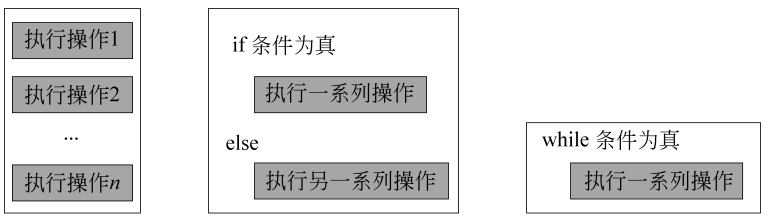
第3章

流 程 控 制

前面介绍的程序都是一条一条语句顺序执行的。程序中的每一条语句都执行一次，而且只能执行一次，这些语句的组织方式称为顺序结构。只使用顺序结构不能解决所有问题，为满足特定情况的需要，常常需要改变程序的顺序流程。计算机科学家为结构化程序定义了三种流程结构：顺序结构、选择结构和循环结构（如图 3.1 所示）。



(a) 顺序　　(b) 选择　　(c) 循环

图 3.1　三种结构

使用这三种结构设计的程序容易理解、调试和修改。本章介绍选择结构和循环结构的实现方法。

3.1　选 择 结 构

计算机之所以有广泛的应用，在于它不仅能简单和按顺序地完成人们事先安排好的一些指令，更重要的是具有逻辑判断能力，能针对不同情况执行不同指令序列，允许程序“选择”适当的动作过程。选择结构根据不同的条件来选择不同的操作，需要使用 if 语句实现。

3.1.1　语句块

语句块不是一种语句，而是一组语句。如果实现选择结构或循环结构，当条件为真时执行或者执行多次的系列操作都用语句块来实现。

1. 创建语句块

在语句前放置空格或 Tab 键缩进语句，即可创建语句块。同一个语句块其所有语句缩进应相同。Python 推荐的缩进方式是使用空格进行的，一个语句块缩进 4 个空格。

2. 语句块的开始与结束

在 Python 中，使用冒号(:)标识语句块的开始，块中的每一条语句都是缩进的(缩进量相同)。在恢复成和原来的块一样缩进量时，表示当前块结束。

在 Java、C、C++ 等高级语言中，使用特殊的字符({)表示一个语句块的开始，用另一个字符(})表示语句块的结束。

语句块的开始与结束方式如图 3.2 所示。

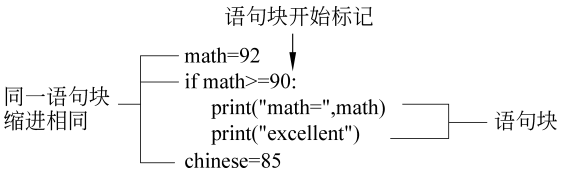


图 3.2　语句块的开始与结束

3.1.2　简单 if 语句

简单 if 语句实现的是“如果……那么……”的执行流程。

例 3.1　输入一个数，输出其绝对值。

输入的数可能是正数，也可能是负数。如果是负数，那么绝对值为其相反数。

```
x=int(input("please input a number:"))
if x<0:
  x=-x
print(x)
```

执行上面的程序，假设输入负数－23，则输出其绝对值 23。如果输入正数 23，则输出正数自身。

1. 简单 if 语句格式

```
if 条件:
  语句块
```

if 关键字后面跟的条件是一个表达式。执行 if 语句时，首先对条件表达式求值，如果结果为真，则执行语句块，否则什么也不做。在例 3.1 中，如果输入的值为负数，x＜0 的值为 True，那么执行语句块 x=-x，对负数求相反数，得到其绝对值。如果输入的是正数，x＜0 的值为 False，不执行语句块。简单 if 语句的流程如图 3.3 所示。

2. 条件表达式

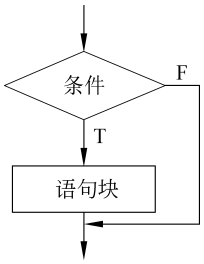


图 3.3　简单 if 语句流程图

if 语句条件表达式的值通常为布尔类型，也称为布尔表达式。条件可以是单条件，也可以是多条件。

常用的单条件包括如下。

- 数字比较，检查两个数字相等(==)、不等(!=)、大于(>)、小于(<)、大于或等于(>=)和小于或等于(<=)。
- 字符串比较，检查两个字符串相等(==)和不等(!=)。

