# Python实验3 运算符和表达式

## 一、实验介绍

在 Python 中你会写大量的表达式。

表达式由运算符和操作数组成，像 2+3 就是一个表达式，其中 2 和 3 是操作数，加号是运算符。

本节实验中我们将学习运算符和表达式的基本用法。

知识点

关系/逻辑运算

表达式

类型转换

## 二、运算符

运算符是一些符号，它告诉 Python 解释器去做一些数学或逻辑操作。一些基本的数学操作符如下所示：

>>> 2 + 3

5

>>> 23.0 - 3

20.0

>>> 22 / 12

1.8333333333333333

只要有任意一个操作数是浮点数，结果就会是浮点数。

进行除法运算时若是除不尽，结果将会是小数，这很自然，如果要进行整除，使用 // 运算符，它将返回商的整数部分。

% 是求余运算符：

>>> 14 % 3

2

2.1 整数运算示例

整数运算符比较容易理解，代码如下：

#!/usr/bin/env python3

days = int(input("Enter days: "))

months = days // 30

days = days % 30

print("Months = {} Days = {}".format(months, days))

运行程序：

在开始获得用户输入的天数，然后获得月份数和天数，最后把这些数打印出来。你可以使用更容易的办法。

#!/usr/bin/env python3

days = int(input("Enter days: "))

print("Months = {} Days = {}".format(\*divmod(days, 30)))

divmod(num1, num2) 返回一个元组，这个元组包含两个值，第一个是 num1 和 num2 相整除得到的值，第二个是 num1 和 num2 求余得到的值，然后我们用 \* 运算符拆封这个元组，得到这两个值。

2.2 关系运算符

你可以使用下面的运算符实现关系运算。

关系运算符

Operator Meaning

< Is less than

<= Is less than or equal to

> Is greater than

>= Is greater than or equal to

== Is equal to

!= Is not equal to

举一些例子：

>>> 1 < 2

True

>>> 3 > 34

False

>>> 23 == 45

False

>>> 34 != 323

True

2.3 逻辑运算符

对于逻辑 与，或，非，我们使用 and，or，not 这几个关键字。

逻辑运算符 and 和 or 也称作短路运算符：它们的参数从左向右解析，一旦结果可以确定就停止。例如，如果 A 和 C 为真而 B 为假，A and B and C 不会解析 C 。作用于一个普通的非逻辑值时，短路运算符的返回值通常是能够最先确定结果的那个操作数。

关系运算可以通过逻辑运算符 and 和 or 组合，比较的结果可以用 not 来取反意。逻辑运算符的优先级又低于关系运算符，在它们之中，not 具有最高的优先级，or 优先级最低，所以 A and not B or C 等于 (A and (notB)) or C。当然，括号也可以用于比较表达式。

下面是一些例子：

>>> 5 and 4

4

>>> 0 and 4

0

>>> False or 3 or 0

3

>>> 2 > 1 and not 3 > 5 or 4

True

2.4 简写运算符

x op= expression 为简写运算的语法形式。其等价于 x = x op expression ，举例如下：

>>> a = 12

>>> a += 13

>>> a

25

>>> a /= 3

>>> a

8.333333333333334

>>> a += (26 \* 32)

>>> a

840.3333333333334

/home/shiyanlou/shorthand.py 示例：

#!/usr/bin/env python3

N = 100

a = 2

while a < N:

print(str(a))

a \*= a

运行程序：

$ cd /home/shiyanlou

$ chmod +x shorthand.py

$ ./shorthand.py

2

4

16

2.5 表达式

通常我们书写表达式的时候，会在每一个运算符左右都放一个空格，这样使代码更可读，如：

a = 234 \* (45 - 56 / 34)

一个用于展示表达式的例子，注意其中运算符的优先级。

#!/usr/bin/env python3

a = 9

b = 12

c = 3

x = a - b / 3 + c \* 2 - 1

y = a - b / (3 + c) \* (2 - 1)

z = a - (b / (3 + c) \* 2) - 1

print("X = ", x)

print("Y = ", y)

print("Z = ", z)

运行程序：

$ cd /home/shiyanlou

$ chmod +x evaluationexp.py

$ ./evaluationexp.py

X = 10.0

Y = 7.0

Z = 4.0

第一个计算的是 x，步骤如下：

9 - 12 / 3 + 3 \* 2 -1

9 - 4 + 3 \* 2 - 1

9 - 4 + 6 - 1

5 + 6 - 1

11 - 1

10

由于括号的存在，y 和 z 的计算方式不同，你可以自己去验证它们。

2.6 类型转换

我们可以手动的执行类型转换。

类型转换函数 转换路径

float(string) 字符串 -> 浮点值

int(string) 字符串 -> 整数值

str(integer) 整数值 -> 字符串

str(float) 浮点值 -> 字符串

>>> a = 8.126768

>>> str(a)

'8.126768'

可以分别尝试下前面的四个类型转换函数。

2.7 程序示例

2.7.1. evaluateequ.py

这个程序计算数列 1/x+1/(x+1)+1/(x+2)+ ... +1/n，我们设 x = 1，n = 10。

#!/usr/bin/env python3

sum = 0

for i in range(1, 11):

sum += 1.0 / i

print("{:2d} {:6.4f}".format(i , sum))

运行程序：

2.7.2. quadraticequation.py

这个程序用来求解二次方程式：

#!/usr/bin/env python3

import math

a = int(input("Enter value of a: "))

b = int(input("Enter value of b: "))

c = int(input("Enter value of c: "))

d = b \* b - 4 \* a \* c

if d < 0:

print("ROOTS are imaginary")

else:

root1 = (-b + math.sqrt(d)) / (2 \* a)

root2 = (-b - math.sqrt(d)) / (2 \* a)

print("Root 1 = ", root1)

print("Root 2 = ", root2)

运行程序：

2.7.3. salesmansalary.py

这个程序计算一位数码相机销售人员的工资。他的基本工资是 1500，每售出一台相机他可以得到 200 并且获得 2% 的抽成。程序要求输入相机数量及单价。

#!/usr/bin/env python3

basic\_salary = 1500

bonus\_rate = 200

commision\_rate = 0.02

numberofcamera = int(input("Enter the number of inputs sold: "))

price = float(input("Enter the total prices: "))

bonus = (bonus\_rate \* numberofcamera)

commision = (commision\_rate \* numberofcamera \* price)

print("Bonus = {:6.2f}".format(bonus))

print("Commision = {:6.2f}".format(commision))

print("Gross salary = {:6.2f}".format(basic\_salary + bonus + commision))

运行程序：

## 三、总结

本节实验知识点回顾：

关系/逻辑运算

表达式

类型转换

除了数值运算，关系和逻辑运算也是程序的重要组成部分。另外 Python 是强类型语言，所以必要的时候需要手动进行类型转换。