Министерство образования Республики Беларусь

УО «Брестский государственный технический университет»

Кафедра ИИТ

**Лабораторная работа №7**

По дисциплине: “Языки программирования”

Тема: «Изучение NumPy. Сравнение производительности с классическими библиотеками Python»

**Выполнил**:

студент 2 курса

группы ПО-7

Курмыса Е.Е.

**Проверила:**

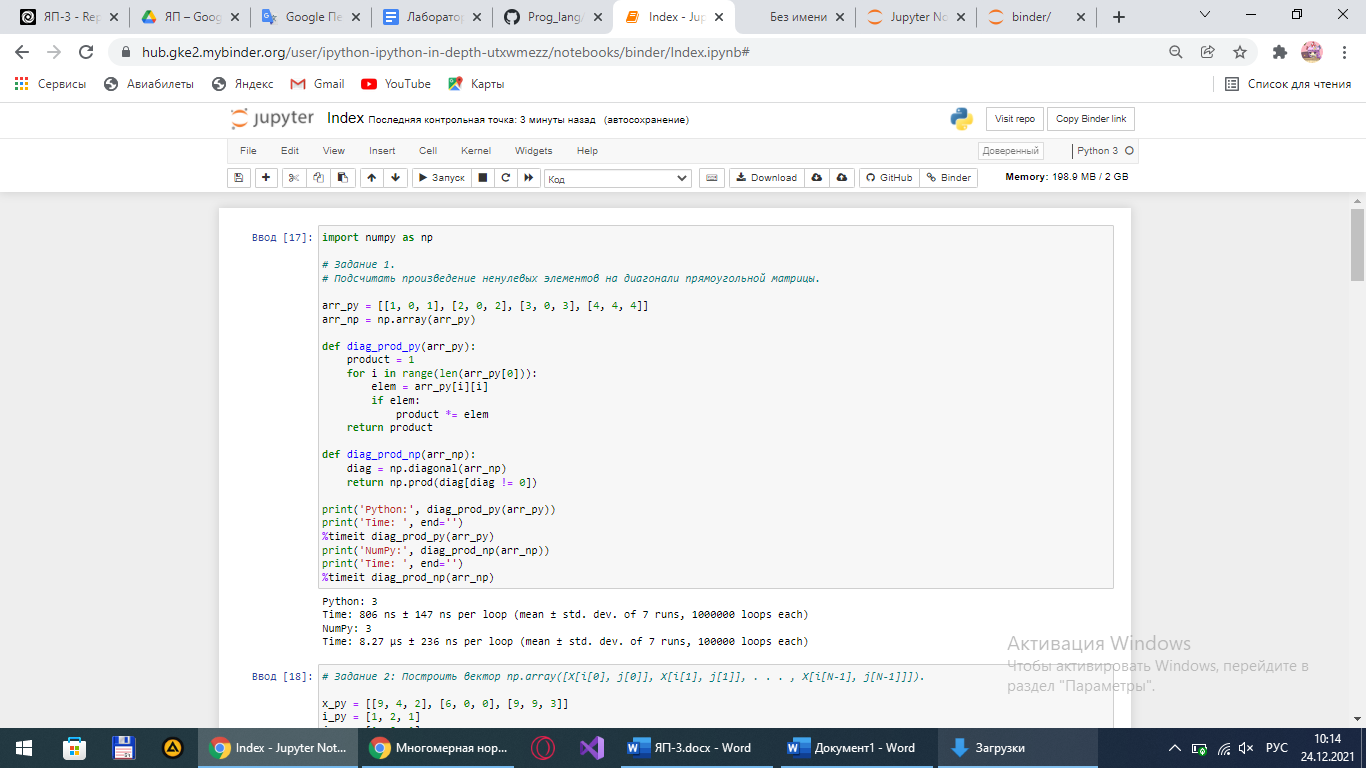
Дряпко А. В.

Брест 2021

**Цель работы**: изучение некоторых функций NumPy и сравнение работоспособности, точности и времени выполнения с аналогичными функциями, написанными на Python либо Scipy.

