项目 3

流程控制

项目介绍

结构化程序不仅可以使计算机程序设计更加系统、有条理，还可以使软件的执行效率得到显著的提升、软件的调试和维护的代价得以降低。程序的 3 种基本结构是顺序结构、选择结构和循环结构。

任务安排

任务 1　进行结构化程序设计。

任务 2　选择结构程序设计方法。

任务 3　进行循环结构程序设计。

学习目标

- ✧ 掌握结构化程序设计和简单流程图绘制的方法。
- ✧ 掌握 Python 顺序结构程序设计方法。
- ✧ 掌握 Python 选择结构程序设计方法。
- ✧ 掌握 Python 循环结构程序设计方法。

任务 1　进行结构化程序设计

本任务要求学生了解计算机中的结构化程序设计，学会流程图的绘制；掌握 Python 中顺序结构程序的设计方法，会使用输入函数和输出函数。

3.1.1　结构化程序设计

20 世纪 60 年代，被称为结构程序设计之父的荷兰著名计算机科学家艾兹格·W·迪科斯彻（E.W.Dijikstra）提出了 3 种基本控制结构，不仅使计算机程序设计更加系统、有条理，也使程序的执行效率得到显著的提升、调试和维护的代价得以降低。这 3 种基本结构就是顺序结构、选择结构和循环结构。

为了能更加清晰、直观地表示程序设计的思路，可以利用图形的方式将程序算法的编程思路表示出来，这种图形被称为流程图（Flowchart）。在国家标准规范《GB/T 1526—1989 信息处理——数据流程图、程序流程图、系统流程图、程序网络图和系统资源图的文件编制符号及约定》中，给出了各种表示操作、数据、流向的符号。

- 端点符号：表示程序流程的起始或结束。结构化程序应仅有一个起点、一个终点，如图 3.1 所示。
- 数据符号（平行四边形）：表示数据，如图 3.2 所示。
- 处理符号（长方形）：表示各种处理功能，如图 3.3 所示。

图 3.1　端点符号

图 3.2　数据符号

图 3.3　处理符号

- 判断符号（菱形）：表示进行判断，有一个入口，可以有若干个可供选择的出口，但针对具体的条件仅有一个流经的出口，如图 3.4 所示。
- 基本流线符号（直线）：表示数据流或控制流，可以利用箭头表示流向，如图 3.5 所示。

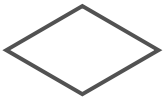

图 3.4　判断符号

图 3.5　基本流线符号

在编写程序前，可以先利用流程图的形式，将程序算法的编程思路表示出来，使程序算法被更加清晰、直观地展现出来，帮助用户进一步优化和调试程序。

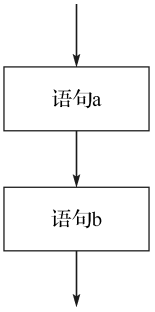

图 3.6　顺序结构流程图

1．顺序结构

顺序结构是最简单的一种程序结构，是指程序语句按照自上而下的顺序依次执行，其流程图如图 3.6 所示。在先执行完语句 a 后，再执行语句 b。

2．选择结构

选择结构也被称为判断结构、条件结构或分支结构，是指先判断是否满足给定的条件，再执行对应的操作语句，其流程图如图 3.7 所示。

先判断条件是否成立，如果条件成立，则执行语句 a；如果条件不成立，则执行语句 b。语句 a 和语句 b，对每次执行来说，都只能选择其中一条流程路径。

3．循环结构

循环结构可以在满足指定条件的情况下重复执行某些语句，提高程序语句的使用效率。循环结构有多种形式，可以先判断条件是否成立，当条件成立时再执行循环；也可以先执行语句，再进行条件判断，当条件成立时跳出循环，如图 3.8 所示。

