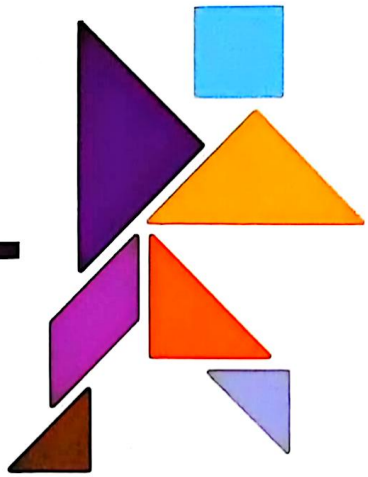
21世纪高等学校计算机
应用技术系列教材

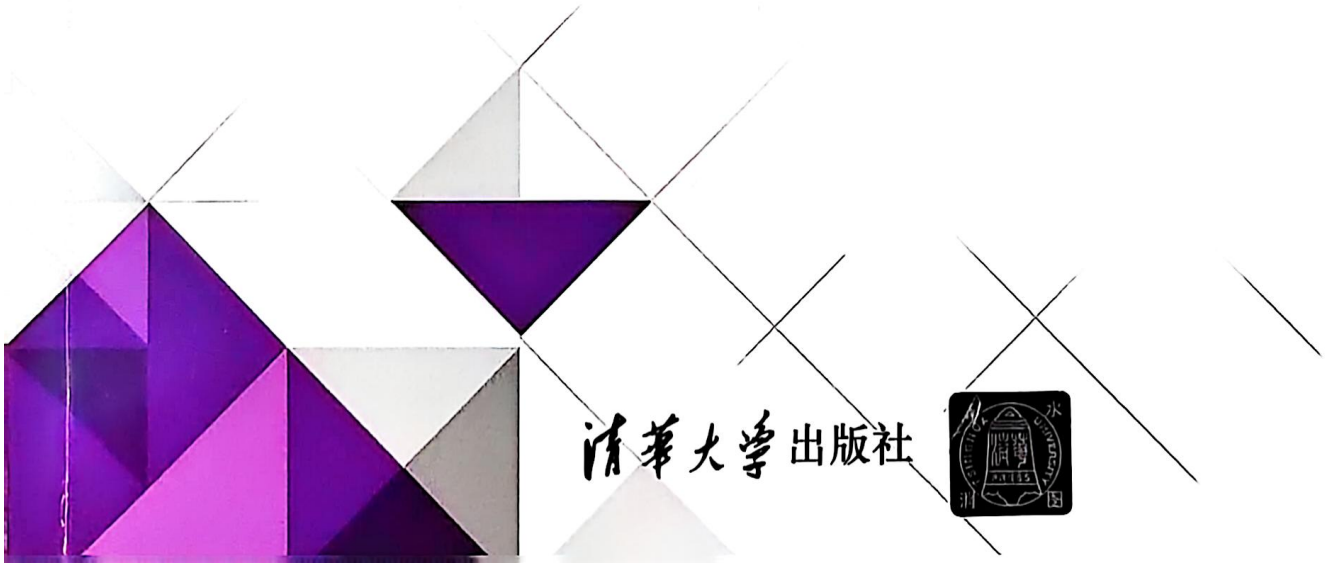
Python程序设计

（思政版）微课视频版



◎ 王霞 王书芹 郭小荟 梁银 刘小洋 宋杰鹏 魏思政 编著

《教学课件》 《教学大纲》 640分钟视频讲解 《知识导图》 《思政元素》
《程序源码》 《习题答案》 《丰富实例》 《实战任务》



清华大学出版社

目　录

```
        y = 2 * x - 1
    else:
        y = 3 * x - 11
print("x = " + str(x) + ",f(x) = " + str(y))
```

运行结果：

input x: 0.5 x = 0.5,f(x) = 0.5	input x: 5 x = 5.0,f(x) = 9.0	input x: 20 x = 20.0,f(x) = 49.0

当然，实例 3.8 也可以不使用嵌套语句实现，而使用多分支结构实现。

```
x = float(input("input x: "))
if x <= 1:
    y = x
elif x < 10:
    y = 2 * x - 1
else:
    y = 3 * x - 11
print("x = " + str(x) + ",f(x) = " + str(y))
```

很明显，多分支结构比分支嵌套可读性更强，Python 之禅中有一句话："Flat is better than nested."，扁平化总比嵌套好，所以能扁平化时尽量不要用嵌套。

3.3 循环结构

如果需要重复执行某条或者某些指令，例如"中国诗词大赛"中的"飞花令"，选手要根据给定的关键字，在给定的时间内轮流背诵含有关键字的诗句，直至时间结束。重复执行类似动作就是循环结构。Python 提供两种循环结构语句：while 循环和 for…in 循环。前者根据条件返回值的情况决定是否执行循环体，后者采用遍历的形式指定循环范围。要更加灵活地操纵循环语句的流向，还需要使用 break、continue 和 pass 等语句。

3.3.1 while 循环


视频讲解

while 循环也称为无限循环，是由条件控制的循环运行方式，一般用于循环次数难以提前确定的情况。while 循环的语法格式为：

```
while 条件表达式:
    循环体
[else:
    语句块]
```

其中，"条件表达式"可以是任何非空或者非零的表达式，"循环体"可以是单条语句或语句块，方括号内的 else 子句可以省略。

流程图如图 3.5 所示。

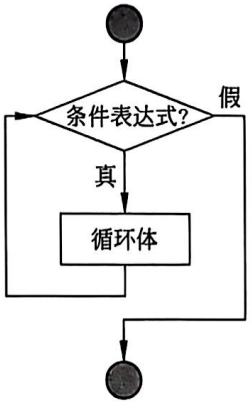


图 3.5 while 循环结构流程图

执行过程为先判断条件表达式,如果结果为真,则执行循环体,继续进行条件判断;否则循环结束。

【实例 3.9】 求 $\sum_{i=1}^{100} i$。

算法分析:设计循环算法需要考虑循环三要素:循环初值、结束条件以及增量(步长)。本例中,循环变量为 i,初值为 1,结束条件或者终值为 100,步长为 1。另外还需要一个变量存储累加和,其初值为 0。对应的结构流程图如图 3.6 所示。

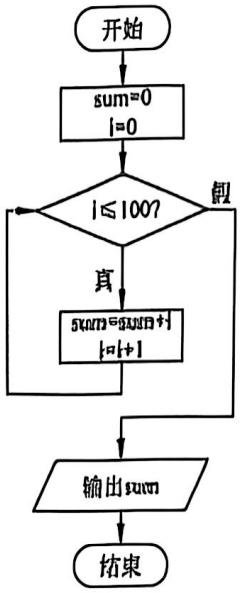


图 3.6 求累加和结构流程图

源代码:

```
1  sum, i = 0, 0
2  while i <= 100:
3      sum += i
4      i = i + 1
5  print("sum = " + str(sum))
```

运行结果:

```
sum = 5050
```

拓展:1+3+…+99、$\prod_{i=1}^{100} i$、$\sum_{i=1}^{n} i$、$\sum_{i=m}^{n} i$ 等类似累加和或者累乘积的计算。

