**Домашняя работа №2**

Тема “Визуализация данных в Matplotlib”

**Задание 1**

Загрузите модуль pyplot библиотеки matplotlib с псевдонимом plt, а также библиотеку numpy с псевдонимом np.

Примените магическую функцию %matplotlib inline для отображения графиков в Jupyter Notebook и настройки конфигурации ноутбука со значением 'svg' для более четкого отображения графиков.

Создайте список под названием x с числами 1, 2, 3, 4, 5, 6, 7 и список y с числами 3.5, 3.8, 4.2, 4.5, 5, 5.5, 7.

С помощью функции plot постройте график, соединяющий линиями точки с горизонтальными координатами из списка x и вертикальными - из списка y.

Затем в следующей ячейке постройте диаграмму рассеяния (другие названия - диаграмма разброса, scatter plot).

**Решение:**

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

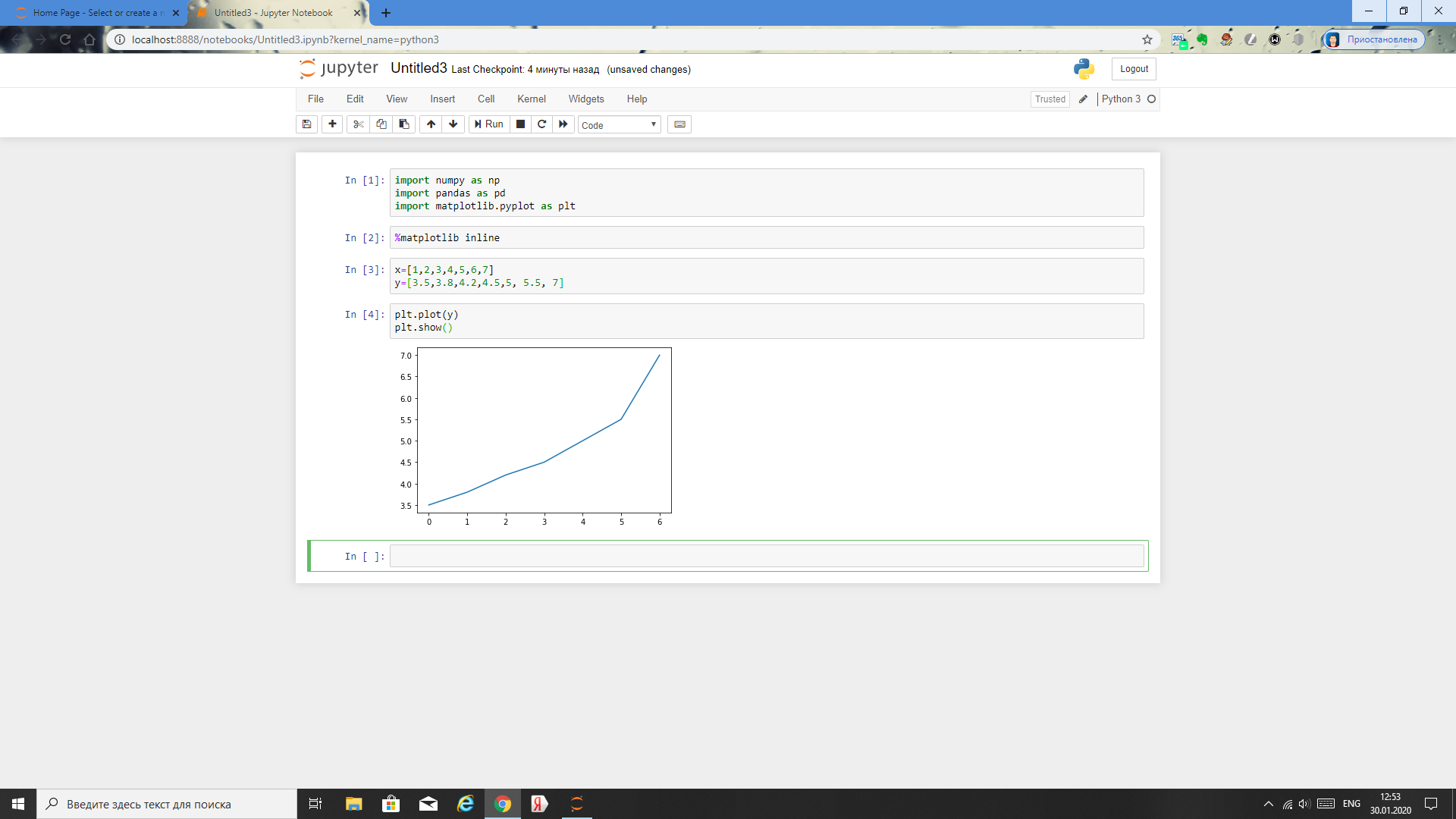
%matplotlib inline

x=[1,2,3,4,5,6,7]

y=[3.5,3.8,4.2,4.5,5, 5.5, 7]

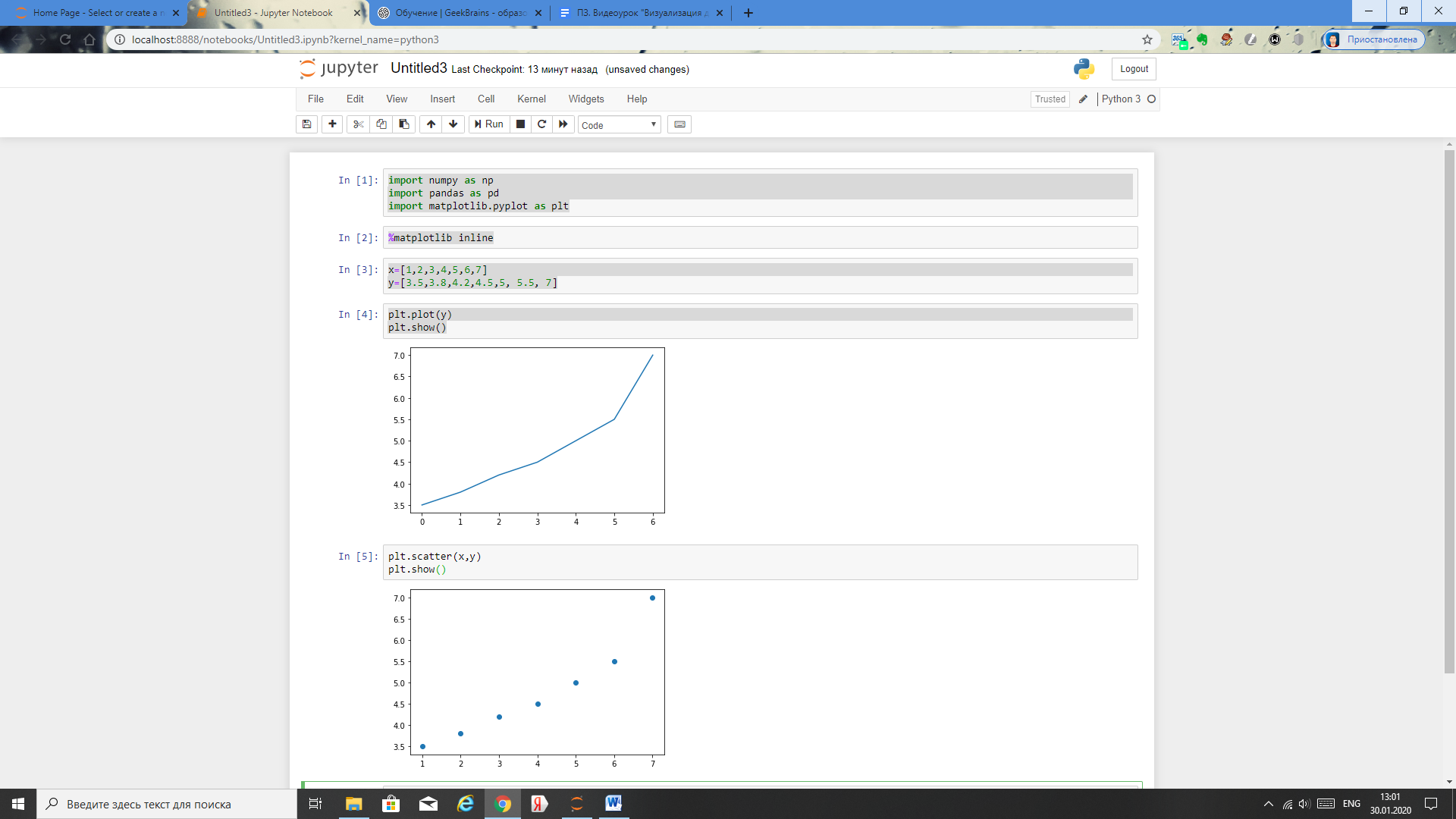
plt.plot(y)

plt.show()



plt.scatter(x,y)

plt.show()



**Задание 2**

С помощью функции linspace из библиотеки Numpy создайте массив t из 51 числа от 0 до 10 включительно.

