第3章 数值计算

3.1 数值数据类型

计算机刚研发出来时，它们主要被视为数字处理器，也就是协助科学家完成科学计算的工作，现在这仍然是一个重要的应用。正如你所见，涉及数学公式的问题很容易转化为Python程序。在本章中，我们将仔细考查一些程序，这些程序的目的是执行数值计算。计算机程序存储和操作的信息通常称为“数据”。不同类型的数据以不同的方式存储和操作。比如下面这个计算零钱的程序：

```
#3_1 change.py
#A program to calculate the value of some change in dollars
def main():
    print("Change Counter")
    print()
    print("请输入你的各种硬币个数.")
    yuan = eval(input("有多少 1 元的硬币: "))
    fifty_cents = eval(input("有多少 5 角的硬币: "))
    twenty_cents = eval(input("有多少 2 角的硬币: "))
    ten_cents = eval(input("有多少 1 角的硬币: "))
    total =yuan * 1.0 + fifty_cents * .50 + twenty_cents * .20 + ten_cents * .10
    print()
    print("你拥有的硬币总额是", total)

main()
```

下面是输出示例：

```
Change Counter

请输入你的各种硬币个数.
有多少 1 元的硬币: 5
有多少 5 角的硬币: 3
有多少 2 角的硬币: 4
```

```
有多少 1 角的硬币：6

你拥有的硬币总额是 7.9
```

这个程序实际上操作两种不同的数字。用户输入的值(5，3，4，6)是整数，它们没有任何小数部分。硬币的值(1.0，.50，.20，.10)是分数的十进制表示。在计算机内部，整数和具有小数部分的数值的存储方式是不同的，从技术上讲，这是两种不同的“数据类型”。

对象的数据类型决定了它可以具有的值以及可以对它执行的操作。整数用 int 表示，int 型的值可以是正数和负数。具有小数部分的数字为浮点(float)型。当需要判断数字是什么数据类型时，可以使用 Python 内置的 type()函数，如下所示。

```
>>> type(5)
<class 'int'>
>>> type(5.0)
<class 'float'>
>>> a = 5
>>> type(a)
<class 'int'>
>>> a = 5.0
>>> type(a)
<class 'float'>
```

从上面可以看到两种数据类型：int 型和 float 型。那为什么会有多种数据类型呢？其中的一个原因是程序风格，如 int 型表示的数据不能带有小数部分，比如不能说有 3.8 只小动物等；另一个原因是各种数据类型的操作效率问题，如对 int 型的处理，计算机一般比较快。当然，当今的处理器，CPU 的浮点运算是高度优化的，可能与 int 型运算一样快。

int 型和 float 型之间的另一个区别是，float 型可以表示数学中的实数。我们会看到，存储值的精度(或准确度)存在限制。由于 float 型不精确，而 int 型总是精确的，所以一般的经验法则应该是：如果不需要小数值，就使用 int 型。

当然，Python 除了支持 int 型和 float 型，还支持长整(long)型和复数(complex)型。表 3-1 列出了一些数值类型的示例。

表 3-1　一些数值类型的示例

int	long	float	complex
10	519243964254752	0.0	3.14j
−10	−47218852996548	3.21	3j
0o56(八进制)	0o57345422342332 51(八进制)	56.25	9.322e−32j
−0o56(八进制)	−0o573454223433251(八进制)	−32.54e100	3e+23j
0x10(十六进制)	0xDEFABCECBDAECBFBAE		
−0x10(十六进制)	−0x19323		

```
>>> i = 10
>>> i
10
>>> a = 0o56
>>> a
46
>>> b = 0x10
>>> b
16
>>> c = 519243964254752                #十进制长整型
>>> c
519243964254752
>>> d = 0o563454223423452232 51        #八进制长整型
>>> d
836738412174452393
>>> e = 0xDEFABCECBDAECBFBAE           #十六进制长整型
>>> e
4113244760468049623982
>>> f = -32.54e100                     #浮点数
>>> f
-3.254e+101
>>> g = 3e+23j                         #复数 3e 为实部,23j 为虚部
>>> g
3e+23j
```

