项目 3

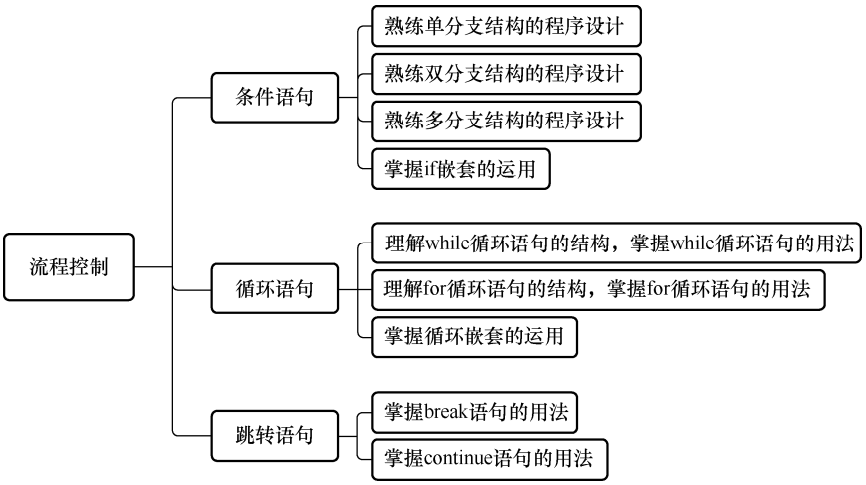
流程控制

项目描述

程序默认按自上而下的顺序执行，但可以使用特定语句更改程序的执行顺序，即实现程序的流程控制，它是构建程序逻辑的核心内容。可通过条件语句（如 if、elif、else）和循环语句（如 for、while），分别实现分支逻辑和循环操作，使程序能够根据不同的条件执行不同的代码块；可通过跳转语句控制跳出或中断循环，最终实现流程控制。

本项目通过条件语句、循环语句、跳转语句三个任务的学习和运用，着重提升学习者的逻辑思维能力、创新能力以及解决问题的能力，为后续项目的学习打下坚实的基础。

知识地图



任务 1　条件语句

| 任务引入 |

在日常的学习和生活中，人们常常会依据现有条件做选择，计算机程序也不例外。小王最近开始进入选择结构的学习，让程序根据不同情况做出不同反应，就像“如果……就……否则……”的思考方式。那么，Python 是如何让计算机程序具备这种“决策”能力的？如何根据条件结果执行对应的操作呢？

| 知识准备 |

条件语句是一种分支选择结构语句，它根据表达式的值的情况来选择执行哪些语句，选择结构根据给定的条件满足或不满足，分别执行不同的语句。Python 提供了实现选择结构的 if 语句，其中较常用的是 if-else 结构。Python 分支结构分为单分支结构、双分支结构、多分支结构。

一、单分支结构

if 语句是最简单的条件语句，语句的执行过程是：计算表达式的值，若值为 True，则执行语句块，然后执行 if 语句的后续语句；若值为 False，则跳出选择结构，继续向下执行 if 语句的后续语句（如图 3-1-1 所示）。

其语法格式如下：

```
if 表达式:
    语句块
```

说明：若表达式的值非 0，则执行 if 内的语句块，否则直接执行后面的语句。每个条件后面都要使用冒号（:），表示接下来的是满足条件后要执行的语句块。使用缩进来划分语句块，相同缩进量的语句在一起组成一个语句块。

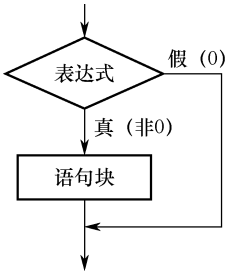


图 3-1-1　单分支结构执行过程

实例 3.1.1　排列数值

代码如下：

```
# 使用 if 结构排列 a 和 b 的值
# 定义变量
a = 20
b =10
print('a =',a,'b =',b)
if   a > b:
      t=a
      a=b
```

```
        b=t
print('按从小到大排列：',a,b)
```

运行结果如下：

```
a = 20    b = 10
按从小到大排列：10, 20
```

注意

在 Python 中要注意缩进量，条件语句根据缩进量来判断执行语句的归属。

实例 3.1.2　成绩判断

输入学生成绩，判断成绩是否及格。

代码如下：

```
# 使用 if 结构输入 s 成绩值进行判断
s=float(input('请输入数学成绩：'))
if (s<60):
    print('你的成绩未及格' )
print('谢谢使用')
```

