第 3 章

Python 的基本流程控制

流程控制对于任何一门编程语言来说都是非常重要的，因为它提供了控制程序如何执行的方法。如果没有流程控制，整个程序都将按照从上至下的顺序来执行，而不能根据用户的需求决定程序执行的顺序。本章将对 Python 中的选择结构、循环结构、跳转等流程控制语句进行介绍。

3.1 基本语句及顺序结构

语句是 Python 程序的过程构造块，用于定义函数、定义类、创建对象、变量赋值、调用函数、控制分支、创建循环等。Python 语句分为简单语句和复合语句。简单语句包括表达式语句、赋值语句、assert 语句、pass 语句、return 语句、break 语句、continue 语句、import 语句等。复合语句包括 if 语句、while 语句、for 语句、try 语句、函数定义、类定义等。

计算机在解决某个具体问题时，主要有 3 种情形，分别是顺序执行所有的语句、选择执行部分语句和循环执行部分语句。对应程序设计中的 3 种基本结构是顺序结构、选择结构和循环结构。

3.1.1 基本语句

1. 赋值语句

使用赋值号(=)将右边的值(表达式)赋给左边变量的语句称为赋值语句。例如：

```
name='李福'
age=18
score=82.5
value=3+2j
```

上述 4 条赋值语句分别实现：为变量 name 赋值一个字符串，为变量 age 赋值一个整数，为变量 score 赋值一个浮点数，为变量 value 赋值一个复数。

2. 复合型赋值语句

复合赋值语句是使用复合运算符(包括算术复合运算符和位复合运算符)的赋值语句，包括序列赋值、多目标赋值和复合赋值等。

(1) 序列赋值,例如:

```
x,y=10,20
```

序列赋值可以为多个变量分别赋予不同的值,变量之间用英文逗号隔开。实际上是利用元组和序列解包实现的。例如:

```
a,b,c,d,e='hello'
```

上述语句的功能是分别将 5 个字符依次赋值给 5 个变量,a 的值为“h”,b 的值为“e”,以此类推。又如:

```
name,age,addr,tel=['李四',20,'北京','18601001234']
```

上述语句的功能是分别将右侧的 4 个值赋值给左边的 4 个变量,name 的值为“李四”,age 的值为 20,以此类推。

Python 可以通过序列赋值语句实现两个变量值的交换。例如:

```
math=80
English=75
math,English =English,math
```

执行以上两条语句之后,math 与 English 的值发生了互换,math 的值为 75,English 的值为 80。

(2) 多目标赋值。多目标赋值就是将同一个值赋值给多个变量。例如:

```
x=y=z=20
```

多目标赋值通常只用于赋予数值或字符串这种不可变类型,如果欲赋予可变类型(如列表类型),则可能会出现问题。

(3) 复合赋值。

- +=加法赋值运算符:c+=a 等效于 c=c+a。
- -=减法赋值运算符:c-=a 等效于 c=c-a。
- *=乘法赋值运算符:c*=a 等效于 c=c*a。
- /=除法赋值运算符:c/=a 等效于 c=c/a。
- %=取模赋值运算符:c%=a 等效于 c=c%a。
- **=幂赋值运算符:c**=a 等效于 c=c**a。
- //=取整除赋值运算符:c//=a 等效于 c=c//a。

Python 语句涉及许多程序构造要素,将在本书后续章节陆续阐述。

3.1.2　顺序结构

程序工作的一般流程为:数据输入、运算处理、结果输出。顺序结构是指为了解决某些实际问题,自上而下依次执行各条语句,其流程如图 3-1 所示。

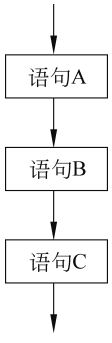

图 3-1　顺序结构流程图

下面通过几个例子学习使用顺序结构解决各种常见问题。

【例 3-1】 编写程序，从键盘输入语文、数学、英语三门功课的成绩，计算并输出平均成绩，要求平均成绩小数点后保留 1 位。

分析：程序的执行流程为：输入三门功课成绩、计算平均成绩、输出平均成绩。输入时使用转换函数将字符串转换为浮点数，输出时采用格式输出方式控制小数点的位数。

```
chinese=float(input("请输入您的语文成绩:"))
math=float(input("请输入您的数学成绩:"))
english=float(input("请输入您的英语成绩:"))
average=( chinese +math +english)/3
print("您的平均成绩为:%.1f" %average)
```