Создайте массив Numpy под названием f, содержащий косинусы элементов массива t.

Постройте линейную диаграмму, используя массив t для координат по горизонтали,а массив f - для координат по вертикали. Линия графика должна быть зеленого цвета.

Выведите название диаграммы - 'График f(t)'. Также добавьте названия для горизонтальной оси - 'Значения t' и для вертикальной - 'Значения f'.

Ограничьте график по оси x значениями 0.5 и 9.5, а по оси y - значениями -2.5 и 2.5.

**Решение:**

t=np.linspace(0, 10, num=51)

t

f=np.cos(t)

f

plt.plot(t,f, color='g')

plt.show()

plt.plot(t,f, color='g')

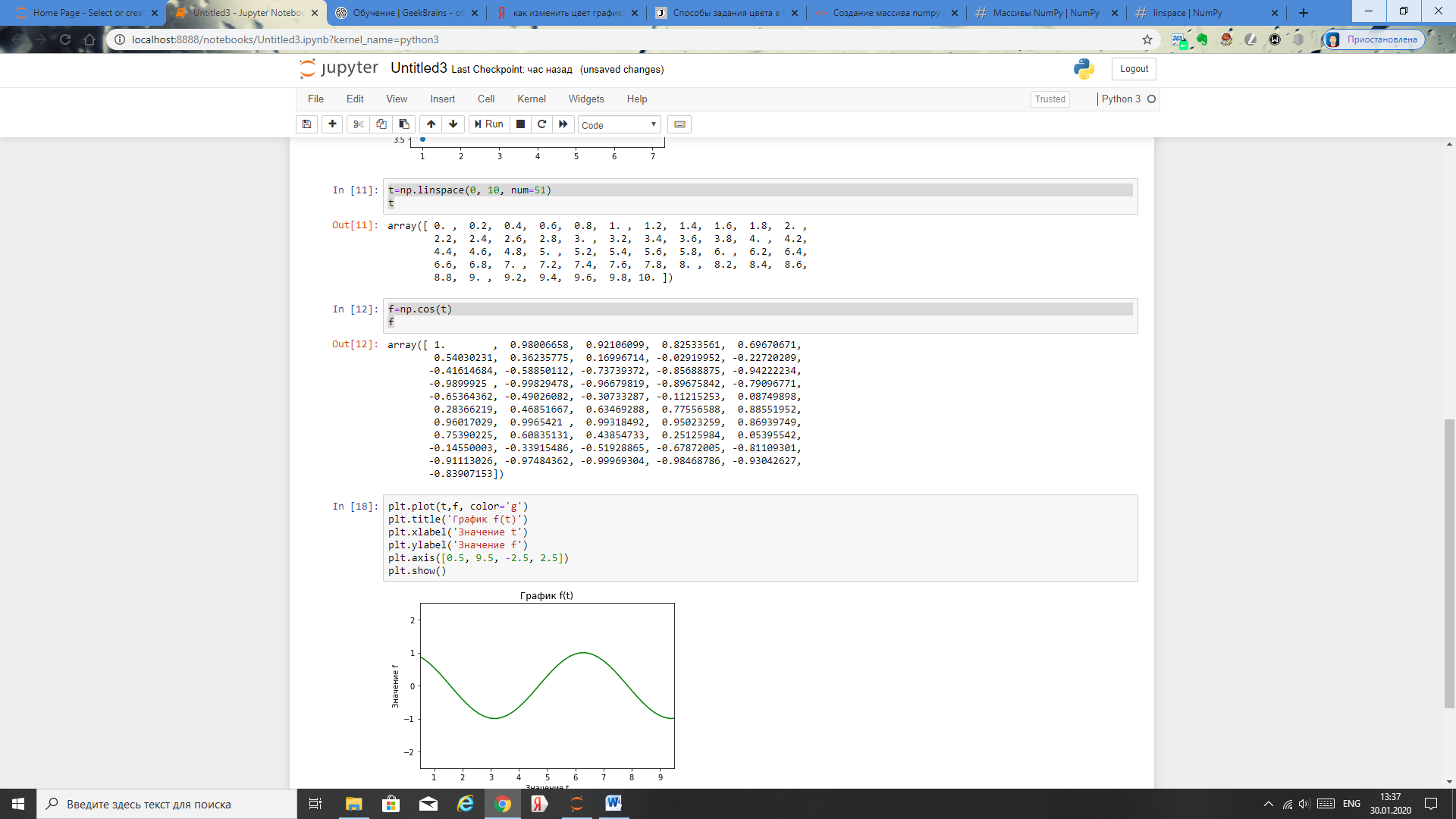
plt.title('График f(t)')

plt.xlabel('Значение t')

plt.ylabel('Значение f')

plt.axis([0.5, 9.5, -2.5, 2.5])

plt.show()



**\*Задание 3**

С помощью функции linspace библиотеки Numpy создайте массив x из 51 числа от -3 до 3 включительно.

Создайте массивы y1, y2, y3, y4 по следующим формулам:

y1 = x\*\*2

y2 = 2 \* x + 0.5

y3 = -3 \* x - 1.5

y4 = sin(x)

Используя функцию subplots модуля matplotlib.pyplot, создайте объект matplotlib.figure.Figure с названием fig и массив объектов Axes под названием ax,причем так, чтобы у вас было 4 отдельных графика в сетке, состоящей из двух строк и двух столбцов. В каждом графике массив x используется для координат по горизонтали.В левом верхнем графике для координат по вертикали используйте y1,в правом верхнем - y2, в левом нижнем - y3, в правом нижнем - y4.Дайте название графикам: 'График y1', 'График y2' и т.д.

Для графика в левом верхнем углу установите границы по оси x от -5 до 5.

Установите размеры фигуры 8 дюймов по горизонтали и 6 дюймов по вертикали.

Вертикальные и горизонтальные зазоры между графиками должны составлять 0.3.

**Решение:**

x=np.linspace(-3, 3, num=51)

x

y1=x\*\*2

y2=2\*x+0.5

y3=-3\*x-1.5

y4=np.sin(x)

fig, ax=plt.subplots(nrows=2, ncols=2)

ax1, ax2, ax3, ax4 = ax.flatten()

ax1.plot(x, y1)

ax1.set\_title('График y1')

ax2.plot(x, y2)

ax1.set\_xlim(-5,5)

ax2.set\_title('График y2')

ax3.plot(x, y3)

