

# 人工智能程序设计

---

M1 Python程序设计基础

3 程序控制结构

张 莉

---



# 程序控制结构



sequence  
structure



selection  
structure



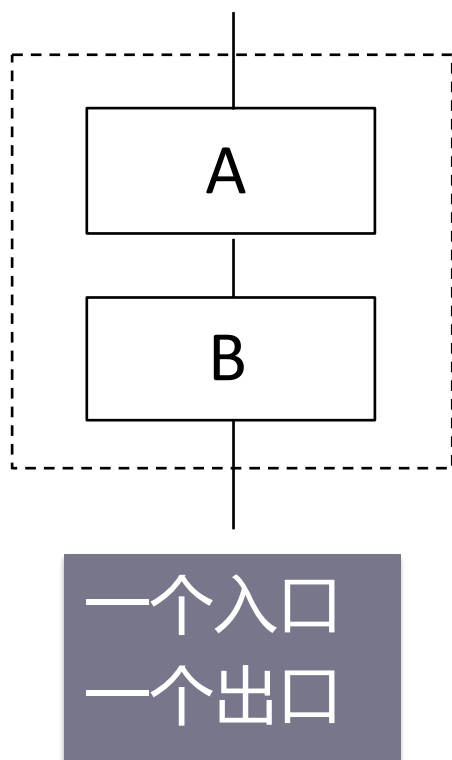
repetition  
structure

# 1 人工智能程序设计 顺序结构

**顺序**

- 1. 赋值语句**
- 2. 基本输入输出语句**

# 顺序结构



*# Filename: seq.py*

string = 'Hello, World!'

print(string)

# 赋值语句

普通  
赋值

$r = 2$

增量  
赋值

$m /= 5$

链式  
赋值

$c = b = a$

多重  
赋值

$p, r = 3, 5$

# 多重赋值的本质

**S**<sub>ource</sub>

```
>>> 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

# 多重赋值



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

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

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

```
>>> x
```

```
5
```

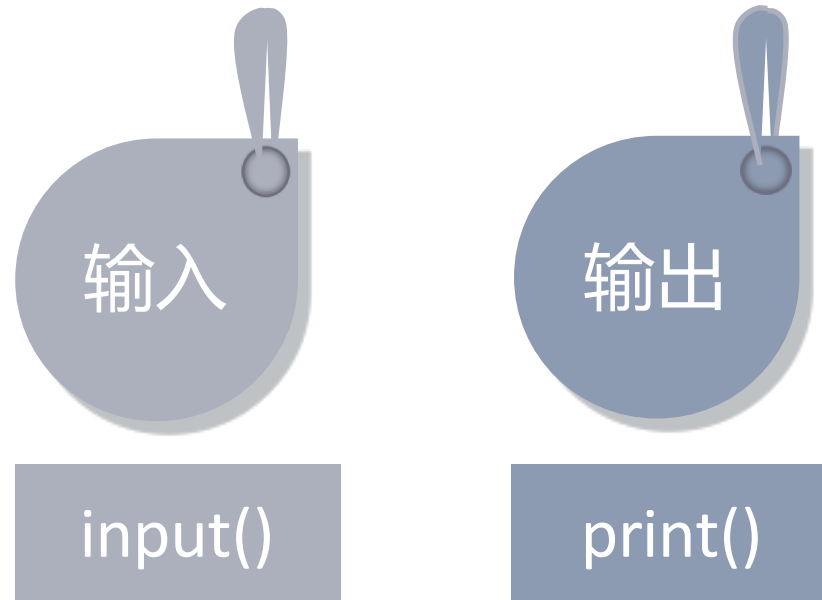
```
>>> y
```

```
3
```

语法糖  
syntactic sugar



# 输入/输出



# 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



# 数据输入—完成如下输入任务

1. 如何输入获得两个字符串？（若输入  
abc def或abc,def）
2. 如何输入获得两个整数？（若输入  
34,567）
3. 如何输入后获得一个元素均为数值型的  
列表？（若输入12,3.4,567或[12,3.4,567]）



## 数据输入—完成如下输入任务

