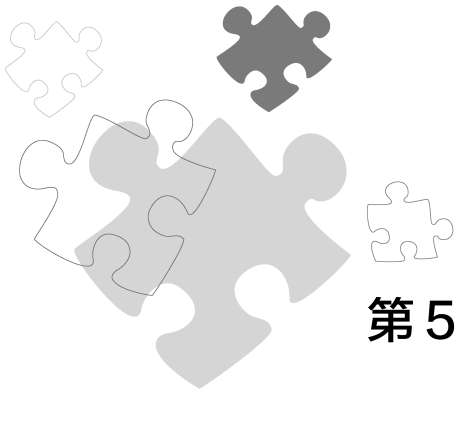

第5章　循环结构

现代的计算机每秒可以完成亿万次的运算和操作，用户不可能为此写亿万条指令去让计算机运行，解决的方法是将一个复杂功能变成若干个简单功能的重复，然后让计算机重复执行这些简单功能来最终完成这个复杂功能。

循环结构是实现让计算机重复执行一件工作的基本方法，也是程序的一种基本结构。所谓循环，就是重复地执行某些操作。例如，小王 2011 年每个月收入 3000 元，他的收入每年增长 10%，房价每年增长 5%，他想买上 2011 年价值 50 万元的房子需要到哪一年？如果他想在 2020 年前买房，他的年收入增长最少应达到多少？这样的问题用计算机的循环结构去求解是最方便的。在 Python 语言中，可以实现循环结构的语句有两种：while 语句和 for 语句。

5.1　while 循环结构

while 循环结构是根据条件来判断是否需要继续重复执行语句块，故可以称为条件循环。无论循环的次数是否预知，都可以使用 while 语句来循环执行语句块。

while 语句包括两种形式：

- 基本 while 语句；
- 扩展 while 语句。

1. 基本 while 语句

格式：

```
while 条件表达式:
    语句块
```

while 语句的执行过程：先计算条件表达式的值，若条件表达式的值为 True，则执行语句块，并返回条件处，重新计算条件表达式值后决定是否重复执行语句块；若条件表达式的值为 False，则循环结束，执行 while 语句之后的后续语句。其中的语句块称为循环体。

注意：

- 条件表达式多数为比较表达式或逻辑表达式，但也可以是其他计算结果的表达式；

- 条件表达式后必须加冒号；
- 相对于 while 所在行，循环体中的所有语句行都应向右缩进对齐，并保持一致的缩进方式；
- 先计算条件表达式的值，再执行循环体，故 while 的循环体有可能一次也不被执行；
- 如果条件表达式的值永远为 True，则循环将无限次的执行下去（俗称死循环）。若正在执行的程序中包含死循环时，需要按 Ctrl＋C 快捷键来中断程序执行。编写循环代码时应在循环体内改变条件表达式中各变量的值，从而使条件表达式的结果为 False，以避免死循环的发生。

【例 5-1】 编写程序求 1＋2＋3＋…＋100 的值。

解题步骤：

Step1：初始化变量，i＝1，sum＝0。i 表示被加数 1～100，sum 存放 1～100 的累加值。

Step2：判断 i＜＝100 的值是否为真，若为真，将 i 累加到 sum，然后 i 加 1。

Step3：重复 Step2 直到 i＜＝100 为假，退出循环。

Step4：输出累加值 sum。

代码如下：

```
#1.  i = 1
#2.  sum = 0
#3.  while i <= 100:
#4.      sum += i
#5.      i = i + 1
#6.  print(sum)
```

运行程序，结果如下：

```
5050
```

【例 5-2】 编写程序，求令 1 * 2 * 3 * …n＞＝100 000 成立的最小 n 值。

解题步骤：

Step1：初始化变量，i＝1，r＝1。整型变量 i 用于存放乘数，变量 r 存放乘积。

Step2：判断 r＜100 000 的值是否为真，若为真，转 Step4，若为假，转 Step6。

Step3：将 i 加 1，然后 r＝r * i。

Step4：转 Step3。

Step5：循环结束，输出最后的乘数 i。

代码如下：

```
#1.  i = 1
#2.  r = 1
#3.  while r <= 100000:
#4.      r *= i
#5.      i = i + 1
#6.  print(i)
```

运行程序，结果如下：

```
10
```

【例 5-3】 输入一个整数，求它的各位数字之和。

```
#1.  n = int(input())
#2.  sum = 0
#3.  while n:
#4.      r = n % 10
#5.      n = n // 10
#6.      sum += r
#7.  print('sum = ', sum)
```

