Python程序设计

第四讲 程序流程控制
循环结构

张 华
WHU

第四讲 程序流程控制—循环结构

- while语句
- for语句
- range对象
- break语句
- continue语句
- 程序设计举例

引例

引例：购物

…

while *购物单上还有商品*

购买一件商品，并且把它从清单上划掉

…

- 如果条件“购物单上还有商品”成立时，就执行动作“购买一件商品，并且把它从清单上划掉”；
- 如果该条件一直成立，这个动作就会重复执行；
- 最终（该买的商品都买到），即该条件不成立，循环过程就会终止，程序将执行这个循环结构之后的第一条语句。

两种循环语句

- Python主要有for循环和while循环两种形式的循环结构。
  - while循环一般用于循环次数难以提前确定的情况，当然也可以用于循环次数确定的情况。
  - for循环一般用于循环次数可以提前确定的情况，尤其适用于枚举或遍历序列或迭代对象中元素的场合。
- 多个循环可以嵌套使用，并且还经常和选择结构嵌套使用来实现复杂的业务逻辑。

while循环

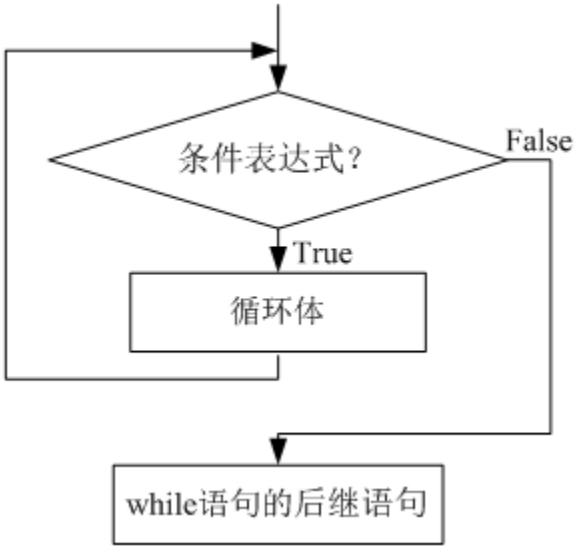
while循环结构的完整语法形式

while 条件表达式:

循环体

[else:

else子句代码块]



- 注意：
  - 如果循环因为条件表达式不成立或序列遍历结束而自然结束时则执行**else**结构中的语句；
  - 如果循环是因为执行了**break**语句而导致循环提前结束则不会执行**else**中的语句。

while循环

案例：计数器控制的确定循环

- 编写程序，输入某班10个学生某次考试的成绩，输出平均分。

```
sum = 0.0

i=0
while i<10:
    score = float(input('Input score:'))
    sum += score
    i += 1

print('average is', sum/i)
```

```
Input score:1
Input score:2
Input score:3
Input score:4
Input score:5
Input score:6
Input score:7
Input score:8
Input score:9
Input score:10
average is 5.5
```

while循环

案例：标记控制的不确定循环

- 编写程序，输入某班若干个学生某次考试的成绩，输出平均分。

定义标记值：输入成绩时输入无效值表示结束输入

```
sum = 0.0
i=0

score = float(input('Input score (invalid one to end):'))

while 0<=score<=100:
    sum += score
    i += 1
    score = float(input('Input score (invalid one to end):'))

if i>0:
    print('average is', sum/i)
else:
    print('no valid score')
```

while循环

练习：取款机密码验证

- 在取款机上取款时需要输入6位银行卡密码。下面模拟一个简单的取款机（密码只有1位数字），
  - 每次要求用户输入1位数字密码，密码正确输出“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5次后输出“密码输入错误，您还可以输入1次密码”，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”。

while循环

取款机密码验证的参考实现

```
password = 8   #预设密码
trymax = 6
trycount = 0
while trycount<6:
    inpwd = int(input('请输入1位数密码：')
    )if inpwd == password:
        trycount=99
        print('密码输入正确，正进入系统...')
    else:
        trycount+=1
        if trymax-trycount==1:
            print('密码输入错误，您还可以输入1次密码')
        elif trycount<trymax:
            print('密码输入错误，您已经输入{}次'.format(trycount))
        else:
            print('您的卡将被锁死，请和发卡行联系')
```

for循环

for循环结构的完整语法形式

for 取值 in 序列或迭代对象:

    循环体

[else:

    else子句代码块]

- 用来遍历序列或可迭代对象集合中的元素。
- 序列或可迭代对象
  - 序列：列表、字符串、元组等
  - 字典、集合
  - 文件
  - 迭代器：range、enumerate、zip、map等
  - 生成器函数

for循环

案例

- 使用循环结构遍历并输出列表中的所有元素。

```
a_list = ['a', 'b', 'mpilgrim', 'z', 'example']
i=0
for v in a_list:
    print('列表的第', i+1, '个元素是：', v)
    i+=1
```

```
a_list = ['a', 'b', 'mpilgrim', 'z', 'example']
for i, v in enumerate(a_list):
    print('列表的第', i+1, '个元素是：', v)
```

第二种方法：把下标和值一起读出来了
enumerate() 函数用于将一个可遍历的数据对象(如列表、元组或字符串)组合为一个索引序列，同时列出数据和数据下标，一般用在 for 循环当中。

range对象

range对象

- range()是Python的一个内置函数，返回一个迭代器对象，语法格式为

  range([start,] end [, step])

  - 有range(stop)、range(start, stop)和range(start, stop, step)三种用法。
  - 该函数返回具有惰性求值特点的range对象，其中包含左闭右开区间[start,end)内以step为步长的整数。
  - 参数start默认为0，step默认为1。

```
>>> range(5)                          #start默认为0，step默认为1
range(0, 5)
>>> list(_)
[0, 1, 2, 3, 4]
>>> list(range(1, 10, 2))             #指定起始值和步长
[1, 3, 5, 7, 9]
>>> list(range(9, 0, -2))             #步长为负数时，start应比end大
[9, 7, 5, 3, 1]
```

程序设计案例

问题

- 输出1~100之间能被7整除但不能同时被5整除的所有整数。

```
for i in range(1, 101):
    if i%7==0 and i%5!=0:
        print(i)
```

➢ 或者

```
values = [i for i in range(1,101) if i%7==0 and i%5!=0]
for i in values:
    print(i)
```

程序设计案例

问题

- 使用嵌套的循环结构打印九九乘法表。

```
for i in range(1, 10):
    for j in range(1, i+1):
        print('{0}*{1}={2}'.format(i,j,i*j), end='   ')
    print() #打印空行
```

```
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
```

程序设计案例

问题

- 计算1+2+3+…+99+100的结果。

```
s = 0
for i in range(1, 101):                    #不包括101
    s += i
else:
    print(s)
```

- 或者

```
>>> sum(range(1,101))
```

程序设计案例

问题

- 编写代码，输出由星号*组成的菱形图案，并且可以灵活控制图案的大小。

```
def main(n):
    for i in range(n):
        print((' * '*i).center(n*3))
    for i in range(n, 0, -1):
        print((' * '*i).center(n*3))

main(6)
main(10)
```

