项目 3　计 算 能 手

本项目首先学习在编辑器中直接输入算式，练习键盘输入，熟悉算式的编写。然后，通过编写程序实现互动：询问出题、获取题目、打印答案。在学习数学的时候，数字有哪些类型呢？有整数、小数、分数等。Python 的数字类型涉及整数、浮点数等。

除了学习算式的输入和数字的表达，还需要巩固变量的使用，学习 while 循环的基本用法。

任务 3.1　在解释器中做计算题

进入 Python Shell 解释器就可以自由出题了，输入题目，按下 Enter 键，Python 马上就能给出答案。

对照图 3-1，在键盘上找一找 Python 能认识的运算符号。表 3-1 列出了键盘输入方式以及常见算式举例。

图 3-1　键盘示意图

表 3-1　运算符号及示例

名　称	符　号	键盘输入方式	算 式 举 例
加法	+	Shift++	3+2
减法	–	–	10–8
乘法	*	Shift+*	8*3
除法	/	/	20/5
小括号	( )	Shift+ ( )	(8+2) *2
取余	%	Shift+%	10%7

由易到难，试试输入如图 3-2 所示算式，然后接着出题，试着输入你认为最复杂的算式。看起来，Python 做计算题似乎不需要思考时间！

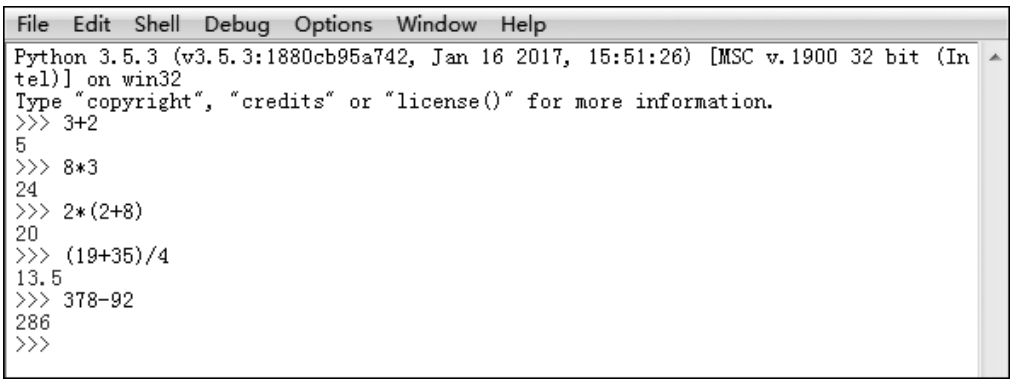

图 3-2　数字计算

还能想到哪些算式？跟同学一起输入同样的算式，每一台计算机里的 Python 计算出来的结果都是一样的吗？

任务 3.2　X=X+2

如果用变量表示一个数，怎样写加法算式呢？例如，*X*=2，*X*=*X*+7，这两个等式，按照数学课学习的计算方法，是不成立的。如图 3-3 所示，试试输入这两个算式，看看 *X* 是多少？

Python 是怎样计算的呢？尽可能多地写一写类似的算式，自己先算一算答案，再让 Python 算一算，比较答案是否一致。

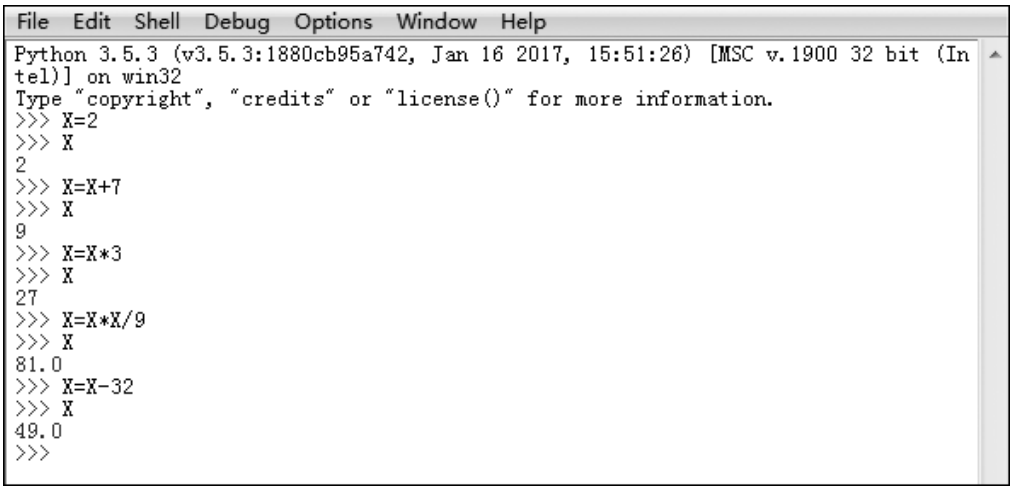

图 3-3　变量 *X* 的运算

随着算式的改变，变量 X 表示的数值也在变化，并通过 = 符号存放到变量 X 中。这个过程叫作赋值。例如，把数字 2 赋值给 X，再将 X+7 的运算结果赋值给 X,其实就是把 2+7 的计算结果赋值给 X,此时变量 X 的值是 9。

从图 3-3 中可以发现，所有的计算结果是没有分数形式的，只有整数或者小数。可是，明明计算结果是整数 81，为何 Python 却显示 81.0 呢?

任务 3.3　给 Python 出计算题

能够熟练地输入算式以后，本任务学习编写一个完整的程序。当给计算机出一道计算题后，按下 Enter 键，屏幕即显示结果。学习中，按照由易到难的过程，首先是简单的一次问答，然后是多次问答，以及循环结构。

1. 单次运行

想要保存编写的程序，就需要创建一个 **.py 的文件。编写完成的程序可以保存、打开、调试和修改。首先需要明确任务是什么，思考用什么办法实现，再动手编写。

（1）任务描述。

屏幕显示“输入一道计算题：”，等待输入计算题之后，屏幕打印出“正确答案是：”。

（2）任务分析。

需要使用 input 函数获取问题信息，使用 print 函数输出答案，使用 problem 作为变量，存储获取的计算题。

（3）解锁新技能。

使用 eval 函数，可以将字符串转化为一个数值，并返回这个数值。可以简单地理解为“去掉最外层引号”。例如，输入算式 3+2，程序将运行代码 problem=“3+2”，当 eval 剥去了“3+2”外面的引号后，会对它进行解析，满足要求后进行计算，并返回计算结果。

相反地，如果输入了一个不正确的算式，Python 解析后，认为它不能计算，还会判断它是不是一个变量，如果是就会输出这个变量的内容，如果不是就会报错。这是 Python 在给你提示，需要重新检查代码了。

（4）代码编写和调试。

① 打开 IDLE，依次选择 File → New File 命令，新建一个文件。

② 输入程序代码，还可以为其增加注释，一般以红色显示，用 # 开头，如图 3-4 所示。

③ 依次选择 Run → Run Module 命令后，提示保存文件，保存后自动弹出运行窗口。

④ 输入一道计算题，输入完成，按下 Enter 键，显示计算结果，如图 3-5 所示。

⑤ 重复步骤③和④，可以再次运行程序。

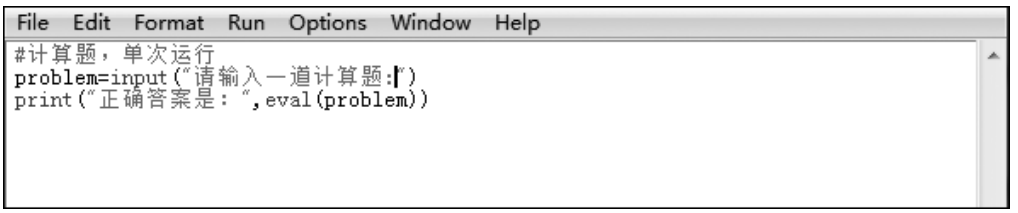

图 3-4　计算题代码编写（单次）

图 3-5　计算题代码调试

2. 按顺序执行（多次重复）

通过调试可以发现，输入一道计算题，计算完成后，程序就运行结束了。如果要实现连续多次做不同的计算题，可以多次重复编写上面的两行代码，如图 3-6 所示。想要做几次不同的计算，就需要重复编写几组代码。