```
>>> x, y = input('Enter two strs: ').split()    # split(',')  
Enter two strs: abc def
```

```
>>> x, y = eval(input('Enter two nums: '))  
Enter two nums: 3,4
```

```
>>> lst = list(eval(input('Enter a list: ')))  
Enter a list: 12,3.4,567
```

或

```
>>> lst = eval(input('Enter a list: '))  
Enter a list: [12,3.4,567]
```

# 输出函数print()

输出语句的一般形式：

```
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



## 数据输出—完成如下输出任务

1. 如何在输出数据中加入一个非空白分隔符？（若数据为12和345）
2. 如何换行输出所有数据？（若数据为12和345）
3. 如何将循环输出的所有数据放在同一行输出？



## 数据输出—完成如下输出任务

```
>>> x, y = 12, 345
```

```
>>> print(x, y)
```

```
12 345
```

```
>>> print(x, y, sep = ',')
```

```
12,345
```

```
>>> print(x);print(y) # 默认end参数功能为换行
```

```
>>> 循环控制条件: print(x, end = ',')
```

## 数据输出—完成如下输出任务

```
>>> x, y = 12, 345  
>>> ...  
x = 12, y = 345
```

```
>>> x, y = 12.34, 567.89  
>>> ...  
The result is 12.3 and 567.9.
```



