第3章 流程控制

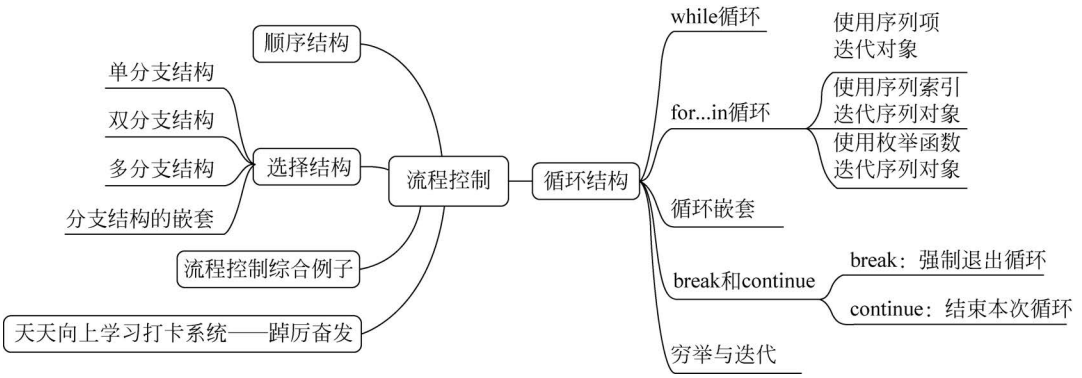
能力目标

【应知】 理解选择和循环的意义和基本实现语句。

【应会】 掌握单分支、双分支及多分支选择结构语句的使用方法；掌握实现无限循环操作的 while 语句、实现遍历操作的 for...in 语句、用于提前结束循环的 break 和 continue 语句。

【难点】 嵌套语句的使用，穷举法和迭代法的使用。

知识导图



流程控制也称控制流程，是计算机运算领域的专用语，是指程序运行时指令（或程序、子程序、代码段）运行或求值的顺序。流程控制对于任何一门编程语言都是至关重要的，它提供了控制程序执行的方法。Python 语言提供了顺序结构、选择结构和循环结构 3 种流程控制。

3.1 顺序结构

顺序结构是程序中最简单的流程控制结构，按照代码出现的先后顺序依次执行。程序中的代码大多是顺序执行的，其结构流程图如图 3.1 所示。

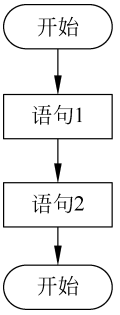


图 3.1 顺序结构流程图

本章之前编写的代码大多采用顺序结构。

【实例 3.1】 输出指定格式的日期。

```
# 处理日期和时间的模块库 datetime.date 是表示日期的类,datetime.datetime 是表示日期
# 时间的类
import datetime
# datetime.day.today()用于获取当前的日期,返回格式为 YYYY-mm-dd。
today = datetime.date.today()
oneday = datetime.timedelta()     # datetime.timedelta()表示两个时间之间的时间差
yesterday = today - oneday
tomorrow = today + oneday
print("今天是:" + str(today))
# strftime()函数接收时间元组,并返回可读字符串表示的时间,格式由参数 format 决定。
print("昨天是:" + yesterday.strftime("%y/%m/%d"))
print("明天是:" + tomorrow.strftime("%m-%d-%Y"))
```

其中,%Y:四位数的年份表示(0000～9999)。

%y:两位数的年份表示(00～99)。

%m:两位数的月份表示(01～12)。

%d:月份内的某一天(1～31)。

运行结果如下:

```
今天是:2023-09-26
昨天是:23/09/25
明天是:09-27-2023
```

3.2　选择结构

选择结构也称分支结构,用于判断给定条件,再根据判定结果控制程序流程。例如,日常生活中常见的登录即为选择结构,用户先输入用户名和密码,系统在数据库中查找并匹配。如果两者都与数据库中的记录保持一致,则登录成功,可以继续下面的操作;否则要重新输入或退出系统。Python 中常用的分支结构有单分支、双分支和多分支 3 种类型。

3.2.1　单分支结构

单分支结构是指只有一个分支,满足判断条件则执行相应语句。现实生活中的“如果天下雨,地会就湿”对应的就是单分支结构。

其结构化流程图如图 3.2 所示。

其语法结构为:

```
if 条件表达式:
    语句块
```

执行过程为:先判断条件,如果执行结果为真,则执行后续语句块,否则什么也不执行。

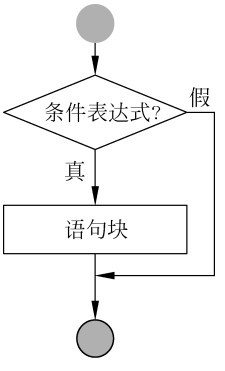


图 3.2　单分支结构流程图

说明:

(1)“条件表达式”可以是逻辑表达式、条件表达式、算术表达式等任意类型的表达式,只要能判断非零或非空即可。“语句块”可以是一条语句,也可以是多条语句。多条语句时,需保证

缩进对齐一致。

（2）“条件表达式”后面一定要加冒号“：”，这是初学者易犯错的地方。

【实例 3.2】 根据出生年份判断是否为成年人。

```
import datetime
year = int(input("请输入出生年份:"))
if datetime.date.today( ).year - year >= 18:   # datetime.date.today( ).year 表示获取
                                               # 当前日期的年份
    print("是成年人!")
```

运行结果如下：

```
请输入出生年份:2015
```

```
请输入出生年份:2005
是成年人!
```

【实例 3.3】 两个整数升序排列并输出。

```
a, b = input("input a,b:").split(" ")   # 一行输入多个数,用空格分开
print("排序前:" + a + "," + b)
if int(a) > int(b):
    a, b = b, a   # 交换 a,b 两个数
print("排序后:" + a + "," + b)
```

运行结果如下：

```
input a,b:3 2
排序前:3,2
排序后:2,3
```

说明：在 Python 中，可直接使用语句“a，b = b，a”交换两个值，而在其他高级语言中必须引入中间变量实现交换，即“t=a，a=b，b=t”，这正是 Python 语言的精妙之处。

拓展：可以尝试实现 3 个整数升序排列，更多数的排序需采用其他高级数据类型实现。

3.2.2 双分支结构

若条件成立时需要执行某些操作，不成立时需执行另一些操作，则需采用双分支结构。例如身份验证时，密码正确可以登录系统，密码错误则要重新输入。其结构化流程图如图 3.3 所示。

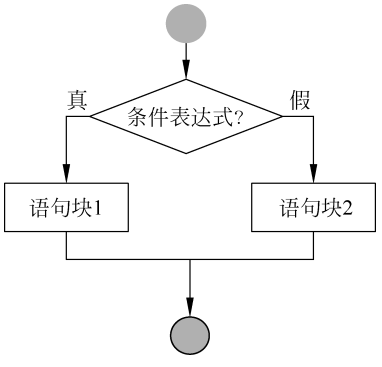


图 3.3 双分支结构流程图

对应的语法结构为：

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

其执行过程为：先判断条件表达式，如果结果为真或非零，则执行语句块 1，否则执行语句块 2。

注意：

（1）双分支结构中的 else 语句不能独立存在，即有 else，一定有相应的 if，但有 if，不一定有 else。

（2）else 后面不需要加也不宜加条件表达式。

【实例 3.4】 求两个数中的较大者（此例题可使用 4 种方法实现）。

方法 1：使用单分支结构

```
a, b = input("input a,b:").split(" ")
max = int(a)
if (int(max) < int(b)):
      max = int(b)
print("较大的数为:" + str(max))
```

方法 2：使用双分支结构

```
a, b = input("input a,b:").split(" ")
if (int(a) > int(b)):
      print("较大的数为:" + a)
else:
      print("较大的数为:" + b)
```

方法 3：使用三目运算符

```
a, b = input("input a,b:").split(" ")
max = int(a) if int(a) > int(b) else int(b)
print("较大的数为:" + str(max))
```

方法 4：使用内置函数 max

```
a, b = input("input a,b:").split(" ")
print("较大的数为:" + str(max(a, b)))
```

三目运算符也称三元运算符，其语法格式为：

```
(True_statements) if (expression) else (False_statements)
```

运算规则为：先对逻辑表达式 expression 求值，如果逻辑表达式返回 True，则执行并返回 True_statements 的值；如果逻辑表达式返回 False，则执行并返回 False_statements 的值。

很明显，三目运算符是双分支结构的一种紧凑表现形式。“条件为真的语句”和“条件为假的语句”可包含多条，语句格式有两种。

（1）多条语句以英文**逗号**隔开，每条语句都会执行，程序返回多条语句的返回值组成的元组。如：

```
>>> a, b = input("input a,b:").split(" ")
>>> s = "No cross, no crown.", "a 大于 b" if a > b else "a 小于或等于 b"
>>> print(s)
```

当输入 10、20 时，运行结果为：

```
('No cross, no crown.', 'a 小于或等于 b')
```

（2）多条语句以英文**分号**隔开，每条语句都会执行，程序只返回第一条语句的值。如：

```
>>> a, b = input("input a,b:").split(" ")
>>> s = "No cross, no crown."; "a 大于 b" if a > b else "a 小于或等于 b"
>>> print(s)
```

当输入 10、20 时，运行结果为：

```
No cross, no crown.
```

另外，三目运算符支持嵌套，通过嵌套三目运算符，可进行更复杂的判断。

3.2.3 多分支结构

在很多情况下，供用户选择的操作有多种，例如，根据空气质量指数判断天气状况并提供生活建议，或者根据百分制成绩判断成绩等级等。使用程序语句实现时，就可以使用多分支结构进行处理。其结构化流程图如图 3.4 所示。

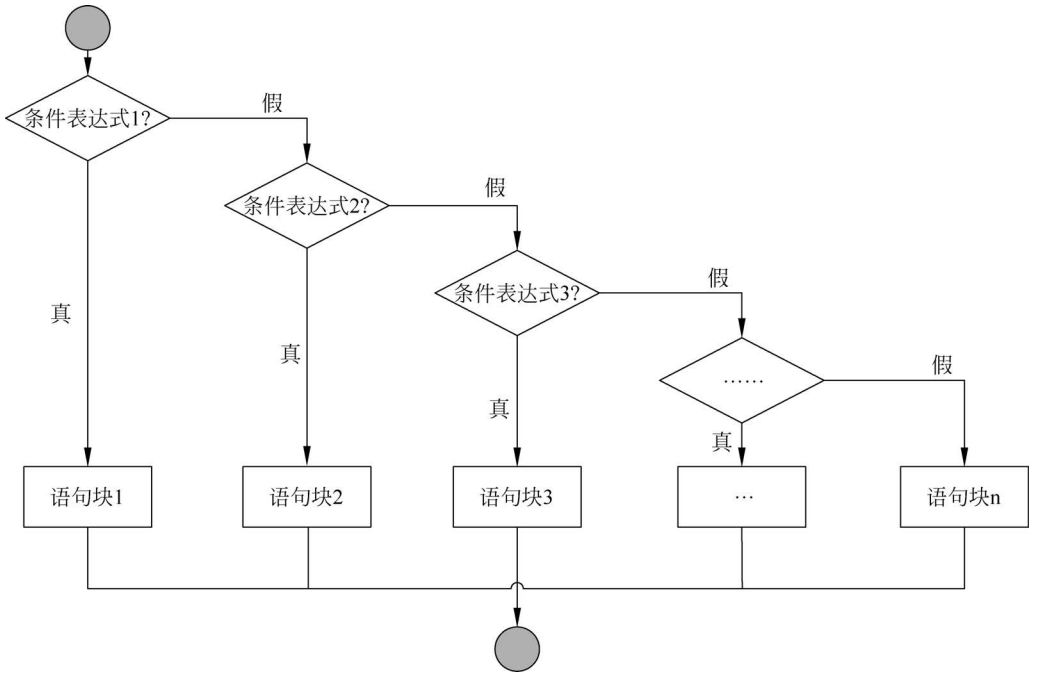


图 3.4　多分支结构流程图

对应的语法格式为：

```
if 条件表达式 1:
       语句块 1
elif   条件表达式 2:
       语句块 2
elif   条件表达式 3:
       语句块 3
…
else:
       语句块 n
```

执行过程为：先判断条件表达式 1，如果结果为真，则执行语句块 1；否则判断条件表达式 2，如果结果为真，则执行语句块 2……只有在所有表达式都为假的情况下，才执行 else 后的语句块 n。

【实例 3.5】　计算阶梯电价（阶梯电价是按照用户消费的电量分段定价，用电价格随用电量增加呈阶梯状逐级递增的一种电价定价机制，目的是减少资源浪费，提高能源利用效率。2023 年徐州市居民阶梯电价收费规则为：当每月用电量 0～230 度时为第一档，电价是 0.5283 元/度；当每月用电量 231～400 度时为第二档，电度单价在一档单价基础上加 0.05 元/度；当每月用电量为 401 度及以上时，电度单价在一档基础上加 0.3 元/度）。计算结果保留两位小数。

```
1    x = float(input("请输入每月用电量:"))
2    if x < 0:
3        print("输入错误!")
4    else:
5        if x <= 230:
6            y = 0.5283 * x
```

```
    elif x <= 400:
        y = (0.5283 + 0.05) * x
    else:
        y = (0.5283 + 0.3) * x
print("本月电费为 %.2f 元" % (y))
```

运行结果如下：

```
请输入每月用电量:-1
输入错误!
```

```
请输入每月用电量:200
本月电费为 105.66 元
```

```
请输入每月用电量:400
本月电费为 231.32 元
```

```
请输入每月用电量:601
本月电费为 497.81 元
```

【实例 3.6】　根据空气质量指数进行生活建议。

空气质量指数(air quality index，AQI)是根据空气中的各种成分占比，将监测的空气浓度简化为单一概念型数值的形式，将空气污染程度和空气质量状况分级表示，用于反映城市的短期空气质量状况和变化趋势。具体数值及等级如表 3.1 所示。

表 3.1　AQI 数值、对应等级及生活建议

AQI 数值	对 应 等 级	生 活 建 议
0～50	一级 优	空气清新，适宜参加户外活动
51～100	二级 良	可以正常进行户外活动
101～150	三级 轻度污染	敏感人群减少体力消耗大的户外活动
151～200	四级 中度污染	对敏感人群影响较大，减少户外活动
201～300	五级 重度污染	所有人适当减少户外活动
＞300	六级 严重污染	尽量不要留在户外

源代码如下：

```
x = int(input("请输入 AQI 数值:"))
if x < 0:
    print("输入错误!")
else:
    if x <= 50:
        s = "一级,优,空气清新,适宜参加户外活动。"
    elif x <= 100:
        s = "二级,良,可以正常进行户外活动。"
    elif x <= 150:
        s = "三级,轻度污染,敏感人群减少体力消耗大的户外活动。"
    elif x <= 200:
        s = "四级,中度污染,对敏感人群影响较大,减少户外活动。"
    elif x <= 300:
        s = "五级,重度污染,所有人适当减少户外活动。"
    else:
        s = "六级,严重污染,尽量不要留在户外。"
print("空气质量为" + s)
```

运行结果如下：

```
请输入 AQI 数值:200
空气质量为四级,中度污染,对敏感人群影响较大,减少户外活动。
```

【实例 3.7】　学期末，李老师要根据学生的百分制总成绩给出对应等级：成绩 90 分以

上(包含 90 分)等级为“优秀”,成绩 90～75 分(包含 75 分)等级为“良好”,成绩 75～60 分(包含 60 分)等级为“及格”,60 分以下为“不及格”。其中“Python 程序设计”课程的百分制总成绩计算方法为:总成绩=平时成绩×10%+实验成绩×30%+期末成绩×60%(备注:平时成绩、实验成绩和期末成绩满分均为 100 分)。请输入某位学生的平时成绩、实验成绩和期末成绩,输出总成绩及对应等级。

