



Chap5 Program Control Structure

第5章 程序控制结构

Nanjing University

Department of Computer Science and Technology

Department of University Basic Computer Teaching

程序控制结构



sequence
structure



selection
structure

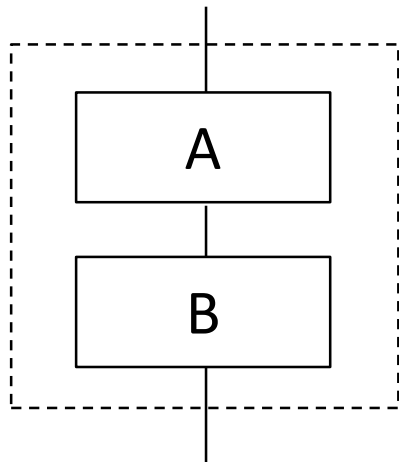


repetition
structure

5.1

顺序结构

顺序结构



一个入口
一个出口



```
# Filename: seq.py  
mystring = 'Hello, World!'  
print(mystring)
```

5.1.1 赋值语句

赋值语句

普通
赋值

$r = 2$

增量
赋值

$m /= 5$

链式
赋值

$b = a = a + 1$

多重
赋值

$p, r = 3, 5$

- 多重赋值的基本形式

变量1, 变量2, ..., 变量n = 表达式1, 表达式2, ..., 表达式n



```
>>> name, age = 'Niuyun', 18
```

```
>>> name
```

```
'Niuyun'
```

```
>>> age
```

```
18
```

多重赋值的本质

Source

```
>>> name, age = 'Niuyun', 18
```

```
>>> temp = 'Niuyun', 18
```

```
>>> temp
```

```
('Niuyun', 18)
```

```
>>> name, age = temp
```

```
>>> name
```

```
'Niuyun'
```

```
>>> age
```

```
18
```

元组打包
Tuple packing

序列解包
Sequence unpacking



```
>>> x = 3
```

```
>>> y = 5
```

```
>>> x, y = y, x
```

```
>>> x
```

```
5
```

```
>>> y
```

```
3
```

语法糖
syntactic sugar

5.1.2 基本输入和输出语句

输入/输出

11



`input()`



`print()`

输入函数input()

输入语句的一般形式：

```
x = input(['输入提示'])
```

返回值类型是str



输入input()函数

返回值类型：str

int()函数

float()函数




```
>>> x = input('Enter an integer between 0 and 10: ')
Enter an integer between 0 and 10: 3
>>> x
'3'
>>> x = int(input('Enter an integer between 0 and 10: '))
Enter an integer between 0 and 10: 3
>>> x
3
>>> y = float(input('Enter the price of the apples: '))
Enter the price of the apples: 4.5
>>> y
4.5
```

输入input()函数

返回值类型 : str

eval()函数



```
>>> z = eval(input('Enter a number: '))
Enter a number: 3
>>> z * 3
9
>>> z = eval(input('Enter the price of every apple: '))
Enter the price of every apple: 3.5
>>> z * 3
10.5
>>> z = eval(input('Enter the price of the apples: '))
Enter the price of the apples: 3 * 3.5
>>> z
10.5
```

输出函数print()

输出语句的一般形式：

输出到标准输出设备

```
print(对象1, 对象2, ..., 对象n, sep = ' ', end = '\n' )
```

- sep表示输出对象之间的分隔符，默认为空格
- 参数end的默认值为'\n'，表示print()函数输出完成后自动换行



输出函数print()

改变sep的值



```
>>> print('1', '2', '3')
```

```
1 2 3
```

```
>>> print('1', '2', '3', sep = ',')
```

```
1,2,3
```


格式化输出形式：

- `print('格式字符串' % (对象1, 对象2, ..., 对象n))`
- `print('格式化模板'.format(对象1, 对象2, ..., 对象n))`

输出函数print()——格式化模板

18

Source

```
>>> "{0} is taller than {1}.".format("Xiaoma", "Xiaowang")
'Xiaoma is taller than Xiaowang.'
>>> age, height = 21, 1.758
>>> print("Age:{0:<5d}, Height:{1:5.2f}".format(age, height))
Age:21  , Height: 1.76
```

{参数的位置:[对齐说明符][符号说明符][最小宽度说明符][.精度说明符][类型说明符]}

符号	描述
b	二进制，以2为基数输出数字
o	八进制，以8为基数输出数字
x	十六进制，以16为基数输出数字，9以上的数字用小写字母（类型符为X时用大写字母）表示
c	字符，将整数转换成对应的Unicode字符输出
d	十进制整数，以10为基数输出数字
f	定点数，以定点数输出数字
e	指数记法，以科学计数法输出数字，用e（类型符是E时用大写E）表示幂
[+]m.nf	输出带符号（若格式说明符中显式使用了符号“+”，则输出大于或等于0的数时带“+”号）的数，保留n位小数，整个输出占m列（若实际宽度超过m则突破m的限制）
0>5d	右对齐，>左边的0表示用0填充左边，>右边的数字5表示输出项宽度为5
<	左对齐，默认用空格填充右边，<前后类似上述右对齐可以加填充字符和宽度
^	居中对齐
{}	输出一个{}