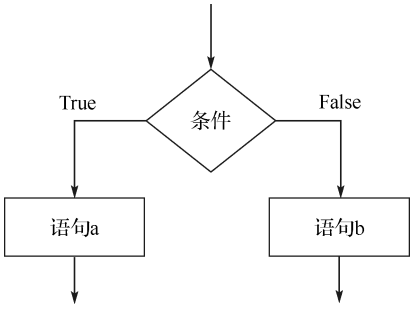

图 3.7　选择结构流程图

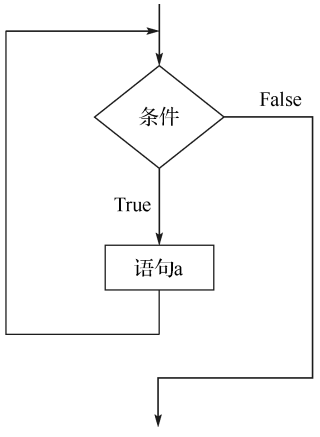

图 3.8　循环结构流程图

当条件成立时，可以不断重复执行语句 a；当条件不成立时，跳出循环，往下执行。

一般程序都是由这 3 种基本结构组成的。就算是大型的复杂程序，也可以像乐高积木

一样，利用顺序、选择和循环这 3 种结构，高效、合理地搭建出来，并且可以方便地进行调试和维护。

3.1.2　输入函数

人们编写程序的目的是处理信息。人们要先向计算机输入需要其处理的数据，再由计算机执行算法按照用户的要求处理数据，当数据处理完成后，再输出，并将结果反馈给用户。对于一个结构化程序来说，不管程序内部多么复杂，都包括输入、处理、输出这 3 部分功能模块。

输入模块负责接收程序要处理的数据，有时可以直接在程序中对初始值进行赋值。但是更多的时候，需要用户从输入设备向计算机输入要处理的数据。在 Python 中，负责接收用户输入的是 input()函数，其语法格式如下。

```
input(prompt=None,/)
```

input()函数可以从键盘上以字符串的形式接收输入，参数 prompt 指的是指示字符串，可以省略，当未省略时，该参数值将出现在屏幕上。一般建议给出提示字符串，以为用户带来更好的体验感。示例代码如下。

```
>>> score = input('请输入分值：')
请输入分值：80
>>> score
'80'
```

当该语句执行时，先在屏幕上显示出提示文字“请输入分值：”，在接收了用户输入的 80 后，变量 score 被赋值为“80”，注意，此时“80”是一个字符串。

input()函数每次都只能从键盘上接收一个值。当然，如果需要同时接收多个值，则可以利用 split()函数。下面这条语句就可以同时在一行中接收两个输入的值。

```
>>> a,b = input('请输入长和宽的值：').split()
请输入长和宽的值：5 3
>>> a
'5'
>>> b
'3'
```

注意，此时在输入时，“5”和“3”之间是用空格分隔的，而且 a 和 b 两个变量接收的仍然是字符串类型的值。

3.1.3 类型转换函数

因为 Python 的输入函数 input()接收的是字符串值，所以在输入数值进行处理时需要进行数据类型的转换。Python 提供了内置函数，可以将字符串类型的数据转换为整型、浮点型或复数型等类型的数据。

示例代码如下。

```
int()    #可以将输入的数字格式的字符串转换为整数
float()  #可以将字符串转换为浮点数
```

3.1.4 输出函数

输出模块负责将处理完成的结果反馈给用户。在一般情况下，处理完成的结果利用输出函数可以直接显示在屏幕上。在 Python 中，负责将处理结果显示在屏幕上的是 print()函数，其常用的语法格式如下。

```
print(value, ..., sep=' ', end='\n')
```

利用 print()函数，可以输出多个变量或常量的值，多个值之间是用逗号分隔的。参数 sep 是指在输出的值之间插入的分隔字符，在默认情况下是一个空格；参数 end 是指在最后的值之后追加的结束字符，在默认情况下是换行符。示例代码如下。

```
>>> print('Hello, world!')          #输出字符串
Hello, world!
>>> print(2)                       #输出数值
2
>>> a = 5
>>> print(a)                       #输出变量值
5
>>> print('a 的值是',': ',a)         #输出多个值
a 的值是 :  5
>>> print(1,3,5,sep=';')           #以分号为分隔符输出 3 个数字
1;3;5
#以分号为结束符输出 3 条语句
>>> print(1,end=';');print(3,end=';');print(5,end=';')
1;3;5;
```

3.1.5　任务实现

1．绘制流程图

针对任务描述的要求，绘制出流程图，如图 3.9 所示。

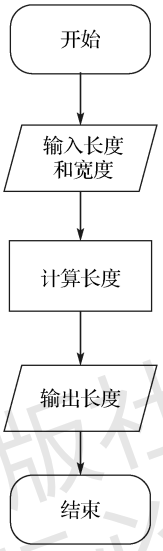

图 3.9　流程图

2．设计程序

根据所绘制的流程图，可以看到这是一个非常典型的顺序结构程序，只需要自上而下依次执行，就可以得到结果。在程序中，一般都包含了输入、处理和输出模块。新建程序并输入设计好的代码，如图 3.10 所示。

3．调试运行

程序运行后，输入窗户的长度和宽度，就可以得到所需要购买的彩带长度了。程序运行结果如图 3.11 所示。

```
File  Edit  Format  Run  Options  Window  Help
# 顺序结构程序设计

# 输入并进行数据类型的转换
a = int(input("请输入窗户的长度："))
b = int(input("请输入窗户的宽度："))

# 数据处理
c = (a+b)*2

# 输出
print("需要彩带的长度为：",c)
```

图 3.10　输入代码

```
请输入窗户的长度：5
请输入窗户的宽度：2
需要彩带的长度为：14
```

图 3.11　程序运行结果

任务拓展

如果输入的是小数，则应该如何设计这个程序？

任务 2　进行选择结构程序设计

场景一：李同学是学校某社团的负责人，他今天要去社团办公室为社团成员出具社团经历证书。出门前他先看了一下天气预报，如果天气预报说会下雨，他就带上雨伞。

场景二：在学校门口，值班的门卫询问他是否携带了学生证。如果携带了学生证，则可以从左门直接出示学生证后进入校园；如果没有携带学生证，则需要从右门扫码查验后进入校园。

场景三：到了社团办公室，李同学拿出了所有社团成员的参加活动的记录。参加活动超过 12 次（含 12 次）的同学，将获得“优秀”的社团经历证书；如果参加活动次数未到 12 次但超过了 10 次（含 10 次），将获得“良好”的社团经历证书；如果参加活动次数未到 10 次但超过了 6 次（含 6 次），则获得“合格”的社团经历证书；如果参加活动次数连 6 次都没有达到，就不能获得社团经历证书。

请利用选择结构来描述以上 3 个场景。

3.2.1　选择结构

我们每天都面临着许多选择——天气有没有下雨？如果下雨了，就要带上雨伞。我们也经常会碰到要满足某些条件才能完成的事情——考试成绩超过 60 分，才能算通过考试，等等。在结构化程序中，当满足指定条件时才能执行对应操作的结构称为选择结构、分支结构或条件结构。

有时，在满足指定条件后，才能执行指定操作，如果没有满足条件，则什么都不做，这种结构被称为单分支选择结构，因为只有一边分支有语句要执行，其流程图如图 3.12 所示。

有时，在满足指定条件后，执行某个操作；而在未满足条件时，执行另一个操作，这种结构被称为双分支选择结构，即两边分支都有可能被执行到，其流程图如图 3.13 所示。

有时，在某条分支中，要判断是否满足某个条件，这种结构被称为多分支选择结构或嵌套分支结构，其流程图如图 3.14 所示。

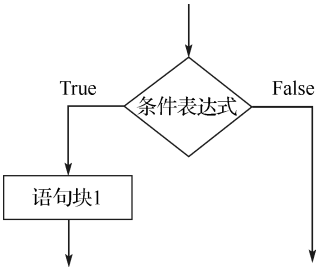

图 3.12　单分支选择结构流程图

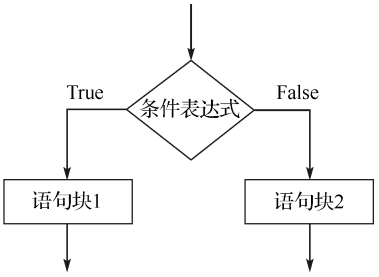

图 3.13　双分支选择结构流程图

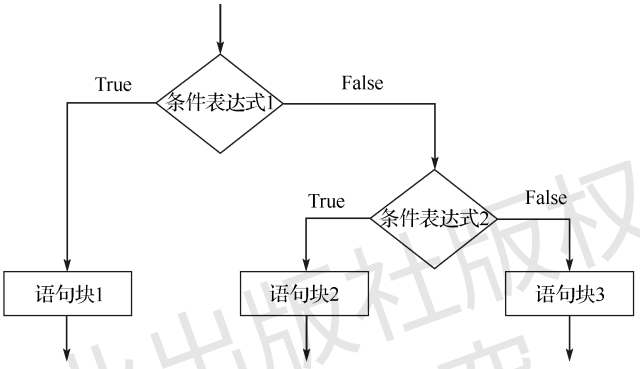

图 3.14　多分支选择结构流程图

3.2.2　单分支语句

在 Python 中，可以利用 if 语句来实现如图 3.12 所示的单分支选择结构，其语法格式如下。

```
if <条件表达式>:
    语句块 1
```

当条件表达式的值为 True 时，执行语句块 1 中的语句；当条件表达式的值为 False 时，直接执行语句块 1 后面没有缩进的语句。

注意事项如下。

- 条件表达式后面有冒号“:”。
- 语句块 1 前面有缩进。
- 当取消缩进时，表示已经退出条件语句体。

单分支语句示例程序如图 3.15 所示。

当“1>2”条件未成立时，不执行语句块中的“print('条件成立')”语句，直接退出单分支选择结构，执行后续语句“print('继续执行')”；当“1<2”条件成立时，执行语句块

