第 3 章

分支与循环

学习目标

知识目标

✧ 掌握 Python 语言程序的 3 种基本流程控制结构：顺序结构、分支结构（if 语句）、循环结构（while 语句、for 语句）。

✧ 理解 if-elif-else 语句的语法和逻辑结构，学会编写多条件判断程序。

✧ 掌握 while 语句和 for 语句的使用方法，理解循环变量的作用。

✧ 熟悉 break、continue 在循环结构中的应用，避免产生死循环和冗余计算。

✧ 了解嵌套和中止循环的使用场景，并能正确编写多层循环程序。

能力目标

✧ 能够使用 if 语句实现不同条件下的程序逻辑控制。

✧ 能够使用 while 语句处理不确定次数的循环任务，如用户输入验证。

✧ 能够使用 for 语句遍历列表、字符串、字典等可迭代对象。

✧ 能够结合 break、continue 等优化循环结构，提高程序执行效率。

✧ 能够通过条件判断和循环完成数据筛选、统计、查找等任务。

素质目标

✧ 培养结构化编程思维，学会使用代码清晰地表达逻辑关系。

✧ 提升问题拆解能力，能够将实际问题转换为程序流程。

✧ 养成规范编程习惯，能够编写清晰、易读的流程控制程序。

3.1　案例

扫一扫

3.1.1　猜数游戏（一次猜数机会）

本章将对猜数游戏进行逐步分解，先从设置一次猜数机会开始。

编写一个猜数游戏程序，要求随机输入一个 1 ~ 10 范围内的数字，并提供一次猜数机会。

```
1   import random
2   secret = random.randint(1,10)
3   guess = 0
4   print("请你猜一猜从 1 到 10，会是哪个数字？")
5   print("你只有一次机会哦！")
6   guess = eval(input("请输入你猜的数字："))
7   if guess < secret:
8       print ("太小了！！！！！！！！！！")
9   elif guess > secret:
10      print ("太大了！！！！！！！！！！")
11  else:
12      print("恭喜你，猜对了！")
13  print("秘密数字为：" + str(secret))
```


案例说明

- 第 1 行和第 2 行：使用 import 语句导入 random 包，获取一个 1 ~ 10 范围内的随机数。
- 第 7 ~ 12 行：使用 if-elif-else 语句判断猜的数字与随机数的大小关系，从而给出相应的提示。
- 第 13 行：输出随机数。

注意 if-elif-else 语句结构，可以通过缩进来判断语句块的归属。由于第 12 行与第 13 行的缩进不同，因此第 13 行已经跳出了 if-elif-else 语句结构。在 if、elif 和 else 所在行末尾都有一个冒号，这是 Python 语言比较特殊的语法要求。

扫一扫

3.1.2　猜数游戏（多次猜数机会）版本 1

3.1.1 节介绍的案例中实现了一次猜数机会的功能，接下来对程序进行修改，实现多次猜数机会的功能。

```
1   import random
2   secret = random.randint(1,10)
3   guess = 0
4   tries = 0
```

```
5   print ("请你猜一猜从 1 到 10，会是哪个数字？")
6   print("你只有 3 次机会哦！")
7   while tries <3:   # 提供 3 次猜数机会
8       guess = eval(input("请输入你猜的数字："))
9       tries = tries +1
10      if guess < secret:
11          print ("太小了!!!!!!!!!!")
12          continue
13      elif guess > secret:
14          print ("太大了!!!!!!!!!!")
15          continue
16      else:
17          print("恭喜你，猜对了！")
18          break
19
20  if guess != secret:
21      print("很可惜，你猜错了！")
22  print("正确的数字为：" + str(secret))
```

案例说明

本案例通过循环语句来实现多次猜数机会的功能。

- 第 4 行：通过定义变量 tries 来记录用户尝试的次数。每执行一次 while 语句，次数都要加 1，否则 while 语句会进入死循环。
- 第 7 行：使用 while 语句，控制循环次数为 3。
- 第 8～18 行：循环语句块，采用缩进格式表示语句块都属于该 while 语句的循环体。
- 第 20 行：缩进格式发生变化，表示从该行开始不属于 while 语句的循环体。
- 第 12 行、第 15 行：都加入了一个 continue，表示结束本次循环，进入下一次循环，程序可以不用再执行本次循环的其他后续内容，从而提高程序执行效率。但是此处使用 continue 的意义不大，想一想这是为什么？

本案例的目标是用户最多可以有 3 次猜数机会，如果猜对了，那么程序在第 18 行会给出一个退出循环的动作，即 break。利用 break 会中止 while 语句，从而跳出循环。

3.1.3 猜数游戏（多次猜数机会）版本 2

扫一扫

3.1.2 节介绍的案例中使用 while 语句实现了多次猜数机会的功能，本节将使用 for 语句实现相同的功能。

```
1   import random
2   secret = random.randint(1,10)
3   guess = 0
4   print ("请你猜一猜从 1 到 10，会是哪个数字？")
5   print("你只有 3 次机会哦！")
6   for i in range(3):   # 提供 3 次猜数机会
```

```
7       guess = eval(input("请输入你猜的数字："))
8       if guess < secret:
9           print ("太小了!!!!!!!!!! ")
10          continue
11      elif guess > secret:
12          print ("太大了!!!!!!!!!! ")
13          continue
14      else:
15          print("恭喜你，猜对了！")
16          break
17
18  if guess <> secret:
19      print("很可惜，你猜错了！")
20  print("正确的数字为："+str(secret))
```

案例说明

第 6 行：使用 range(3)生成一个 0～2 的列表。由于 for 语句对使用 range(3)生成的列表进行了遍历，因此该循环将执行 3 次。

整个案例已不再需要使用变量 tries 来记录用户尝试的次数。for 语句在执行过程中，将自动对循环次数进行控制。

练一练：使用刚刚学过的方法完成下列练习。

（1）编写一个猜数游戏程序，要求随机输入一个 0～10 范围内的数字，实现一次猜数机会的功能。

（2）编写一个猜数游戏程序，要求随机输入一个−10～10 范围内的数字，实现 5 次猜数机会的功能。

备注：两种方法都要试一试哦！

3.2　知识梳理

扫一扫

3.2.1　常用运算符

在程序中经常会遇到使用各类运算符的情况，Python 语言支持的运算符的种类有很多，包括算术运算符、关系运算符、逻辑运算符、赋值运算符、位运算符、成员运算符、身份运算符等。