当存在多个条件时，需要根据条件之间的逻辑关系，用 and 或 or 对各个条件进行逻辑运算。

例 3.2　输入一个数，如果是偶数，输出提示。

```
x=int(input("请输入一个数:"))
print("你输入的是",x)
if x%2==0:
   print(x,"是偶数")
```

例 3.3　小明今年 12 岁，再过 10 年，小明还是不是青少年？

```
x=12
if x+10<13 or x+10>19:
  print("10 年后小明不是青少年")
if x+10>=13 and x+10<=19:
  print("10 年后小明是青少年")
```

假设青少年定义为 13～19 岁。是青少年的条件可以描述为大于或等于 13 岁并且小于或等于 19 岁；不是青少年的条件可以描述为小于 13 岁或者大于 19 岁(如图 3.4 所示)。

图 3.4　多条件之并且及或者关系

3. 非布尔类型返回值的条件表达式

如果条件表达式的类型是数值类型、字符串或其他的对象类型，Python 会根据取值情况认定 True 或 False。

认定是 False 的包括布尔型 False、整数 0、浮点数 0.0、null 类型 None、空字符串''、空列表[]、空元组()、空字典{}等，也就是将数值 0 和各种空对象类型认定为 False。

其他情况认定为 True。

```
x=1
if x:
    print("x=1")
```

上面的程序的运行结果是输出 x=1。

3.1.3 if-else 语句

if-else 语句实现的是“如果……那么……否则……”的执行流程，在条件为真时执行一个操作并在条件为假时执行另一个操作。if-else 语句类似于简单的 if 语句，但其中的 else 语句能够指定条件为假时要执行的操作。

例 3.4 用 if-else 语句实现例 3.3。

```
x=12
if x+10>=13 and x+10<=19:
  print("10 年后小明是青少年")
else:
  print("10 年后小明不是青少年")
```

1. if-else 语句格式

```
if 条件:
  语句块 1
else:
  语句块 2
```

if 后面跟的条件是布尔表达式。执行 if-else 语句时，首先对条件求值，如果结果为真，则执行语句块 1，否则执行语句块 2。if-else 语句的流程图如图 3.5 所示。

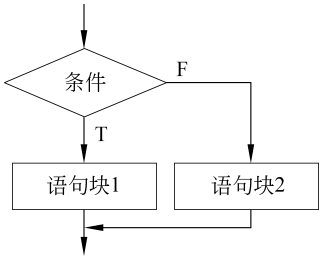


图 3.5 if-else 语句流程图

2. if 语句嵌套

if-else 实现了两个分支的流程处理，如果处理多分支情况，则可以在 if 语句块中包含 if 语句，即 if 语句实现嵌套。

例 3.5 用 if-else 语句实现符号函数。

$$y=\begin{cases}1 & (x>0)\\ 0 & (x=0)\\ -1 & (x<0)\end{cases}$$

符号函数取值包括三种情况，可以分解为 x＞0 和 x≤0 两种情况，x≤0 又可分解为 x＜0 和 x＝0 两种情况（如图 3.6 所示）。

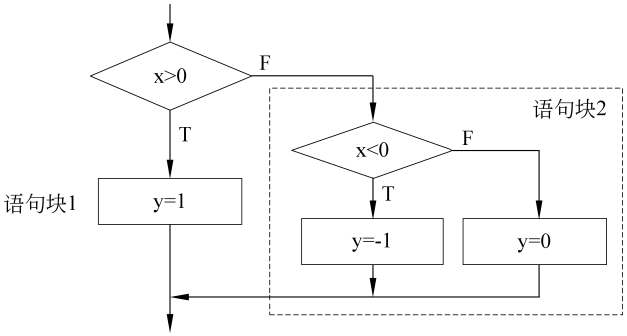


图 3.6　嵌套流程

```
x=0.5
if x>0:
  y=1
else:
  if x<0:
    y=-1
  else:
    y=0
print("y=",y)
```

