Python中，for循环能够用于任何可迭代对象，而不仅仅是序列。

列表，字典，元组，集合，字符串，文件对象均为可迭代对象。

通过for循环遍历列表，字典，元组，集合，字符串时，其实就是调用迭代器，有点类似C++ STL中的迭代器。

例：程序iterator\_test/iterator\_test1.py

from collections import Iterable  
  
# 可迭代对象  
l1 = [1, 2, 3]  
t1 = (1, 2)  
d1 = {'a': 1, 'b': 2}  
s1 = 'abc'  
set1 = {1, 2, 3}  
print(isinstance(l1, Iterable)) # True  
print(isinstance(t1, Iterable)) # True  
print(isinstance(d1, Iterable)) # True  
print(isinstance(s1, Iterable)) # True  
print(isinstance(set1, Iterable)) # True  
print()  
  
# 文件迭代器  
# 通过迭代的方式按行读取  
# 不要把整个文件都加载到内存中  
# 效率比使用readLines()要高  
with open('data.txt') as file\_obj:  
 for line in file\_obj: # 调用file\_obj.\_\_next()\_\_并catch StopIteration  
 print(line, end='')  
print()  
print()  
  
# 手动迭代  
L = [1, 2, 3]  
it1 = iter(L)  
print(it1.\_\_next\_\_()) # 1  
print(it1.\_\_next\_\_()) # 2  
print(it1.\_\_next\_\_()) # 3  
# print(it1.\_\_next\_\_()) # StopIteration

输出为：

True

True

True

True

True

abc

123

python

Hello, world

1

2

3

列表解析：

例：程序iterator\_test/iterator\_test2.py

# 解析文件  
lines = [line.rstrip().upper() for line in open('data.txt')]  
print(lines) # ['ABC', '123', 'PYTHON', 'HELLO, WORLD']  
  
# 嵌套解析  
s = [x + y for x in 'abc' for y in 'lmn']  
print(s) # ['al', 'am', 'an', 'bl', 'bm', 'bn', 'cl', 'cm', 'cn']

range，map，zip和字典迭代器

例：程序iterator\_test/iterator\_test3.py

# range  
R = range(10)  
it1 = iter(R) # range不是自身的迭代器，需要使用iter函数  
print(next(it1)) # 0  
print(next(it1)) # 1  
  
# map  
M = map(abs, (-3, -2, -1))  
print(next(M)) # 3，map是自身的迭代器  
print(next(M)) # 2  
print(next(M)) # 1  
  
Z = zip((1, 2, 3), (10, 20, 30))  
print(next(Z)) # (1, 10)，zip是自身的迭代器  
print(next(Z)) # (2, 20)  
  
# 多个迭代器 VS 单个迭代器  
# range支持多个迭代器  
# iter()返回一个新的对象  
R1 = range(1, 10)  
print(id(R1)) # 2556288144704  
print(id(iter(R1))) # 2556255252176  
  
# map, zip, filter等支持单个迭代器  
# iter()返回自身  
M1 = map(abs, (-3, -2, -1))  
print(id(M1)) # 2556288879528  
print(id(iter(M1))) # 2556288879528  
  
# 同一个range的两个迭代器互不影响  
itr1 = iter(R1)  
itr2 = iter(R1)  
print(next(itr1)) # 1  
print(next(itr2)) # 1  
  
# 同一个map的两个迭代器迭代同一个对象  
itm1 = iter(M1)  
itm2 = iter(M1)  
print(next(itm1)) # 3  
print(next(itm2)) # 2  
print()  
  
# 字典迭代器  
D = dict(a=1, b=2, c=3)  
print(D) # {'a': 1, 'b': 2, 'c': 3}  
  
# dict的keys()不是列表，可迭代  
K = D.keys()  
print(type(K)) # <class 'dict\_keys'>  
  
itk = iter(K)  
print(next(itk)) # a  
print(next(itk)) # b  
  
# dict的values()不是列表，可迭代  
V = D.values()  
itv = iter(V)  
print(next(itv)) # 1  
print(next(itv)) # 2  
  
# 字典本身的迭代器返回key  
itd = iter(D)  
print(next(itd)) # a