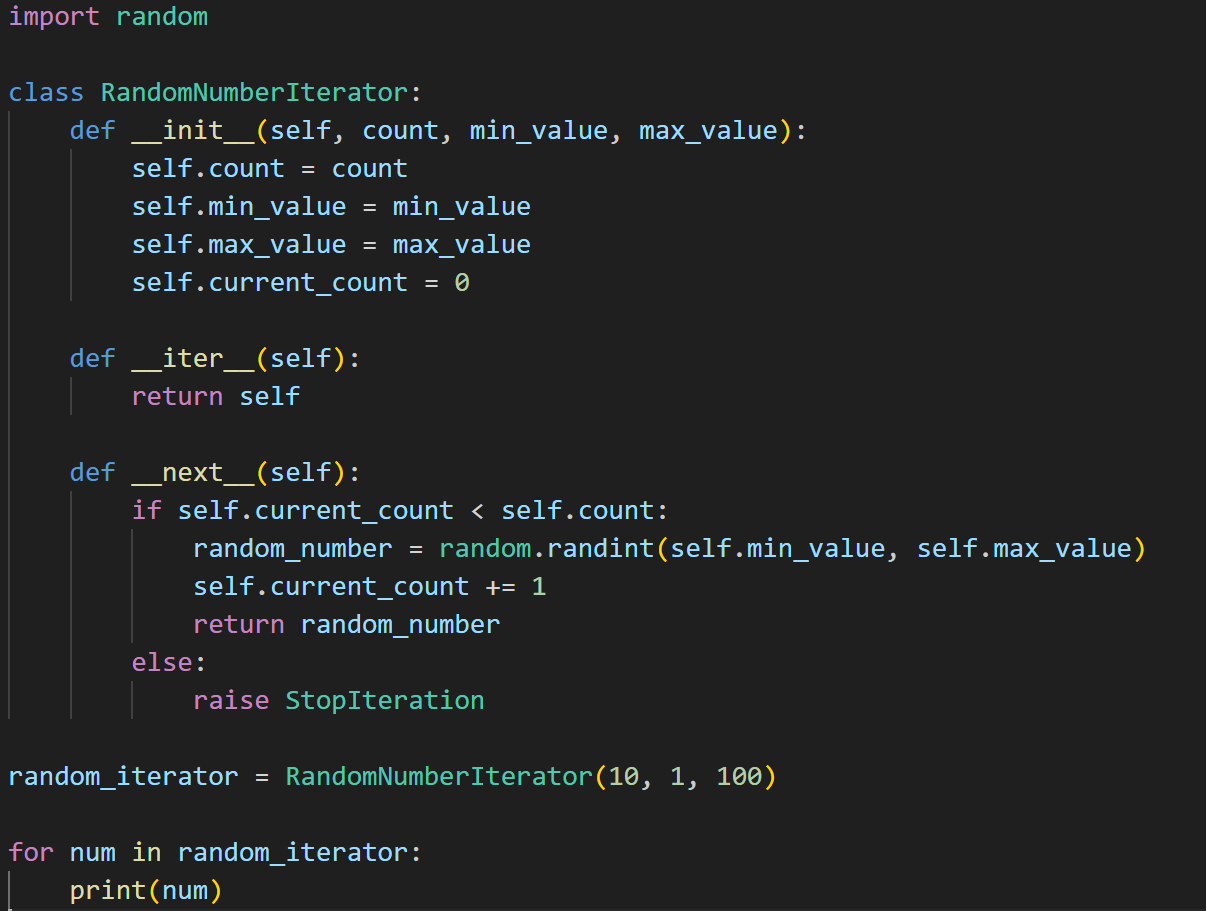
**Задача 1.**

****

**1. Назначение программы**

Программа реализует **итератор для генерации случайных чисел** в заданном диапазоне. Код демонстрирует:

* Создание пользовательского итератора с использованием методов \_\_iter\_\_ и \_\_next\_\_.
* Генерацию последовательности случайных чисел.
* Ограничение количества чисел параметром count.

**2. Описание работы кода**

**2.1. Класс RandomNumberIterator**

**Назначение:** Генерация случайных чисел в рамках заданных условий.

**Атрибуты класса:**

* count (int): Количество чисел для генерации.
* min\_value (int): Минимальное значение числа.
* max\_value (int): Максимальное значение числа.
* current\_count (int): Счетчик сгенерированных чисел.

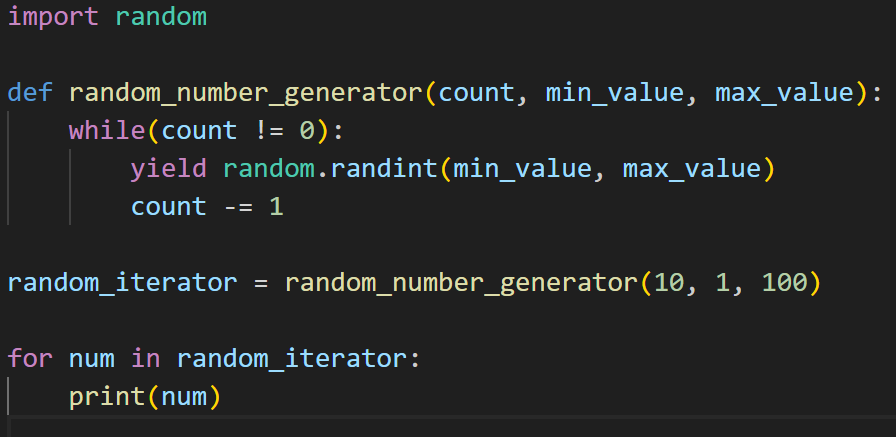
**Методы класса:**

1. **\_\_iter\_\_**
   * Возвращает сам объект итератора.
2. **\_\_next\_\_**
   * Генерирует случайное число в диапазоне [min\_value, max\_value] через random.randint().
   * Увеличивает счетчик current\_count.
   * Если достигнут лимит (current\_count >= count), вызывает исключение StopIteration.

**2.2. Взаимодействие с итератором**

1. Создается объект random\_iterator с параметрами:
   * count=10 (10 чисел),
   * min\_value=1,
   * max\_value=100.
2. Итератор используется в цикле for для вывода всех сгенерированных чисел.

**Задача 2.**



**1. Назначение программы**

Программа реализует **генератор случайных чисел** в заданном диапазоне с использованием функции-генератора. Код демонстрирует:

* Использование ключевого слова yield для создания итератора.
* Генерацию последовательности случайных чисел с ограничением по количеству.
* Упрощенный подход к работе с итерируемыми объектами.

**2. Описание работы кода**

**2.1. Функция random\_number\_generator**

**Назначение:** Генерация случайных чисел в рамках заданных параметров.

**Аргументы:**

* count (int): Количество чисел для генерации.
* min\_value (int): Минимальное значение числа.
* max\_value (int): Максимальное значение числа.

**Логика работы:**

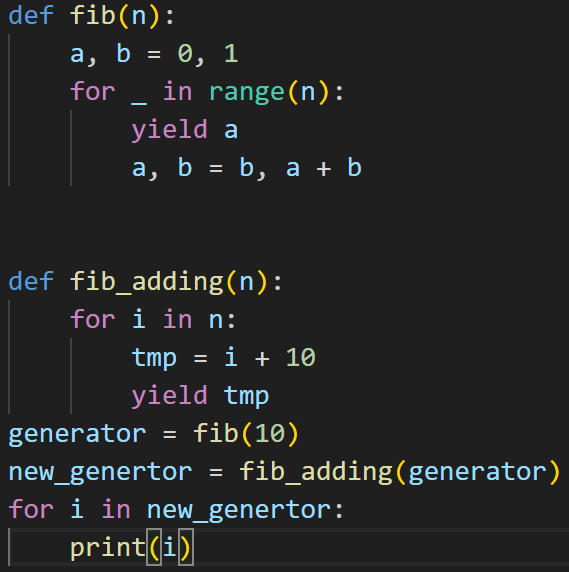
1. Цикл while выполняется, пока count не станет равен 0.
2. На каждой итерации:
   * Генерируется случайное число через random.randint(min\_value, max\_value).
   * Используется yield для возврата числа и приостановки работы функции до следующего вызова.
   * Счетчик count уменьшается на 1.

**Особенность:**  
Функция является **генератором** — она возвращает итератор, который генерирует значения "на лету", не храня всю последовательность в памяти.

**2.2. Взаимодействие с генератором**

1. Создается объект random\_iterator с параметрами:
   * count=10 (10 чисел),
   * min\_value=1,
   * max\_value=100.
2. Итератор используется в цикле for для вывода всех сгенерированных чисел.

**Задача 3.**

****

**1. Назначение программы**

Программа демонстрирует работу с **генераторами** для:

1. Генерации последовательности чисел Фибоначчи.
2. Модификации элементов последовательности (добавление числа 10 к каждому элементу).  
   Код иллюстрирует цепочку генераторов, где результат первого используется как вход для второго.

**2. Описание работы кода**

**2.1. Функция fib(n)**

**Назначение:** Генерация первых n чисел Фибоначчи.

**Логика работы:**

1. Инициализация начальных значений: a = 0, b = 1.
2. Цикл на n итераций:
   * Возврат текущего значения a через yield.
   * Обновление значений: a, b = b, a + b.

**Пример:**  
Для n = 5 последовательность: 0, 1, 1, 2, 3.

**2.2. Функция fib\_adding(n)**

**Назначение:** Добавление числа 10 к каждому элементу входной последовательности.

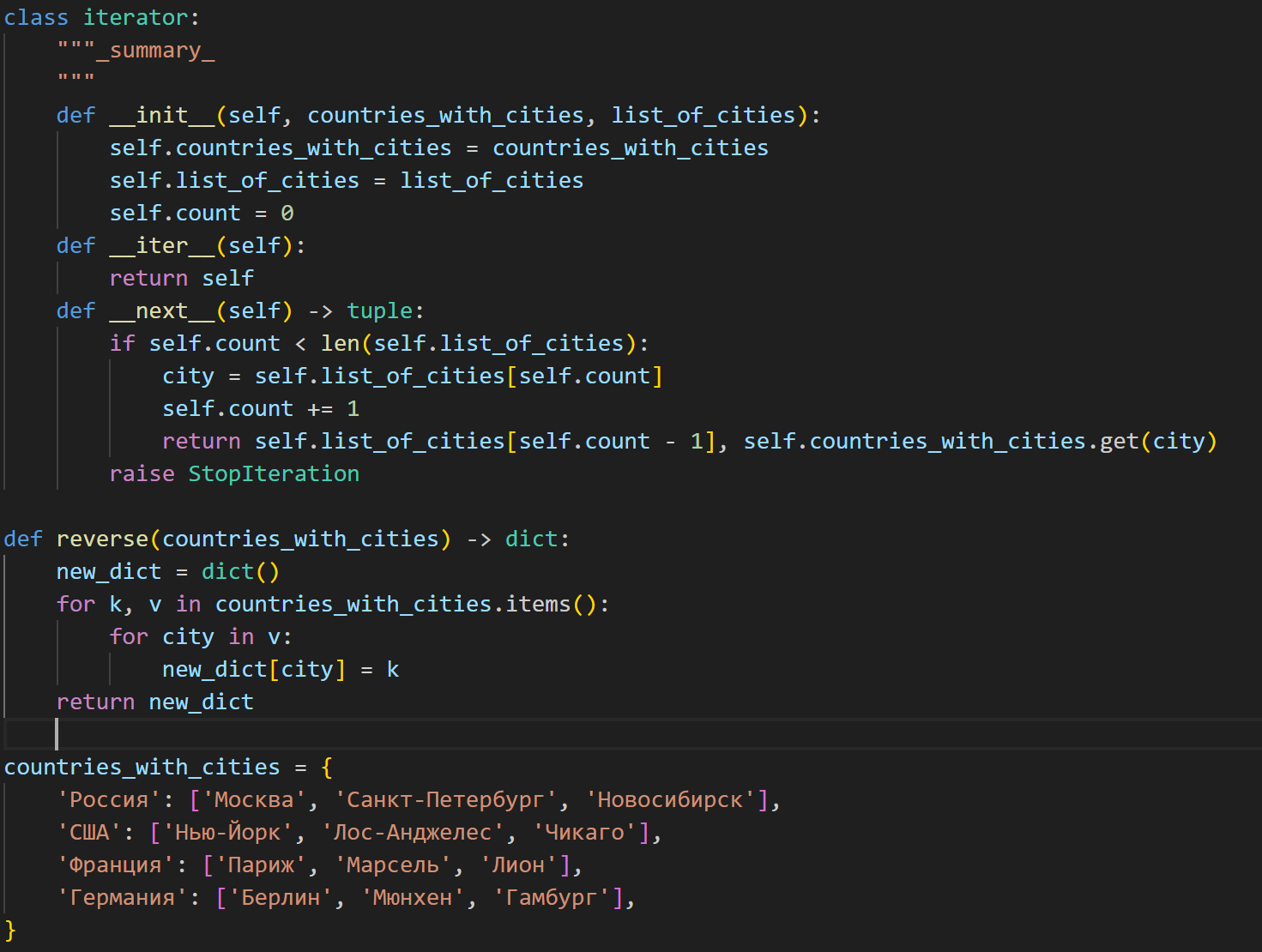
**Логика работы:**

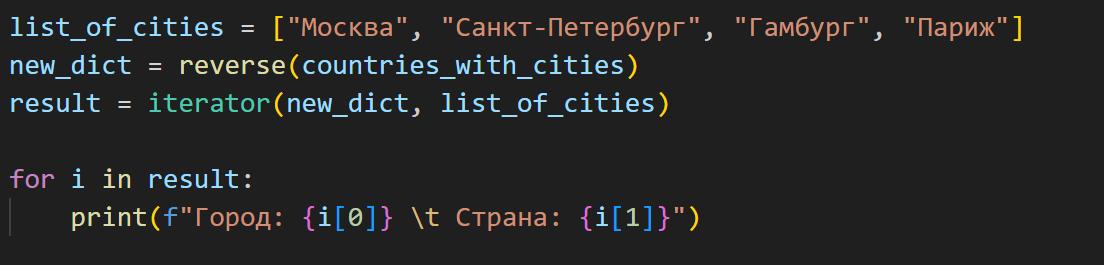
