Московский государственный технический университет им. Н.Э. Баумана

Факультет «Информатика и системы управления»
Кафедра ИУ5 «Системы обработки информации и управления»

Курс «Базовые компоненты интернет-технологий»

Отчет по домашнему заданию на тему «разработка комплексного приложения на языке Python»

Выполнил:
студент группы ИУ5-34Б:
Такташова Дарья Юрьевна
Подпись и дата:

Проверил:
преподаватель каф. ИУ5
Гапанюк Ю.Е
Подпись и дата:

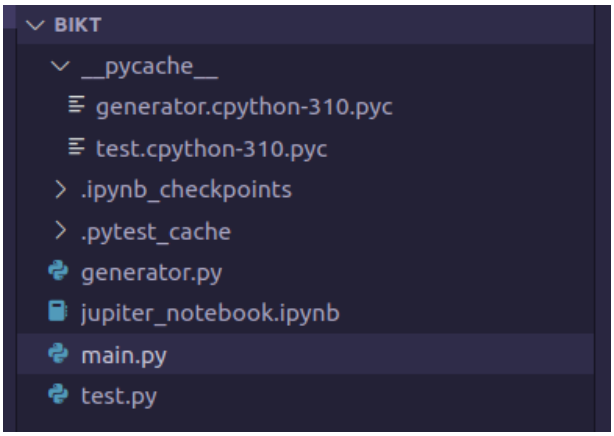
Москва, 2022 г.

Задание:

С использованием механизма итераторов или генераторов реализуйте с помощью концепции ленивых вычислений одну из последовательностей OEIS. Примером могут являться числа Фибоначчи.

1. Для реализованной последовательности разработайте 3-5 модульных тестов, которые, в том числе, проверяют то, что последовательность поддерживает ленивые вычисления.
2. Разработайте веб-сервис с использованием фреймворка Flask, который возвращает N элементов последовательности (параметр N передается в запросе к сервису).
3. Создайте Jupyter-notebook, который реализует обращение к веб-сервису с использованием библиотеки requests и визуализацию полученных от веб-сервиса данных с использованием библиотеки matplotlib.

Код программы:



generator.py(генератор простых чисел):

```python
def prime_generator(n):
    sieve = set(range(2, n+1))
    while sieve:
        prime = min(sieve)
        yield prime
        sieve -= set(range(prime, n+1, prime))
```

main.py:

```python
from flask import Flask
from generator import prime_generator

app = Flask(__name__)


@app.route('/')
def index():
    return "Генератор простых чисел"

@app.route('/<int:n>')
def count_number(n):
    return str(list(prime_generator(n)))


if __name__ == "__main__":
    app.run(debug=True)
```

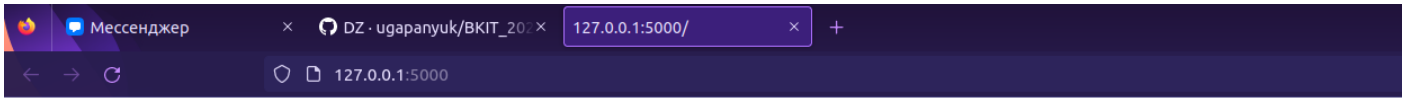
test.py:

```python
import unittest
from generator import prime_generator


class TestPrime(unittest.TestCase):
    def test_prime_generator_1(self):
        res = [i for i in prime_generator(10)]
        expected = [2, 3, 5, 7]
        self.assertEqual(res, expected)

    def test_prime_generator_2(self):
        res = [i for i in prime_generator(1)]
        expected = []
        self.assertEqual(res, expected)

    def test_prime_generator_3(self):
        res = [i for i in prime_generator(100)]
        expected = [2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47, 53, 59, 61, 67, 71, 73, 79, 83, 89, 97]
        self.assertEqual(res, expected)

if __name__ == "__main__":
    unittest.main()
```

Результаты выполнения программы:

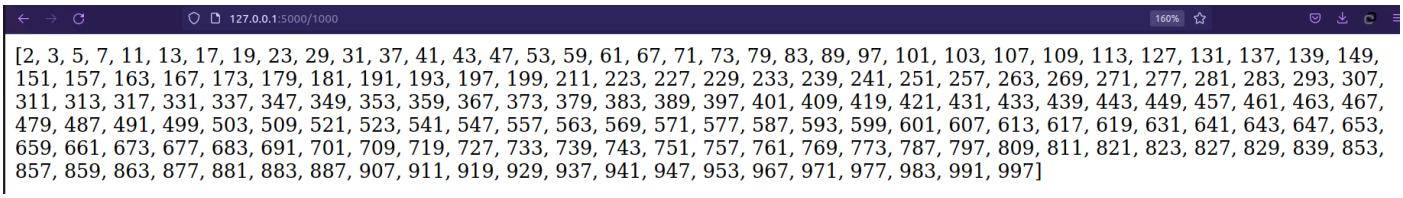
```
dasha@dasha-BOHB-WAX9:~/Desktop/BIKT/BIKT$ python3 main.py
 * Serving Flask app 'main'
 * Debug mode: on
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
 * Running on http://127.0.0.1:5000
Press CTRL+C to quit
 * Restarting with stat
 * Debugger is active!
 * Debugger PIN: 485-179-104
```

http://127.0.0.1:5000

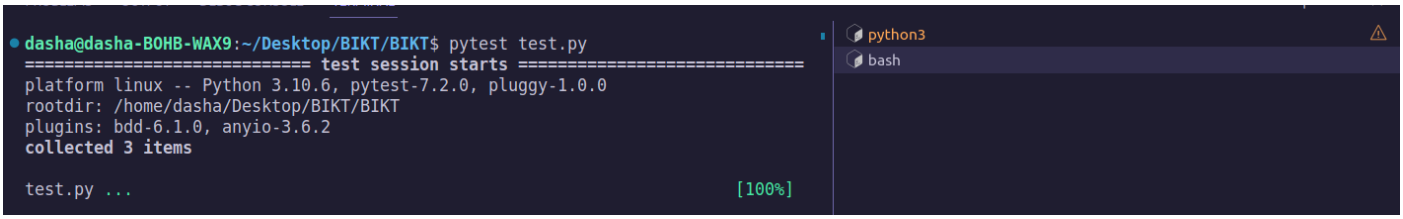


Генератор простых чисел

http://127.0.0.1:5000/1000



Запуск тестов при поднятом сервере:



Jupiter-notebook: