第5章

程序结构：选择执行

条件选择语句是判断某个条件是否成立，然后选择执行程序中的某些语句块。与顺序结构比较，选择语句使程序不再完全按照语句的先后顺序执行，而是根据某种条件是否成立来决定执行的路径，它体现了程序具有逻辑判断的基本智能。

5.1 选择——条件执行语句

1. 单条件选择结构语法

单条件选择语句的语法如下：

```
1  if 条件表达式:          # 条件表达式只允许用关系运算符" == ";不允许用赋值运算符" = "
2      语句块              # 如果条件表达式为 True,则执行语句块,执行完后结束 if 语句
```

2. 双条件选择结构语法

双条件选择语句结构如图 5-1 所示，双条件选择语句的语法如下：

```
1  if 条件表达式:          # 条件表达式只允许用关系运算符" == ";不允许用赋值运算符" = "
2      语句块 1            # 如果条件表达式为 True,则执行语句块 1,执行完后结束 if 语句
3  else:                   # 否则
4      语句块 2            # 如果条件表达式为 False,则执行语句块 2,执行完后结束 if 语句
```

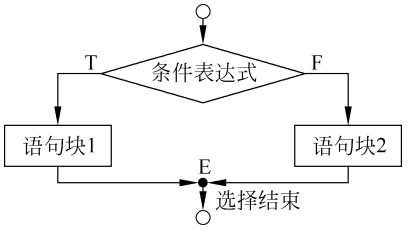

图 5-1 双条件选择语句结构

在 if 语句中，保留字 if 可以理解为“如果”；条件表达式往往采用关系表达式(如 x>=60)，条件表达式的值只能是 True(T)或者 False(F)；语句结尾的冒号(:)可以理解为“则”；保留字 else 可以理解为“否则”。

注意：if 语句的冒号(:)不可省略，if 语句块可以有多行，但是 if 语句块内部必须缩进 4 个空格，并且保持垂直对齐。

3. 条件选择语句执行方法

(1) 无论条件表达式的值为真(True)还是假(False)，一次只能执行一个分支的语句块。简单地说，程序不能同时执行语句块 1 和语句块 2。

（2）无论执行哪一个语句块，都必须能脱离选择结构。

【例 5-1】　双条件选择语句说明如图 5-2 所示。

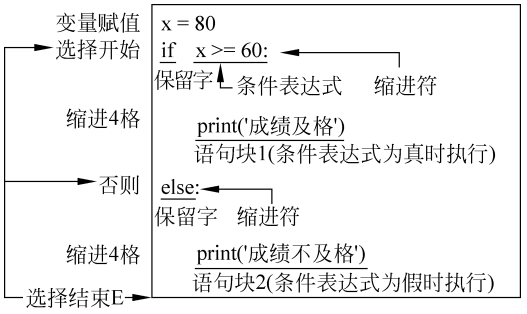

图 5-2　双条件选择语句说明

案例分析：Python 根据条件表达式的值为 True 还是 False，来决定怎样执行 if 语句中的代码。如果条件表达式的值为 True，Python 就执行 if 后的语句块 1；如果条件表达式的值为 False，Python 将忽略语句块 1，选择执行 else 后的语句块 2。程序如下：

```
1   x = 80                          # 变量赋值
2   if x >= 60:                     # 如果条件表达式 x >= 60 为真，则执行语句 3
3       print('成绩及格')           # 执行完本语句后，结束 if 语句(不执行语句 4～5)
4   else:                           # 否则，执行语句 5(x >= 60 为假时，跳过语句 3)
5       print('成绩不及格')         # 执行完本语句后，结束 if 语句
>>>  成绩及格                       # 程序输出
```

双条件选择程序结构适用于二选一的应用场景，两个语句块中总有一个语句块会被执行。如果有多种选择的要求，应当采用多条件选择语句结构。

4. 多条件选择结构语法

当条件选择有多个项目时，可以使用多条件选择语句 if-elif。语句 elif 是 else if 的缩写，语句结构如图 5-3 和图 5-4 所示，多条件选择的语法如下：

```
1   if 条件表达式 1:                # 判断条件 1
2       语句块 1                    # 条件 1 为 True，则执行语句块 1，执行完后结束 if - elif 语句
3   elif 条件表达式 2:              # 条件 1 不满足时，继续判断条件 2
4       语句块 2                    # 条件 2 为 True，则执行语句块 2，执行完后结束 if - elif 语句
5   elif 条件表达式 3:              # 条件 2 也不满足时，继续判断条件 3
6       语句块 3                    # 条件 3 为 True，则执行语句块 3，执行完后结束 if - elif 语句
7   else:                           # 条件 1、2、3 都不满足时(注意，此处没有条件表达式)
8       语句块 4                    # 执行语句块 4，执行完后结束 if - elif 语句
```