运行结果 1 如下：

```
请输入数学成绩：56
你的成绩未及格
谢谢使用
```

运行结果 2 如下：

```
请输入数学成绩：80
谢谢使用
```


二、双分支结构

if-else 语句不仅需要处理满足条件的情况，也需要处理不满足条件的情况。语句的执行过程是：计算表达式的值，若值为 True，则执行语句块 1，否则执行 else 后面的语句块 2，语句块 1 或语句块 2 执行后再执行 if 语句的后续语句（如图 3-1-2 所示）。

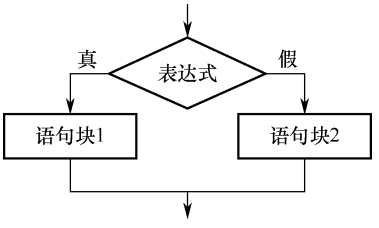


图 3-1-2　双分支结构执行过程

其语法格式如下：

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

说明：若表达式的值非 0，则执行语句块 1，否则执行语句块 2。

实例 3.1.3　判断密码是否正确

代码如下：

```
# 使用 if-else 结构判断密码是否正确
ma= input("请输入你的密码：")
if ma=='123456':
    print('你输入的密码正确')
```

```
else:
    print('你输入的密码错误')
```

运行结果如下：

```
请输入你的密码：123456
你输入的密码正确
```

在 Python 中，判断是否相等可以使用两个等号（==），单个等号则用于赋值。

实例 3.1.4　求三角形的面积

输入三角形的三条边的边长，求三角形的面积。

分析：设 a、b、c 表示三角形的三条边的边长，则构成三角形的充要条件是任意两条边之和大于第三条边，即 $a+b>c$，$a+c>b$，$b+c>a$。如果该条件满足，则可按照海伦公式计算三角形的面积：

$$s=\sqrt{p(p-a)(p-b)(p-c)}\,,\quad p=(a+b+c)/2$$

代码如下：

```
# 使用 if-else 结构判断测试结果
from math import sqrt
a,b,c=eval(input("a,b,c="))
if a+b>c and a+c>b and b+c>a:
    p=(a+b+c)/2
    s=sqrt(p*(p-a)*(p-b)*(p-c))
    print(f"a={a},b={b},c={c}")
    print(f"area={s}")
else:
    print(f"a={a},b={b},c={c}")
    print("input data error")
```

运行结果 1 如下：

```
a ,b,c=6,7,5
a=6,b=7,c=5
area=14.696938456699069
```

运行结果 2 如下：

```
a,b,c=3,4,5
a=3,b=4,c=5
area=6.0
```

实例 3.1.5　判断闰年

输入年份 year，判断是否是闰年，若是闰年则输出“x 年是闰年”，否则输出“x 年不是闰年”。

分析：年份只要满足下列两个条件之一，就是闰年。

① 普通闰年：年份是 4 的倍数但不是 100 的倍数。

② 世纪闰年：年份是 400 的倍数。

例如，1900 年、2021 年不是闰年，而 2000 年、2024 年是闰年。根据题目要求正好用双分支 if-else 语句实现。

代码如下：

```
# 使用 if-else 结构判断输入值 year 是不是闰年
x=int(input("请输入年份：")
if (x % 100 !=0 and   x%4==0)or (x % 400==0)：
    print("%d 年是闰年"%(x ))
else:
    print("%d 年不是闰年"%(x ))
```

运行结果 1 如下：

```
请输入年份：2021
2021 年不是闰年
```

运行结果 2 如下：

```
请输入年份：2024
2024 年是闰年
```

实例 3.1.6　判断奇偶数

输入一个整数，判断这个整数是奇数还是偶数，并输出判断结果。

分析：在整数中，能被 2 整除的数是偶数，不能被 2 整除的数是奇数。

代码如下：

```
# 使用 if-else 结构判断输入值的奇偶性
num =int (input("请输入一个整数："))
print('输入值：',num)
if num % 2 ==0 :
    print("%d 是偶数" % num)
else:
    print("%d 是奇数" % num )
```

运行结果如下：

```
请输入一个整数：10
10 是偶数
```

三、多分支结构

多分支结构语句 if-elif-else 是在双分支结构的基础上，对各种不同情况进行的进一步区分，elif 是 else-if 的缩写。根据问题的具体情况，可以有多个 elif。其语法格式如下：