运行结果：

```
请输入您的语文成绩:80
请输入您的数学成绩:87
请输入您的英语成绩:98
您的平均成绩为:88.3
```

【例 3-2】 编写程序，从键盘输入圆的半径，计算并输出圆的周长和面积。

分析：在计算圆的周长和面积时需要使用 π 的值，Python 的 math 模块中包含常量 pi，通过导入 math 模块可以直接使用该值，然后使用周长和面积公式进行计算即可。

```
import math
radius=float(input("请输入圆的半径:"))
circumference=2*math.pi*radius
area=math.pi*radius*radius
print("圆的周长为:%.2f" %circumference)
print("圆的面积为:%.2f" %area)
```

运行结果：

```
请输入圆的半径:5
圆的周长为:31.42
圆的面积为:78.54
```

3.2 选择结构

分支结构可以分为单分支结构和多分支结构，用于解决生活中形形色色的选择问题。例如，驾驶员理论考试科目中，成绩达到 90 分的为合格。

选择结构语句，也称为条件判断语句，即按照条件选择执行不同的代码片段。Python 中选择语句主要有 3 种形式，分别为 if 语句、if…else 语句和 if…elif…else 多分支语句。

在其他语言中(例如，C、C++ 、Java 等)，选择语句还包括 switch 语句，也可以实现多

重选择。但是，在 Python 中却没有 switch 语句，所以实现多重选择的功能时，只能使用 if…elif…else 多分支语句或者 if 语句的嵌套。

3.2.1　if 语句

Python 中使用 if 保留字来组成选择语句。if 语句仅处理条件成立的情况，其流程如图 3-2 所示。从图中可以看出，当表达式的值为真时，执行相应的语句块（一条或多条语句）；当表达式的值为假时，直接跳出 if 语句，执行其后面的语句。语法格式：

```
if 表达式:
    语句块
```

其中，表达式可以是一个单纯的布尔值或变量，也可以是比较表达式或逻辑表达式（例如，a>band a!=c），如果表达式为真，则执行“语句块”；如果表达式为假，就跳过“语句块”，继续执行后面的语句，这种形式的 if 语句相当于汉语里的关联词语“如果……就……”。

关键字 if 与表达式之间用空格隔开，表达式后接英文冒号，语句块中的全部语句均缩进 4 个空格，如图 3-3 所示。

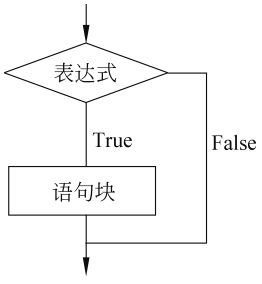

图 3-2　单分支结构流程图

```
if 表达式:
  语句块
```

图 3-3　单分支结构书写格式

【例 3-3】 输入姓名和年龄，判断是否成年。

```
name=input("请输入您的姓名:")
age=int(input("请输入您的年龄:"))
if age>=18:
    print(name,"已经成年")
print("符合驾照考试规定")
```

运行结果：

```
请输入您的姓名:李福
请输入您的年龄:20
李福已经成年
符合驾照考试规定
```

使用 if 语句时，如果只有一条语句，语句块可以直接写到“：”的右侧，例如，“if a>b：max = a”，但是为了程序代码的可读性，建议不要这么做。

if 语句使用过程中的常见错误如下。

(1) if 语句后面未加冒号。

(2) 使用 if 语句时,如果在符合条件时,需要执行多个语句,但是在第二个输出语句的位置没有缩进。

3.2.2 if…else 语句

生活中经常遇到只能二选一的条件,例如,大学毕业是直接就业还是考研深造。Python 中提供了 if…else 语句解决类似问题,其语法格式如下:

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

使用 if…else 语句时,表达式可以是一个单纯的布尔值或变量,也可以是比较表达式或逻辑表达式,如果满足条件,则执行 if 后面的语句块,否则执行 else 后面的语句块,这种形式的选择语句相当于汉语里的关联词语“如果……否则……”。

【例 3-4】 询问年龄,如果年龄大于或等于 18 岁,输出“恭喜! 你成年了。”,如果小于 18 岁,输出“要年满 18 岁才成年,你还差 * 岁”。

```
age =int(input("你的年龄是:"))
if age >=18:
    print("恭喜!你成年了。")
else:
    diff =str(18 -age)
print("要年满 18 岁才成年,你还差 " +diff +" 岁")
```

运行结果:

```
第一种情况:
你的年龄是:20
恭喜!你成年了。
第二种情况:
你的年龄是:15
要年满 18 岁才成年,你还差 3 岁
```

【例 3-5】 编写程序,从键盘输入三条边的边长,判断是否能够构成一个三角形。如果能,则提示可以构成三角形;如果不能,则提示不能构成三角形。

分析:组成三角形的条件是任意两边之和大于第三边,如果条件成立,则能构成三角形;当条件表达式中的多个条件必须全部成立时,条件之间可用 and 运算符连接起来。

```
side1 =float(input("请输入三角形第一条边:"))
side2 =float(input("请输入三角形第二条边:"))
side3 =float(input("请输入三角形第三条边:"))
if (side1 +side2 >side3) and (side2 +side3 >side1)
```

```
                    and (side1 +side3 >side2):
    print(side1,side2,side3,"可以构成三角形")
else:
    print(side1,side2,side3,"不能构成三角形")
```