5. 条件选择语句注意问题

（1）if、elif 都需要写条件表达式，但是 else 不需要写条件表达式。

（2）else、elif 需要与 if 一起使用。

（3）if、elif、else 行尾都有英文冒号（:）。

（4）选择语句 if 从上往下判断，当表达式为 True 时，将该条件选择对应的语句块执行完后，忽略掉剩下的 elif 和 else 语句块，结束 if 语句，即一次只执行一个分支。

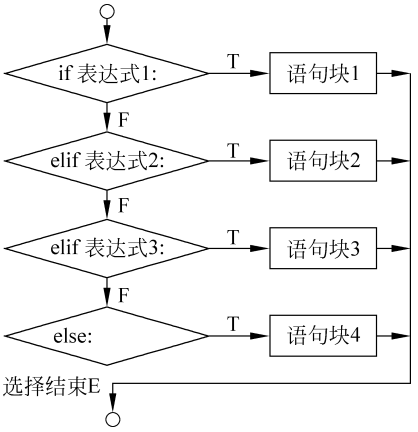

图 5-3　多条件选择语句结构

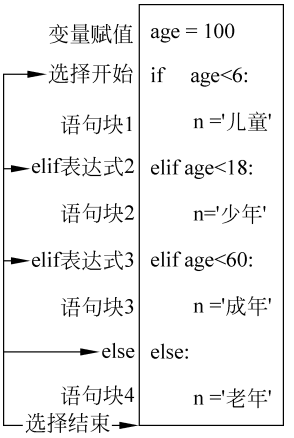

图 5-4　多条件选择语句案例

(5) 多条件语句要多注意条件之间的包含关系,以及变量取值的变化范围。

【例 5-2】 用多条件选择语句实现“石头—剪刀—布”游戏,程序如下:

```
1   import random                                          # 导入标准模块
2
3   player = int(input('【1 = 石头,2 = 剪刀,3 = 布】请出拳:'))    # 玩家输入 1～3 的整数
4   computer = random.randint(1, 3)                        # 程序生成 1～3 的随机数
5   if ((player == 1 and computer == 2) or                 # 如果玩家为 1,程序为 2
6       (player == 2 and computer == 3) or                 # 或者玩家为 2,程序为 3
7       (player == 3 and computer == 1)):                  # 或者玩家为 3,程序为 1
8       print('你赢了,好厉害啊!')                           # 打印玩家胜利
9   elif player == computer:                               # 否则,玩家与程序相同时
10      print('欧耶!平局,再来一盘!')                        # 打印平局
11  else:                                                  # 否则
12      print('哦豁,你输了,还玩吗?')                        # 打印玩家输
>>> 【1 = 石头,2 = 剪刀,3 = 布】请出拳:1                    # 程序输出
    你赢了,好厉害啊!
```

注意:函数 input()返回值为字符串,需要用 int()函数将其转换为整数。

5.2　选择——三元条件选择

1. 三元条件选择语法

三元运算是有 3 个操作数(左值、条件表达式值、右值)的程序语句。在简单条件判断程序中,经常采用三元运算进行条件赋值,三元条件选择语法如下:

```
1   a = 左值 if 条件表达式 else 右值
```

如图 5-5 所示,三元条件选择语句中,首先进行条件判断,条件表达式的值为真(True)时,将左值赋给变量;条件表达式的值为假(False)时,将右值赋给变量。

三元条件选择是单行语句,不是语句块;且语句行尾没有冒号,下一行语句无须缩进。

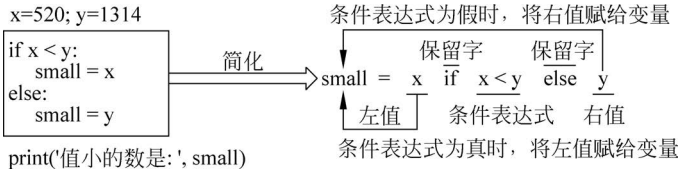

图 5-5　双条件选择与三元条件选择语句的比较

2. 三元条件选择程序设计

三元条件选择语句可以将双条件选择语句简化为一行，这方便了某些特殊应用。

【例 5-3】 双条件选择语句。

```
1   x = 520; y = 1314
2   if x < y:
3       small = x
4   else:
5       small = y
6   print('值小的数是:', small)
>>> 值小的数是: 520
```

