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 «Вычисление последовательностей OEIS с использованием механизма итераторов или генераторов»

Выполнил:
Студент группы ИУ5-32Б:
Секретов Кирилл
Подпись и дата:

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

Москва, 2022 г.

Описание задания

1. С использованием механизма итераторов или генераторов реализуйте с помощью концепции ленивых вычислений одну из последовательностей OEIS. Примером могут являться числа Фибоначчи.
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 и визуализацию полученных от веб-сервиса данных с использованием библиотеки matplotlib.

Текст программы

Задание 1

```
def fibb():
    a, b = 1, 1
    while True:
        yield a
        a, b = b, a+b
```

Задание 2

```
import unittest
from fibbonachi import fibb
from collections.abc import Generator


class Test(unittest.TestCase):
    def test_sequence(self):
        fi = fibb()
        a = [next(fi) for i in range(10)]
        self.assertEqual(len(a), 10)
        self.assertListEqual(a, [1, 1, 2, 3, 5, 8, 13, 21, 34, 55])

        exp = [89, 144, 233, 377, 610]
        for count, val in enumerate(fi):
            if count > 4:
                break
            self.assertEqual(val, exp[count])

    def test_generator(self):
        a = fibb()
        self.assertIsInstance(a, Generator)
        self.assertEqual(next(a), 1)
```

```
        self.assertEqual(next(a), 1)
        self.assertEqual(next(a), 2)
        self.assertEqual(next(a), 3)


    def test_func(self):
        fi = fibb()
        a = list(zip(range(5), fi))
        self.assertEqual(len(a), 5)
        self.assertListEqual(a, [(0, 1), (1, 1), (2, 2), (3, 3), (4, 5)])

        a = list(zip(range(5), fi))
        self.assertEqual(len(a), 5)
        self.assertListEqual(a, [(0, 8), (1, 13), (2, 21), (3, 34), (4, 55)])


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

Задание 3

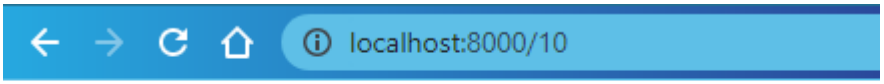
```
from flask import Flask
from fibbonachi import fibb

app = Flask(__name__)

@app.route("/")
def main_page():
    return "<h1>Fibonachi generator</h1>"

@app.route("/<int:num>")
def generate(num):
    fib = fibb()
    return [next(fib) for i in range(num)]

if __name__ == "__main__":
    app.run(host="localhost", port=8000)
```

Задание 4

```
import requests
import matplotlib.pyplot as plt
url = "http://localhost:8000/"
def get_data(num):
    return requests.get(url + str(num)).json()
y = get_data(20)
```

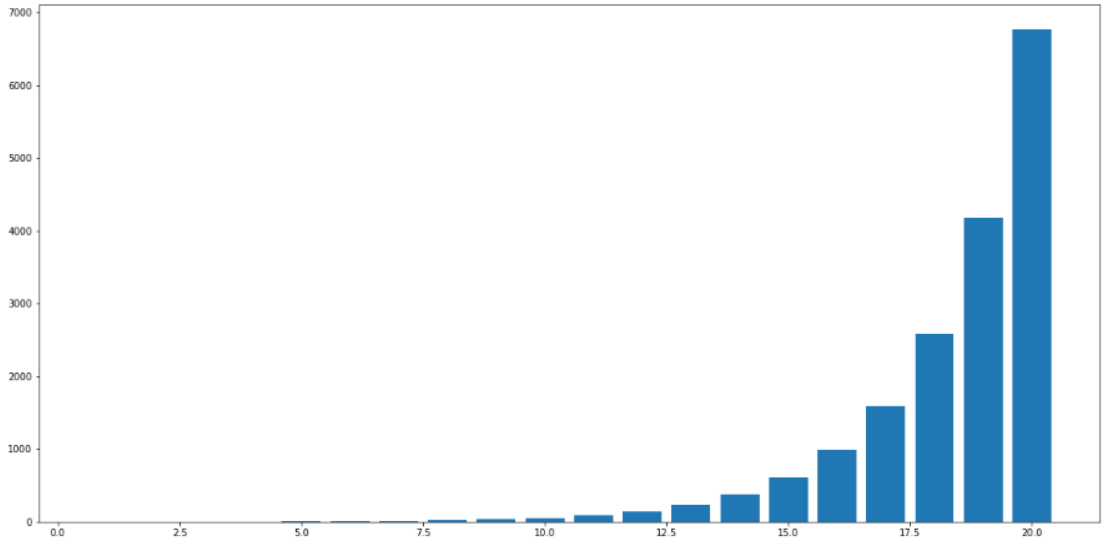
```
x = [i for i in range(1, 21)]
fig = plt.figure(figsize = (20, 10))
plt.bar(x, y)
plt.show()
fig = plt.figure(figsize = (20, 10))
plt.plot(x, y)
plt.show()
```

Примеры выполнения программ




```
[1,1,2,3,5,8,13,21,34,55]
```

```
In [4]: fig = plt.figure(figsize = (20, 10))
        plt.bar(x, y)
        plt.show()
```




```
In [5]: fig = plt.figure(figsize = (20, 10))
        plt.plot(x, y)
        plt.show()
```