Московский государственный технический университет имени Н.Э.Баумана

Факультет Радиотехнический

Кафедра ИУ5 Системы обработки информации и управления

Отчет по домашнему заданию по курсу Базовые компоненты

Исполнитель
Студент группы РТ5-31Б _____ Татаев С.А.
«__»_____ 2022 г.

Проверил
Доцент кафедры ИУ5 _____ Гапанюк Ю.Е.
«__»_____ 2022 г.

г. Москва – 2022

Задание

Задание:

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.

Листинг программы, в которой реализуется последовательность Фибоначчи (generator.py)

```
def fib(n):
    x1, x2 = 0, 1
    for i in range(n):
        yield x1
        x1, x2 = x2, x1 + x2


print(list(fib(5)))
```

Листинг программы, в которой реализуются тесты (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()
```

Листинг программы, в которой реализуется веб-сервис с использованием Flask (flask_app.py)

```
from flask import Flask
import generator

app = Flask('fibonacci sequences')



@app.route('/')
def index():
    return '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)
```

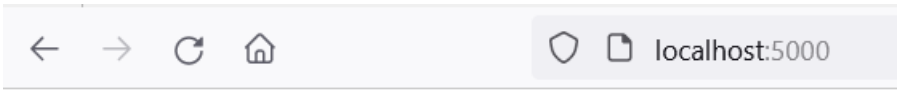
Результаты работы программы generator.py

```
[анализируем generator.py]
[0, 1, 1, 2, 3]
```

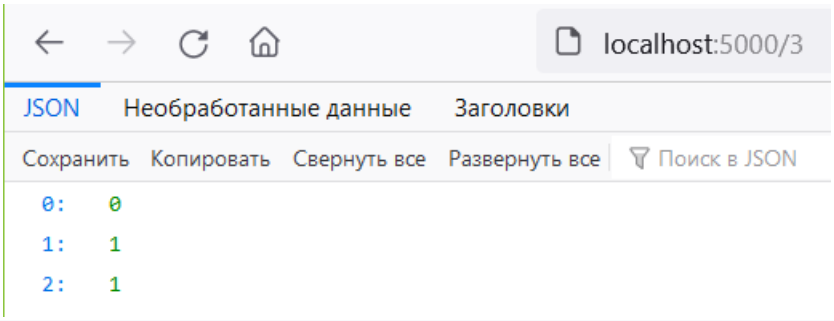
Результаты работы программы test.py

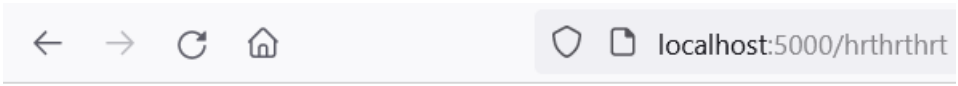
```
[анализируем test.py]
.....
----------------------------------
Ran 5 tests in 6.451s

OK
```

Результаты работы программы flask_app.py



Fibonacci sequence flask app





Oops! Try to enter a number!

Jupyter-notebook dz_bkit.ipynb

jupyter dz_bkit Last Checkpoint: 09.12.2022 (autosaved) Logout

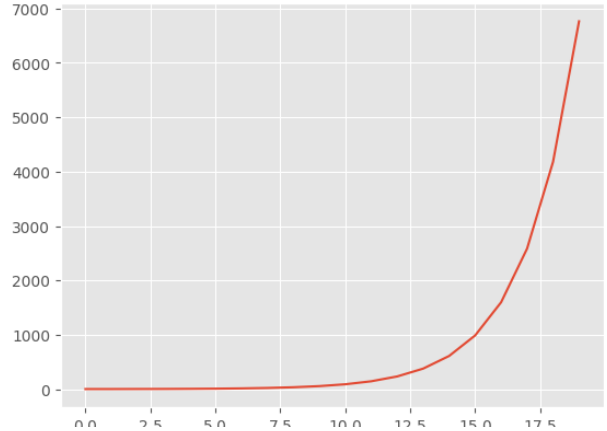
File Edit View Insert Cell Kernel Widgets Help Not Trusted Python 3 (ipykernel)

Run Code

In [2]:
```
import requests
import json
r = requests.get('http://localhost:5000/20')
seq = json.loads(r.text)
print(seq)
```

```
[1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, 233, 377, 610, 987, 1597, 2584, 4181, 6765]
```

In [5]:
```
import matplotlib.pyplot as mpl
import numpy as np
%matplotlib inline
mpl.style.use('ggplot')
axis = mpl.axes()
x = np.arange(0, 20)
axis.plot(x, seq)
mpl.show()
```



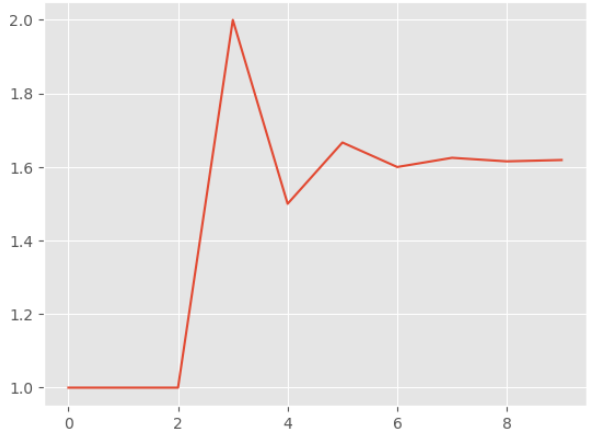
jupyter dz_bkit Last Checkpoint: 09.12.2022 (autosaved) Logout

File Edit View Insert Cell Kernel Widgets Help Not Trusted Python 3 (ipykernel)

Run Code

0.0 2.5 5.0 7.5 10.0 12.5 15.0 17.5

In [6]:
```
def golden_seq(n):
    try:
        r = requests.get('http://localhost:5000/' + str(n))
        seq = json.loads(r.text)
        return seq[len(seq)-1]/seq[len(seq)-2]
    except:
        return 1.0

f = lambda i, j: golden_seq(int(j))
y = np.fromfunction(np.vectorize(f), (1, 10))
x = np.arange(10)
axis = mpl.axes()
axis.plot(x, y[0])
mpl.show()
```



In [ ]: