第五章: 循环

胡兆龙

E-mail:huzhaolong@zjnu.edu.cn

基本框架

- while 循环
- 循环设计策略
- 利用哨兵值控制循环
- for循环
- 嵌套循环
- 最小化数值误差
- break 和 continue

简介

打印字符串 **Programming is fun!** 100 次.

100 times
```
print("Programming is fun!")
print("Programming is fun!")
...
print("Programming is fun!")
```

```
count = 0
while count < 100:
    print("Programming is fun!")
    count = count + 1
```

条件控制循环，
一个表达式

```
for i in range(100):
    print('Prgoramming is fun')
```

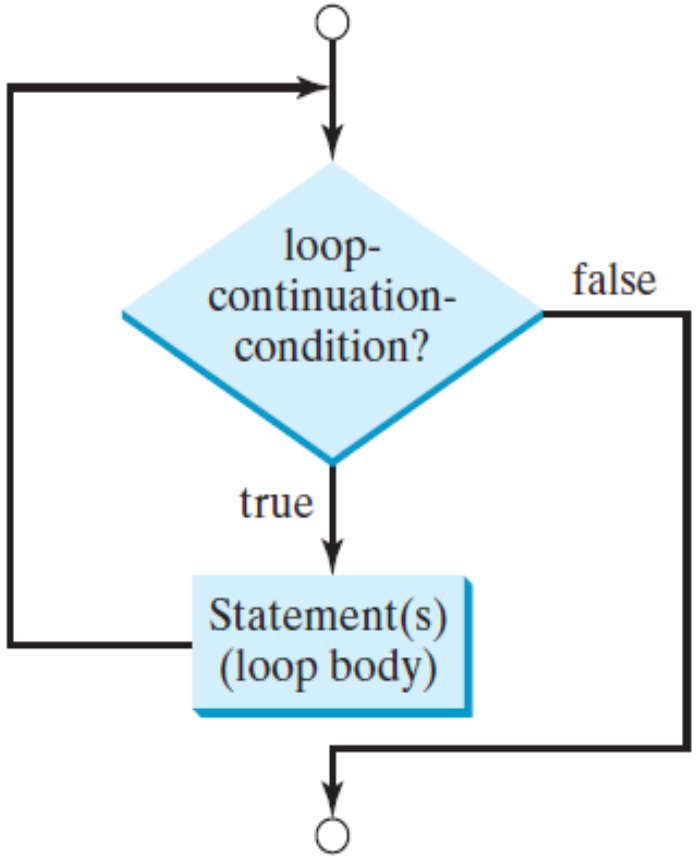
计数器控制循环，
一般知道迭代次数

while 循环

The syntax for the **while** loop is:

```
while loop-continuation-condition:
    # Loop body
    Statement(s)
```