运行结果：

```
请输入三角形第一条边:3
请输入三角形第二条边:4
请输入三角形第三条边:5
3.0 4.0 5.0 可以构成三角形
```

3.2.3　if…elif…else 语句

if…elif…else 语句主要用于处理多种条件的情况，从而解决现实生活中复杂的多重选择问题，其流程如图 3-4 所示。

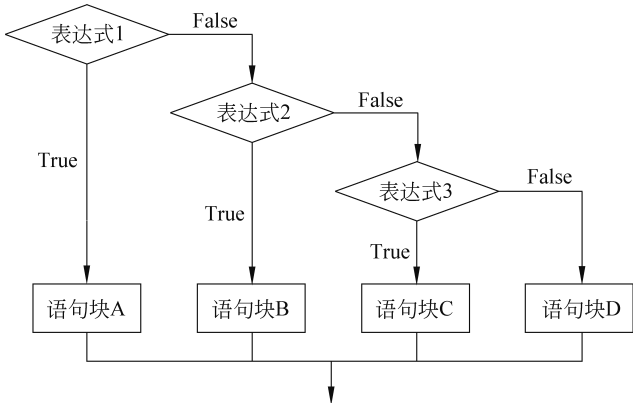

图 3-4　多分支结构流程图

使用 if…elif…else 语句时，表达式可以是一个单纯的布尔值或变量，也可以是比较表达式或逻辑表达式，如果表达式为真，执行语句；而如果表达式为假，则跳过该语句，进行下一个 elif 的判断；只有在所有表达式都为假的情况下，才会执行 else 中的语句。

如果表达式 1 的值为真，则执行相应的语句块 A；如果表达式 1 的值为假，则继续判断表达式 2 的值，如果表达式 2 的值为真，则执行语句块 B；如果表达式 2 的值也为假，则继续判断表达式 3 的值；以此类推，直到所有的表达式都不满足（条件表达式的个数为 1 个或多个）为止，然后执行 else 后面的语句块。

```
if 表达式 1:
    语句块 A
elif 表达式 2:
    语句块 B
elif 表达式 3:
    语句块 C
else:
    语句块 D
```

图 3-5　多分支结构书写格式

书写格式：关键字 if 与表达式 1 之间用空格隔开，表达式 1 后接英文冒号；所有关键字 elif 均与关键字 if 左对齐，elif 与后面的各个表达式之间用空格隔开，表达式后接英文冒号；关键字 else 与关键字 if 左对齐，后接英文冒号；所有语句块左对齐，即所有语句块中的全部语句均缩进 4 个空格，如图 3-5 所示。

【例 3-6】 输入两个数，比较它们的大小并输出其中较大者。

```
x =int(input("请输入第一个整数: "))
y =int(input("请输入第二个整数: "))
if (x ==y):
    print("两数相同!")
elif (x >y):
    print("较大数为:",x)
else:
    print("较大数为:" ,y)
```

运行结果：

```
请输入第一个整数: 2
请输入第二个整数: 3
较大数为: 3
```

如果只考虑一种表达式成立或不成立的结果(即没有 elif 分支)，则多分支的 if 结构转换为双分支的 if 结构。

在使用分支结构时，需要注意以下事项。

(1) 表达式可以是任意类型，如 5>3，x and y>z，3，0 等。其中，3 表示恒真(即 True)，而 0 表示恒假(即 False)。

(2) 可以仅有 if 子句构成单分支结构，但是 else 子句必须与 if 子句配对，不能出现仅有 else 子句没有 if 子句的情况。

(3) 各语句块可以是一条或多条语句，如果是多条语句，则所有语句必须左对齐。

【例 3-7】 编写程序，判断中国合法工作年龄为 18～60 岁，即如果年龄小于 18 的情况为童工，不合法；如果年龄为 18～60 岁为合法工龄；大于 60 岁为法定退休年龄。

```
age =int(input('请输入您的年龄:'))
if age <18:
    print(f'您的年龄是{age},童工一枚')
elif (age >=18) and (age <=60):
    print(f'您的年龄是{age},合法工龄')
elif age >60:
    print(f'您的年龄是{age},可以退休')
```

运行结果：

```
请输入您的年龄:20
您的年龄是 20,合法工龄
```

【例 3-8】 编写程序，调用随机函数生成一个 1～100 的随机整数，从键盘输入数字进行猜谜，给出猜测结果(太大、太小、成功)的提示。

分析：通过引入 random 模块，可以调用其中的 randint(a，b)函数产生介于 a 和 b 之间的随机整数(即产生的随机数大于等于 a 且小于等于 b)，然后从键盘输入一个数字与