5.2

选择结构

5.2.1 条件

语 法

if 表达式(条件):
语句序列

表达式 (条件)

- 简单的数字或字符
- 条件表达式:
 - 关系运算符
 - 成员运算符
 - 逻辑运算符
- True 或 False

语句序列

- 条件为True时执行的代码块
- 同一语句序列必须在同一列上进行相同的缩进（通常为4个空格）

if 语句

23

F_{ile}

Filename: ifpro.py

sd1 = 3

sd2 = 3

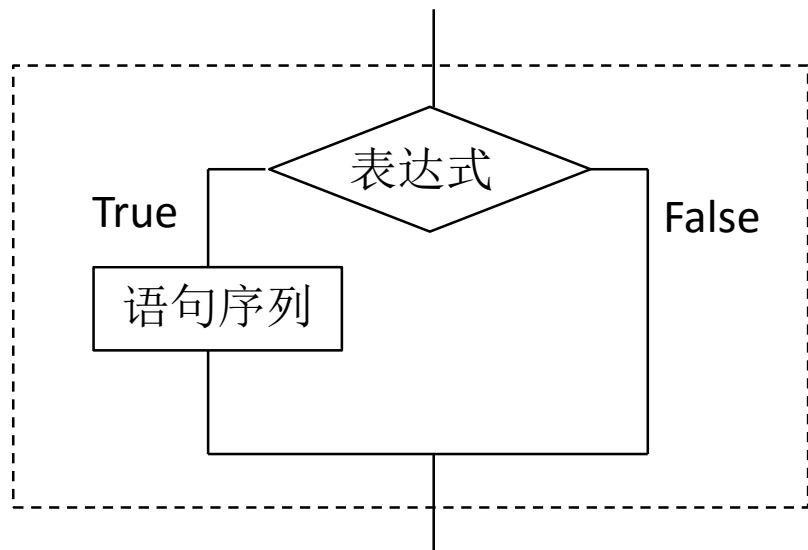
if sd1 == sd2:

 print("the square's area is", sd1*sd2)

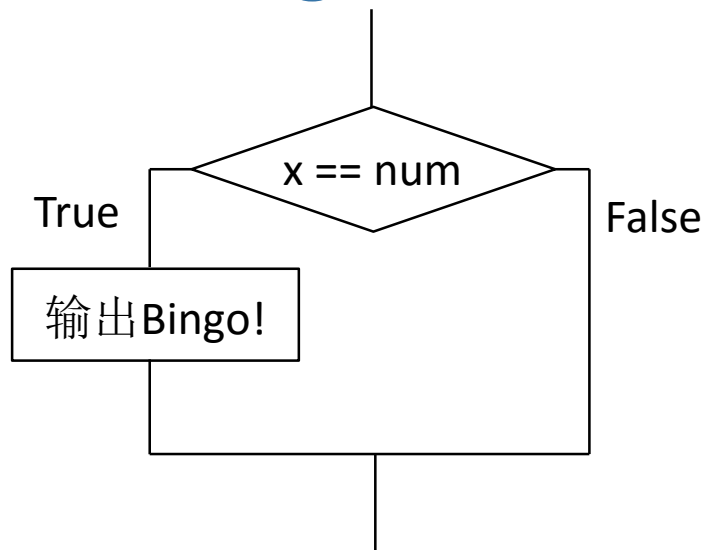
I_{nput} and O_{utput}

the square's area is 9

例5.1 程序随机产生一个0~300之间的整数， 玩家竞猜，若猜中则提示Bingo



单分支结构流程图



猜数字流程图

例5.1 程序随机产生一个0~300之间的整数， 玩家竞猜，若猜中则提示Bingo

F_{ile}

```
# prog5-1.py
from random import randint
x = randint(0, 300)
num = int(input('Please enter a number between 0~300: '))
if num == x:
    print('Bingo!')
```

很难一次性猜对！
而且毫无反馈！

5.2.2 else子句

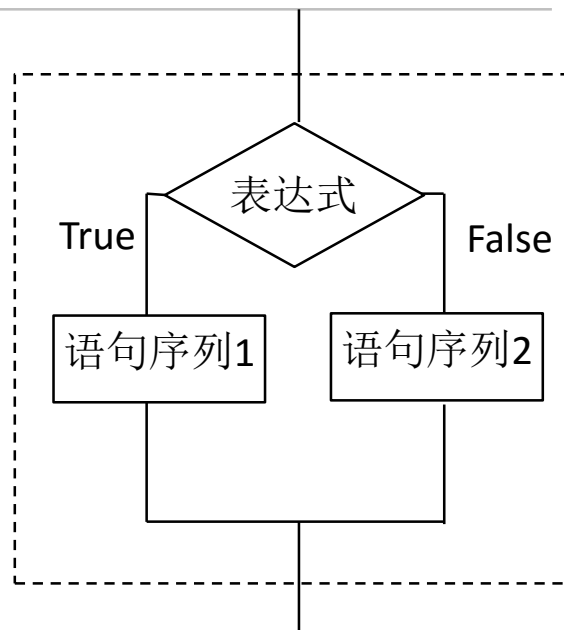
else 语句

语 法

if 表达式:
 语句序列1
else:
 语句序列2

语句序列2

- 表达式条件为 False 时执行的代码块
- 代码块必须缩进
- else 语句不缩进



两路分支结构流程图

例5.2 程序随机产生一个0~300之间的整数，玩家竞猜，若猜中则提示Bingo，否则提示Wrong

File

```
# prog5-2.py
```

```
from random import randint
```

```
x = randint(0, 300)
```

```
num = int(input('Please enter a number between 0~300: '))
```

```
if num == x:
```

```
    print('Bingo!')
```

```
else:
```

```
    print('Wrong!')
```