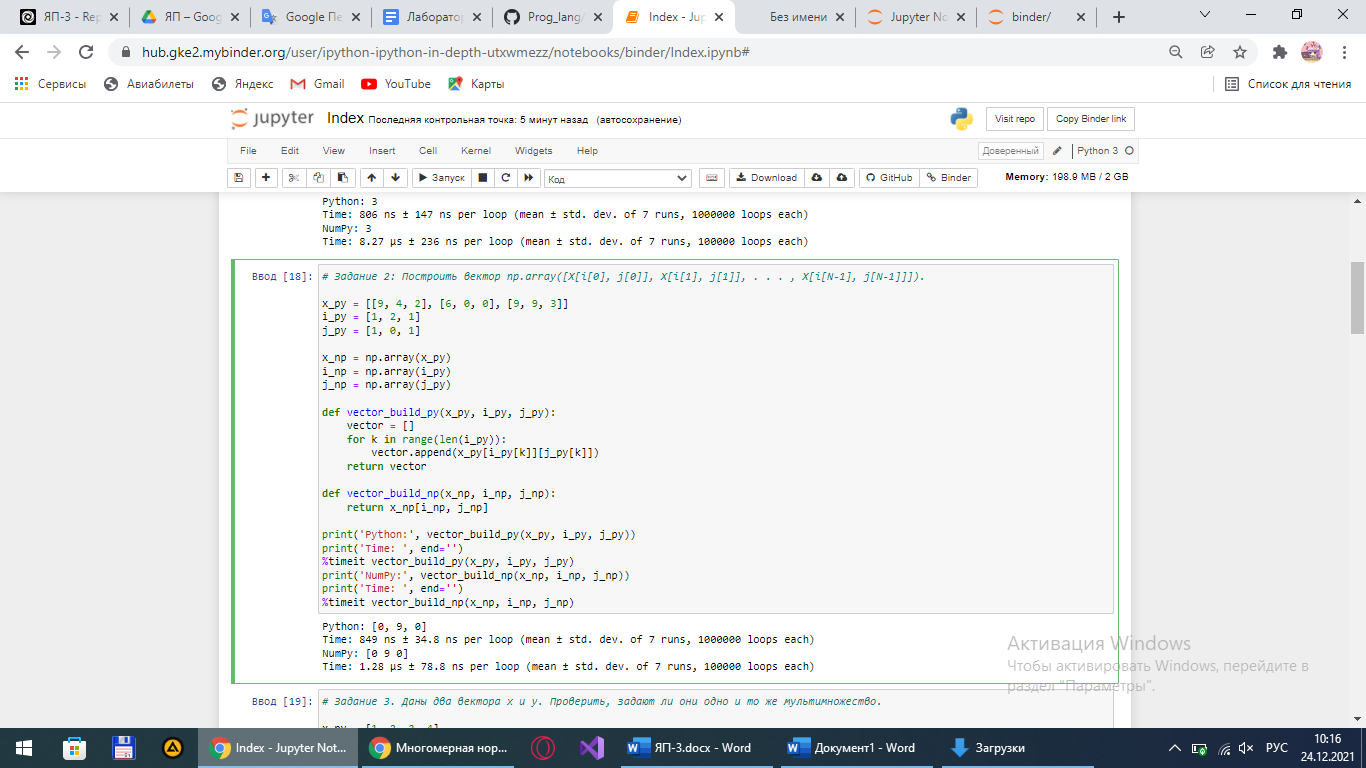
**Ход работы**

1. Подсчитать произведение ненулевых элементов на диагонали прямоугольной матрицы.



