



# Python编程案例教程

# 第3章 判断和循环语句

## 本章导读

在解决实际问题时，我们经常会遇到需要根据不同条件选择不同操作的情况，或者经常会遇到需要重复处理相同或相似操作的情况。Python提供了判断和循环语句用于解决这些问题。

本章首先介绍判断语句，包括简单的if语句、if-else语句、if-elif-else语句和嵌套的if语句，然后介绍while循环和for循环两种循环语句，以及跳出循环语句的方法，最后通过两个典型案例的分析和实现，让读者进一步掌握判断语句和循环语句的使用方法。



# 学习目标

- 掌握简单的if语句、if-else语句、if-elif-else语句和嵌套的if语句的使用方法
- 掌握while循环和for循环语句的使用方法
- 掌握break和continue、pass、else语句的使用方法
- 掌握选择结构程序设计和循环结构程序设计的编程思路

The background of the slide features an abstract design on the left side, composed of several overlapping, semi-transparent red polygons in various shades of red, creating a dynamic, layered effect. The right side of the slide is a solid white background.

**3.1 判断语句**

**3.2 循环语句**

**3.3 典型案例**

# 3.1 判断语句

- ◆ 3.1.1 简单的if语句
- ◆ 3.1.2 if-else语句
- ◆ 3.1.3 if-elif-else语句
- ◆ 3.1.4 嵌套的if语句

## 3.1.1 简单的if语句

if语句允许程序通过判断条件是否成立而选择是否执行指定的语句。

if 判断条件:

语句块



例如：

```
age = 20
```

```
if age >= 18:
```

```
    print("已成年")
```

```
#创建变量age代表年龄，赋值为20
```

```
#判断变量age的值是否大于等于18
```

```
#输出 "已成年"
```



## 3.1.2 if-else语句

if语句只允许在条件为真时指定要执行的语句，而if-else语句还可在条件为假时指定要执行的语句。

if 判断条件:

语句块1

else:

语句块2

例：编写程序，要求输入年龄，判断该学生是否成年（大于等于18岁），如未成年，计算还需要几年能够成年。

```
age = int(input("请输入学生的年龄："))
```

```
if age >= 18:
```

```
    print("已成年")
```

```
else:
```

```
    print("未成年")
```

```
    print("还差", 18 - age, "年成年")
```

#输入变量age的值并转换为整型

#判断age是否大于等于18

#如果是，输出“已成年”

#如果不是

#输出“未成年”

#计算还差几年成年并输出

程序运行效果

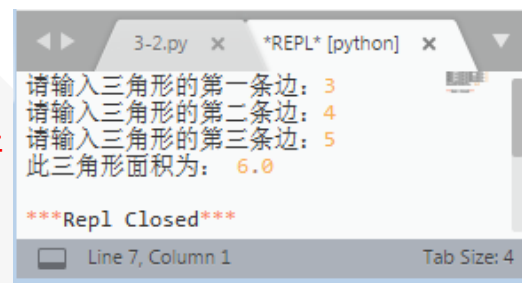


## 3.1.2 if-else语句

例：编写程序，要求输入三角形的三条边，判断是否是三角形。

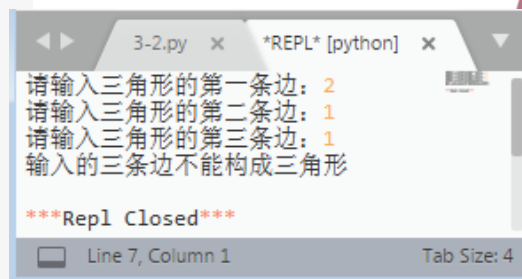
```
a=int(input("请输入三角形的第一条边："))    #输入第一条边并将其转换为整型
b=int(input("请输入三角形的第二条边："))    #输入第二条边并将其转换为整型
c=int(input("请输入三角形的第三条边："))    #输入第三条边并将其转换为整型
if a>0 and b>0 and c>0 and a+b>c and a+c>b and b+c>a:#如果满足构成三角形条件
    print("三边可以构成三角形" )            #输出三边可构成三角形
else:                                          #如不满足条件
    print("输入的三条边不能构成三角形");    #输出提示信息
```

程序运行效果



```
3-2.py x *REPL* [python] x
请输入三角形的第一条边： 3
请输入三角形的第二条边： 4
请输入三角形的第三条边： 5
此三角形面积为： 6.0

***Repl Closed***
Line 7, Column 1 Tab Size: 4
```



```
3-2.py x *REPL* [python] x
请输入三角形的第一条边： 2
请输入三角形的第二条边： 1
请输入三角形的第三条边： 1
输入的三条边不能构成三角形

***Repl Closed***
Line 7, Column 1 Tab Size: 4
```



