top

КОМПЬЮТЕРНАЯ АКАДЕМИЯ

Основы Python

Тема «Итераторы»

Итераторы — это инструмент Python, который позволяет перебирать элементы коллекций (например, списков, строк или множеств) по одному. Итераторы особенно полезны, когда мы работаем с большими данными, так как они загружают элементы только по мере необходимости, не занимая лишнюю память.

Представьте, что читаете книгу по одной странице, а не разворачиваете сразу все страницы. Итераторы работают по такому же принципу: они позволяют получать следующий элемент из коллекции, не загружая всю коллекцию сразу.

Сегодня мы:

- **узнаем**, что такое итераторы и чем они отличаются от итерируемых объектов.
- **научимся** использовать функции iter() и next() для работы с итераторами.
- **разберемся**, как Python автоматически применяет итераторы в циклах for.
- **поймем**, как использовать итераторы для экономии ресурсов при работе с большими коллекциями данных.

Глоссарий к восьмому занятию

Iterator — Объект, который позволяет последовательно перебирать элементы коллекции.

Iterable — Объект, из которого можно создать итератор (например, список или строка).

Iterate — Процесс последовательного получения элементов из коллекции.

iter() — Встроенная функция Python, которая возвращает итератор из итерируемого объекта.

next() — Встроенная функция Python, которая возвращает следующий элемент из итератора.

Глоссарий к восьмому занятию

StopIteration — Ошибка, которая возникает, когда элементы итератора заканчиваются.

Lazy Evaluation — Техника, при которой значения создаются по мере необходимости, а не сразу.

For loop — Конструкция Python, которая автоматически использует итераторы для перебора элементов.

Итераторы

Итератор — это объект в Python, который позволяет перебирать элементы коллекции (например, списка, строки или множества) по одному за раз. Представьте, что у нас есть пачка карт. Мы можем вытягивать по одной карте, не беря их все сразу. Так работает итератор — он «вытаскивает» следующий элемент из коллекции, пока они не закончатся.

Что такое итерируемый объект?

Итерируемый объект (iterable) — это что-то, из чего можно создать итератор.

Примеры итерируемых объектов:

- Списки: [1, 2, 3]
- Строки: "hello"
- Множества: {1, 2, 3}
- Кортежи: (4, 5, 6)

Как работает итератор?

Чтобы превратить итерируемый объект в итератор, используют встроенную функцию **iter()**.

Чтобы получить следующий элемент из итератора — **next()**.

Цикл for и итераторы

В Python мы часто используем итераторы, даже не подозревая об этом.

Например, **цикл for** автоматически превращает итерируемый объект в итератор и перебирает его с помощью **next().**

Что произойдет, если элементы закончатся?

Когда итератор возвращает все элементы, вызов next() вызывает исключение **StopIteration**.

Когда итераторы могут быть полезны?

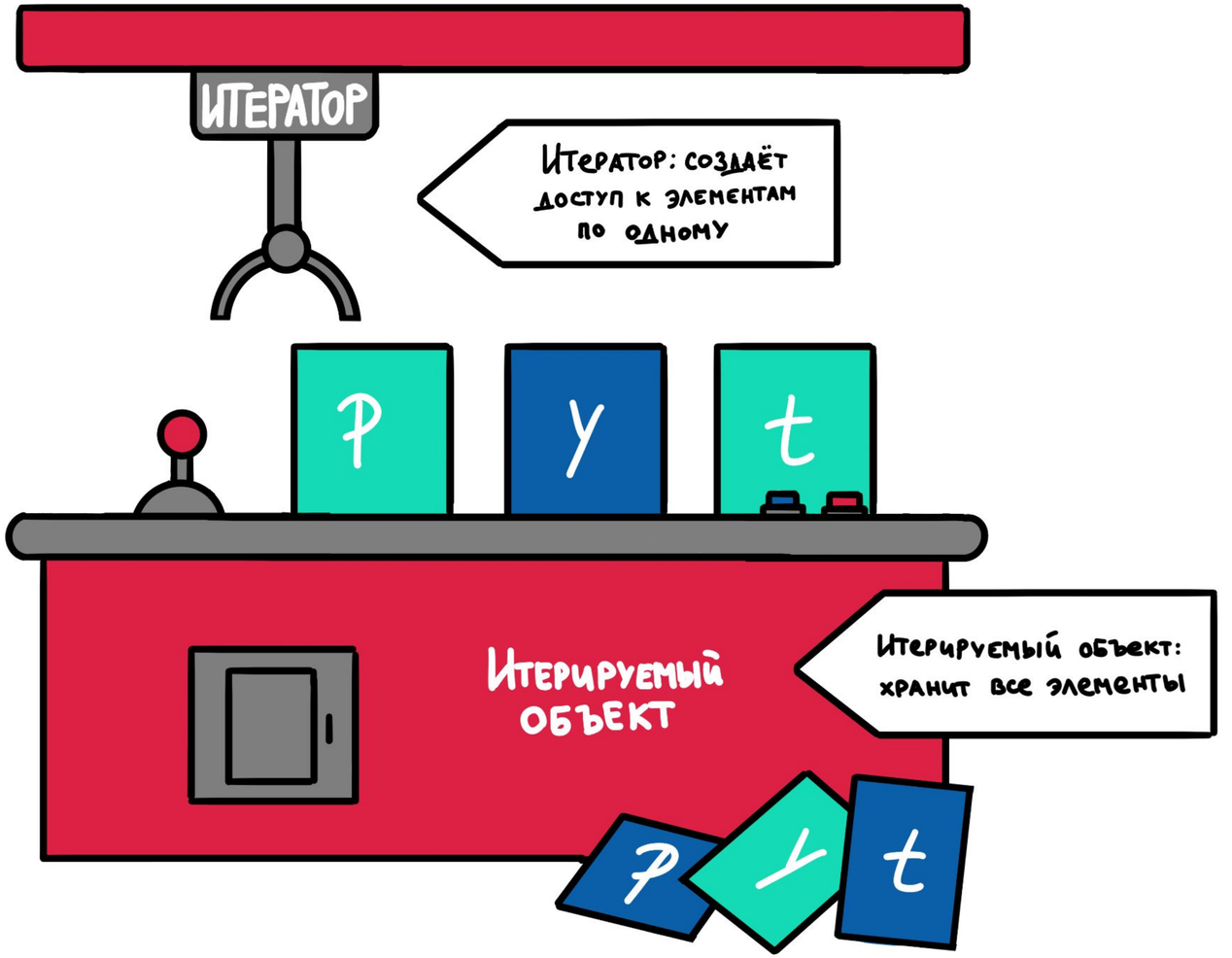
Представьте, что у нас есть огромный файл данных — список студентов с их оценками. Файл настолько большой, что он не помещается в оперативную память целиком. Мы хотим вывести на экран первые 10 строк этого файла, чтобы быстро проверить, что данные загружены правильно.