中的“print('条件成立')”语句，接着执行后续语句“print('完成')”。程序运行结果如图 3.16 所示。

File Edit Format Run Options Window Help

```
# 单分支语句示例

if 1>2:     # 条件未成立
    print('条件成立')
print('继续执行')

if 1<2:     # 条件成立
    print('条件成立')
print('完成')
```

图 3.15　单分支语句示例程序

```
继续执行
条件成立
完成
```

图 3.16　单分支语句示例程序运行结果

3.2.3　双分支语句

在 Python 中，可以利用 if…else 语句来实现如图 3.13 所示的双分支选择结构，其语法格式如下。

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

当条件表达式的值为 True 时，执行语句块 1 中的语句；当条件表达式的值为 False 时，执行语句块 2 中的语句。

注意事项如下。

- if 和 else 此时应成对出现。
- else 后面也有冒号。
- 语句块 1、语句块 2 前面都有缩进，语句块内的缩进应该一致。
- 当取消缩进时，表示已经退出条件语句体。

双分支语句示例程序如图 3.17 所示。

当“1>2”条件未成立时，执行 else 语句块中的“print('条件不成立')”语句，并退出双分支选择结构，执行后续语句“print('完成')”；当“1<2”条件成立时，执行语句块中的“print('条件成立')”语句，接着退出双分支选择结构，执行后续语句“print('完成')”。程序运行结果如图 3.18 所示。

```
File  Edit  Format  Run  Options  Window  Help
# 双分支语句示例

if 1>2:
    print('条件成立')
else:
    print('条件不成立')
print('完成')

if 1<2:
    print('条件成立')
else:
    print('条件不成立')
print('完成')
```

图 3.17　双分支语句示例程序

```
条件不成立
完成
条件成立
完成
```

图 3.18　双分支语句示例程序运行结果

3.2.4　多分支语句

有时仅利用双分支选择结构并不能解决所有的条件分支问题。此时，就需要利用多分支选择结构（嵌套分支结构）。

例如，老师在给出学生成绩时，不同的成绩会对应“优秀”、“良好”、“中等”、“及格”与“不及格”5 个不同的等级，这就需要用到多分支选择结构，其流程图如图 3.19 所示。

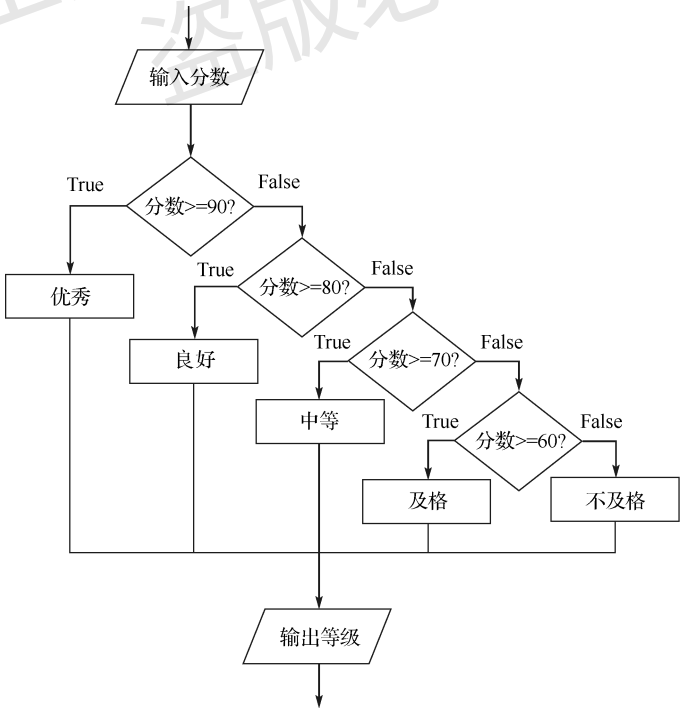

图 3.19　多分支选择结构流程图

此时如果在 if 的语句体中再嵌套多个 if 语句，虽然可以达到任务要求，但是非常复杂，程序难以阅读。Python 提供了多分支选择结构，可以比较方便、高效地解决这种问题，其语法格式如下。

```
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 n>:
    语句块 n
else:
    语句块 n+1
```

程序将依次判断条件表达式，当某个条件表达式的值为 True 时，执行其分支中的语句块；当某条分支中的语句被执行后，退出整个选择结构；当所有的条件表达式的值都为 False 时，执行 else 中的语句块。

多分支语句示例程序如图 3.20 所示。

```
# 多分支示例
score = int(input('请输入分数：'))
if score >= 90:
    print('优秀')
elif score >= 80:
    print('良好')
elif score >= 70:
    print('中等')
elif score >= 60:
    print('及格')
else:
    print('不及格')
```