嵌套的 if 语句也可以写在 if 部分。

3.1.4　if-elif-else 语句

使用嵌套 if 语句可以实现多分支的选择，在分支较多时非常烦琐，Python 提供了一种将 else if 合并的简写方法，格式如下。

```
if 条件 1:
  语句块 1
elif 条件 2:
  语句块 2
…
elif 条件 n:
  语句块 n
else:
  语句块 n+1
```

例 3.6　将百分制成绩转换为五级制。

```
score=int(input("please input a score:"))
if score>=90:
  grade="A"
elif score>=80:
```

```
    grade="B"
elif score>=70:
    grade="C"
elif score>=60:
    grade="D"
else:
    grade="E"
print("grade=",grade)
```

程序运行时，首先测试 score＞＝90 是否为真。若为真，则该成绩为 A，结束 if 语句，否则测试 scroe＞＝80……如果最终进入了 else 的语句块，那么表明 score＜60，成绩为 E，最后输出五级制成绩。程序实现了 5 个分支，将 0～100 划分成[90，100]、[80，90)、[70，80)、[60，70)和[0，60)。

3.2 循 环 结 构

在很多问题中，往往需要有规律地重复某些操作，这些操作在计算机程序中体现为某些语句的重复执行，这就是循环。通过使用循环结构，只要写很少的语句，计算机就会反复执行，完成大量同类运算。

3.2.1 while 语句

Python 中最简单的循环语句是 while，实现“当……时则重复执行……”的流程。

例 3.7 输出 1～100 的所有数。

```
count=1
while count <=100:
   print(count)
   count=count+1
```

执行 while 语句时，首先计算表达式 count＜＝100，如果为 True，则执行下面的语句块，输出 count 的值并将 count 加 1。语句块执行完成后，返回到 while 语句开始位置，重新计算表达式，再次决定是否执行语句块，直到条件为 False，退出循环。

1. while 语句格式及执行流程

```
while 条件:
    语句块
```

while 语句的条件与 if 语句一样，可以是任意类型的表达式，通常是结果为布尔类型的关系表达式或逻辑表达式。

执行 while 语句时，首先计算表达式的值，如果表达式的值为 True，则执行循环体语

句块。然后重新计算表达式的值，再次判断值是否为 True，如果为 True，再执行循环体语句块，如此循环往复；如果表达式的值为 False，则退出循环。

while 语句执行流程如图 3.7 所示。

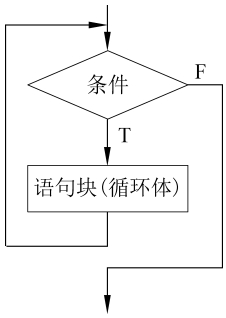


图 3.7　while 语句执行流程

例 3.8　计算 1～100 的所有数之和。

```
sum=0
i=1
while i<=100:
    sum=sum+i
    i=i+1
print(sum)
```

程序利用变量 sum 保存求和结果，加数保存在变量 i 中。第一个数为 1，在变量 i 小于等于 100 时，while 语句条件为真，执行循环体，将变量 i 值加到 sum 中。为了再次执行循环体时加下一个数，将变量 i 加 1。循环体执行结束后，再次判断条件是否为真，如果为真则再次执行循环体。当条件不满足，即 i=101 时，循环结束，输出 sum 的值 5050。

对于循环次数有限的 while 语句，为确保循环能够正常结束，不陷入死循环(也称为无限循环，即条件一直为真)，需要在执行若干次循环后，将条件表达式取值变为假，结束循环。循环体中一定要包含能使循环条件变为假的语句，例如上面代码中的 i=i+1。