Input and Output

Please enter a number between 0~300: 178
Wrong!

else 语句

29

File

Filename: elsepro.py

```
sd1 = int(input('the first side: '))
```

```
sd2 = int(input('the second side: '))
```

```
if sd1 == sd2:
```

```
    print("the square's area is", sd1*sd2)
```

```
else:
```

```
    print("the rectangle's area is", sd1*sd2)
```

Input and Output

the first side: 4

the second side: 4

the square's area is 16

else 语句

F_{ile}

Filename: elsepro-2.py

```
x = eval(input('Please enter the first number: '))
y = eval(input('Please enter the second number: '))
if x >= y:
    t = x
else:
    t = y
```

检查条件“ $x \geq y$ ”是否满足，若满足则取 x 否则取 y 赋给变量 t

else 语句——三元运算符

条件表达式（也称三元运算符）的常见形式如下所述：

$x \text{ if } C \text{ else } y$

F_{ile}

Filename: elsepro-2.py

$x = \text{eval}(\text{input}(\text{'Please enter the first number: '}))$

$y = \text{eval}(\text{input}(\text{'Please enter the second number: '}))$

if $x \geq y$:

$t = x$

else:

$t = y$

$t = x \text{ if } x \geq y \text{ else } y$

5.2.3 elif子句

语 法

```
if 表达式1:  
    语句序列1  
elif 表达式2:  
    语句序列2  
...  
elif 表达式N-1:  
    语句序列N-1  
else:  
    语句序列N
```

