Министерство науки и высшего образования Российской Федерации

Федеральное государственное автономное образовательное учреждение высшего

образования «Южно-Уральский государственный университет (национальный

исследовательский университет)»

Высшая школа экономики и управления

Кафедра Информационных технологий в экономике

Программирование на языке Python (курс молодого бойца)

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА К КУРСОВОЙ РАБОТЕ

по дисциплине «ПРОГРАММИРОВАНИЕ»

ЮУрГУ – 38.03.05.2023.301/15. ПЗ КР

Рецензент, \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Руководитель, \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_/\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_/ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_/\_\_\_\_\_\_\_\_\_\_\_\_\_\_/

«\_\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_2020г. «\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_\_2020г.

Нормоконтролёр, \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Автор, студент группы ЭУ-120

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_/\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_/ Хачлаева А.З./\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_/

«\_\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_2020г.

Работа защищена с оценкой

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_/\_\_\_\_\_\_\_\_\_\_\_\_\_\_/

«\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_2020г.

Челябинск 2020

# АННОТАЦИЯ

Хачлаева А.З. Программирование

на языке Python (курс молодого бойца).

– Челябинск: ЮУрГУ, ЭУ-120, 29 с.,

библиогр. Список - 4

Данная работа показывает мои базовые знания языка программирования Python и библиотеки NumPy. Для этого были рассмотрены три алгоритма сортировки: bubble, select, insert; 31 задача с применением библиотеки NumPy и решение системы линейных уравнений при помощи метода Гаусса.

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ……………………………………………………………………..4

ГЛАВА 1. СОРТИРОВКА ОДНОМЕРНОГО МАССИВА………………….6

ГЛАВА 2. БИБЛИОТЕКА NUMPY. ОСНОВНЫЕ ФУНКЦИИ……………11

ГЛАВА 3. РЕШЕНИЕ СЛАУ МЕТОДОМ ГАУССА НА ЯЗЫКЕ ПРОГРАММИРОВАНИЯ PYTHON…………………………………………25

ЗАКЛЮЧЕНИЕ………………………………………………………………..28

БИБЛИОГРАФИЧЕСКИЙ СПИСОК………………………………………..29

ВВЕДЕНИЕ

Прежде всего разберемся, что собой представляет такой язык программирования, как Python. Python — скриптовый язык программирования. В отличие от С исполняется ровно в тот момент, когда мы его запускаем. До последнего момента программа — это всего лишь текст.

Это интерпретируемый язык — он не компилируется, то есть, как уже было сказано раннее, до запуска представляет из себя обычный текстовый файл. Программировать можно практически на всех платформах, язык хорошо спроектирован и логичен.

Разработка на нем в разы быстрее, потому как приходится писать меньше кода, чем на Java, С и других языках, — он отлично подходит для новичков и людей, которые не любят себя слишком сильно нагружать лишней информацией.

Библиотека языка программирования — это просто набор модулей и функций, которые облегчают некоторые специфические операции с использованием этого языка программирования. В своей работе я рассмотрела одну из таких библиотек.

NumPy — это расширение языка Python, добавляющее поддержку больших многомерных массивов и матриц, вместе с большой библиотекой высокоуровневых математических функций для операций с этими массивами.

В своей курсовой работе я рассмотрела следующие темы:

В первой части курсовой работы мной были проанализированы три алгоритма сортировки: bubble, insert, select. Разумеется, существует множество других алгоритмов, но для подробного изучения и выбора самого легкого и быстрого для использования были предоставлены эти три.

Во второй части работы я рассмотрела библиотеку NumPy. Любая библиотека содержит в себе так называемый набор шаблонов для решения той или иной задачи, что значительно упрощает и ускоряет работу, в нашем случае с линейной алгеброй. Для практического рассмотрения всех преимуществ данного “содействия” невозможно просто прочитать теоретический материал, именно поэтому мне и предстояла задача путем нарешивания постичь это.

В конце же курсовой работы, мне нужно было решить системы линейных уравнений методом Гаусса, с учетом индивидуальных заданий. Использование библиотеки Numpy значительно ускоряет и упрощает эту задачу, в чем мне, к счастью, пришлось убедиться.