源代码如下:

```
usual, expe, final = input("请输入平时成绩、实验成绩和期末成绩(用","分隔):").split(',')
usual, expe, final = eval(usual), eval(expe), eval(final)
total = usual * 0.1 + expe * 0.3 + final * 0.6
print("该生最终成绩为" + str(total), end = ',')
if total > 100 or total < 0:
      print("您的输入有误!")
elif total >= 90:
      print("优秀")
elif total >= 75:
      print("良好")
elif total >= 60:
      print("及格")
else:
      print("不及格")
```

运行结果如下:

```
请输入平时成绩、实验成绩和期末成绩(用","分隔):100,90,90
该生最终成绩为 91.0,优秀
```

3.2.4 分支结构的嵌套

分支结构的嵌套是指实际开发过程中,在一个分支结构中嵌套另一个分支结构。基本语法格式如下:

```
if 条件表达式 1:
     语句块 1
     if 条件表达式 2:
          语句块 2
     else:
          语句块 3
else:
     if 条件表达式 3:
          语句块 4
```

从语法角度讲,选择结构可有多种嵌套形式。程序员可根据需要选择合适的嵌套结构,但一定要注意控制不同级别代码块的缩进量,因为缩进量决定代码块的从属关系。

【实例 3.8】 分段函数求值 $f(x)=\begin{cases} x & x\leqslant 1 \\ 2x-1 & 1<x<10 \\ 3x-11 & x\geqslant 10 \end{cases}$。

源代码如下:

```
x = float(input("input x:"))
if x <= 1:
```

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

运行结果如下：

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

也可以不使用嵌套语句，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"，扁平胜于嵌套，所以能扁平化时尽量不要嵌套。

3.3　循环结构

如果需要重复执行某条或某些指令，例如"中国诗词大赛"中的"飞花令"，选手要根据给定的关键字，在规定的时间内轮流背诵含关键字的诗句，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。要更灵活地操纵循环语句的流向，还需使用 break、continue 和 pass 等语句。

3.3.1　while 循环

while 循环也称无限循环，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"可以是任意非空或非零的表达式，"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。

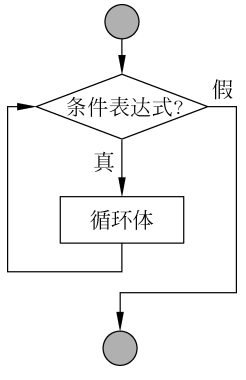


图 3.5　while 循环结构流程图

执行过程：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、结束条件和增量（步长）。本例中，循环变量为 i，初值为 1，结束条件或终值为 100，步长为 1。另外，还需要一个变量存储累加和，其初值为 0。对应的结构化流程图如图 3.6 所示。

源代码如下：

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

运行结果如下：

```
sum = 5050
```

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

【实例 3.10】 求若干名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，终值为学生人数 n，步长为 1。循环体累加每名学生的成绩，循环结束后求成绩和的平均值。其结构化流程图如图 3.7 所示。

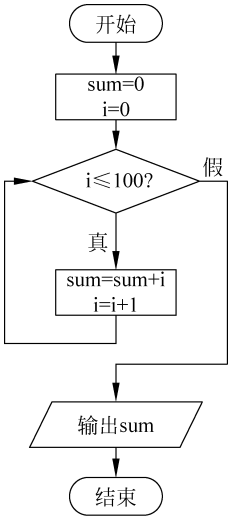


图 3.6 求累加和结构流程图

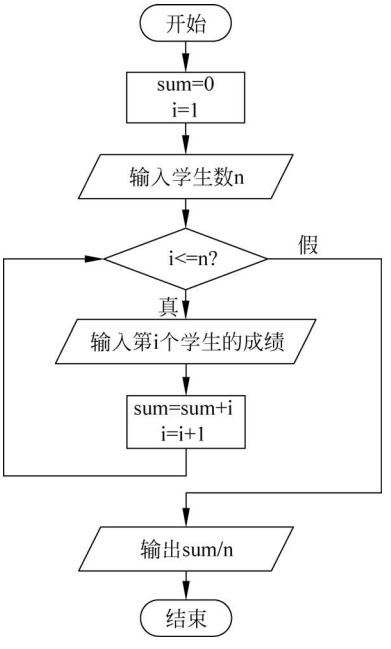


图 3.7 求平均成绩结构流程图

源代码如下：

```
1    sum,i = 0,1
2    n = int(input("请输入学生人数:"))
```

```
3    while i <= n:
4        score = float(input("NO " + str(i) + ": "))
5        sum += score
6        i = i + 1
7    print("平均成绩为 " + str(sum / n) + "分。")
```

运行结果如下：

```
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 分。
```

循环结构中也可使用 else 子句，表示不满足循环条件时程序的执行流程。

【实例 3.11】　循环结构中使用 else 子句示例。

```
1    count = 0
2    while count < 5:
3        print(str(count) + " is less than 5.")
4        count = count + 1
5    else:
6        print(str(count) + " is not less than 5.")
```

运行结果如下：

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

3.3.2　for...in 循环

Python 中的另一种循环结构是 for...in 循环语句，其与 Java、C++等编程语言中的 for 语句不同，更像是 shell 或脚本语言中的 for 循环，可遍历列表、元组、字符串等序列成员，也可用于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，退出循环。

对应的结构化流程图如图 3.8 所示。

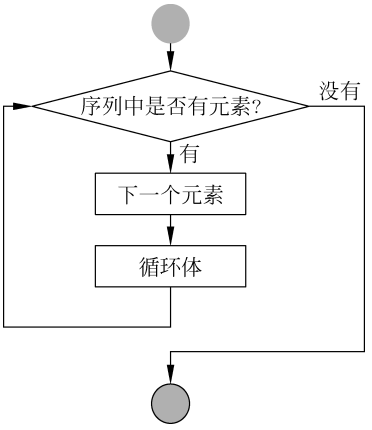


图 3.8　迭代序列 for...in 循环结构流程图

【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)中的每个元素，分别判断其所属类型，并将对应个数加 1，直至遍历结束。结构化流程图如图 3.9 所示。