图 3.20　多分支语句示例程序

当程序执行时，输入分数“83”，此时程序将依次判断是否满足条件，当不满足条件“score >= 90”，但满足下一个条件“score >= 80”时，执行该分支中的语句“print('良好')”，并退出整个选择结构。程序运行结果如图 3.21 所示。

```
请输入分数：83
良好
```

图 3.21　多分支语句示例程序运行结果

3.2.5　任务实现

1．绘制流程图

针对任务描述的要求，绘制出场景一的流程图，如图 3.22 所示。

针对任务描述的要求，绘制出场景二的流程图，如图 3.23 所示。

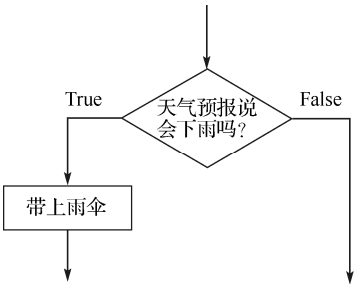

图 3.22　场景一的流程图

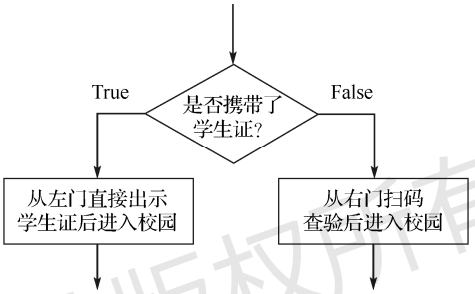

图 3.23　场景二的流程图

针对任务描述的要求，绘制出场景三的流程图，如图 3.24 所示。

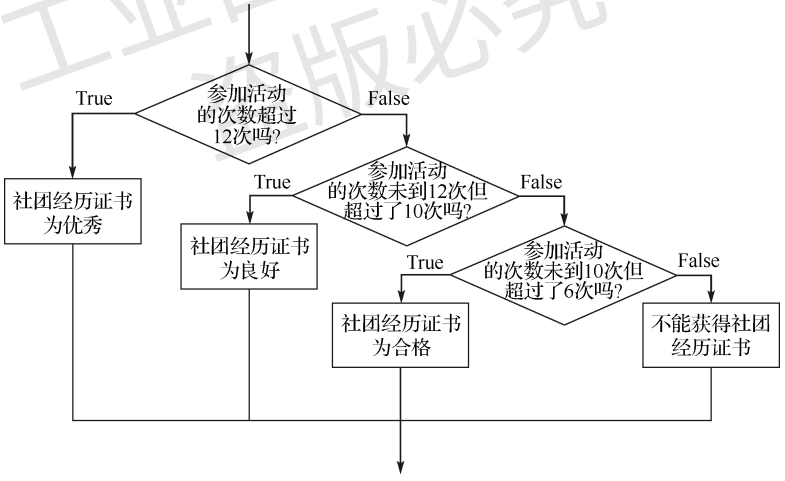

图 3.24　场景三的流程图

2．设计程序并调试运行

场景一：由如图 3.22 所示的流程图可知，这是一个非常典型的单分支程序。新建程序并输入设计好的代码，如图 3.25 所示。

```
# 场景一：单分支示例程序
n = input('天气预报说会下雨吗？（Y/N）')
if n == 'Y':  # 注意大写字母
    print('带上雨伞')
```

图 3.25　单分支示例程序

在执行程序时，如果输入“Y”，则程序运行结果如图 3.26 所示。

```
天气预报说会下雨吗？（Y/N）Y
带上雨伞
```

图 3.26　单分支示例程序运行结果（1）

如果输入“N”，将不显示提醒信息，程序运行结果如图 3.27 所示。

```
天气预报说会下雨吗？（Y/N）N
```

图 3.27　单分支示例程序运行结果（2）

场景二：由如图 3.23 所示的流程图可知，这是一个非常典型的双分支程序。新建程序并输入设计好的代码，如图 3.28 所示。

```
# 场景二：双分支示例程序
n = input('是否携带了学生证？（Y/N）')
if n == 'Y':  # 注意大写字母
    print('从左门直接出示学生证后进入校园')
else:
    print('从右门扫码查验后进入校园')
```

图 3.28　双分支示例程序

在执行程序时，如果输入“Y”，则程序运行结果如图 3.29 所示。

```
是否携带了学生证？（Y/N）Y
从左门直接出示学生证后进入校园
```

图 3.29　双分支示例程序运行结果（1）

如果输入“N”，则程序运行结果如图 3.30 所示。

```
是否携带了学生证？（Y/N）N
从右门扫码查验后进入校园
```

图 3.30　双分支示例程序运行结果（2）

场景三：由如图 3.24 所示的流程图可知，这是一个非常典型的多分支程序。新建程序并输入设计好的代码，如图 3.31 所示。

```
# 场景三：多分支示例程序
n = int(input('请输入参加活动的次数：'))
# 注意input接收的是一个字符串
if n >= 12:
    print('社团经历证书为优秀')
elif n >= 10:
    print('社团经历证书为良好')
elif n >= 6:
    print('社团经历证书为合格')
else:
    print('不能获得社团经历证书')
```

