**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

Выполнил:

студент группы ИУ5-31Б

Корецкий К.В.

Проверил:

Гапанюк Е.Ю.

2022 г.

**Задание**

1. С использованием механизма итераторов или генераторов реализуйте с помощью концепции ленивых вычислений [одну из последовательностей OEIS.](https://ru.wikipedia.org/wiki/%D0%9E%D0%BD%D0%BB%D0%B0%D0%B9%D0%BD-%D1%8D%D0%BD%D1%86%D0%B8%D0%BA%D0%BB%D0%BE%D0%BF%D0%B5%D0%B4%D0%B8%D1%8F_%D1%86%D0%B5%D0%BB%D0%BE%D1%87%D0%B8%D1%81%D0%BB%D0%B5%D0%BD%D0%BD%D1%8B%D1%85_%D0%BF%D0%BE%D1%81%D0%BB%D0%B5%D0%B4%D0%BE%D0%B2%D0%B0%D1%82%D0%B5%D0%BB%D1%8C%D0%BD%D0%BE%D1%81%D1%82%D0%B5%D0%B9) Примером могут являться [числа Фибоначчи.](https://ru.wikipedia.org/wiki/%D0%A7%D0%B8%D1%81%D0%BB%D0%B0_%D0%A4%D0%B8%D0%B1%D0%BE%D0%BD%D0%B0%D1%87%D1%87%D0%B8)
2. Для реализованной последовательности разработайте 3-5 модульных тестов, которые, в том числе, проверяют то, что последовательность поддерживает ленивые вычисления.
3. Разработайте веб-сервис с использованием фреймворка Flask, который возвращает N элементов последовательности (параметр N передается в запросе к сервису).
4. Создайте Jupyter-notebook, который реализует обращение к веб-сервису с использованием библиотеки [requests](https://requests.readthedocs.io/en/latest/) и визуализацию полученных от веб-сервиса данных с использованием библиотеки [matplotlib.](https://matplotlib.org/)

**Текст программы**

generator.py

**def fib(n):**

**a, b = 0, 1**

**for i in range(n):**

**yield a**

**a, b = b, a + b**

test.py

**import unittest**

**from generator import fib**

**import time**

**class fib\_test(unittest.TestCase):**

**def test\_fib\_1(self):**

**a = [i for i in fib(5)]**

**expected = [0, 1, 1, 2, 3]**

**self.assertEqual(a, expected)**

**def test\_fib\_2(self):**

**a = [i for i in fib(10)]**

**expected = [0, 1, 1, 2, 3, 5, 8, 13, 21, 34]**

**self.assertEqual(a, expected)**

**def test\_fib\_3(self):**

**a = [i for i in fib(0)]**

**expected = []**

**self.assertEqual(a, expected)**

**def test\_fib\_4(self):**

**start\_time = time.time()**

**a = fib(200000)**

**end\_time = time.time() - start\_time**

**self.assertLess(end\_time, 1) # if spent time less than a second**

**def test\_fib\_5(self):**

**start\_time = time.time()**

**a = [i for i in fib(200000)]**

**end\_time = time.time() - start\_time**

**self.assertLess(1, end\_time)**

**if \_\_name\_\_ == '\_\_main\_\_':**

**unittest.main()**

flask\_app.py

**from flask import Flask**

**import generator**

**app = Flask('fibonacci sequences')**

**@app.route('/')**

**def index():**

**return 'My cool fibonacci sequence flask app!'**

**@app.route('/<int:n>')**

**def get\_sequence(n):**

**return list(generator.fib(n))**

**@app.errorhandler(404)**

**def page\_not\_found(e):**

**return 'Oops! Try to enter a number!'**

**if \_\_name\_\_ == '\_\_main\_\_':**

**app.run(debug=True)**

notebook.ipynb

**import requests**

**import json**

**r = requests.get('http://localhost:5000/20')**

**seq = json.loads(r.text)**

**print(seq)**

**import matplotlib.pyplot as mpl**

**import numpy as np**

**%matplotlib inline**

**mpl.style.use('seaborn-whitegrid')**

**axis = mpl.axes()**

**x = np.arange(0, 20)**

**axis.plot(x, seq)**

**mpl.show()**

**Экранные формы с примерами выполнения программы**