语句序列2

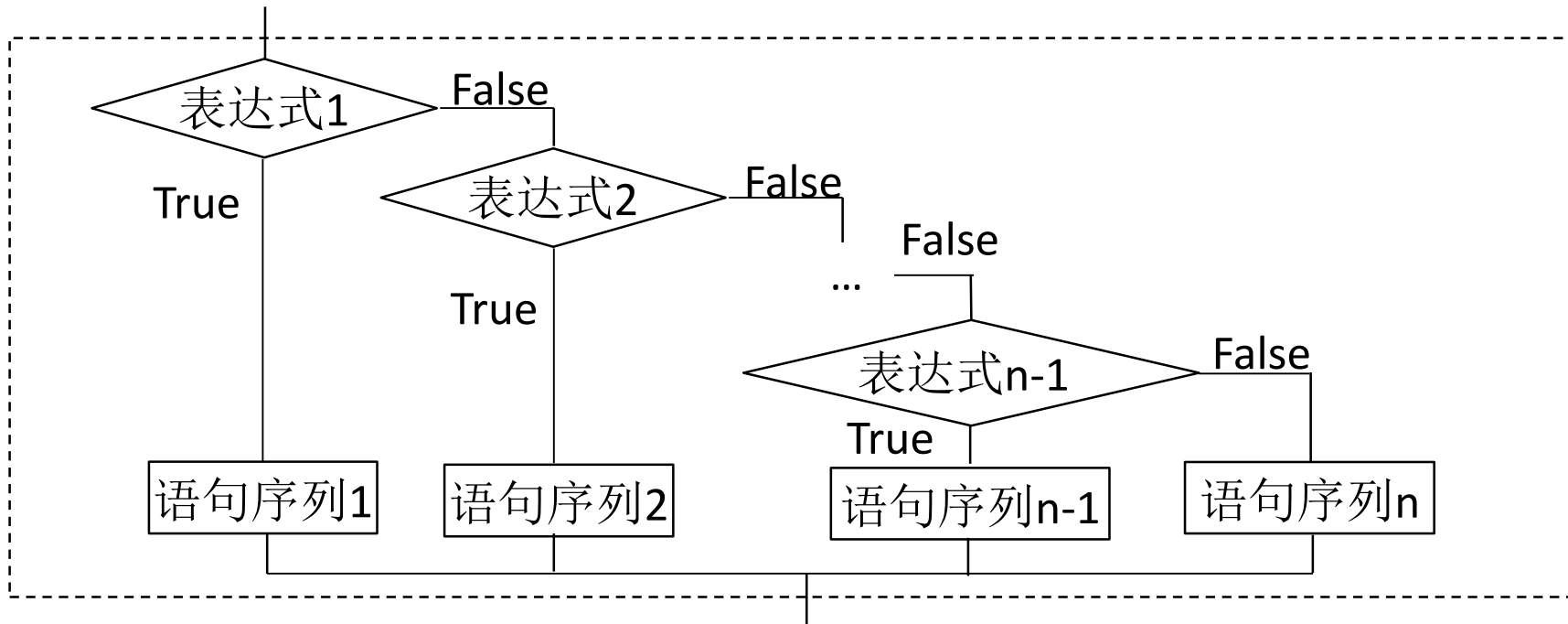
- 表达式2为True时执行的代码块

语句序列N-1

- 表达式N为True时执行的代码块

语句序列N

- 语句序列N是以上所有条件都不满足时执行的代码块



多分支结构流程图

例5.3 猜数字游戏

- 程序随机产生一个0~300之间的整数，玩家竞猜，若猜中则提示Bingo，若猜大了提示Too large，否则提示Too small



Filename: 5-3-1.py

```
from random import randint
x = randint(0, 300)
digit = int(input('Please input a number between 0~300: '))
if digit == x:
    print('Bingo!')
elif digit > x:
    print('Too large, please try again.')
else:
    print('Too small, please try again.')
```

elif 语句

36

File

Filename: elifpro.py

```
k = input('input the index of shape: ')
```

```
if k == '1':
```

```
    print('circle')
```

```
elif k == '2':
```

```
    print('oval')
```

```
elif k == '3':
```

```
    print('rectangle')
```

```
elif k == '4':
```

```
    print('triangle')
```

```
else:
```

```
    print('you input the invalid number')
```

Input and

Output

input the index of shape: 3
rectangle

Input and

Output

input the index of shape: 8
you input the invalid number

5.2.4 嵌套的if语句

嵌套的if语句

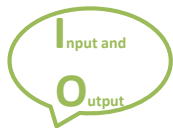
语 法



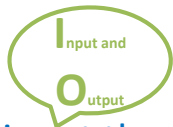
```
1 : if 表达式1:  
2 :     if 表达式2:  
3 :         语句序列1  
4 :     else:  
5 :         语句序列2  
6 : else:  
7 :     if 表达式3:  
8 :         语句序列3  
9 :     else:  
10:         语句序列4
```

条件嵌套

- 同等缩进为同一条件结构



input the index of shape: 3
the first side: 3
the second side: 4
the rectangle's area is 12



input the index of shape: 2
oval



```
# Filename: ifnestpro.py
k = input('input the index of shape: ')
if k == '1':
    print('circle')
elif k == '2':
    print('oval')
elif k == '3':
    sd1 = int(input('the first side: '))
    sd2 = int(input('the second side : '))
    if sd1 == sd2:
        print("the square's area is", sd1*sd2)
    else:
        print("the rectangle's area is", sd1*sd2)
elif k == '4':
    print('triangle')
else:
    print('you input the invalid number')
```

例5.3 猜数字游戏——改写代码



```
# Filename: 5-3-1.py
from random import randint
x = randint(0, 300)
digit = int(input('Please input a number
between 0~300: '))
if digit == x :
    print('Bingo!')
elif digit > x:
    print('Too large, please try again.')
else:
    print('Too small, please try again.')
```



```
# Filename: 5-3-2.py
from random import randint
x = randint(0, 300)
digit = int(input('Please input a number between 0~300: '))
if digit == x :
    print('Bingo!')
else:
    if digit > x:
        print('Too large, please try again.')
    else:
        print('Too small, please try again.')
```


例5.4 符号函数 (sign function)

- 请分别用if-elif-else结构和嵌套的if结构实现符号函数 (sign function) , 符号函数的定义 :

$$\text{sgn}(x) = \begin{cases} -1, & x < 0 \\ 0, & x = 0 \\ 1, & x > 0 \end{cases}$$

例5.4 符号函数



```
# prog5-4-1.py
x = eval(input('Enter a number: '))
if x < 0:
    sgn = -1
elif x == 0:
    sgn = 0
else:
    sgn = 1
print('sgn = {:.0f}'.format(sgn))
```

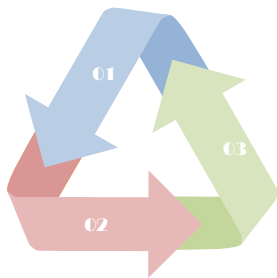


```
# prog5-4-2.py
x = eval(input('Enter a number: '))
if x != 0:
    if x < 0:
        sgn = -1
    else:
        sgn = 1
else:
    sgn = 0
print('sgn = {:.0f}'.format(sgn))
```

5.3

循环

- 循环结构是满足一个指定的条件，每次使用不同的数据对算法中的计算或处理步骤完全相同的部分**重复**计算若干次的算法结构，也称为重复结构



5.3.1 while语句

while 循环

语法

While 表达式:
语句序列 (循环体)

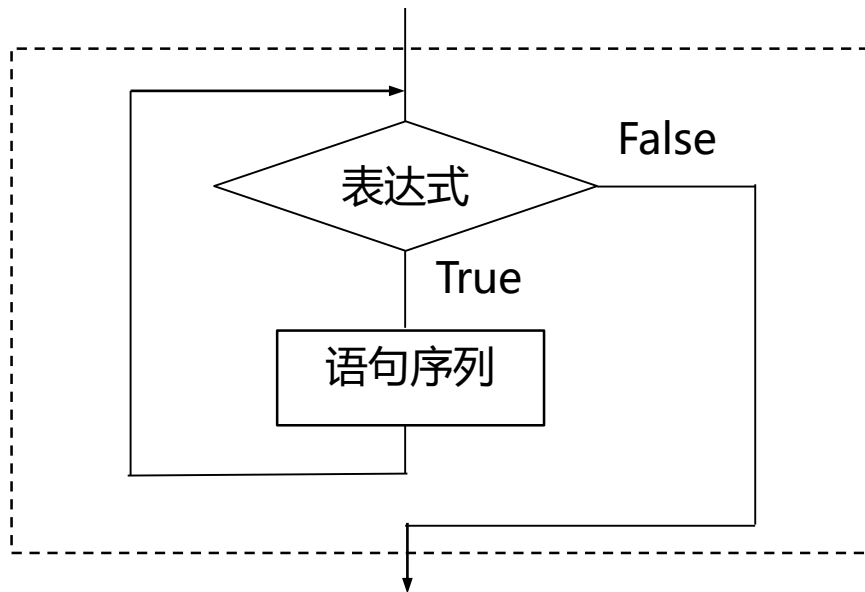
表达式

- 当表达式值为True时执行语句序列代码块
- 继续判断表达式的值是否为True,
- 若是则继续执行循环体,
- 如此周而复始, 直到表达式的值为False或发生异常时停止循环的执行

while 语句

有几点要注意：

- while语句是先判断再执行，所以循环体有可能一次也不执行；
- 循环体中需要包含能改变循环变量值的语句，否则表达式的结果始终是True的话会造成死循环；
- 要注意语句序列的对齐，while语句只执行其后的一条或一组同一层次的语句。



while语句流程图

例5.5 计算 $1+2+\dots+100$ 的值



```
# prog5-5.py
```

```
sum = 0
```

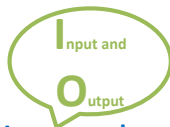
```
i = 1
```

```
while i <= 100:
```

```
    sum += i
```

```
    i += 1
```

```
print('1+2+...+100 = {:d}'.format(sum))
```



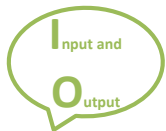
input the index of shape: 2
oval

经典累加问题

例5.6 求两个正整数的最大公约数和最小公倍数。

S1：判断 x 除以 y 的余数 r 是否为0。
若 r 为0则 y 是 x 、 y 的最大公约数，
继续执行后续操作；否则 $y \rightarrow x$ ，
 $r \rightarrow y$ 重复执行第S1步。

S2：输出（或返回） y 。



Enter the first number: 18

Enter the second number: 24

最大公约数 = 6

最小公倍数 = 72

Source

```
# prog5-6.py
```

```
# -*- coding: gb2312 -*-
```

```
x = eval(input("Enter the first number: "))
```

```
y = eval(input("Enter the second number: "))
```

```
z = x * y
```

```
if x < y:
```

```
    x, y = y, x
```

```
while x % y != 0:
```

```
    r = x % y
```

```
    x = y
```

```
    y = r
```

```
print("最大公约数 = ", r)
```

```
print("最小公倍数 = ", z // r)
```

例5.7 计算 π

$$\frac{\pi}{4} = 1 - \frac{1}{3} + \frac{1}{5} - \frac{1}{7} \dots$$

通项的绝对值小于等于 10^{-8}
时停止计算

I nput and O utput

pi = 3.141592633590251

math模块中pi值等于
3.141592653589793

S_{ource}

```
# prog5-7.py
import math
x, sum = 1, 0
sign = 1
k = 1
while(math.fabs(x) > 1e-8):
    sum += x
    k += 2
    sign *= -1
    x = sign / k
sum *= 4
print("pi = {:.15f}".format(sum))
```