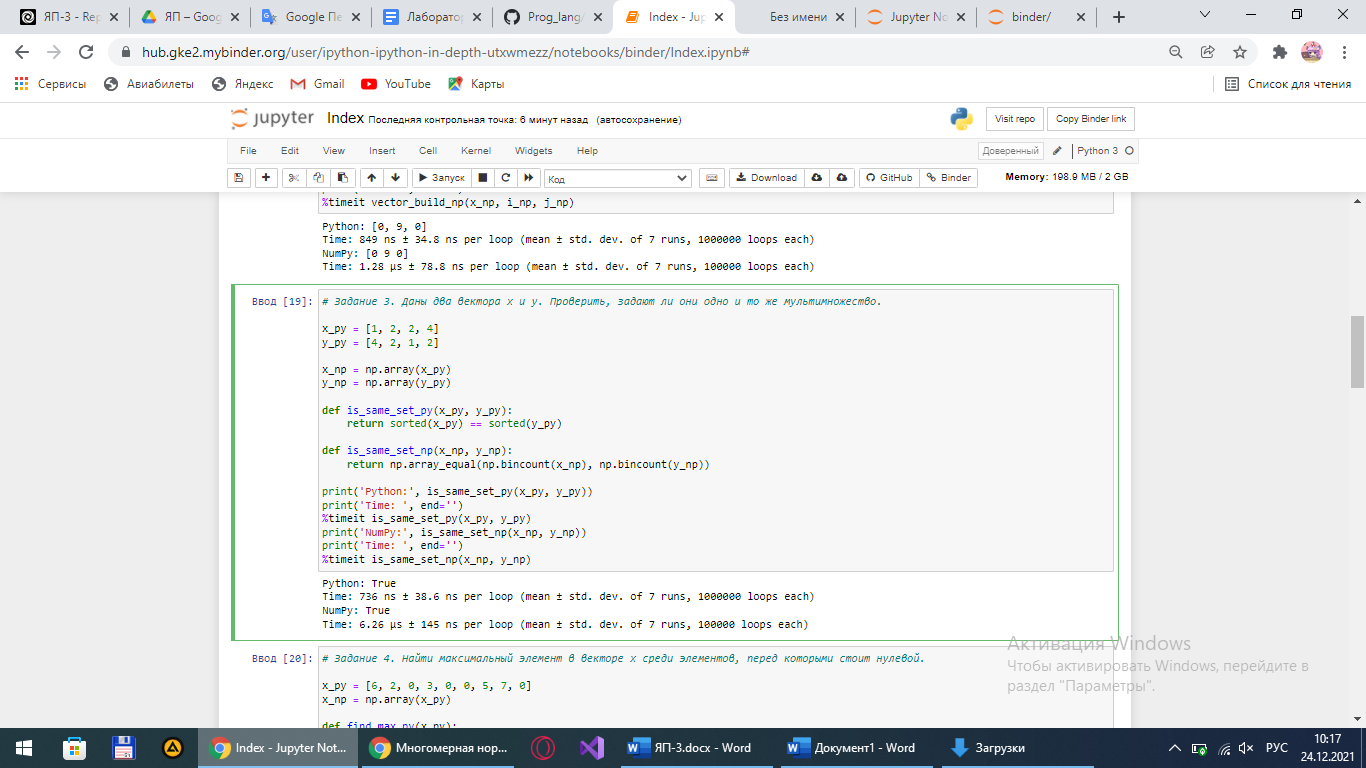
Заметим, что запись на «чистом» Python несколько длиннее записи на NumPy, однако же время работы в 10 раз меньше. Кроме того, сложность написания этого кода не слишком высокая, поэтому первый вариант является более приемлемым.

1. Дана матрица x и два вектора одинаковой длины i и j. Построить вектор np.array([X[i[0], j[0]], X[i[1], j[1]], . . . , X[i[N-1], j[N-1]]]).