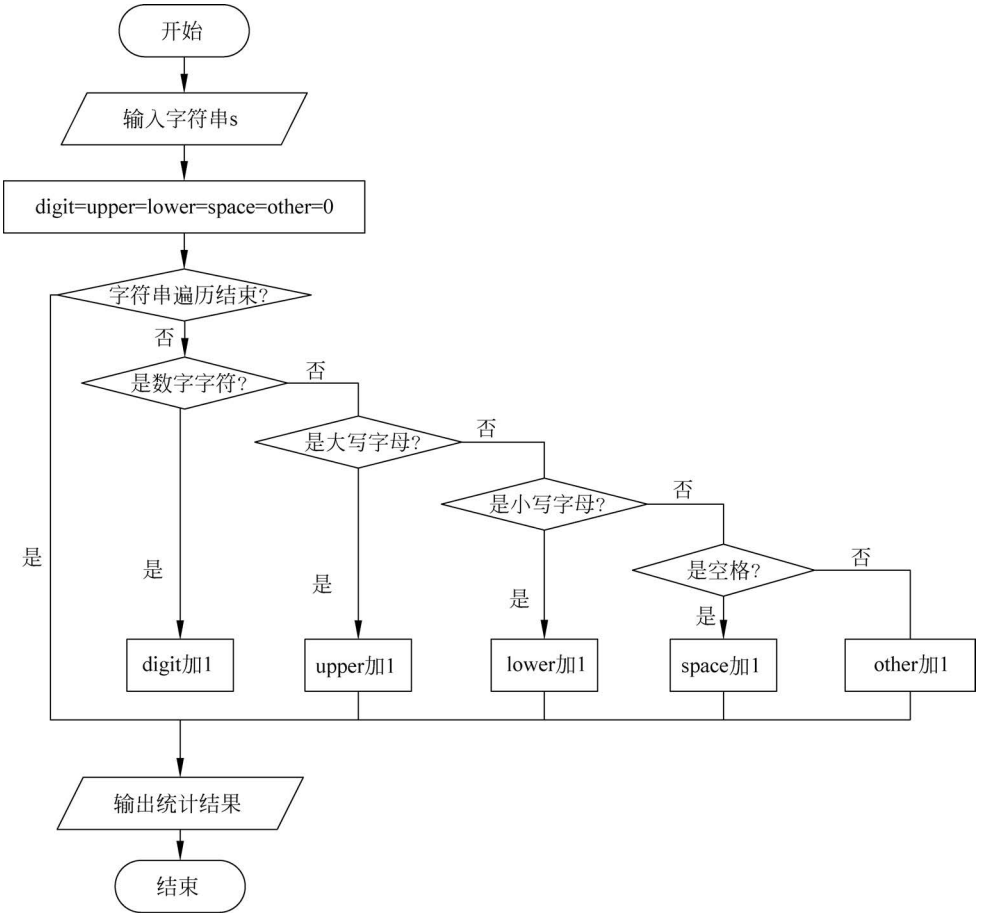


图 3.9　统计字符串中各类字符个数的结构流程图

源代码如下：

```
s = input("请输入一个字符串:")
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
for i in s:
    if i >= '0' and i <= '9':     #判断 i 是否为数字字符
        digit = digit + 1
    elif i >= 'A' and i <= 'Z':   #判断 i 是否为大写字母
        upper = upper + 1
    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))
```

运行结果如下：

```
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 分别用于判断 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，也可使用序列索引遍历列表，语法格式如下：

```
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、空格和其
```

```
    other = 0                                    # 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))
```

运行结果如下：

```
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 函数可得到用于迭代的索引列表，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))
```

运行结果如下：

```
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、元组或字符串）组合为一个索引序列，同时列出数据和下标，一般用于 for...in 循环中。其语法格式为：

```
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】　打印学生花名册。

```
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)
```

运行结果如下：

```
0 李白
1 孟浩然
2 王维
3 李绅
```

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 循环结构都可进行循环嵌套。如：

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

for...in 循环结构可嵌入 while 循环结构，while 循环结构也可嵌入 for...in 循环结构，还可以根据自身需要任意嵌套。

【实例 3.16】　打印九九乘法表(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(迭代变量从 1 到 9)，内循环控制第二个乘数。因为要求按照下三角形打印，所以内循环迭代变量只能从 1 到 i。结构化流程图如图 3.10 所示。

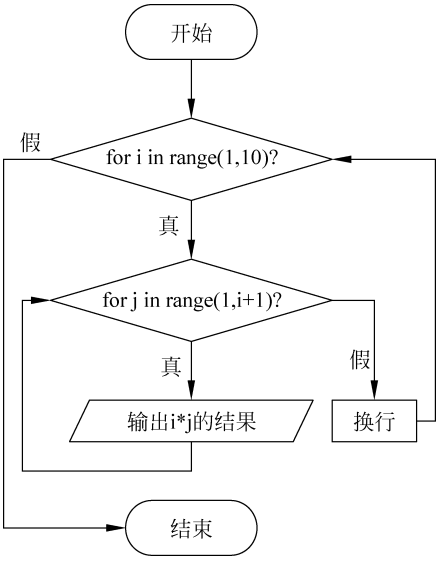


图 3.10　打印九九乘法表结构流程图

源代码如下：

```
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
```

运行结果如下：

```
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：打印上三角九九乘法表、钻石等图形。

【实例 3.17】 求 1!+2!+…+20!。

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

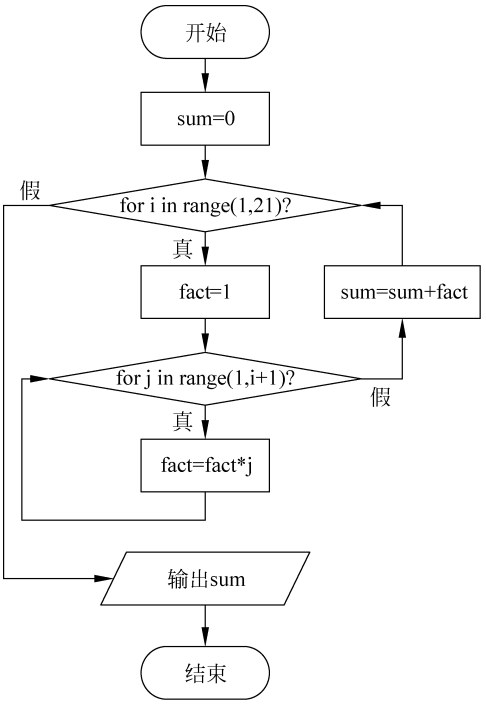


图 3.11 阶乘和结构流程图

源代码如下：

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

运行结果如下：

```
sum = 2561327494111820313
```

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，故可使用单重循环实现。优化后的代码如下：

```
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 计算 n!
    sum = sum + fact
print("sum = " + str(sum))
```

循环结构中可嵌套另一循环结构，也可嵌套选择结构，反之亦然。

【实例 3.18】 列出 1～200 之间的所有素数，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。这个过程需使用单重循环嵌套选择结构实现。而列出 1～200 之间的所有素数也需要循环实现，故使用双重循环完成。流程图如图 3.12 所示。

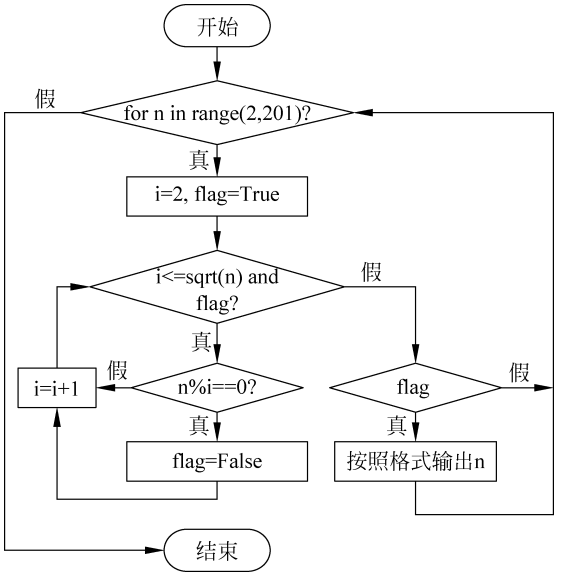


图 3.12　实例 3.18 结构流程图

源代码如下：

```
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
    while i <= math.sqrt(n) and flag:    # 内循环判断每个数是否为素数
        if n % i == 0:                   # 如果能够整除,则不是素数,将 flag 置为 False
            flag = False
        i = i + 1
    if flag:                             # 如果是素数,则打印
        count = count + 1
        print(n, end = "\t")
        if count % 10 == 0:              # 控制每行 10 个
            print()
```

运行结果如下：

```
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    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 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，退出内循环。引入计数变量 count 记录每行打印个数。

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，会一直执行循环体，直至循环条件不满足。但有时需要在某种条件下提前结束循环，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 break 实现。

break 语句可以终止当前循环。一般与 if 语句搭配使用，表示在某种条件下提前结束循环。

【实例 3.19】 break 语句示例。

```
1    n = int(input("n:"))
2    for i in range(1, 11):
3        if i == n:
4            break
5    print(i, end = ",")
```

运行结果如下：

```
n:5
1,2,3,4,
```

```
n:20
1,2,3,4,5,6,7,8,9,10,
```

从运行结果可以看出，当 i 的迭代次数小于 10 时，循环会提前结束。

注意：使用嵌套循环时，break 语句只跳出最内层的循环。

【实例 3.20】 列出 1～200 的所有素数，要求每行输出 10 个数(break 版)。

```
1    import math                          # math 模型库，sqrt()函数需要使用
2    count = 0
3    for n in range(2, 201):              #外循环遍历数据
4        i = 2
5        while i <= math.sqrt(n):         #内循环判断是否为素数
6            if n % i == 0:               #能整除，则不是素数，判断结束
7                break
8            i = i + 1
9        if i > math.sqrt(n):             #是素数
10           count = count + 1
11           print(n, end = "\t")
12           if count % 10 == 0:          #控制每行 10 个
13               print()
```

说明：结束内循环有两种途径：①n是素数，正常结束，即所有的n%i!=0，此时i>sqrt(n)；②n%==0，即n不是素数，提前结束循环。所以break版与标志变量版在输出素数时条件正好相反。

有时只在一定条件下不执行本次循环体，而继续执行下一轮循环，此时需使用另一种语句——continue。

continue是另一种提前结束循环的语句，与break不同，continue只结束本次循环，继续后续操作。

【实例3.21】 continue语句示例。