1. Принимает итерабельный объект n (например, генератор).
2. Для каждого элемента i в n:
   * Вычисляет tmp = i + 10.
   * Возвращает tmp через yield.

**2.3. Взаимодействие генераторов**

1. Создается генератор generator = fib(10), возвращающий первые 10 чисел Фибоначчи.
2. Создается модифицированный генератор new\_genertor = fib\_adding(generator).
3. Элементы new\_genertor выводятся через цикл for.

**Задача 4.**

****

****

**1. Назначение программы**

Программа реализует **итератор для сопоставления городов с их странами** на основе заданных данных. Код демонстрирует:

* Создание обратного словаря для поиска страны по городу.
* Использование пользовательского итератора для последовательного доступа к данным.
* Организацию данных в словарях и их обработку.

**2. Описание работы кода**

**2.1. Класс iterator**

**Назначение:** Итерация по списку городов и возврат пар (город, страна).

**Атрибуты:**

* countries\_with\_cities: Словарь, где ключи — страны, значения — списки городов.
* list\_of\_cities: Список городов для обработки.
* count: Счетчик текущей позиции в списке.

**Методы:**

1. **\_\_next\_\_**
   * Возвращает кортеж (город, страна) для текущего города из list\_of\_cities.
   * При достижении конца списка вызывает StopIteration.