该随机数进行比较，并输出判断结果。

```
import random
randnumber=random.randint(1,100)
guess=int(input("请输入您的猜测:"))
if guess>randnumber:
    print("您的猜测太大")
elif guess<randnumber:
    print("您的猜测太小")
else:
    print("恭喜您猜对了")
```

运行结果：

```
请输入您的猜测:20
您的猜测太小
```

3.2.4　分支语句嵌套

当有多个条件需要满足并且条件之间有递进关系时，可以使用分支语句的嵌套。其中，if 子句、elif 子句以及 else 子句中都可以嵌套 if 语句或者 if…elif…else 子句。

书写格式：嵌套的 if 语句要求以锯齿形缩进格式书写，以便分清层次关系。

【例 3-9】 我国的婚姻法规定，男性 22 岁为合法结婚年龄，女性 20 岁为合法结婚年龄。因此如果要判断一个人是否到了合法结婚年龄，首先需要使用双分支结构判断性别，再用递进的双分支结构判断年龄，并输出判断结果。

```
sex=input("请输入您的性别(M 或者 F):")
age=int(input("请输入您的年龄(1~20):"))
if sex=='M':
    if age>=22:
        print("到达合法结婚年龄")
    else:
        print("未到合法结婚年龄")
else:
    if age>=20:
        print("到达合法结婚年龄")
    else:
        print("未到合法结婚年龄")
```

运行结果：

```
请输入您的性别(M 或者 F):F
请输入您的年龄(1~20):28
到达合法结婚年龄
```

【例 3-10】 编写程序，从键盘输入用户名和密码，要求先判断用户名再判断密码，如

果用户名不正确，则直接提示用户名输入有误；如果用户名正确，则进一步判断密码，并给出判断结果的提示。

分析：因为要求先判断用户名再判断密码，所以本程序的一种做法是使用 if 语句的嵌套，外层 if 语句用于判断用户名，用户名正确时进入内层 if 语句判断密码并给出判断结果，如果用户名不正确，则直接给出错误提示。

```
username=input("请输入您的用户名:")
password=input("请输入您的密码:")
if username=="admin":
    if password=="123456":
        print("输入正确,恭喜进入!")
    else:
        print("密码有误,请重试!")
else:
    print("用户名有误,请重试!")
```

运行结果：

```
请输入您的用户名:admin
请输入您的密码:123456
输入正确,恭喜进入!
```

【例 3-11】 编写程序，开发一个小型计算器，从键盘输入两个数字和一个运算符，根据运算符(+、-、*、/)进行相应的数学运算，如果不是这 4 种运算符，则给出错误提示。

分析：因为需要根据 4 种运算符号的类别执行相应的运算，所以使用多分支 if…elif…else 语句；对于除法运算而言，由于除数不能为 0，因此需要使用嵌套的 if 语句来判断除数是否为 0，并执行相应的运算。

```
first=float(input("请输入第一个数字:"))
second=float(input("请输入第二个数字:"))
sign=input("请输入运算符号:")
if sign=='+':
    print("两数之和为:",first+second)
elif sign=='-':
    print("两数之差为:",first-second)
elif sign=='*':
    print("两数之积为:",first*second)
elif sign=='/':
    if second!=0:
        print("两数之商为:",first/second)
    else:
        print("除数为 0 错误!")
else:
    print("符号输入有误!")
```

运行结果：

```
请输入第一个数字:2
请输入第二个数字:3
请输入运算符号:+
两数之和为: 5.0
```

if 选择语句可以有多种嵌套方式，开发程序时可以根据自身需要选择合适的嵌套方式，但一定要严格控制好不同级别代码块的缩进量。

3.3　循 环 结 构

循环问题渗透在日常生活的方方面面，例如，学生上学，每天从宿舍到教室，往返于这两个点。类似这样反复做同一件事的情况，称为循环。

循环主要有两种类型：重复一定次数的循环，称为计次循环，如 for 循环；一直重复，直到条件不满足时才结束的循环，称为条件循环，只要条件为真，这种循环会一直持续下去，如 while 循环。

3.3.1　while 语句

while 循环是通过一个条件来控制是否要继续反复执行循环体中的语句。while 语句用于在满足循环条件时重复执行某件事情，其流程如图 3-6 所示。

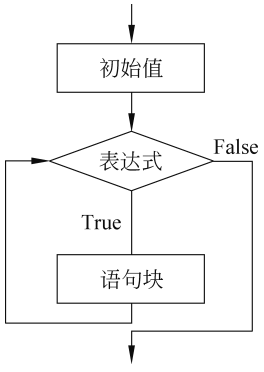

图 3-6　while 循环结构流程图

从图中可以看出，当表达式的值为真时，执行相应的语句块（循环体），然后再判断表达式的值，如果为真，则继续执行语句块；当表达式的值为假时，检查其后面是否有 else 子句（因为可选，所以流程图未画出），如果有，则执行 else 子句；如果没有，则直接跳出 while 语句，执行其下面的语句。