5.3.2 for语句

for 循环

语 法

for 变量 in 可迭代对象:
语句序列

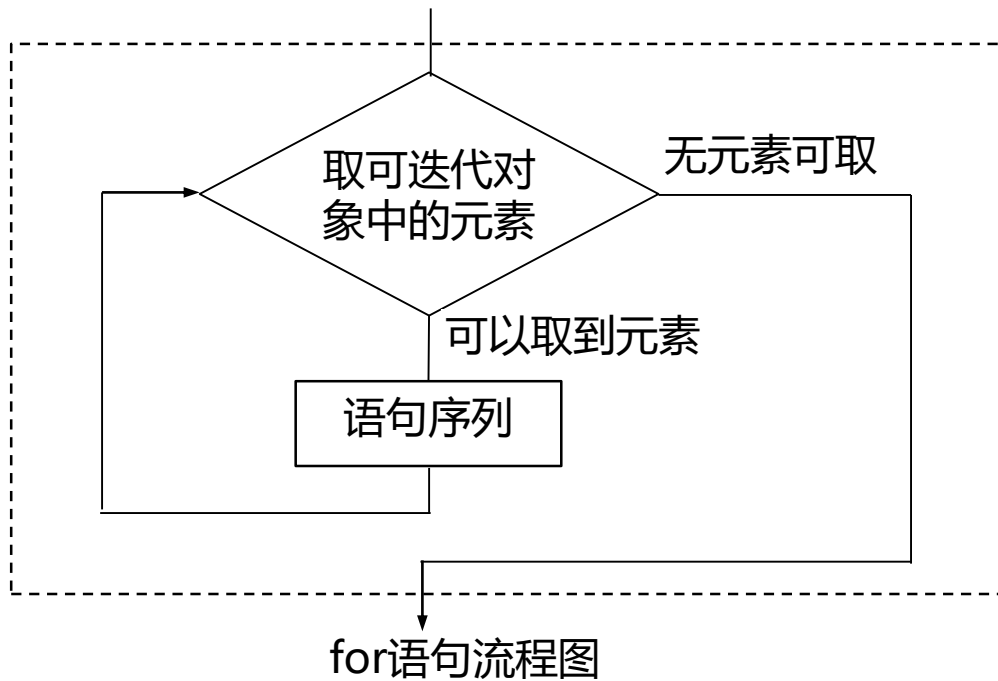
可以明确循环的次数

- 遍历一个数据集内的成员
- 在列表解析中使用
- 生成器表达式中使用

可迭代对象

- String
- List
- Tuple
- Dictionary
- File

可迭代对象指可以按次序迭代（循环）的对象，包括序列、迭代器（iterator）如`enumerate()`函数产生的对象以及其他可以迭代的对象如字典的键和文件的行等。执行时变量取可迭代对象中的一个值，执行语句序列，再取下一个值，执行语句序列



猜数字游戏

- 程序随机产生一个0~300间的整数，玩家竞猜，允许猜多次，系统给出“猜中”、“太大了”或“太小了”的提示。



```
# Filename: guessnum2.py
from random import randint
x = randint(0, 300)
for count in range(5):
    digit = int(input('Please input a number between 0~300: '))
    if digit == x :
        print('Bingo!')
    elif digit > x:
        print('Too large, please try again.')
    else:
        print('Too small, please try again.')
```

for 循环



apple
banana
Pear
(0, 'a')
(1, 'b')
(2, 'c')



```
# Filename: prog_for.py  
for fruit in ['apple', 'banana', 'pear']:  
    print(fruit)  
for item in enumerate(['a', 'b', 'c']):  
    print(item)
```

迭代器 or 列表？

列表，在每次取值时会一次性获取所有值，如果值多的话，会占用更多的内存；而迭代器则是一个接一个计算值，在使用时计算一个值时获取一个值，占内存少。

for循环——可迭代对象

56

序列

01

02

序列索引

迭代器

03

04

字典的键

文件的行

05

for 语句迭代——序列迭代

S_{ource}

```
>>> s = ['I', 'love', 'Python']
>>> for word in s:
    print(word, end = ' ')
I love Python
>>> for i in range(1, 5):
    print(i * i)
1
4
9
16
```

S_{ource}

```
>>> s = 'Python'
>>> for c in s:
    print(c)
P
y
t
h
o
n
>>> for i in range(3,11,2):
    print(i, end = ' ')
3 5 7 9
```

for 语句迭代——序列索引迭代

58

S_{source}

```
>>> s = ['I', 'love', 'Python']
```

```
>>> for i in range(len(s)):
        print(s[i], end = ' ')
```

```
I love Python
```

for 语句迭代——迭代器迭代

59

Source

```
>>> courses = ['Maths', 'English', 'Python']
```

```
>>> scores = [88, 92, 95]
```

```
>>> for c, s in zip(courses, scores):  
    print('{0} - {1:d}'.format(c, s))
```

```
Maths - 88
```

```
English - 92
```

```
Python - 95
```

for 语句迭代——其他迭代

S
ource

```
>>> d_stock = {'AXP': '78.51', 'BA': '184.76', 'CAT': '96.39'}
```

```
>>> for k, v in d_stock.items():  
        print('{0:>3}: {1}'.format(k, v))
```

```
AXP: 78.51
```

```
BA: 184.76
```

```
CAT: 96.39
```

```
>>> for k in d_stock.keys():  
        print(k, d_stock[k])
```

```
AXP 78.51
```

```
BA 184.76
```

```
CAT 96.39
```

例5.8 求斐波纳契 (Fibonacci) 数列前20项

61

斐波纳契数列：

0 , 1 , 1 , 2 , 3 , 5 , 8 , 13 , 21 ,
34 , 55 , 89 , 144 , ...

$$\begin{cases} F_0 = 0 \\ F_1 = 1 \\ F_{I+1} = F_{I-1} + F_I \end{cases}$$

F_{ile}

```
# prog5-8.py
```

```
f = [0] * 20
```

```
f[0], f[1] = 0, 1
```

```
for i in range(2, 20):
```

```
    f[i] = f[i-1] + f[i-2]
```

```
print(f)
```

I nput and O utput

[0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, 233, 377, 610, 987, 1597, 2584, 4181]

