Python语言中的运算符

Python 语言支持很多种运算符，下面的表格按照运算符的优先级从高到低，对 Python 中的运算符进行了罗列。有了变量和运算符，我们就可以构造各种各样的表达式来解决实际问题。在计算机科学中，**表达式是计算机程序中的句法实体，它由一个或多个常量、变量、函数和运算符组合而成，编程语言可以对其进行解释和计算以得到另一个值**。不理解这句话没有关系，但是一定要知道，不管使用什么样的编程语言，构造表达式都是非常重要的。

运算符	描述
`[]`、`[:]`	索引、切片
`**`	幂
`~`、`+`、`-`	按位取反、正号、负号
`*`、`/`、`%`、`//`	乘、除、模、整除
`+`、`-`	加、减
`>>`、`<<`	右移、左移
`&`	按位与
`^`、`|`	|
`<=`、`<`、`>`、`>=`	小于等于、小于、大于、大于等于
`==`、`!=`	等于、不等于
`is`、`is not`	身份运算符
`in`、`not in`	成员运算符
`not`、`or`、`and`	逻辑运算符
`=`、`+=`、`-=`、`*=`、`/=`、`%=`、`//=`、`**=`、`&=`、`|=`、`^=`、`>>=`、`<<=`	赋值运算符

> **说明**：所谓优先级就是在一个运算的表达式中，如果出现了多个运算符，应该先执行什么再执行什么的顺序。编写代码的时候，如果搞不清楚一个表达式中运算符的优先级，可以使用圆括号（小括号）来确保运算的执行顺序。

算术运算符

Python 中的算术运算符非常丰富，除了大家最为熟悉的加、减、乘、除之外，还有整除运算符、求模（求余数）运算符和求幂运算符。下面的例子为大家展示了算术运算符的使用。

```
"""
算术运算符

Version: 1.0
Author: 骆昊
"""
print(321 + 12)     # 加法运算，输出333
print(321 - 12)     # 减法运算，输出309
print(321 * 12)     # 乘法运算，输出3852
print(321 / 12)     # 除法运算，输出26.75
print(321 // 12)    # 整除运算，输出26
print(321 % 12)     # 求模运算，输出9
print(321 ** 12)    # 求幂运算，输出119690695022892891542061732241
```

算术运算需要先乘除后加减，这一点跟数学课本中讲的知识没有区别，也就是说乘除法的运算优先级是高于加减法的。如果还有求幂运算，求幂运算的优先级是高于乘除法的。如果想改变算术运算的执行顺序，可以使用英文输入法状态下的圆括号（小括号），写在圆括号中的表达式会被优先执行，如下面的例子所示。

```
"""
算术运算的优先级

Version: 1.0
Author: 骆昊
"""
print(2 + 3 * 5)           # 17
print((2 + 3) * 5)         # 25
print((2 + 3) * 5 ** 2)    # 125
print(((2 + 3) * 5) ** 2)  # 625
```

赋值运算符

赋值运算符应该是最为常见的运算符，它的作用是将右边的值赋给左边的变量。赋值运算符还可以跟上面的算术运算符放在一起，组合成复合赋值运算符，例如： `a += b` 相当于 `a = a + b` ， `a *= a + 2` 相当于 `a = a * (a + 2)` 。下面的例子演示了赋值运算符和复合赋值运算符的使用。

```
"""
赋值运算符和复合赋值运算符

Version: 1.0
Author: 骆昊
"""
a = 10
b = 3
a += b        # 相当于：a = a + b
a *= a + 2    # 相当于：a = a * (a + 2)
print(a)      # 大家算一下这里会输出什么
```

赋值运算构成的表达式本身不产生任何值，也就是说，如果你把一个赋值表达式放到 `print` 函数中试图输出表达式的值，将会产生语法错误。为了解决这个问题，Python 3.8 中引入了一个新的赋值运算符 `:=` ，我们称之为海象运算符，大家可以猜一猜它为什么叫这个名字。海象运算符也是将运算符右侧的值赋值给左边的变量，与赋值运算符不同的是，运算符右侧的值也是整个表达式的值，看看下面的代码大家就明白了。

```
"""
海象运算符

Version: 1.0
Author: 骆昊
"""
# SyntaxError: invalid syntax
# print((a = 10))
# 海象运算符
print((a := 10))  # 10
print(a)          # 10
```