测试程序，运行结果如下：

```
请输入一个整数:293↙
sum = 14
```

2. 扩展 while 语句

格式：

```
while 条件表达式:
    语句块 1
else:
    语句块 2
```

扩展 while 语句增加的 else 子句与将在 5.3 节中介绍的 break 语句有关。当 while 语句是因为某次条件值为 False 而结束循环时，程序会执行 else 之后的语句块 2。如果循环体语句块 1 中，因为执行了 break 语句而结束循环时，就不会执行 else 后的语句块 2。

例 5-2 的代码也可以这么编写，执行效果是相同的。

```
#1.  i = 1
#2.  r = 1
#3.  while r <= 100000:
#4.      r *= i
#5.      i = i + 1
#6.  else:
#7.      print(i)
```

5.2　for 循环结构

for 循环结构中，循环的控制需要有序列或可迭代对象参与，循环过程中对序列或可迭代对象中的元素逐一处理，故又称为遍历循环。

for 循环结构适用的情况主要有：

- 循环次数已知；
- 需要遍历处理 Python 的序列结构或可迭代对象中的每个元素。

for 语句的格式为：

```
for 循环变量 in 遍历结构:
    语句块 1
[else:
    语句块 2]
```

for 循环的执行过程：从遍历结构中逐一提取元素，放入循环变量，循环次数就是元素的个数，每次循环中的循环变量值就是遍历结构中提取的当前元素值。

可选的 else 部分执行方式和 while 语句类似。如果全部元素被遍历后，结束执行循环体，则执行 else 后的语句块 2；若因在语句块 1 中执行了 break 语句而结束循环时，不会执行 else 后的语句块 2。

注意：

(1) for 循环的循环次数等于序列结构或可迭代对象的元素个数。

(2) 若需要按指定次数循环，可以使用 range()函数产生的 range 对象来配合控制循环。range()函数的格式：

格式一：

```
range(stop)
```

格式二：

```
range(start, stop[, step])
```

格式一生成从 0 开始到 stop－1 的连续整数，格式二生成从 start 开始到 stop－1 的间隔为 step 的整数。函数返回的是 range 序列对象。例如：

```
>>> range(5)
range(0, 5)
>>> range(3,10,3)
range(3, 10, 3)
```

(3) 若循环的处理内容是序列中的元素，或处理内容与序列结构中的元素相关，则非常适合采用 for 循环。

(4) 循环变量的取值是对序列结构或可迭代对象的当前元素值的复制，而不是元素值本身，故修改循环变量的值，无法改变序列结构和可迭代对象的值，因此循环变量的变化次数是恒定不变的。例如：

```
#1.  x = [1, 2]
#2.  for i in x:
#3.      print(i, end = '')
#4.      i = i - 3
#5.      print(i)
#6.  print(x)
```

运行结果为：

```
1 -2
2 -1
[1, 2]
```

(5) for 循环中，若控制用的序列结构的元素增加或减少，则循环的执行情况就比较复杂。下面列举了几种情况。

代码一：

```
#1.  x = [1, 2, 3, 4]
#2.  for i in x:
#3.      print(i, end = ' ')
#4.      print(x.pop())
#5.      print(x)
#6.  print('The loop end!')
#7.  print(x)
```

运行结果：

```
1 4
[1, 2, 3]
2 3
[1, 2]
The loop end!
[1, 2]
```

上述代码的#3 行输出的是从序列中依次取得的元素值，#4 行的 pop()方法是弹出列表的最后一个元素，返回值就是弹出值。由于第二次循环执行后，列表中只剩下了两个元素。而遍历循环的执行次数只与序列的元素个数有关，因此执行两遍循环后就结束了。

代码二：

```
#1.  x = [1, 2]
#2.  for i in x:
#3.      print(i, end = ' ')
#4.      if i != 5: x.append(5)              # 列表原地增加元素
#5.      print(x)
#6.  print('The loop end!')
#7.  print(x)
```

运行结果：

```
1 [1, 2, 5]
2 [1, 2, 5, 5]
5 [1, 2, 5, 5]
5 [1, 2, 5, 5]
The loop end!
[1, 2, 5, 5]
```

上述代码的#3行是输出从序列中依次取出的元素值，#4行是当这个取出的值不为5时将在末尾增加一个5。最初的列表共有两个不为5的元素，所以在执行前两次循环时，都会在末尾添加一个5，最后总共存在4个元素，所以循环总共执行了4次。

代码三：

```
#1.  x = [1, 2]
#2.  for i in x:
#3.      print(i, end = ' ')
#4.      if i != 5: x = x + [5]          # 列表非原地增加元素
#5.      print(x)
#6.  print('The loop end!')
#7.  print(x)
```

运行结果：

```
1 [1, 2, 5]
2 [1, 2, 5, 5]
The loop end!
[1, 2, 5, 5]
```

上述代码与代码二的区别仅在于#4行的列表添加元素方式不同。代码二的列表添加元素是原地添加，将会影响到原有的循环次数；而代码三中的列表添加元素是非原地添加，不会影响到循环次数。

【例5-4】 用for语句求1+2+3+…+99+100。

```
#1.  sum = 0
#2.  for i in range(1, 101):                # range()函数产生的数不包括101
#3.      sum += i
#4.  print(sum)
```

运行程序，结果如下：

```
5050
```

注意：#2行中range()函数的第2个参数是101，而不是100，因为range()函数的范围有“左闭右开”的特点。

【例5-5】 用for语句求一个整数的各位数字之和。

```
#1.  n = input("请输入一个整数: ")
#2.  sum = 0
#3.  for i in n:
#4.      sum += int(i)
#5.  print(sum)
```

测试程序，运行结果如下：

```
请输入一个整数：382↙
13
```

注意：字符串也是序列结构，for 循环会遍历字符串中的每个字符，再利用 int()转换每个字符即可获得各位上的数字。

【例 5-6】 编写程序找出所有三位水仙花数。所谓水仙花数是指其各位数字的立方和等于该数本身。例如，$153=1^3+5^3+3^3$，所以 153 是水仙花数。

```
#1.  for i in range(100, 1000):
#2.      a, temp = divmod(i, 100)
#3.      b, c = divmod(temp, 10)
#4.      if a ** 3 + b ** 3 + c ** 3 == i:
#5.          print(i)
```

运行程序，结果如下：

```
153
370
371
407
```

5.3　循环控制语句

与循环结构相关的语句还有 break、continue 和 pass 语句，它们可以改变循环执行的流程。其中，break 和 continue 语句只能用于循环体内，pass 语句还可用于其他控制结构。

5.3.1　break 语句

循环体中使用 break 语句，可以跳出包含 break 语句的那层循环，从而提前结束该循环。跳出循环后，继续执行当前层循环的后续语句。

break 语句一般是和 if 语句、循环的 else 子句一起结合使用的。

【例 5-7】 输入一个正整数 n，判断它是否为素数。素数就是只能被 1 和自身整除的数。

解题思路：判断 n 是否为素数，可以按素数的定义进行判断，用 n 依次除以 2～$n-1$ 的所有数，只要发现有一个数能够被 n 整除，马上可以结束循环，判定 n 不是素数。如果没有一个能够被 n 整除的数，则 n 为素数。

```
#1.  n = int(input('请输入一个整数：'))
#2.  for i in range(2, n):
#3.      if n % i == 0:
#4.          print(n, '不是素数')
#5.          break
#6.  else:
#7.      print(n, '是素数')
```

测试一的结果：

```
请输入一个整数：13↙
13 是素数
```

测试二的结果：

```
请输入一个整数：24↙
24 不是素数
```

5.3.2 continue 语句

循环体中使用 continue 语句，可以提前结束本次循环体代码的执行，不再执行本语句后循环体中的其他语句，跳回到循环结构首行，重新判断循环条件，并根据重判结果决定是否继续循环。

与 break 语句类似，continue 语句一般也是和 if 语句结合使用。continue 语句只是结束本次循环，而不终止整个循环的执行；break 语句则是使整个循环提前终止。

【例 5-8】 改用 continue 语句找所有三位水仙花数。

```
#1.  for i in range(100, 1000):
#2.      a, temp = divmod(i, 100)
#3.      b, c = divmod(temp, 10)
#4.      if a ** 3 + b ** 3 + c ** 3 != i:
#5.          continue
#6.      print(i)
```

运行程序，结果如下：

```
153
370
371
407
```

程序中#4 行的条件成立时，就会执行#5 行中的 continue 语句，并让程序转回到循环结构的开始，即#6 行的代码被跳过了。而当#4 行中的条件不成立时，#6 行的 print()函数会被执行，即输出水仙花数。

