第 3 章 程序的控制结构

导学

本章主要介绍 Python 程序设计中的程序控制结构，通过本章的学习读者可以掌握 Python 程序设计的基本控制结构及程序设计技巧。

了解：3 种基本程序控制结构的流程图，以及程序设计的基本方法。

掌握：3 种基本程序控制结构，并能够进行具体程序的设计。

Python 程序设计有 3 种基本程序控制结构：顺序结构、分支结构和循环结构。程序设计通常以顺序结构为主框架，程序语句按先后顺序逐条命令执行。当程序中需要判断某些条件或多次重复处理某些事件时，可以使用分支结构或循环结构控制程序的执行流程。

3.1 顺 序 结 构

顺序结构是程序设计的基本架构结构，在一个没有分支结构和循环结构的程序中，它按程序文件中命令语句的先后顺序，逐条依次执行。顺序结构程序流程图如图 3-1 所示，在图中，有一个程序入口、一个程序出口，程序运行过程中依次执行语句 1 和语句 2。

图 3-1 顺序结构的流程图

下面是一个顺序结构程序的例子。

【例 3-1】 BMI(Body Mass Index)指数，即体质指数，计算公式为体质指数(BMI)＝体重(kg)÷身高^2(m)，是目前国际上常用的衡量人体胖瘦程度以及是否健康的一个标准。编写一个求体质指数的程序，该程序为顺序结构设计。代码如下：

```
w = float(input("请输入您的体重(kg):"))          # 输入体重值(以 kg 为单位)
h = float(input("请输入您的身高(m):"))           # 输入身高值(以 m 为单位)
B = w/h ** 2                                     # 计算 BMI 指数
print("您的 BMI 指数为",B)                        # 输出 BMI 指数
```

上述代码的输出结果如图 3-2 所示。

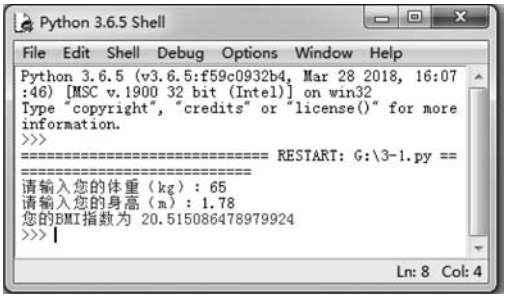

图 3-2　例 3-1 运行结果

3.2　分 支 结 构

分支结构就是按照设计好的条件，经过判断后有选择地执行程序中的某些特定语句序列，或使程序跳转到指定语句后继续运行。在 Python 程序设计中，分支结构包括单分支结构、双分支结构和多分支结构。

3.2.1　if 语句

if 语句属于单分支结构，其语句格式如下：

```
if 表达式:
    语句序列
```

单分支结构程序功能：程序运行到 if 语句时，判断条件表达式是否成立，如果条件表达式的值为 True，则执行内嵌的语句序列；如果条件表达式的值为 False，则不做任何操作。单分支结构的流程图如图 3-3 所示。

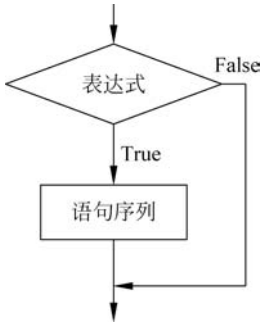

图 3-3　单分支结构的流程图

下面是一个单分支结构程序的例子。

【例 3-2】 能被 2 整除的数是偶数。编写一个判断整数是否是偶数的程序，该程序为单分支结构设计。代码如下：

```
x = int(input("请输入一个整数:"))                    # 输入一个整数
if x % 2 == 0:                                       # 判断 x 是否为偶数
    print("这个数是偶数")                             # 条件表达式值为 True,输出该数是偶数
```

提示：在 Python 程序设计中，通过命令行的缩进标识语句序列的开始与结束。如例 3-2 中 if 语句所包含的语句序列为该程序中的第 3 条命令，该条命令起始位置比第 2 条命令的起始位置向右缩进 4 个空格。

上述代码的输出结果如图 3-4 所示。

图 3-4　例 3-2 运行结果

【例 3-3】 编写一个判断整数是否是奇数的程序。代码如下：

```
x = int(input("请输入一个整数:"))                    # 输入一个整数
if x % 2!= 0:                                        # 判断 x 是否为奇数
    print("这个数是奇数")
```

提示："！＝"为比较两个对象是否不相等。

3.2.2　if...else 语句

if...else 语句属于双分支结构，其语句格式如下：

```
if 表达式:
   语句序列 1
else:
   语句序列 2
```

双分支结构程序功能：程序运行到 if 语句时，判断条件表达式是否成立。如果条件表达式的值为 True，则执行内嵌的语句序列 1；如果条件表达式的值为 False，则执行 else 后面内嵌的语句序列 2。双分支结构的流程图如图 3-5 所示。

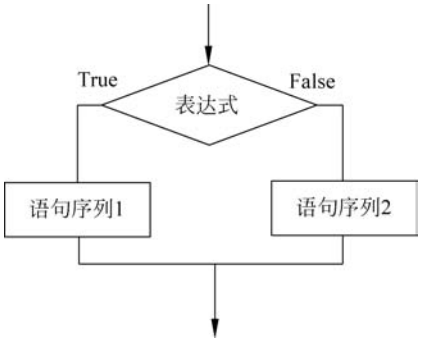

图 3-5　双分支结构的流程图

下面是一个双分支结构程序的例子。

【例 3-4】 整数中，能被 2 整除的数是偶数，不能被 2 整除的数是奇数。编写一个判断整数是偶数还是奇数的程序，该程序为单分支结构设计。代码如下：

```
x = int(input("请输入一个整数:"))                    # 输入一个整数
if x % 2 == 0:                                       # 判断 x 是否为偶数
    print("这个数是偶数")                             # 条件表达式值为 True,输出该数是偶数
else:
    print("这个数是奇数")                             # 条件表达式值为 False,输出该数是奇数
```

上述代码的输出结果如图 3-6 所示。

图 3-6 例 3-4 运行结果

例 3-4 程序也可以根据例 3-2 和例 3-3 设计为单分支结构。代码如下：

```
x = int(input("请输入一个整数:"))                    # 输入一个整数
if x % 2 == 0:                                       # 判断 x 是否为偶数
    print("这个数是偶数")                             # 条件表达式值为 True,输出该数是偶数
if x % 2!= 0:                                        # 判断 x 是否为奇数
    print("这个数是奇数")
```

3.2.3 if...elif...else 语句

if...elif...else 语句属于多分支结构，其语句格式如下：

```
if 表达式 1:
   语句序列 1
elif 表达式 2:
   语句序列 2
   …
elif 表达式 n:
   语句序列 n
else
   语句序列 n + 1
```

多分支结构程序功能：程序运行到 if 语句时，判断条件表达式 1 是否成立。如果条件表达式的值为 True，则执行内嵌的语句序列 1；如果条件表达式的值为 False，则依次判断每个 elif 条件表达式是否成立，如果表达式值为 True，则运行其下面的语句序列，如果所有的条件表达式都不成立，则执行 else 后面的语句序列。多分支结构的流程图如图 3-7 所示。

