Python运算符及代码举例

运算这一概念起源于数学，即通过运算法使参与运算的元素得出确定且可重复的结果。作为计算机的核心功能，运算架构起计算机系统的逻辑体系。计算机运算并不局限于普通的数学计算，它更贴近于“逻辑推算”这一概念，其根本目的就是实现逻辑推算。

运算符是运算法则的具体体现。Python提供了算术运算符、赋值运算符、比较运算符、逻辑运算符、 按位运算符、身份运算符和成员运算符7类运算符，从而实现了丰富多样的运算功能。

今天给大家详解讲解一下算术运算符、赋值运算符、比较运算符、逻辑运算符这几类~

01 算术运算符

算术运算符是对运算数进行算术运算的一系列符号，能够满足一般的运算需求。Python中的算术运算符如下所示。

- +：加，两个对象相加
- –：减，得到负数或一个数减去另一个数
- *：乘，两个数相乘或返回一个被重复若干次的字符串
- /：除，x除以y

扩展：

- %：取余，返回除法的余数
- **：幂，返回x的y次幂
- //：取整除，返回商的整数部分

算术运算结果的数字类型与运算数的类型有关。进行除法（/）运算时，不管商为整数还是浮点数，运算结果始终为浮点数。要得到整型的商，需要用双斜杠（//）做整除，且除数必须是整型的。对于其他的运算，只要任一运算数为浮点数，运算结果就是浮点数。Python算术运算的基础使用方法如下所示。

```
num_int = 4
num_float = 4.0
```

```
print('整数与浮点数的和为：', num_int + num_float)
#Out[1]: 整数与浮点数的和为： 8.0
print('整数与浮点数的差为：', num_int - num_float)
#Out[2]: 整数与浮点数的差为： 0.0
print('整数与浮点数的积为：', num_int * num_float)
#Out[3]: 整数与浮点数的积为： 16.0
print('浮点数与整数的商为：', num_float / num_int)
#Out[4]: 浮点数与整数的商为： 1.0
print('浮点数对整数取模结果为：', num_float % num_int)
#Out[5]: 浮点数对整数取模结果为： 0.0
print('浮点数的整数次幂为：', num_float ** num_int)
#Out[6]: 浮点数的整数次幂为： 256.0
```

02 赋值运算符

赋值运算符用于变量的赋值和更新。Python的赋值运算符除基础赋值运算符（=）外，还包括加法赋值运算符、减法赋值运算符等。严格地说，除基础赋值运算符外，其他都属于特殊的赋值运算符。Python中的赋值运算符如下所示。

- =：赋值运算
- +=：加法赋值运算
- -=：减法赋值运算
- *=：乘法赋值运算
- /=：除法赋值运算
- %=：取模赋值运算
- **=：幂赋值运算
- //=：取整除赋值运算

Python赋值运算的基础使用方法如下所示。

```
num_int1 = 4
print('赋值后num_int1为：', num_int1)
#Out[7]: 赋值后num_int1为： 4
num_int1 = 4 + 6
print('赋值后num_int1为：', num_int1)
#Out[8]: 赋值后num_int1为： 10
num_int1 = 4 * 2
print('赋值后num_int1为：', num_int1)
#Out[9]: 赋值后num_int1为： 8
num_int1 = 4 / 2
print('赋值后num_int1为：', num_int1)
#Out[10]: 赋值后num_int1为： 2.0
```

```
num_int1 = 4 % 2
print('赋值后num_int1为：', num_int1)
#Out[11]: 赋值后num_int1为： 0
num_int1 = 4 ** 2
print('赋值后num_int1为：', num_int1)
#Out[12]: 赋值后num_int1为： 16
```

03 比较运算符

比较运算符用于对比数之间的大小或是否相等。Python中的比较运算符如下所示。

- ==：表示等于，比较对象是否相等
- !=：表示不等于，比较两个对象是否不等
- >：表示大于，返回x是否大于y
- <：表示小于，返回x是否小于y。所有比较运算符返回1表示真，返回0表示假。这分别与特殊的变量True和False等价。注意，这些变量名的首字母大写
- >=：表示大于等于，返回x是否大于等于y
- <=：表示小于等于，返回x是否小于等于y

比较运算符也可用于字符之间的比较。Python中的字符使用ASCII编码，每个字符都有属于自己的ASCII码，字符比较的本质是字符ASCII码的比较。Python比较运算的基础使用方法如下所示。

```
num_int = 4
num_float = 4.0
print('num_int与num_float是否相等：', num_int == num_float)
#Out[13]: num_int与num_float是否相等： True
print('num_int与num_float是否不相等：', num_int != num_float)
#Out[14]: num_int与num_float是否不等： False
print('num_int是否大于num_float：', num_int > num_float)
#Out[15]: num_int是否大于num_float： False
print('num_int是否小于num_float：', num_int < num_float)
#Out[16]: num_int是否小于num_float： False
print('num_int是否大于等于numfloat：', num_int >= num_float)
#Out[17]: num_int是否大于等于numfloat： True
print('num_int是否小于等于num_float：', num_int <= num_float)
#Out[18]: num_int是否小于等于num_float： True
```

04 逻辑运算符

逻辑运算即判断事物之间的“与”“或”“非”关系，Python中的逻辑运算符包含and、or、not，如下所示。

- and，x and y：表示与，x为False时，“x and y”返回False，否则返回y的计算值
- or，x or y：表示或，x为True时，“x or y”返回x的值，否则返回y的计算值
- not，not x：表示非，x为True时，“not x”返回False，否则返回True

Python逻辑运算的基础使用方法如下所示。

```
num_bool1 = False
num_bool2 = True
print('num_bool1 and num_bool2返回值为：', num_bool1 and num_bool2)
#Out[19]: num_bool1 and num_bool2返回值为： False
print('num_bool1 or num_bool2返回值为：', num_bool1 or num_bool2)
#Out[20]: num_bool1 or num_bool2返回值为： True
print('not num_bool2的返回值为：', not (num_bool2))
#Out[21]: not num_bool2的返回值为： False
```