例 3.9　利用莱布尼茨级数求圆周率的值。

莱布尼茨级数为

$$\frac{\pi}{4}=\sum_{k=0}^{\infty}\frac{(-1)^k}{2k+1}$$

级数的各项在 k 为奇数时分子为－1，偶数时分子为 1；也就是级数上一项分子为 1，下一项分子为－1。

```
t=1
sum=0.0
k=0
while k<=1000:
  sum=sum+t/(2*k+1)
  t=-t
  k=k+1
print("pi=",sum*4)
```

程序运行后输出 pi= 3.1425916543395442。

如果将循环次数修改为 30000000，计算结果会更精确，输出 pi= 3.1415926869232984。

2. continue 语句

continue 语句在循环结构中执行时，将会立即结束本次循环，重新开始下一轮循环；也就是说，跳过循环体中在 continue 语句之后的所有语句，继续下一轮循环。

例 3.10 由大到小输出 10 以下不能被 3 整除的自然数。

```
x=10
while x>=0:
  if x%3==0:
    x=x-1
    continue
  print(x,end= " ")
  x=x-1
```

输出结果为 10 8 7 5 4 2 1。

当 x 为 3 的倍数时，使用 continue 结束本次循环，如 3、6、9 等不输出。由于循环变量修改在 print 语句之后，为正确进入下一次循环，在 continue 之前修改循环变量 x 的值。

使用 print()函数输出时，默认情况下会输出一个换行符。print()输出的结束符号是通过参数 end= "\n"设置的，本例中将结束符号修改为空格字符，在 print()函数中加入 end=" "，end 参数与输入内容间用逗号分隔。

print(x,end= " ")执行时输出 x 和一个空格。

3. break 语句

break 语句在循环结构中执行时，将会跳出循环结构，转而执行 while 语句后的语句，即不管循环条件是否为假，遇到 break 语句将提前结束循环。break 语句只结束当前 while 语句的循环。

例 3.11 由大到小输出 10 以下的自然数，遇到能被 3 整除的自然数时结束循环。

```
x=10
while x>=0:
  if x%3==0:
    break
  print(x,end= " ")
  x=x-1
```

输出结果为 10。

continue 语句和 break 语句的执行流程如图 3.8 所示。

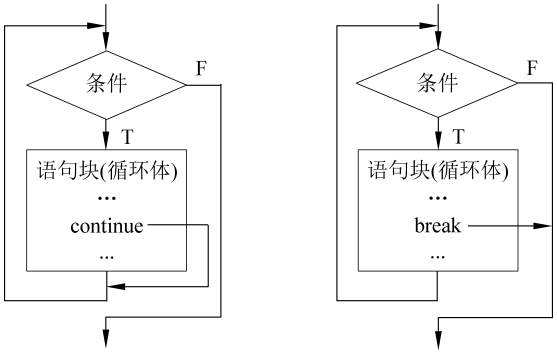


图 3.8 break 语句与 continue 语句的流程

4. 循环次数控制

while 语句既可以实现次数固定的循环，也可以实现次数不固定的循环。无论是哪种情况，程序中一定要有能够使循环条件变为假或退出循环的代码，避免运行时陷入死循环。

在实现次数固定的循环时，通常设置一个变量控制循环次数，先对循环变量赋初始值；在条件中使用循环变量设置表达式，在未达到循环次数前表达式值为真；在循环体中修改循环变量取值，最终经过若干次循环后条件表达式值为假而结束循环。如图 3.9(a)所示。

在实现次数不固定的循环时，在循环体中使用 if 语句判断循环是否结束。在 if 语句条件为真时，通常使用 break 语句结束循环，如图 3.9(b)所示。

例 3.12 编写程序，计算满足 $1^2+2^2+3^2+\cdots+n^2<1000$ 的 n 的最大值。

方法一：使用条件表达式控制循环流程。

```
i=0
sum=0
while sum<1000:
  i=i+1
  sum=sum+i*i
print(i-1)
```

方法二：使用 if 语句和 break 语句控制循环过程。

```
i=0
sum=0
while True:
  i=i+1
  sum=sum+i*i
  if sum>1000:
    break
print(i-1)
```

方法二中的条件表达式为 True，while 语句会一直执行循环体。在循环体执行过程中，当 sum>1000 时，执行 break 语句退出循环。退出循环时，i 变量为最大 n 值的下一次循环取值，最大 n 值为 $i-1$。

