
图解
Python
可能是迄今为止
最易懂的Python视频教程

```
"""迭代器"""

"""
    可以用于for-in语句的对象被称为可迭代（Iterable）对象。例如：range、列表、元组、字符串、字典、集合、生成器，都是可迭代对象。
    可以调用内置函数isinstance()判断一个对象是否是可迭代对象。标准库模块collections中的类Iterable用于表示可迭代对象。
"""
from collections import Iterable

print(isinstance([1, 2, 3], Iterable))  # True
print(isinstance('abc', Iterable))      # True
print(isinstance((i * i for i in range(1, 7)), Iterable))   # True
```

```
"""
    如果一个可迭代对象可以作为内置函数next()的实参从而支持惰性推算，那么该对象被称为
迭代器（Iterator）对象。
    对于range、列表、元组、字符串、字典和集合等可迭代对象，都不可以作为内置函数next()的实参，
而生成器可以。所以，生成器是迭代器的一种。
    可以调用内置函数isinstance()判断一个对象是否是迭代器对象。标准库模块collections中的
类Iterator用于表示迭代器对象。
"""
from collections import Iterator

L = [1, 2, 3]
s = 'abc'

# next(L) # TypeError: 'list' object is not an iterator
# next(s) # TypeError: 'str' object is not an iterator

print(isinstance(L, Iterator))  # False
print(isinstance(s, Iterator))  # False
print(isinstance((i * i for i in range(1, 7)), Iterator))   # False
```

```
"""
    可以调用内置函数iter()把不支持惰性推算的可迭代对象转换为迭代器对象。
"""
iter_L = iter(L)
iter_s = iter(s)

print(isinstance(iter_L, Iterator)) # True
print(isinstance(iter_s, Iterator)) # True

print(next(iter_L)) # 1
print(next(iter_s)) # a
```