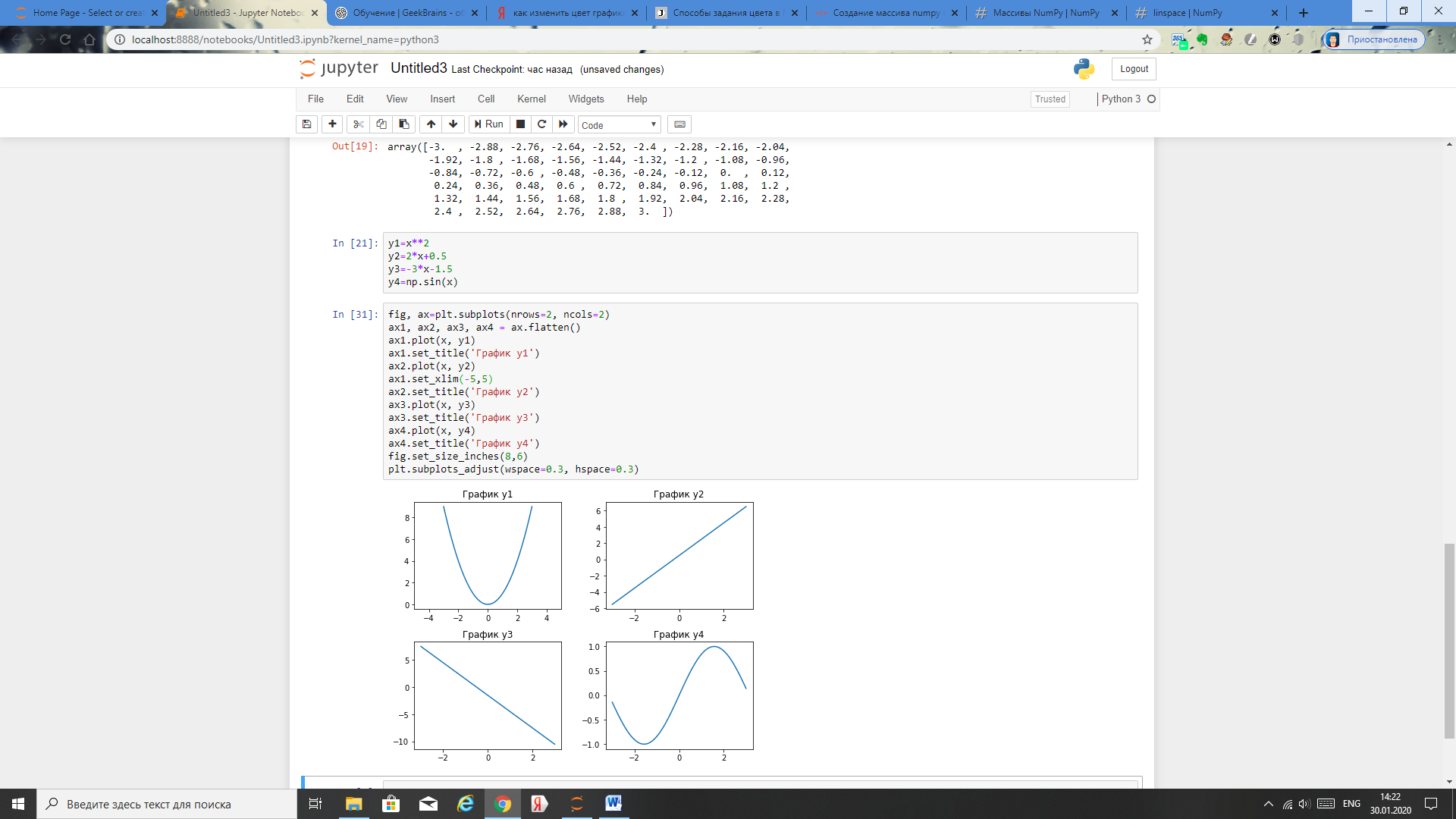
ax3.set\_title('График y3')

ax4.plot(x, y4)

ax4.set\_title('График y4')

fig.set\_size\_inches(8,6)

plt.subplots\_adjust(wspace=0.3, hspace=0.3)



**\*Задание 4**

В этом задании мы будем работать с датасетом, в котором приведены данные по мошенничеству с кредитными данными: Credit Card Fraud Detection (информация об авторах: Andrea Dal Pozzolo, Olivier Caelen, Reid A. Johnson and Gianluca Bontempi. Calibrating Probability with Undersampling for Unbalanced Classification. In Symposium on Computational Intelligence and Data Mining (CIDM), IEEE, 2015).

Ознакомьтесь с описанием и скачайте датасет creditcard.csv с сайта Kaggle.com по ссылке:

<https://www.kaggle.com/mlg-ulb/creditcardfraud>

Данный датасет является примером несбалансированных данных, так как мошеннические операции с картами встречаются реже обычных.

Импортруйте библиотеку Pandas, а также используйте для графиков стиль “fivethirtyeight”.

Посчитайте с помощью метода value\_counts количество наблюдений для каждого значения целевой переменной Class и примените к полученным данным метод plot, чтобы построить столбчатую диаграмму. Затем постройте такую же диаграмму, используя логарифмический масштаб.

На следующем графике постройте две гистограммы по значениям признака V1 - одну для мошеннических транзакций (Class равен 1) и другую - для обычных (Class равен 0). Подберите значение аргумента density так, чтобы по вертикали графика было расположено не число наблюдений, а плотность распределения. Число бинов должно равняться 20 для обеих гистограмм, а коэффициент alpha сделайте равным 0.5, чтобы гистограммы были полупрозрачными и не загораживали друг друга. Создайте легенду с двумя значениями: “Class 0” и “Class 1”. Гистограмма обычных транзакций должна быть серого цвета, а мошеннических - красного. Горизонтальной оси дайте название “Class”.

**Решение:**

data=pd.read\_csv('D:\credit\_csv\creditcard.csv')

data.shape

plt.style.use('fivethirtyeight')

data\_info = data['Class'].value\_counts()

data\_info

plt.plot(data['Class'], data['Class'].values)

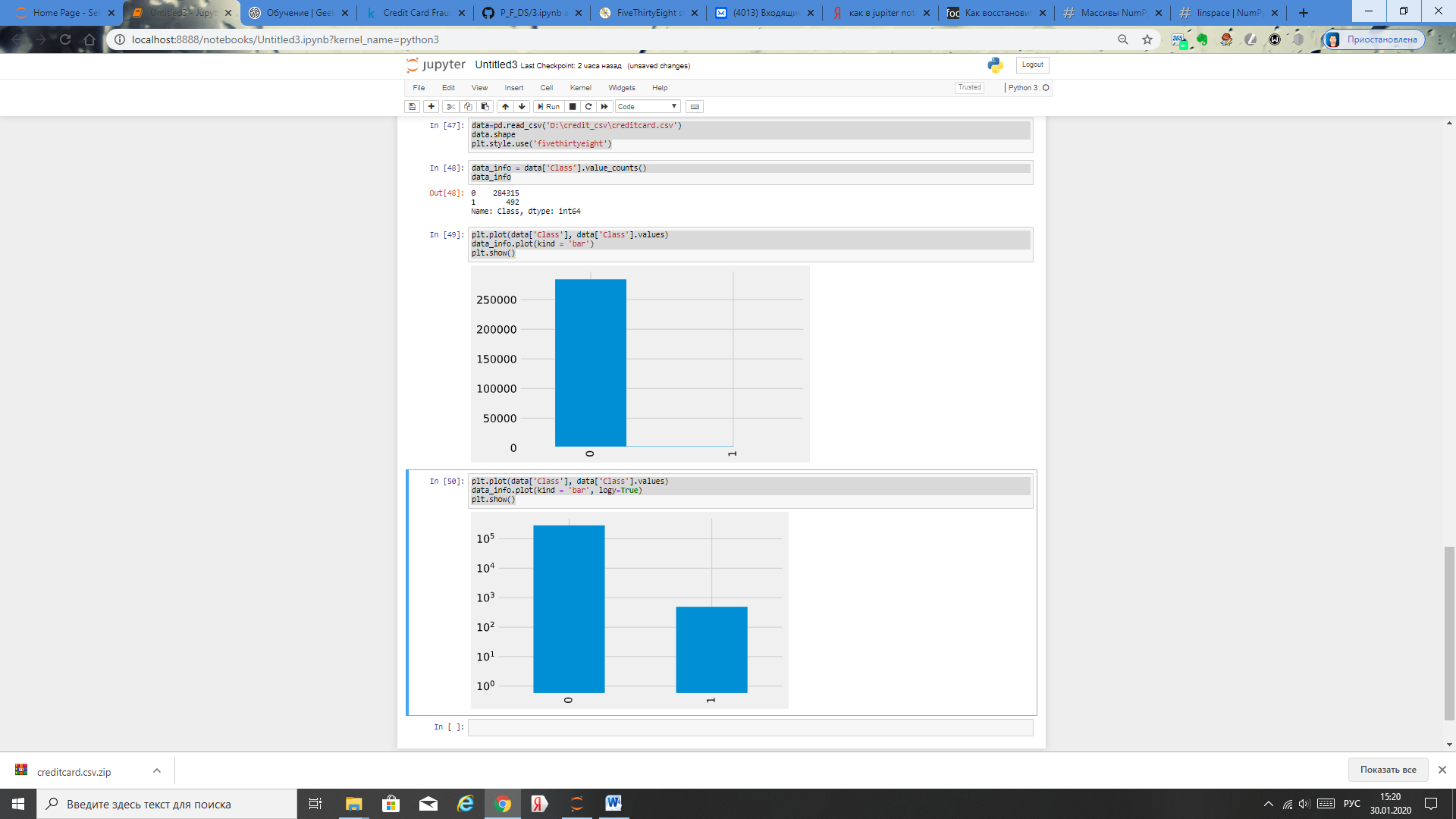
data\_info.plot(kind = 'bar')

plt.show()

plt.plot(data['Class'], data['Class'].values)

data\_info.plot(kind = 'bar', logy=True)

plt.show()