```
n = int(input("n:"))
for i in range(1, 11):
    if i == n:
        continue
    print(i, end = ",")
```

运行结果如下：

```
n:5
1,2,3,4,6,7,8,9,10,
```

```
n:20
1,2,3,4,5,6,7,8,9,10,
```

从运行结果可以看出，当n<10时，不执行本次循环，而继续执行后续循环。

【实例3.22】 break与continue语句的区别示例。

```
import random
n = random.randint(0, 10)
print("您选择的是", n)
for i in range(1,11):
    if i == n:
        print(i, "结束了。")
        break
    if i % 3 != 0:
        print(i, "继续!")
        continue
    print('I love Python!')
```

运行结果如下：

```
您选择的是 4
1 继续!
2 继续!
I love Python!
4 结束了。
```

```
# 满足 i % 3!= 0,执行 continue,进行下一次循环

# 两个判断条件都不满足
# 满足 i == n,执行 break,退出循环
```

此程序段的执行过程也可用图3.13表示。

3.3.5　穷举与迭代

1. 穷举

穷举法也称枚举法或列举法，是计算机求解问题时常用的算法，用于解决通过公式推导、规则演绎等方法不能解决的问题。其基本思想是：不重复、不遗漏地列举所有可能的情

```
import random

n = random.randint(0, 10)
print("您选择的是", n)
for i in range(1,11):
    if i == n:
        print(i, "结束了。")
        break
    if i % 3 != 0:
        print(i, "继续！")
        continue
    print('I love Python!')
```

图 3.13　break 与 continue 语句的区别示意图

况，以便从中寻找满足条件的结果。采用穷举法解决实际问题时，主要使用循环结构嵌套选择结构实现——循环结构用于列举所有可能的情况，而选择结构用于判断当前条件是否为所求解，其一般框架为：

```
for 循环变量 x 的所有可能的值：
    if x 满足指定条件：
        x 即为所求解
```

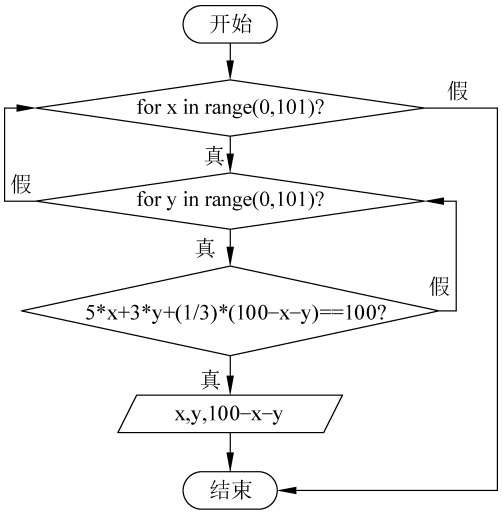


图 3.14　百钱买百鸡结构流程图

【实例 3.23】（百钱买百鸡）我国古代数学家张丘建在《算经》一书中提出的数学问题：鸡翁一值钱五，鸡母一值钱三，鸡雏三值钱一。百钱买百鸡，问鸡翁、鸡母、鸡雏各几何？

算法分析：假设有 x 只鸡翁（即公鸡），y 只鸡母（即母鸡），则鸡雏（小鸡）有（100－x－y）只。根据题意，一只公鸡需要五钱，一只母鸡需要三钱，一只小鸡需要一钱，故可以列方程 $5*x+3*y+\frac{1}{3}*(100-x-y)=100$。这是一个不定式方程，不能利用普通的公式推导得出结论，最常用的解法是枚举，将所有可能的情况一一列举出来。流程图如图 3.14 所示。

```
for x in range(101):
    for y in range(101):
        if 5 * x + 3 * y + (1/3) * (100 - x - y) == 100:
            print(x, y, (100 - x - y))
```

运行结果如下：

```
0 25 75
4 18 78
8 11 81
12 4 84
```

当然，此程序可以优化。由题意可知，公鸡最多 20 只，母鸡最多 33 只，所以可将程序优化为：

```
for x in range(21):
    for y in range(34):
        if 5 * x + 3 * y + (1/3) * (100 - x - y) == 100:
            print(x, y, (100 - x - y))
```

2. 迭代

迭代是另一种常用的循环算法，其利用计算机运行速度快，适合重复操作的特点，使计算机对一组语句进行重复操作，且后一次的操作数基于前一次的执行结果。用迭代法求解实际问题时，需要考虑两方面的问题。

（1）确定迭代变量：由旧值直接或间接递推而来的变量就是迭代变量。

（2）建立迭代关系式：迭代关系式即“循环不变式”，是一个直接或间接由旧值递推出新值的表达式。

【实例 3.24】（斐波那契数列）意大利著名的数学家斐波那契在《计算之书》中提出了一个有趣的兔子问题：一对成年兔子每个月恰好生下一对小兔子（一雌一雄）。年初时，只有一对小兔子。第一个月结束时，它们成长为成年兔子，第二个月结束时，这对成年兔子将生下一对小兔子。这种成长与繁殖的过程会一直持续，并假设生下的小兔子都不会死，那么一年之后共有多少对小兔子？（为清楚描述数列，打印前 20 项）

算法分析：斐波那契数列（Fibonacci sequence），又称黄金分割数列、兔子数列。从问题的描述可以发现，年初时只有一对小兔子，第二个月这对小兔子长成中兔子（兔子总数为 1 对）；第三个月中兔子长成大兔子并生下一对小兔子（兔子总数为 2 对）；第四个月小兔子长成中兔子，大兔子再生下一对小兔子（兔子总数为 1+1+1=3 对）……可用表 3.2 描述这个过程。

表 3.2　斐波那契数列的变化过程

月数	小兔子对数	中兔子对数	大兔子对数	兔子总数
1	1	0	0	1
2	0	1	1	1
3	1	0	1	2
4	1	1	1	3
5	2	1	2	5
6	3	2	3	8
…	…	…	…	…

可以看出，斐波那契数列为 1，1，2，3，5，8，…从第三个数开始，后一个数为前两个数之和。可使用迭代法求解，迭代表达式为：

$$F(n)=\begin{cases}1 & n=1 \parallel n=2\\ F(n-1)+F(n-2) & n>2\end{cases}$$

其中，n 为 1 或 2 时为迭代出口。

流程图如图 3.15 所示。

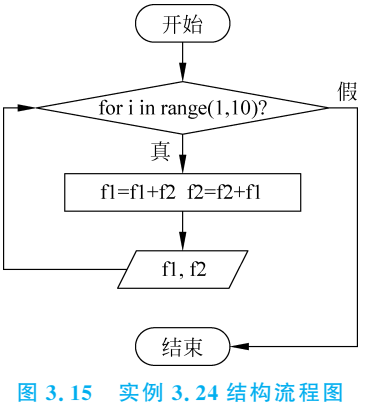


图 3.15　实例 3.24 结构流程图

```
f1 = f2 = 1                    # 迭代变量的初值
print(f1, f2, end = " ")       # 先输出前两个数
```

```
    for f_index in range(1, 10):        # 每次输出两个数,一共输出 10 个数
        f1 = f1 + f2                    # 迭代表达式,后一个数为前两个数的和
        f2 = f2 + f1
        print(f1, f2, end = " ")
```

运行结果如下：