语法格式：

```
while 条件表达式:
循环体
```

循环体是指一组被重复执行的语句。

【例 3-12】　将“不忘初心”输出 3 次。

```
i=1
while i <=3:
    print("不忘初心")
i=i+1
```

运行结果：

```
不忘初心
```

```
不忘初心
不忘初心
```

在使用 while 语句时，需要注意以下事项。

（1）与 if 语句类似，while 语句的表达式可以是任意类型，如 x!=y，x>3 or x<5，-5 等。

（2）循环体中的语句块有可能一次也不执行，例 3-12 中若初始值 i=4，则语句块不会执行。

（3）语句块可以是一条或多条语句，例 3-12 中 while 子句的语句块为两条语句，else 子句中的语句块为一条语句。

（4）程序中需要包含使循环结束的语句，例 3-12 中若缺少语句 i=i+1，则程序无法终止。

在 while 循环中，如果表达式的值恒真，循环将一直执行下去，无法靠自身终止，从而产生死循环。例如：

```
while 1:
    print("Python是一门编程语言")
```

书写程序时，有时要尽量避免死循环，但是在某些特定场合中死循环却具有十分重要的作用，如嵌入式编程、网络编程中等。

【例 3-13】 编写程序，用下列公式计算 π 的近似值，直到最后一项的绝对值小于10^{-6}为止。

$$\frac{\pi}{4}\approx 1-\frac{1}{3}+\frac{1}{5}-\frac{1}{7}+\frac{1}{9}-\cdots$$

分析：观察 π 的计算公式可知，循环变量的初始值为 1，循环条件为循环变量的绝对值大于等于 10^{-6}，循环变量值的变化规律如上式所示，每项的分母比前一项增加 2，符号与前一项相反。

```
import math
n=1                                  #变量自增值
t=1                                  #每项值
total=0                              #π/4的值
flag=1                               #标记位
while math.fabs(t)>=1e-6:            #当每项值的绝对值大于1e-6时进行计算
    total=total+t
    flag=-flag
    n=n+2
    t=flag*1.0/n
print("π=%f"%(total*4))
```

运行结果：

```
π=3.141591
```

【例 3-14】 取款机输入密码模拟。一般在取款机上取款时需要输入 6 位银行卡密码，接下来模拟一个简单的取款机(只有 1 位密码)，每次要求用户输入 1 位数字密码，密码正确则输出“密码输入正确，正进入系统!”；如果输入错误，输出“密码输入错误，您已经输入 * 次”，密码连续输入错误 6 次后输出“您的卡将被锁死，请和发卡行联系”。

分析：每次输入 1 个字符，默认密码为 0，连续输入 6 次错误后提醒并退出。

```
password=0
i =1
while i <7:
    num =input("请输入一位数字密码:")
    num =int(num)
    if num ==password:
        print("密码输入正确,正进入系统!")
        i=7
    else:
        print("密码输入错误,您已经输错",i,"次")
    i +=1
if i ==7:
    print("您的卡将被锁死,请和发卡行联系!")
```

运行结果：

```
请输入一位数字密码:6
密码输入错误,您已经输错 1 次
请输入一位数字密码:2
密码输入错误,您已经输错 2 次
请输入一位数字密码:0
密码输入正确,正进入系统!
```

3.3.2　for 语句和 range()内建函数

for 循环语句是一个计次循环，通常适用于枚举或遍历序列，以及迭代对象中的元素。一般应用在循环次数已知的情况下。基本语法如下：

```
for 迭代变量 in 对象:
    循环体
```

其中，迭代变量用于保存读取出的值；对象为要遍历或迭代的对象，该对象可以是任何有序的序列对象，如字符串、列表和元组等；循环体为一组被重复执行的语句。

1. 进行数值循环

在使用 for 循环时，最基本的应用就是进行数值循环。循环可以帮助人们解决很多重复的输入或计算问题。利用数值循环输出 3 遍“不忘初心”，代码如下：

```
for i in [1,2,3]:
    print("不忘初心")
```

【例 3-15】 利用数值循环输出列表的值，如输出["pku","tsinghua","fudan","sjtu","nju","zju","ustc"]中的值。

```
for i in ["pku","tsinghua","fudan","sjtu","nju","zju","ustc",
          "hit",xjtu"]:
    print(i)
```