图 3.31　多分支示例程序

在执行程序时，分别输入“15”、“11”、“6”和“5”，程序运行结果如图 3.32 所示。

```
请输入参加活动的次数：15
社团经历证书为优秀

请输入参加活动的次数：11
社团经历证书为良好

请输入参加活动的次数：6
社团经历证书为合格

请输入参加活动的次数：5
不能获得社团经历证书
```

图 3.32　多分支示例程序运行结果

任务拓展

李同学还想为社团的管理程序设置登录密码，如果输入正确的密码“gikest”，则显示欢迎信息，否则拒绝登录，请利用双分支语句实现这个功能。

任务 3　进行循环结构程序设计

3.3.1　循环结构

有时，我们会碰到需要重复做的事情。例如，喜欢听的歌曲经常会被反复地播放。此时，我们只需要重复播放这一首歌曲，而不需要将这首歌在歌单里复制 *N* 遍。同样，在编

写程序时，也会碰到需要重复多次执行的语句。为了提高程序的利用率，我们可以重复地执行同一段代码，而不需要将这段代码复制多遍。在结构化程序中，当在特定条件下会重复执行某些语句时，该结构被称为循环结构。

常见的循环结构有当型循环和直到型循环。当型循环先判断条件是否成立，当条件成立时执行循环；直到型循环先执行语句，再进行条件判断，当条件不成立时，一直执行循环，直到条件成立时跳出循环。当型循环与直到型循环的流程图如图 3.33、图 3.34 所示。

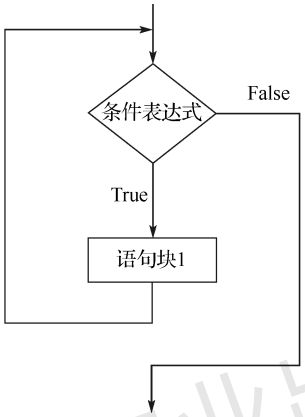

图 3.33　当型循环流程图

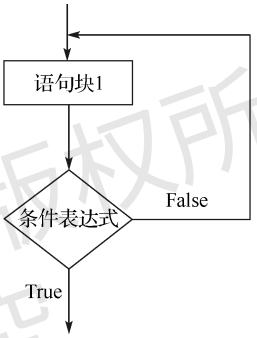

图 3.34　直到型循环流程图

在 Python 中，常用的有 while 循环和 for…in 循环，它们都属于当型循环。

3.3.2　while 循环

在 Python 中，用户可以利用 while 循环来实现循环结构，其语法格式如下。

```
while <条件表达式>:
    语句块 1
```

当条件表达式的值为 True 时，执行语句块 1 中的语句；语句块 1 执行完后，返回循环头部，继续判断此时条件表达式的值是否为 True，如果条件成立，则再次执行语句块 1，只有当条件表达式的值为 False 时，才会跳出循环体，直接执行语句块 1 后面没有缩进的语句。

注意事项如下。

- 条件表达式后面有冒号“:”。
- 语句块 1 前面有缩进。
- 当取消缩进时，表示已经退出循环体。

一般在 while 的循环体中，应该有对循环变量的控制语句，即循环的条件不应该是永远成立的，否则将进入死循环。

例如，如图 3.35 所示的程序，利用 while 循环求 1+2+……+10 的整数和。需要注意的是，在循环体中，要有能改变循环条件表达式的值的语句。

```
# while循环示例程序
nsum = 0
i = 1
while i <= 10:
    nsum += i
    i += 1  # 改变条件表达式的i的值
print('1+2+……+10的整数和为：',nsum)
```

图 3.35　while 循环示例程序

当 i<=10 时，执行循环体中的语句，否则退出循环。当每次循环执行时，变量的值都在变化，如表 3.1 所示。可以看出，变量 nsum 即为 1+2+……+10 的整数和，而变量 i 每次循环后都加 1，当 i=11 时，循环条件不成立，退出循环。

