

# Итераторы в Python



## Выбирайте работу уверенно



Отметка «Компания проверена» присваивается работодателям, которые были проверены Авито по определенным параметрам. Отметка видна в результатах поиска и в самих объявлениях. Подробнее на [https://www.avito.ru/employer/verified\\_employer](https://www.avito.ru/employer/verified_employer). Демонстрируемые объявления являются вымышленными.

12+

 **avito.ru** РЕКЛАМА · 16+

Выбрать

[Справочник по языку Python3.](#) / Итераторы в Python

Большинство объектов - контейнеров можно зациклить, используя [инструкцию for ... in](#):

```
for element in [1, 2, 3]:
    print(element)

for element in (1, 2, 3):
    print(element)

for key in {'one':1, 'two':2}:
    print(key)

for char in '123':
    print(char)

for line in open('myfile.txt'):
    print(line, end='')
```

Этот стиль доступа понятен, лаконичен и удобен. [Использование итераторов](#) пронизывает и объединяет Python. За кулисами оператор for вызывает функцию [iter\(\)](#) для объекта контейнера. Функция возвращает объект итератора, который определяет метод `__next__()`, который, в свою очередь обращается к элементам в контейнере по одному за раз. Когда нет больше элементов, `__next__()` возбуждает [исключение StopIteration](#), которое указывает циклу о завершении.

Простой [итератор](#) можно создать, применив к [последовательности](#) встроенную [функцию iter\(s\)](#). Что бы получить элемент итератора, необходимо вызвать метод `__next__()` с помощью встроенной функции [next\(\)](#).

Пример показывает, как все это работает:

```
>>> s = 'abc'
>>> it = iter(s)
>>> it
# <iterator object at 0x00A1DB50>
>>> next(it)
# 'a'
>>> next(it)
# 'b'
>>> next(it)
# 'c'
>>> next(it)
# Traceback (most recent call last):
#   File "<stdin>", line 1, in <module>
#     next(it)
# StopIteration
```