5.3.3 pass 语句

pass 语句又称空语句，执行 pass 语句时不做任何操作。需要使用 pass 语句的情况：

- 用 pass 语句保证控制结构的完整性。在某些情况下，代码的某处必须要有至少一个语句，但是实际无事可做，这时就可以使用 pass 语句，相当于一个占位语句。
- 模块化设计时，用 pass 语句占位。Python 提供了模块设计手段，用户可以先建立程序的主模块，而每个模块代码的细化需要逐渐完成，在初期可以先在各模块中用 pass 语句占位，然后再一个一个模块细化代码。

5.4　循环的嵌套

循环语句是允许嵌套的。一个循环体内的语句中包含另一个循环语句，称为循环的嵌套。根据处在包含关系中的不同位置，一个循环嵌套中包括外层循环和内层循环。内层循环可以继续循环嵌套，即为多层循环。较常用的循环嵌套是双重循环和三重循环。

使用循环嵌套应注意以下几个问题：

(1) 外循环每执行一次，内循环就要执行一个完整的循环；

(2) 可以使用不同的循环语句相互嵌套，来解决复杂问题；

(3) 使用 break 语句，只能跳出 break 语句所在那层的循环，而不能跳出整个嵌套循环中的所有层循环；

(4) 嵌套循环中，应注意语句的缩进，错误的缩进会使语句属于错误的循环层。

【例 5-9】 编制程序，打印如下九九乘法表。

1＊1＝1

1＊2＝2 2＊2＝4

1＊3＝3 2＊3＝6　3＊3＝9

1＊4＝4 2＊4＝8　3＊4＝12 4＊4＝16

1＊5＝5 2＊5＝10 3＊5＝15 4＊5＝20 5＊5＝25

1＊6＝6 2＊6＝12 3＊6＝18 4＊6＝24 5＊6＝30 6＊6＝36

1＊7＝7 2＊7＝14 3＊7＝21 4＊7＝28 5＊7＝35 6＊7＝42 7＊7＝49

1＊8＝8 2＊8＝16 3＊8＝24 4＊8＝32 5＊8＝40 6＊8＝48 7＊8＝56 8＊8＝64

1＊9＝9 2＊9＝18 3＊9＝27 4＊9＝36 5＊9＝45 6＊9＝54 7＊9＝63 8＊9＝72 9＊9＝81

解题思路：程序使用双重循环，使用两个循环变量 i 和 j，i 用于控制外循环，即控制打印九九乘法表的行数，j 用于控制内循环，即控制九九乘法表每一行打印的内容。

```
#1.  for i in range(1, 10):
#2.      for j in range(1, i + 1):
#3.          s = str(j) + '*' + str(i) + '=' + str(i * j)
#4.          print(s, end = '')
#5.      print()
```

请仔细体会一下为什么＃3 行要把 str(j)放在 str(i)的前面？

【例 5-10】 求 100 以内的全部素数。并将找到的素数按每行 5 个的形式输出在屏幕上。

解题思路：程序使用二重循环，使用两个循环变量 i 和 j，i 用于控制外循环，依次判断 2～99 是否为素数，j 用于控制内循环，判断 i 是否能被 j 整除。

```
#1.  n = 0
#2.  for i in range(2, 100):
#3.      for j in range(2, i):
#4.          if i % j == 0:
#5.              break
```

```
#6.       else:
#7.           print(i, end = '\t')
#8.           n += 1                #n用来记录是否需要换行,每发现一个素数n便加1
#9.           if n == 5:            #若n值为5,则需要换行,并将n清零后,重新计数
#10.              n = 0
#11.                print()
```

运行程序,结果如下:

```
2   3   5   7   11
13  17  19  23  29
31  37  41  43  47
53  59  61  67  71
73  79  83  89  97
```

5.5 循环结构程序举例

循环结构是在主程序顺序结构中,部分使用了循环结构语句,使得程序执行时根据条件选择性地反复执行某一段代码。

【例 5-11】 设计采用欧几里得算法求两个自然数的最大公约数的程序。

欧几里得算法求最大公约数的原理如下:假设用 gcd(m,n)表示 m 和 n 的最大公约数,则 gcd(m,n)必定等于 gcd(n,m%n),因为 m%n 的结果必定小于 n,则求最大公约数的等价数对(n,m%n)会越来越小,直到 m%n 的结果为 0。而 m%n==0 则意味着 n 就是最初两个数的最大公约数。例如:

gcd(14,35)→gcd(35,14)→gcd(14,7)→gcd(7,0)

14%35　　35%14　　14%7

程序代码如下:

```
#1.  m, n = eval(input("请输入一对整数m,n:"))
#2.  while n:
#3.      m, n = n, m % n
#4.  print(m)
```