ГЛАВА 1. СОРТИРОВКА ОДНОМЕРНОГО МАССИВА

Алгоритм сортировки — это алгоритм для упорядочивания элементов в списке. В случае, когда элемент списка имеет несколько полей, поле, служащее критерием порядка, называется ключом сортировки. На практике в качестве ключа часто выступает число, а в остальных полях хранятся какие-либо данные, никак не влияющие на работу алгоритма. Разберем три самых популярных и в это время самых простых алгоритма.

**Сортировка выбором (Selection Sort)**

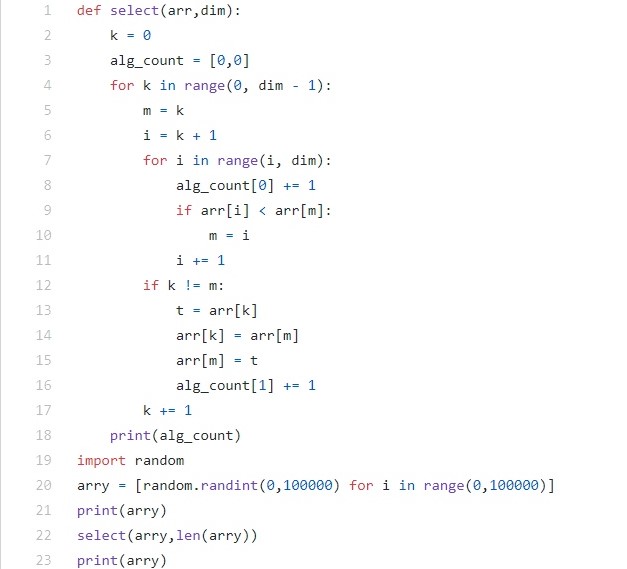
Этот алгоритм сегментирует список на две части: отсортированные и несортированные. Он постоянно удаляет наименьший элемент из несортированного сегмента списка и добавляет его в отсортированный сегмент.

**Объяснение:**

На практике нам не нужно создавать новый список для отсортированных элементов, мы будет обрабатывать крайнюю левую часть списка как отсортированный сегмент. Затем мы ищем во всем списке наименьший элемент и меняем его на первый элемент.

Теперь мы знаем, что первый элемент списка отсортирован, мы получаем наименьший элемент из оставшихся элементов и заменяем его вторым элементом. Это повторяется до тех пор, пока последний элемент списка не станет оставшимся элементом для изучения.

**Внешний вид:**

****

**Сортировка методом пузырька (bubble):**

Сортировка пузырьком - это метод сортировки массивов и списков путем последовательного сравнения и обмена соседних элементов, если предшествующий оказывается больше последующего.

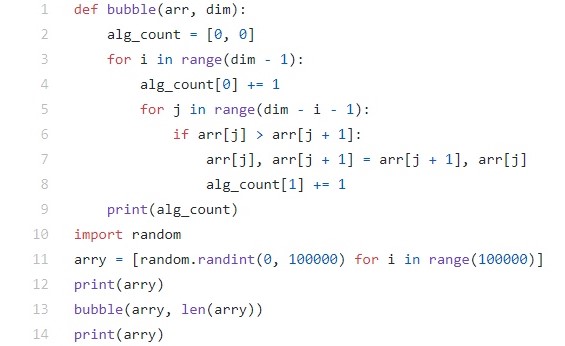
**Ход работы:**

В процессе выполнения данного алгоритма элементы с большими значениями оказываются в конце списка, а элементы с меньшими значениями постепенно перемещаются по направлению к началу списка. Образно говоря, тяжелые элементы падают на дно, а легкие медленно всплывают подобно пузырькам воздуха.

В сортировке методом пузырька количество итераций внешнего цикла определяется длинной списка минус единица, так как когда второй элемент становится на свое место, то первый уже однозначно минимальный и находится на своем месте.

Количество итераций внутреннего цикла зависит от номера итерации внешнего цикла, так как конец списка уже отсортирован, и выполнять проход по этим элементам смысла нет.

**Внешний вид:**

****

**Сортировка включениями (insert):**

Сортировка вставками (англ. Insertion sort) — алгоритм сортировки, в котором элементы входной последовательности просматриваются по одному, и каждый новый поступивший элемент размещается в подходящее место среди ранее упорядоченных элементов.