【例 5-4】 三元条件选择语句。

```
1   x = 520; y = 1314
2   small = x if x < y else y
3   print('值小的数是:', small)
>>> 值小的数是: 520
```

【例 5-5】 用三元条件选择判断奇数和偶数，程序如下：

```
1   num = int(input('请输入一个整数:'))
2   result = '偶数' if num % 2 == 0 else '奇数'        # 用三元运算判断奇数和偶数
3   print('您输入的是:', result)
>>> 请输入一个整数:28                                  # 程序输出
    您输入的是: 偶数
```

案例 4：双条件选择——一元二次方程求根

1. 一元二次方程求根方法

一元二次方程求根的方法很多，最常用的是公式法。一元二次方程都可以化为 $ax^2+bx+c=0(a\neq 0)$的形式，其中 x^2 是二次项，a 是二次项系数；bx 是一次项，b 是一次项系数；c 是常数项。它的求根公式如下：

$$x=\frac{-b\pm\sqrt{b^2-4ac}}{2a} \tag{5-1}$$

其中 b^2-4ac 称为一元二次方程根的判别式，用符号 Δ 表示。当 $\Delta<0$ 时，方程没有实数根（实数范围内无解）；当 $\Delta=0$ 时，方程有两个相等的实数根，它们是 $x=-b/(2a)$；当 $\Delta>0$ 时，方程有两个不等的实数根，这时只要把方程的三个系数代入公式(5-1)计算即可。

2. 程序设计：用公式法求解一元二次方程

【例 5-6】 对方程 $44x^2+123x-54=0$ 求根。

案例分析：对一元二次方程编程求根时，需要考虑以下问题。

(1) 代数式求解需要将代数式转换成程序规定的算术表达式，转换方法可以参考表 2-3。

(2) 方程求根需要对有无实数根进行判断，因此需要用到 if 条件选择语句。

(3) 解一元二次方程需要进行开方运算，因此需要导入 math 数学模块。

案例实现程序如下：

```
1  import math                                          # 导入标准模块
2
3  print('请输入方程 a * x ** 2 + b * x + c = 0 的系数:')   # 打印提示信息
4  a = float(input('二次项系数 a = '))                    # 将输入字符串转换为浮点数
5  b = float(input('一次项系数 b = '))
6  c = float(input('常数项 c = '))
7  delta = b * b - 4 * a * c                            # 计算方程判别式 delta
8  if delta < 0:                                        # 如果判别式 delta 小于 0
9      print('方程无解')                                  # 打印输出信息
10     exit()                                           # 函数 exit()为退出程序
11 else:                                                # 否则
12     root1 = (-b + math.sqrt(delta))/(2 * a)          # 计算方程根 1
13     root2 = (-b - math.sqrt(delta))/(2 * a)          # 计算方程根 2
14 print(f'x1 = {root1:.2f}    x2 = {root2:.2f}')       # "f"为格式化输出,":.2f"为保留 2 位
                                                        # 小数
>>> 请输入方程 a * x ** 2 + b * x + c = 0 的系数:          # 程序输出
    二次项系数 a = 44
    一次项系数 b = 123
    常数项 c = -54
    x1 = 0.39    x2 = -3.18
```

程序第 14 行，打印语句中，参数 f 为格式化，花括号"{ }"中 root 为变量名，冒号为分隔符，参数".2f"为保留 2 位小数。具体语法参见 4.4 节。

3. 程序设计：用符号计算求解一元二次方程

【例 5-7】 用符号计算对一元二次方程 $ax^2+bx-c=0$ 求根，程序如下：

```
1  import sympy                                         # 导入第三方包
2
3  print('请输入方程 a * x ** 2 + b * x + c = 0 的系数:')
4  a = float(input('输入二次项系数 a = '))                 # 输入方程系数
5  b = float(input('输入一次项系数 b = '))
6  c = float(input('输入常数项 c = '))
7  x = sympy.symbols('x')                               # 定义符号 x
8  equation = sympy.Eq(a * x ** 2 + b * x + c, 0)       # 构造方程
9  root = sympy.solve(equation, x)                      # 求解方程
10 print(f'x1 = {root[0]};x2 = {root[1]}')               # 打印结果
```

```
>>> 请输入方程 a * x ** 2 + b * x + c = 0 的系数:              # 程序输出
    输入二次项系数 a = 44
    输入一次项系数 b = 123
    输入常数项 c = -54
    x1 = -3.18123904902659;x2 = 0.385784503572045
```

4. 程序设计：判断闰年

【例 5-8】 输入一个年份数，判断它是否为闰年，程序如下：

```
1  year = int(input('请输入年份【如 2023】:'))
2  if  year % 400 == 0 or year % 4 == 0 and year % 100 != 0:     # 符号 % 为模运算
3      print('闰年')
4  else:
5      print('平年')
>>> 请输入年份【如 2023】:2000                                   # 程序输出
    闰年
```

程序单词：delta(德尔塔，Δ)，Eq(等于)，equation(方程)，exit(退出)，float(字符串转浮点数)，input(输入)，math(数学模块)，root(根)，year(年)。

5. 编程练习

练习 5-1：编写和调试例 5-6 和例 5-7 的程序，掌握 if 语句的使用方法。

案例 5：多条件选择——BMI 指数计算

1. 程序案例：计算 BMI 指数

BMI(Body Mass Index，体质指数)是国际上常用于衡量人体健康程度的指标(见表 5-1)，BMI 值过高或过低都不利于身体健康。

表 5-1　BMI 指数参考标准

BMI 分类	WHO 标准	中国参考标准 NAT	相关疾病发病的危险性
体重过低	BMI＜18.5	BMI＜18.5	低(其他疾病危险性增加)
正常范围	18.5≤BMI＜25	18.5≤BMI＜24	平均水平
超重	BMI≥25	BMI≥24	增加
肥胖前期	25≤BMI＜30	24≤BMI＜28	增加
Ⅰ度肥胖	30≤BMI＜35	28≤BMI＜30	中度增加
Ⅱ度肥胖	35≤BMI＜40	30≤BMI＜40	严重增加
Ⅲ度肥胖	BMI≥40.0	BMI≥40.0	非常严重增加

BMI 指数计算方法为：BMI＝体重(kg)/身高^2(m)。

【例 5-9】 设计程序，根据输入数据，判断 BMI 指标是否正常。

案例分析：编程实现 BMI 指数计算时，需要考虑以下问题：

(1) 对表 5-1 进行简化和分类，使它更适合于程序处理(如多条件选择结构)。用专业术语来讲，就是创建表格形式的数学模型。分类结果如表 5-2 所示。

表 5-2　适合程序处理的 BMI 指标分类

BMI 指标	WHO 标准	NAT 标准
BMI＜18.5	偏瘦	偏瘦
18.5＜＝BMI＜24	正常	正常
24＜＝BMI＜25	正常	偏胖
25＜＝BMI＜28	偏胖	偏胖
28＜＝BMI＜30	偏胖	肥胖
BMI＞30	肥胖	肥胖

(2) 根据表 5-2,利用多条件选择语句进行编程处理。

2. 程序设计：计算 BMI 指数

案例实现程序如下：

```
1  height = (eval(input('请输入您的身高(厘米):')))/100          # 输入数据
2  weight = eval(input('请输入您的体重(公斤):'))                # 输入数据
3  BMI = weight /((height**2)                                  # 计算 BMI 值
4  print(f'您的 BMI 指数为:{BMI:.2f}')                          # 打印指数值(2 位小数)
5  who = nat = ''                                              # 变量初始化(为空)
6  if BMI < 18.5:                                              # BMI 指标判断
7      who, nat = '偏瘦', '偏瘦'                               # 指标赋值
8  elif 18.5 <= BMI < 24:                                      # 多重判断
9      who, nat = '正常', '正常'
10 elif 24 <= BMI < 25:
11     who, nat = '正常', '偏胖'
12 elif 25 <= BMI < 28:
13     who, nat = '偏胖', '偏胖'
14 elif 28 <= BMI < 30:
15     who, nat = '偏胖', '肥胖'
16 else:
17     who, nat = '肥胖', '肥胖'
18 print(f'国际 BMI 标准:{who}; 国内 BMI 标准:{nat}')          # 打印结论
>>> 请输入您的身高(厘米):175                                  # 程序输出
    请输入您的体重(公斤):76
    您的 BMI 指数为:24.82
    国际 BMI 标准:正常; 国内 BMI 标准:偏胖
```

程序单词：BMI(体质指数),eval(字符转数字),format(格式),height(身高),nat(国内标准值),weight(体重),who(国际标准值)。

3. 编程练习

练习 5-2：输入三角形三条边的长度,判断能否组成三角形。如果能够组成三角形,判断是普通三角形,还是直角三角形,还是等腰三角形。程序输出如下所示：

```
>>> 请输入三角形 a 边长度:12
    请输入三角形 b 边长度:23
    请输入三角形 c 边长度:45
    不能组成三角形
```