测试一的结果:

```
请输入一对整数m,n:14,35↙
7
```

测试二的结果:

```
请输入一对整数m,n:32,15↙
1
```

【例 5-12】 猜数游戏。随机生成 20 以内的整数，用户输入整数猜测该数，程序提示输入的数是偏大、偏小，还是正确。

解题思路：本题要用到随机生成数，需要先导入模块 random，并利用其中的 randint() 函数生成指定范围的随机整数。

```
#1.  import random
#2.  goal = random.randint(1, 20)
#3.  x = int(input('你猜:'))
#4.  n = 0
#5.  while x != goal:
#6.      n += 1
#7.      if x > goal:
#8.          x = int(input('偏大!再猜:'))
#9.      elif x < goal:
#10.         x = int(input('偏小!再猜: '))
#11. print('恭喜你,猜对了!共猜了{}次'.format(n))
```

某一次的测试结果如下：

```
你猜:10 ↙
偏大!再猜:5 ↙
偏小!再猜:8 ↙
偏小!再猜:9 ↙
恭喜你,猜对了!共猜了 3 次
```

【例 5-13】 买房计划。2011 年张三年收入为 8 万元，其中 70%用于存款购房，此时的房价为 50 万元，张三准备贷款购房，首付 50%。假定张三的年收入每年以固定速度增长，房价也以每年 10%的固定速度增长。张三如果想在 6 年内买房，他的年收入增长最少要达到多少，要求增长率精确到小数点后两位，即精确到 xx.xx%。

解题思路：假设收入增长率为 r，则初始化 r=1，然后令 r=r+0.0001，计算张三 6 年内能否存够首付，不断修改 r 值直到能存够首付为止。

```
#1.  r = 1
#2.  while True:
#3.      sr = 8 * 0.7                          #首年的存入金额
#4.      sum = 0                               #总存款
#5.      fj = 50 * 0.5                         #第一年房子的首付
#6.      r += 0.0001
#7.      for i in range(0, 6):
#8.          sum += sr                         #年底的总存款
#9.          sr = sr * r                       #下一年度的存入的金额
#10.         fj = fj * 1.1                     #下一年度的房子首付金额
#11.     if sum >= fj:
#12.         print('最低年收入增长率是{:.2%}'.format(r - 1))
#13.         break
```

运行程序，结果如下：

最低年收入增长率是 10.98 %

5.6 习　　题

1. Python 有哪些循环控制语句？简述各自的执行过程。

2. 简述 break 语句和 continue 语句的作用。

3. 假设 abcd 是一个 4 位整数，将它分成两段，即 ab 和 cd，使之相加求和后再平方后的值等于原数 abcd。编写程序，找出满足该关系的所有 4 位整数。

4. 一个球从某个高度 h 米处落下，每次落地反弹回原来高度的一半，再落下。编写程序，求该球第 10 次落地时，共经过了多少米？第 10 次反弹高度是多少米？

5. 某镇现有人口 x 万(x 为浮点数)，按每年 0.1%的增长速度，n 年后将有多少人？输出结果时保留小数点后 4 位。

6. 现有某路口连续一小时内的 60 个监测数据，每个监测数据是一分钟内通过路口的车辆数。假定一分钟内的车辆数超过 30 辆视为繁忙，低于 30 辆视为空闲。现编程统计，这一小时内，该路口的空闲状态最长持续了多少分钟？注：检测数据可以输入，也可以用随机函数生成。

7. 任意整数 a 和 b，求 a/b 转换为小数后，小数点后第 n 位的数字是多少？编写程序，输入 a、b 和 n 后，输出第 n 位的数字。

8. 编程统计某个给定范围[a, b]的所有整数中，数字 3 出现的次数。

9. 输出所有满足以下条件的 3 位整数：该数是素数，该数的个位和十位数之和被 10 除，所得余数正好就是该数的百位数。例如，293 是素数并且(3＋9)被 10 除的余数是 2，所以 293 是满足条件的 3 位数整数。

10. 编写程序，输入一个正整数和一个 n，求比该正整数的第 n 个小的素数。例如，输入 60，n＝3，则应输出 47。

11. 已知大鱼 5 元一条，中鱼 3 元一条，小鱼 1 元三条。编写程序，用 100 元买 100 条鱼，求能买大鱼、中鱼、小鱼各多少条。

12. 编写程序，统计 100 元人民币兑换成 1 元、2 元和 5 元的所有兑换方案个数。