【实例 3.10】 求若干个学生某门课程的平均成绩。

算法分析:循环变量为学生人数,初值为 0,终值为学生个数 n,步长为 1。循环体累加每个学生的成绩,循环结束后求成绩的平均值。其结构流程图如图 3.7 所示。

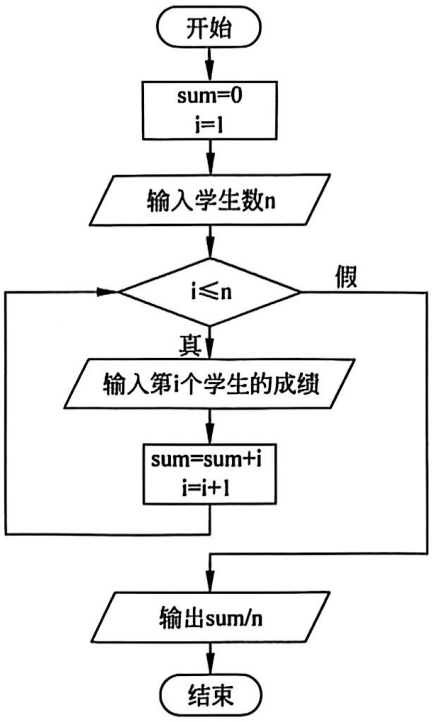


图 3.7　求平均成绩结构流程图

源代码：

```
sum,i = 0,1
n = int(input("请输入学生人数："))
while i <= n:
    score = float(input("NO " + str(i) + ": "))
    sum += score
    i = i + 1
print("平均成绩为 " + str(sum / n))
```

运行结果：

```
请输入学生人数：5
NO 1: 100
NO 2: 85.5
NO 3: 98.7
NO 4: 95
NO 5: 65
平均成绩为 88.84
```

循环结构中也可以使用 else 子句，用于表示不满足循环条件时程序的执行流程。

【实例 3.11】　循环结构使用 else 子句。

源代码：

```
count = 0
while count < 5:
    print(str(count) + " is less than 5.")
```

```
    count = count + 1
else:
    print(str(count) + " is not less than 5.")
```

运行结果：

```
0 is less than 5.
1 is less than 5.
2 is less than 5.
3 is less than 5.
4 is less than 5.
5 is not less than 5.
```

可见，当满足循环条件“count＜5”时，执行循环体，当不满足循环条件时，执行“print(str(count)＋" is not less than 5.")”。

视频讲解

3.3.2 for…in 循环

Python 提供的另一种循环结构是 for…in 循环。Python 提供的 for 循环语句与 Java、C++等编程语言提供的 for 语句不同，更像是 shell 或是脚本语言中的 for 循环，可以遍历如列表、元组、字符串等序列成员，也可以用在列表解析和生成器表达式中。

1. 使用序列项迭代序列对象

通过 for…in 循环可以迭代序列对象的所有成员，并在迭代结束后，自动结束循环，其语法如下：

```
for iterating_var in list:
    循环体
```

其中，iterating_var 为迭代变量，list 为序列(字符串、列表、元组、字典、集合)。执行时，迭代变量依次取序列中元素的值，直至取完，循环退出。

对应的结构流程图如图 3.8 所示。

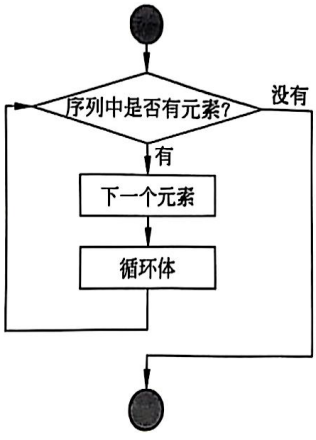


图 3.8　迭代序列 for…in 循环结构流程图

【实例 3.12】 统计字符串中各类字符的个数。

算法分析：使用迭代变量遍历序列(字符串)中的每一个元素，分别判断所属类型，并将对应个数加 1，直至遍历结束。结构流程图如图 3.9 所示。

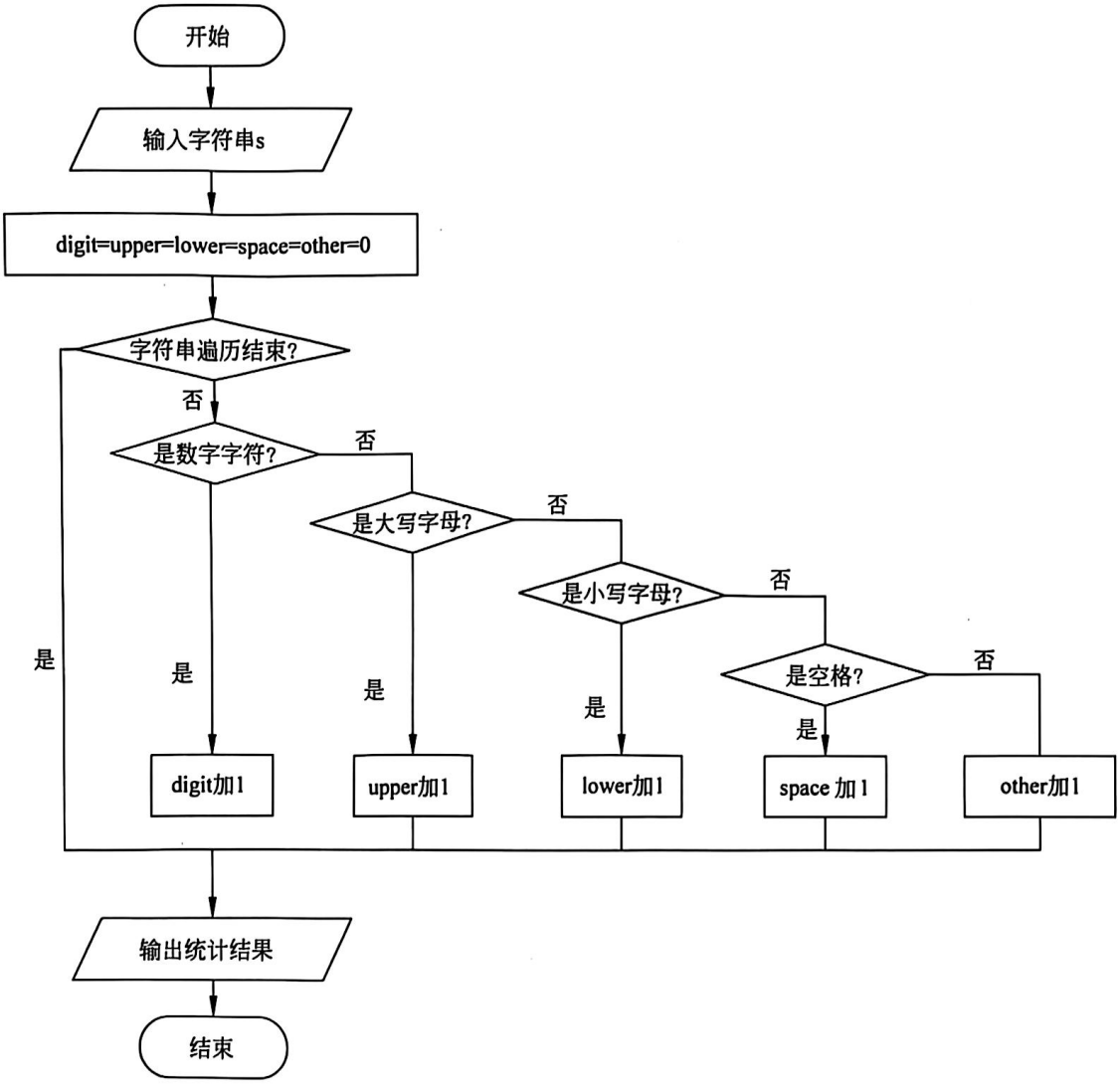


图 3.9 统计字符串中各类字符个数的结构流程图

源代码：

```
1  s = input("请输入一个字符串：")
2  digit,upper,lower,space,other = 0,0,0,0,0   #数字字符,大写字符,小写字符,空格字符,
                                               #其他字符的个数
3  for i in s:
4      if i >= '0' and i <= '9':               #判断 i 是否为数字字符
5          digit = digit + 1
6      elif i >= 'A' and i <= 'Z':             #判断 i 是否为大写字母
7          upper = upper + 1
8      elif i >= 'a' and i < 'z':              #判断 i 是否为小写字母
```

```
        lower = lower + 1
    elif i == ' ':   #判断 i 是否为空格字符
        space = space + 1
    else:   #其他字符
        other = other + 1
print("数字字符: %d\n 大写字母: %d\n 小写字母: %d\n 空格字符: %d\n 其他字符: %d
\n" % (digit,upper,lower,space,other))
```

运行结果:

```
请输入一个字符串: Life is short, we need Python!
数字字符: 0
大写字母: 2
小写字母: 21
空格字符: 5
其他字符: 2
```