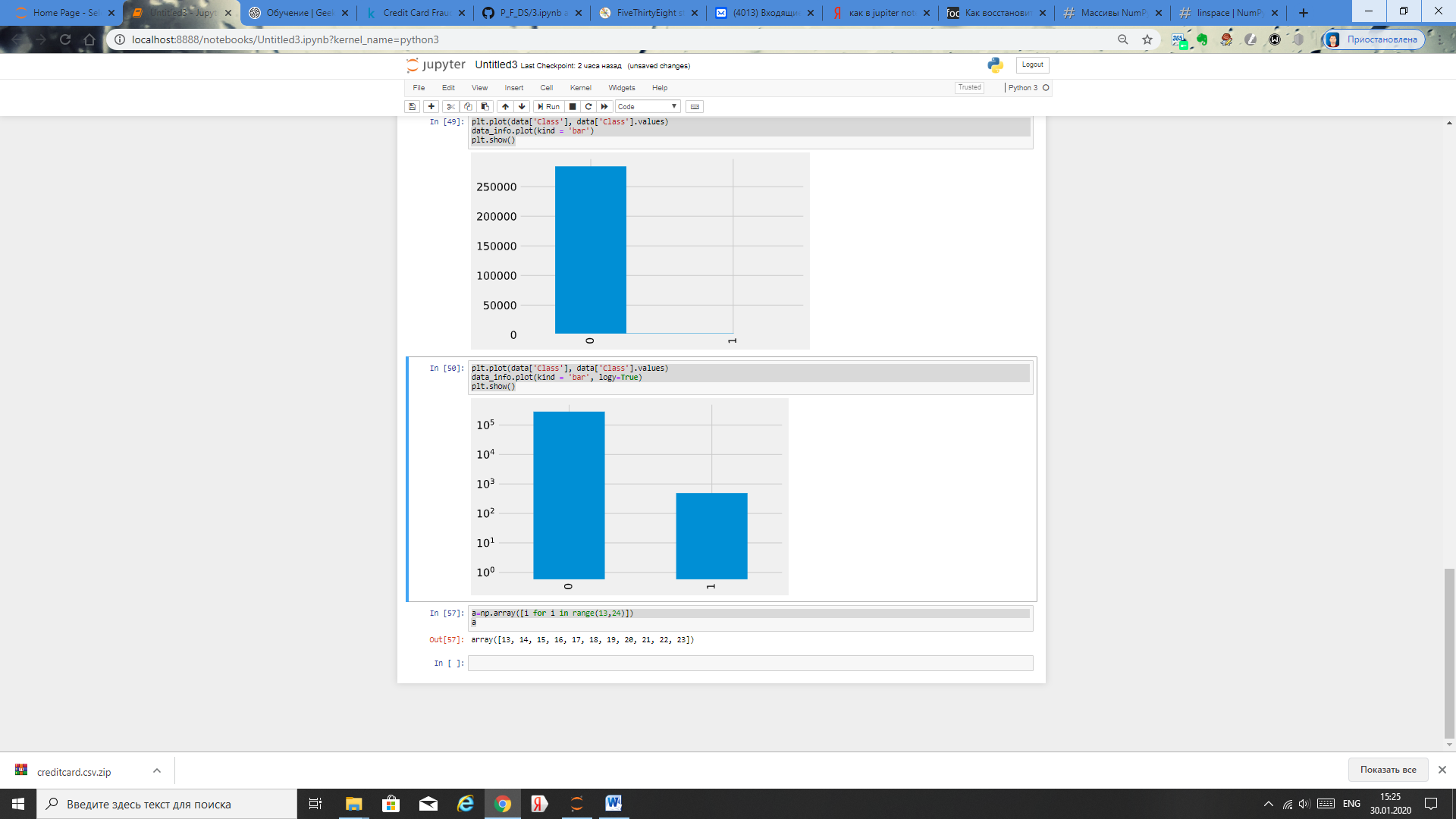
**\*\*Задание на повторение материала**

1. Создать одномерный массив Numpy под названием a из 12 последовательных целых чисел чисел от 12 до 24 невключительно

**Решение:**

a=np.array([i for i in range(13,24)])

a



1. Создать 5 двумерных массивов разной формы из массива a. Не использовать в аргументах метода reshape число -1.

**Решение:**

b1=np.array([[a[0:3]], [a[0:3]]])

b1

b2=np.array([[a[4:6]], [a[4:6]]])

b2

b3=np.array([[a[:5]], [a[1:3]]])

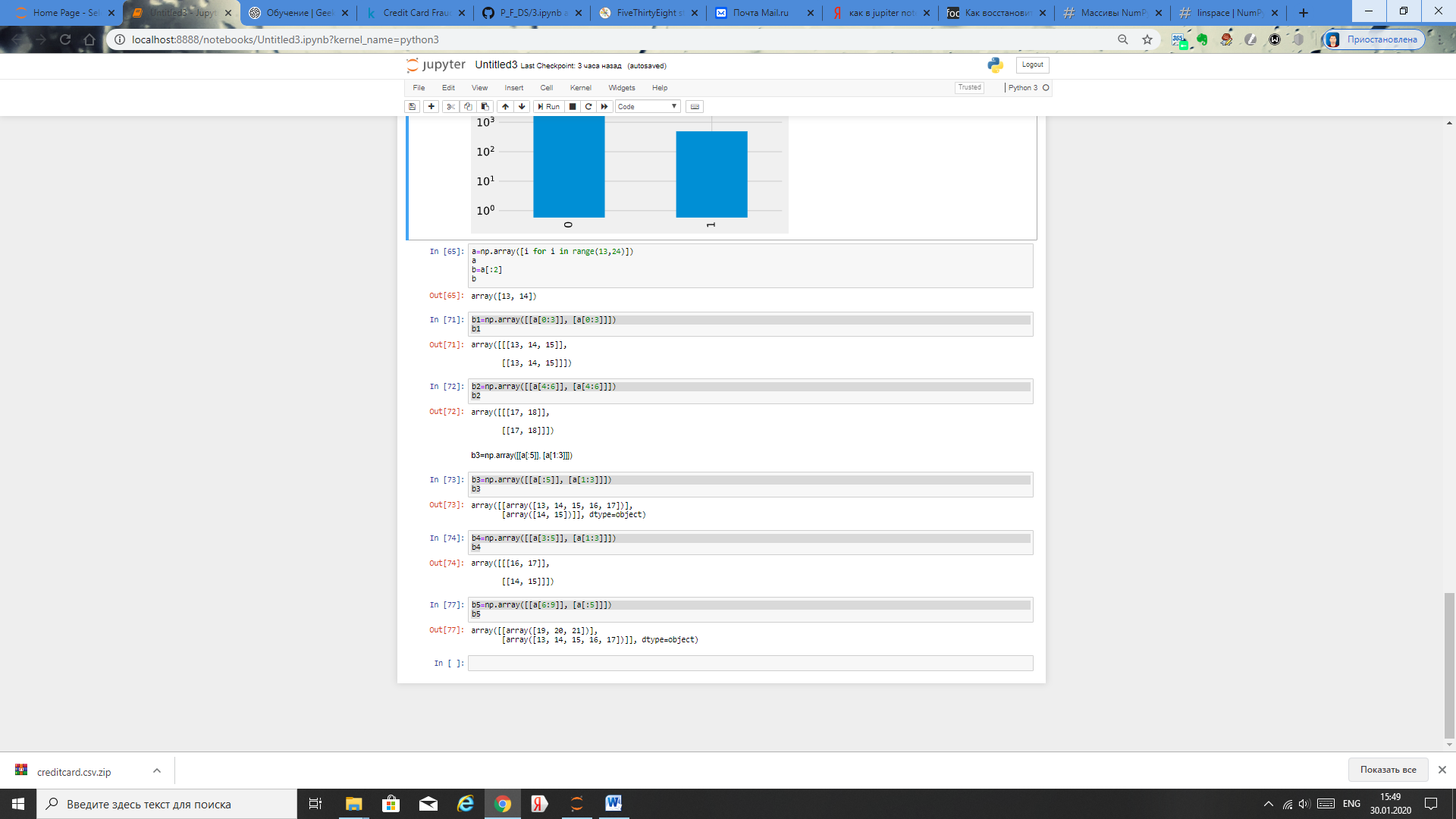
b3

b4=np.array([[a[3:5]], [a[1:3]]])