### 3.1.3 if-elif-else语句

编程时常常需要判定一系列的条件，一旦其中某一个条件为真就立刻停止。

if 判断条件1:

    语句块1

elif 判断条件2:

    语句块2

...

elif 判断条件n:

    语句块n

else :

    语句块n+1



### 3.1.3 if-elif-else语句

例：学生成绩可分为百分制和五级制，将输入的百分制成绩score，转换成相应的五级制成绩后输出。

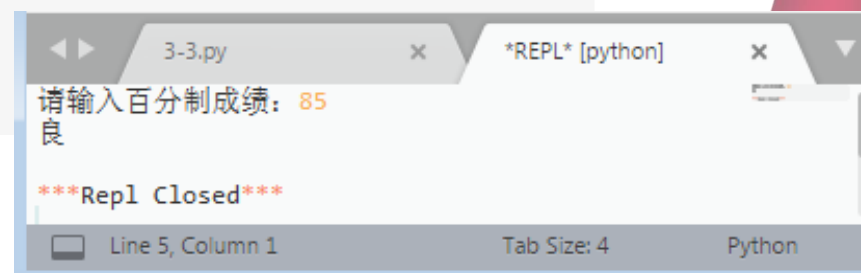
百分制	五级制	百分制	五级制
$90 \leq \text{score} \leq 100$	优	$60 \leq \text{score} < 70$	及格
$80 \leq \text{score} < 90$	良	$0 \leq \text{score} < 60$	不及格
$70 \leq \text{score} < 80$	中	$\text{score} > 100$ 或 $\text{score} < 0$	无意义



### 3.1.3 if-elif-else语句

```
score=int(input("请输入百分制成绩："))#输入分数score的值并将其转化为整数
if score>100 or score<0:                #当分值不合理时显示出错信息
    print("输入数据无意义")
elif score>=90:                          #当成绩大于等于90小于等于100时，输出“优”
    print("优")
elif score>=80:                          #当成绩大于等于80小于90时，输出“良”
    print("良")
elif score>=70:                          #当成绩大于等于70小于80时，输出“中”
    print("中")
elif score>=60:                          #当成绩大于等于60小于70时，输出“及格”
    print("及格")
else:                                    #以上条件都不满足
    print("不及格")                      #输出不及格
```

程序运行效果



### 3.1.4 嵌套的if语句

在if语句中又包含一个或多个if语句时，称为if语句的嵌套。

if 判断条件1:

if 判断条件2:

语句块1

else:

语句块2

else:

if 判断条件3:

语句块3

else:

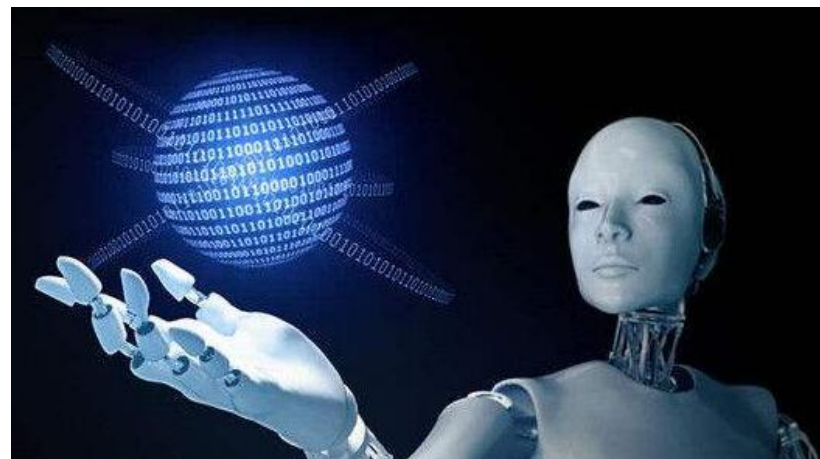
语句块4

} 内嵌if

} 内嵌if

提示

内嵌if可以是简单的if语句，也可以是if-else语句，还可以是if-elif-else语句。一定要注意if嵌套语句的逐层缩进，保持同级缩进相同。



### 3.1.4 嵌套的if语句

例：编写程序，实现输入三个整数，输出最大值。

```
a=int(input("请输入a的值:"))
```

```
b=int(input("请输入b的值:"))
```

```
c=int(input("请输入c的值:"))
```

```
if a>b:
```

```
    if a>c:
```

```
        max=a
```

```
    else:
```

```
        max=c
```

```
else:
```

```
    if b>c:
```

```
        max=b
```

```
    else:
```

```
        max=c
```

```
print("max=",max)
```

```
#输入a的值并转换为整数
```

```
#输入b的值并转换为整数
```

```
#输入c的值并转换为整数
```

```
#a>b
```

```
#a>b并且a>c，最大值为a
```

```
#a>b并且c>a，最大值为c
```

```
#a<b
```

```
#b>a并且b>c，最大值为b
```

```
#b>a并且c>b，最大值为c
```

```
#输出最大值max
```