其中,实现字符分类的循环体也可用内置函数代替,如:

```
if i.isdigit( ):          #判断 i 是否为数字字符
    digit = digit + 1
elif i.isupper( ):        #判断 i 是否为大写字母
    upper = upper + 1
elif i.islower( ):        #判断 i 是否为小写字母
    lower = lower + 1
elif i.isspace( ):        #判断 i 是否为空格字符
    space = space + 1
else:
    other = other + 1
```

内置函数 i.isdigit()、i.isupper()、i.islower()、i.isspace()分别用来判断 i 是否为数字字符、大写字母、小写字母和空格字符。

2. 使用序列索引迭代序列对象

在 for…in 循环结构中,也可以使用序列索引来遍历列表,语法如下:

```
for index in range(len(list)):
    循环体
```

其中,index 为序列的索引项,内置函数 range()为计数函数,len()获取序列长度。

【实例 3.13】 统计字符串中各类字符的个数(range 版)。

```
s = input("请输入一个字符串: ")
digit = upper = lower = space = other = 0   #数字字符、大写字母、小写字母、空格字符
                                            #和其他字符的个数
for i in range(len(s)):
    if s[i].isdigit( ):                     #判断 i 是否为数字字符
        digit = digit + 1
```

```
    elif s[i].isupper( ):   #判断 i 是否为大写字母
        upper = upper + 1
    elif s[i].islower( ):   #判断 i 是否为小写字母
        lower = lower + 1
    elif s[i].isspace( ):   #判断 i 是否为空格字符
        space = space + 1
    else:
        other = other + 1
print("数字字符：%d\n大写字母：%d\n小写字母：%d\n空格字符：%d\n其他字符：%d
\n" % (digit, upper, lower, space, other))
```

运行结果：

```
请输入一个字符串：I am a student, I am 20 years old!
数字字符：2
大写字母：2
小写字母：20
空格字符：8
其他字符：2
```

使用 range()函数可以得到用来迭代的索引列表，使用索引下标“[]”可以方便快捷地访问序列对象。另外，还可以使用 range()函数实现类似 Java、C++等传统编程语言的 for 循环结构，即从循环三要素角度出发设计循环结构，语法格式为：

```
range([start,] end[, step = 1])
```

其中，range()函数会返回一个整数序列，可选项 start 为序列初值(循环变量初值)，end 为序列终止值(循环变量终值，且不含 end 本身)，可选项 step 为步长或增量，默认为 1。

【实例 3.14】 求 $\sum_{i=1}^{100} i$ (range 版)。

```
sum = 0
for i in range(1, 101, 1):
    sum = sum + i
print("sum = " + str(sum))
```

运行结果：

```
sum = 5050
```

显然，此时的 for…in 循环与 while 循环完全等价。

3. 使用枚举函数迭代序列对象

Python 内置函数 enumerate()用于将一个可遍历的数据对象(列表、元组或者字符串)组合成一个索引序列，同时列出数据和下标，一般用于 for…in 循环中。语法格式为：

```
for index, iterating_var in enumerate(list, start_index = 0):
    循环体
```

其中,index 返回索引计数,iterating_var 为与索引计数相对应的索引对象成员,list 为待遍历的序列对象,start_index 为返回的起始索引计数,默认值为 0。

【实例 3.15】 输出学生花名册。

```
1  name_list = ["李白", "孟浩然", "王维", "李绅"]  # name_list 的数据类型为列表 list
2  for index, name in enumerate(name_list):
3      print(index, name)
```

运行结果:

```
0 李白
1 孟浩然
2 王维
3 李绅
```

视频讲解

3.3.3 循环嵌套

允许在一个循环结构中嵌入另一个循环结构,称为循环嵌套。在 Python 中,for…in 循环结构和 while 循环结构都可以进行循环嵌套。如:

```
while condition_expression 1:
    for index in range(len(list)):
        循环体
```

不仅 for 循环结构可以嵌入到 while 循环结构中,while 循环结构也可以嵌入到 for…in 循环结构中,同样也可以根据自身需要任意嵌套。

【实例 3.16】 输出九九乘法表(下三角形)。

算法分析:从结构看,九九乘法表是二维形式,单重循环无法实现。从内容看,第一个乘数每行一致,第二个乘数同行每列依次加 1,故使用两重循环。外循环控制第一个乘数(迭代变量为 1~9),内循环控制第二个乘数。因为要求按照下三角形输出,所以内循环迭代变量只能为 1~i。结构流程图如图 3.10 所示。