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

提示

（1）程序执行了一个分支以后，其余分支不再执行。

（2）多个条件表达式同时满足条件，只执行第一个与之满足的语句块。

多分支结构语句执行过程如图 3-1-3 所示。

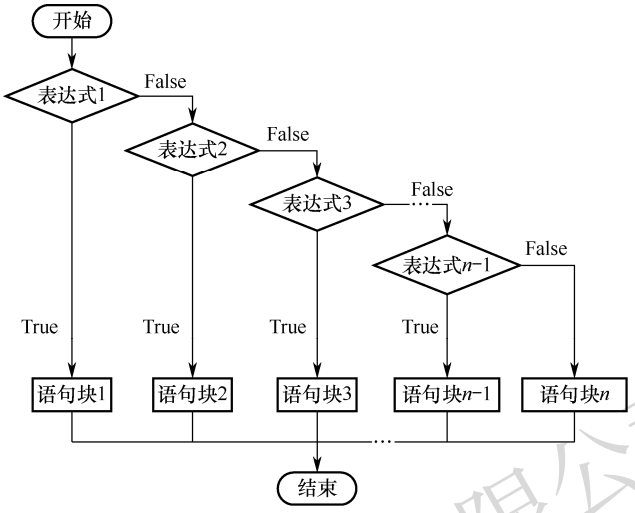


图 3-1-3　多分支结构语句执行过程

说明：程序运行后首先计算“表达式 1”的值是否为 True，若是则执行“语句块 1”；否则，继续计算“表达式 2”的值是否为 True，若是则执行“语句块 2”；依次类推，若所有表达式的值都为 False，则执行“语句块 *n*”。

实例 3.1.7　成绩等级分类

输入学生的成绩，根据成绩进行等级分类，90 分及以上为 A 等生，80～89 分为 B 等生，70～79 为 C 等生，60～69 为 D 等生，59 分及以下为 E 等生。

分析：将学生成绩分为 5 个分数段，然后根据各分数段的成绩，输出不同的等级，程序分 5 个分支，可以用多分支 if-elif-else 语句实现。

代码如下：

```
# 使用 if-elif-else 结构判断学生成绩等级
sc= int(input("请输入学生成绩："))
if sc >=90 and sc<=100:
   print('A 等生')
elif sc>=80 and sc<=89:
      print('B 等生')
elif sc>=70 and sc<=79:
      print('C 等生')
elif sc>=60 and sc<=69:
      print('D 等生')
else:
      print('E 等生')
```

运行结果 1 如下：

```
请输入学生成绩：45
E 等生
```

运行结果 2 如下：

```
请输入学生成绩：96
```

```
A 等生
```

实例 3.1.8　季节判断

通过键盘输入一个四季权限的级别整数（1～4），如果输入 1，则输出一个结果“春天”；如果输入 2，则输出一个结果“夏天”；如果输入 3，则输出一个结果“秋天”；如果输入 4，则输出一个结果“冬天”；如果输入其他数字，则输出“权限输入错误”。

代码如下：

```
# 使用 if-elif-else 结构判断输入值的范围
m= int(input("请输入一个数（1～4）："))
if m==1:
    print('春天')
elif m==2:
    print('夏天')
elif m==3:
    print('秋天')
elif m==4:
    print('冬天')
else:
    print('权限输入错误')
```

运行结果如下：

```
请输入一个数（1～4）：3
秋天
```

实例 3.1.9　求函数值

编写一个程序求分段函数 $f(x)=\begin{cases}3x+2, & x<-1\\ x, & -1\leqslant x\leqslant 1\\ 2x+3, & x>1\end{cases}$ 的值，并用它求出 $f(0)$ 的值。

代码如下：

```
# 使用 if-elif-else 结构求分段函数 f(x)的值
# 当 x<−1 时,f(x)=3x+2;
# 当−1≤x≤1 时，f(x)=x;
# 当 x>1 时，f(x)=2x+3;
x = 0   # 定义变量
if  x<−1:
    y=3*x+2
elif  −1<=x and x<=1:
    y=x
else:
    y=2*x+3
print('x =',x,'f(x) =',y)
```

运行结果如下：

```
x = 0 f(x) = 0
```

实例 3.1.10　商品促销

编写程序，当输入顾客实际购物金额时，计算并输出优惠价。某商场采用购物打折的方式促销，折扣表如表 3-1-1 所示。

表 3-1-1　折扣表

购 物 金 额	折　　扣	购 物 金 额	折　　扣
1000 元到 2000 元（不含）	九折	3000 元及以上	七折
2000 元到 3000 元（不含）	八折		