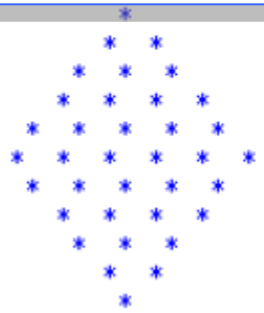


图 4-5　n=6 的运行效果

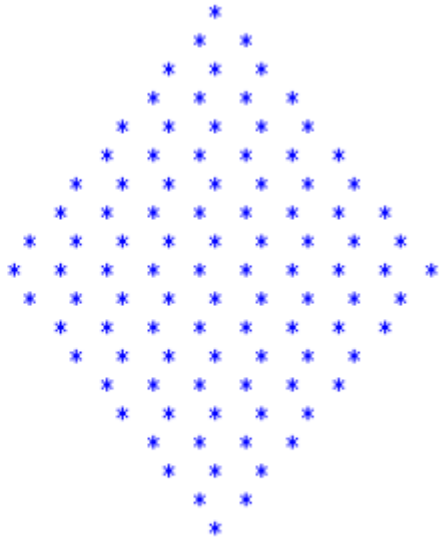


图 4-6　n=10 的运行效果

break与continue语句

跳转语句

break

continue

使用说明：

- 用在for循环和while循环内部。
- 一旦break语句被执行，将使得break语句所属层次的循环提前结束。
- 执行break语句后，不会执行for和while的else子句。
- continue语句的作用是提前结束本次循环，忽略continue之后的所有语句，提前进入下一次循环。

break与continue语句

案例

- 用枚举法判断一个大于1的自然数是否是素数。

```
n = int(input('input an integer (>1):'))

if n>1:
    for i in range(2, n):
        if n%i == 0:
            print('is not a prime number.')
            break
    else:
        print('is a prime number.')
else:
    print('invalid input')
```

break与continue语句

案例

- 输出小于100的最大素数。

```
for n in range(100, 1, -1):
    if n%2 == 0:
        continue

    for i in range(3, int(n**0.5)+1, 2):
        if n%i == 0:
            #结束内循环
            break
    else:
        print(n)
        #结束外循环
        break
```

程序设计案例（迭代法）

问题

- 用如下近似公式求自然对数的底数e的值，直到最后一项的绝对值小于10^{-6}为止

$$e \approx 1 + \frac{1}{1!} + \frac{1}{2!} + \cdots\cdots + \frac{1}{n!}$$

```
e = ai = i = 1
while ai>=1e-6:
    e += ai     # e是迭代变量

    i += 1
    ai /= i     # ai也是迭代变量

print('e =', e)
```

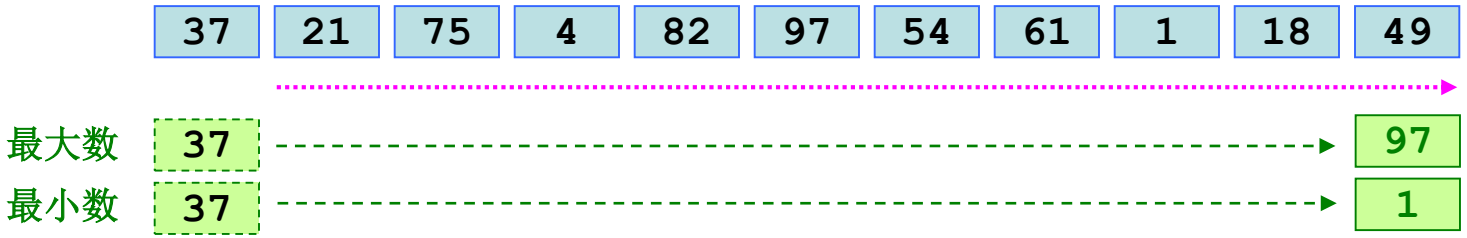
注意符号变量的运用

程序设计案例

问题

- 任意输入10个数，找出最大数和最小数

分析与设计



- 数的范围无法确定
- 第一个数作为（此时的）最大数和最小数
- 将其余的数与最大数、最小数依次进行比较
- 每次根据比较的结果更新（此时的）最大数和最小数
- 用计数器控制的循环实现

程序设计案例

找最大数（续）

```
maxValue = eval(input('输入一个整数：'))
for i in range(1, 10):
    value = eval(input('再输入一个整数：'))
    if maxValue<value:
        maxValue = value

print('the max value is', maxValue)
```

或者

```
values = []
for i in range(10):
    value = eval(input('输入一个整数：'))
    values.append(value)

print('the max value is', max(values))
```

小结

- 当某些操作需要重复执行时，使用循环结构。
- 构造循环结构的流程控制语句
  - while
  - for
- 在循环语句中可以使用的跳转语句
  - break
  - continue