# 输出函数print()

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) )
```

```
printf("x = %d, y = %d", x, y)
```

```
print("x = %d, y = %d" % (x, y))
```

# 输出函数print()——格式化模板

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>b</b>	二进制，以2为基数输出数字
<b>o</b>	八进制，以8为基数输出数字
<b>x</b>	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
<b>c</b>	字符，将整数转换成对应的Unicode字符输出
<b>d</b>	十进制整数，以10为基数输出数字
<b>f</b>	定点数，以定点数输出数字
<b>e</b>	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
<b>[+]m.nf</b>	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）
<b>0&gt;5d</b>	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
<b>&lt;</b>	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
<b>^</b>	居中对齐
<b>{{}}</b>	输出一个{}

# 2 人工智能程序设计 选择结构

例 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

# Filename: guessnum.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!

# if 语句

## 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）

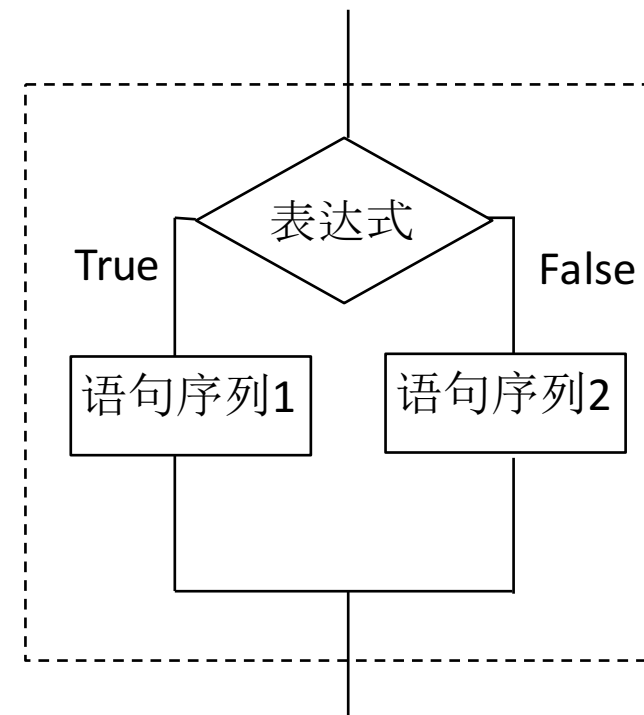
## else 语句

### 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 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: guessnum.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

## 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

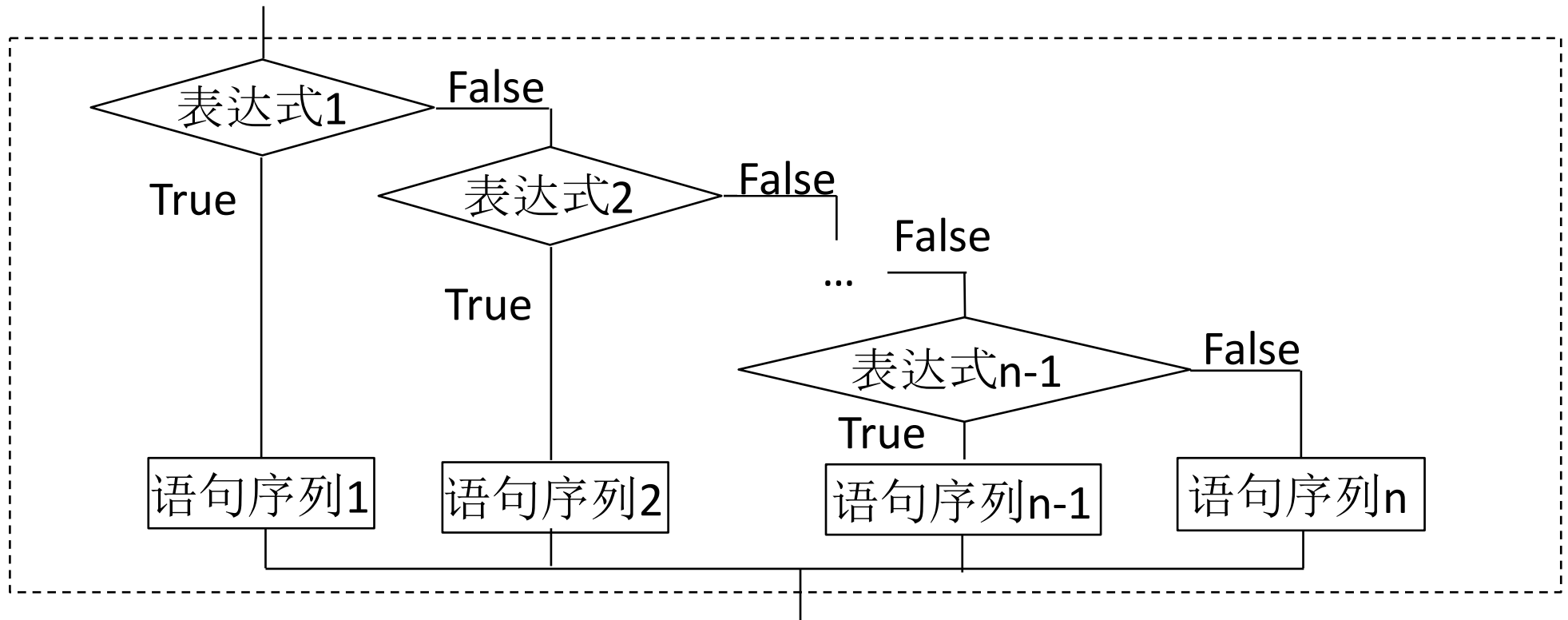
## 语句序列N-1

- 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

# elif 语句



多分支结构流程图

# 例 猜数字游戏——猜中或未猜中



```
# Filename: guessnum.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: guessnum.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.')
```

# 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
```

dangling else

## 例 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 符号函数

File

```
# Filename: prog1.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))
```

File

```
# Filename: prog2.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))
```