方法三：使用 if 语句和标志变量控制循环过程。

在循环存在多种条件时，为了让程序变得更为整洁，可定义一个变量作为循环条件，这个变量称为标志。在标志为 True 时继续执行循环体，在任何条件不满足而将标志的值设为 False 时退出循环。这样，在 while 语句中就只需要检查标志的当前值是否为 True，所有检查是否满足条件的程序都放在其他地方。如图 3.9(c)所示。

```
i=0
sum=0
flag=True
while flag:
  i=i+1
```

```
    sum=sum+i*i
    if sum>1000:
      flag=False
print(i-1)
```

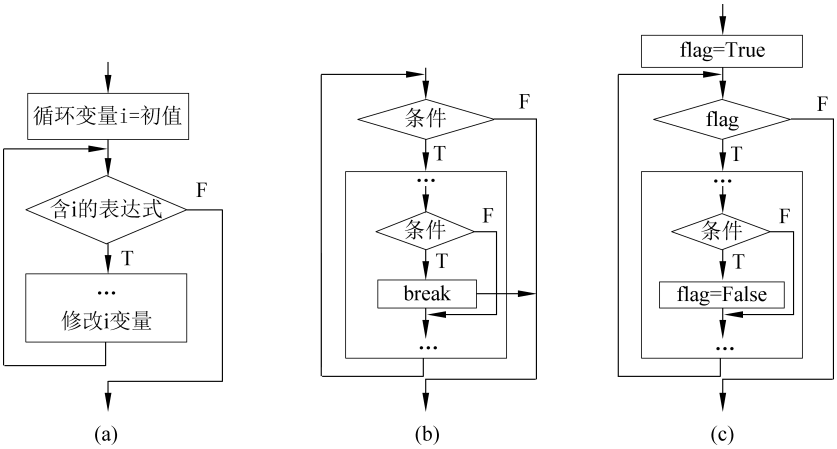


图 3.9　while 语句的循环次数控制方法

上面三种方法的输出结果都是 13。

5. 交互式循环

while 语句可以编写交互式循环，一般用于进行输入控制，由用户决定是继续输入（执行循环）还是退出。

例 3.13　输入一组整数，求输入整数的和。

```
sum=0
while True:
  t=input("please input a integer(input 'n' to exit):")
  if t!='n':
    sum+=int(t)
  else:
    break
print("sum=",sum)
```

程序运行结果如图 3.10 所示。

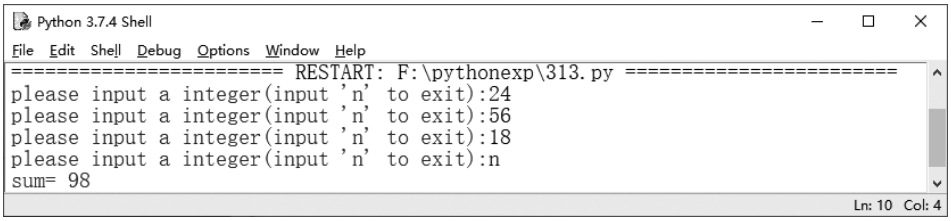


图 3.10　输入整数求和

3.2.2 while-else 语句

while 循环结束后，如果想确定是否执行了 break 语句，可以根据循环变量的取值进行判断。

例 3.14 判断正整数 n 是否是素数。

素数是只能被 1 和自身整除的数，如果 n 能够被 2～n－1 的任何一个数整除，则 n 不是素数。

```
n=int(input("please input a integer(>2):"))
i=2
while i<n:
  if n%i==0:
    break
  i=i+1
if i==n:
  print(n,"is a prime")
else:
  print(n,"is not a prime")
```

在执行循环过程中，n 如果能被 2～n－1 的任何一个数整除，则会执行 break，结束循环，此时 i＜n，表示 n 是素数；如果一直没有整除，在不满足 i＜n 条件时 while 循环正常结束，此时 i 的值为 n，n 是素数。