源代码:

```
1  for i in range(1, 10):
2      for j in range(1, i+1):
3          print(str(i) + "*" + str(j) + "=" + str(i * j), end=" ")
4      print()      # 每行末尾换行
```

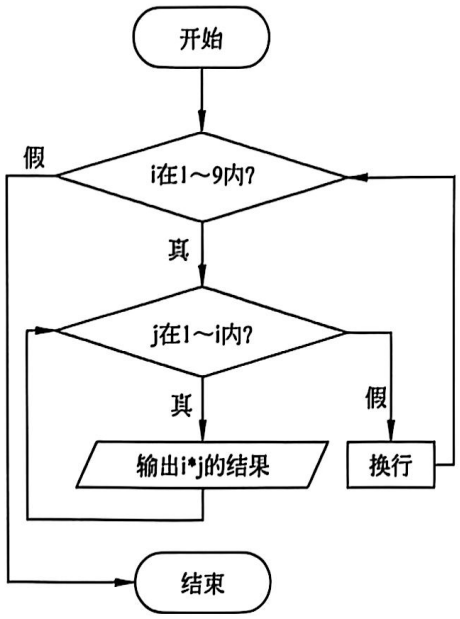


图 3.10　九九乘法表结构流程图

运行结果：

```
1 * 1 = 1
2 * 1 = 2 2 * 2 = 4
3 * 1 = 3 3 * 2 = 6 3 * 3 = 9
4 * 1 = 4 4 * 2 = 8 4 * 3 = 12 4 * 4 = 16
5 * 1 = 5 5 * 2 = 10 5 * 3 = 15 5 * 4 = 20 5 * 5 = 25
6 * 1 = 6 6 * 2 = 12 6 * 3 = 18 6 * 4 = 24 6 * 5 = 30 6 * 6 = 36
7 * 1 = 7 7 * 2 = 14 7 * 3 = 21 7 * 4 = 28 7 * 5 = 35 7 * 6 = 42 7 * 7 = 49
8 * 1 = 8 8 * 2 = 16 8 * 3 = 24 8 * 4 = 32 8 * 5 = 40 8 * 6 = 48 8 * 7 = 56 8 * 8 = 64
9 * 1 = 9 9 * 2 = 18 9 * 3 = 27 9 * 4 = 36 9 * 5 = 45 9 * 6 = 54 9 * 7 = 63 9 * 8 = 72 9 * 9 = 81
```

拓展：输出上三角九九乘法表、钻石形等图形。

【实例 3.17】 求 1!＋2!＋…＋20!。

算法分析：求 n! 需要用循环结构实现，累加和也需要用循环结构实现，故采用双重循环。外循环计算累加和，内循环求 n!。结构流程图如图 3.11 所示。

源代码：

```
sum = 0                          #累加和
for i in range(1, 21):           #外循环,用于累加和
    fact = 1                     #存放 n!
    for j in range(1, i + 1):    #内循环,用于计算 i!
        fact = fact * j
    sum = sum + fact
print("sum = " + str(sum))
```