运行结果：

```
pku
tsinghua
fudan
sjtu
nju
zju
ustc
```

利用列表可以输出一些简单重复的内容，但如果循环次数过多，如要实现从 1 到 1 00 的累加，可以使用 range()函数。

range()函数是 Python 内置的函数，用于生成一系列连续的整数，多用于 for 循环语句中。其语法格式如下：

```
range(start, end, step)
```

参数说明：

- start：用于指定计数的起始值，可以省略，如果省略，默认值为 0。
- end：用于指定计数的结束值(但不包括该值，如 range(7)得到的值为 0～6，不包括 7)，不能省略。当 range()函数中只有一个参数时，即表示指定计数的结束值。
- step：用于指定步长，即两个数之间的间隔可以省略，如果省略则表示步长为 1。例如，range(1,7)将得到 1、2、3、4、5、6。

若指定 step 为 0，则抛出 ValueError 异常。

当 step 为正时，range 的值由公式：r[i]＝start＋step＊i，而 i>=0 并且 r[i]<stop。

当 step 为负时，range 的值由公式：r[i]＝start＋step＊i，而 i>=0 并且 r[i]>stop。

在使用 range()函数时，如果只有一个参数，那么表示指定的是 end；如果是两个参数，则表示指定的是 start 和 end；只有三个参数都存在时，最后一个才表示步长。

【例 3-16】 计算 1＋2＋3＋4＋…＋100 的结果。

```
print("计算 1+2+3+4+…+100 的结果为:")
result=0
for i in range(1,101,1):
    result +=i
print(result)
```

运行结果：

```
计算 1+2+3+4+…+100 的结果为:
```

```
5050
```

2. 遍历字符串

使用 for 循环语句除了可以循环数值，还可以逐个遍历字符串。

【例 3-17】 以遍历方式计算出“黑化肥发灰会挥发；灰化肥挥发会发黑”中“发”在字符串中出现的次数。

```
word = '黑化肥发灰会挥发；灰化肥挥发会发黑'
sum = 0
for letter in word:
    if letter == '发':
        sum += 1
print(sum)
```

运行结果：

```
4
```

【例 3-18】 编写程序，解决以下问题。

4 个人中有一人做了好事，已知有 3 个人说了真话，根据下面的对话判断是谁做的好事。

A 说：不是我；

B 说：是 C；

C 说：是 D；

D 说：C 胡说。

分析：做好事的人是 4 个人其中之一，因此可以将 4 个人的编号存入列表中，然后使用 for 循环依次判断；有 3 个人说了真话，将编号依次代入，使用 if 语句判断是否满足“3 人说真话”(3 个逻辑表达式的值为真)的条件，如果满足，则输出结果。

```
for iNum in ['A', 'B', 'C', 'D']:
    if (iNum != 'A') + (iNum == 'C') + ( iNum== 'D') + (iNum != 'D') == 3:
        print(iNum, "做了好事!")
```

运行结果：

```
C 做了好事!
```

3. 迭代对象

从理论上来说，循环对象和 for 循环调用之间还有一个中间层，该层将循环对象转换为可迭代对象。这一转换通过使用 iter()函数实现。但从逻辑层面上常常可以忽略这一层，所以循环对象和可迭代对象常常相互指代对方。

后续章节所要讲述的列表、元组、字符串、集合等都是可迭代对象。可迭代对象指的是可以返回一个迭代器的对象，如果不清楚哪个是可迭代对象，可以通过 Python 内建的 iter()函数测试。例如 print(iter(range(1,100,1)))，运行后显示：<range_iterator object at 0x000000000209F750>，即 iter()函数为 range 返回了 range_iterator 对象。

3.3.3 循环语句嵌套

在Python中,允许在一个循环体中嵌入另一个循环,这称为循环嵌套。在Python中,for循环和while循环都可以进行循环嵌套。

为了解决复杂的问题,可以使用循环语句的嵌套,嵌套层数不限,但是循环的内外层之间不能交叉。其中,双层循环是一种常用的循环嵌套,循环的总次数等于内外层次数之积。例如:

```
for i in range(1,3):
    for j in range(1,4):
        print  (i*j,end=" ")
```

当外层循环变量i的值为1时,内层循环j的值从1开始,输出i*j的值并依次递增,因此输出"1 2 3",内层循环执行结束;然后回到外层循环,i的值递增为2,内层循环变量i的值重新从1开始,输出i*j的值,并依次递增,输出"2 4 6"。因此,程序的运行结果为"1 2 3 2 4 6"。

【例3-19】 编写程序,使用双重循环输出九九乘法表。

分析:由于需要输出9行9列的二维数据,因此需要使用双重循环,外层循环用于控制行数,内层循环用于控制列数。为了规范输出格式,可以使用print语句的格式控制输出方式。其中,"\t"的作用是跳到下一个制表位。

```
for i in range(1,10):
    for j in range(1,i+1):
        d=i * j
        print('%d*%d=%-2d'%(j,i,d),end=' ')
    print()
```