值的数据类型决定了可以进行什么样的操作。Python 支持对数值的一般数学运算。表 3-2 总结了这些操作。

表 3-2　Python 内置的数值操作

操　作　符	操　　作	操　作　符	操　　作
＋	加	**	指数
－	减	abs()	绝对值
*	乘	//	整数除
/	浮点除	%	取余

看看如下的 Python 交互：

```
>>> 3 + 4
7
>>> 3.0 + 4.0
7.0
>>> 3 * 4
```

```
12
>>> 3.0 * 4.0
12.0
>>> 4 ** 3
64
>>> 4.0 ** 3
64.0
>>> 4.0 ** 3.0
64.0
>>> abs(5)
5
>>> abs(-3.5)
3.5
```

在大多数情况下，程序员不必明确操作的是什么类型的数据。整数加法和浮点数加法计算方法相同。但除法会有所不同，在 Python 3.0 之后提供了两种不同的运算符，以前常见的符号斜线(/)用于常规除法，双斜线(//)用于整型除法。通过代码来比较它们的差异：

```
>>> 100 / 3
33.333333333333336
>>> 100 // 3
33
>>> 100.0 / 3.0
33.333333333333336
>>> 100.0 // 3.0
33.0
>>> 100.00 // 3.00
33.0
>>> 100 % 3
1
>>> 100.0 % 3.0
1.0
```

请注意，操作符“/”总是返回一个浮点数。即使操作数是 int 型，除法产生的结果也是带小数的浮点数。这与 C 系列语言和 Java 语言有所不同。你可能注意到了，在 Python 中，100/3 和 100.0/3.0 的结果最后都有一个 6，这是因为浮点数总是近似值。

要获得返回整数结果的除法，可以使用整数除法运算“//”，因为整数除法总是产生一个整数。可以把整数除法看作整除。表达式 10//3 得到 3。虽然整数除法的结果总是一个整数，但结果的数据类型却取决于操作数的数据类型。整数整除浮点数得到一个浮点数，它的分数分量为 0。

最后两行展示了余数运算%。请再次注意，结果的数据类型取决于操作数的类型。

求余数的操作采用“%”,可以把一个数用“//”和“%”表示出来,例如:

```
a=(a//b)(b)+(a%b)
```

3.2 类型转换和舍入

前面已经知道了同种数据类型数据的基本运算,但在某些情况下,可能会将两种不同的数据类型进行计算,这时就要将一种数据类型转换成另一种数据类型进行运算。例如一个整型数据与一个浮点型数据相加,此时 Python 会如何处理? 比如计算下面示例:

```
x = 5.0 + 2
```

如果两者都为整型或都为浮点型,此时结果很简单,分别为 7 和 7.0。当遇到示例这种情况时,我们首先想到的是将两者变为同一种数据类型,其中一种是将浮点型 5.0 变为整型 5,然后再与 2 相加;另一种是将整型 2 变为浮点型 2.0,再与 5.0 相加。

通常情况下,将浮点型转换为整型会使计算结果不精确,从而导致发生严重的错误,因为当浮点型小数点后不为 0 时,将浮点型转换为整型,小数部分就会被截去,从而导致结果错误。而将整型转换为浮点型,只需要在整型后加上小数点即可,所以,在混合类型表达式中,Python 会自动将整型转换为浮点数,并执行浮点运算以产生浮点数结果,这种转换,称为“隐式转换”。

当然,程序员也可以通过显式转换来指定数据的类型转换。Python 提供了内置的转换函数 int()和 float()。

```
>>> int(3.14)
3
>>> float(3)
3.0
>>> float(int(3.14))
3.0
>>> int(float(3))
3
```

如你所见,转换为整型就是丢弃浮点值的小数部分,该值将被截断,而不是舍入。对数值进行四舍五入的通常方法是使用内置的 round()函数,它可以将数值四舍五入到最接近的整数值。

```
>>> round(3.14)
3
>>> round(3.5)
4
```

请注意，像这样调用 round()函数将会产生一个整型的值。因此，对 round()的简单调用是将浮点型转换为整型的另一种方法。

```
>>> pi = 3.141592653589793
>>> round(pi, 2)
3.14
>>> round(pi,3)
3.142
```

类型转换函数 int()和 float()还可以将数字字符串转换为数字类型：

```
>>> int("1234")
1234
>>> float("1234")
1234.0
>>> float("1234.5")
1234.5
```

通过上面的分析，完全可以利用这种方法替代 eval()函数，获取用户的数字数据，这特别有用，而且降低了“代码注入”攻击的风险。下面是零钱计数程序的一个改进版本：

```
#3_2 new_change.py
#A program to calculate the value of some change in dollars
def main():
    print("Change Counter")
    print()
    print("请输入你的各种硬币个数")
    yuan = int(input("有多少 1 元的硬币: "))
    fifty_cents = int(input("有多少 5 角的硬币: "))
    twenty_cents = int(input("有多少 2 角的硬币: "))
    ten_cents = int(input("有多少 1 角的硬币: "))
    total =yuan * 1.0 + fifty_cents * .50 + twenty_cents * .20 + ten_cents * .10
    print()
    print("你拥有的硬币总额是", total)

main()
```

注意：在 input 语句中使用的是 int()函数而不是 eval()函数，可以确保用户只能输入有效的整数。任何非法(非整数)输入都会导致程序崩溃和错误消息，从而避免代码注入攻击的风险。另外，还有一个好处是，这个版本的程序强调输入应该是整数。

使用数字类型转换代替 eval()函数的唯一的缺点是，它不支持同时输入(即不能在单个输入中获取多个值)。

```
>>> #simultaneous input using eval
>>> x,y = eval(input("Enter (x,y): "))
Enter (x,y): 3,4
>>> x
3
>>> y
4
>>> #does not work with float
>>> x,y = float(input("Enter (x,y): "))
Enter (x,y): 3,4
Traceback (most recent call last):
File "<stdin>", line 1, in <module>
ValueError: could not convert string to float: '3,4'
```

这样代价很小,但换来了额外的安全性。在第 5 章,你将学习如何克服这个限制。作为一种良好的实践,在编写程序时,应该尽可能使用适当的类型转换函数去代替 eval()函数。

3.3 使用 math 库

Python 标准库 math 提供了很多有用的数学函数,利用 math 库可以使用浮点值完成复杂的数学运算,包括三角函数运算、对数运算等。库就是一个模块,包含了很多封装好的函数表达式,可以极大地方便程序员解决问题,如利用 math 库来解决二次函数问题。

在数学中学习过,一元二次方程的标准式为 $ax^2+bx+c=0$,求该方程的解,可以利用一元二次方程求解公式来进行求解:

$$x=\frac{-b\pm\sqrt{b^2-4ac}}{2a}$$

这里可以利用 math 库来完成该方程的求解。首先要求用户输入方程中参数 a、b、c 的值,最后通过程序处理输出方程的两个解。

```
#3_3 math.py
#A program to math
import math

def main():
    print("求解一元二次方程")
    print()
    a = float(input("输入系数 a: "))
    b = float(input("输入系数 b: "))
```

```
    c = float(input("输入系数 c: "))
    discRoot = math.sqrt(b * b - 4 * a * c)
    root1 = (-b + discRoot) / (2 * a)
    root2 = (-b - discRoot) / (2 * a)
    print()
    print("方程的解为:", root1, root2 )

main()
```

该程序调用了 math 库中的 sqrt()函数来求平方根。但这个程序仍然存在一个问题,大家都知道求解一元二次方程时,只有在 $b^2-4ac\geqslant 0$ 时方程有实数解,否则方程无实数解。所以当我们输入的系数使得 $b^2-4ac<0$ 时,会导致程序崩溃。如何解决这个问题,大家可以思考一下。

熟练利用各种库,可以将复杂代码简单化,简化编写步骤,增强可读性。Python 的标准库 math 还提供了很多的函数,如表 3-3 所示。

表 3-3　math 库的一些函数

Python 的函数	数　　学	描　　述
sqrt(x)	$\sqrt{x}$	x 的平方根
sin(x)	sin(x)	x 的正弦值
cos(x)	cos(x)	x 的余弦值
tan(x)	tan(x)	x 的正切值
asin(x)	arcsin(x)	x 的反正弦
acos(x)	arccos(x)	x 的反余弦
atan(x)	arctan(x)	x 的反正切
log(x)	ln(x)	x 的自然对数
log10(x)	lgx	x 的常用对数
exp(x)	e^x	e 的 x 次方
ceil(x)	[x]	最小的大于或等于 x 的值
floor(x)	[x]	最大的小于或等于 x 的值