代码如下：

```
# 使用 if-elif-else 结构求分段函数的值
money = eval(input('请输入你的购物金额:'))
if (money >= 1000) and (money < 2000):
    level = 0.9
    cash = money * level
    print('折后金额为:' + str(cash))
elif (money >= 2000) and (money < 3000):
    level = 0.8
    cash = money * level
    print('折后金额为:' + str(cash))
else:
    level = 0.7
    cash = money * level
    print('折后金额为:' + str(cash))
```

运行结果 1 如下：

```
请输入你的购物金额:3000
折后金额为 2100
```

运行结果 2 如下：

```
请输入你的购物金额:2500
折后金额为 2000
```


四、分支结构的嵌套

if 结构嵌套是指 if 或 if-else 语句中包含 if 或 if-else 语句，在满足一个 if 条件之后，在它的语句块里对新的 if 语句再进行一次判断。使用时要注意不同级别代码块的缩进量，因为缩进量决定了代码块的从属关系。

前面介绍了 3 种形式的 if 选择结构，这 3 种结构还可以互相嵌套。

（1）在单分支（if）结构中嵌套二分支（if-else）结构。其语法格式如下：

```
if 表达式 1:
    语句块 1
    if 表达式 2:
        语句块 2
    else:
        语句块 3
```

（2）在嵌套语句中，可以把二分支（if-else）结构放在另一个二分支（if-else）结构中。其语法格式如下：

格式 1：

```
if 表达式 1:
    语句块 1
     if 表达式 2:
          语句块 2
     else:
          语句块 3
else:
     语句块 4
```

格式 2：

```
if 条件 1:
     语句块 1
else:
     if 条件 2：
       语句块 2
     else:
       语句块 3
```

（3）在嵌套语句中，可以把多分支（if-elif-else）结构放在另一个多分支（if-elif-else）结构中。其语法格式如下：

```
if 表达式 1:
    语句块 1
     if 表达式 2:
          语句块 2
     elif 表达式 3:
          语句块 3
     else:
          语句块 4
elif 表达式 4:
     语句块 5
else:
     语句块 6
```

（4）在嵌套语句中，可以把双分支（if-elif-else）结构放在另外两个分支（if-elif-else）结构中。其语法格式如下：

```
if 条件 1:
    if 条件 2：
      语句块 1
    else:
      语句块 2
else:
    if 条件 3：
      语句块 3
    else:
      语句块 4
```

实例 3.1.11　验证用户名和密码

编写程序，从键盘输入用户名和密码进行验证。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。

代码如下：

```
# 使用 if-elif-else 结构嵌套举例
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！")
```

运行结果如下：

```
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！
```

实例 3.1.12　找到最大值

输入 3 个整数，输出其中的最大值。

分析：3 个整数 a、b、c，如果 a>b 为真，则需要继续判断 a>c（这里使用了嵌套），进而确定最大值是 a 还是 c。如果 a>b 为假，则继续判断 b>c，若为真则最大值为 b，否则最大值为 c。

代码如下：

```
# 使用 if-elif-else 结构嵌套举例
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
elif b>c:
    max=b
else:
    max=c
print("最大值为: %d" % max)
```

运行结果如下:

```
请输入 a 的值:6
请输入 b 的值:8
请输入 c 的值:9
最大值为:9
```

五、综合实例

实例 3.1.13　判断三角形类型

已知三角形三条边的边长，判断三角形的类型。

一个三角形的三条边的边长为 a、b、c（a>0、b>0、c>0），且任意两条边的边长之和大于第三条边，若满足：

$b^2+c^2>a^2$，则这个三角形是锐角三角形；

$b^2+c^2=a^2$，则这个三角形是直角三角形；

$b^2+c^2<a^2$，则这个三角形是钝角三角形。

运行结果如下：

```
请输入三角形三条边的长度：3 4 5
这个三角形是直角三角形
请输入三角形三条边的长度：0 1 2
输入数据不正确，输入的边长应该大于 0
请输入三角形三条边的长度：1 2 6
输入数据不正确，需要重新输入
```

任务 2　循环语句

| 任务引入 |

在日常工作和学习中，我们经常会遇到一些重复性的事务，如每年四季更替、人们绕着操场跑圈、地球一直绕着太阳转等。程序开发同样会出现代码的重复执行。因此，小王开始学习循环结构的编程。那么，Python 程序中的循环语句有哪些？如何运用循环语句来解决实践问题？

| 知识准备 |

在利用 Python 进行数值实验或工程计算时，经常用到循环结构。在循环结构中，被重复执行的语句块称为循环体。常用的循环结构有两种：while 循环与 for 循环。下面分别简要介绍相应的用法。

循环是指多次重复执行某个语句块。循环有三个要素：循环变量、循环体和循环终止条件。循环结构的执行过程（如图 3-2-1 所示）：先判断表达式，若其逻辑值为真则执行语句块，然后继续判断……若表达式的逻辑值为假则退出循环。

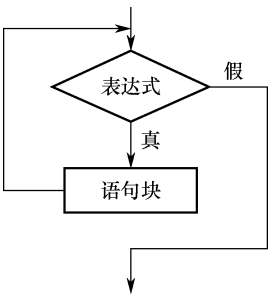


图 3-2-1　循环结构的执行过程

一、while 循环

若不知道所需要的循环到底要执行多少次，可以选择 while 循环，这种循环以 while 开头。其语法格式如下：

```
while　表达式:
可执行语句 1
…
可执行语句 n
```

while 语句中的表达式表示循环条件，可以是结果能解释为 True 或 False 的任何表达式，常用的是关系表达式和逻辑表达式。表达式后面必须加冒号，语句块是重复执行的部分，由多条语句组成，称作循环体。

其中，表达式即循环控制语句，一般是由逻辑运算或关系运算及一般运算组成的表达式。若表达式的值非 0，则执行一次循环，否则停止循环。这种循环方式在编写数值算法时用得非常多。一般来说，能用 for 循环实现的程序也能用 while 循环实现。还有一种带 else 的格式，其一般语法格式如下：

```
while 条件表达式:
    语句块 1
[else:
    语句块 2]
```

说明：else 子句是可选的，较少使用。

当表达式的值为真时，执行相应的语句块（循环体），然后判断表达式的值，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，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。

实例 3.2.1　实现累加

利用 while 语句实现 1～100 的累加。

代码如下：

```
s = 0
i = 0
while i <= 100:
    s= s+i
    i = i + 1
print('1～100 的和为:',s)
```

运行结果如下：

```
1～100 的和为:5050
```

实例 3.2.2　求和

利用 while 语句求 10 以内奇数的和。

代码如下：

```
# 求 10 以内奇数的和
s = 0
i = 1
while i <= 10:   # 冒号
    s = s + i
    i=i+2
print("sum =", s)              # 此语句应在 while 循环外面
```

运行结果如下：

```
sum =25
```

实例 3.2.3　累加求和

利用 while 语句实现 s=1*2+2*3+…+n*(n+1)累加求和。

代码如下：

```
# 求两数乘积之和
n =int(input("请输入一个整数："))
s = 0
i = 1
while i <= n :   # 冒号
    s = s + i*(i+1)
    i=i+1
print("sum =", s)              # 此语句应在 while 循环外面
```

运行结果如下：

```
请输入一个整数：3
sum =20
```

二、for 循环

Python 还提供了另一种循环结构：for 循环。它的形式比 while 循环更简捷，不需要初始赋值，不改变循环变量，降低了编程出错的可能性，而且运行效率更高，只是灵活性稍差。for 循环结构的执行过程如图 3-2-2 所示。在 for 循环中，循环次数一般情况下是已知的，除非用其他语句提前终止循环。这种循环以 for 开头。其语法格式如下：

```
for <target_list> in <expression_list>:
    <statement1>
    <statement2>
    <statement3>
else:
    <statementn>
```

其中，target_list 是循环变量，在每次循环中，用于接收存储从<expression_list>中获取的元素值，expression_list 是一个迭代对象，可以是元组、字符串、字典、集合等；缩进的语句都是循环体。变量每取一次值，循环便执行一次，直到迭代对象的最后一项。迭代变量<target_list>无特殊意义，一般用 i 表示。