1. 算术运算符

Python 语言支持的算术运算符如表 3-1 所示。

表 3-1　Python 语言支持的算术运算符

算术运算符	描　　述
+	加法运算符：两个数相加
–	减法运算符：得到负数或两个数相减
*	乘法运算符：两个数相乘或返回一个被重复若干次的字符串
/	除法运算符：两个数相除
%	取模运算符：返回除法的余数
**	取幂运算符：如 x**y，表示返回 x 的 y 次幂
//	整除运算符：返回商的整数部分

【例 3-1】算术运算符

```
1   x, y = 35, 10
2   s = 0
3
4   print("x = {0}, y = {1}".format(x,y))
5   s = x + y
6   print("s = x + y, s=", s)
7
8   s = x – y
9   print("s = x – y,s=", s)
10
11  s = x * y
12  print("s = x * y, s=", s)
13
14  s = x / y
15  print("s = x / y, s=", s)
16
17  s = x % y
18  print("s = x % y, s=", s)
19
20  a, b = 3, 4
21  print("a = {0}, b = {1}".format(a,b))
22  c = a ** b
23  print("c = a**b, c=", c)
24
25  a, b = 26, 7
26  c = a // b
27  print("c = a//b, c=", c)
28
```

输出结果如下。

```
x = 35, y = 10
s = x + y, s= 45
s = x – y, s= 25
s = x * y, s= 350
s = x / y, s= 3.5
```

```
s = x % y, s= 5
a = 3, b = 4
c = a**b, c= 81
c = a//b, c= 3
```

2. 关系运算符

Python 语言支持的关系运算的返回值都是布尔值，即 True 或 False。Python 语言支持的关系运算符如表 3-2 所示。

表 3-2　Python 语言支持的关系运算符

关系运算符	描　　述
==	等于运算符
!=	不等于运算符
>	大于运算符
<	小于运算符
>=	大于或等于运算符
<=	小于或等于运算符

注意，a!=b 支持 Python 2 和 Python 3；a< >b 支持 Python 2，但不支持 Python 3。

【例 3-2】关系运算符

```
1  x, y = 35, 10
2  print("x = {0}, y = {1}".format(x, y))
3
4  print("x == y, 关系运算结果为:", x == y)
5  print("x != y, 关系运算结果为:", x != y)
6  print("x > y, 关系运算结果为:", x > y)
7  print("x < y, 关系运算结果为:", x < y)
8  print("x >= y, 关系运算结果为:", x >= y)
9  print("x <= y, 关系运算结果为:", x <= y)
```

输出结果如下。

```
x = 35, y = 10
x == y, 关系运算结果为: False
x != y, 关系运算结果为: True
x > y, 关系运算结果为: True
x < y, 关系运算结果为: False
x >= y, 关系运算结果为: True
x <= y, 关系运算结果为: False
```

3. 逻辑运算符

Python 语言支持的逻辑运算的返回值比较特殊，但是可以放在 if 语句中用于判断，若结果非 0 则为 True，否则为 False。Python 语言支持的逻辑运算符如表 3-3 所示。

表 3-3　Python 语言支持的逻辑运算符

逻辑运算符	描　　述	逻辑表达式
and	逻辑与运算符	x and y
or	逻辑或运算符	x or y
not	逻辑非运算符	not x

【例 3-3】逻辑运算符

```
1  x, y = 35, 10
2  print("x = {0}, y = {1}".format(x,y))
3
4  print("x and y, 逻辑运算结果为:", x and y)
5  print("x or y, 逻辑运算结果为:", x or y)
6  print(" not x, 逻辑运算结果为:", not x)
```

输出结果如下。

```
x = 35, y = 10
x and y, 逻辑运算结果为: 10
x or y, 逻辑运算结果为: 35
not x, 逻辑运算结果为: False
```

想一想：执行下面的逻辑表达式，其输出结果是什么？

（1）带圆括号的逻辑表达式：

```
>>> not (True and True)
>>> not (True and (False or True))
>>> not (True or (False or True))
```

（2）不带圆括号的逻辑表达式：

```
>>> not True and False or True
>>> True or False or True or False
>>> False and not False or True
>>> False and False or True
```


4. 赋值运算符

Python 语言支持的赋值运算符如表 3-4 所示。

表 3-4　Python 语言支持的赋值运算符

赋值运算符	描　　述	实　　例
=	简单的赋值运算符	x = y，将 y 赋给 x
+=	加法赋值运算符	x += y，等效于 x = x + y
−=	减法赋值运算符	x −= y，等效于 x = x − y
*=	乘法赋值运算符	x *= y，等效于 x = x * y
/=	除法赋值运算符	x /= y，等效于 x = x / y
%=	取模赋值运算符	x %= y，等效于 x = x % y
**=	幂赋值运算符	x **= y，等效于 x = x ** y
//=	取整赋值运算符	x //= y，等效于 x = x // y

【例 3-4】赋值运算符

```
1   x, y = 10, 2
2   print("x = {0}, y = {1}".format(x, y))
3   x = y
4   print("x = y,    x=", x)
5   x, y = 10, 2
6   x += y
7   print("x += y, x=", x)
8   x, y = 10, 2
9   x -= y
10  print("x -= y, x=", x)
11  x, y = 10, 2
12  x *= y
13  print("x *= y, x=", x)
14  x, y = 10, 2
15  x /= y
16  print("x /= y, x=", x)
17  x, y = 10, 2
18  x %= y
19  print("x %= y, x=", x)
20  x, y = 10, 2
21  x **= y
22  print("x **= y, x=", x)
23  x, y = 10, 2
24  x //= y
25  print("x //= y, x=", x)
```


输出结果如下。

```
x = 10, y = 2
x = y, x= 2
x += y, x= 12
x -= y, x= 8
x *= y, x= 20
x /= y, x= 5.0
x %= y, x= 0
x **= y, x= 100
x //= y, x= 5
```

5. 位运算符

位运算符是把数字看作二进制数来进行计算的。Python 语言支持的位运算符如表 3-5 所示。

表 3-5　Python 语言支持的位运算符

位 运 算 符	描　　述
&	按位与运算符：参与运算的两个值，如果两个相应位都为 1，那么该位的结果为 1，否则为 0
|	按位或运算符：只要对应的两个二进制位有一个为 1，结果就为 1
^	按位异或运算符：当两个对应的二进制位相异时，结果为 1

续表

位 运 算 符	描　　述
~	按位取反运算符：对数据的每个二进制位取反，即把 1 变为 0，把 0 变为 1
<<	左移动运算符：运算数的各二进制位全部左移若干位，由<<右侧的数指定移动的位数，高位丢弃，低位补 0
>>	右移动运算符：把>>左侧的运算数的各二进制位全部右移若干位，由>>右侧的数指定移动的位数

【例 3-5】位运算符

```
1   a = 9    # 9 = 0000 1001
2   b = 3    # 3 = 0000 0011
3   c = 0
4   c = a & b    # 1 = 0000 0001
5   print("a & b =", c)
6
7   c = a | b    # 11 = 0000 1011
8   print("a | b=", c)
9
10  c = a ^ b    # 10 =0000 1010
11  print("a ^ b=", c)
12
13  c = ~a    # −10 = 1111 0110
14  print("~a;=", c)
15
16  c = a << 2    # 36 = 0010 0100
17  print("a << 2=", c)
18
19  c = a >> 2    # 2 = 0000 0010
20  print("a >> 2=", c)
```


输出结果如下。

```
a & b= 1
a | b= 11
a ^ b= 10
~a;= −10
a << 2= 36
a >> 2= 2
```

6. 成员运算符

除了以上运算符，Python 语言还支持成员运算符，如表 3-6 所示。

表 3-6　Python 语言支持的成员运算符

成员运算符	描　　述
in	如果在指定的序列中找到值，那么返回 True，否则返回 False
not in	如果在指定的序列中没有找到值，那么返回 True，否则返回 False

【例 3-6】成员运算符

```
1  a = 6
2  clist = [1, 2, 3, 4, 5]
3  print(a in clist)
4  print(a not in clist)
```

输出结果如下。

```
False
True
```

7. 身份运算符

身份运算符用于比较两个对象的存储单元。Python 语言支持的身份运算符如表 3-7 所示。

表 3-7　Python 语言支持的身份运算符

身份运算符	描　述
is	判断两个标识符是否引用自同一个对象
is not	判断两个标识符是否引用自不同的对象

【例 3-7】身份运算符

```
1  a, b = 6, 10
2  print(a is b)
3  print(a is not b)
4  print(id(a) == id(b) )          # id 方法的返回值就是对象的内存地址
5
6  b = 6
7  print(id(a) == id(b) )          # id 方法的返回值就是对象的内存地址
```

输出结果如下。

```
False
True
False
True
```

其中，id 方法用于获取对象的内存地址。

is 用于判断两个变量的引用对象是否为同一个，==用于判断引用变量的值是否相等。

【例 3-8】is 与==的区别

```
1  a = [1, 2, 3, 4]
2  b = a
3  print(a is b)
4  print(a == b)
5  print(id(a) == id(b))              # id 方法的返回值就是对象的内存地址
```

```
6
7   b = a[:]                    # 通过切片进行浅复制
8   print(a is b)
9   print(a == b)
10  print(id(a) == id(b))       # id 方法的返回值就是对象的内存地址
```

输出结果如下。

```
True
True
True
False
True
False
```

8. 运算符的优先级

表 3-8 按从高到低的顺序列出了运算符的优先级。

表 3-8　运算符的优先级（从高到低）

运　算　符	描　　述
**	指数运算符
~、+、−	按位翻转运算符、一元加法运算符、一元减法运算符
*、 /、 %、//	乘法运算符、除法运算符、取模运算符和整除运算符
+、−	多元加法运算符、多元减法运算符
>>、<<	右移运算符、左移运算符
&	按位与运算符
^ 、|	按位异或运算符、按位或运算符
<=、<、>、>=	小于或等于运算符、小于运算符、大于运算符、大于或等于运算符
==、!=	等于运算符、不等于运算符
=、 %=、 /=、 //=、−=、+=、*=、**=	赋值运算符
is、is not	身份运算符
in、not in	成员运算符
and、or、not	逻辑运算符

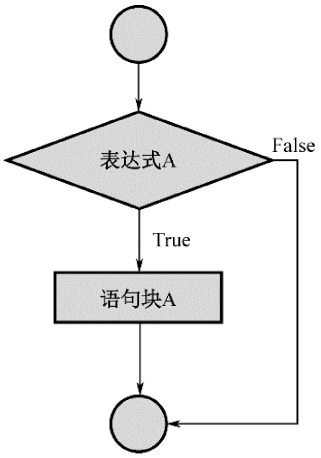
另外，可以使用圆括号处理优先级，圆括号中的数据优先处理，圆括号可以嵌套使用。

3.2.2　if 语句

扫一扫

Python 语言的 if 语句通过一条或多条语句的执行结果（True 或 False）来决定执行的语句块。if 语句有如下 4 种形式。

1. 简单 if 语句

程序流程图	代 码 结 构
	if 表达式 A： 　　若表达式 A 为真，则执行语句块 A

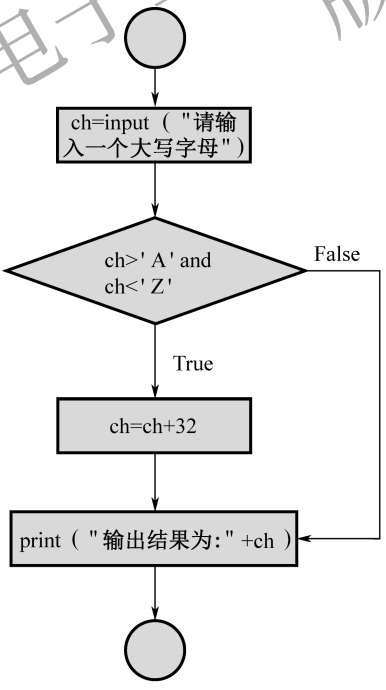
具体说明如下。

在简单 if 语句中，表达式 A 用于确定程序流程。若表达式 A 为真，即表达式 A 的计算结果非 0 或为 True，则执行语句块 A。

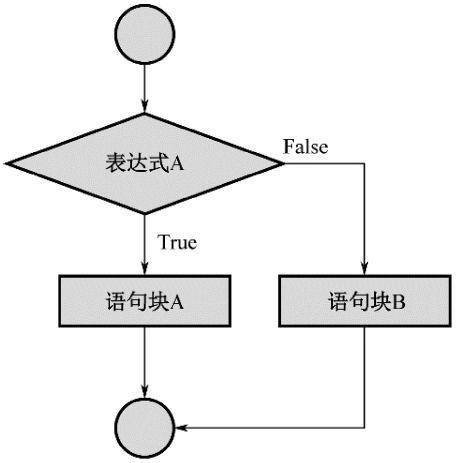
每个条件后都要使用冒号，表示接下来是满足条件后要执行的语句块。

使用缩进来划分语句块，相同缩进的语句一起组成一个语句块。

【例 3-9】输入一个大写字母，并对输入的内容进行转换，最终以小写字母输出

程序流程图	代 码 实 现
	ch = input("请输入一个大写字母") if (ch>'A' and ch < 'Z'): 　　ch = ch + 32 print ("输出结果为：" + ch)

