Реферат на тему:

Итераторы в Python

Студентка: Васильева Елизавета Дмитриевна

Группа: 2.1

17.09.2024

Глава 1

Введение

Итераторы — это ключевой концепт в Python, который позволяет создавать и работать с последовательностями объектов. Они являются основой многих встроенных структур данных, таких как списки, кортежи и множества, а также используются для создания пользовательских структур данных.

Глава 2

Основная часть - 1

2.1. Раздел 1.1

Итератор — это объект, который реализует два метода: iter и next. Метод iter возвращает сам итератор, а метод next возвращает следующий элемент в последовательности. Если в последовательности больше нет элементов, метод next должен вызвать исключение StopIteration. Пример итератора изображен на картинке 2.1.

```
class MyIterator:
    def __init__(self, max_value):
        self.max_value = max_value
        self.current_value = 0

    def __iter__(self):
        return self

    def __next__(self):
        if self.current_value &lt; self.max_value:
            self.current_value += 1
            return self.current_value
        else:
            raise StopIteration

my_iter = MyIterator(5)
for i in my_iter:
    print(i)
```

Рис. 2.1: Пример простого итератора

2.1.1. Подраздел 1.1.1

В Python итераторы используются с помощью ключевого слова *for*. Оно позволяет перебирать элементы последовательности, вызывая метод next для каждого элемента до вызова исключения StopIteration. Итераторы в Python можно использовать не только для встроенных структур данных, но и для пользовательских. Это позволяет создавать более сложные и гибкие структуры данных, которые могут быть обработаны с помощью стандартного синтаксиса *for*.

2.1.2. Подраздел 1.1.2

Стоит отдельно остановиться на том, что цикл *for*, в Python, устроен несколько иначе, чем в большинстве других языков. Он больше похож на *for...each*, или же *for...of*. Если же, мы перепишем цикл for с помощью цикла while, используя индексы, то работать такой подход будет только с последовательностями. А с итерируемыми объектами, последовательностями не являющимися, не будет.

2.1.3. Подраздел 1.1.3

Теперь формализуем протокол итератора целиком:

1. Чтобы получить итератор мы должны передать функции iter итерируемый объект.
2. Далее мы передаём итератор функции next.
3. Когда элементы в итераторе закончились, порождается исключение StopIteration.

Особенности:

1. Любой объект, передаваемый функции iter без исключения TypeError — итерируемый объект.

2. Любой объект, передаваемый функции next без исключения TypeError — итератор.

3. Любой объект, передаваемый функции iter и возвращающий сам себя — итератор.

Плюсы итераторов: Итераторы работают ”лениво” . А это значит, что они не выполняют какой-либо работы, до тех пор, пока мы их об этом не попросим. Таким образом, мы можем оптимизировать потребление ресурсов ОЗУ и CPU, а так же создавать бесконечные последовательности.

2.2. Раздел 2

Для сравнения удобства различных моделей итераторов на практике были реализованы итераторы для нескольких типовых задач и результаты этого сравнения свели в следующую таблицу 2.1 («+» означает «удобно», «−» — неудобно и «0» — не очень удобно.)

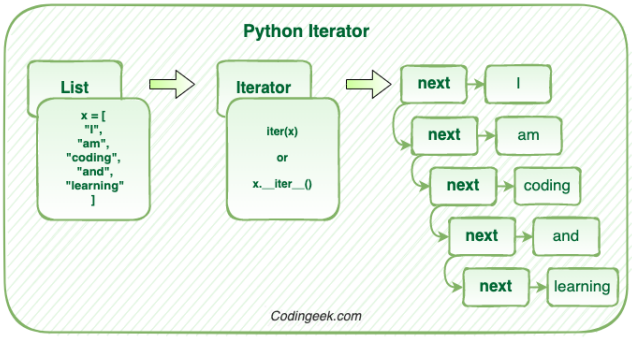
Язык	Массив/Диапозон	Связный список	Чтение файла по строкам
C	+	0	0
Python	+	0	+
Java	+	0	-
yield	+	+	+

Таблица 2.1: Сравнительная таблица

Глава 3

Заключение

Итераторы — это мощный инструмент в Python, который позволяет работать с последовательностями объектов. Они используются для перебора элементов встроенных структур данных, таких как списки и кортежи, а также для создания пользовательских структур данных. Используйте итераторы для упрощения работы с последовательностями объектов и для повышения гибкости вашего кода.



Оглавление

1 Введение **1**

2 Основная часть - 1 **2**

2.1 Раздел 1.1 2

2.1.1 Подраздел 1.1.1 3

2.1.2 Подраздел 1.1.2 3

2.1.3 Подраздел 1.1.3 3

2.2 Раздел 2 4

3 Заключение **5**

3.1 Список литературы 6

3.1. Список литературы

1. Статья на Habr

2. Статья Алексея Кодова