```
1 1 2 3 5 8 13 21 34 55 89 144 233 377 610 987 1597 2584 4181 6765
```

3.4 流程控制综合例子

【实例 3.25】 设计小型的加减乘除测试程序(由系统随机给出 10 道加减乘除运算题目,运算数和运算符都由系统随机给出,系统自动给出答题结果和运算时间)。

算法分析：此实例需要循环与多分支结构嵌套,循环负责控制题目数量,分支结构检测加减乘除并进行相应计算。流程图如图 3.16 所示。

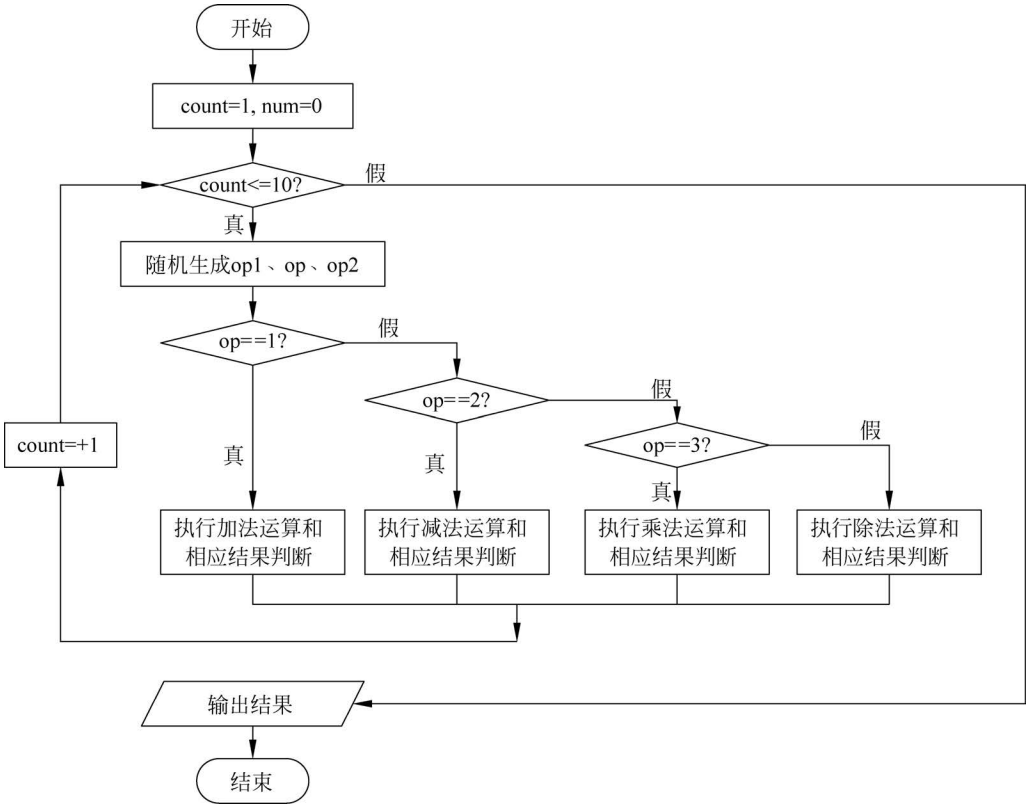


图 3.16 实例 3.25 结构流程图

源代码如下(为简单起见,将操作数规定为不大于 10 的数)：

```
import random                       # 随机函数库
import time                         # 时间库
count, num = 1, 0                   # 分别表示题目数量,答对的题目数量
```

```
begin_time = time.time()          # 取当前系统时间,单位为秒
while count < 11:
    op1 = random.randint(1, 11) # 随机生成一个不大于 10 的整数,作为第一个操作数
    op2 = random.randint(1, 11) # 随机生成一个不大于 10 的整数,作为第二个操作数
    op = random.randint(1, 4)   # 随机生成一个不大于 4 的数,作为操作符
    if op == 1:                 # 加法运算
        print("第" + str(count) + "题:" + str(op1) + "+" + str(op2) +"=",end="")
        result = int(input())
        if result == op1 + op2:
            print("正确!")
            num = num + 1          # 答对题目数量加 1
        else:
            print("错误!")
    elif op == 2:                  # 减法运算
        if op1 < op2:              # 保证被减数>减数
            op1, op2 = op2, op1
        print("第" + str(count) + "题:" + str(op1) + "-" + str(op2) +"=",end="")
        result = int(input())
        if result == op1 - op2:
            print("正确!")
            num = num + 1          # 答对题目数量加 1
        else:
            print("错误!")
    elif op == 3:                  # 乘法运算
        print("第" + str(count) + "题:" + str(op1) + "×" + str(op2)+"=",end="")
        result = int(input())
        if result == op1 * op2:
            print("正确!")
            num = num + 1          # 答对题目数量加 1
        else:
            print("错误!")
    else:
        while op1 % op2 != 0:      # 保证整除
            op1 = random.randint(1,11)# 随机生成一个不大于 10 的数,作为第一个操作数
            op2 = random.randint(1,11)# 随机生成一个不大于 10 的数,作为第二个操作数
            if op1 < op2:          # 保证被除数>除数
                op1, op2 = op2, op
        if op1 < op2:              # 保证被除数>除数
            op1, op2 = op2, op
        print("第" + str(count) + "题:" + str(op1) + "÷" + str(op2)+"=",end="")
        result = int(input())
        if result == op1 // op2:   # 为简单起见,采用整除
            print("正确!")
            num = num + 1          # 答对题目数量加 1
        else:
            print("错误!")
    count = count + 1
end_time = time.time()             # 获取系统当前时间
print("答对" + str(num) + "道题目, 得分" + str(10 * num) + "分,", end=' ')
print("用时为 %.2f 秒." % float(end_time - begin_time))    # 两次时间差为运行时间
```

运行结果如下：

```
第 1 题:7 + 2 = 9
正确!
第 2 题:5 × 4 = 20
正确!
第 3 题:7 × 2 = 14
正确!
第 4 题:10 − 5 = 5
正确!
第 5 题:10 + 9 = 19
正确!
第 6 题:8 ÷ 4 = 2
正确!
第 7 题:10 − 6 = 4
正确!
第 8 题:10 − 7 = 3
正确!
第 9 题:5 + 8 = 13
正确!
第 10 题:11 − 2 = 9
正确!
答对 10 道题目，得分 100 分，用时为 17.33 秒。
```

【实例 3.26】 模拟“剪刀石头布”五局三胜猜拳游戏：选手和计算机轮流猜拳五次，三次胜利才算赢。

算法分析： 选手输入选项（“剪刀”“石头”“布”），计算机随机给出选项，按照游戏规则——“布”>“石头”>，“石头”>“剪刀”，“剪刀”>“布”进行评判和计数，一旦一方满足五局三胜，则游戏结束。流程图如图 3.17 所示。

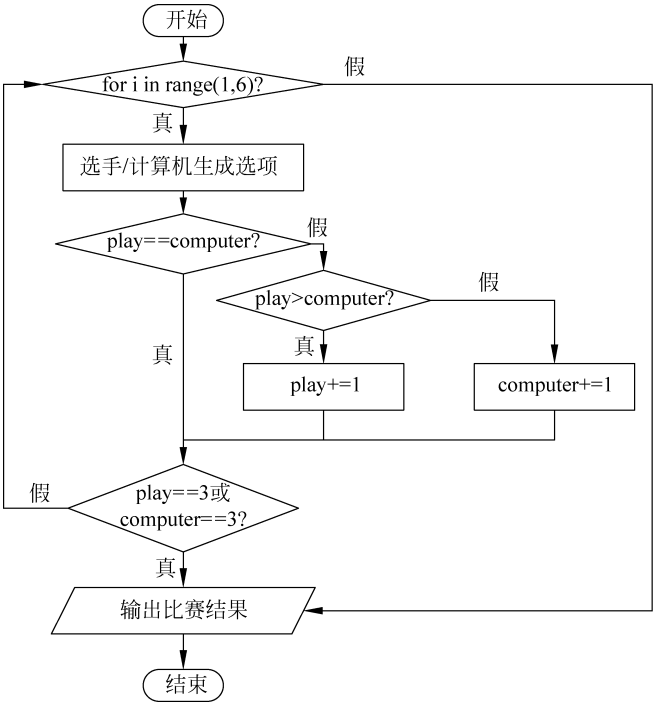


图 3.17　五局三胜猜拳游戏结构流程图

源代码如下：