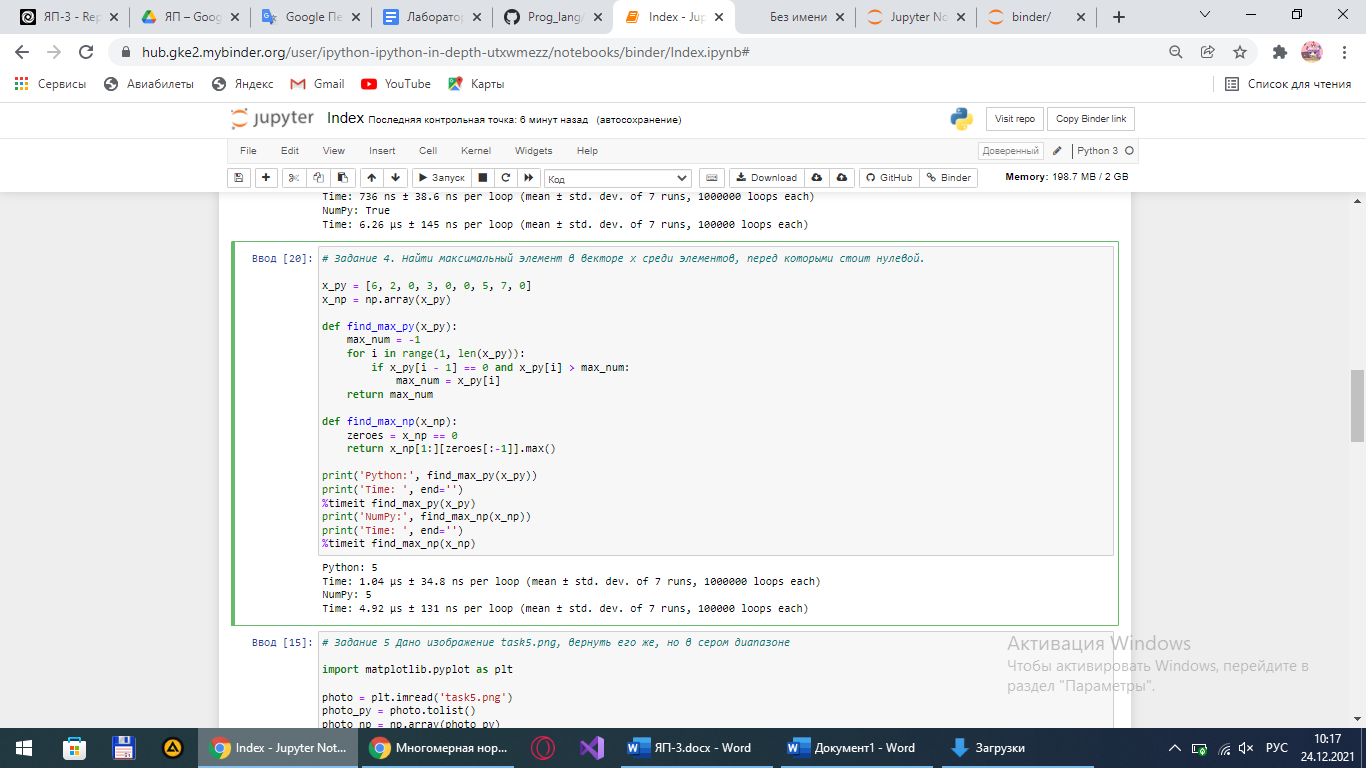
Здесь же запись на NumPy гораздо проще чистого Python, также время выполнения работы слегка больше (в 1.5 раза). В данном случае предпочтение остаётся за NumPy.

1. Даны два вектора x и y. Проверить, задают ли они одно и то же мультимножество.



