**Лабораторная работа №3**

РАБОТА С ИТЕРАТОРАМИ, ГЕНЕРАТОРАМИ. РАБОТА С ГЕНЕРАТОРНЫМИ ВЫРАЖЕНИЯМИ

**Вариант 12.**

**Цель работы:** изучить понятия итератора и генератора в Python, а также их преимущества; ознакомиться с примерами их пользования.

**Краткая теория**

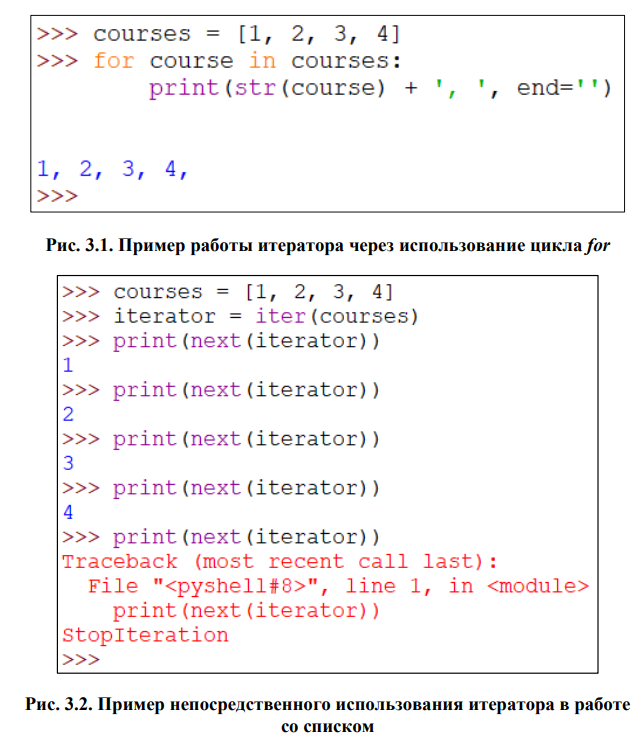
**Итераторы** – популярный поведенческий паттерн проектирования для последовательного обхода коллекции, который позволяет не раскрывать их внутреннего представления.

**Итерируемый объект** – это такой объект, от которого можно получить итератор. В Python итерируемым объектом является такой объект, от которого встроенная функция iter() возвращает итератор.

**Итератором в Python является объект**, который реализует метод \_\_next\_\_ без аргументов и метод \_\_iter\_\_. Метод - \_\_next\_\_ должен вернуть следующий элемент или ошибку StopIteration.

Преимущества использования итераторов, как было сказано выше, заключается в возможности «указывать» на определенный объект коллекции и при этом скрывать его структуру. Все последовательности (list, tuple, range) в Python являются итерируемыми объектами.

Основным местом работы с итераторами в данной лабораторной работе будет использование цикла for. Например, при переборе элементов списка или другой последовательности, используя цикл for, фактически происходит обращение к итератору данной последовательности с просьбой выдать следующий элемент. Когда элементы в последовательности заканчиваются, очередное обращение к следующему объекту итератора сгенерируют исключение, которое при использовании цикла for обрабатывается незаметно для пользователя. На рис. 3.1 представлен обход списка с помощью цикла for, а на рис. 3.2 непосредственное использование итератора в работе со списком.