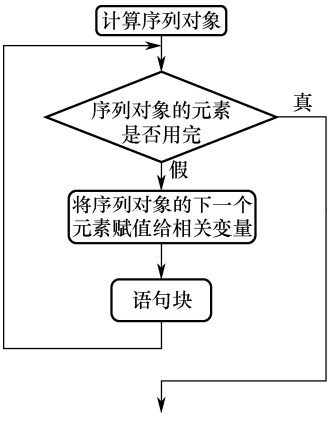


图 3-2-2　for 循环结构的执行过程

实例 3.2.4　累加

利用 for 语句实现 1～100 的累加。

代码如下：

```
sum = 0
# 生成 1～100 的数字序列
for i in range(1,101,1):
    sum = i + sum
print('1～100  的和为',sum)
```

运行结果如下：

```
1～100 的和为 5050
```

在上面的实例中，range 表示 i 从 1 开始迭代，每次加 1，直到 i 变成 100，迭代结束，因此当 i=101 时不执行语句，for 循环是 100 次迭代。

实例 3.2.5　输出元素

利用 for 语句输出列表中的元素。

代码如下：

```
list = {1,2,3,4,5,6,7,8}      # 定义列表 list
for i in list:      # 迭代对象是列表
    print(i,end=' ')
```

运行结果如下：

```
1 2 3 4 5 6 7 8
```

实例 3.2.6　输出数据

利用 for 语句输出元组中的数据。

代码如下：

```
for x in (4,6,8,10,12,14,16):      # 迭代对象是元组
    print(x,end=" ")
```

运行结果如下：

```
4 6 8 10 12 14 16
```

实例 3.2.7　输出元素

利用 for 循环中的 else 子句输出列表中的元素。

代码如下：

```
for i in range(1,15):      # 生成 1 到 14 的数字序列
    print(i ,end=" ")
else:
    print(i)
```

如果没有 else 子句，会输出 1～14。增加 else 子句后输出结果如下：

```
1 2 3 4 5 6 7 8 9 10 11 12 13 14 14
```

注意，最后一个数字“14”输出了两次，第二次是由 else 子句输出的。这个例子清晰地表明了 for 循环中的 else 子句是“最后一次循环终止时”被运行的，这一点与 while 循环中 else 子句的运行情况是不同的。

实例 3.2.8　求阶乘

利用 for 循环求 $n!$，即 $1\times2\times3\times4\times\cdots\times n$。

代码如下：

```
n =int(input("请输入一个整数："))
s=1
for i in range(1,n+1):      # 生成 1 到 n 的数字序列
    s=s*i
print("%d!=%d"%(n,s))
```

运行结果如下：

```
请输入一个整数：5
5!=120
```

实例 3.2.9　水仙花数

利用 for 循环输出所有的“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数本身，例如，153 是“水仙花数”，153 = 1×1×1 + 5×5×5 + 3×3×3。

代码如下：

```
for i in range(100,1000):
a=i%10
b=i//10%10
c=i//100
if a**3+b**3+c**3==i:
    print("%d 是水仙花数"%i)
```

运行结果如下：

```
153 是水仙花数
370 是水仙花数
371 是水仙花数
407 是水仙花数
```

三、循环嵌套

Python 允许在一个循环中嵌入另一个循环，成为循环嵌套。例如，while 与 while、for 与 for、while 与 for 构成循环嵌套。

1．while-while 格式

语法格式如下：

```
#while 循环语句中嵌套 while 循环语句
while  循环条件 1:                    # 外层循环
    语句块 1
    while  循环条件 2:                # 内层循环
        语句块 2
```

说明：以 while 循环语句中嵌套 while 循环语句的循环嵌套为例，如果外层的循环条件 1 为 True，则执行语句块 1，并对内层的循环条件 2 进行判断；如果循环条件 2 也为 True，则执行语句块 2，直到循环条件 2 为 False 为止；内层循环结束后继续判断外层的循环条件 1，如此重复执行直到循环条件 1 为 False 时，结束外层循环。

实例 3.2.10　利用 while 循环嵌套打印图形

代码如下：

```
i=1
while i<5:
 j=0
    while j<i:
        print("*",end="")
        j+=1
    print()
```

```
        i=i+1
```

运行结果如下：

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

2．for-for 格式

语法格式如下：

```
#for 循环语句中嵌套 for 循环语句
for 临时变量 1 in 可迭代对象 1:              # 外层循环
    语句块 1
    for 临时变量 2 in 可迭代对象 2:          # 内层循环
        语句块 2
```