# 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：  
**x if C else y**

F<sub>ile</sub>

*# Filename: elsepro.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

**t = x if x >= y else y**

## 选择

1. if语句
2. else子句
3. elif子句
4. 嵌套的if语句
5. 条件表达式



# 3 人工智能程序设计 循环结构

# 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: guessnum.py
from random import randint

x = randint(0, 300)
i = 0
while i <= 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.')
    i += 1
```

# while 循环

## 语法

**While 表达式:**  
**语句序列（循环体）**

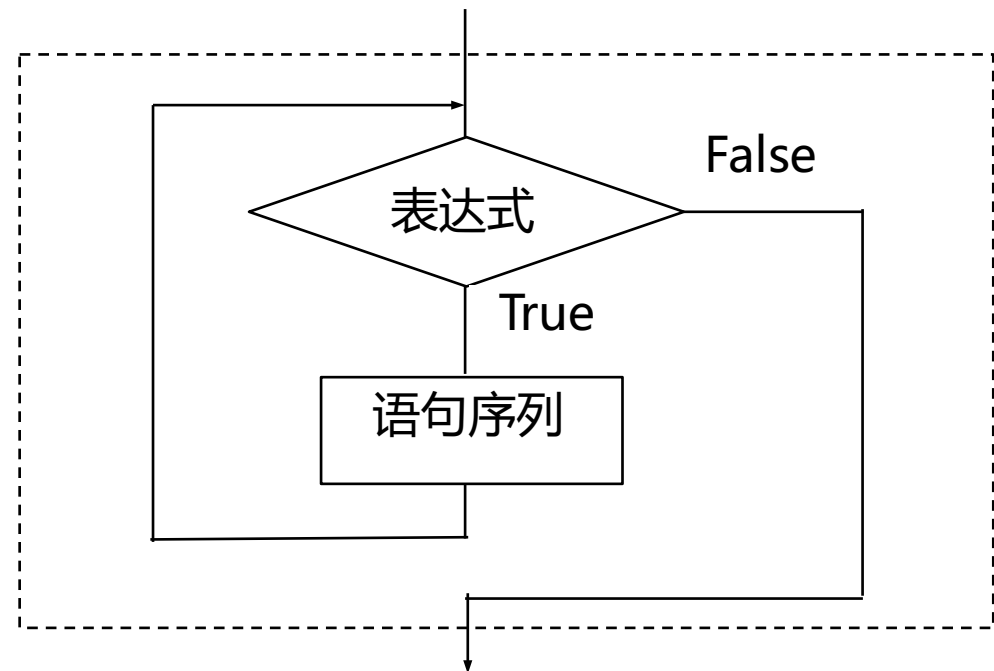
## 表达式

- 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

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。

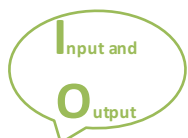


while语句流程图

# 例 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



# Filename: gcd.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 = ", y)

print("最小公倍数 = ", z // y)

## 例 计算 $\pi$

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

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.pi :  
3.141592653589793

Source

# Filename: pi.py

import math

x, s = 1, 0

sign = 1

k = 1

while math.fabs(x) > 1e-8:

    s += x

    k += 2

    sign \*= -1

    x = sign / k

s \*= 4

print("pi = {:.15f}".format(s))

# 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: guessnum.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

## 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

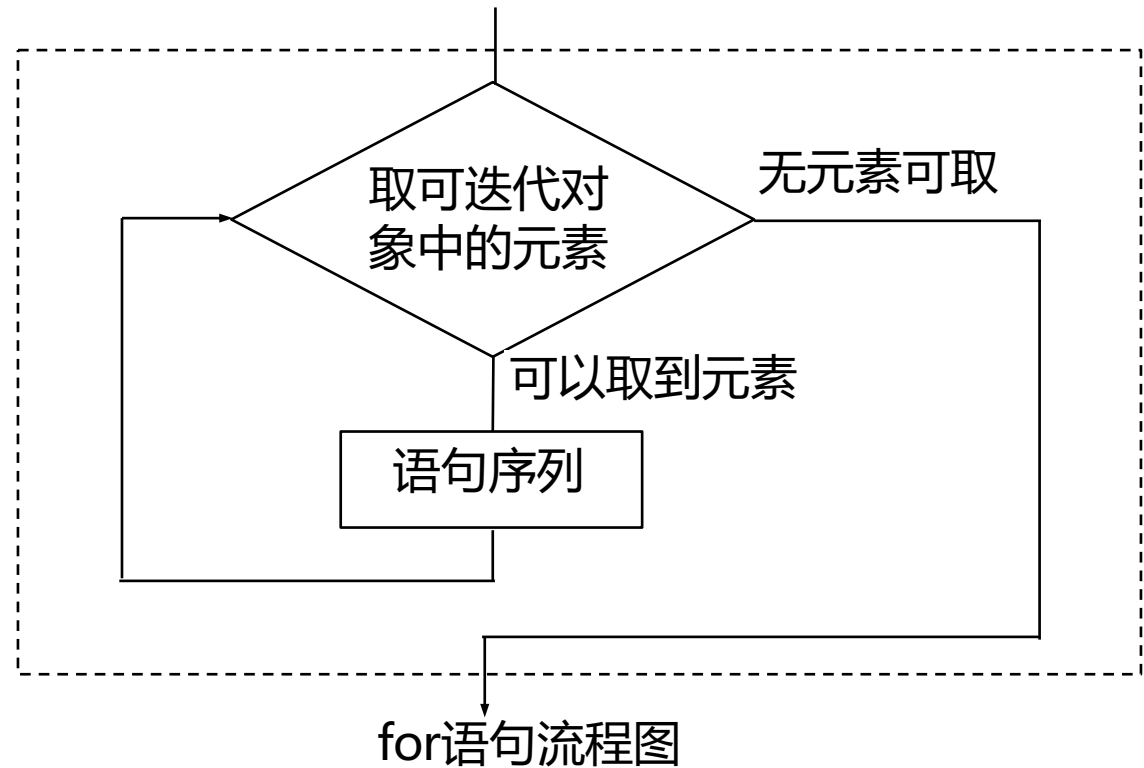
- String
- List
- Tuple
- Dictionary
- File



# for 循环

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

