**Лабораторная работа № 3**

**РАБОТА С ИТЕРАТОРАМИ, ГЕНЕРАТОРАМИ. РАБОТА С ГЕНЕРАТОРНЫМИ ВЫРАЖЕНИЯМИ**

**Вариант 10.**

**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.

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

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.

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.

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.

Перебрать генератор можно используя цикл for как и при работе с итераторами. Однако стоит заметить, что перебрать второй раз генератор не получится, потому что объектгенератор уже сгенерировал данные по описанной в нем формуле. Поэтом генераторы стоит использовать, когда требуется один раз пройтись по итерируемому объекту.

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.

**Задание.** **Вариант 10.**

Написать функцию, которая принимает список, состоящий из n элементов, и целое число x и с помощью генераторной функции выводит на консоль только те элементы входящего списка, которые больше числа x.

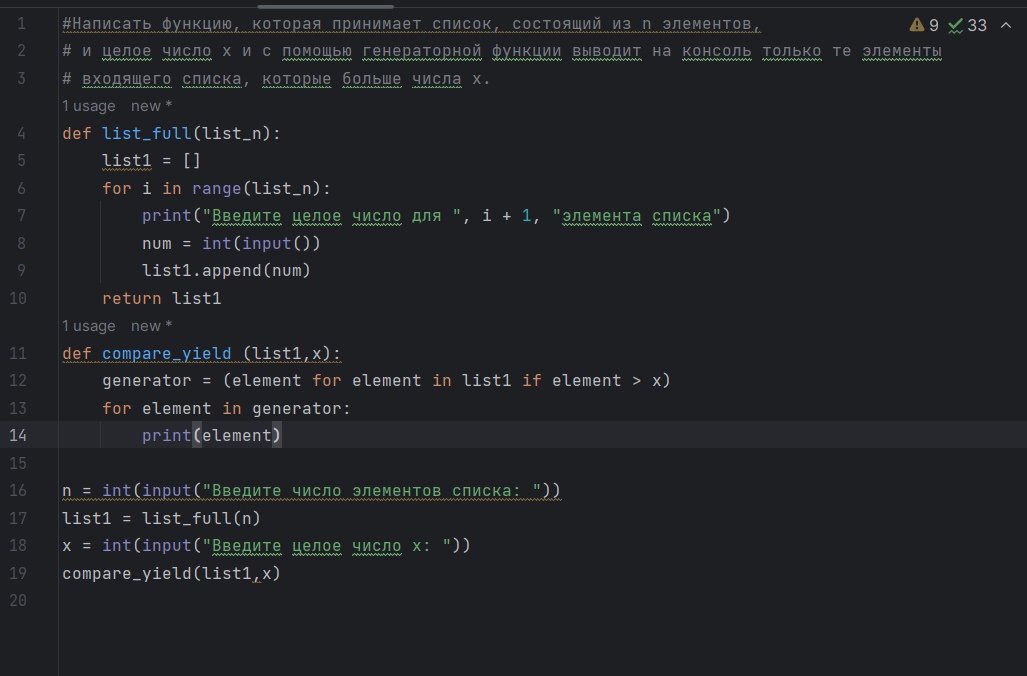


Рисунок 3.1 – код программы.

Результат работы программы представлен на рисунках 3.2

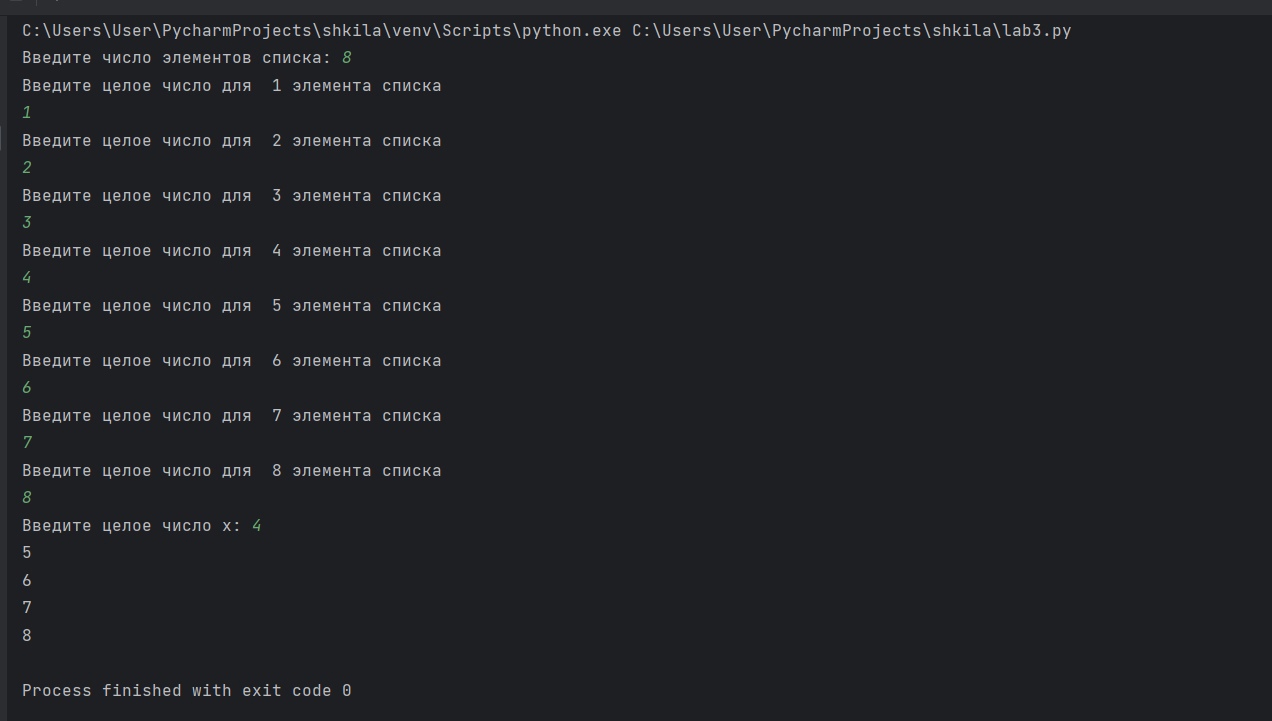


Рисунок 3.2 – результат работы программы

**Вывод**: в результате проделанной работы изучили понятия итератора и генератора в Python, а также их преимущества; ознакомились с примерами их пользования.

|  |  |
| --- | --- |
| Выполнили | Рогов М.А. |
| Проверил | Елкин Н.С. |