**2.2. Функция reverse**

**Назначение:** Создание обратного словаря, где ключи — города, значения — страны.

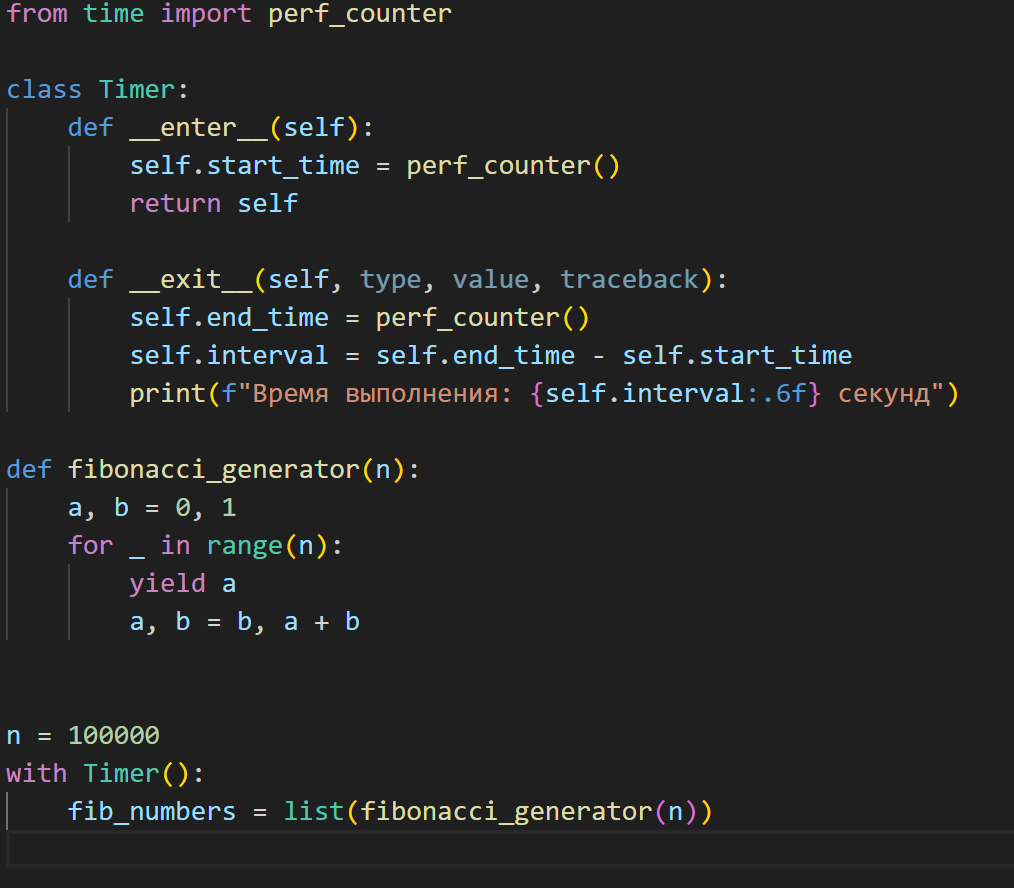
**Логика работы:**

* Для каждой страны и её городов в исходном словаре:
  + Каждый город добавляется в new\_dict с указанием страны.

**2.3. Основной код**

1. Создается исходный словарь countries\_with\_cities и список list\_of\_cities.
2. Функция reverse генерирует обратный словарь new\_dict.
3. Создается итератор result для обработки списка городов.
4. В цикле выводятся пары (город, страна).

**Задача 5.**



**1. Назначение программы**

Программа демонстрирует:

* Замер времени выполнения операций с использованием **менеджера контекста** (with).
* Генерацию последовательности чисел Фибоначчи через **генератор** для оптимизации памяти.
* Эффективную работу с большими объемами данных (100,000 элементов).

**2. Описание работы кода**

**2.1. Класс Timer**

**Назначение:** Измерение времени выполнения блока кода.

**Методы:**

1. **\_\_enter\_\_**
   * Запускает таймер с помощью perf\_counter() (высокоточный счетчик времени).
   * Возвращает объект self для доступа к результатам.
2. **\_\_exit\_\_**
   * Останавливает таймер и вычисляет интервал self.interval.
   * Выводит время выполнения с точностью до 6 знаков после запятой.

**2.2. Функция fibonacci\_generator(n)**

**Назначение:** Генерация первых n чисел Фибоначчи.

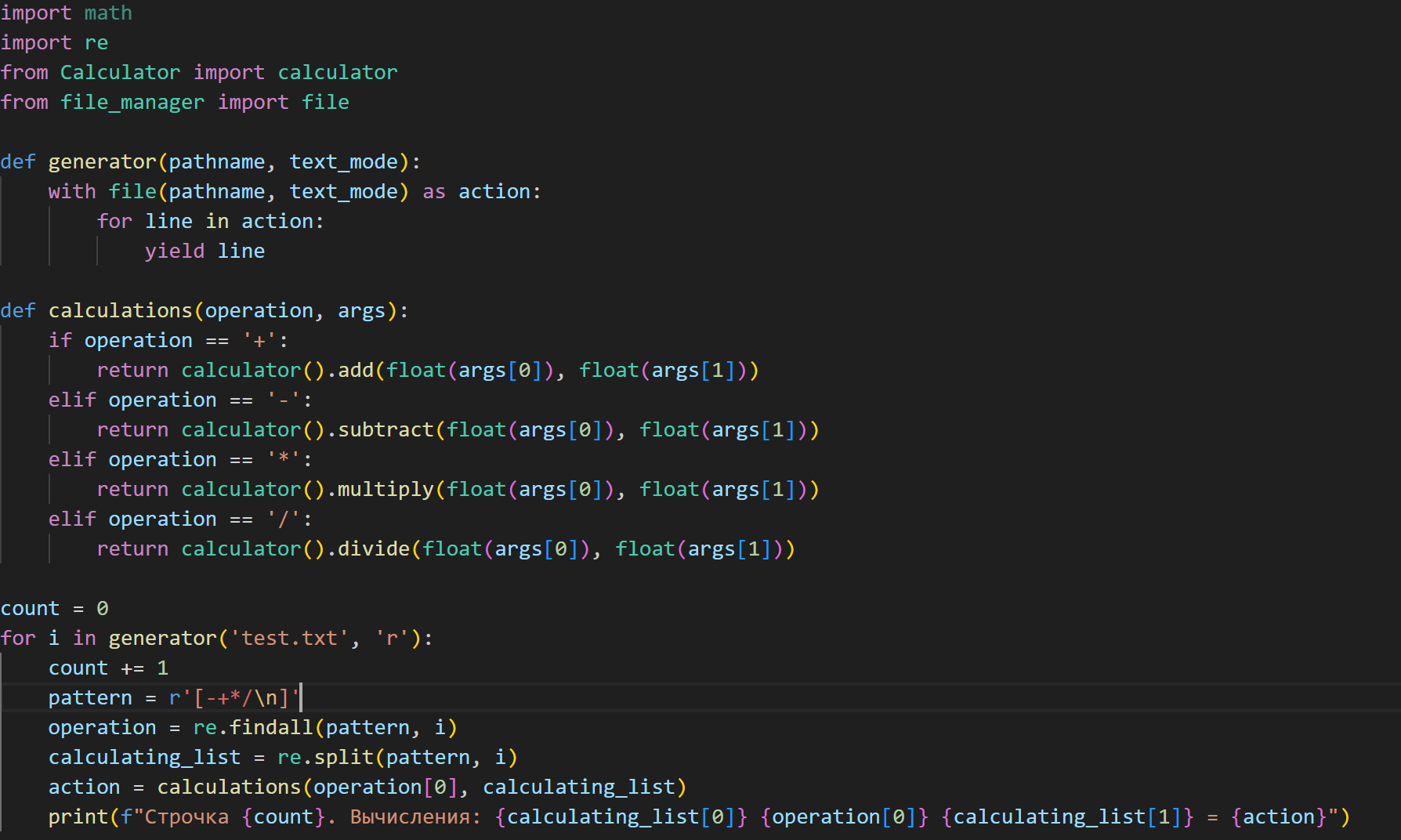
**Логика работы:**

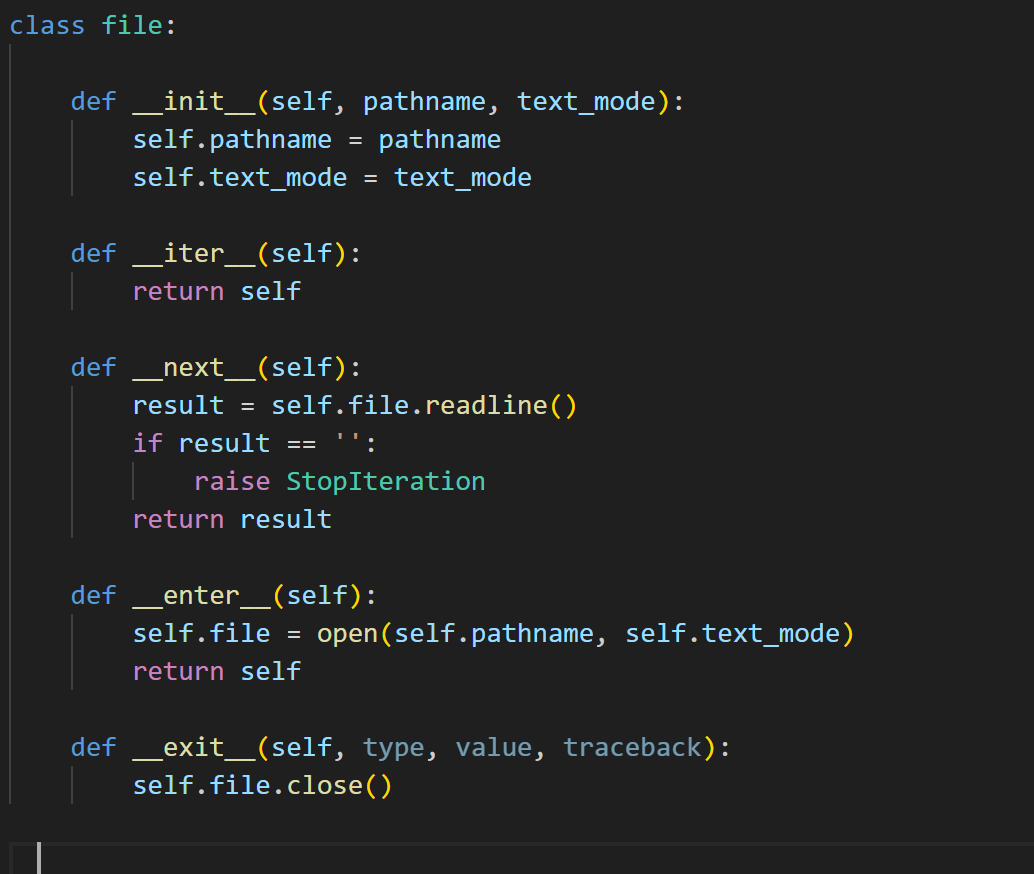
1. Инициализирует начальные значения a = 0, b = 1.
2. В цикле n раз:
   * Возвращает текущее значение a через yield.
   * Обновляет значения: a, b = b, a + b.

**2.3. Основная логика**

1. Устанавливается n = 100000.
2. В блоке with Timer():
   * Генератор преобразуется в список fib\_numbers, что вызывает генерацию всех чисел.
   * Замеряется время выполнения этой операции.

**Задача 6.**





И расширенный калькулятор из предыдущих задач

**1. Назначение программы**

Программа реализует **модульную систему для математических вычислений и работы с файлами**, включая:

1. Класс calculator для арифметических и статистических операций с контролем точности.
2. Класс file для безопасного чтения файлов через контекстный менеджер и итератор.
3. Скрипт для автоматической обработки математических операций из текстового файла.

