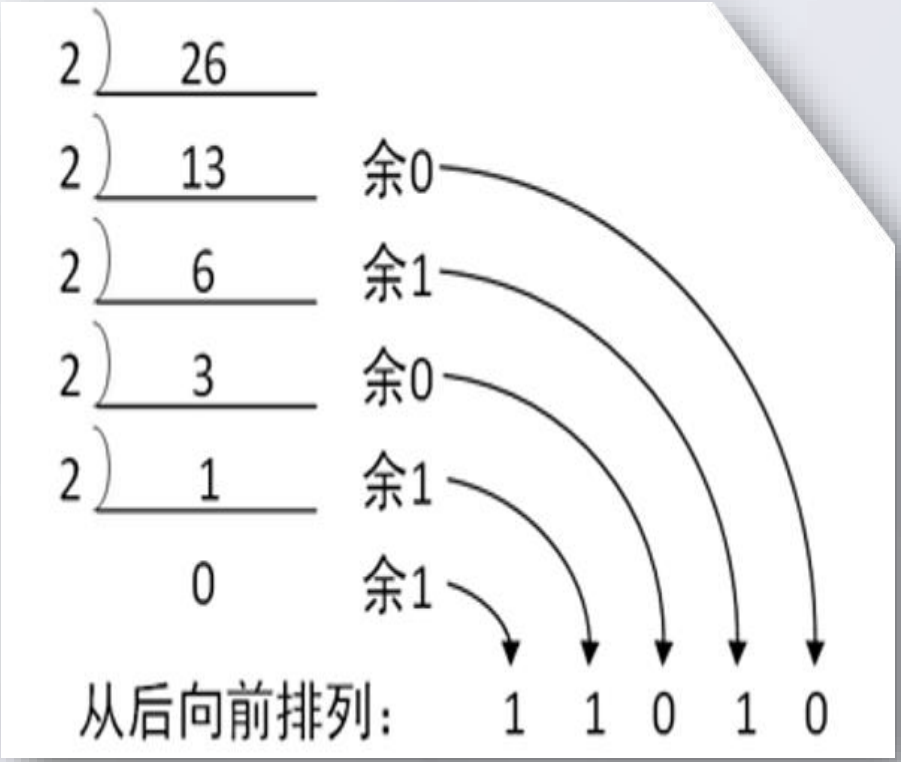
最后将余数按照从后至前的顺序排列，即得到转换后的二进制数。

提示

“除基取余法”中的“基”是指基数，基数即为一种数制中可用数码的个数。二进制可用的数码只有0和1两个，所以二进制的基数是2。

十进制转二进制

例如：



二进制转十进制

二进制数转十进制数的规则是“按权展开求和”，即将二进制数的每一位写成数码乘以位权的形式，再对乘积求和。

例如

对于二进制数11010B，其对应的十进制数为：

$11010B=1*2^4+1*2^3+0*2^2+1*2^1+0*2^0$

$=1*16+1*8+0*4+1*2+0*1$

$=16+8+0+2+0$

$=26$

位运算符

位运算是指对二进制数进行逐位运算。

运算符	使用方法	功能描述
&（按位与）	y&x	如果y和x对应位都为1，则结果中该位为1；否则，该位为0
|（按位或）	y|x	如果y和x对应位都为0，则结果中该位为0；否则，该位为1
^（按位异或）	y^x	如果y和x对应位不同，则结果中该位为1；否则，该位为0
<<（左移位）	y<<x	将y左移x位（右侧补0）
>>（右移位）	y>>x	将y右移x位（左侧补0）
~（按位取反）	~x	如果x的某位为1，则结果中该位为0；否则，该位为1

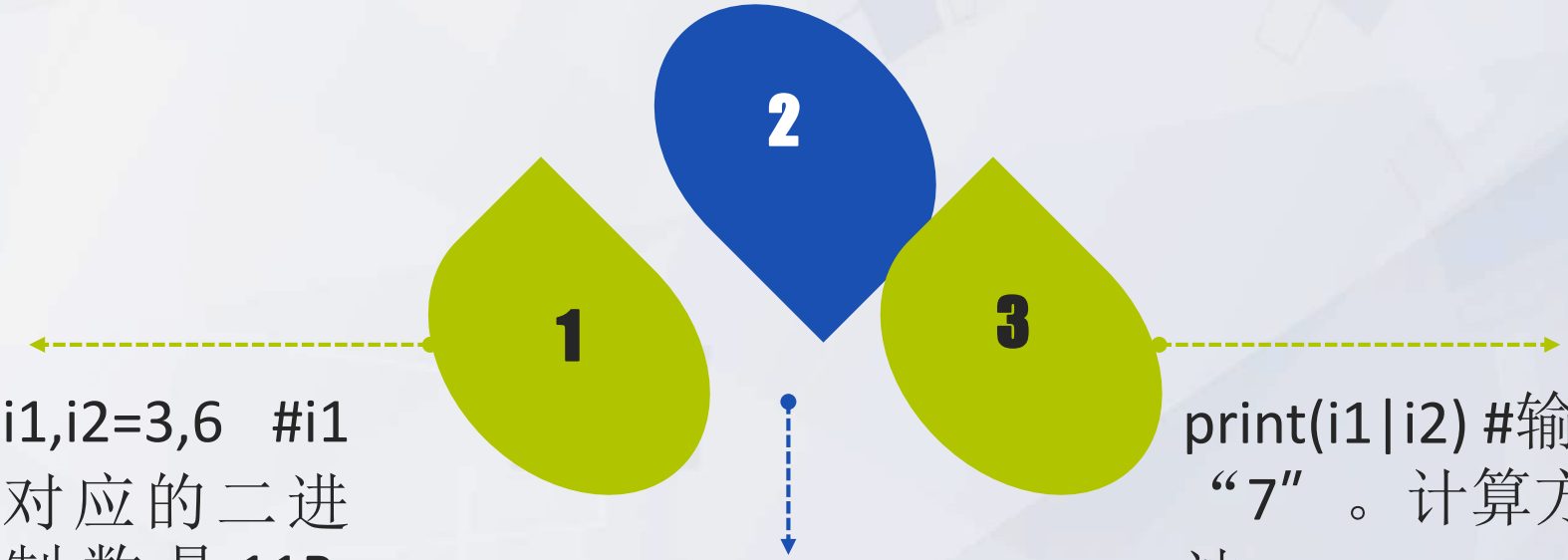
位运算示例

011B
&110B

010B

位运算示例

例如：



i1,i2=3,6 #i1对应的二进制数是11B，i2对应的二进制数是110B

print(i1&i2) #输出“2”。计算方法：011B&110B=010B=2

print(i1|i2) #输出“7”。计算方法：011B|110B=111B=7

位运算示例

4

print(i1^i2) #输出“5”。计算方法：
011B^110B=101B=5

5

print(i1<<1) #输出“6”。计算方法：
11B<<1=110B=6

6

print(i1>>1) #输出“1”。计算方法：
11B>>1=1B=1

身份运算符和成员运算符

身份运算符

身份运算用于比较两个对象是否对应同样的存储单元。

运算符	使用方法	功能描述
is	x is y	如果x和y对应同样的存储单元，则返回True；否则，返回False
is not	x is not y	如果x和y不对应同样的存储单元，则返回True；否则，返回False

程序在运行时，输入数据和输出数据都是存放在内存中。内存中的一个存储单元可以存储一个字节的数据，每个存储单元都有一个唯一的编号，称为内存地址。根据数据类型不同，其所占用的内存大小也不同。一个数据通常会占据内存中连续多个存储单元，起始存储单元的地址称为该数据的内存首地址。利用id函数可以查看一个数据的内存首地址。

x is y等价于id(x)==id(y)，即判断x和y的内存首地址是否相同；x is not y等价于id(x)!=id(y)，即判断x和y的内存首地址是否不相同。

身份运算符

例如：

```
1   x,y=15,15
2   print(x is y) #输出“True”
3   print(x is not y) #输出“False”
4   print(x is 15) #输出“True”
5   x,y=[1,2,3],[1,2,3]
6   print(x is y) #输出“False”
7   print(x==y) #输出“True”
8   print(x is [1,2,3]) #输出“False”
9   x=y
10  print(x is y) #输出“True”
```

提示：

如果赋值运算符“=”的右操作数也是一个变量，则赋值运算后左操作数变量和右操作数变量会对应同样的存储单元。

成员运算符

成员运算符

成员运算用于判断一个可迭代对象（序列、集合或字典）中是否包含某个元素。

运算符	使用方法	功能描述
in	x in y	如果x是可迭代对象y的一个元素，则返回True；否则，返回False
not in	x not in y	如果x不是可迭代对象y的一个元素，则返回True；否则，返回False

成员运算符

例如：

```
1    x,y=15,['abc',15,True]
2    print(x in y) #输出“True”
3    x=20
4    print(x not in y) #输出“True ”
5    y=(20,'Python')
6    print(x in y) #输出“True ”
7    x,y='Py','Python'
8    print(x in y) #输出“True ”
9    x,y=20,{15,20,25}
10   print(x in y) #输出“True ”
11   x,y='one',{'one':1,'two':2,'three':3}
12   print(x in y) #输出“True ”
13   print(1 in y) #输出“False ”
```

提示：

使用成员运算符判断一个数据是否是字典中的元素，实际上就是判断该数据是否是字典中某个元素的键。

序列运算符和运算符优先级

序列运算符

用于序列操作的运算符

运算符	使用方法	功能描述
+（拼接）	x+y	将序列x和序列y中的元素连接，生成一个新的序列
*（重复）	x*n	将序列x中的元素重复n次，生成一个新的序列

序列运算符

例如：

```
1   x,y=[12,False],['abc',15,True]
2   z=x+y #x和y拼接后的结果赋给z
3   print(z) #输出“[12, False, 'abc', 15, True]”
4   s1,s2='我喜欢学习','Python'
5   s=s1+s2 #s1和s2拼接后的结果赋给s
6   print(s) #输出“我喜欢学习Python”
7   x_3=x*3 #将序列x的元素重复3次，生成一个新序列并赋给x_3
8   print(x_3) #输出“[12, False, 12, False, 12, False] ”
9   s_3=s*3 #将字符串s重复3次，生成一个新字符串并赋给s_3
10  print(s_3) #输出“我喜欢学习Python我喜欢学习Python我喜欢学习Python”
```