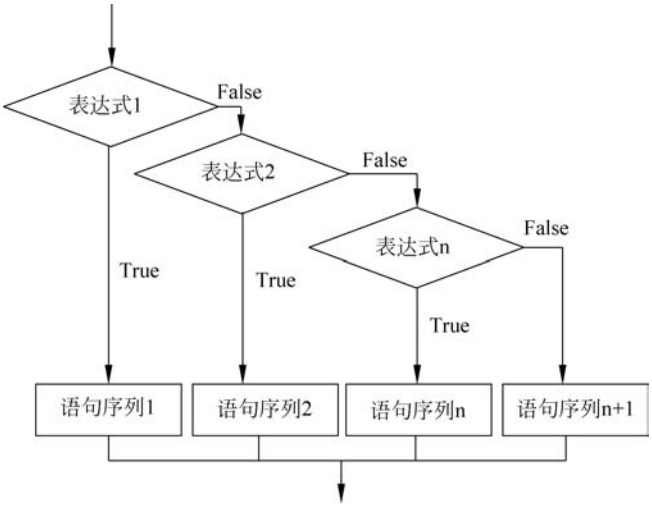

图 3-7　多分支结构的流程图

下面是一个多分支程序结构程序的例子。

【例 3-5】 例 3-1 中用 Python 实现了 BMI 指数的计算，成年人 BMI 数值的划分标准如下：BMI 低于 18.5，体重过轻；BMI 位于 18.5～23.9，体重正常；BMI 位于 24～27，体重过重；BMI 位于 28～32，肥胖；BMI 高于 32，非常肥胖。编写一个根据 BMI 指数判断体重情况的程序，该程序为多分支结构设计。代码如下：

```
B = float(input("请输入您的 BMI 指数:"))                    # 输入 BMI 指数
# 通过多分支结构判断体重等级
if B > 32:                                                  # BMI 高于 32,非常肥胖
    print("您的体重评定等级是非常肥胖")
elif B <= 32 and B >= 28:                                   # BMI 位于 28～32 之间,肥胖
    print("您的体重评定等级是肥胖")
elif B <= 27 and B >= 24:                                   # BMI 位于 24～27 之间,体重过重
    print("您的体重评定等级是过重")
elif B <= 23.9 and B >= 18.5:                               # BMI 位于 18.5～23.9 之间,体重正常
    print("您的体重评定等级是正常")
else:                                                       # BMI 低于 18.5,体重过轻
    print("您的体重评定等级是过轻")
```

上述代码的输出结果如图 3-8 所示。

图 3-8　例 3-5 运行结果

上面的程序也可以通过分支结构嵌套实现，代码如下：

```
B = input("请输入您的 BMI 指数: ")                    # 输入 BMI 指数
# 通过分支语句嵌套判断体重等级
if B > 32:                                          # BMI 高于 32,非常肥胖
    print("您的体重评定等级是非常肥胖")
    if B >= 28:                                     # BMI 位于 28～32 之间,肥胖
        print("您的体重评定等级是肥胖")
        if B >= 24:                                 # BMI 位于 24～27 之间,体重过重
            print("您的体重评定等级是过重")
            if B >= 18.5:                           # BMI 位于 18.5～23.9 之间,体重正常
                print("您的体重评定等级是正常")
                   elif:                            # BMI 低于 18.5,体重过轻
                       print("您的体重评定等级是过轻")
```

3.2.4 pass 语句

在 Python 程序设计中，pass 语句的作用相当于空语句，当暂时没有确定如何实现功能时，可以使用 pass 语句来进行“占位”。例如以下程序：

```
x = 0
a = input("输入 a 的值")
b = input("输入 b 的值")
if a < b:
   pass                        # 如果 a 的值小于 b 的值,那么执行 pass 语句
else:
   x = a                       # 如果 a 的值大于等于 b 的值,那么将 a 的值赋给 x
   print(x)
```

3.2.5 try...except 语句

在 Python 程序设计中，try...except 语句可以用来进行对程序运行异常的检测与处理。try...except 语句格式如下：

```
try:
   被检测的语句序列
except <异常名>:
   异常处理语句序列
```

例如以下程序：

```
try:
    x = 1/0
except ZeroDivisionError:                    # 除数为 0 异常
    print("除数为 0")
```

3.3 循 环 结 构

循环结构是指程序在执行的过程中,其中的某段语句序列被重复执行若干次。Python 程序设计提供了 for 循环和 while 循环。

3.3.1 while 语句

while 语句即 while 循环,其语句格式如下:

```
while 表达式:
    语句序列
```

while 循环程序结构程序功能:程序每次运行到“while 表达式:”语句时,判断条件表达式是否成立。如果条件表达式的值为 True,则反复执行循环体内的语句序列;如果条件表达式的值为 False,则循环结束。while 循环结构的流程图如图 3-9 所示。

下面是一个 while 循环程序结构程序的例子。

【例 3-6】 编写一个计算 1+2+3+…+100 的程序,该程序用 while 循环结构设计。代码如下:

```
total = 0                          #变量 total 用来保存最终的和
number = 1                         #变量 number 用来保存 1～100 的整数
while number <= 100:               #求 1～100 的和
    total = total + number
    number = number + 1
print("1 到 100 之和为:",total)
```

上述代码的输出结果如图 3-10 所示。

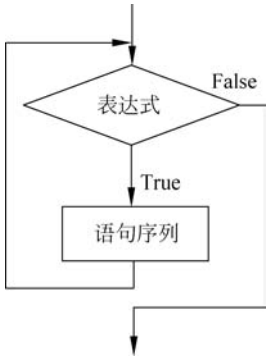

图 3-9 while 循环结构的流程图

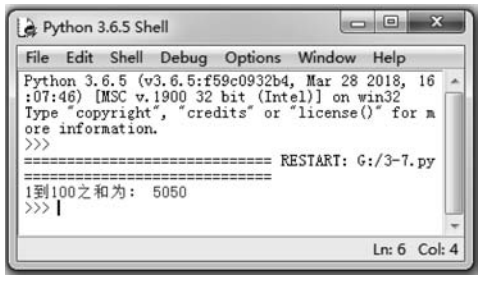

图 3-10 例 3-6 运行结果

注意:如果程序每次运行到 while 语句时,表达式的值都为真,则该程序为死循环,程序运行将无法结束。例如以下程序:

```
total = 0
number = 1
```

```
#由于 number 始终等于 1,表达式"number <= 100"始终为真,该循环为死循环
while number <= 100:
    total = total + number
print("1 到 100 之和为:",total)
```

3.3.2 for 语句

for 语句即 for 循环,其语句格式如下:

```
for 变量 in 序列:
   语句序列
```

for 循环程序结构程序的功能:变量遍历序列中的每个值。每取一个值,如果这个值在序列中,则执行语句序列,返回后,再取下一个值,再判断,直到遍历完成,退出循环。序列可以是列表、元组或字符串。

下面是一个 for 循环结构程序的例子。

【例 3-7】 编写一个计算 1+2+3+…+10 和的程序,该程序用 for 循环结构设计。代码如下:

```
total = 0                                  #变量 total 用来保存最终的和
for x in [1,2,3,4,5,6,7,8,9,10]:           #变量 x 用来循环控制
    total = total + x
print("1 到 10 之和为:",total)
```

上述代码的输出结果如图 3-11 所示。

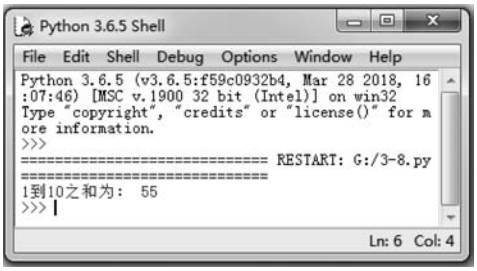

图 3-11 例 3-7 运行结果

例 3-7 程序也可以根据 range()函数来设计(关于函数见第 4 章)。代码如下:

```
total = 0                                  #变量 total 用来保存最终的和
for x in range(1,11):                      #变量 x 用来循环控制
    total = total + x
print("1 到 10 之和为:",total)
```

提示:

范围函数 range(start,stop[,step])所表示的计数范围从 start 开始,到 stop−1 结束,step 为计数变化的步长值,默认为 1。例如:上面程序中的 range(1,10)的步长值为 1,表示 1～10 的整数。

3.3.3 循环嵌套结构

Python 程序设计允许在一个循环体中嵌入另一个循环体，对于 while 循环和 for 循环，两种循环语句可以自身嵌套，也可以相互嵌套，嵌套的层次没有限制。循环嵌套执行时，每执行一次外层循环，其内层循环必须在循环执行结束后，才能进入到外层循环的下一次循环中。在设计嵌套程序时，注意在一个循环体内包含另一个完整的循环结构。