2. if-else 语句

程序流程图	代 码 结 构
	if 表达式 A： 　　若表达式 A 为真，则执行语句块 A else： 　　若表达式 A 为假，则执行语句块 B

具体说明如下。

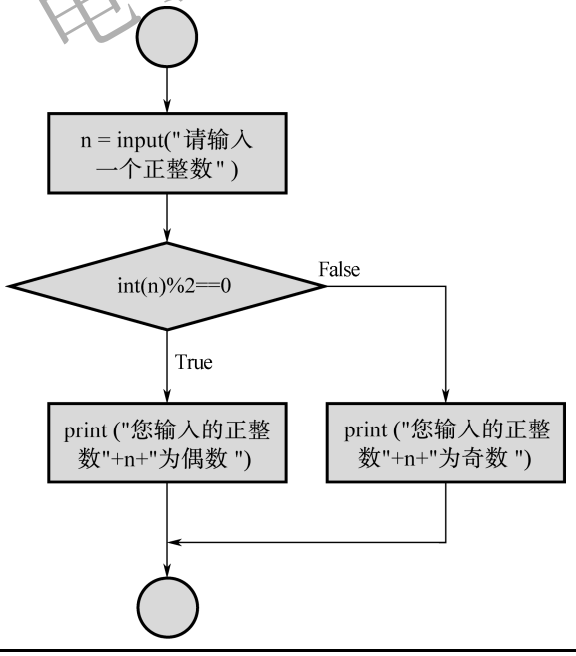
在 if-else 语句中，若表达式 A 为真，则执行语句块 A；否则，执行语句块 B。

其中，else 语句块是可选的，可以根据具体情况决定是否包含它。

Python 语言的一个与众不同的地方，就是使用缩进来标识语句块。在 Python 语言中，多一个或少一个空格都可能出现错误。因此，在同一个语句块中，所有语句的缩进必须相同。

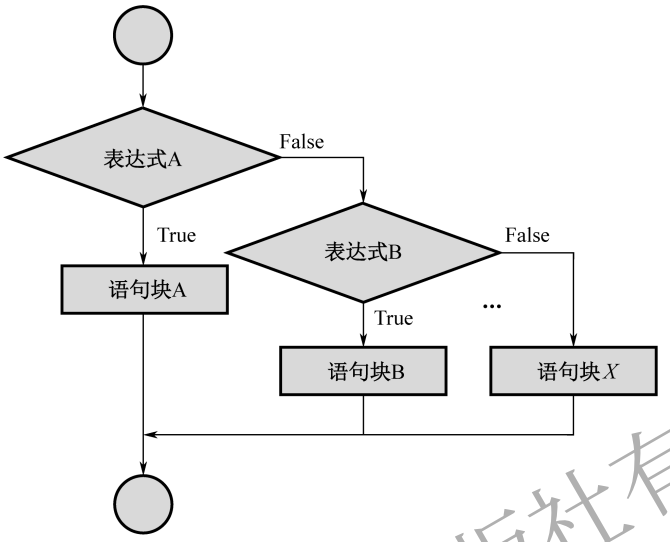
【例 3-10】输入一个正整数，判断该正整数是奇数还是偶数

程序流程图	代 码 实 现

	n = input("请输入一个正整数")
	if int(n)%2==0:
	　　print ("您输入的正整数"+n+"为偶数")
	else:
	　　print ("您输入的正整数"+n+"为奇数")

3. if-elif-else 语句

if-elif-else 语句用于多重分支的选择。

程序流程图	代 码 结 构
	if 表达式 A： 　　若表达式 A 为真，则执行语句块 A elif 表达式 B： 　　若表达式 B 为真，则执行语句块 B … else： 　　若所有条件都不满足，则执行语句块 *X*

具体说明如下。

在 if-elif-else 语句中，若表达式 A 为真，则执行语句块 A；否则，进入下一个分支，计算表达式 B；若表达式 B 为真，则执行语句块 B；以此类推，若所有条件都不满足，则执行语句块 *X*。

【例 3-11】英语成绩判断

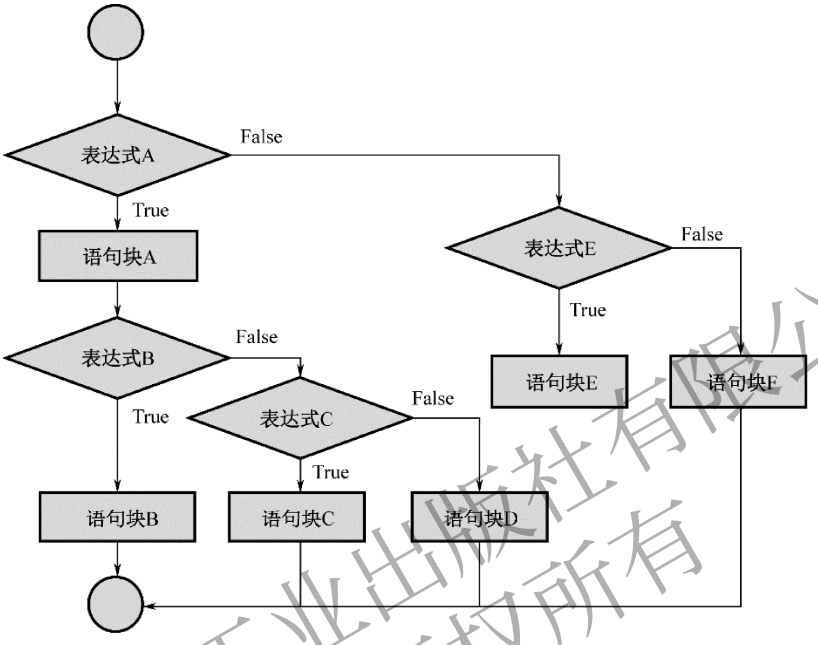
```
1   grade = int(input("请输入你的英语成绩: "))
2   if grade < 0 or grade >100:
3       print("输入错误!")
4   elif grade >= 90 :
5       print("你的英语成绩为优秀！")
6   elif grade >= 80 :
7       print("你的英语成绩为良好！")
8   elif grade >= 70 :
9       print("你的英语成绩为中等！")
10  elif grade >= 60 :
11      print("你的英语成绩为及格！")
12  else：
13      print("你的英语成绩为不及格！")
```