> **提示**：上面第 8 行代码如果不注释掉，运行代码会看到 `SyntaxError: invalid syntax` 错误信息，注意，这行代码中我们给 `a = 10` 加上了圆括号，如果不小心写成了 `print(a = 10)` ，会看到 `TypeError: 'a' is an invalid keyword argument for print()` 错误信息，后面讲到函数的时候，大家就会明白这个错误提示是什么意思了。

比较运算符和逻辑运算符

比较运算符也称为关系运算符，包括 `==` 、 `!=` 、 `<` 、 `>` 、 `<=` 、 `>=` ，我相信大家一看就能懂。需要提醒的是比较相等用的是 `==` ，请注意这里是两个等号，因为 `=` 是赋值运算符，我们在上面刚刚讲到过。比较不相等用的是 `!=` ，跟数学课本中使用的≠并不相同，Python 2 中曾经使用过 `<>` 来表示不等于，在 Python 3 中使用 `<>` 会引发 `SyntaxError` （语法错误）。比较运算符会产生布尔值，要么是 `True` ，要么是 `False` 。

逻辑运算符有三个，分别是 `and` 、 `or` 和 `not` 。 `and` 字面意思是“而且”，所以 `and` 运算符会连接两个布尔值或者产生布尔值的表达式，如果两边的布尔值都是 `True` ，那么运算的结果就是 `True` ；左右两边的布尔值有一个是 `False` ，最终的运算结果就是 `False` 。当然，如果 `and` 运算符左边的布尔值是 `False` ，不管右边的布尔值是什么，最终的结果都是 `False` ，这时运算符右边的布尔值会被跳过（专业的说法叫短路处理，如果 `and` 右边是一个表达式，那么这个表达式不会执行）。 `or` 字面意思是“或者”，所以 `or` 运算符也会连接两个布尔值或产生布尔值的表达式，如果两边的布尔值有任意一个是 `True` ，那么最终的结果就是 `True` 。当然， `or` 运算符也是有短路功能的，当它左边的布尔值为 `True` 的情况下，右边的布尔值会被短路（如果 `or` 右边是一个表达式，那么这个表达式不会执行）。 `not` 运算符的后面可以跟一个布尔值，如果 `not` 后面的布尔值或表达式是 `True` ，那么运算的结果就是 `False` ；如果 `not` 后面的布尔值或表达式是 `False` ，那么运算的结果就是 `True` 。

```
"""
比较运算符和逻辑运算符的使用

Version: 1.0
Author: 骆昊
"""
flag0 = 1 == 1
flag1 = 3 > 2
flag2 = 2 < 1
flag3 = flag1 and flag2
flag4 = flag1 or flag2
flag5 = not flag0
print('flag0 =', flag0)     # flag0 = True
print('flag1 =', flag1)     # flag1 = True
print('flag2 =', flag2)     # flag2 = False
print('flag3 =', flag3)     # flag3 = False
print('flag4 =', flag4)     # flag4 = True
print('flag5 =', flag5)     # flag5 = False
print(flag1 and not flag2)  # True
print(1 > 2 or 2 == 3)      # False
```

> **说明**：比较运算符的优先级高于赋值运算符，所以上面的 `flag0 = 1 == 1` 先做 `1 == 1` 产生布尔值 `True` ，再将这个值赋值给变量 `flag0` 。 `print` 函数可以输出多个值，多个值之间可以用 `,` 进行分隔，输出的内容默认以空格分开。

运算符和表达式应用

例子1：华氏温度转摄氏温度

> **要求**：输入华氏温度将其转换为摄氏温度，华氏温度到摄氏温度的转换公式为： $C = (F - 32) / 1.8$ 。

```
"""
将华氏温度转换为摄氏温度

Version: 1.0
Author: 骆昊
"""
f = float(input('请输入华氏温度: '))
c = (f - 32) / 1.8
print('%.1f华氏度 = %.1f摄氏度' % (f, c))
```

> **说明**：上面代码中的 `input` 函数用于从键盘接收用户输入，由于输入的都是字符串，如果想处理成浮点小数来做后续的运算，可以用我们上一课讲解的类型转换的方法，用 `float` 函数将 `str` 类型处理成 `float` 类型。

上面的代码中，我们对 `print` 函数输出的内容进行了格式化处理， `print` 输出的字符串中有两个 `%.1f` 占位符，这两个占位符会被 `%` 之后的 `(f, c)` 中的两个 `float` 类型的变量值给替换掉，浮点数小数点后保留1位有效数字。如果字符串中有 `%d` 占位符，那么我们会用 `int` 类型的值替换掉它，如果字符串中有 `%s` 占位符，那么它会被 `str` 类型的值替换掉。