【例 3-8】 编写一个输出由“*”组成的直角三角形(9 行)的程序，该程序为循环嵌套结构设计。代码如下：

```
for i in range(1,10):                          #range(1,10)表示 1～9 的整数
    for j in rang(1,i+1):                      #range(1,i+1)表示 1～i 的整数
        print("*",end="\t")                    #行中每个值以"\t"隔开,"\t"为制表符
    print()                                    #换行
```

【例 3-9】 参照例 3-8，可以编写一个输出“9×9 乘法表”的程序。代码如下：

```
for i in range(1,10):                          #range(1,10)表示 1～9 的整数
    for j in rang(1,i+1):                      #range(1,i+1)表示 1～i 的整数
        print(i*j,end="\t")                    #行中每个值以"\t"隔开,"\t"为制表符
    print()                                    #换行
```

上述代码的输出结果如图 3-12 所示。

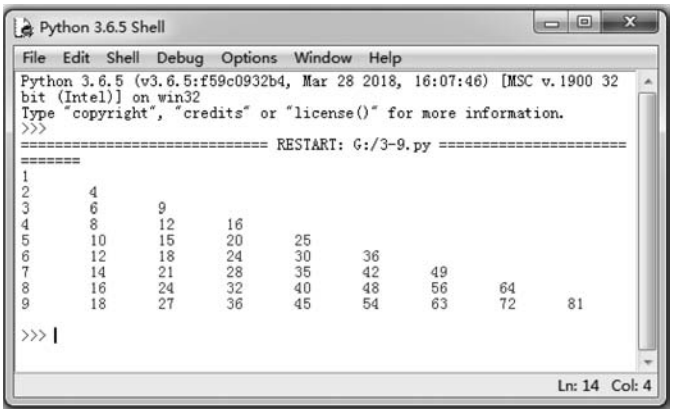

图 3-12 例 3-9 运行结果

提示：例 3-9 也可以用 while 循环嵌套来实现。代码如下：

```
i=1
while i<=9:                                    #外层循环
    j=1
    while j<=i:                                #内层循环的终点是一个变化的量
        print(i*j,end="\t")
        j=j+1                                  #内层循环变量的变化
    i=i+1                                      #外层循环变量的变化
    print()
```

3.3.4 break 和 continue 语句

break 语句和 continue 语句都可以放在循环体内，通常放在循环体内的分支语句结构中。break 语句的作用是结束当前循环，使得整个循环提前结束；continue 语句的作用是忽略 continue 之后的语句，提前回到下一次循环。具体的流程变化如图 3-13 所示。

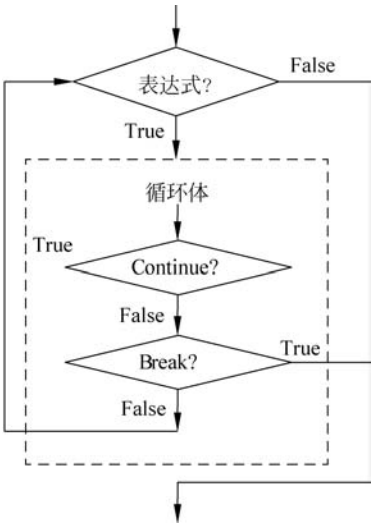

图 3-13　break 语句和 continue 语句流程示意图

break 语句和 continue 语句的用法如下：

```
i = 1
while i < 10:
    i = i + 1
    if i % 2!= 0:                    # 奇数时跳过输出
        continue
    print(i)                         # 偶数时输出
i = 1
while 1:                             # 循环条件为真
    print(i)
    i = i + 1
    if i > 10:                       # 当 i 值大于 10 时,循环结束
        break
```

注意：当程序设计为死循环，然后中途判断用 break 退出循环时，称为半路循环。例如以下程序：

```
a = - 1
while 1:
    a += 1
    if a == 10:
        break
```

本 章 小 结

本章主要介绍了 Python 程序设计中的顺序结构、分支结构和循环结构 3 种控制结构，以及 3 种结构中可以用到的 pass 语句、break 语句和 continue 语句。通过对本章的学习，能够掌握 Python 程序设计的基本语法和基本思路，为后续章节的学习打下良好的基础。