```
lst = ['C++', 'Python', 'Java']  
for item in lst:  
    print(item, len(item))
```



# for循环——可迭代对象

序列

01

字典的键

03

迭代器

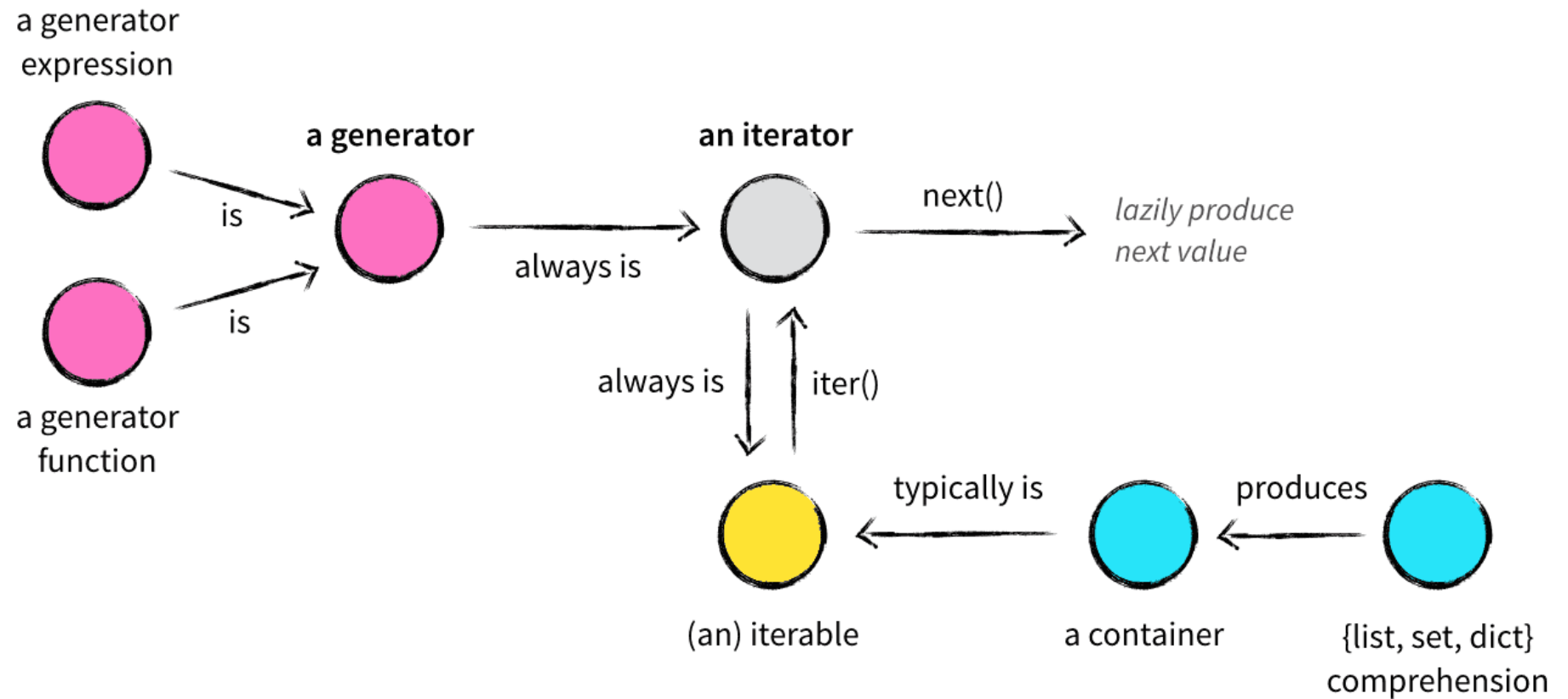
05

序列索引

02

文件的行

04



From: <https://www.pythonic.eu>

## 可迭代对象 与迭代器

```
>>> lst = [1,2,3]
>>> x = iter(lst)
>>> x
<list_iterator object at 0x000001DB4E4730F0>
>>> next(x)
1
>>> next(x)
2
>>> next(x)
3
>>> next(x)
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
StopIteration
```

for循环是语法糖吗？

```
from collections import Iterator
isinstance([1,2,3], Iterator)
```

for循环两次遍历一个列表与  
迭代器有区别吗？

# for 语句迭代——序列迭代

Source

```
>>> 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
```

Source

```
>>> 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

**S**<sub>ource</sub>

```
>>> 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

**S**ource

```
>>> 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

Source

```
>>> 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 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 输出公司代码和股票价格

用序列索引迭代

I nput and O utput

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

File

# Filename: comp1.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 输出公司代码和股票价格

用序列迭代

I nput and O utput

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

F  
ile

```
# Filename: comp2.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)
```

## 例 求斐波纳契 ( Fibonacci ) 数列前20项

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

F<sub>ile</sub>

# Filename: fib.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)

F<sub>ile</sub>

# Filename: fib.py

count = 20

i = 0

a, b = 0, 1

print(a, b, sep = '\n')

while i < count-1:

    print(b)

    a, b = b, a + b

    i += 1

I<sub>nput and</sub> O<sub>utput</sub>

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

## 例 计算 $1+2!+3!+\dots+n!$

```
n = int(input("Enter the max n: "))
i, term, s = 1, 1, 0
while i <= n:
    term *= i
    s += term
    i += 1
print(s)
```

递推法

## 例 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



Nested Loops

File

*# Filename: change.py*

```
i, j, k = 0, 0, 0
```

```
count = 0
```

```
for i in range(21):
```

```
    for j in range(51):
```

```
        k = 100 - 5 * i - 2 * j
```

```
        if k >= 0:
```

```
            count += 1
```

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

Input and Output

count = 541

## 例 输出n\*n乘法口诀表并按样例所示格式输出

File

```
# Filename: nmuln.py
```

```
n = int(input("n: "))
```

```
for i in range(1, n+1):
```

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

```
        print("{}*{}={}".format(j, i, i*j), end = ' ')
```

```
    print("")
```

Input and Output

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

## 例 寻找[1,n]之间满足条件的整数个数

借助列表



```
# Filename: findnums.py
```

```
n = int(input("n: "))
```

```
lst = []
```

```
for num in range(1, n+1):
```

```
    if num % 3 == 0 and num % 5 == 0:
```

```
        lst.append(num)
```

```
print("There are {} nums.".format(len(lst)))
```



## 例 数字筛选

输入一个2（包含）至9（包含）之间的一位数字，输出1-100中剔除了包含该数字、该数字的倍数的所有数字，输出满足条件的数，要求一行输出10个数字，数字之间用“,”分隔。

I nput and O utput

1,2,3,4,5,6,8,9,10,11  
12,13,15,16,18,19,20,22,23,24  
25,26,29,30,31,32,33,34,36,38  
39,40,41,43,44,45,46,48,50,51  
52,53,54,55,58,59,60,61,62,64  
65,66,68,69,80,81,82,83,85,86  
88,89,90,92,93,94,95,96,99,100

File

# Filename: picknums.py

```
num = int(input('Enter the number: '))
```

```
n = 0
```

```
ln = ''
```

```
for i in range(101):
```

```
    s = str(i)
```

```
    if i % num != 0 and s.find(str(num)) == -1:
```

```
        ln = ln + s + ','
```

```
        n += 1
```

```
    if n % 10 == 0:
```

```
        print(ln[:-1])
```

```
        ln = ''
```

```
print(ln[:-1])
```

按功能需求选  
择数据类型

# 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

**F**<sub>ile</sub>  
# *Filename: breakpro.py*  
s = 0  
i = 1  
**while** i < 10:  
    s += i  
    **if** s > 10:  
        **break**  
    i += 1  
**print**('i = {0:d}, sum = {1:d}'.format(i, s))

**I**<sub>nput and</sub> **O**<sub>utput</sub>  
i=5, s=15

# break 语句

while i < 10:



while True:



# Filename: breakpro.py

s = 0

i = 1

while True:

    s += i


    if s > 10:

        break

    i += 1

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

# break 语句



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

10 5

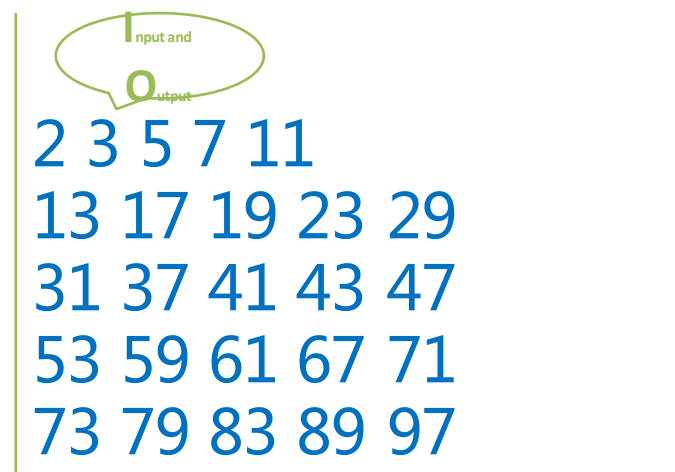
## 例 输出2~100之间的素数，每行显示5个

素数 ( prime ) :

只能被1和n自身整除的正整数n。

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。



$$2 \sim n/2$$

or

$$2 \sim \sqrt{n}$$

## 例 输出2~100之间的素数，每行显示5个

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: prime.py
from math import sqrt
j, count = 2, 0
while j <= 100:
    i = 2
    k = int(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
```

# 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

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"




# continue语句

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 = ' ')
```

## 例 输入一个整数，并判断其是否为素数

- 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



```
# Filename: prime.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 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。



```
# Filename: guessnum.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()
else:
    print('I quit and byebye!')
```