3.4　累积结果：阶乘

在数学中,一个正整数的阶乘是所有小于或等于该数的正整数的积,且 0 的阶乘为 1,自然数的阶乘记为 n!。

大家肯定遇到过这样的问题,6 个人随便站队排列,求一共有多少种可能性？这是一

个排列组合问题，可以很快地给出答案，一共有 6！种排列组合，更详细一点为 6×5×4×3×2×1＝720 种排列方式。

编写一个程序，让计算机来处理用户输入数字的阶乘。程序的基本结构遵循“输入、处理、输出”模式：

```
※ ----伪代码----
※ 输入要计算阶乘的数,n
※ 计算 n 的阶乘,fact
※ 输出 fact
```

整个过程的难点在第二步求解 n 的阶乘。为了求解阶乘，先来看看阶乘 n 的表达式：n!＝n×(n－1)×(n－2)×...×2×1。从表达式可以看到，每次相乘的数都比前一个数小 1，且整个过程是一个重复相乘（即累乘）的过程，所以，可以使用循环结构来实现这个过程。因此得到一般模式如下：

```
※ ----伪代码----
※ 初始化累加器变量
※ 循环直到得到最终结果
※ 更新累加器变量的值
```

前面已经学习了 range()函数，range(n)产生一个数字序列，从 0 开始，增长到 n，但不包括 n。range 还有一些其他调用方式，可用于产生不同的序列。可以利用 range()函数的两个参数，即 range(start,n)，产生一个以值 start 开始的序列，增长到 n，但不包括 n。range()函数还有 3 个参数的使用方法，如 range(start,n,step)，这个形式十分类似于双参数，它的第 3 个参数 step 的作用是作为每个数字之间的增量，即前后两个数字的差值。

利用 range()函数可以得到其中一种求解阶乘的方法，如下：

```
#3_4 factorial.py
#A program to factorial
def main():
    n = int(input("请输入 n:"))
    fact = 1
    for i in range(1,n+1):
        fact = fact * i
    print( n,"的阶乘为",fact)

main()
```

当然，求解阶乘还有很多别的方法，这只是其中的一种。有兴趣的话，读者可以自己再设计几种实现阶乘求解的程序。

本章小结

本章介绍了数字的计算，主要是整型和浮点型数据的计算。对于数值类型的转换和运用，要注意的是，Python 一般会把整型转换为浮点型进行计算。内置的 round()函数能将数字四舍五入到最近的整数值。本章还举例说明了如何实现阶乘的运算。除此之外，列举了一些 math 库的函数，方便进行一些数学运算。

知识扩展：运算符优先级

优先级和结合性是 Python 表达式中比较重要的两个概念，它们决定了先执行表达式中的哪一部分。Python 支持几十种运算符，被划分成将近二十个优先级，只有相同优先级别的运算符才遵循从左到右计算，否则优先级高的运算符优先计算，运算符优先级如表 3-4 所示。

表 3-4　运算符优先级

Python 运算符	运算符说明	优　先　级	结　合　性
( )	小括号	19	无
x[i]或 x[i1：i2 [：i3]]	索引运算符	18	左
x.attribute	属性访问	17	左
**	乘方	16	右
～	按位取反	15	右
＋(正号)、－(负号)	符号运算符	14	右
*、/、//、%	乘除	13	左
＋、－	加减	12	左
>>、<<	位移	11	左
&	按位与	10	右
^	按位异或	9	左
|	按位或	8	左
＝＝、!＝、＞、＞＝、＜、＜＝	比较运算符	7	左
is、is not	is 运算符	6	左
in、not in	in 运算符	5	左
not	逻辑非	4	右
and	逻辑与	3	左
or	逻辑或	2	左
exp1，exp2	逗号运算符	1	左