对于重复或相似的代码，采用逐行复制、粘贴的方法，只对一些参数略

加修改即可，甚至不需要修改，就可以快速写完很长的代码，如图 3-6 所示。

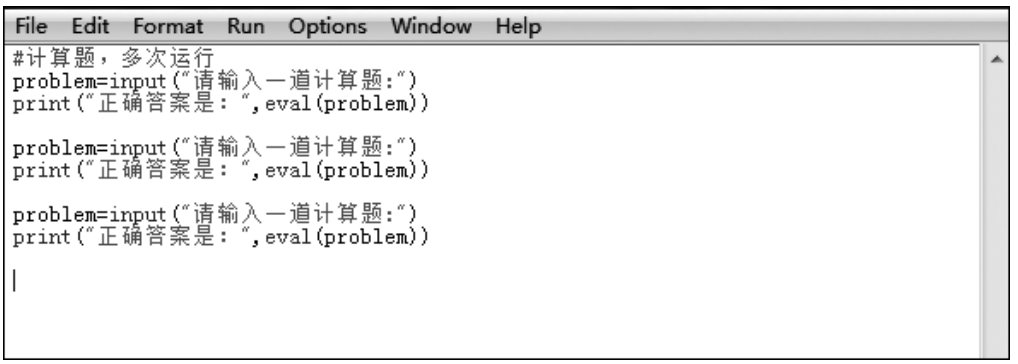

图 3-6　计算题代码编写（多次）

任务 3.4　程序优化：while 循环

从任务 3.3 论述可见，需要做 3 次计算，就要重复 3 次编写计算代码，若需要做 30 次计算，就要重复 30 次。遇到这种有规律的、反复执行某段功能程序的情况时，可以对程序进行优化，引入新指令，为程序增加循环结构，让程序看起来更简洁。

1. 任务描述

通过编程，实现输入一道计算题，显示答案后，再次输入一道计算题，显示答案，如此循环，直到输入 q 字符结束出题。

2. 任务分析

将重复的顺序结构简化为循环结构。计算机能够循环执行任务，需要知道什么时候开始循环，什么时候结束循环。也就是什么时候要求出题，什么时候停止出题。如果不想出题的时候，按某个键“退出”就可以了。“退出”的英文单词是 quit。因此，取首字母，将循环条件设置为判断输入的内容是否是 q。当输入内容是字母 q 时，就结束循环。

3. 解锁新技能

while 指令可以实现类似的循环结构。它的意思是“当……时候就……”。

语法格式如下：

```
while（循环条件）:
    循环主体
```

首先画出流程图，理清思路，如图 3-7 所示。

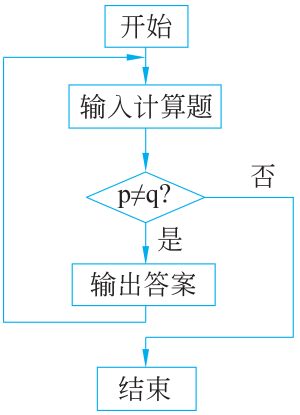

图 3-7　计算题 while 循环流程图

4. 代码编写和调试

为了简化字母拼写，可以用字母 p 作为变量名，存储输入的计算题。

（1）打开 IDLE，依次选择 File → New File 命令，新建一个文件。

（2）输入程序代码，还可以为其增加注释，一般以红色显示，用 # 开头，如图 3-8 所示。

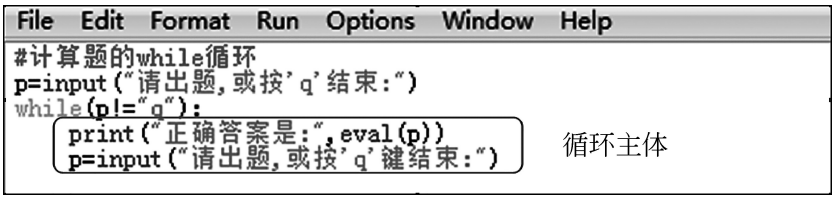

图 3-8　计算题的 while 循环代码

（3）依次选择 Run → Run Module 命令后，提示保存文件，保存后自动弹出运行窗口。

（4）运行结果：在运行窗口中出现“请出题，或按 'q' 结束：”的一行文字，在冒号后输入要计算的数学式子，再按 Enter 键，在下一行显示计算结果；显示结果后又自动弹出“请出题，或按 'q' 键结束：”的一行文字，可以继续出题或输入 q 后结束程序运行，图 3-9 中是连续输入 4 个计算题后的界面。

```
============
请出题,或按'q'结束:18**2
正确答案是: 324
请出题,或按'q'键结束:18*18
正确答案是: 324
请出题,或按'q'键结束:398-256
正确答案是: 142
请出题,或按'q'键结束:32/4+(7-2)
正确答案是: 13.0
请出题,或按'q'键结束:q
>>>
```

图 3-9　计算题 while 运行结果

5. 关键技能

（1）语法格式：while 语句后有“：”，循环主体代码要缩进，可以理解为“逢冒号必缩进”。

（2）“!=”是不等于符号，它包括一个“！”和一个“=”。

（3）屏幕出现光标 >>>，表示程序结束。

（4）18**2，表示 18 的平方，也就是 18*18。注意与 18*2 进行区分。

任务 3.5　扩展阅读：Python 的基本数据类型

Python 中有 6 个基本数据类型，分别是 Number（数字）、String（字符串）、List（列表）、Tuple（元组）、Set（集合）、Dictionary（字典）。其中，不可变数据类型有 3 个：Number（数字）、String（字符串）、Tuple（元组）；可变数据类型有 3 个：List（列表）、Set（集合）、Dictionary（字典）。

1. 数字型

Python 3 支持整型（int）、浮点型（float）、布尔型（bool）、复数（complex）

4 种数字类型。

（1）整型。Python 中的整数没有长度限制，如输入代码：print（9**999），计算 9 的 999 次方，观察运行结果。恐怕任何语言都不能允许如此长的整型数。

整数类型有十进制、二进制、八进制和十六进制，每种写法使用不同的前缀进行区别。最常见的就是十进制写法，如 32000、898 等。无论写代码时使用哪种进制，运行时，Python 都会自动转换为十进制输出。

十六进制写法：加前缀 0x，出现 0~9 和 A~F 的数字和字母组合。

八进制写法：加前缀 0o，出现 0~7 的数字组合。

二进制写法：加前缀 0b，只有 0 和 1 的数字组合。

可以使用 print 指令输入这些数据类型，体会进制的书写规范和 Python 的执行结果。例如，可以使用下面这些语句，观察运行结果。

```
print (0x10)
print (0o10)
print (0b10)
```

（2）浮点型。Python 语言中带小数点的数都可以称为浮点数，一般把长度短的叫作小数。浮点数只能以十进制表示，不能加前缀，否则会报语法错误。

与整型不同，浮点数有长度限制，边界值为 max=1.7976931348623157e+308，min=2.2250738585072014e–308。

（3）布尔型。布尔型就是通常说的逻辑，使用比较运算符连接，表示对和错。例如，print（100==100.0），就是比较两个值是否相等，运行结果为 true，而不是括号中的内容。

（4）复数。Python 中的复数这样来表示：1 + 1j，虚部为 1，仍不可省略。例如：

```
print((1 + 2j).real) # 输出实部 float 类型
print((1 + 2j).imag) # 输出虚部 float 类型
```

运行结果是

```
1.0
2.0
```

2. 字符串

通俗来说，字符串就是字符组成的一串内容，Python 中用成对的单引号或双引号括起来，用 3 个单引号或双引号可以使字符串内容保持原样输出，可以包含 Enter 等特殊字符，在 Python 中字符串是不可变对象。

最为熟悉的是 input（“……”），引号内部的数据就是字符串类型。如果没有外面的引号，Python 会认为这可能是一个没有定义的变量，然后报错。

字符串可以进行很多操作，常见的有字符串长度、字符串连接、字符串索引、字符串切片等。伴随着编程学习的深入，这些操作方法可以解决更多问题。

3. 其他几种基本数据类型

其他几种基本数据类型，在基础入门阶段只做了解就可以，因为比较少用到，理解起来也较为复杂。其中的列表数据类型，在后续的 for 循环学习中会有一些浅显的应用。

此外，Python 的数据类型还可以相互转换，如 str(x)，就是将对象 x 转换为字符串 eval（str），用来计算在字符串中的有效 Python 表达式，并返回一个值。

总之，数据类型是所有计算机语言中必不可少的基础知识，也是非常重要的组成成分。按照 Python 的规则，正确使用这些数据类型，才能编写出完美的代码，从而进行各种各样的控制，完成各种各样的任务。

任务 3.6　总结和评价

（1）展示程序运行结果。

（2）谈谈自己对每一行程序代码的理解。

（3）项目 3 已完成，在表 3-2 中画☆，最多画 3 个☆。

表 3-2　项目 3 评价表

评 价 描 述	评 价 结 果
我能正确输入算式	
我能编写项目中的“单次运行”程序，并调试成功	
我能熟练地使用键盘输入代码，细致且有耐心	
我能说出对项目中的 while 语句的理解	
我能编写项目中的“while 循环”，并调试成功	

（4）编程挑战如下。

① a=3，b=5，编写程序并交换 a、b 的值，使 a=5，b=3。

提示：交换两个变量中的数值，不是简单地给变量赋值。试想：要把两个杯子里的溶液互换，需要使用第三个杯子。用类似的思路可以完成这个编程挑战！

② 设计程序，用 while 循环交换 a、b 的值。

提示：首先要进行变量初始化。while 循环可以做到输入任意两个数值，都可以进行数值交换，需要用到 input 函数和 print 函数。打印时，应保证格式清晰。