除了使用循环变量判断流程执行情况外，也可以使用 while-else 语句。

```
while 条件:
    语句块 1
else:
    语句块 2
```

执行 while-else 语句时，首先计算表达式的值，如果表达式的值为 True，则执行循环体语句块 1，然后重新计算表达式的值，再次判断值是否为 True；如果表达式的值为 False，则执行语句块 2。

执行 break 语句退出循环时，不会执行 else 对应的语句块。

例 3.15 判断正整数 n 是否是素数(用 while-else 语句实现)。

```
n=int(input("please input a integer(>2):"))
i=2
while i<n:
  if n%i==0:
    print(n,"is not a prime")
    break
  i=i+1
else:
  print(n,"is a prime")
```

3.2.3 for 语句

本节只介绍 for 语句用于循环时的使用方法，for 语句的其他应用在后续章节中介绍。

在 while 语句实现对 1～100 的数求和的例子中，我们设置了一个循环变量。在循环过程中，循环变量依次取 1～100 的各个数值；在循环体中，每次将循环变量值加到和中。for 语句在循环时每次取出数列中的一个值，然后执行一次循环体。

1. range()函数

range()函数生成一系列的数字，一般格式为

```
range([起始数值,]终止数值[,步长])
```

range()函数生成从起始数值到终止数值(不含终止数值)间的数字序列。起始值和步长为可选项，起始值默认为 0，步长默认值为 1。例如，range(1,5)得到的数字序列为 1，2,3,4；range(2,11,2)得到的数字序列为 2,4,6,8,10；range(5)得到的数字序列是 0,1,2,3,4；range(2,－2,－1)得到的数字序列为 2,1,0,－1。

2. for 语句格式

```
for 变量 in 遍历对象:
  语句块
```

执行 for 语句时，遍历对象中的每一个元素都会赋值给变量，然后为每个元素执行一遍循环体。

例 3.16 使用 for 语句计算 1～100 的所有数之和。

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

程序执行过程如图 3.11 所示。

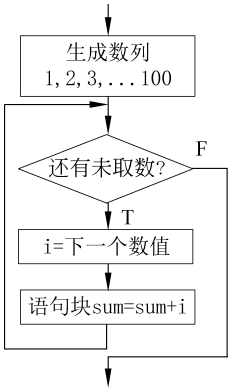


图 3.11 例 3.16 的 for 语句执行流程

3. for 语句中使用 continue 和 break 语句

for 语句中可以使用 continue 和 break 语句，用法与 while 语句中使用 continue 和 break 语句是一样的。

3.2.4 循环嵌套

一个循环的循环体内包含另外一个循环语句，这被称为循环嵌套。循环嵌套时，外层循环执行一次，内层循环从头到尾执行一遍。while 语句和 for 语句不仅可以自身嵌套，

还可以互相嵌套。

例 3.17 打印九九乘法表。

九九乘法表共有 9 行，如图 3.12 所示。每行的列数与所在的行数相同，如第 1 行有 1 列，第 9 行有 9 列。使用外层循环输出每一行内容。对 row 行（row 取值 1～9），使用内层循环输出 1～row 列内容。输出每列内容时需要输出行数、列数及行数与列数之积，这三项都是整数，与 * 等字符连接时需要先使用 str()将整数转换为字符串。

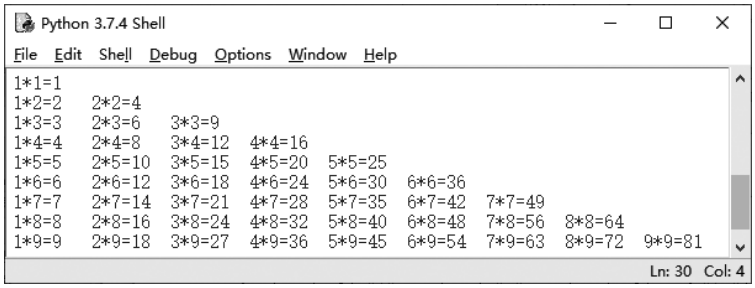


图 3.12 九九乘法表

方法一：使用 while 语句实现。

```
#coding=UTF-8
row=1                          #外层循环变量
while row<=9:                  #外层循环,每执行一次循环输出一行
  col=1                        #内层循环变量
  while col<=row:              #内层循环,每执行一次循环输出一列
    print(str(col)+"*"+str(row)+"="+str(row*col),end="\t")
                               #输出一列内容和制表符
    col=col+1
  print()                      #一行输出结束 print 输出默认的换行
  row=row+1
```