课程思政：创造了国产软件的骄傲——求伯君

金山办公于2019年11月18日，正式在上交所科创板挂牌交易，股票简称“金山办公”，市值超过600亿。其主打产品WPS Office，大家不会陌生。正如金山集团董事长、小米集团董事长兼CEO雷军所说：“金山WPS是一家有梦想、有使命感的公司。31年坚持做一件事，并且越做越好！”

回顾WPS的成长之路，不得不提一个人，他就是WPS创作者、金山软件创始人求伯君。求伯君是金山软件股份有限公司创始人之一，有“中国第一程序员”之称。他是个名副其实的学霸，高考以数学满分、县里第一的成绩考上了国防科技大学。

1. 天赋与钻研劲并存

毕业后的一天，求伯君有个同学的打印机出问题了，请他过去帮忙。求伯君发现故障原因是打印机驱动不兼容。他的钻研劲儿上来了，认真思考着：“我为什么不搞个通用的打印机驱动呢？”于是他用了9个晚上，写了一个5万行汇编程序语言的支持多种打印机的驱动程序，把打印机驱动的问题解决了。同学建议他去四通公司，四通公司的人看见求伯君的打印驱动以后，马上提出要买下这个程序的全部版权。后来，求伯君就留在了四通公司。同样的情况再次出现了，四通公司的合作伙伴金山公司的老板张旋龙，有一批机器的输入输出系统出现了问题，计算机无法启动。他手下团队里的那些专业人士都对这个问题一筹莫展，四通公司就派了求伯君去试试看，求伯君只花了一晚上就把问题给解决了。

2. 办公软件WPS 1.0横空出世

到了金山以后，求伯君打算做一件大事，他觉得当时市面上主流的汉字处理系统Word Star不好用，想写一个更好的出来。于是，求伯君对着一台386计算机开始埋头苦写代码，只要醒着，就不停地写；困得看不清计算机屏幕了，才躺下来眯一会儿。有时忙得两三天才记起来吃一顿饭。就这样敲了一年零四个月的代码后，求伯君愣是一个人敲完了122000行代码，WPS 1.0横空出世！

WPS的诞生具有跨时代的意义，它极大地提升了中文办公的效率。短短的时间内就成了中国计算机的标配，占据了90%的中国市场。求伯君也因此名扬四海，25岁的求伯君被人们称为“中国第一程序员”。

3. 金山陷入危机

1996年，微软公司希望金山公司将WPS格式与微软公司共享，使两者可以兼容。当时WPS只有DOS系统版本，微软公司得到兼容格式后，快速将WPS的老用户转移到Windows平台，抢占了中国的市场份额，整个金山公司岌岌可危，求伯君于是准备用3年时间重写WPS。因为亏损，没有资金，求伯君二话没说就把自己的房子卖了。在市场上

沉默了几年之后，金山公司推出了 WPS 97，凭借以往的用户基础，两个月内就售出了 13000 多套。

4. 中国第一代程序员心中国产软件的骄傲

在求伯君的身上集结着很多的第一：1988 年成功开发国内第一套文字处理软件 WPS 1.0；1995 年推出中国第一个游戏软件《中关村启示录》等。值得一提的是，面对微软公司的 Office 在国内一统江湖的竞争局面，WPS 2005 实现了与 Office 在内容和格式上的“深度兼容”，满足了用户在使用习惯上的要求，而整个软件采用开发式架构，更有利于对未来各种格式的升级。多年来肩扛民族软件大旗的金山公司宣称，WPS 2005 的发布意味着 Office 和 WPS 之战，将由战略相持阶段转向 WPS 全面反攻阶段。2011 年 11 月 18 日，求伯君宣布退休。现如今，他已经不再插手繁杂的事务。但是，WPS 始终是求伯君年少时梦想的化身，始终是中国第一代程序员心中国产软件的骄傲。

也许从世俗的名利上来讲，求伯君没有比尔·盖茨改变世界的野心，没有登上福布斯，没有上名人榜。这些我们以为的成功，他似乎从来都没有在乎过。求伯君被称为“中国第一程序员”，不是因为他熬得了写程序的苦，也不是因为他写代码的能力强大到无人可及，更多的是他在那个中国处处被国外卡脖子的年代，让我们自己的软件，一直站着，没有跪下！