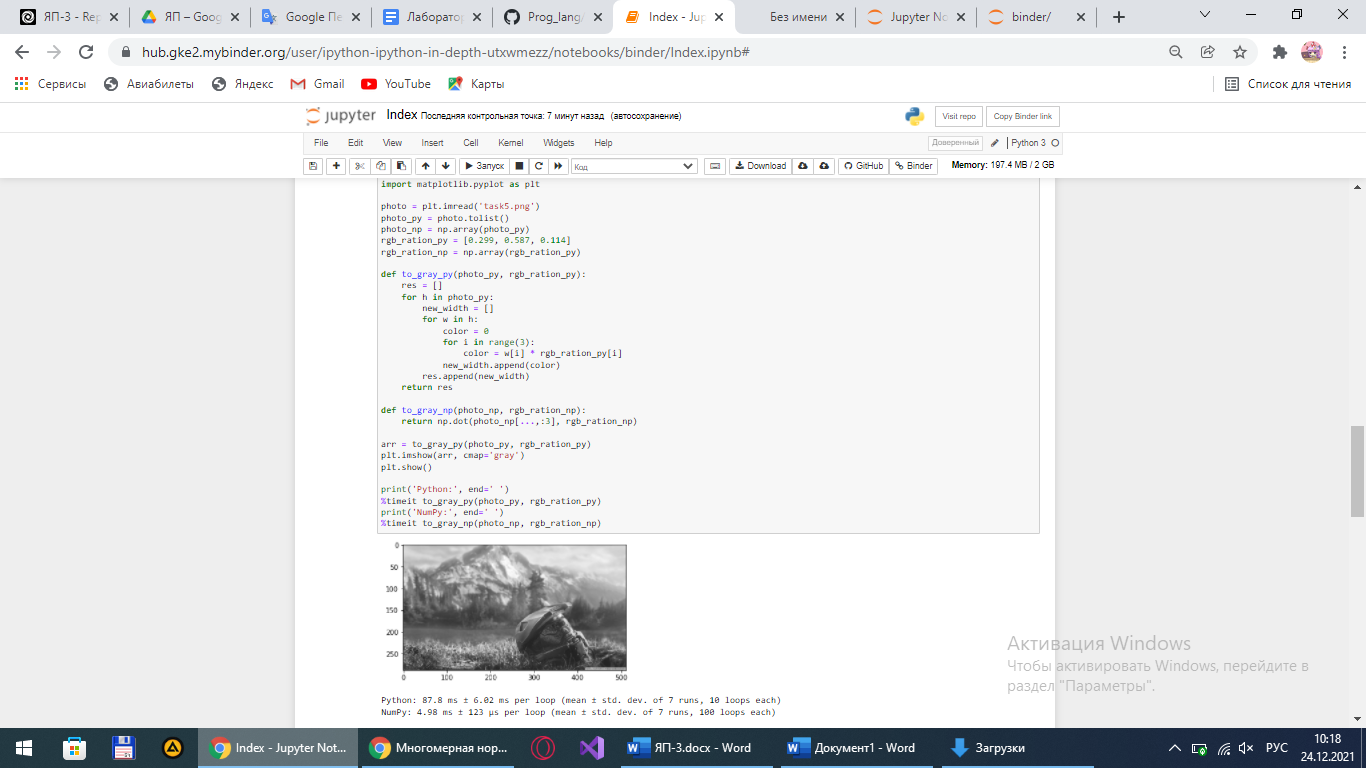
Ситуация обратна задаче 2: запись на Python короче и проще записи на NumPy + быстрее примерно в 9 раз, тем самым показывая, что задачи по мультимножествам предпочтительнее по времени писать без NumPy.

1. Найти максимальный элемент в векторе x среди элементов, перед которыми стоит нулевой.