例5.9 输出公司代码和股票价格

假设已有若干道琼斯工业指数成分股公司某个时期的财经数据，包括公司代码、公司名称和股票价格：

```
>>> stockList = [('AXP', 'American Express Company', '78.51'),  
                  ('BA', 'The Boeing Company', '184.76'),  
                  ('CAT', 'Caterpillar Inc.', '96.39')]
```

从数据中获取公司代码和股票价格对并输出。

例5.9 输出公司代码和股票价格

用序列索引迭代

Input and Output

```
{'CAT': '96.39', 'BA': '184.76', 'AXP': '78.51'}
```

File

```
# prog5-9-1.py
```

```
stockList = [('AXP', 'American Express Company',  
              '78.51'), ('BA', 'The Boeing Company',  
                        '184.76'), ('CAT', 'Caterpillar Inc.', '96.39')]
```

```
aList = []
```

```
bList = []
```

```
for i in range(3):
```

```
    aStr = stockList[i][0]
```

```
    bStr = stockList[i][2]
```

```
    aList.append(aStr)
```

```
    bList.append(bStr)
```

```
stockDict = dict(zip(aList,bList))
```

```
print(stockDict)
```

例5.9 输出公司代码和股票价格

用序列迭代



```
{'CAT': '96.39', 'BA': '184.76', 'AXP': '78.51'}
```



```
# prog5-9-2.py
```

```
stockList = [('AXP', 'American Express Company',  
              '78.51'), ('BA', 'The Boeing Company',  
              '184.76'), ('CAT', 'Caterpillar Inc.', '96.39')]
```

```
stockDict = {}
```

```
for data in stockList:
```

```
    stockDict[data[0]] = data[2]
```

```
print(stockDict)
```


5.3.3 嵌套循环

- while语句和for语句可以嵌套自身语句结构，也可以相互嵌套，可以呈现各种复杂的形式。

例5.10 编写程序统计一元人民币换成一分、两分和五分的所有兑换方案个数

67



prog5-10.py

i, j, k = 0, 0, 0 # i,j,k分别代表五分、两分和一分的数量

count = 0

for i in range(21):

for j in range(51):

k = 100 - 5 * i - 2 * j

if k >= 0: #k是一分，可以是任意个

count += 1

print('count = {:d}'.format(count))



count = 541

可以用while替换for

例5.11 两个列表的新组合

- 从两个列表中分别选出一个元素，组成一个元组放到一个新列表中，要求新列表中包含所有的组合

Input and Output

```
[('C++', 2), ('C++', 3), ('C++', 4),  
( 'Java', 2), ('Java', 3), ('Java', 4),  
( 'Python', 2), ('Python', 3),  
( 'Python', 4)]
```

File

```
# prog5-11.py
```

```
result = []
```

```
pdlList = ['C++', 'Java', 'Python']
```

```
creditList = [2, 3, 4]
```

```
for pdl in pdlList:
```

```
    for credit in creditList:
```

```
        result.append((pdl, credit))
```

```
print(result)
```

5.3.4 break, continue语句

break 语句

- break语句终止当前循环，转而执行循环之后的语句

循环非正常结束



Filename: breakpro.py

```
sum = 0
```

```
i = 1
```

```
while i < 10:
```

```
    sum += i
```

```
    if sum > 10:
```

```
        break
```

```
    i += 1
```

```
print('i = {0:d}, sum = {1:d}'.format(i, sum))
```



i=5, sum=15

sum是1+2+3+...不断累加的和，当i等于5时sum第一次大于10，执行break语句跳出while循环语句，继而去执行print语句

break 语句

“while i < 10:” 意义不大，常会将此条语句替换成另一种在Python中常与break语句一起使用的循环控制语句 “while True:”



Filename: breakpro.py

```
sum = 0
```

```
i = 1
```

```
while True:
    sum += i
    if sum > 10:
        break
    i += 1
```

```
print('i = {0:d}, sum = {1:d}'.format(i, sum))
```

while i < 10:




while True:

break 语句

i 和 j 分别等于5和10时 i 和 j 的乘积第一次等于50，如果 break语句可以跳出两重循环的话则输出结果应该是 “5 10”，而程序的实际输出结果是 “10 5”，所以break只能跳出紧包层即break所在层次的循环。

两重循环



```
for i in range(11):  
    for j in range(11):  
        if i * j >= 50:  
            break  
print(i, j)
```


例5.12 输出2~100之间的素数，每行显示5个

素数 (prime) : 只能被1和n自身整除的正整数n (13是素数，6不是素数)。

素数判断算法：

- 若n不能被2~n-1范围内的任一个整数整除n就是素数，否则n不是素数
- 如果发现n能被某个整数整除可立即停止继续判断n是否能被范围内其他整数整除。



```
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
```

$2 \sim n/2$

or

$2 \sim \sqrt{n}$

例5.12 输出2~100之间的素数，每行显示5个

for 循环和break

Input and Output

```
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
```

File

```
# Filename: 5-12.py
from math import sqrt
j = 2 ; count = 0
while j <= 100:
    i = 2
    k = sqrt(j)
    while i <= k:
        if j%i == 0: break
        i += 1
    if i > k:
        count += 1
        if count % 5 == 0:
            print(j, end = '\n')
        else:
            print(j, end = ' ')
    j += 1
```

for 循环和break

I nput and O utput

```
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
```