输出结果如下。

```
请输入你的英语成绩: 85
你的英语成绩为良好!
```

4. if 语句的嵌套

用户可以根据实际程序的要求，把一个 if-elif-else 语句放在另一个 if-elif-else 语句中，即 Python 语言支持 if 语句的嵌套。

程序流程图	代 码 结 构
	if 表达式 A： 　执行语句块 A 　if 表达式 B： 　　执行语句块 B 　elif 表达式 C： 　　执行语句块 C 　else： 　　执行语句块 D elif 表达式 E： 　执行语句块 E else： 　执行语句块 F

【例 3-12】输入一个数字，判断它能否同时被 2 和 3 整除

```
1  num=int(input("输入一个数字："))
2  if num%2==0:
3      if num%3==0:
4          print ("你输入的数字可以同时被 2 和 3 整除")
5      else:
6          print ("你输入的数字可以被 2 整除，但不能被 3 整除")
7  else:
8      if num%3==0:
9          print ("你输入的数字可以被 3 整除，但不能被 2 整除")
10     else:
11         print   ("你输入的数字不能同时被 2 和 3 整除")
```

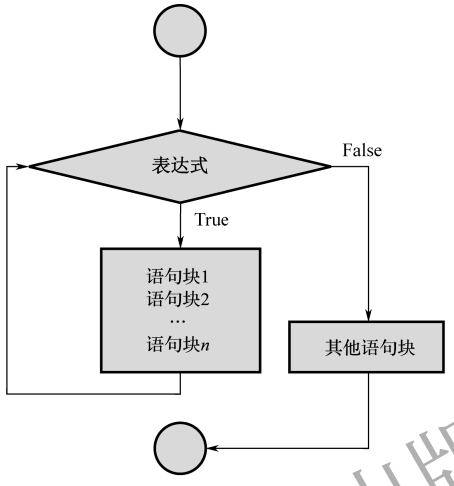
输出结果如下。

```
输入一个数字：12
你输入的数字可以同时被 2 和 3 整除
```

注意，if 语句的嵌套包含多个条件，用于复杂的条件判断。用户可以根据需要使用任意多个 elif 语句块。

扫一扫

3.2.3　while 语句

程序流程图	代 码 结 构
	while 表达式： 　　# 若表达式为真，则执行语句块 1～语句块 *n* 　　语句块 1 　　语句块 2 　　… 　　语句块 *n* 其他语句块

具体说明如下。

- 在 while 语句中，若表达式为真，即表达式的计算结果非 0 或为 True，则执行语句块（1 ~ *n*）。
- 计算表达式并根据计算结果决定是否再次执行语句块（1 ~ *n*）。
- 如此往复循环，直至表达式的计算结果为 0 或 False。
- 注意表达式后的冒号和语句块（1 ~ *n*）的缩进格式。
- 若循环体在执行过程中出现 break，则循环中止；若循环是嵌套的，则出现 break 只中止所在层的循环。
- Python 语言不支持 do-while 语句。

下面分别演示 while 语句的 5 种使用方式。

1. 循环次数确定，使用循环变量

【例 3-13】计算 1 ~ 100 范围内整数的总和

```
1    sum = 0
2    i= 1
3    while i <= 100:
4        sum = sum + i
5        i += 1
6    print("1 ~ 100 范围内整数的总和为：%d" % (sum))
```

输出结果如下。

```
1 ~ 100 范围内整数的总和为：5050
```

2. 循环次数不确定，直至表达式的计算结果为 0 或 False

【例 3-14】输入一个整数字，求它的所有因子

```
1  j = 2
2  i = eval(input("请输入一个整数："))
3  answer = "它的所有因子为："
4  while i > j:
5      if   i % j == 0:
6          answer += str(j) + ","
7      j += 1
8  print(outStr)
```

输出结果如下。

```
请输入一个整数：12
它的所有因子为：2,3,4,6,
```

【例 3-14】中输入变量 i 的值以后，可求变量 i 的所有因子。其特点是循环开始时并不知道循环次数，一切由条件决定。

【例 3-15】使用 while 语句在屏幕中输出整数 0～9

```
1  i = 0
2  while i <10:
3      print (i)
4  i = i + 1   #   变量 i 的值不断递增，从而确保循环能够结束
```

3. 通过设置表达式的计算结果永远不为 False 来实现无限循环

【例 3-16】无限循环

```
1  while 1 == 1 :   # 表达式永远成立
2      num = int(input("输入一个数字："))
3      print ("你输入的数字是：", num)
4  print ("再见!")
```

单击 PyCharm 中的“停止”按钮可以退出当前的无限循环。无限循环在服务器上客户端的实时请求中比较常用，但在平时的程序编写中要慎重使用。

4. while-else 语句

在 while-else 语句中，当返回值为 False 时，可执行 else 语句块。

【例 3-17】while-else 语句

```
1  count = 0
2  while count < 5:
3      print (count, " 小于  5")
4      count = count + 1
```

```
5    else:
6        print (count, "  大于或等于  5")
```

输出结果如下。

```
0   小于  5
1   小于  5
2   小于  5
3   小于  5
4   小于  5
5   大于或等于  5
```

5. 简单语句组

如果 while 语句的循环体中只有一条语句，那么可以将该语句与 while 语句的循环体写在同一行。

【例 3-18】简单语句组

```
    while 1==1: print ('Hello DerisWeng!')
```

输出结果中将出现无数遍的 Hello DerisWeng!。

扫一扫

3.2.4　嵌套和中止循环

循环可以嵌套，并可使用 break 中止。

【例 3-19】求 20 以内的所有质数

```
1   i = 2  # 初始化变量 i，从 2 开始（最小的质数是 2）
2   while i < 21:
3       # 初始化变量 j，从 2 开始（最小的质数是 2）
4       j = 2
5       is_prime = True
6       # 内层循环：检查变量 i 能否被 2 ~ sqrt(i) 范围内的任何数整除
7       # 使用 j * j <= i 代替 j <= sqrt(i)，避免进行浮点数运算
8       while j * j <= i:
9           # 如果变量 i 能被变量 j 整除，那么变量 i 不是质数
10          if i % j == 0:
11              is_prime = False  # 标记变量 i 不是质数
12              break  # 跳出内层循环，无须继续检查其他因子
13          j += 1
14      # 如果内层循环完整执行（没有找到能整除变量 i 的因子），那么变量 i 是质数
15      if is_prime:
16          print(i, '是质数')  # 输出结果
17      # 检查下一个数字
18      i += 1
```