**2. Описание работы кода**

**2.1. Файл 1: file**

**Класс file** реализует:

* **Контекстный менеджер** (\_\_enter\_\_, \_\_exit\_\_) для автоматического открытия/закрытия файла.
* **Итератор** (\_\_iter\_\_, \_\_next\_\_) для построчного чтения файла.

**Логика работы:**

* При входе в контекст (with) файл открывается в указанном режиме.
* Метод \_\_next\_\_ читает строки до конца файла.

**2.2. Файл 2: Основной скрипт**

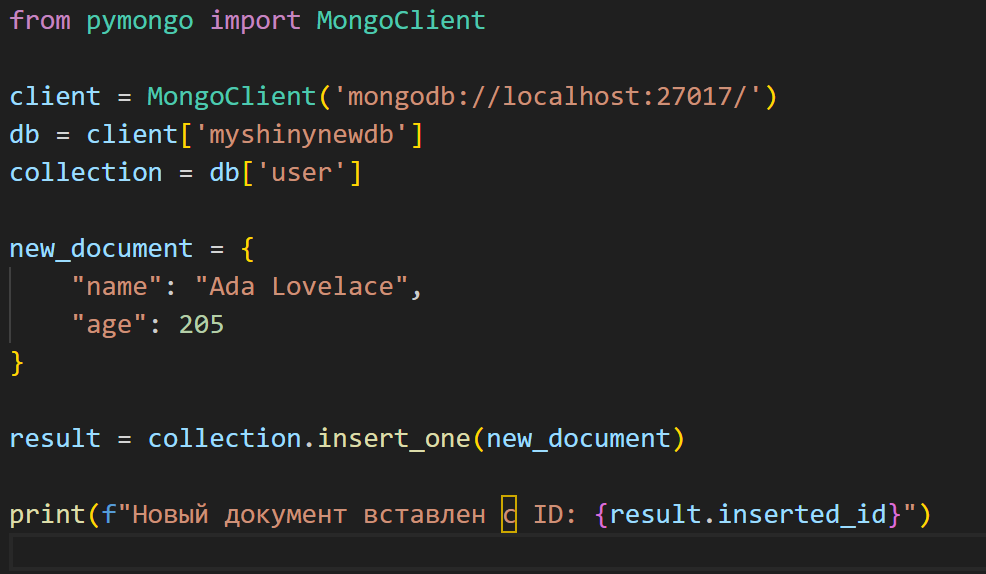
**Функции:**

1. **generator**:
   * Использует класс file для построчного чтения файла через генератор.
2. **calculations**:
   * Выполняет арифметические операции с помощью класса calculator.

**Логика работы:**

1. Чтение строк из файла test.txt.
2. Парсинг строк с помощью регулярных выражений (операция и числа).
3. Выполнение операции и вывод результата.

**Задача 7.**

  
**1. Назначение программы**

Программа демонстрирует **базовое взаимодействие с MongoDB** с использованием библиотеки pymongo. Код выполняет:

* Подключение к локальной базе данных MongoDB.
* Создание новой базы данных и коллекции.
* Вставку документа в коллекцию.
* Вывод уникального идентификатора (ID) добавленного документа.

**2. Описание работы кода**

**2.1. Подключение к MongoDB**

1. **MongoClient('mongodb://localhost:27017/')**
   * Создает клиент для подключения к MongoDB, работающему на локальном сервере (порт 27017).
2. **client['myshinynewdb']**
   * Выбирает базу данных с именем myshinynewdb. Если база не существует, она будет создана автоматически.
3. **db['user']**
   * Выбирает коллекцию user в базе данных. Если коллекция отсутствует, она создается при первой вставке документа.

**2.2. Создание и вставка документа**

1. **Создание документа:**

new\_document = {

"name": "Ada Lovelace",

"age": 205

}

1. **Вставка документа:**

result = collection.insert\_one(new\_document)

1. **Вывод ID:**

print(f"Новый документ вставлен с ID: {result.inserted\_id}")