方法二：使用 for 语句实现。

```
for row in range(1,10):
  for col in range(1,row+1):
    print(str(col)+"*"+str(row)+"="+str(row*col),end="\t")
  print()
```

例 3.18 求 100～200 的全部素数。

在例 3.15 的基础上，用一个嵌套的 for 循环即可处理。

```
for i in range(100,201):
  for j in range(2,i):
    if i%j==0:
      break
  else:
```

```
    print(i,end=" ")
```

程序运行后，输出如下。

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

例 3.19 百鸡百钱问题。100 元买 100 只鸡，其中公鸡 5 元 1 只，母鸡 3 元 1 只，小鸡 1 元 3 只，要求每种鸡至少有一只，求所有的购买方案。

每种鸡的购买数是不确定的，只能从 1 开始进行判断。

```
for i in range(1,100):
  for j in range(1,100):
    if i*5+3*j+(100-i-j)/3==100:
      print(i,j,100-i-j)
```

程序中 i 表示公鸡数，j 表示母鸡数，小鸡数为 100-i-j。输出结果为

```
4 18 78
8 11 81
12 4 84
```

习　题　3

一、选择题

1. 以下可以终结一个循环的保留字是(　　)。

   A. if　　B. break　　C. exit　　D. continue

2. 以下 if 语句中(　　)是正确的。

   A. if a>=22:　　B. if (a>=22)　　C. if (a=>22)　　D. if a>=22

3. 以下关键字(　　)是用于给 if 语句添加其他条件语句的。

   A. else if　　B. elseif　　C. elif　　D. 以上都不是

4. 以下代码中(　　)是正确的 for 循环语句。

   A. for(a = 0; a < 3; a++)　　B. for a in range(3)

   C. for a loop 3:　　D. for a in range(1,3):

5. 以下代码中(　　)是正确的 while 循环语句。

   A. while loop a < 10　　B. while a < 10:

   C. while(a < 10)　　D. while loop a < 10:

6. 以下可以只终结本次循环的保留字是(　　)。

   A. if　　B. break　　C. exit　　D. continue

7. 下列程序的运行结果是(　　)。

```
n=0
while n<=2:
```

```
  print(n)
```

A. 2　　B. 3

C. 死循环，无限个 0　　D. 有语法错误

8. 以下代码的输出结果为(　　)。

```
x,y,z=True,False,False
if x or y and z:
  print("yes")
else:
  print("no")
```

A. yes　　B. no　　C. 有语法错误　　D. 运行错误

9. 以下代码的输出结果为(　　)。

```
if None:
  print("Hello")
```

A. False　　B. Hello　　C. 没有任何输出　　D. 语法错误

10. 在 if-elif-else 的多个语句块中只会执行一个语句块，这样的说法(　　)。

A. 正确　　B. 错误

C. 根据条件决定　　D. Python 中没有 elif 语句

11. 在 Python 中，for 和 while 与 else 语句的关系是(　　)。

A. 只有 for 才有 else 语句　　B. 只有 while 才有 else 语句

C. for 和 while 都可以有 else 语句　　D. for 和 while 都没有 else 语句

12. 以下代码的输出结果为(　　)。

```
i=sum=0
while i<=4:
  sum+=i
  i=i+1
print(sum)
```

A. 0　　B. 10

C. 4　　D. 以上结果都不对

13. 以下描述正确的是(　　)。

A. break 语句用于终止当前循环

B. continue 语句用于跳过当前剩余要执行的代码，执行下一次循环

C. break 和 continue 语句通常与 if、if-else 和 if-elif-else 语句一起使用

D. 以上说法都是正确的

14. 以下代码的输出结果为(　　)。

```
x,y,z=True,False,False
if not x or y:
  print(1)
```

```
elif not x or not y and z:
  print(2)
elif not x or y or not y and x:
  print(3)
else:
  print(4)
```