F ile

```
from math import sqrt
for i in range(2,101):
    k = int(sqrt(i)); flag = 1
    for j in range(2,k+1):
        if i%j == 0:
            flag = 0
            break
    if ( flag ):
        count += 1
        if count % 5 == 0:
            print(i, end = '\n')
        else:
            print(i, end = ' ')
```

continue 语句

- 在while和for循环中，continue语句的作用：
 - 跳过循环体内continue后面的语句，并开始新一轮循环
 - while循环则判断循环条件是否满足
 - for循环则判断迭代是否已经结束

continue语句

程序的功能是对于在1~20之间的数，当它是3的倍数时执行print(i, end = ' ')函数调用语句，当它不是3的倍数时执行continue语句，跳过其后的print()函数调用语句继续执行下一轮循环




```
for i in range(1,21):  
    if i % 3 != 0:  
        continue  
    print(i, end = ' ')
```



3 6 9 12 15 18


continue语句

循环中的break :



```
for i in range(1,21):  
    if i % 3 != 0:  
        break  
    print(i, end = ' ')
```


循环中的continue :



```
for i in range(1,21):  
    if i % 3 != 0:  
        continue  
    print(i, end = ' ')
```


break	continue
break语句跳出所有轮循环	continue语句则是跳出本轮循环
没有任何输出	输出1-20之间所有3的倍数"3 6 9 12 15 18"

循环中的continue :



```
for i in range(1,21):  
    if i % 3 != 0:  
        continue  
    print(i, end = ' ')
```

循环中的替代continue :



```
for i in range(1,21):  
    if i % 3 == 0:  
        print(i, end = ' ')
```

5.3.5 循环结构中的else子句

- 循环中的else子句：
 - 如果循环代码从break处终止，跳出循环
 - 正常结束循环，则执行else中代码

例5.14 输入一个整数，并判断其是否为素数

82

如果输入的整数有有效因子（除了1和它本身的因子）则可以直接输出非素数的结论并继而执行break语句跳出循环，如果没有执行过break语句则表示循环正常结束，也就是整数并无有效因子则表明它是一个素数，则执行else子句的输出语句。



```
# Filename: 5-14.py
from math import sqrt
num = int(input('Please enter a number: '))
j = 2
while j <= int(sqrt(num)):
    if num % j == 0:
        print('{:d} is not a prime.'.format(num))
        break
    j += 1
else:
    print('{:d} is a prime.'.format(num))
```

例5.15 猜数字游戏（想停就停，非固定次数）

程序随机产生一个0~300间的整数，玩家竞猜，允许玩家自己控制游戏次数，如果猜中系统给出提示并退出程序，如果猜错给出“太大了”或“太小了”的提示，如果不想继续玩可以退出并说再见。

File

```
# Filename: guessnum3.py
from random import randint
x = randint(0, 300)
go = 'y'
while (go == 'y'):
    digit = int(input('Please input a number between 0~300: '))
    if digit == x:
        print('Bingo!')
        break
    elif digit > x:
        print('Too large, please try again.')
    else:
        print('Too small, please try again.')
    print('Input y if you want to continue.')
    go = input()
    print(go)
else:
    print('I quit and byebye!')
```

5.3.6 特殊的循环—列表解析

- Python中有一种特殊的循环，通过for语句结合if语句，利用其他列表动态生成新列表，这种特殊的轻量级循环称为列表解析（list comprehension，也译作列表推导式）。

列表解析的语法形式

- 列表解析中的多个for语句相当于是for结构的嵌套使用

```
[ 表达式 for 表达式1 in 序列1  
    for 表达式2 in 序列2  
    ...  
    for 表达式N in 序列N  
    if 条件 ]
```

创建一个从0到9的简单的整数序列；



```
>>> [x for x in range(10)]  
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
```

对range(10)中每一个值求平方数；



```
>>> [x ** 2 for x in range(10)]  
[0, 1, 4, 9, 16, 25, 36, 49, 64, 81]
```


对range(10)中的每一个值加入if语句“if x ** 2 < 50”，只生成比50小的平方数；



```
>>> [x ** 2 for x in range(10) if x ** 2 < 50]  
[0, 1, 4, 9, 16, 25, 36, 49]
```

使用嵌套的for语句。(x + 1, y + 1)的所有组合就包含了x从0-1, y也从0-1的变化。

A small orange speech bubble icon containing the word "Source" in a stylized font.

```
>>> [(x + 1, y + 1) for x in range(2) for y in range(2)]  
[(1, 1), (1, 2), (2, 1), (2, 2)]
```

例5.12 列表解析方法

91

S

ource

```
>>> pdlList = ['C++', 'Java', 'Python']  
>>> creditList = [2, 3, 4]  
>>> [(pdl, credit) for pdl in pdlList for credit in creditList]  
[('C++', 2), ('C++', 3), ('C++', 4), ('Java', 2), ('Java', 3), ('Java',  
4), ('Python', 2), ('Python', 3), ('Python', 4)]
```