说明：以 for 循环语句中嵌套 for 循环语句的循环嵌套为例，首先访问外层循环中可迭代对象 1 的第一个元素，执行语句块 1，访问内层循环中可迭代对象 2 的第一个元素，执行语句块 2；然后访问内层循环中可迭代对象 2 的下一个元素，执行语句块 2，如此重复执行，直至内层循环中可迭代对象 2 的所有元素都被访问完，结束内层循环；继续访问外层循环中可迭代对象 1 的下一个元素，所有外层循环中的元素都被访问完毕，结束外层循环。

实例 3.2.11　利用 for 循环嵌套打印图形

代码如下：

```
for i in range(1,7):
    for j in range(0,i):
print("*",end="")
    print()
```

运行结果如下：

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

3．for-while/while-for 格式

语法格式如下：

```
#for 循环语句中嵌套 while 循环语句
for 临时变量 in 可迭代对象:                  # 外层循环
    语句块 1
    while 循环条件:                          # 内层循环
        语句块 2
# while 循环语句中嵌套 for 循环语句
while 循环条件 1:                            # 外层循环
    语句块 1
    for 临时变量 in 可迭代对象:              # 内层循环
        语句块 2
```

实例 3.2.12　利用 for 与 while 循环嵌套打印图形

代码如下：

```
for i in range(1,5):
j=0
    while j<i:
        print("*",end="")
        j+=1
    print()
```

运行结果如下：

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

四、综合实例

实例 3.2.13　输出九九乘法表

九九乘法表起源于中国，其历史可追溯到先秦时期。当时，《荀子》《管子》等古籍中已出现部分乘法口诀内容。2002 年出土的湘西里耶秦简及湖北荆州秦家咀墓地出土的战国楚简《九九术》，证实战国秦代时九九乘法表已通行。

要求：编写 Python 程序，利用循环结构实现输出九九乘法表。

运行结果参考如下：

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

任务 3　跳转语句

| 任务引入 |

小王在学习循环语句时，发现满足条件的情况下程序会一直执行，但有时候需要跳出循环，以提高程序的执行效率。为了让程序学会灵活“转弯”，小王想学习如何改变执行路线。那么，Python 程序的跳转语句有哪些？如何使用跳转语句？

| 知识准备 |

在程序开发的过程中，有时候需要终止某次循环而不执行本次循环的后续代码，而有时候则需要结束整个循环而继续执行后面的代码。Python 程序中要实现类似这样的跳转需要使用 break、continue 等跳转语句。

一、break 语句

break 语句用于结束当前正在执行的循环（for、while)，转而执行这些循环后面的语句。break 语句常与 if 语句搭配使用，表示在某种条件下跳出循环。break 语句的语法格式如下：

```
While |for  循环:
        if 条件表达式:
              break
```

说明：通常 break 语句和 if 语句连用，用于满足某条件时，执行到 break 语句，结束循环。

实例 3.3.1　for 与 break 结合使用

代码如下：

```
for i in range(1,20):
    print("*")
    if(i==5):
            print("循环结束")
            break
```

运行结果如下：

```
*
*
*
*
*
循环结束
```

实例 3.3.2　依次输出 1，2，3，…，*n*

代码如下：

```
n =int(input("请输入一个整数："))
i=1
while   True:              #这是一个无限循环
print(i,end=" ")
    i +=1
if   i>n:     #满足终止条件
    break   #提前终止循环
```

说明：其实这个问题并不需要用 break 语句来解决，这么写只是为了让读者理解 break 语句的作用。

运行结果如下：

```
请输入一个整数：5
1 2 3 4 5
```

二、continue 语句

在 for 循环和 while 循环中，还可以使用另一种流程控制语句：continue 语句。它的作用是终止当前这次循环，立即进入下一次循环。一般情况下，continue 语句会出现在 if 语句中，常见的语法格式如下：

```
while|for 循环语句:
if 条件表达式:
    continue
```

实例 3.3.3　过滤字符

编写程序，从键盘输入一段字符串，如果其中包括“a”字符（可能出现 0 次、1 次或者多次），则输出时过滤掉该字符，其他字符内容原样输出。

代码如下：

```
str=input("请输入一字符串：  ")
for   word in str:
   if   word=="a ":      #满足终止条件
        continue          #跳出本次循环
   print( word, end=" ")
```

运行结果如下：

```
请输入一字符串：bbauilat
b b u i l t
```

说明：continue 语句可用来结束本次循环，紧接着执行下一次循环。和 break 语句相比，continue 语句没有跳出循环，它只会终止执行本次循环后面的代码，直接继续执行一次循环。