输出结果如下。

```
2 是质数
3 是质数
5 是质数
7 是质数
11 是质数
13 是质数
17 是质数
19 是质数
```

需要注意以下程序。

```
if j >= i/2:
    print (i, '是质数')
```

它们是对 while j<i/2:中条件的再次利用，若循环正常执行，则变量 j 的值必然大于或等于 i/2；若在循环过程中执行了 break，则变量 j 的值将小于 i/2。

扫一扫

3.2.5 for 语句

在访问列表的过程中使用了一种新的循环语句，即 for 语句。在 Python 语言中，for 语句经常用来遍历一个集合中的所有元素，主要有以下两种形式。

1. 形式 1

```
for rec in My_list
  循环体 LB
```

其中，My_list 是一个列表，按元素索引顺序将元素依次赋给 rec。如果 My_list 中有多个元素，那么循环体 LB 将执行多次，同时，可以使用 rec 在循环体中遍历 My_list。

2. 形式 2

```
for i in range(N)
  循环体 LB
```

这种形式的循环是一个变体，因为 range(N)的意义在于生成一个 0 ~ N−1 的整数列表。也可以通过指定 range 函数的范围生成整数列表。

for i in range(N, M)的范围是从 N 到 M−1，其中 i 的范围是[N,M−1]。

range(a, b, c)的含义是从 a 开始，步长为 c，结束值小于 b。

【例 3-20】使用 for 语句在屏幕中输出整数 0 ~ 9

```
1  for i in range(10):          # 默认情况下，初始值为 0
2      print (i)
```

【例 3-21】在屏幕中输出整数 1 ~ 9

```
1  for i in range(1,10):        # 从 1 开始，输出到 9
2      print (i)
```

【例 3-22】在屏幕中输出整数 1 ~ 10

```
1  for i in range(1,11):    # 从 1 开始，输出到 10
2      print (i)
```

或者

```
1  for i in range(10):
2      print (i+1)
```

【例 3-23】在屏幕中按相反的顺序输出整数

```
1  for i in range(10,0,-1):
2      print (i)
```

在 range(10,0,−1)中，第 1 个参数 10 和第 2 个参数 0 指明了整数的范围，第 3 个参数−1 表示步长。

当然，还有另一种方法。

```
1  for i in range(10):
2      print (10-i)
```

注意，如果使用 for 语句进行循环，那么需要使用 range 函数构造一个列表。从这个角度来看，使用 for 语句与 range 函数进行单纯的循环的效率相比使用 while 语句进行循环的效率要低。

为了解决这个问题，对单纯的循环可以使用 xrange 函数（xrange 函数在 Python 3 中已不再使用）。例如：

```
for i in xrange(1,100)
```

xrange(1,100)仅产生一个 1 ~ 100 的循环范围，并不产生列表，这样就提高了效率。若只需要产生循环范围，则建议使用 for 语句与 xrange 函数进行循环。

3.3　小结与习题

3.3.1　小结

本章案例通过 if 语句的条件分支和 while 语句分别实现了猜数游戏的一次猜数机会和多次猜数机会的功能。另外，本章介绍了在程序中经常会遇到使用各类运算符的情况，Python 语言支持的运算符的种类有很多，包括算术运算符、关系运算符、逻辑运算符、赋值运算符、位运算符、成员运算符、身份运算符等。在流程控制方面，本章还介绍了 if 语句、while 语句、嵌套和中止循环，以及 for 语句等相关知识。

通过对本章的学习，读者应学会 Python 语言中 if 语句、while 语句和 for 语句的使用方法，不但能够使用 if 语句确定程序流程，还能够使用 while 语句进行循环和使用 for 语句遍历集合。同时，通过实训，读者将能够熟练使用 Python 语言的常用运算符，以及使用流程控制语句解决实际编程问题。

3.3.2 习题

（1）编写一个程序，要求输出一个九九乘法表，格式为左对齐。

（2）填写下面的表格。

m	n	m == n	m != n	m and n	m or n	not m
False	False					
False	True					
True	False					
True	True					

（3）编写一个程序，要求输入三角形的 3 条边，并判断这个三角形是否为等腰三角形。

（4）编写一个程序，要求输入两个整数，并将这两个整数按从小到大的顺序输出。

3.4 课外拓展

下面介绍 PyCharm 常用的快捷键。

1. 编辑（Editing）快捷键

编辑（Editing）快捷键如表 3-9 所示。

表 3-9 编辑（Editing）快捷键

快 捷 键	功 能 说 明	快 捷 键	功 能 说 明
Ctrl + Space	完成基本代码	Ctrl + Shift + W	回到之前的状态
Ctrl + Alt + Space	快速导入任意类	Ctrl + Shift +] / [	选定语句块结束位置/起始位置
Ctrl + Shift + Enter	完成语句	Alt + Enter	快速修正
Ctrl + P	查看参数（在方法中调用参数）	Ctrl + Alt + L	格式化代码
Ctrl + Q	快速查看文件	Ctrl + Alt + O	优化导入
Shift + F1	打开外部文件，进入文件主页	Ctrl + Alt + I	自动缩进
Ctrl + F1	显示错误描述	Tab / Shift + Tab	缩进当前行或不缩进当前行
Alt + Insert	自动生成代码	Ctrl+X / Shift+Delete	剪切当前行或选定语句块到剪贴板中
Ctrl + O	重载方法	Ctrl+C / Ctrl+Insert	复制当前行或选定语句块到剪贴板中
Ctrl + Alt + T	选中	Ctrl+V / Shift+Insert	从剪贴板中粘贴
Ctrl + /	行注释或取消行注释	Ctrl + Shift + V	从最近的缓冲区粘贴
Ctrl + Shift + /	块注释	Ctrl + D	复制选定的区域或行

续表

快　捷　键	功 能 说 明	快　捷　键	功 能 说 明
Ctrl + W	选定增加的语句块	Ctrl + Backspace	删除整个字符串
Ctrl + Y	删除选定的行	Ctrl+ Numpad+ / –	展开语句块或折叠语句块（当前位置为函数、注释等）
Ctrl + Shift + J	添加智能线	Ctrl + Shift + Numpad+ / –	展开所有语句块或折叠所有语句块
Ctrl + Enter	切割智能线	Ctrl + F4	关闭执行的选项卡
Shift + Enter	另起一行	Ctrl + Delete	删除到字符结束
Ctrl + Shift + U	在选定的区域或语句块间切换		