条件运算符

Conditional Expression

运算符	使用方法	功能描述
[] if [] else []	20 if 2>1 else 30	Expression1 if Condition else Expression2

Example 1:>>> x if x > 0 else -x
Example 2:>>> m - 20 if m > 100 else m
Example 3:>>> m*0.8 if m > 100 else m

运算符优先级

在一个表达式中，通常会包含多个运算，这就涉及到了运算的顺序，其由两个因素确定：运算符的优先级和运算符的结合性。

优先级

- 对于具有不同优先级的运算符，会先完成高优先级的运算，再完成低优先级的运算。
- 例如，表达式3+5*6中，“*”优先级高于“+”，因此会先计算5*6，再计算3+30。

结合性

对于具有相同优先级的运算符，其运算顺序由结合性来决定。结合性包括左结合和右结合两种，左结合是按照从左向右的顺序完成计算，而右结合是按照从右向左的顺序完成计算。

例如，表达式5-3+6中，“-”和“+”优先级相同，它们是左结合的运算符，因此会先计算5-3，再计算2+6；表达式a=b=1中，“=”是右结合的运算符，因此会先计算b=1，再计算a=b。

优先级	运算符	描述
1	**	幂运算(乘方)
2	~、+、-	按位取反、正号、负号
3	*、/、//、%	乘/序列重复、除、整除、模
4	+、-	加/序列连接、减
5	>>、<<	右移位、左移位
6	&	按位与
7	^	按位异或
8	|	按位或
9	>、<、>=、<=、==、!=、is、is not、in、not in	比较运算符、身份运算符、成员运算符
10	not	逻辑非
11	and	逻辑与
12	or	逻辑或
13	If else	条件运算符
14	=、+=、-=、*=、/=、//=、%=、**=	赋值运算符

3.2总结

- 另外一个需要说明的，也是与其他有些语言略有不同的运算符是“%”。在Python中，除去前面已经介绍过的字符串格式化用法之外，该运算符还可以对整数和浮点数计算余数。但是由于浮点数的精确度影响，计算结果可能略有误差。

```
>>> 3.1%2
1.1
>>> 6.3%2.1
2.0999999999999996
>>> 6%2
0
>>> 6.0%2
0.0
>>> 6.0%2.0
0.0
>>> 5.7%4.8
0.9000000000000004
```

3.2总结

- Python中很多运算符有多重含义，在程序中运算符的具体含义取决于操作数的类型，将在后面章节中根据内容组织的需要陆续进行展开。例如“*”运算符就是Python运算符中比较特殊的一个，它不仅可以用于数值乘法，还可以用于列表、字符串、元组等类型，当列表、字符串或元组等类型变量与整数进行“*”运算时，表示对内容进行重复并返回重复后的新对象。

```
>>> 3*2 #整数相乘
6
>>> 2.0*3 #浮点数与整数相乘
6.0
>>> (3+4j)*2 #复数与整数相乘
(6+8j)
>>> (3+4j)*(3-4j) #复数与复数相乘
(25+0j)
>>> "a"*10 #字符串重复
'aaaaaaaaaa'
>>> [1,2,3]*3 #列表重复
[1, 2, 3, 1, 2, 3, 1, 2, 3]
>>> (1,2,3)*3 #元组重复
(1, 2, 3, 1, 2, 3, 1, 2, 3)
```

3.2总结

- 在Python中，单个任何类型的对象或常数属于合法表达式，使用表1-5中运算符连接的变量和常量以及函数调用的任意组合也属于合法的表达式。

```
>>> a = [1,2,3]
>>> b = [4,5,6]
>>> c = a+b
>>> c
[1, 2, 3, 4, 5, 6]
>>> d = list(map(str, c))
>>> d
['1', '2', '3', '4', '5', '6']
>>> import math
>>> list(map(math.sin, c))
[0.8414709848078965, 0.9092974268256817, 0.1411200080598672, -
0.7568024953079282, -0.9589242746631385, -0.2794154981989258 6]
>>> 'Hello' + ' ' + 'world'
'Hello world'
>>> 'welcome ' * 3
'welcome welcome welcome '
>>> ('welcome,'*3).rstrip(',')+'!'
'welcome,welcome,welcome!'
```

3.2总结

■ 在Python中逗号“,”并不是运算符，而只是一个普通分隔符。

```
>>> 'a' in 'b', 'a'
(False, 'a')
>>> 'a' in ('b', 'a')
True
>>> x = 3, 5
>>> x
(3, 5)
>>> 3 == 3, 5
(True, 5)
>>> x = 3+5, 7
>>> x
(8, 7)
```