

**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:

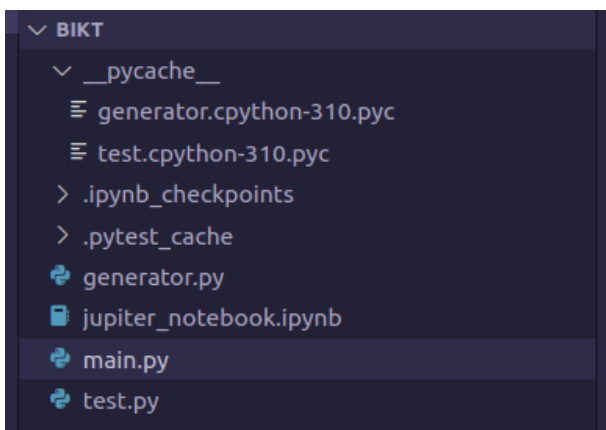
Москва, 2022 г.

## 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):

```
1 def prime_generator(n):
2     sieve = set(range(2, n+1))
3     while sieve:
4         prime = min(sieve)
5         yield prime
6         sieve -= set(range(prime, n+1, prime))
7
```

### main.py:

```

1  from flask import Flask
2  from generator import prime_generator
3
4  app = Flask(__name__)
5
6
7  @app.route('/')
8  def index():
9      return "Генератор простых чисел"
10
11
12  @app.route('/<int:n>')
13  def count_number(n):
14      return str(list(prime_generator(n)))
15
16
17  if __name__ == "__main__":
18      app.run(debug=True)

```

test.py:

```

1  import unittest
2  from generator import prime_generator
3
4
5  class TestPrime(unittest.TestCase):
6      def test_prime_generator_1(self):
7          res = [i for i in prime_generator(10)]
8          expected = [2, 3, 5, 7]
9          self.assertEqual(res, expected)
10
11      def test_prime_generator_2(self):
12          res = [i for i in prime_generator(1)]
13          expected = []
14          self.assertEqual(res, expected)
15
16      def test_prime_generator_3(self):
17          res = [i for i in prime_generator(100)]
18          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]
19          self.assertEqual(res, expected)
20
21  if __name__ == "__main__":
22      unittest.main()

```

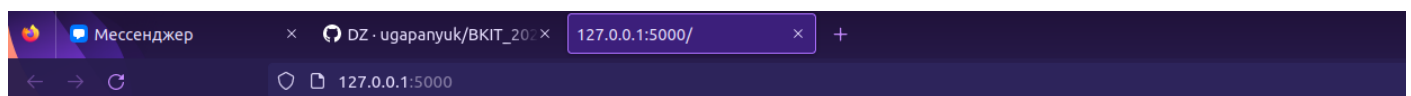
Результаты выполнения программы:

```

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>

```
[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, 101, 103, 107, 109, 113, 127, 131, 137, 139, 149, 151, 157, 163, 167, 173, 179, 181, 191, 193, 197, 199, 211, 223, 227, 229, 233, 239, 241, 251, 257, 263, 269, 271, 277, 281, 283, 293, 307, 311, 313, 317, 331, 337, 347, 349, 353, 359, 367, 373, 379, 383, 389, 397, 401, 409, 419, 421, 431, 433, 439, 443, 449, 457, 461, 463, 467, 479, 487, 491, 499, 503, 509, 521, 523, 541, 547, 557, 563, 569, 571, 577, 587, 593, 599, 601, 607, 613, 617, 619, 631, 641, 643, 647, 653, 659, 661, 673, 677, 683, 691, 701, 709, 719, 727, 733, 739, 743, 751, 757, 761, 769, 773, 787, 797, 809, 811, 821, 823, 827, 829, 839, 853, 857, 859, 863, 877, 881, 883, 887, 907, 911, 919, 929, 937, 941, 947, 953, 967, 971, 977, 983, 991, 997]
```

## Запуск тестов при поднятом сервере:

```
dasha@dasha-B0HB-WAX9:~/Desktop/BIKT/BIKT$ pytest test.py
===== test session starts =====
platform linux -- Python 3.10.6, pytest-7.2.0, pluggy-1.0.0
rootdir: /home/dasha/Desktop/BIKT/BIKT
plugins: bdd-6.1.0, anyio-3.6.2
collected 3 items

test.py ... [100%]
```

## Jupyter-notebook:

The screenshot shows a Jupyter Notebook interface with the following content:

```
In [1]: import requests
import json

r = requests.get('http://localhost:5000/100')
r.text

Out[1]: '[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]'
```

```
In [6]: import matplotlib.pyplot as plt

y = json.loads(r.text)
plt.plot(y)

Out[6]: [<matplotlib.lines.Line2D at 0x7f8d3d3653c0>]
```

The plot shows a line graph with the x-axis ranging from 0 to 25 and the y-axis ranging from 0 to 100. The data points represent prime numbers, showing an increasing trend.

In the foreground, a terminal window titled "Терминал" displays the following output:

```
[I 23:33:32.161 NotebookApp] Serving notebooks from local directory: /home/dasha
[I 23:33:32.161 NotebookApp] Jupyter Notebook 6.5.2 is running at:
[I 23:33:32.161 NotebookApp] http://localhost:8888/?token=05ac761547f11923aed3fdb1d6d6c235a845545478f114c5
[I 23:33:32.161 NotebookApp] or http://127.0.0.1:8888/?token=05ac761547f11923aed3fdb1d6d6c235a845545478f114c5
[I 23:33:32.161 NotebookApp] Use Control-C to stop this server and shut down all kernels (twice to skip confirmation).
[C 23:33:32.192 NotebookApp]

To access the notebook, open this file in a browser:
file:///home/dasha/.local/share/jupyter/runtime/nbserver-15922-open.html
Or copy and paste one of these URLs:
http://localhost:8888/?token=05ac761547f11923aed3fdb1d6d6c235a845545478f114c5
or http://127.0.0.1:8888/?token=05ac761547f11923aed3fdb1d6d6c235a845545478f114c5
[I 23:33:47.307 NotebookApp] Kernel started: f848f2c0-71ac-451a-a333-770d6ec86c5a, name: python3
[IPKernelApp] ERROR | No such comm target registered: jupyter.widget.control
[IPKernelApp] WARNING | No such comm: 0d1b8788-5ffc-44ab-92b4-634df9227eff
```