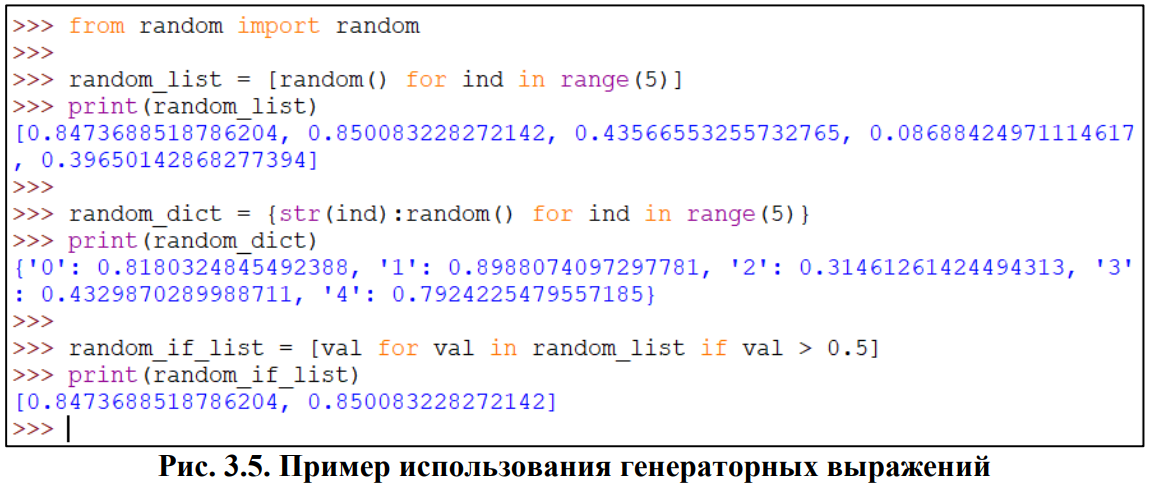


**Генератор (генераторная функция)** – это функция, которая возвращает подвид итератора, генерирующий значения. Основное их отличие в том, что они экономят память за счет того, что хранят не все значения, например, списка, а только его предыдущий элемент, предел и формулу, по которой рассчитывается следующий элемент. Данная функция вместо return содержит ключевое слово yield, которое возвращает объект-генератор, а 42 не выполняет сразу весь код. В Python имеется возможность создавать генераторный объект в сокращенной форме. Для этого используются круглые скобки [11].

Для того, чтобы получить итератор последовательности, необходимо в качестве аргумента x встроенной функции iter(x) 41 передать эту последовательность. Затем для получения следующего элемента последовательности с помощью другой встроенной функции next(iter) в качестве аргумента iter передавать полученный итератор.

В Python вводится такое понятие как **list comprehension**, которое в русскоязычном литераторе встречается в виде генераторных выражений или генераторов списков. В данном пособии будем употреблять первый вариант.

Основная задача использования генераторных выражений – это быстрое создание и заполнение списков, словарей или множеств. Отметим, что генераторные выражения – это всего лишь «синтаксический сахар», иначе говоря, они не вносят никаких дополнительных преимуществ, кроме как удобство. Пример использования генераторных выражений списков и словарей представлен на рис. 3.5.



Из рис. 3.5 видно, что для создания списка с помощью генераторного выражения требуется использовать конструкцию, похожую на сокращенную форму генераторной функции за исключением того, что используются квадратные скобки вместо круглых. Для генерации словаря используются фигурные скобки и требуется генерировать пару ключ-значение.

Также в генераторном выражении можно использовать условия. Это отражено с помощью списка random\_if\_list на рис. 3.5. Элемент добавляется в создающийся список только, если соответствует заданному условию.

Таким образом, генераторы являются мощной языковой конструкцией, которая не только упрощает написание кода, но и снижает потребление памяти и мощности ресурсов процессора. Однако не стоит злоупотреблять генераторными выражениями, т.к. в некоторых случаях это может сильно снизить читаемость кода.

**Задание 12.** **Вариант 12.**

Написать функцию, которая принимает список списков и с помощью генераторного выражения создает и возвращает новый список, который содержит все элементы входящих списков.

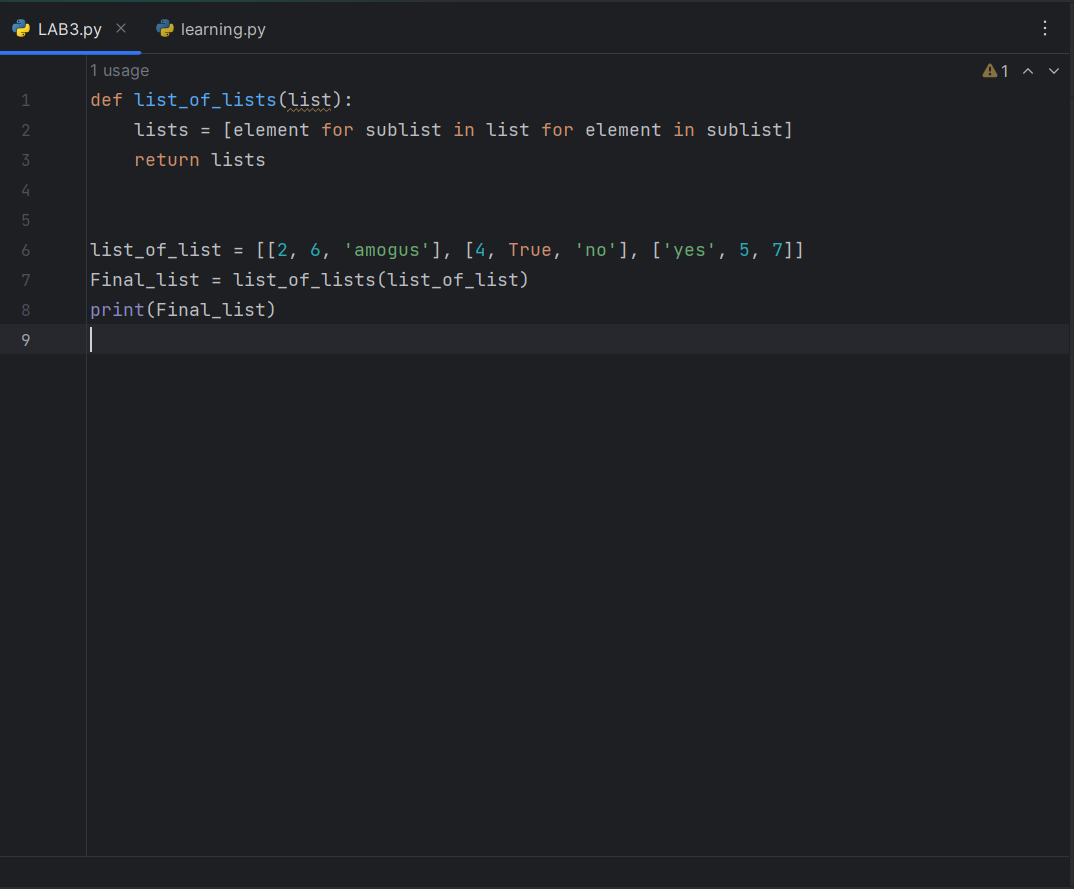


Рисунок 1.1 – код программы.

Результат работы программы представлен на рисунках 1.2

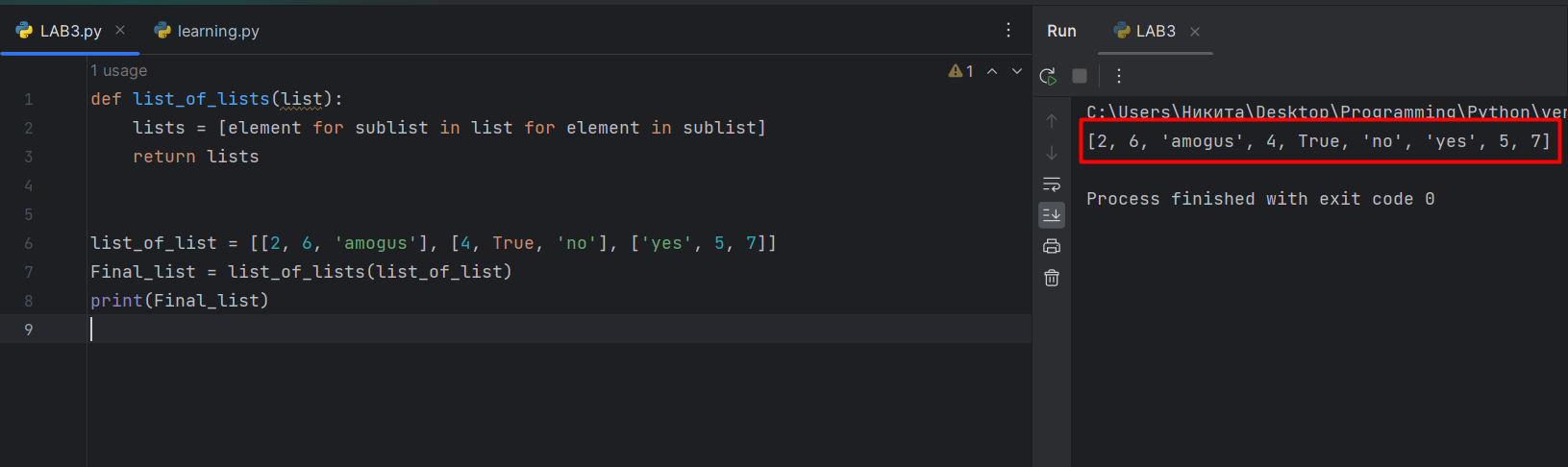


Рисунок 1.2 – результат работы программы.

**Вывод**: в результате проделанной были изучены основные способы взаимодействия с итераторами, генераторами, а также генераторными выражениями

|  |  |
| --- | --- |
| Выполнили | Свичко Д.Е. |
| Проверил | Елкин Н.С. |