推荐使用一个tab键
或者4个空格键



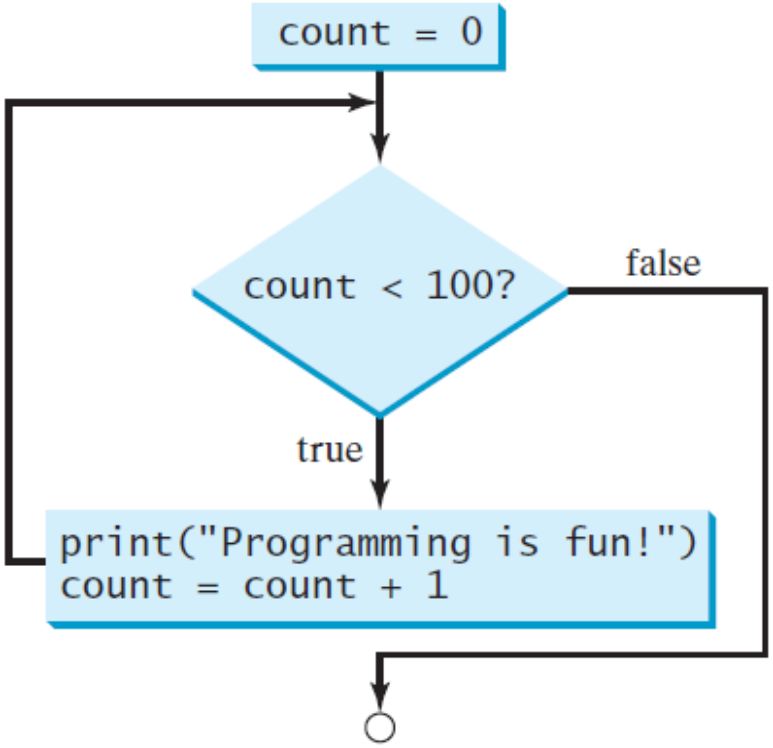
(a) A while loop

while 循环

loop-continuation-condition

```
count = 0
while count < 100:
    print("Programming is fun!")
    count = count + 1
```

loop body



while 循环

```
sum = 0
i = 1
while i < 10:
    sum = sum + i
    i = i + 1
print("sum is", sum)
```

i = 9

i = 10

sum = 1 + 2 + … + 9

```
sum = 0
i = 1
while i < 10:
    sum = sum + i
    i = i + 1
```

条件始终满足

无限循环（死循环），用 ctrl + c 跳出循环。

while 循环

```
count = 0
while count <= 100:
    print("Programming is fun!")
    count = count + 1
```

迭代了 101 次,被称为:偏离 1 的误差。

结果是什么?

```
count = 0
while (count < 9):
    print('The count is:', count)
    count = count + 1
```

while 循环

```
i = 1
while i < 10:
    if i % 2 == 0:
        print(i)
```

(a) 死循环

这些循环体将会循环多少次？每个程序的结果是什么？

```
i = 1
while i < 10:
    if i % 2 == 0:
        print(i)
        i += 1
```

(b) 死循环

```
i = 1
while i < 10:
    if i % 2 == 0:
        print(i)
    i += 1
```

2
4
6
8

(c)

循环设计策略

Step 1: 找出需要被循环的语句

Step 2: 将循环语句写成如下形式：

while True:

    语句（statements）

Step 3: 编写循环条件，并给出控制循环的语句，如

while 循环条件:

    语句（statements）

    控制语句

例子：猜一个数

示例：

```
Guess a magic number between 0 and 100
Enter your guess: 50  ↵Enter
Your guess is too high
Enter your guess: 25  ↵Enter
Your guess is too low
Enter your guess: 42  ↵Enter
Your guess is too high
Enter your guess: 39  ↵Enter
Yes, the number is 39
```

例子：猜数

```
import random
number = random.randint(0, 100)
print("Guess a magic number between 0 and 100")
# Prompt the user to guess the number
guess = eval(input("Enter your guess: "))
```

```
while True:
    # Prompt the user to guess the number
    guess = eval(input("Enter your guess: "))

    if guess == number:
        print("Yes, the number is", number)
    elif guess > number:
        print("Your guess is too high")
    else:
        print("Your guess is too low")
```

例子：猜数

LISTING 5.3 GuessNumber.py

```
import random
number = random.randint(0, 100)
print("Guess a magic number between 0 and 100")
guess = -1
while guess != number:
    # Prompt the user to guess the number
    guess = eval(input("Enter your guess: "))
    if guess == number:
        print("Yes, the number is", number)
    elif guess > number:
        print("Your guess is too high")
    else:
        print("Your guess is too low")
```

例子：多道减法题测试

Step 1: Generate two single-digit integers for `number1` and `number2`.

Step 2: If `number1 < number2`, swap `number1` with `number2`.

Step 3: Prompt the student to answer, "What is number1 – number2?"

Step 4: Check the student's answer and display whether the answer is correct.

