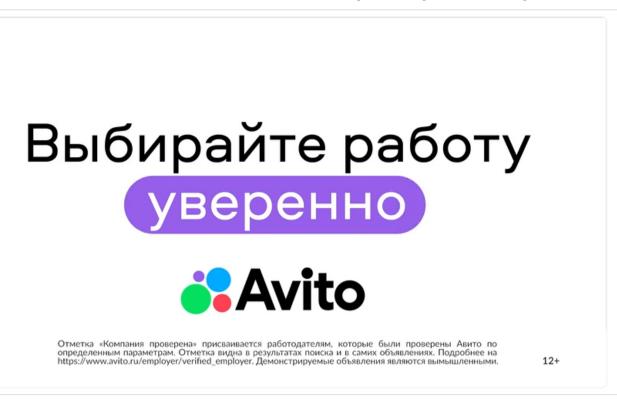
13.09.2023, 23:40 Итераторы в Python

Итераторы в Python





Сообщить об ошибке.

Выбирайте работу уверенно на Авито.

Проверенные работодатели получают отметку «Компания проверена».

Выбрать

<u>Справочник по языку Python3.</u> / Итераторы в Python

Большинство объектов - контейнеров можно зациклить, используя <u>инструкцию for ... in</u>:

```
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. <u>Использование итераторов</u> пронизывает и объединяет Python. За кулисами оператор for вызывает функцию <u>iter()</u> для объекта контейнера. Функция возвращает объект итератора, который определяет метод <u>next</u>(), который, в свою очередь обращается к элементам в контейнере по одному за раз. Когда нет больше элементов, <u>next</u>() возбуждает <u>исключение StopIteration</u>, которое указывает циклу о завершении.

Простой <u>итератор</u> можно создать, применив к <u>последовательности</u> встроенную <u>функцию iter(s)</u>. Что бы получить элемент итератора, необходимо вызвать метод __next__() с помощью встроенной функции <u>next()</u>.

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
```

13.09.2023, 23:40 Итераторы в Python

Обратите внимание, что после использования итератор it остался пустым и его невозможно больше использовать. Итераторы не поддерживают <u>извлечение срезов</u> и <u>получение элемента по индексу</u>. К ним так же нельзя применить <u>функцию len()</u> (узнать длину итерации).

Поведение итератора в классах.

Посмотрев на механику протокола итератора, легко добавить поведение итератора в классы. Для этого, определите метод __iter__(), который возвращает объект с методом __next__(). Если класс определяет метод __next__(), то __iter__() может просто вернуть self:

```
class Reverse:
    """Итератор для циклического перебора последовательности
            в обратном направлении.
    """

def __init__(self, data):
            self.data = data
            self.index = len(data)

def __iter__(self):
            return self

def __next__(self):
            if self.index == 0:
                raise StopIteration
            self.index = self.index - 1
            return self.data[self.index]
```

```
>>> rev = Reverse('spam')
>>> iter(rev)

#<__main__.Reverse object at 0x00A1DB50>
>>> for char in rev:
... print(char)
...

# m
# a
# p
# s
```

ХОЧУ ПОМОЧЬ ПРОЕКТУ



13.09.2023, 23:40 Итераторы в Python

<u>DOCS-Python.ru</u>™, 2023 г.

(Внимание! При копировании материала ссылка на источник обязательна)

@docs_python_ru