表 3.1　while 循环示例中的条件和变量值

序次	条件	nsum	i
初值	/	0	1
第 1 次循环	1<=10（成立）	1	2
第 2 次循环	2<=10（成立）	3	3
第 3 次循环	3<=10（成立）	6	4
第 4 次循环	4<=10（成立）	10	5
第 5 次循环	5<=10（成立）	15	6
第 6 次循环	6<=10（成立）	21	7
第 7 次循环	7<=10（成立）	28	8
第 8 次循环	8<=10（成立）	36	9
第 9 次循环	9<=10（成立）	45	10
第 10 次循环	10<=10（成立）	55	11
第 11 次循环	11<=10（不成立），退出循环		

while 循环示例程序运行结果如图 3.36 所示。

```
1+2+…… +10的整数和为： 55
```

图 3.36　while 循环示例程序运行结果

3.3.3　for…in 循环

while 循环在有明确的循环边界时使用比较方便，但是如果没有在循环体中设置可以改变循环条件的语句，则会进入死循环。在 Python 中，用户还可以利用 for…in 循环来实现循环结构，其语法格式如下。

```
for <元素> in <序列>:
    语句块 1
```

for…in 循环可以自动遍历序列中的每一个元素，对每一个元素都执行一次语句块 1。当遍历了序列中的每一个元素后，会自动结束循环，转而继续执行循环体后面的语句。

注意事项如下。

- for 和 in 此时应成对出现。
- for…in 语句后面有冒号“:”。
- 语句块 1 前面有缩进。
- 当取消缩进时，表示已经退出循环体。

图 3.37 所示为 for…in 循环示例程序。

```
# for...in循环示例程序
for i in 'computer':  #遍历字符串中的字母
    print(i)
```

图 3.37　for…in 循环示例程序（1）

```
c
o
m
p
u
t
e
r
```

图 3.38　for…in 循环示例程序运行结果（1）

此时，for…in 循环将自动遍历字符串“computer”中的每一个字符，对每一个字符都执行一次语句块“print(i)”。遍历了每一个字符后会自动结束循环。for…in 循环示例程序运行结果如图 3.38 所示。

如图 3.39 所示，for…in 循环将自动遍历列表“['hello','Python','world']”中的每一个元素，对每一个元素都执行一次语句块“print(i)”。遍历了列表中的所有元素后会自动结束循环。

```
# for...in循环示例程序
for i in ['hello','Python','world']:  #遍历列表中的元素
    print(i)
```

图 3.39　for…in 循环示例程序（2）

for…in 循环示例程序运行结果如图 3.40 所示。

```
hello
Python
world
```

图 3.40　for…in 循环示例程序运行结果（2）

3.3.4　for…in 循环与 range()函数

for…in 循环可以与 range()函数相结合，使循环的功能更加强大。range()函数的语法格式如下。

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

其中，start 表示起始值，stop 表示结束值，step 表示步长。

range()函数用于生成一个从起始值开始，到结束值（但不包含结束值）结束，间隔为步长值的整数序列。步长可省略，默认的步长为 1。range()函数也可以只指明结束值，如 range(stop)，此时默认的起始值为 0，步长为 1。

表 3.2 所示为 range()函数使用示例。

表 3.2　range()函数使用示例

示例	start	stop	step	具体序列值
range(2,8)	2	8	1（省略）	[2,3,4,5,6,7]
range(1,9,2)	1	9	2	[1,3,5,7]
range(5)	0（省略）	5	1（省略）	[0,1,2,3,4]

当 for…in 循环与 range()函数结合使用时，程序将遍历 range()函数指定的序列中的每一个数值，并重复执行循环体中的语句。例如，可以将利用 while 循环求 1+2+……+10 的整数和的示例，改写成利用 for…in 循环和 range()函数相结合来实现。

```
# for...in 循环与 range()函数示例程序
nsum = 0
for i in range(1,11):
    nsum += i
print('1+2+……+10的整数和为：',nsum)
```

图 3.41　for…in range()循环示例

此时，程序将遍历 range(1,11)所指定的序列[1,2,3,4,5,6,7,8,9,10]中的每一个数值，并重复执行循环体中的语句，即 nsum 进行 1+2+……+10 的累加。for…in 循环与 range()函数示例程序运行结果如图 3.42 所示。

```
1+2+……+10的整数和为： 55
```

图 3.42　for…in 循环与 range()函数示例程序运行结果

3.3.5　任务实现

1．绘制流程图

计算 1^2、2^2、3^2……19^2、20^2 的值，并绘制出流程图，如图 3.43 所示。

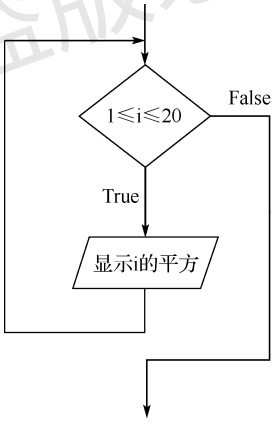

图 3.43　流程图

2．设计程序并调试运行

根据所绘制的流程图可以看到，这是一个非常典型的循环程序，可以利用 for…in 循环与 range()函数来实现。新建程序并输入设计好的代码，如图 3.44 所示。

```
# 循环程序
for i in range(1,21):
    print(i,'的平方是：',i**2)
```