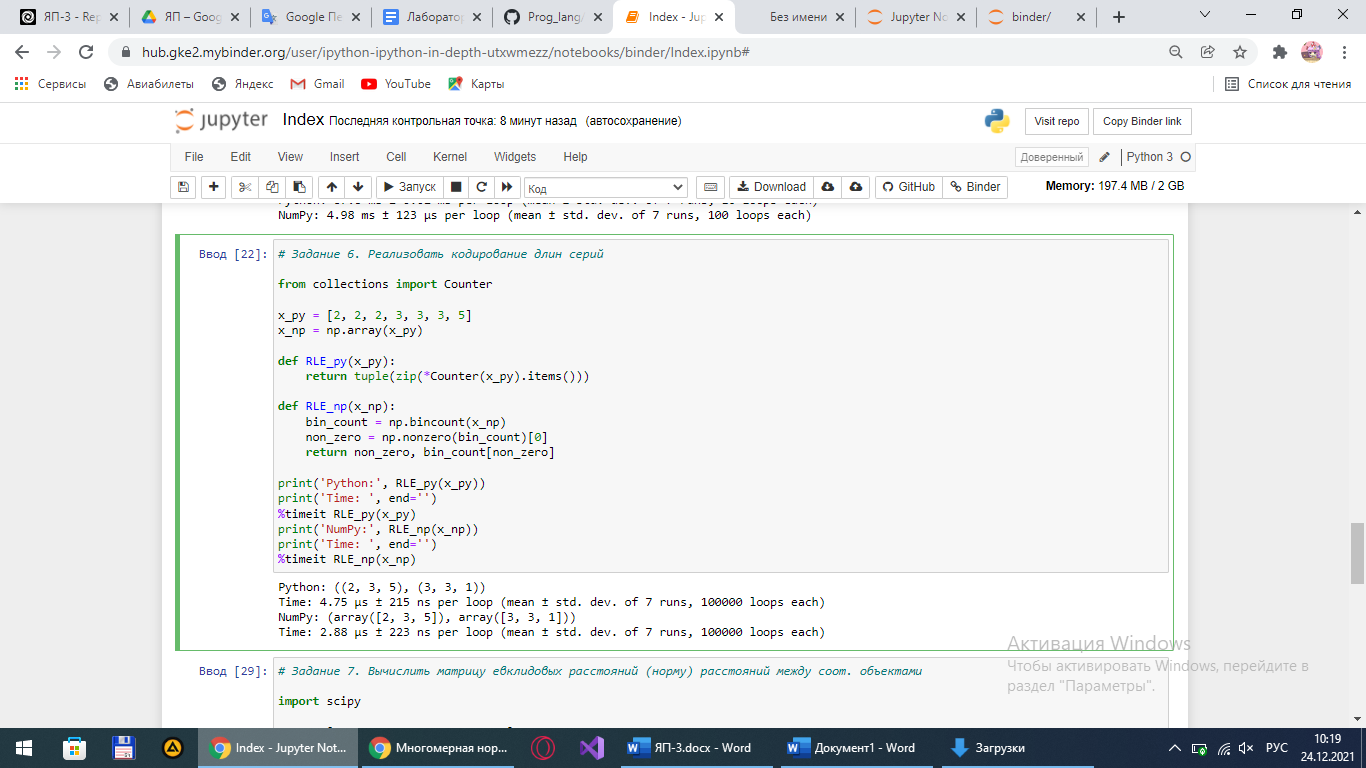
Запись на NumPy несколько короче записи на чистом Python, однако такой код будет работать в 5 раз медленнее. Исходя из этого, можно написать, что в данном случае оба варианта являются достаточно приемлемыми в зависимости от той или иной ситуации.

1. Дан трёхмерный массив, содержащий изображение, размера (height, width, numChannels), а также вектор длины numChannels. Сложить каналы изображения с указанными весами, и вернуть результат в виде матрицы размера (height, width). Считать реальное изображение можно при помощи функции scipy.misc.imread (если изображение не в формате png, установите пакет pillow: conda install pillow).  
   Преобразуйте цветное изображение в оттенки серого, использовав коэффициенты np.array([0.299, 0.587, 0.114]).



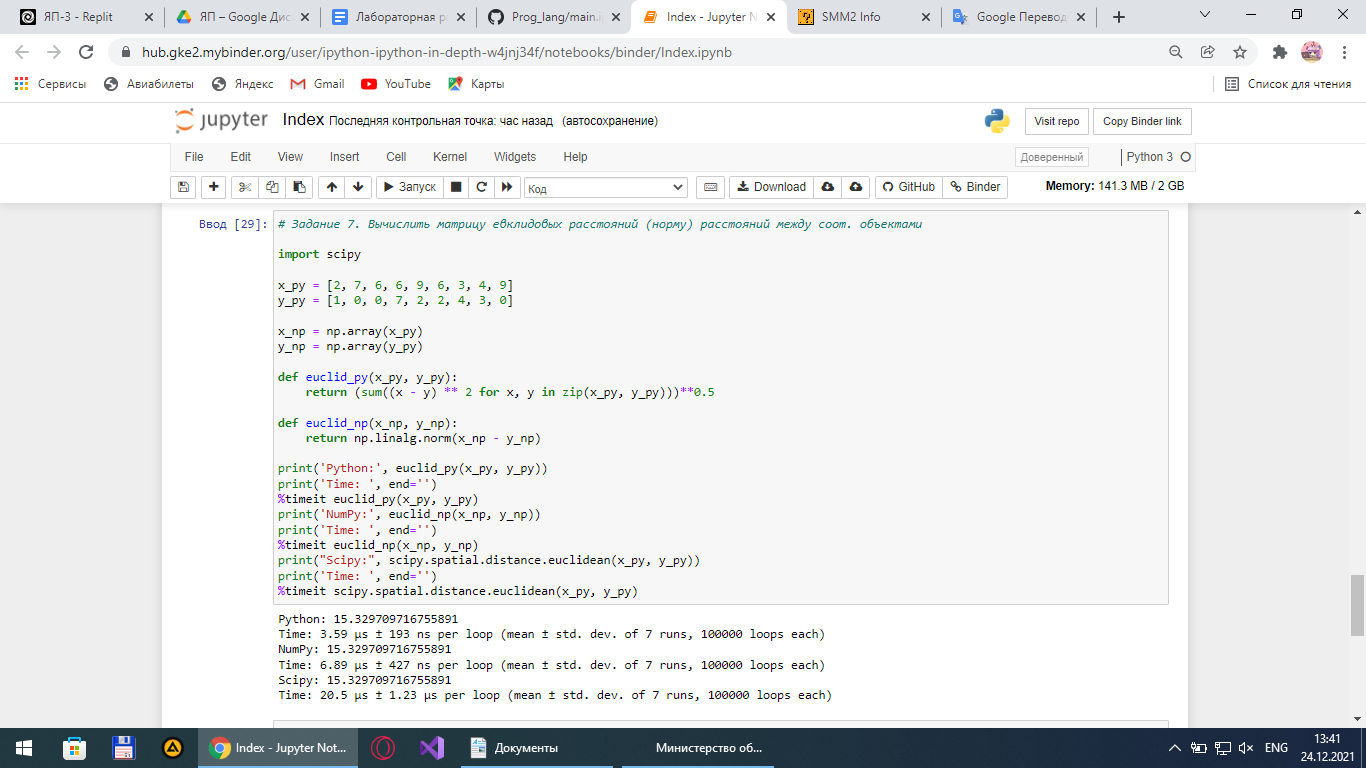
Здесь видим, что NumPy справляется с данной задачей куда быстрее, чем чистый Python + запись гораздо короче и читабельнее. Следовательно, для задач с обработкой изображений NumPy подходит лучше.

1. Реализовать кодирование длин серий (Run-length encoding). Дан вектор x. Необходимо вернуть кортеж из двух векторов одинаковой длины. Первый содержит числа, а второй - сколько раз их нужно повторить.