```
import random                    # 随机函数库
count1 = count2 = 0              # count1 和 count2 分别表示选手和计算机获胜的次数
for i in range(1, 6):            # 最多进行五局
    print("第 %d 局:" % i)
    play = input("选手:")# 选手输入选项并进行分类"剪刀" --> 1,"石头" --> 2,"布" --> 3
    play = 1 if play == "剪刀" else (2 if play == "石头" else 3)  # 三目运算符嵌套
    computer = random.randint(1, 3) # 计算机随机生成选项:1. - 剪刀,2. - 石头,3. - 布
    # 三目运算符嵌套输出计算机选项,可读性较差
    print("计算机:剪刀") if computer == 1 else (print("计算机:石头") if computer ==
    2 else print("计算机:布"))
    if play == computer: # 进行判断并计数
        print("选项一样")
    elif (play == 1 and computer == 2) or (play == 2 and computer == 3) or ( play ==
    3 and computer == 1):
                                     # "剪刀"<"石头","石头"<"布","布"<"剪刀"
        print("计算机赢")
        count2 += 1
    else:
        print("选手赢")
        count1 += 1
    if count1 == 3 or count2 == 3:              # 三胜退出
        break
#输出最终结果
if count1 > count2:
    print("最终选手胜出")
elif count1 < count2:
    print("最终计算机胜出")
else:
    print("平局")
```

运行结果如下：

```
第 1 局:
选手:剪刀
计算机:布
选手赢
第 2 局:
选手:石头
计算机:布
计算机赢
第 3 局:
选手:剪刀
计算机:石头
计算机赢
第 4 局:
选手:石头
计算机:石头
选项一样
第 5 局:
选手:布
计算机:剪刀
计算机赢
最终计算机胜出
```

【实例 3.27】 用 1、3、5、8 几个数字，能组成的互不相同且无重复数字的三位数各是多少（每行输出 10 个数字）？总共有多少个？（蓝桥杯全国软件大赛青少年创意编程 Python 组）

算法分析：使用穷举法解决问题，循环结构列出所有可能，选择结构进行判断。

源代码如下：

```
data = [1, 3, 5, 8]                    # 列表存储数字，列表的内容在后续章节中详细介绍
count = 0                              # 满足条件的数的个数
for i in data:                         # 穷举法进行判断，循环结构穷举所有可能
    for j in data:
        for k in data:
            if i != j and j != k and k != i:   # 选择结构进行判断是否满足给定条件
                count += 1                     # 个数加 1
                print(100 * i + 10 * j + k, end = " ")   # 输出数字
                if count % 10 == 0:            # 每行 10 个
                    print( )
print("\n一共" + str(count) + "个数字互不相同且无重复数字的三位数")
```

运行结果如下：

```
135 138 153 158 183 185 315 318 351 358
381 385 513 518 531 538 581 583 813 815
831 835 851 853
一共 24 个数字互不相同且无重复数字的三位数
```

3.5 天天向上学习打卡系统——踔厉奋发

3.5.1 思政导入

1951 年国庆节来临之际，中央人民政府政务院邀请全国各地的英模人物进京参加国庆观礼。受邀代表中，有位名叫马毛姐的 16 岁安徽姑娘，特别引人注目。因为她是年龄最小的代表，受到毛泽东主席的亲切接见。主席不仅关切地询问她念书情况，还送她一本精美的笔记本，并在扉页上题词：“好好学习，天天向上。”随即，这 8 个字的题词迅速在全国传播开来，成为天下少年共同的读书誓言。

其实，“好好学习，天天向上”来源于中国儒家经典《礼记·大学》。汤之《盘铭》曰：苟日新，日日新，又日新。原本说的是洗澡问题，如果今日洗去了一身的污垢，以后每天都要把污垢洗干净，如此坚持天天洗。商汤王将这句“苟日新，日日新，又日新”刻在洗澡盆上，说明这不仅仅是洗澡问题，引申为精神上的洗礼、品德上的修炼、思想上的改造。同样地，《庄子·知北游》提出“澡雪而精神”，《礼记·儒行》中也有“澡身而浴德”的说法。

3.5.2 案例任务

天天向上学习打卡系统是一个具有日期显示、学习经验值计算和进一步建议功能的模拟系统。在“显示日期”模块中，显示当天的日期和星期。在“计算学习经验值”模块中，设置

标准学习时长为 8 小时，根据用户设定的打卡周期，计算每天的学习经验值并进行累计。在“进一步建议”模块中，根据学习经验值进行学习推荐：如果每天学习时长少于标准时长的 80%，则建议“您的学习时间偏少，需要加强时间利用率，提高学习效率！”如果每天学习时长大于标准时长的 120%，则建议“您的学习时间偏多，需要注意休息，加强体育锻炼！”

3.5.3　案例分析和实现

根据任务描述，程序实现可分为如下几步。

1. 通过 datatime 库获取当天日期，并输出相关信息。

2. 根据用户输入的打卡天数 n，进行 n 次循环。在每次循环中，根据用户输入的当天学习时长进行学习经验值的计算（学习经验值＝当天学习时长/标准时长），并累加学习时长和学习经验值。

3. 根据学习经验值进行进一步学习时长建议。

流程图如图 3.18 所示。

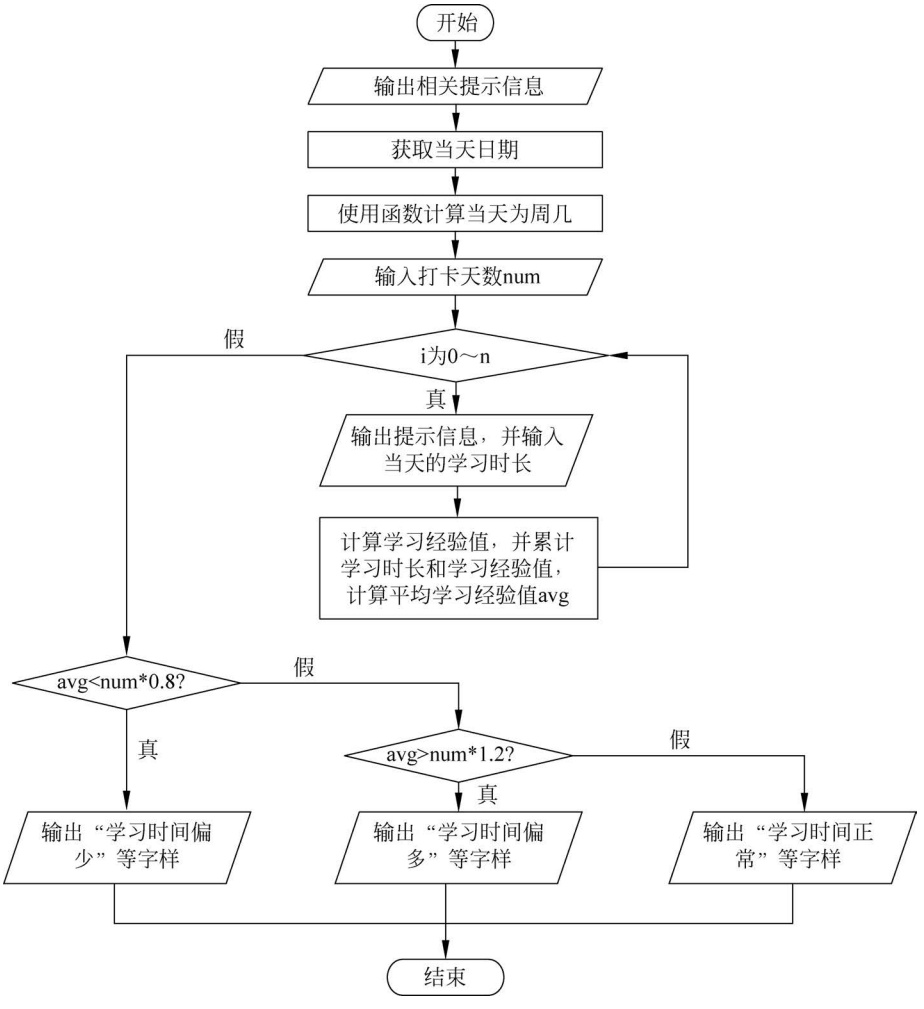


图 3.18　天天向上学习打卡系统结构流程图

源代码如下：

```
import datetime                                    #导入日期时间库
print('-'*11+'苟日新,日日新,又日新。'+'-'*11)
print("欢迎您使用天天向上学习打卡系统,标准学习时长为 8 小时/天。")
sum,avg=0,0                                        #分别表示学习总时长和总学习经验值
flag = False
today=datetime.date.today()                        #获取当天日期
print("今天是"+today.strftime("%Y年%m月%d日"),end=",星期")
week=today.isoweekday()                            #计算当天为周几
if week==1:
    print("一。")
elif week==2:
    print("二。")
elif week==3:
    print("三。")
elif week == 4:
    print("四。")
elif week==5:
    print("五。")
elif week==6:
    print("六。")
else:
    print("日。")
num=int(input("请输入您需要打卡的天数:"))
after_nday=today+datetime.timedelta(days=num)      #当天日期后 num 天
print("需要打卡的时间段为:"+today.strftime("%Y年%m月%d日")+"～"+after_nday.
strftime("%Y年%m月%d日"))
for i in range(num):
    this_day=today+datetime.timedelta(days=i);
    count=float(input("请输入"+this_day.strftime("%Y年%m月%d日")+"学习时
    长(小时):"))
    sum+=count                                     #累计学习时长
    value=count/8                                  #计算学习经验值
    print("您今天的学习经验值为%.2f。"%value)
    avg+=value                                     #累计学习经验值
print(f"恭喜您,完成这次监督学习!\n您{num:d}天的总学习时长为{sum:.2f}小时,获得总
学习经验值{avg:.2f}。")
# 根据学习经验值进行建议
if avg<num*0.8:
    print("您的学习时间偏少,需要加强时间利用率,提高学习效率!")
elif avg>num*1.2:
    print("您的学习时间偏多,需要注意休息,加强体育锻炼!")
else:
    print("您的学习时间把握非常好,继续保持!")
```

运行结果如下：

```
---------- 苟日新,日日新,又日新。 ----------
欢迎您使用天天向上学习打卡系统,标准学习时长为 8 小时/天。
今天是 2023 年 05 月 05 日,星期五。
请输入您需要打卡的天数:5
需要打卡的时间段为:2023 年 05 月 05 日～2023 年 05 月 10 日
请输入 2023 年 05 月 05 日学习时长(小时):5
```

```
您今天的学习经验值为 0.62。
请输入 2023 年 05 月 06 日学习时长(小时):6
您今天的学习经验值为 0.75。
请输入 2023 年 05 月 07 日学习时长(小时):7
您今天的学习经验值为 0.88。
请输入 2023 年 05 月 08 日学习时长(小时):8
您今天的学习经验值为 1.00。
请输入 2023 年 05 月 09 日学习时长(小时):9
您今天的学习经验值为 1.12。
恭喜您,完成这次监督学习!
您 5 天的总学习时长为 35.00 小时,获得总学习经验值 4.38。
您的学习时间把握非常好,继续保持!
```

3.5.4 总结和启示

本案例模拟天天向上学习打卡系统,使用标准库 datetime 显示当前日期和相关星期信息,使用循环结构模拟 n 天的打卡过程,使用选择结构进行进一步学习建议,即综合使用前面两章所学知识进行设计和模拟。通过这个案例,可以很好地理解和掌握数据类型和程序控制结构。当然,该案例实现功能较为简单,但是随着后续知识的讲授和掌握,大家可以使用列表或字典存储学习时长、学习经验值等相关信息,也可以使用文件或数据库存放学习建议等,从而实现更复杂、更真实的打卡系统。

“日新月异,天行健,君子以自强不息。”“好好学习,天天向上”是一种向上、阳光的心态和状态。在这个瞬息万变的时代,科技日新月异,知识更新周期快,需持续学习,与时俱进,自我迭代,使每日的自己优于昨日的自己。蓝图已经绘就,号角已经吹响。我们要踔厉奋发、勇毅前行,努力创造更加灿烂的明天。

3.6 本章小结

本章详细介绍了 Python 的流程控制,主要包括顺序结构、单分支选择结构、双分支选择结构、多分支选择结构、while 循环结构、for...in 循环结构、break 语句和 continue 语句的概念和用法。在讲解过程中,结合大量实例,生动形象地演示了每种结构和语句的使用。最后结合天天向上学习打卡系统进行思政引导——踔厉奋发。在学习本章内容时,可以模仿实例,梳理算法流程,动手实践,熟练掌握 Python 流程控制语句的使用。

3.7 巩固训练

【训练 3.1】 编写程序,判断用户输入的年份是否为闰年(判断闰年的条件是:能被 400 整除或能被 4 整除但不能被 100 整除)。

【训练 3.2】 已知三角形边长,利用海伦公式求三角形面积和周长(海伦公式:若三角形的三条边长为 a、b、c,则 p=(a+b+c)/2,面积 area=$\sqrt{p*(p-a)*(p-b)*(p-c)}$)。

【训练 3.3】 某小学的学优生评定标准如下:语文、数学、英语和科学四科的总分不低

于 380 分，且每科成绩不低于 95 分。编程判断某位同学是否为学优生。

【训练 3.4】（简易版个税计算器）设某公司员工小王每月税前工资为 salary，五险一金等扣除为 insurance，其他专项扣除为 other，请编程计算小王每月应缴纳税额 tax 和实发工资 payroll（结果保留两位小数）。

注：应缴纳税额＝税前收入－5000（起征点）－五险一金扣除－其他扣除

个人所得税＝应缴纳税额×适用税率－速算扣除数

实发工资＝税前工资－个人所得税－五险一金

税率表如表 3.3 所示。

表 3.3　最新居民个人工资、薪金所得税税率表

级数	应纳税所得额	预扣率（%）	速算扣除数
1	不超过 3000 元的部分	3	0
2	超过 3000 元至 12000 元的部分	10	210
3	超过 12000 元至 25000 元的部分	20	1410
4	超过 25000 元至 35000 元的部分	25	2660
5	超过 35000 元至 55000 元的部分	30	4410
6	超过 55000 元至 80000 元的部分	35	7160
7	超过 80000 元的部分	45	15160

【训练 3.5】 幸运 52 猜数游戏（模仿幸运 52 中猜价钱游戏，编写程序，计算机随机产生一个正整数，让用户猜，并提醒用户猜大了还是猜小了，直到用户猜对为止，计算用户猜对一个数所用的秒数）。

【训练 3.6】 求出所有的水仙花数（水仙花数是指一个 3 位数，其每位数字的 3 次幂之和等于其本身。例如：1＊1＊1＋5＊5＊5＋3＊3＊3＝153）。

【训练 3.7】 模拟打印超市购物小票。输入商品名称、价格、数量，算出应付金额。用户输入大额面值，实现找零和抹零功能，最后打印购物小票。运行结果如图 3.19 所示。

```
Python超市收银系统
商品个数:2
商品名称  单价      数量
egg 5.85 1.89
milk 48.5 1
应付金额:59.56
实收:100
Python超市购物小票
共购买2件商品
商品名称  单价      数量
egg      5.85     1.89
milk         48.5     1.0
应付:59.56
实收:100.0
找零40.4
```

图 3.19　打印超市购物小票

【训练 3.8】 有一个分数数列 $\frac{2}{1},\frac{3}{2},\frac{5}{3},\frac{8}{5},\frac{13}{8},\cdots$，编程计算此数列的前 20 项之和（结果保留两位小数）。

【训练 3.9】 每行 10 个输出所有的 4 位“回文数”（“回文数”是一种特殊的数字，从左边读和从右边读的结果是一模一样的）。

【训练 3.10】 编程实现：输出 1～1000 中包含 3 的数字。如果 3 是连在一起的（如 233），则在数字前加 &；如果此数字是质数，则在数字后加上＊（例如，3，13＊，23＊，&33，43＊…&233＊…）。