图 3.44　任务 3 程序代码

当程序运行时，将遍历 range(1,21)所指定的序列中的每一个数值，并重复执行循环体中的语句。程序运行结果如图 3.45 所示。

```
1 的平方是： 1
2 的平方是： 4
3 的平方是： 9
4 的平方是： 16
5 的平方是： 25
6 的平方是： 36
7 的平方是： 49
8 的平方是： 64
9 的平方是： 81
10 的平方是： 100
11 的平方是： 121
12 的平方是： 144
13 的平方是： 169
14 的平方是： 196
15 的平方是： 225
16 的平方是： 256
17 的平方是： 289
18 的平方是： 324
19 的平方是： 361
20 的平方是： 400
```

图 3.45　任务 3 程序运行结果

任务拓展

如果陈老师想实现从键盘上输入一个数，输出从 1 开始到该数的平方表的功能，该如何改写这个程序？

项目考核

一、单选题

1．下面不是 Python 的关键字的是（　　）。

A．while　　B．for　　C．elif　　D．this

2．下面能将字符串类型转换为整数类型的函数是（　　）。

A．num()　　B．str()　　C．int()　　D．float()

3．关于 input()函数的使用，下面表述错误的是（　　）。

A．input()函数可以用无参的形式调用

B．如果输入“1”，则 input()函数会返回整数值“1”

C．整数值可以作为 input()函数的参数传入

D．input()函数可以作为 while 语句的条件

4．下面需要使用分支流程的是（　　）。

A．从 0 到 99 求和

B．计算某场考试的班级平均分

C．根据长和宽计算长方形面积

D．判断两个数的大小关系

5．在 Python 循环流程中，可以用于跳过本轮循环的语句是（　　）。

A．continue　　B．pass

C．return　　D．break

6．将 range(5, 0, −1)的返回值转换为列表是（　　）。

A．[5, 4, 3, 2, 1]　　B．[5, 4, 3, 2, 1, 0]

C．[0, 1, 2, 3, 4]　　D．[0, 1, 2, 3, 4, 5]

7．下面不能作为 for…in 循环的遍历对象的是（　　）。

A．'ABCD'　　B．range(10)

C．[1, 2, 3, 4, 5]　　D．2024

8．在下面程序的横线处填入（　　），可以让循环体执行 10 次。

```
i = ________
while i <= 10:
    i+=1
```

A．0　　B．1

C．2　　D．3

9．已知某项目评分为 x（1≤x≤5），现在需要将其分为 A、B、C 三档，并将分档结果保存在变量 y 中，需要在下面程序的横线处填入（　　）。

```
if x < 2:
    y = 'C'
```

```
    ____ x < 4:
        y = 'B'
    else:
        y = 'A'
```

A．if　　B．else

C．elif　　D．else if

10．如果 Python 程序陷入了死循环，则可以按（　　）组合键强行退出。

A．Ctrl+C　　B．Ctrl+A

C．Ctrl+Z　　D．Ctrl+V

二、填空题

1．当 if 后面的表达式值为_____时，不会执行 if 语句块内的代码。

2．在 Python 中，获取控制台输入的函数为_____。

3．range()函数的第 3 个参数名为_____。

4．range()函数的返回值类型为_____。

5．range(2, 5)[−1]的结果是_____。

6．下面程序的运行结果是_____。

```
x = 1
s = 0
while s < 50:
    s += x
    x += 1
print(s)
```

7．下面程序的运行结果是_____。

```
x = 1
s = 0
while s < 50:
    if x%2 == 1:
        s += x
    x += 1
print(s)
```

8．表达式 sum(range(0, 10, 2))的结果是_____。

9．当输入 1 时，input() * 3 的结果是_____。

10．使用_____语句可以跳出循环。

三、编程题

1．编写程序，输入班级总分和班级人数，计算班级的平均分并输出。

2．编写程序，已知优秀评级条件为分数≥90，输入分数，输出是否达到优秀评级的结论。

3．编写程序，利用 Python 判断二次函数 $x^2-5x+6=0$ 是否有解，如果有解，则求出其解。

4．你被困在一个神奇的山洞中，洞口有一扇铁门，上面的字条告诉你开门密码就是 1～9999 中满足正序和逆序都为 7 的倍数的数字总数。编写程序，找到这个密码。

5．编写程序，求解满足 $x-y=xy//10$ 的有序数对(x, y)的个数，其中 x、y 均为小于 100 的自然数。