运行结果:

```
1*1=1
1*2=2  2*2=4
1*3=3  2*3=6  3*3=9
1*4=4  2*4=8  3*4=12 4*4=16
1*5=5  2*5=10 3*5=15 4*5=20 5*5=25
1*6=6  2*6=12 3*6=18 4*6=24 5*6=30 6*6=36
1*7=7  2*7=14 3*7=21 4*7=28 5*7=35 6*7=42 7*7=49
1*8=8  2*8=16 3*8=24 4*8=32 5*8=40 6*8=48 7*8=56 8*8=64
1*9=9  2*9=18 3*9=27 4*9=36 5*9=45 6*9=54 7*9=63 8*9=72 9*9=81
```

【例3-20】 编写程序,使用双重循环输出如图3-7所示三角形图案。

分析:观察可知图形包含5行,因此外层循环执行5次;每行的内容由三部分组成:第一部分为输出空格,第二部分为输出星号,第三部分为输出回车,分别通过两个for循环和一条print语句实现。

```
    *
   ***
  *****
 *******
*********
```

图3-7 三角形图案

```
for i in range(1,6):
    for j in range(5-i):
        print(" ",end=" ")
    for j in range(1,2*i):
        print("*",end=" ")
print("\n")
```

3.4　转移和中断语句

当循环条件一直满足时，程序将会一直执行下去。如果希望在中间离开循环，也就是 for 循环结束计数之前，或者 while 循环找到结束条件之前，有以下两种途径。

(1) 使用 break 语句完全终止循环。

(2) 使用 continue 语句直接跳到下一次循环。

3.4.1　break 语句

break 语句可以终止当前的循环，包括 while 和 for 在内的所有控制语句。以独自一人沿着操场跑步为例，原计划跑 10 圈。可是在跑到第 3 圈的时候，遇到自己的女神或者男神，于是果断停下来，终止跑步，这就相当于使用了 break 语句提前终止了循环。

break 语句的语法比较简单，只需要在相应的 while 或 for 语句中加入即可。break 语句一般会结合 if 语句进行搭配使用，表示在某种条件下跳出循环。如果使用嵌套循环，break 语句将跳出最内层的循环。

1. 在 while 语句中使用 break 语句

一般语法格式：

```
while 条件表达式 1:
    执行代码
if 条件表达式 2:
    break
```

其中，条件表达式 2 用于判断何时调用 break 语句跳出循环。

【例 3-21】 输出字母或数字的 ASCII 值。编写一个程序，用户输入字母和数字时，输出该字母或数字的 ASCII 值。当用户输入非数字或字母(如特殊符号“@”“*”等)时，退出程序。(数字 0～9 的 ASCII 值为 48～57，字母 A～Z，a～z 的 ASCII 值为 65～90，97～122。)

分析：通过 ord()函数判断字符的 ASCII 码值，如果输入字母或数字，输出 ASCII 值。如果遇到 7，退出。

```
strnum="0"
while ord(strnum) !=55:                    #输入数字 7,输出 7 的 ASCII 码值,然后退出程序
    strnum =input("请输入一个字母或数字:")
    if len(strnum) ==1:
```

```
        if ord(strnum) in range(65,91) or ord(strnum) in range(97,123)
                        or ord(strnum) in range(48,58):
            print(ord(strnum))
        else:
            print("输入字符不合法,退出程序!")
            break
    else:
        print("输入长度超过一个字符,请重新输入")
        strnum="0"
```

2. 在 for 语句中使用 break 语句

一般语法格式：

```
for 迭代变量 in 对象:
    if 条件表达式:
        break
```

其中,条件表达式用于判断何时调用 break 语句跳出循环。

【例 3-22】 输入一个整数,判断是否为素数。只能被 1 和自身整除的数字为素数。例如输入 9,判断 9 能否被 2～8 整除,如果能,说明不是素数;如果都不能,说明是素数。

```
number =int(input("请输入整数:"))  #9    2~8

if number <2:
    print("不是素数")
else:
    for i in range(2, number):
        if number %i ==0:
            print("不是素数")
            break          #如果有结论了,就不需要再和后面的数字比较了
        else:
            print("是素数")
```

3. 半路循环

前面介绍过死循环的概念,在死循环程序中,通过添加 break 语句终止程序的执行,称为半路循环。

【例 3-23】 通过输入一行字符,演示半路循环的使用。

```
number=1
while 1:
    print("Python 是一门编程语言")
    if number>=5:
        break
    number=number+1
```

运行结果：

```
Python 是一门编程语言
Python 是一门编程语言
Python 是一门编程语言
Python 是一门编程语言
Python 是一门编程语言
```