b4

b5=np.array([[a[6:9]], [a[:5]]])

b5



1. Создать 5 двумерных массивов разной формы из массива a. Использовать в аргументах метода reshape число -1 (в трех примерах - для обозначения числа столбцов, в двух - для строк).

**Решение:**

Используя массивы a1-a5 из задания №2 создаём новые массивы:

c1=np.reshape(b1, (2,3))

c1

c2=np.reshape(b2, (2,2))

c2

c3=np.reshape(b3, (1,2))

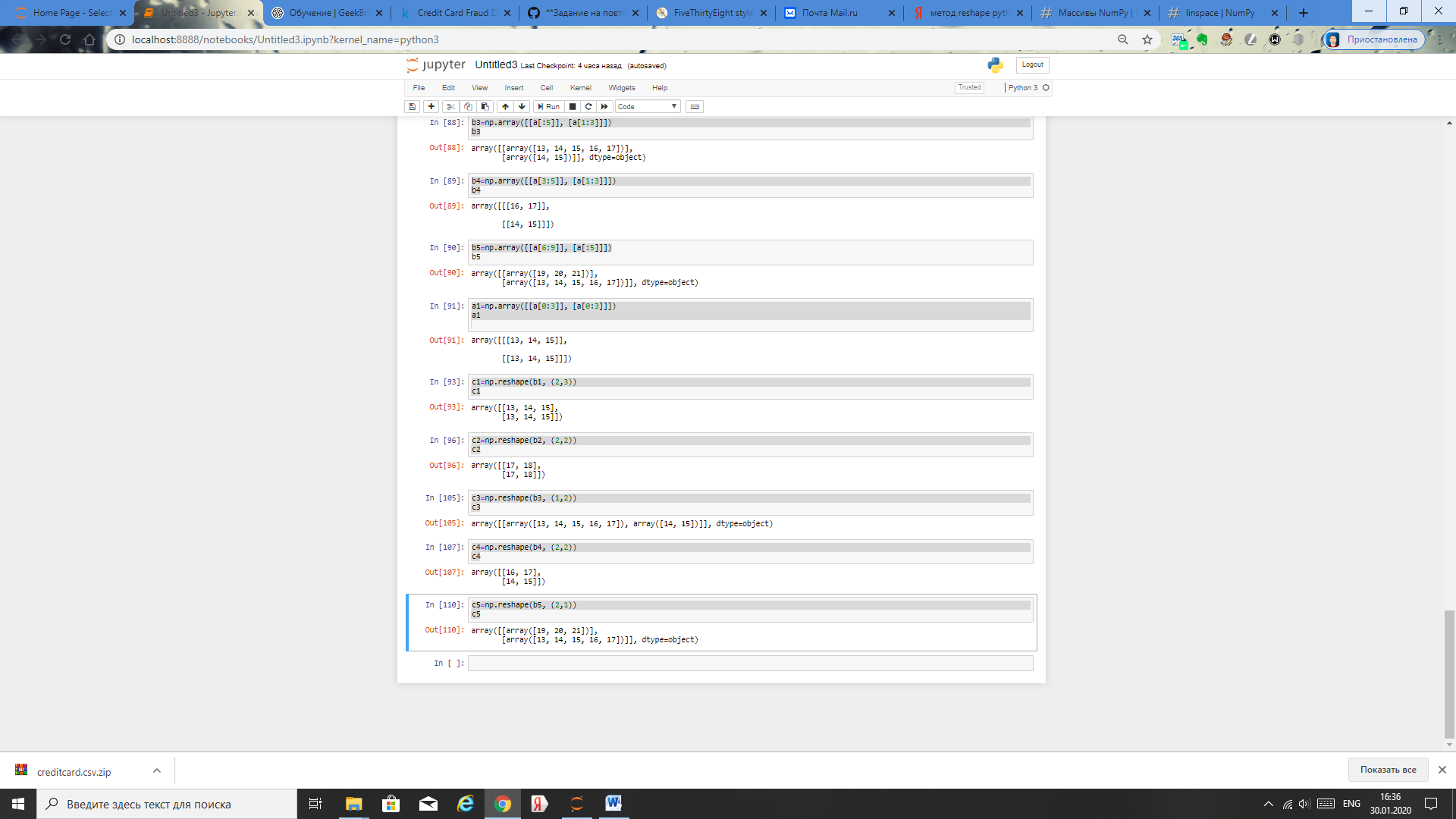
c3

c4=np.reshape(b4, (2,2))

c4

c5=np.reshape(b5, (2,1))

c5



1. Можно ли массив Numpy, состоящий из одного столбца и 12 строк, назвать одномерным?

**Решение:** нет, нельзя, потому что массив – это строка, а столбец из 12 строк – это 12 массивов с 1 значением

1. Создать массив из 3 строк и 4 столбцов, состоящий из случайных чисел с плавающей запятой из нормального распределения со средним, равным 0 и среднеквадратичным отклонением, равным 1.0. Получить из этого массива одномерный массив с таким же атрибутом size, как и исходный массив.

**Решение:**

x = np.random.randn(3, 4)

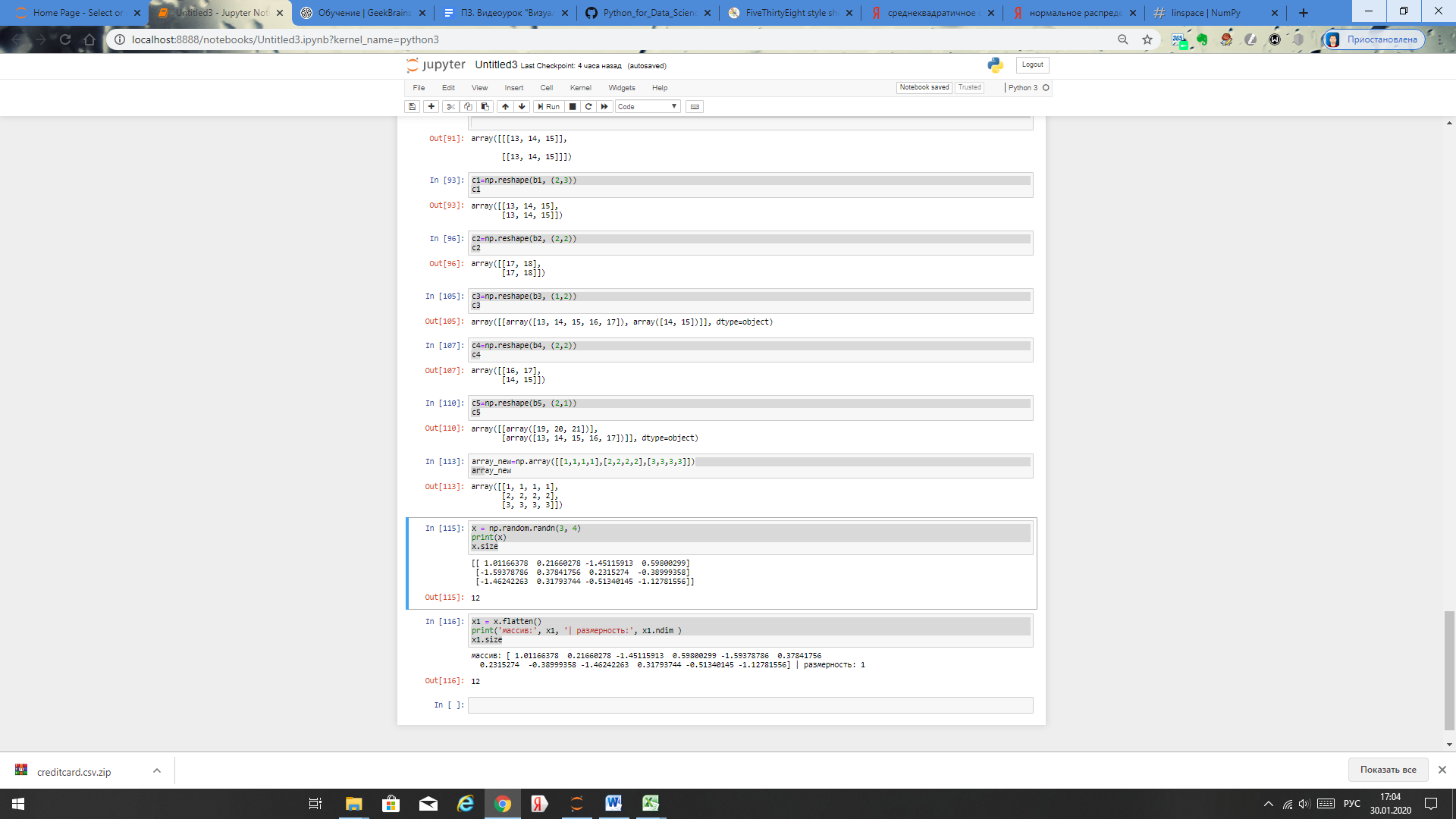
print(x)

x.size

x1 = x.flatten()

print('массив:', x1, '| размерность:', x1.ndim )

x1.size



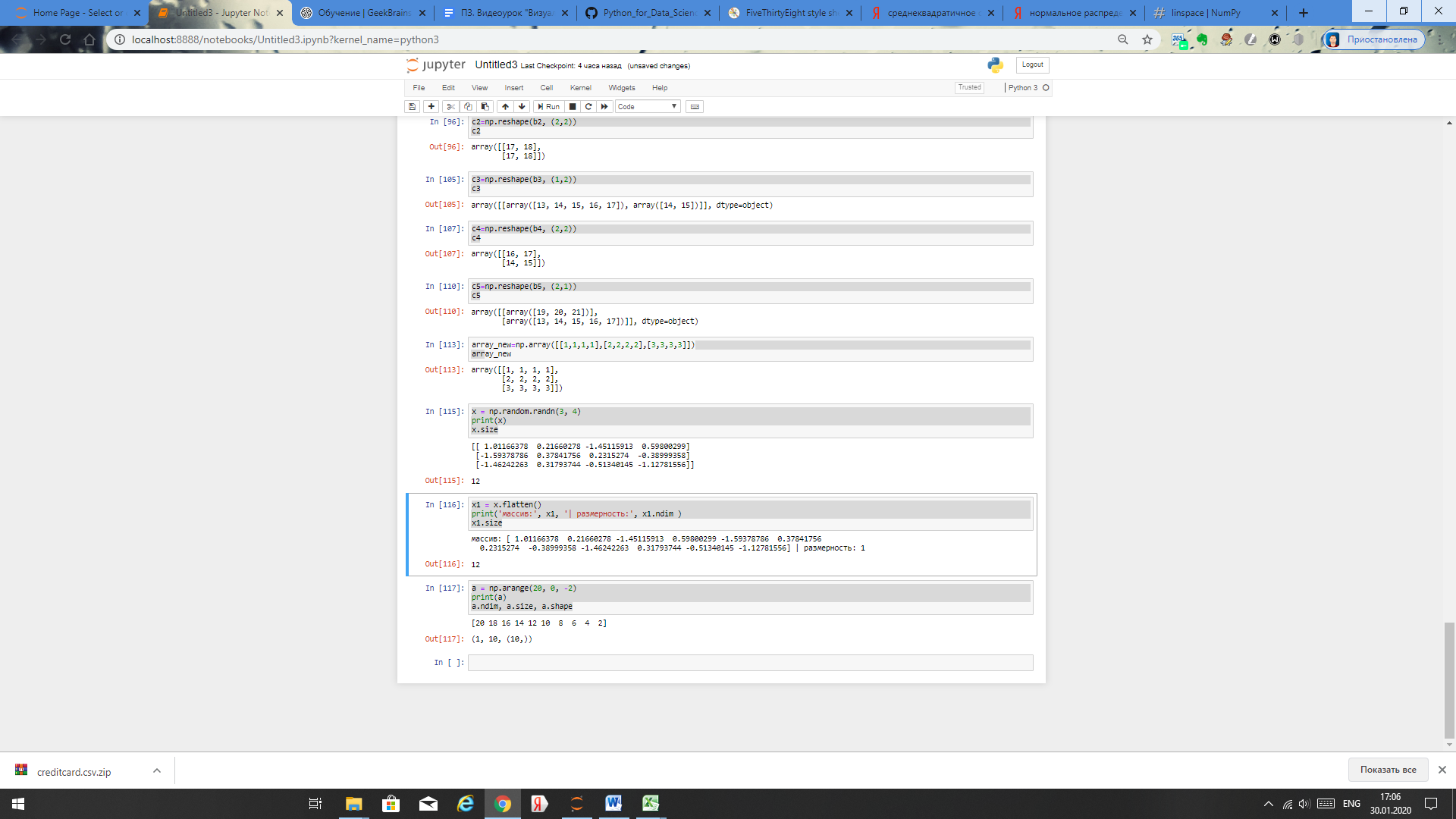
1. Создать массив a, состоящий из целых чисел, убывающих от 20 до 0 невключительно с интервалом 2.

Решение:

a = np.arange(20, 0, -2)

print(a)

a.ndim, a.size, a.shape



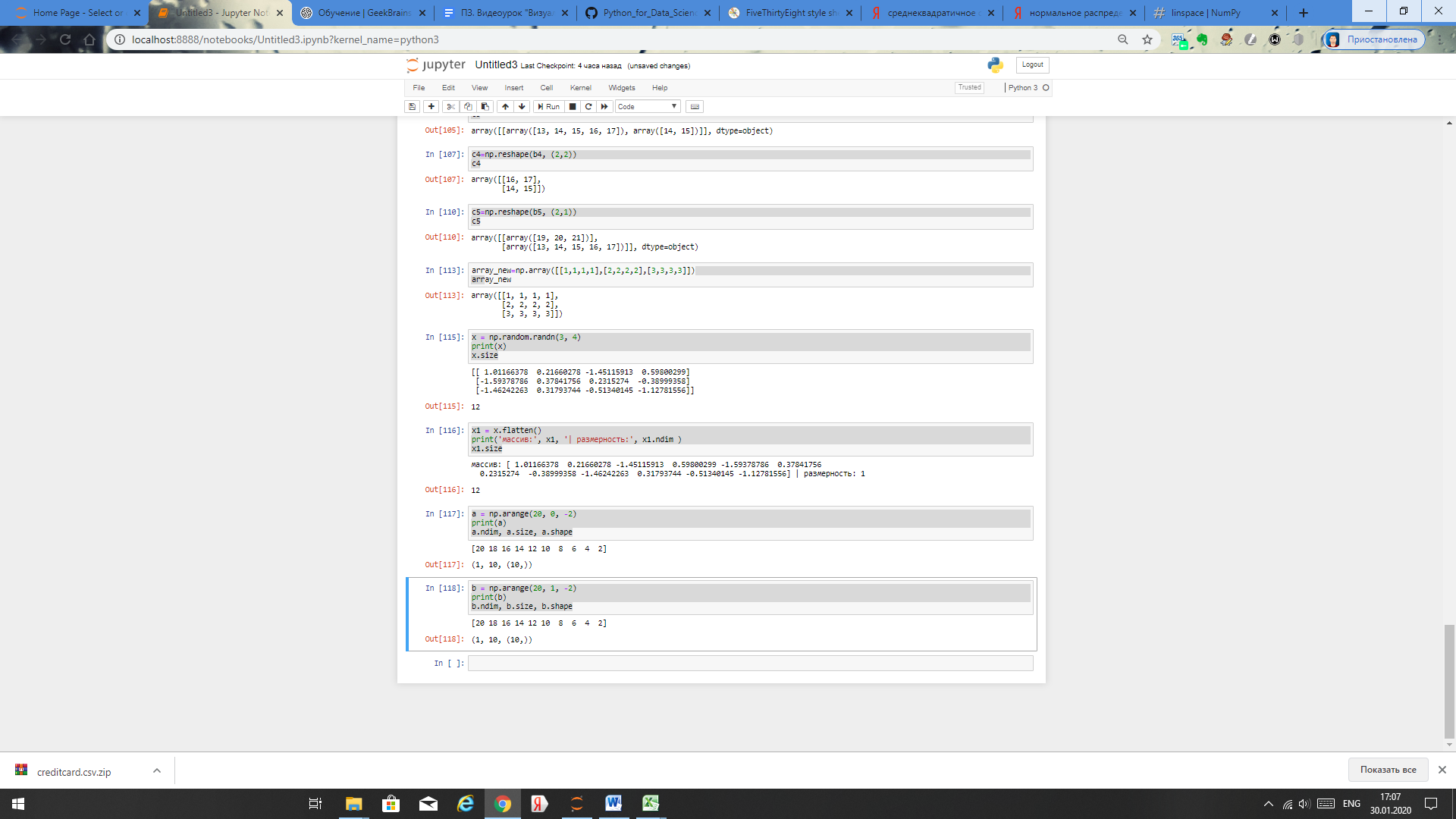
1. Создать массив b, состоящий из 1 строки и 10 столбцов: целых чисел, убывающих от 20 до 1 невключительно с интервалом 2. В чем разница между массивами a и b?

Решение:

b = np.arange(20, 1, -2)

print(b)

b.ndim, b.size, b.shape



1. Вертикально соединить массивы a и b. a - двумерный массив из нулей, число строк которого больше 1 и на 1 меньше, чем число строк двумерного массива b, состоящего из единиц. Итоговый массив v должен иметь атрибут size, равный 10.

**Решение:**

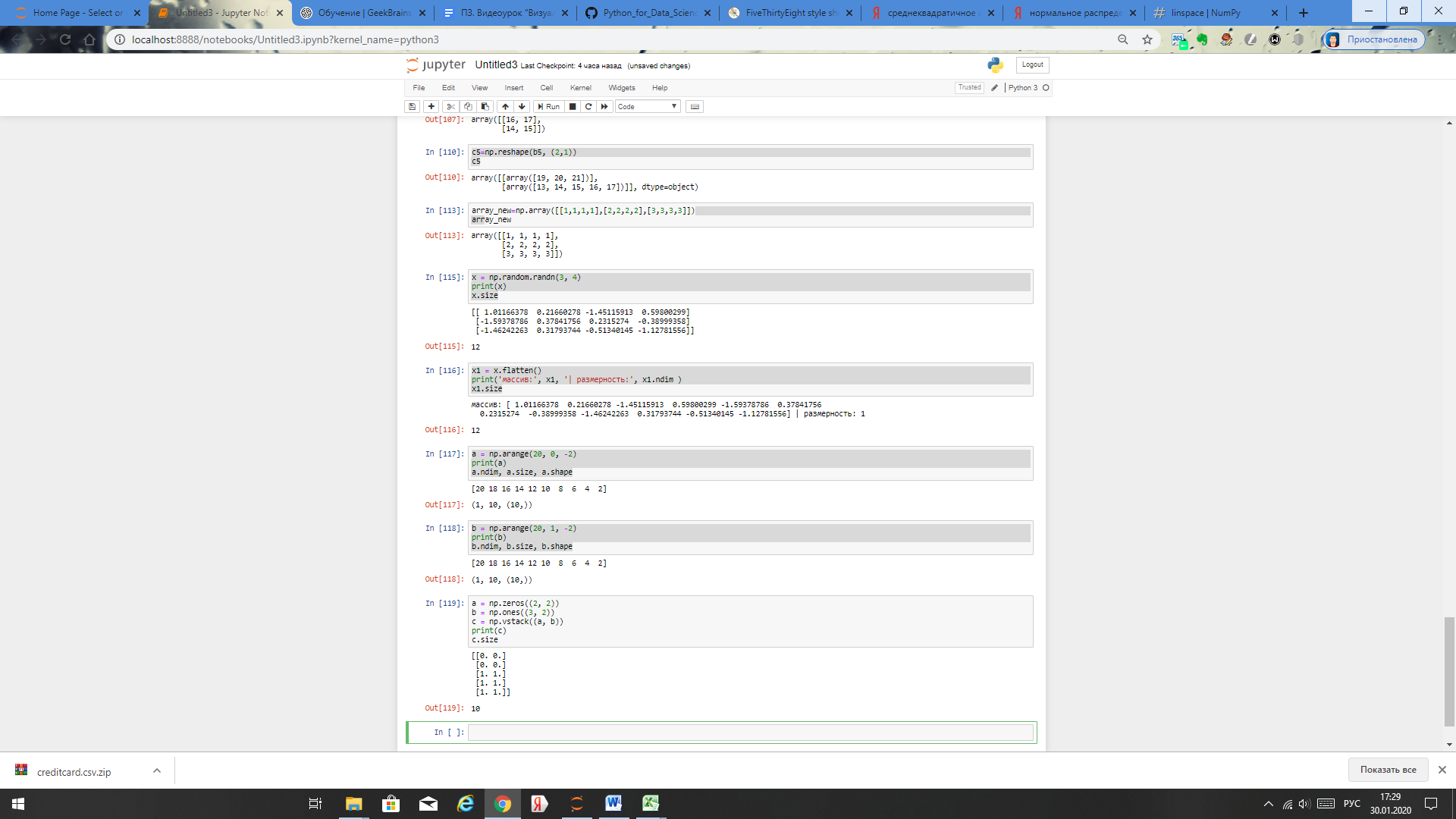
a = np.zeros((2, 2))

b = np.ones((3, 2))

c = np.vstack((a, b))

print(c)

c.size



1. Создать одномерный массив а, состоящий из последовательности целых чисел от 0 до 12. Поменять форму этого массива, чтобы получилась матрица A (двумерный массив Numpy), состоящая из 4 строк и 3 столбцов. Получить матрицу At путем транспонирования матрицы A. Получить матрицу B, умножив матрицу A на матрицу At с помощью матричного умножения. Какой размер имеет матрица B? Получится ли вычислить обратную матрицу для матрицы B и почему?

**Решение:**

a = np.arange(12)

print(a)

A = a.reshape(4, 3)

print(A)

At = A.T

print(At)