程序运行效果



```
3-4.py x *REPL* [python] x
请输入a的值:6
请输入b的值:8
请输入c的值:1
max= 8

***Repl Closed***
Line 7, Column 1 Tab Size: 4 Python
```

## 3.2 循环语句

- ◆ 3.2.1 while循环语句
- ◆ 3.2.2 for循环语句
- ◆ 3.2.3 循环嵌套
- ◆ 3.2.4 break和continue语句

## 3.2.1 while循环语句

while循环语句的基本格式如下：

while 判断条件:

语句块

#循环体

### 提示

while循环语句是“先判断，后执行”。如果刚进入循环时条件就不满足，则循环体一次也不执行。还需要注意的是，一定要有语句修改判断条件，使其有为假的时候，否则将出现“死循环”。



### 3.2.1 while循环语句



例：编写程序，求 $S=1+2+3+\dots+100$ 的值。

注意

- (1) 变量初始化描述要完整、准确。
- (2) 在循环体中应有使循环趋向于结束的语句。

```
i=1                #创建变量i，赋值为1
S=0                #创建变量S，赋值为0
while i<=100:      #循环，当i>100时结束
    S+=i            #求和，将结果放入S中
    i+=1            #变量i加1
print("S=1+2+3+...+100=",S)  #输出S的值
```

程序运行效果

```
S=1+2+3+...+100= 5050
[Finished in 0.2s]
```



6 lines, 146 characters selected

Tab Size: 4



## 3.2.2 for循环语句

### ► 1 . for循环语句的语法结构

基本格式：

```
for 变量in 序列:  
    语句块
```



例如：

```
for x in "python":  
    print(x)
```

提示

Python中的for循环常用于遍历列表、元组、字符串以及字典等序列中的元素。

## 3.2.2 for循环语句

### ► 2 . for循环语句与range()函数

for循环语句经常与range()函数一起使用，range()函数是Python的内置函数，可创建一个整数列表。

range()函数的语法是：

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

步长，默认为1。

计数到stop结束，但不包括stop。

计数从start开始，默认是从0开始。



例如：

`range(5)`等价于`range(0,5)`

`range(0,5)`是`[0,1,2,3,4]`

`range(0,5)`等价于`range(0,5,1)`

## 3.2.2 for循环语句

例：用for语句求 $S=1+2+3+\dots+100$ 的值。

<code>S=0</code>	#创建变量S，赋值为0
<code>for i in range(1,101):</code>	#循环变量i从1循环到100
<code>S+=i</code>	#求和，将结果放入S中
<code>print("S=1+2+3+...+100=",S)</code>	#输出S的值

程序运行效果

```
S=1+2+3+...+100= 5050
[Finished in 0.2s]
```



6 lines, 146 characters selected

Tab Size: 4

### 3.2.3 循环嵌套

一个循环语句的循环体内包含另一个完整的循环结构，称为循环的嵌套。

- ◆ 嵌在循环体内的循环称为内循环。
- ◆ 嵌有内循环的循环称为外循环。
- ◆ 内嵌的循环中还可以嵌套循环，这就是多重循环。

两种循环语句while语句和for语句可以互相嵌套，自由组合。外层循环体中可以包含一个或多个内层循环结构。

#### 注意

各循环必须完整包含，相互之间不允许有交叉现象。

### 3.2.3 循环嵌套



例：编写一个程序，输出以下乘法表。

```
for x in range(1,10):           #循环变量x从1循环到9
    for y in range(1,x+1):       #循环变量y从1循环到x+1
        print(y,"*",x,"=",x*y,"",end="")    #输出乘法表达式
    print("")                   #输出空字符串，作用是为了换行
```

程序运行效果

```
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
[Finished in 0.2s]
```

## 3.2.4 其他语句

### ► 1. break语句

我们可以使用break语句跳出循环体，而去执行循环下面的语句。在循环结构中，break语句通常与if语句一起使用，以便在满足条件时跳出循环。

例：计算满足条件的最大整数n，使得 $1+2+3+\dots+n \leq 10000$ 。

```
n=1                #创建变量n，赋值为1
S=0                #创建变量S，赋值为0
while True:        #循环
    S+=n            #求和，将结果放入S中
    if S>10000:     #当S>10000时
        break       #跳出循环
    n+=1            #变量n加1
print("最大整数n为",n-1,"使得1+2+3+...+n<=10000。") #输出n-1的值
```