Используя итераторы, мы можем обрабатывать строки файла по одной, не загружая весь файл в память. Это эффективно и безопасно.

Итераторы ≠ итерируемые объекты

Итератор — это объект, который позволяет перебирать элементы итерируемого объекта по одному.

Итерируемый объект — это коллекция данных (список, строка, множество), которую можно перебрать.



top
КОМПЬЮТЕРНАЯ АКАДЕМИЯ

Практикум

Задача 1: Перебор данных из списка заказов

Ситуация: мы работаем над приложением для кафе. У нас есть список заказов клиентов, и мы хотим обработать их по одному, чтобы избежать перегрузки системы при обработке всех заказов сразу.

Задача: создайте программу, которая:

- Использует итератор для последовательного перебора списка заказов.
- Выводит каждый заказ по одному с помощью next().
- Обрабатывает ошибку StopIteration, если заказы закончились.

Задача 2: Перебор шагов инструкции

Ситуация: мы работаем над приложением для кухни, где повара должны выполнять шаги рецепта по одному. Нам нужно создать программу, которая помогает повару видеть текущий шаг инструкции, а после завершения всех шагов сообщает, что рецепт закончен.

Задача: создайте программу, которая:

- Использует итератор для последовательного перебора шагов рецепта.
- Выводит текущий шаг по одному с помощью next().
- Обрабатывает ошибку StopIteration, если шаги закончились.

Итоги занятия

Сегодня мы разобрались:

- что такое **итераторы** и **итерируемые объекты**.
- **как использовать** функции iter() и next() для работы с итераторами.
- как Python автоматически **использует итераторы в циклах for.**
- **как обрабатывать ошибку StopIteration** при использовании итераторов вручную.
- **почему итераторы удобны** для работы с большими данными и как они помогают экономить память.

Домашняя работа

Описание задачи: создайте небольшую программу для управления списком дел. Пользователь может выполнять задачи по одной, используя итератор. После выполнения всех задач программа сообщает, что всё выполнено. Эта задача поможет вам лучше понять, как использовать итераторы для последовательного перебора данных.

Указания:

1. Программа начинается с заранее заданного списка задач, например: ["Сделать домашнюю работу", "Помыть посуду", "Прочитать книгу"].
2. Преобразуйте этот список задач в итератор с помощью функции iter().
3. Программа должна предлагать пользователю ввести команду:
   - next — выполнить следующую задачу.
   - list — показать все задачи, которые остались невыполненными.
   - exit — выйти из программы.

Указания:

4. Если пользователь выбирает next, программа должна выводить текущую задачу.

5. Если задачи закончились, нужно вывести сообщение: "Все задачи выполнены".

6. Программа должна корректно обрабатывать ситуацию, когда все задачи уже выполнены (ошибка StopIteration не должна приводить к завершению программы).

7. Программа должна быть удобной и понятной для пользователя.

Указания:

Нужно учесть:

1. Если пользователь вводит команду, которая не поддерживается (next, list, exit), программа должна сообщить об этом.
2. Убедитесь, что программа "понимает", когда задач больше нет, и корректно завершает работу.

Ожидаемый результат:

Программа позволяет пользователю:

1. Перебирать задачи по одной.
2. Проверять оставшиеся задачи.
3. Завершить выполнение, как только все задачи сделаны.

Критерии оценивания:

4 балла — Корректность: Итератор корректно перебирает задачи.
После завершения всех задач программа сообщает, что они выполнены.

4 балла — Понятность программы: Пользователь может легко понять, какие команды доступны, и как их вводить.
Программа предоставляет понятные и дружелюбные сообщения.

4 балла — Обработка ошибок: Программа не завершает работу с ошибками (например, если пользователь ввёл некорректную команду или задачи закончились).

Максимальный балл за задание — 12 баллов.