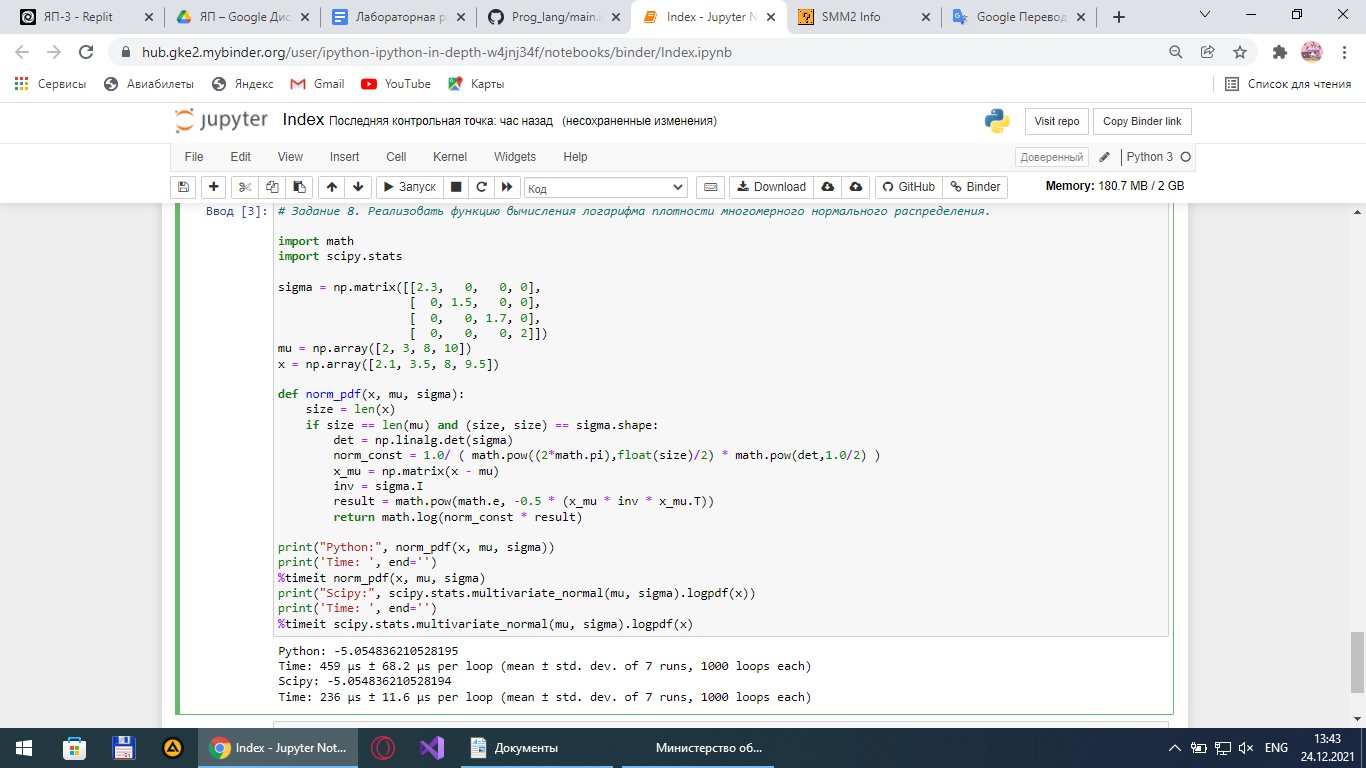
Здесь мы имеем относительно одинаковые результаты как по коду, так и по времени, поэтому можно рекомендовать для подобных задач оба варианта.

1. Даны две выборки объектов - X и Y. Вычислить матрицу евклидовых расстояний между объектами.  
   Сравнить с функцией scipy.spatial.distance.euclidean.



Оба варианта работают быстрее варианта на scipy, однако, если код на Python выглядит немного замысловато, но работает быстро, вариант на NumPy имеет противоположные качества: работает медленнее, но понятнее. Итак, в данном случае предпочтение уделяется обоим вариантам.

8. Реализовать функцию вычисления логарифма плотности многомерного нормального распределения. Входные параметры: точки X, размер (N, D), мат. ожидание m, вектор длины D, матрица ковариаций C, размер (D, D). Разрешается использовать библиотечные функции для подсчета определителя матрицы, а также обратной матрицы, в том числе в невекторизованном варианте. Сравнить с scipy.stats.multivariate\_normal(m, C).logpdf(X) как по скорости работы, так и по точности вычислений



Наконец, вариант на Python/NumPy работает медленее, чем на Scipy, плюс требует больше знаний и кода, нежели однострочный multivariate\_normal. Следовательно, здесь предпочтение отдаётся последнему варианту.

**Общие выводы:** NumPy — достаточно удобная библиотека ЯП Python, которая иногда позволяет сократить вычисления и ускорить их. Однако, как было указано в примерах выше, в некоторых случаях вариант от NumPy работает медленнее того же варианта, но на чистом Python; иногда запись выглядит сложнее, иногда нет. В ряде случаев один вариант выигрывает по читабельности, а другой — по времени. Таким образом, для достаточно сложных вычислений в плане линейной алгебры, обработки фотографий и прочего NumPy является достаточно мощным инструментом. В других случаях, если важно время исполнения, можно обойтись и без данной библиотеки.