2. 查找/替换（Search/Replace）快捷键

查找/替换（Search/Replace）快捷键如表 3-10 所示。

表 3-10　查找/替换（Search/Replace）快捷键

快　捷　键	功 能 说 明	快　捷　键	功 能 说 明
F3	查找下一个	Shift + F3	查找上一个
Ctrl + R	替换	Ctrl + Shift + F	连续两次按 Shift 键进行全局查找（可以在整个项目中查找某字符串，如查找某函数名字符串）
Ctrl + Shift + R	全局替换		

3. 执行（Running）快捷键

执行（Running）快捷键如表 3-11 所示。

表 3-11　执行（Running）快捷键

快　捷　键	功 能 说 明	快　捷　键	功 能 说 明
Alt + Shift + F10	配置执行模式	Alt + Shift + F9	配置调试模式
Shift + F10	执行	Shift + F9	调试
Ctrl + Shift + F10	配置执行编辑器	Ctrl + Alt + R	执行 manage.py 任务

4. 调试（Debugging）快捷键

调试（Debugging）快捷键如表 3-12 所示。

表 3-12　调试（Debugging）快捷键

快　捷　键	功 能 说 明	快　捷　键	功 能 说 明
F8	跳过	F7	进入
Shift + F8	退出	Alt + F9	执行游标
Alt + F8	验证表达式	Ctrl + Alt + F8	快速验证表达式
F9	恢复程序	Ctrl + F8	断点开关
Ctrl + Shift + F8	查看断点		

5. 搜索相关（Usage Search）快捷键

搜索相关（Usage Search）快捷键如表 3-13 所示。

表 3-13　搜索相关（Usage Search）快捷键

快　捷　键	功 能 说 明	快　捷　键	功 能 说 明
Alt / Ctrl + F7	在文件中查找用法	Ctrl + Shift + F7	在文件中高亮显示用法
Ctrl + Alt + F7	显示用法		

6. 导航（Navigation）快捷键

导航（Navigation）快捷键如表 3-14 所示。

表 3-14　导航（Navigation）快捷键

快　捷　键	功 能 说 明	快　捷　键	功 能 说 明
Ctrl + N	跳转到类	Ctrl + Shift + N	跳转到符号
Alt + Right / Left	跳转到下一个编辑的选项卡或跳转到上一个编辑的选项卡	F12	回到先前的工具窗口
Esc	从工具窗口回到编辑窗口	Shift + Esc	隐藏执行的、最近执行的窗口
Ctrl + Shift + F4	关闭主动执行的选项卡	Ctrl + G	查看当前行号、字符号
Ctrl + E	打开最近使用的文件列表	Ctrl+Alt+Left / Right	后退或前进
Ctrl+Shift+Backspace	导航到最近的编辑区域	Alt + F1	查找当前文件或标识
Ctrl+B / Click	跳转到声明	Ctrl + Alt + B	跳转到代码实现位置
Ctrl + Shift + I	查看快速定义	Ctrl + Shift + B	跳转到类型声明位置
Ctrl + U	跳转到父方法、父类	Alt + Up / Down	跳转到上一个方法或跳转到下一个方法
Ctrl +] / [	跳转到语句块结束处或跳转到语句块开始处	Ctrl + F12	弹出文件结构
Ctrl + H	显示类型层次结构	Ctrl + Shift + H	显示方法层次结构
Ctrl + Alt + H	调用层次结构	F2 / Shift + F2	显示下一条高亮错误或显示上一条高亮错误
F4 / Ctrl + Enter	编辑资源或查看资源	Alt + Home	显示 F11 的书签开关
Ctrl + Shift + F11	书签助记开关	Ctrl + #[0-9]	跳转到标识的书签
Shift + F11	显示书签		

7. 重构（Refactoring）快捷键

重构（Refactoring）快捷键如表 3-15 所示。

表 3-15　重构（Refactoring）快捷键

快　捷　键	功 能 说 明	快　捷　键	功 能 说 明
F5 / F6	复制或剪贴	Alt + Delete	安全删除
Shift + F6	重命名	Ctrl + F6	更改签名
Ctrl + Alt + N	内联	Ctrl + Alt + M	提取方法
Ctrl + Alt + V	提取属性	Ctrl + Alt + F	提取字段
Ctrl + Alt + C	提取常量	Ctrl + Alt + P	提取参数

8. 控制（VCS/Local History）快捷键

控制（VCS/Local History）快捷键如表 3-16 所示。

表 3-16　控制（VCS / Local History）快捷键

快　捷　键	功 能 说 明	快　捷　键	功 能 说 明
Ctrl + K	提交项目	Ctrl + T	更新项目
Alt + Shift + C	查看最近的变化	Alt + BackQuote(')	快速弹出 VCS

9. 模板（Live Templates）快捷键

模板（Live Templates）快捷键如表 3-17 所示。

表 3-17　模板（Live Templates）快捷键

快　捷　键	功 能 说 明	快　捷　键	功 能 说 明
Ctrl + Alt + J	为当前行使用模板	Ctrl +J	插入模板

10. 基本（General）快捷键

基本（General）快捷键如表 3-18 所示。

表 3-18　基本（General）快捷键

快　捷　键	功 能 说 明	快　捷　键	功 能 说 明
Alt + #[0-9]	打开相应的工具窗口	Ctrl + Alt + Y	同步
Ctrl + Shift + F12	最大化编辑开关	Alt + Shift + F	为搜索对话框添加文本类型过滤器
Alt + Shift + I	根据配置检查当前文件	Ctrl + BackQuote(')	快速切换当前计划
Ctrl + Alt + S	打开设置窗口	Ctrl + Shift + A	查找所有动作
Ctrl + Tab	在窗口间进行切换		

素养拓展要点：

（1）安装 Anaconda，并列举其中 Jupyter Notebook 的快捷键。

（2）比较 PyCharm 与 Jupyter Notebook 的快捷键的相同点和不同点。

3.5　实训

3.5.1　分支

一、实训目的

（1）熟练使用 Python 语言的常用运算符。

（2）掌握 if 语句的使用方法。

二、练习

1. 选择题

（1）当变量 x 为大于 1 的奇数时，计算结果为 0 的表达式是（　　）。

A. x%2==1　　B. x/2　　C. x%2!=0　　D. x%2==0

（2）在嵌套使用 if 语句时，Python 语言规定 else 总是（　　）。

A. 和之前与其具有相同缩进的 if 相匹配

B. 和之前与其最近的 if 相匹配

C. 和之前的首个 if 相匹配

D. 和之前与其最近且不带 else 的 if 相匹配

（3）下列语句中正确的是（　　）。

A. min = x if x < y else y　　B. max = x > y ? x : y

C. if (x > y) print x　　D. if 1>2: print("hello")

2. 填空题

（1）假设 a=3，b=4，c=5，写出下列表达式的计算结果。

表 达 式	计 算 结 果
a+b>c and b==c	
not(a>b) and not c or 1	
a<c and c<b	
a<c<b	
a<b or c<b	