写一个程序：生成五道题，学生回答所有问题后，记录正确答案的个数。

例子：多道减法题测试

```
import random
import time

correctCount = 0  # Count the number of correct answers
count = 0   # Count the number of questions
NUMBER_OF_QUESTIONS = 5  # Constant

startTime = time.time() # Get start time

while count < NUMBER_OF_QUESTIONS:
    # Generate two random single-digit integers
    number1 = random.randint(0, 9)
    number2 = random.randint(0, 9)
```

```
15      # If number1 < number2, swap number1 with
16      if number1 < number2:
17          number1, number2 = number2, number1
19     # Prompt the student to answer "What is number1 - numbe
20     answer = eval(input("What is " + str(number1) + " - " +
21         str(number2) + "? "))
24      if number1 - number2 == answer:
25          print("You are correct!")
26          correctCount += 1
27      else:
28          print("Your answer is wrong.\n", number1, "-",
29              number2, "is", number1 - number2)
30
31      # Increase the count
32      count += 1
34  endTime = time.time() # Get end time
35  testTime = int(endTime - startTime) # Get test time
36  print("Correct count is", correctCount, "out of",
37      NUMBER_OF_QUESTIONS, "\nTest time is", testTime, "seconds")
```

```
What is 1 - 1? 0  ↵Enter
You are correct!

What is 7 - 2? 5  ↵Enter
You are correct!

What is 9 - 3? 4  ↵Enter
Your answer is wrong.
9 - 3 is 6

What is 6 - 6? 0  ↵Enter
You are correct!

What is 9 - 6? 2  ↵Enter
Your answer is wrong.
9 - 6 is 3

Correct count is 3 out of 5
Test time is 10 seconds
```

利用哨兵值控制循环

LISTING 5.5 SentinelValue.py

```
data = eval(input("Enter an integer (the input ends " +
    "if it is 0): "))

# Keep reading data until the input is 0
sum = 0
while data != 0:
    sum += data

    data = eval(input("Enter an integer (the input ends " +
        "if it is 0): "))

print("The sum is", sum)
```

步哨式控制循环

哨兵值: data == 0

```
Enter an integer (the input ends if it is 0): 4
Enter an integer (the input ends if it is 0): 0
```

利用哨兵值控制循环

注意

在循环控制条件中不要利用浮点值来比较相等，因为浮点值只是近似值，不精确，存在计算误差。

下面为一个计算 **1 + 0.9 + 0.8 + ... + 0.1**的程序：

```
item = 1
sum = 0

while item != 0:
    sum += item
    item -= 0.1

print(sum)
```

死循环

for 循环

通常, for 循环语句如下：

```
for var in sequence:
    # Loop body
    statements
```

```
for i in range(initialValue, endValue):
    # Loop body
```

```
i = initialValue
while i < endValue:
    # Loop body
    i += 1
```

Range 函数

函数 range(a, b) 返回一个整数序列： a, a + 1, ..., b - 2, b - 1. 默认a的值为0.

randrange(a,b)
in random module

```
>>> for v in range(4, 8):
...     print(v)
...
4
5
6
7
```

range(a, b, k),
其中 **k** 代表步长.

```
>>> for v in range(3, 9, 2):
...     print(v)
...
3
5
7
```

```python
for i in range(10):
    if i % 2 == 0:
        i = 1
    print(i)
```

结果是什么？

1
1
1
3
1
5
1
7
1
9

while 循环和 for 循环之间的转换

```
sum = 0
for i in range(1001):
    sum = sum + i
```

```
sum = 0
i = 0
while i < 1001:
    i = i + 1
    sum = sum + i
```

嵌套循环

```
for iterating_var in sequence:
    for iterating_var in sequence:
        statements(s)
    statements(s)
```

内循环

外循环

```
while expression:
    while expression:
        statements(s)
    statements(s)
```

```
for i in range(5):
    for j in range(i+1,5):
        print(i,j)
```

```
0 1
0 2
0 3
0 4
1 2
1 3
1 4
2 3
2 4
3 4
```