3.4.2　continue 语句

continue 语句的作用没有 break 语句强大，它只能终止本次循环而提前进入到下一次循环中。仍然以独自一人沿着操场跑步为例，原计划跑步 10 圈。当跑到第 2 圈一半的时候，遇到自己的女神或者男神也在跑步，于是果断停下来，跑回起点等待，制造一次完美邂逅，然后从第 3 圈开始继续。

continue 语句的语法比较简单，只需要在相应的 while 或 for 语句中加入即可。continue 语句一般会结合 if 语句进行搭配使用，表示在某种条件下，跳过当前循环的剩余语句，然后继续进行下一轮循环。如果使用嵌套循环，continue 语句将只跳过最内层循环中的剩余语句。

1. 在 while 语句中使用 continue 语句

一般语法结构：

```
while 条件表达式 1
    执行代码
if 条件表达式 2
    continue
```

其中，条件表达式 2 用于判断何时调用 continue 语句跳出循环，开始下一次循环。

【例 3-24】 编写重复猜数游戏。要求：如果没有猜对，提示大了或小了；如果猜对了，提示正确，并显示猜了多少次。

```
import random
#生成随机数(包含两端)
random_number =random.randint(1, 100)

count =0
while True:
    count+=1
    input_number =int(input("请输入:"))
    if input_number >random_number:
        print("大了")
    elif input_number <random_number:
        print("小了")
    else:
        print("正确,猜了"+str(count)+"次")
        break
```

运行结果：

```
请输入:68
小了
请输入:98
大了
请输入:
```

由于是随机产生的数，每次运行的判断是不同的。

2. 在 for 语句中使用 continue 语句

一般语法结构：

```
for 迭代变量 in 对象:
    if 条件表达式:
        continue
```

其中，条件表达式用于判断何时调用 continue 语句跳出循环。

【例 3-25】 编写程序，从键盘输入一段文字，如果其中包括“色”字（可能出现 0 次、1 次或者多次），则输出时过滤掉该字，其他内容原样输出。

分析：从键盘输入一段文字，可以使用 for 循环依次取出其中的每个字，然后通过 if 语句进行判断，如果有“色”字，则使用 continue 语句跳出本次循环（不输出该字），进入下一轮循环条件的判断。

```
sentence=input("请输入一段文字:")
for word in sentence:
    if word=="色":
        continue
    print(word,end="")
```

运行结果：

```
请输入一段文字:谈虎色变
谈虎变
```

【例 3-26】 编写程序，从键盘输入密码，如果密码长度小于 6，则要求重新输入。如果长度等于 6，则判断密码是否正确，如果正确则中断循环，否则提示错误并要求继续输入。

分析：因为程序没有执行次数规定，所以循环条件设置为恒真，首先判断输入长度，如果输入长度过短，则直接使用 continue 语句中断本轮循环并进入下一轮输入；如果输入长度正确，则进行密码判断，如果正确，则使用 break 语句中断循环，否则提示错误并进入下一轮输入。

```
while 1:
    password=input("请输入密码:")
    if len(password)<6:
        print("长度为 6 位,请重试!")
```

```
        continue
    if password=="123456":
        print("恭喜您,密码正确!")
        break
    else:
        print("密码有误,请重试!")
```

运行结果：

```
请输入密码:123
长度为 6 位,请重试!
请输入密码:123456
恭喜您,密码正确!
```

【例 3-27】 逢 7 拍腿游戏。几个朋友一起玩“逢 7 拍腿”游戏，即从 1 开始依次数数，当数到 7(包括尾数是 7 的情况)或 7 的倍数时，则不说出该数，而是拍一下腿。现在编写程序，计算从 1 数到 99，一共要拍多少次腿？(前提是每个人都没有出错的情况下。)

解题思路：通过在 for 循环中使用 continue 语句实现“逢 7 拍腿”游戏，即计算从 1 数到 100(不包括 100)，一共要拍多少次腿。

```
total=99
for num in range(1,100):
    if num %7 ==0:
        continue
    else:
        strnum =str(num)
        if strnum.endswith('7'):          #判断是否以 7 结束
            continue
    total -=1
print("从 1 数到 99 共拍腿",total,"次")
```

运行结果：

```
从 1 数到 99 共拍腿 22 次
```

3.4.3　pass 语句

在 Python 中还有一个 pass 语句，表示空语句。它不做任何事情，一般起到占位作用。

【例 3-28】 在应用 for 循环输出 10～20 (不包括 20)的偶数时，在不是偶数时，使用 pass 语句占个位置，方便以后对不是偶数的数进行处理。

```
for i in range(10,20):
    if i%2 ==0:
        print(i,end=' ')
    else:
```