#### 程序运行效果

```
最大整数n为 140 ,使得1+2+3+...+n<=10000。
[Finished in 0.1s]
```

9 lines, 211 characters selected

Tab Size

## 3.2.4 其他语句

### ► 2 . continue语句

有时并不希望终止整个循环的操作，而只希望提前结束本次循环，接着执行下次循环，这时可以用continue语句。与break语句不同，continue语句的作用是结束本次循环，即跳过循环体中continue语句后面的语句，开始下一次循环。

例：输出1 ~ 20之间所有的奇数。

程序运行效果

```
for n in range(1,21):  
    if n%2==0:  
        continue  
    else:  
        print(n)
```

#循环，n的取值为1到20  
#判断n是否为偶数  
#当n为偶数时跳出本次循环  
#当n为奇数时输出n的值

```
1  
3  
5  
7  
9  
11  
13  
15  
17  
19  
[Finished in 0.2s]
```

6 lines, 132 characters selected

Tab Size: 4

## 3.2.4 其他语句

### ► 3 . pass语句

Python pass 是空语句，是为了保持程序结构的完整性。

pass 不做任何事情，一般用做占位语句。

例：输出1 ~ 20之间所有的奇数。

程序运行效果

```
# 输出 Python 的每个字母
for letter in 'Python':
    if letter == 'h':
        pass
    print '这是 pass 块'
print '当前字母:', letter

print "Good bye!"
```

```
当前字母 : P
当前字母 : y
当前字母 : t
这是 pass 块
当前字母 : h
当前字母 : o
当前字母 : n
Good bye!
[Finished in 0.3s]
```



## 3.2.4 其他语句

### ► 4 . else语句

除了判断语句，Python中的for循环和while循环也可以使用else语句。循环中的else语句是在循环完成之后执行的

例：输出1 ~ 20之间所有的奇数。

程序运行效果

```
# 使用else语句
for i in range(3):
    print("第",i+1,"循环")
else:
    print("循环结束了！")
```

```
第 1 循环
第 2 循环
第 3 循环
循环结束了！
[Finished in 0.2s]
```

The background of the slide features a complex, abstract design composed of various shades of red and pink. These colors are arranged in a series of overlapping, angular, and polygonal shapes that create a sense of depth and movement. The design is most prominent on the left side, where it transitions into a clean, white space on the right.

## 3.3 典型案例

### 3.3.1 猜拳游戏

例：编写程序，模仿猜拳游戏，要求输入两个用户的不同手型，判断输赢后输出。

```
player1=int(input("请用户1输入：0(剪刀) 1(石头) 2(布):"))#获取用户1输入的信息并赋值
player2=int(input("请用户2输入：0(剪刀) 1(石头) 2(布):")) #获取用户2输入的信息并赋值
if player1<0 or player1>2 or player2<0 or player2>2:    #输入了游戏规则以外的数字
    print("请遵守游戏规则")                             #输出 "请遵守游戏规则"
else:                                                     #输入正确
    if ((player1==0) and (player2==2)) or ((player1==1) and (player2==0)) or ((player1==2) and (player2==1)):
        #用户1所有能获胜的判断条件
        print("用户1获得胜利")                           #输出 "用户1获得胜利"
    elif player1==player2:                                #用户1输入与用户2相同
        print("平局，再来一局")                          #输出 "平局，再来一局"
    else:                                                  #用户2获胜
        print("用户2获得胜利")                            #输出 "用户2获得胜利"
```

程序运行效果



```
3-10.py x *REPL* [python] x
请用户1输入：0(剪刀) 1(石头) 2(布):0
请用户2输入：0(剪刀) 1(石头) 2(布):2
用户1获得胜利

***Repl Closed***
Line 6, Column 1 Tab Size: 4 Python
```

### 3.3.2 百钱买百鸡问题

中国古代数学家张丘建在他的《算经》中提出了一个著名的“百钱买百鸡问题”：鸡翁一，值钱五；鸡母一，值钱三；鸡雏三，值钱一；百钱买百鸡，问翁、母、雏各几何？编程实现将所有可方案输出在屏幕上。

```
for cock in range(0,20+1):           #鸡翁范围在0到20之间
    for hen in range(0,33+1):         #鸡母范围在0到33之间
        for biddy in range(3,99+1):  #鸡雏范围在3到99之间
            if (5*cock+3*hen+biddy/3)==100: #判断钱数是否等于100
                if (cock+hen+biddy)==100:  #判断购买的鸡数是否等于100
                    if biddy%3==0:         #判断鸡雏数是否能被3整除
                        print ("鸡翁:",cock,"鸡母:",hen,"鸡雏:",biddy)    #输出
```

程序运行效果

```
鸡翁：0 鸡母：25 鸡雏：75
鸡翁：4 鸡母：18 鸡雏：78
鸡翁：8 鸡母：11 鸡雏：81
鸡翁：12 鸡母：4 鸡雏：84
[Finished in 0.1s]
```

7 lines, 366 characters selected

Tab 5

**感谢您的观看**