（2）在算术运算符、关系运算符、逻辑运算符、赋值运算符、_____、_____、_____等中，优先级最高的运算符是_____，优先级最低的运算符是_____。

（3）判断一个字符是数字的表达式为_____。

（4）判断一个字符是字母的表达式为_____。

（5）在 Python 语言中，使用_____表示逻辑真，使用_____表示逻辑假。

三、实训任务

任务 1：猜数游戏

编写一个程序，要求随机输入一个 0 ~ 10 范围内的整数，实现一次猜数机会的功能。

程序编写于下方

任务 2：学生成绩等级评定

根据学生考试成绩评定成绩等级，成绩与等级的对应关系如下。

成　　绩	等　　级
score>= 90	A
80<=score<90	B

续表

成　　绩	等　　级
70<=score<80	C
60<=score<70	D
score<60	E

编写一个程序，要求实现学生成绩等级的评定。

程序编写于下方

任务 3：输入字符判断

编写一个程序，要求输入一个字符，判断该字符是数字、字母、空格还是其他。

程序编写于下方

任务 4：身体质量指数判断

扫一扫

编写一个程序，判断身体质量指数。身体质量指数（BMI）是指用体重（单位：kg）除以身高（单位：m）的平方得出的数字，是目前国际上常用的衡量人体胖瘦程度，以及健康与否的标准，具体内容如下。

身体质量指数	< 18.5	18.5 ~ 24.9	25.0 ~ 29.9	>29.9
身 体 情 况	消瘦	正常	超重	肥胖

程序编写于下方

任务 5：企业发放奖金判断

编写一个程序，要求输入当月利润，求应发放的奖金总数。对于企业发放的奖金，根据

利润多少计算。当利润低于或等于 10 万元时，奖金可提 10%；当利润高于 10 万元、低于 20 万元时，低于 10 万元的部分按 10%提成，高于 10 万元的部分按 7.5%提成；当利润高于 20 万元、低于或等于 40 万元时，高于 20 万元的部分按 5%提成；当利润高于 40 万元、低于或等于 60 万元时，高于 40 万元的部分按 3%提成；当利润高于 60 万元、低于或等于 100 万元时，高于 60 万元的部分按 1.5%提成；当利润高于 100 万元时，高于 100 万元的部分按 1%提成。

程序编写于下方

任务 6：月份判断

使用 if 语句编写一个程序，判断输入的月份应该有多少天（2 月定为 28 天）。

程序编写于下方

四、拓展任务

任务 1：验证码大小写转换

一般在登录网站时都会输入验证码。在输入验证码时，无论用户输入的是大写字母还是小写字母，验证时都会忽略大小写的差异，认为是相同的字符。这说明系统已经对验证码中的字符和用户输入的字符进行了大小写转换。那么这种转换是如何实现的呢？编写一个程序，要求输入一组字符，无论该字符是大写字母还是小写字母，都将其转换为小写字母，并输出结果。

程序编写于下方

任务 2：产品促销

某商场采用购物打折的方式进行促销，具体促销方式如下。

购 买 金 额	折　　扣
1000 元及以上	九折
2000 元及以上	八折
3000 元及以上	七折

编写一个程序，要求当输入客户实际购买金额后，计算优惠价，并输出结果。

程序编写于下方

任务 3：闰年判断

编写一个程序，要求输入一个年份，计算它是否为闰年，并输出结果。闰年的条件是：能被 4 整除但不能被 100 整除或能被 400 整除。

程序编写于下方

任务 4：月份判断

使用 if 语句编写一个程序，判断输入的月份应该有多少天（2 月根据是否为闰年来判断是 28 天还是 29 天）。

程序编写于下方

3.5.2 循环

一、实训目的

（1）掌握 while 语句的使用方法。
（2）掌握 for 语句的使用方法。
（3）能够使用流程控制语句解决实际编程问题。

二、练习

（1）循环无休止地进行下去的状态被称为_____。
（2）使用下列 while 语句输出 1 2 3 4 5 6 8 9 10，请将下列 while 语句补充完整。

```
count=1
_____ count <= 10:
    _____ count != 7:
        _____(count)
    count+=1
```

（3）循环可以嵌套_____层。

三、实训任务

任务 1：猜数游戏

编写一个程序，要求随机输入一个 0~100 范围内的整数，实现 6 次猜数机会的功能。

程序编写于下方

任务 2：字符个数统计

编写一个程序，要求输入一行字符，统计其中的英文字母、空格、数字及其他字符的个数，并输出结果。

程序编写于下方

任务 3：输出水仙花数

扫一扫

编写一个程序，要求输出所有的“水仙花数”。“水仙花数”是指一个三位数，其各位数字的立方和等于该三位数本身。例如，153 = $1^3+5^3+3^3$，该数即“水仙花数”。

程序编写于下方

任务 4：用数字组数

编写一个程序，要求使用 1、2、3、4 来组数，并输出结果。

程序编写于下方

任务 5：评委评分统计

已知某比赛有 7 个评委，选手的得分为这 7 个评委评分的总和。编写一个程序，要求分别使用 while 语句和 for 语句实现统计功能。

程序编写于下方
使用 while 语句：
使用 for 语句：

任务 6：break 和 continue 的使用

编写一个程序，要求输入若干字符，对输入的英文字母原样输出，对其他字符不输出，直

到按 Enter 键后结束。

程序编写于下方

四、拓展任务

任务 1：韩信点兵

淮安民间流传着一则故事——韩信点兵。话说韩信带领 1500 个士兵打仗，战死四五百人，于是韩信要求士兵排队。通过 3 个人一排，多出 2 个人；5 个人一排，多出 4 个人；7 个人一排，多出 6 个人的情况，韩信很快说出士兵为 1049 个人。

已知 3 个队伍的多出人数，分别为非负整数 a、b 和 c。计算军队的总人数。编写一个程序，要求实现上述功能。

程序编写于下方

任务 2：数数游戏

已知有 n 个人围成一圈，顺序排号。从第 1 个人开始报数，凡报到 5 的人退出圈子，计算最后留下的人是原来的第几号。编写一个程序，要求实现上述功能。

程序编写于下方