除了上面格式化输出的方式外，Python 中还可以用下面的办法来格式化输出，我们给出一个带占位符的字符串，字符串前面的 `f` 表示这个字符串是需要格式化处理的，其中的 `{f:.1f}` 和 `{c:.1f}` 可以先看成是 `{f}` 和 `{c}` ，表示输出时会用变量 `f` 和变量 `c` 的值替换掉这两个占位符，后面的 `:.1f` 表示这是一个浮点数，小数点后保留1位有效数字。

```
"""
将华氏温度转换为摄氏温度

Version: 1.1
Author: 骆昊
"""
f = float(input('请输入华氏温度: '))
c = (f - 32) / 1.8
print(f'{f:.1f}华氏度 = {c:.1f}摄氏度')
```

例子2：计算圆的周长和面积

> **要求**：输入一个圆的半径（r），计算出它的周长（ $2\pi r$ ）和面积（ πr^2 ）。

```
"""
输入半径计算圆的周长和面积

Version: 1.0
Author: 骆昊
"""
radius = float(input('请输入圆的半径: '))
perimeter = 2 * 3.1416 * radius
area = 3.1416 * radius * radius
print('周长: %.2f' % perimeter)
print('面积: %.2f' % area)
```

Python 中有一个名为 `math` 的内置模块，该模块中定义了名为 `pi` 的变量，它的值就是圆周率。如果要使用 Python 内置的这个 `pi` ，我们可以对上面的代码稍作修改。

```
"""
输入半径计算圆的周长和面积

Version: 1.1
Author: 骆昊
"""
import math

radius = float(input('请输入圆的半径: '))
perimeter = 2 * math.pi * radius
area = math.pi * radius ** 2
print(f'周长: {perimeter:.2f}')
print(f'面积: {area:.2f}')
```

> **说明**：上面代码中的 `import math` 表示导入 `math` 模块，导入该模块以后，才能用 `math.pi` 得到圆周率的值。

这里其实还有一种格式化输出的方式，是 Python 3.8 中增加的新特性，大家直接看下面的代码就明白了。

```
"""
输入半径计算圆的周长和面积

Version: 1.2
Author: 骆昊
"""
import math

radius = float(input('请输入圆的半径: '))  # 输入: 5.5
perimeter = 2 * math.pi * radius
area = math.pi * radius ** 2
print(f'{perimeter = :.2f}')  # 输出: perimeter = 34.56
print(f'{area = :.2f}')       # 输出: area = 95.03
```

> **说明**：假如变量 `a` 的值是 `9.87` ，那么字符串 `f'{a = }'` 的值是 `a = 9.87` ；而字符串 `f'{a = :.1f}'` 的值是 `a = 9.9` 。这种格式化输出的方式会同时输出变量名和变量值。

例子3：判断闰年

要求：输入一个 1582 年以后的年份，判断该年份是不是闰年。

```
"""
输入年份，闰年输出True，平年输出False

Version: 1.0
Author: 骆昊
"""
year = int(input('请输入年份: '))
is_leap = year % 4 == 0 and year % 100 != 0 or year % 400 == 0
print(f'{is_leap = }')
```

> **说明**：对于格里历（Gregorian calendar），即今天我们使用的公历，判断闰年的规则是：1. 公元年份非 4 的倍数是平年；2. 公元年份为 4 的倍数但非 100 的倍数是闰年；3. 公元年份为 400 的倍数是闰年。格里历是由教皇格里高利十三世在 1582 年 10 月引入的，作为对儒略历（Julian calendar）的修改和替代，我们在输入年份时要注意这一点。上面的代码通过 `%` 来判断 `year` 是不是 `4` 的倍数、 `100` 的倍数、 `400` 的倍数，然后用 `and` 和 `or` 运算符将三个条件组装在一起，前两个条件要同时满足，第三个条件跟前两个条件的组合只需满足其中之一。

总结

通过上面的讲解和例子，相信大家已经感受到了运算符和表达式的力量。实际编程中的很多问题，都需通过构造表达式来解决，所以变量、运算符、表达式对于任何一门编程语言都是极为重要的基础。如果本节课的内容有什么不理解的地方，一定不要着急进入下一课，先在评论区留言讨论，我会及时解答大家的问题。