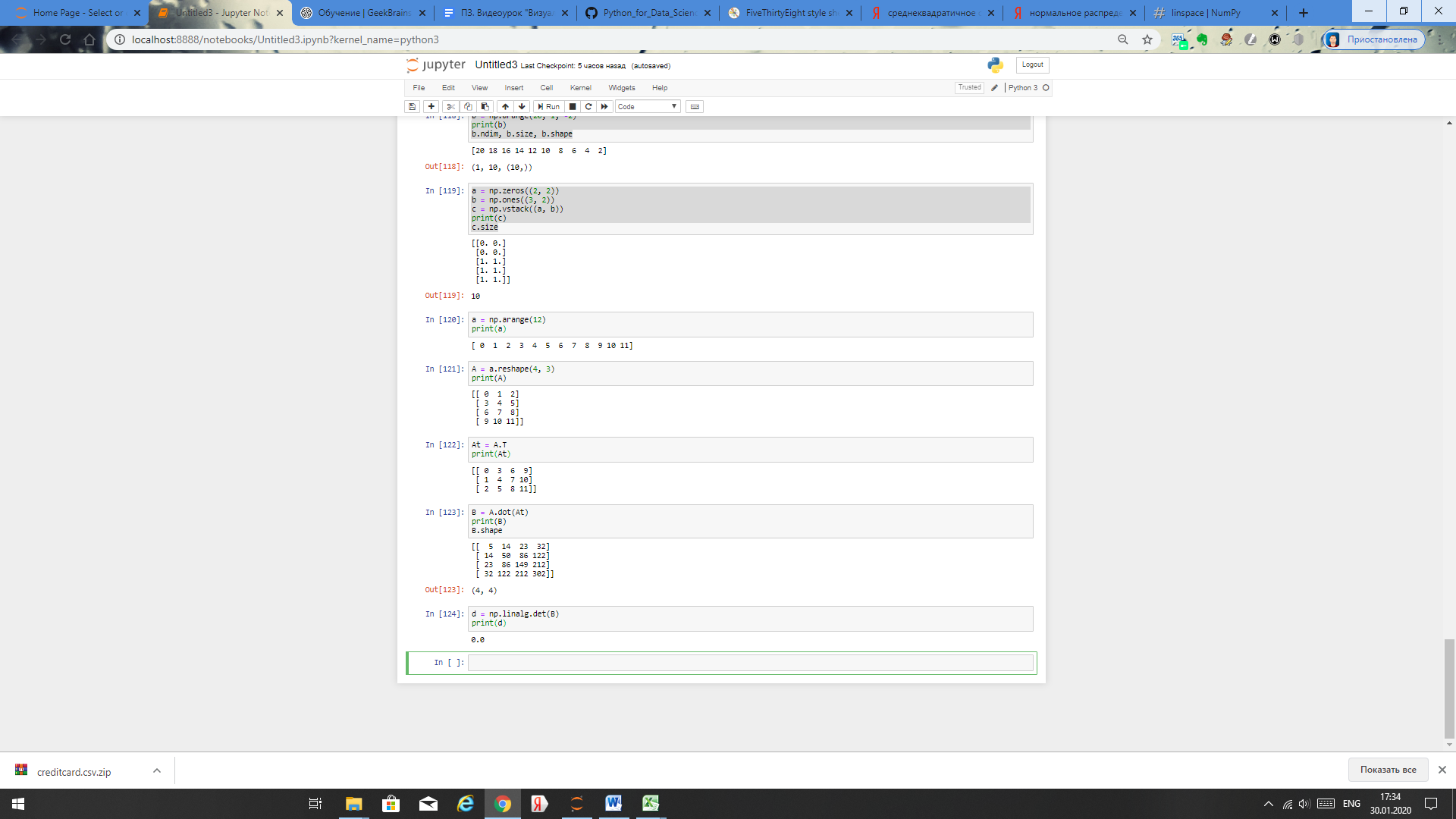
B = A.dot(At)

print(B)

B.shape

d = np.linalg.det(B)

print(d)



1. Инициализируйте генератор случайных числе с помощью объекта seed, равного 42.

**Решение:**

np.random.seed(42)

1. Создайте одномерный массив c, составленный из последовательности 16-ти случайных равномерно распределенных целых чисел от 0 до 16 невключительно.

Решение:

c = np.random.randint(0, 16, 16)

print(c)

1. Поменяйте его форму так, чтобы получилась квадратная матрица C. Получите матрицу D, поэлементно прибавив матрицу B из предыдущего вопроса к матрице C, умноженной на 10. Вычислите определитель, ранг и обратную матрицу D\_inv для D.

Решение:

C = c.reshape(4, 4) print(C)

D = B + C \* 10

print(D)

d = np.linalg.det(D)

print(d)

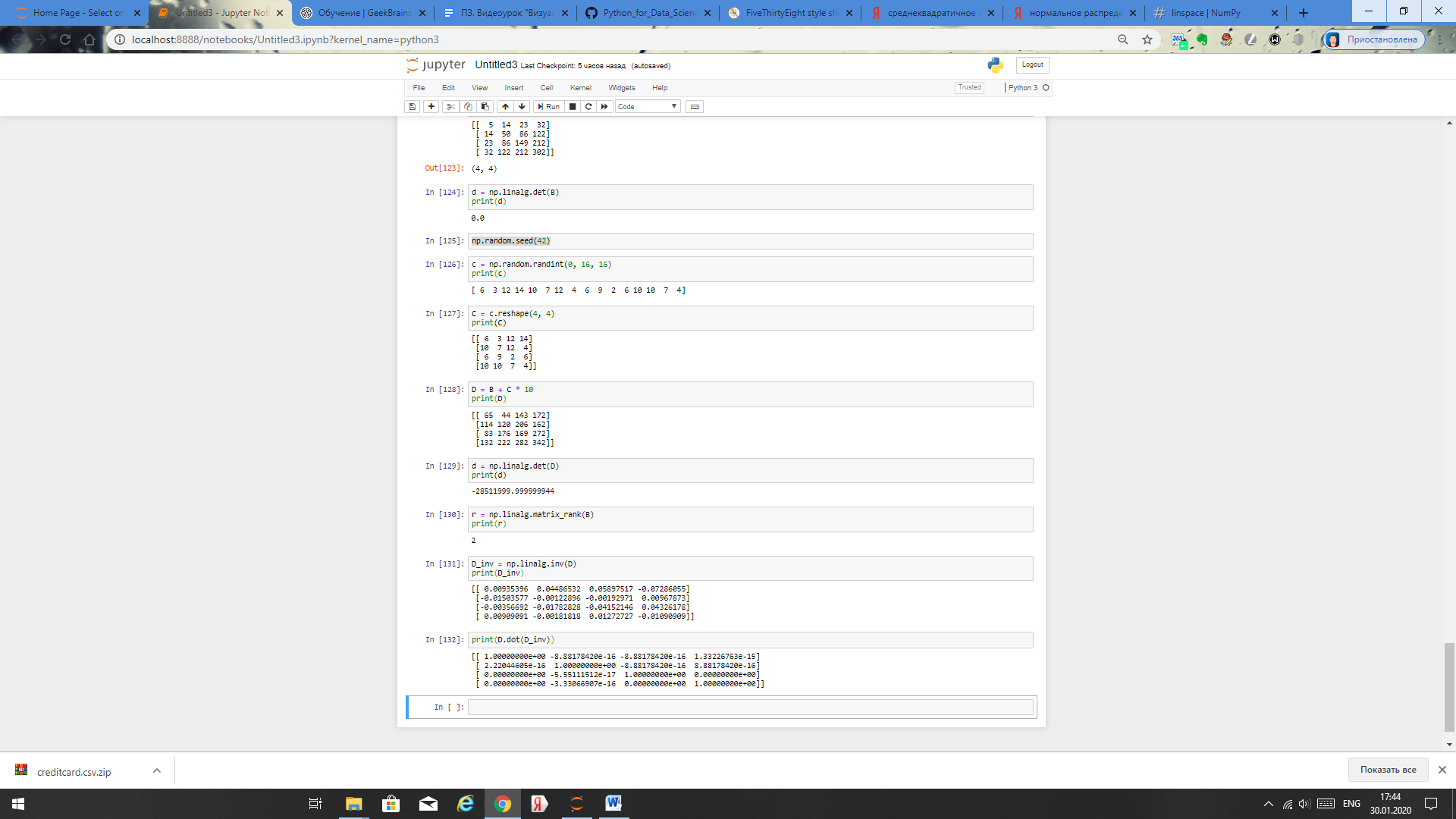
r = np.linalg.matrix\_rank(B)

print(r)

D\_inv = np.linalg.inv(D)

print(D\_inv)

print(D.dot(D\_inv))



1. Приравняйте к нулю отрицательные числа в матрице D\_inv, а положительные - к единице. Убедитесь, что в матрице D\_inv остались только нули и единицы. С помощью функции numpy.where, используя матрицу D\_inv в качестве маски, а матрицы B и C - в качестве источников данных, получите матрицу E размером 4x4.  Элементы матрицы E, для которых соответствующий элемент матрицы D\_inv равен 1, должны быть равны соответствующему элементу матрицы B, а элементы матрицы E, для которых соответствующий элемент матрицы D\_inv равен 0, должны быть равны соответствующему элементу матрицы C.

**Решение:**

print(D\_inv)

D\_inv = np.where(D\_inv < 0, 0, 1)

print(D\_inv)

E = np.where(D\_inv > 0, B, C)

print(E)