实例 3.3.4　输出奇数

编写一程序，输出 1 到 10 之间的奇数。

代码如下：

```
j=0
for num in range(1,11):
        if num % 2==0:
            continue
        print(num,end=" ")
```

运行结果如下：

```
1 3 5 7 9
```

三、综合实例

实例 3.3.5　猜数字

一个简单的猜数字游戏，程序会生成一个随机数，玩家需要猜测这个数字。如果玩家

输入负数，程序会提示并跳过。

具体要求如下：

（1）游戏开始，程序随机生成一个 1～100 之间的整数作为目标数字；

（2）玩家输入“0”，则结束猜测，同时公布正确答案；

（3）玩家输入负整数，则忽略并提示；

（4）玩家输入非数值，则提示有误（如“请输入有效数值”）；

（5）玩家输入任意正整数后，程序给出反馈：猜大了或猜小了。

运行结果参考如下：

```
欢迎来到猜数字游戏！
我已经想好了一个 1 到 100 之间的数字，你能猜出来吗？
请输入你的猜测（输入 0 退出）：50
太大了！再试一次。
请输入你的猜测（输入 0 退出）：40
太大了！再试一次。
请输入你的猜测（输入 0 退出）：30
太大了！再试一次。
请输入你的猜测（输入 0 退出）：20
太小了！再试一次。
请输入你的猜测（输入 0 退出）：25
太大了！再试一次。
请输入你的猜测（输入 0 退出）：23
```


项目实战

实战 1：剪刀石头布游戏

剪刀石头布游戏又称“猜丁壳”“猜拳”，是比较古老的游戏，游戏的起源可追溯到汉朝的手势令。在中国很小的孩子都会玩这个游戏，它的规则很简单：石头克剪刀，剪刀克布，布克石头。这个游戏的主要目的是解决争议，因为三者相互制约，因此不论平局几次，总会有胜负的时候。

运行结果如下：

```
--欢迎光临“剪刀石头布”游戏---
请输入您的选择(0 退出 1 玩游戏):6
您输入的内容不合法
请输入您的选择(0 退出 1 玩游戏):1
请输入(0 剪刀 1 石头 2 布):7
您输入的内容不合法
请输入(0 剪刀 1 石头 2 布):0
您出的是剪刀
电脑出的是剪刀

平局
------欢迎光临“剪刀石头布”游戏-----
```

```
请输入您的选择(0 退出 1 玩游戏):1
请输入(0 剪刀 1 石头 2 布):2
您出的是布
电脑出的是石头
恭喜恭喜，您赢了!!
-------欢迎光临“剪刀石头布”游戏------请输入您的选择(0 退出 1 玩游戏):0
**************谢谢使用**************
```

设计一个剪刀石头布游戏，游戏规则如下。

（1）玩家出剪刀或石头或布。

（2）电脑随机输出一个剪刀或石头或布。

（3）将玩家出的结果和电脑出的结果进行比对。

实战 2：百钱买百鸡

《张丘建算经》中有这样一道著名的百鸡问题：“鸡翁一，值钱五；鸡母一，值钱三；鸡雏三，值钱一。百钱买百鸡，问鸡翁、母、雏各几何？”其意为：公鸡每只 5 元，母鸡每只 3 元，小鸡三只 1 元。用 100 元买 100 只鸡，问公鸡、母鸡和小鸡各能买多少只?

数学分析：

假设：公鸡数量为 x，母鸡数量为 y，小鸡数量为 z。

根据题意得到以下方程：

$x + y + z = 100$

$5x + 3y + z/3 = 100$

算法分析：

采用穷举法求解，100 元买公鸡最多可买 100//5=20 只，买母鸡最多可买 100//3=33 只，所以，x 从 0 变化到 20，y 从 0 变化到 33，则 $z = 100 - x - y$，但必须是 3 的倍数。只要判断第二个条件是否满足即可。

运行结果如下：

```
公鸡: 0  母鸡: 25  小鸡: 75
公鸡: 4  母鸡: 18  小鸡: 78
公鸡: 8  母鸡: 11  小鸡: 81
公鸡: 12  母鸡: 4  小鸡: 84
```

实施步骤：

（1）根据分析写出内外循环语句循环次数。

（2）使用嵌套 for 循环调用指定行与列。

（3）使用 if 语句进行控制并输出结果。