运行结果：

```
sum = 2561327494111820313
```

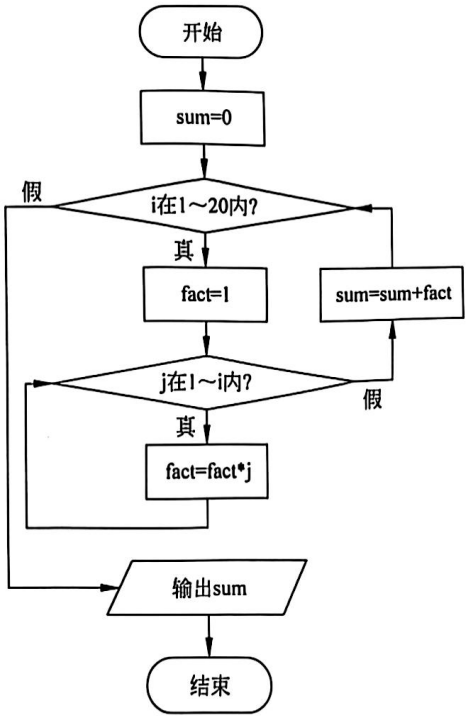


图 3.11　阶乘和结构流程图

观察运行过程可以发现,双重循环计算累加和时,每次都是从 1 开始计算 n!。事实上 n!=(n−1)!*n,故可以使用单重循环实现。优化后的代码:

```
sum = 0                  #累加和
fact = 1                 #存放 n!
for i in range(1, 21):
    fact = fact * i      #直接使用 n!=(n-1)!*n 来计算 n!
    sum = sum + fact
print("sum=" + str(sum))
```

循环结构中可以嵌套另一个循环结构,也可以嵌套选择结构,反之亦然。

【实例 3.18】 列出 1～200 的所有素数,要求每行输出 10 个数(标志变量版)。

算法分析:素数是一个只能被 1 和本身整除的自然数。判断 n 是否为素数的方法为依次除以 2～$\sqrt{n}$,如果能整除则不是素数。这个过程需要使用单重循环嵌套选择结构实现,而列出 1～200 的所有素数也需要用循环实现,故使用双重循环完成。流程图如图 3.12 所示。

源代码:

```
import math                  #math 模型库,sqrt()函数需要使用
count = 0                    #累计素数个数
for n in range(2, 201):      #外循环遍历 2～200
    i,flag = 2,True
```

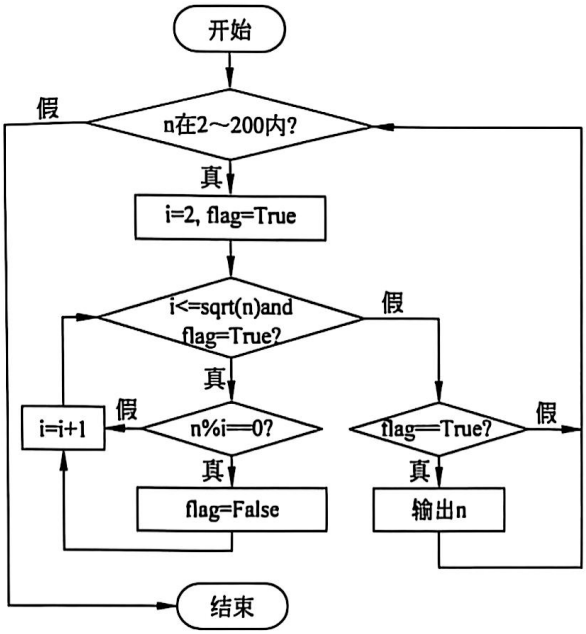


图 3.12　实例 3.18 结构流程图

```
while i <= math.sqrt(n) and flag:    # 内循环对每一个数进行是否为素数的判断
    if n % i == 0:                   # 如果能够整除,则不是素数,flag 置为 False
        flag = False
    i = i + 1
if flag:                             # 如果是素数,则输出
    count = count + 1
    print(n, end = "\t")
    if count % 10 == 0:              # 控制每行 10 个
        print()
```

运行结果：

```
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  101 103 107 109 113
127 131 137 139 149 151 157 163 167 173
179 181 191 193 197 199
```

说明：本实例中外循环使用 for…in 结构，内循环使用 while 结构，并在内循环中嵌入分支结构进行整除判断。引入标志变量 flag 来标志 n 是否为素数(默认值为 True)，一旦判断能够整除(即不是素数)，则修改 flag 值为 False，内循环条件执行为否，内循环退出。引入计数变量 count 来记录每行输出个数。


视频讲解

3.3.4　break 和 continue

在循环结构中，大多数情况下当循环条件满足时，循环体会一直被执行，直到循环条件不满足。但有时需要在某种条件下使循环提前结束，实现方法有两种：一是使用标志变量，如实例 3.18 中的 flag，通过 flag 值的变化，在循环未正常结束时提前退出循环；二是使用