注意: while 循环和 for 循环可以相互嵌套。

例子：乘法口诀表

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

一个 for 循环

嵌套循环

例子：乘法口诀表

```
print("            Multiplication Table")
# Display the number title
print("   |", end = '')
for j in range(1, 10):
    print("  ", j, end = '')
print()  # Jump to the new line
print("-------------------------------------")

# Display table body
for i in range(1, 10):
    print(i, "|", end = '')
    for j in range(1, 10):
        # Display the product and align properly
        print(format(i * j, "4d"), end = '')
    print()  # Jump to the new line
```

例子：乘法口诀表

```
print(' '* 12,'Multiplication Table')
print(' '* 4, end = '')
for i in range(1,10):
    print(format(i,'>4d'),end = '')
print()
print('-'*40)
for i in range(1,10):
    print(i,end = '|   ')
    for j in range(1,10):
        print(format(j * i,'>4d'), end = '')
    print()
```

嵌套循环

给出下面程序的结果

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

```
i = 5
while i >= 1:
    num = 1
    for j in range(1, i + 1):
        print(num, end = "xxx")
        num *= 2
    print()
    i -= 1
```

嵌套循环

Question: 用嵌套循环编写出如下表的程序

							1							
						1	2	1						
					1	2	4	2	1					
				1	2	4	8	4	2	1				
			1	2	4	8	16	8	4	2	1			
		1	2	4	8	16	32	16	8	4	2	1		
	1	2	4	8	16	32	64	32	16	8	4	2	1	
1	2	4	8	16	32	64	128	64	32	16	8	4	2	1

找规律：1，确定多少行；2，空格数，数值获取方法

```
                                1
                            1   2   1
                        1   2   4   2   1
                    1   2   4   8   4   2   1
                1   2   4   8  16   8   4   2   1
            1   2   4   8  16  32  16   8   4   2   1
        1   2   4   8  16  32  64  32  16   8   4   2   1
    1   2   4   8  16  32  64 128  64  32  16   8   4   2   1
```

```python
for i in range(1,9):
    for j in range(9-1-i,0,-1): #get the spaces
        print(' '*4, end = '')
    for j in range(0,i):
        print(format(2**j,'>4d'),end = '') #the left side
    if i > 1:
        for j in range(i-1,0,-1):
            print(format(2**(j-1),'>4d'),end = '') #the right side
    print('\n')    # a newline
```

```
                                1
                            1   2   1
                        1   2   4   2   1
                    1   2   4   8   4   2   1
                1   2   4   8  16   8   4   2   1
            1   2   4   8  16  32  16   8   4   2   1
        1   2   4   8  16  32  64  32  16   8   4   2   1
    1   2   4   8  16  32  64 128  64  32  16   8   4   2   1
```

```
for i in range(1, 9):
    print('    ' * (8-i), end = '')
    for j in range(0, i):
        print(format(2 ** j, '4d'), end ='')
    for j in range(1, i):
        print(format(2 ** (i-j-1), '4d'), end = '')
    print()
```

最小化数值误差

LISTING 5.7 TestSum.py

```
# Initialize sum
sum = 0

# Add 0.01, 0.02, ..., 0.99, 1 to sum
i = 0.01
while i <= 1.0:
    sum += i
    i = i + 0.01

# Display result
print("The sum is", sum)
```

$i \approx 1.00000000x$

```
The sum is 49.5
```

最小化数值误差

```
sum = 0

# Add 0.01, 0.02, ..., 0.99, 1 to sum
i = 0.01
for count in range(100):
    sum += i
    i = i + 0.01

# Display result
print("The sum is", sum)
```

After this loop, `sum` is `50.5`.

例子：找出最大公约数

比如：16 和 24 的最大公约数是 8。

假设两个数分别为 n1 和 n2。

```
# Prompt the user to enter two integers
n1 = eval(input("Enter first integer: "))
n2 = eval(input("Enter second integer: "))

gcd = 1
k = 2
while k <= n1 and k <= n2:
    if n1 % k == 0 and n2 % k == 0:
        gcd = k
    k += 1

print("The greatest common divisor for",
    n1, "and", n2, "is", gcd)
```

k <= min(n1, n2)

例子：找出最大公约数

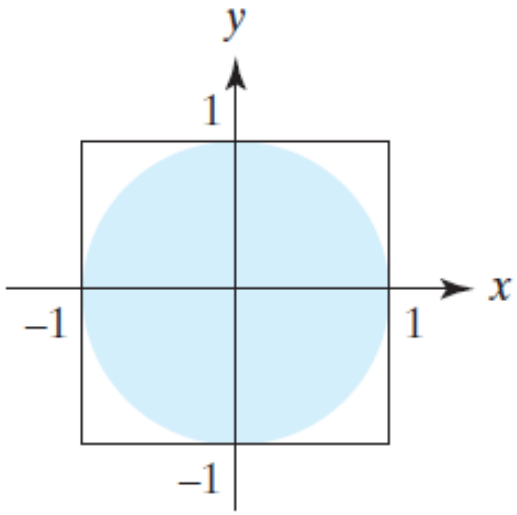
比如：16 和 24 的最大公约数是 8。

假设两个数分别为 n1 和 n2。

min

```
n1,n2 = eval(input('please enter two interges: '))
for i in range(1,max(n1 + 1,n2 + 1)):
    if n1 % i == 0 and n2 % i == 0:
        a = i
print('the Greatest Common Divisor is',a)
```

例子：利用蒙特卡洛方法计算pi



假设圆的半径为1，那么圆的面积为 pi，正方形的面积则为4，因为一个点随机撒在这个圆内的概率为:

circleArea / squareArea = π / 4.

因此我们可以通过该方程计算 pi。

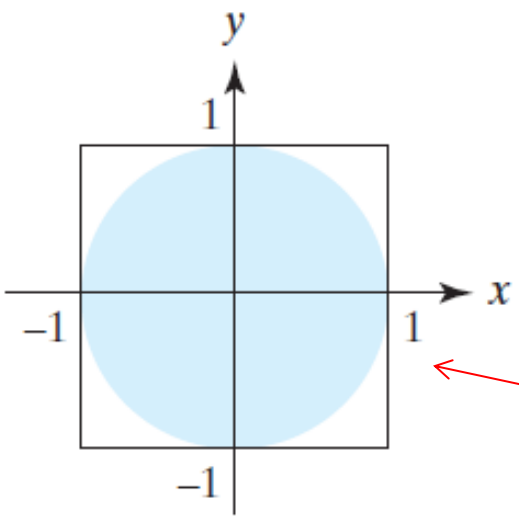
Step 1: 确定撒点次数

Step 2: 确定 x 和 y 的坐标

Step 3: 统计在圆内的点数

Step 4: 计算 pi = 4 *圆内点数/总的撒点次数

例子：利用蒙特卡洛方法计算pi




```
import random

NUMBER_OF_TRIALS = 1000000  # Constant
numberOfHits = 0

for i in range(NUMBER_OF_TRIALS):
    x = random.random() * 2 - 1
    y = random.random() * 2 - 1

    if x * x + y * y <= 1:
        numberOfHits += 1

pi = 4 * numberOfHits / NUMBER_OF_TRIALS

print("PI is", pi)
```

break 和 continue

循环体中的关键字 **break** 帮助退出整个循环。

LISTING 5.11 TestBreak.py

```
sum = 0
number = 0

while number < 20:
    number += 1
    sum += number
    if sum >= 100:
        break

print("The number is", number)
print("The sum is", sum)
```

```
The number is 14
The sum is 105
```

break 和 continue

循环体中的关键字 **continue** 帮助退出或跳过当前的迭代。

LISTING 5.12 TestContinue.py

```
sum = 0
number = 0

while number < 20:
    number += 1
    if number == 10 or number == 11:
        continue
    sum += number

print("The sum is", sum)
```

10 和 11 并没加进来.

```
The sum is 189
```

break 和 continue

给出下面程序的结果

```
balance = 1000
while True:
    if balance < 9:
        break
    balance = balance - 9

print("Balance is", balance)
```

Balance is 1

```
balance = 1000
while True:
    if balance < 9:
        continue
    balance = balance - 9

print("Balance is", balance)
```

死循环

break 和 continue

给出下面程序的结果

```
for i in range(1, 4):
    for j in range(1, 4):
        if i * j > 2:
            break

        print(i * j)

    print(i)
```

```
for i in range(1, 4):
    for j in range(1, 4):
        if i * j > 2:
            continue

        print(i * j)

    print(i)
```

例子：显示下面表中的素数

素数：大于 2，且只能被自己和 1 整除的数。如 2, 3, 5, 7 都为素数, 但 **4**, **6**, **8**, 和 **9** 都不是。

Question: 写一个程序，给出前 50 个素数，每行包含 10 个素数。

```
The first 50 prime numbers are

   2    3    5    7   11   13   17   19   23   29
  31   37   41   43   47   53   59   61   67   71
  73   79   83   89   97  101  103  107  109  113
 127  131  137  139  149  151  157  163  167  173
 179  181  191  193  197  199  211  223  227  229
```

例子：显示下面表中的素数

这个问题可以被分成以下几个任务。

- 决定一个特定的数是否为素数。
- 对于 number=2、3、4、5、6、…，测试这个数字是否为素数。
- 统计所有素数的个数。
- 显示每个素数，每行显示 10 个。

(1) NUMBER_OF_PRIMES = 50
(2) 利用变量 count 追踪素数的个数，初始时 count = 0
(3) 因为 素数是从大于1 的整数开始，我们初始时设为 **2**

例子：显示下面表中的素数

```
while count < NUMBER_OF_PRIMES:
    Test if number is prime

    if number is prime:
        Display the prime number and increase count

    Increment number by 1
```

```
Use a Boolean variable isPrime to denote whether
    the number is prime; Set isPrime to True initially

for divisor in range(2, number / 2 + 1):
    if number % divisor == 0:
        Set isPrime to False
        Exit the loop
```

例子：显示下面表中的素数

```
NUMBER_OF_PRIMES = 50  # Number of primes to display
NUMBER_OF_PRIMES_PER_LINE = 10 # Display 10 per line
count = 0  # Count the number of prime numbers
number = 2  # A number to be tested for primeness
print("The first 50 prime numbers are")
while count < NUMBER_OF_PRIMES:
    # Assume the number is prime
    isPrime = True  # Is the current number prime?
    divisor = 2
    while divisor <= number / 2:
        if number % divisor == 0:
            # If true, the number is not prime
            isPrime = False  # Set isPrime to false
            break # Exit the for loop
        divisor += 1
```

例子：显示下面表中的素数

```
    # Display the prime number and increase the cou
    if isPrime:
        count += 1  # Increase the count

        print(format(number, "5d"), end = '')
        if count % NUMBER_OF_PRIMES_PER_LINE == 0:
            # Display the number and advance to the
            print()  # Jump to the new line

    # Check if the next number is prime
    number += 1
```

例子：显示下面表中的素数

```
count,lines,numberOfPrime = 0,10,50
number = 2
print('The first prime are')
print()
while count < numberOfPrime:
    for i in range(2,(number//2+1)):
        if number % i == 0:
            break
    else:
        count += 1
        print(format(number,'>5d'),end = '')
        if count % lines == 0:
            print()
    number += 1
```

这是可以的

Exercises

- P129 5.1
- P130 5.7
- P130 5.9, 5.13, 5.18
- P133 5.26