A. 1　　B. 2　　C. 3　　D. 4

15. 关于结构化程序设计所要求的基本结构，以下选项中描述错误的是(　　)。

A. 重复(循环)结构B. 选择(分支)结构　C. goto 跳转　　D. 顺序结构

二、编程题

1. 编写程序，用户输入 4 个数值(整数或浮点数)。先计算出三个数的平均值，然后把平均值与第四个数比较。如果相等，则程序在屏幕上输出 Equal。

```
Enter first number:4.5
Enter second number:3
Enter third number:3
Enter four number:3.5
Equal
```

2. 体重指数 BMI＝体重(千克)/身高(米)的平方，即 kg/m^2。体重过轻：BMI＜18；体重正常：BMI ＝18～25；体重过重：BMI＞25。编写程序，输入体重和身高，输出身体状况信息和建议(如“您的体重过重，请加强锻炼”)。

3. 编写使用函数 range( )的 for 循环语句，输出如下序列。

(1) 0　1　　(2) 0　　(3) 3 4 5 6　　(4) 1　　(5) 0　3　　(6) 5　9　13　17　21

4. 编写程序，用户输入一个正整数 n，在屏幕上输出前 4 个 n 的整数倍数。

5. 编写程序，用户输入一个正整数 n，在屏幕上输出 0～n(不包括 n)的平方。

6. 编写程序，用户输入一个正整数 n，在屏幕上输出 n 的所有正因子。例如输入 49，输出 1、7 和 49。

7. 编写程序，用户输入一个四位数的整数，使用标准的算术运算符处理该整数，在屏幕上输出各位数字。例如输入 1234，最后在屏幕上按顺序输出 1、2、3、4。

8. 编写一段 while 循环，计算以下值。

(1) 前 n 个数的和：$1+2+3+\cdots+n$。

(2) 前 n 个奇数的和：$1+3+5+\cdots+(2n-1)$。

(3) 求用户输入的一系列数字的总和，直到输入值为 999(注意，999 不应该加入总和)。

(4) 整数 n 可以被 2 除(使用整数除法)的次数，直到结果为 1(即 $log^2 n$)。

(5) 从 5～100 找出能被 5 或 7 整除的数。

(6) 输出 1000～1200 的素数。

9. 使用 for 语句实现第 8 题中的各小题。

10. 有四个数字 1、2、3、4，能组成多少个互不相同且无重复数字的三位数？各是多

少？（百位、十位、个位的数字取值都可以是 1、2、3、4，但是不能相等）。

11. 输入两个均不超过 9 的正整数 a 和 n，要求编写程序求 a＋aa＋aaa＋＋…＋aa…a（n 个 a）之和。提示：aa＝a×10＋a，例如 99＝9×10＋9。

12. 编写程序，计算序列 1＋1/3＋1/5＋…的前 N 项之和，N 从键盘输入。

13. 编写程序，计算交错序列 1－2/3＋3/5－4/7＋5/9－6/11＋…的前 N 项之和，N 从键盘输入。

14. 一个球从 100 米高度自由落下，每次落地后反跳回原高度的一半，再落下，再反弹。求它在第 10 次落地时，共经过多少米？第 10 次反弹多高？

15. 输出所有的“水仙花数”。例如，153 是水仙花数，因为 $153=1^3+5^3+3^3$。

16. 一个数如果恰好等于它的因子之和，则这个数称为“完数”。例如，6 的因子为 1、2、3，而 6＝1＋2＋3，因此 6 是完数。编写程序，输出 1000 之内的所有完数。

17. 输入三个整数 x、y、z，把这三个数由小到大输出。

18. 设有 5 分制测验，等级为 5——A、4——B、3——C、2——D、1——E、0——F。编写程序，输入测验成绩并打印出相应的等级。