**Ход работы:**

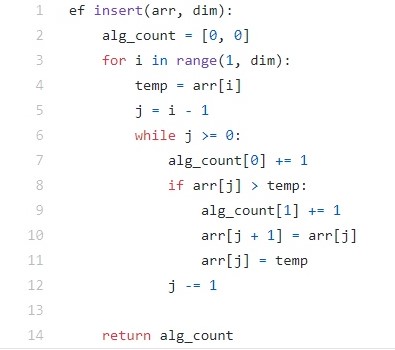
На вход алгоритма подаётся последовательность n чисел.

Входная последовательность на практике представляется в виде массива с n элементами. На выходе алгоритм должен вернуть перестановку исходной последовательности

В начальный момент отсортированная последовательность пуста. На каждом шаге алгоритма выбирается один из элементов входных данных и помещается на нужную позицию в уже отсортированной последовательности до тех пор, пока набор входных данных не будет исчерпан. В любой момент времени в отсортированной последовательности элементы удовлетворяют требованиям к выходным данным алгоритма.

Данный алгоритм можно ускорить при помощи использования бинарного поиска для нахождения места текущему элементу в отсортированной части. Проблема с долгим сдвигом массива вправо решается при помощи смены указателей.

**Внешний вид:**

****

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Массив | Показатель | 1000 элементов | | |
|  |  | Insert | Select | Bubble |
| Упорядоченный | Кол-во сравнений(С) | 499500 | 499500 | 999 |
| Кол-во перемещений(М) | 0 | 0 | 0 |
| Обратно упорядоченный | Кол-во сравнений(С) | 499500 | 499500 | 499500 |
| Кол-во перемещений(М) | 499500 | 250000 | 499500 |
| Случайный | Mean(C) | 499500 | 499500 | 496365,4 |
| Mean(M) | 249939 | 5265,02 | 249789,1 |

**Результаты алгоритмов:**

Сортировка вставками наиболее эффективна, когда массив уже частично отсортирован и когда элементов массива не много. Если элементов меньше 10, то данный алгоритм является лучшим. Пузырьковая сортировка является неэффективной, поскольку она должна переставлять элементы до того, как станет известна их окончательная позиция. Эти “пустые” операции обмена затратны. Для случайных значений лучшим выбором сортировки является Select, поскольку в выборе заранее известно, в какое место поставим элемент.

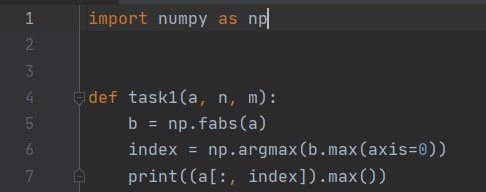
ГЛАВА 2. БИБЛИОТЕКА NUMPY. ОСНОВНЫЕ ФУНКЦИИ

**Python** становится все популярнее и популярнее. Много людей начинают изучать этот язык и встречают на своё пути библиотеку **NumPy**. Она насчитывает в себе множество различных функций и подмодулей, потому запомнить всё просто напросто невозможно. Благодаря таким библиотекам становится возможным более легкое выполнение определенных задач без необходимости написания излишнего кода.

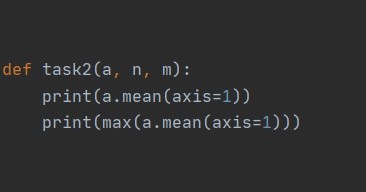
В Python’е библиотека Numpy незаменима для работы с числовыми массивами, векторами и матрицами, а также позволяет строить графики и гистограммы.

Далее прилагаю решение задач, каждая из которых была решена с помощью функций библиотеки Numpy.

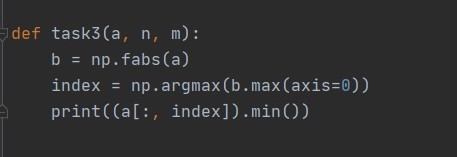
1. Создать прямоугольную матрицу A, имеющую N строк и M столбцов со случайными элементами. Найти наибольший элемент столбца матрицы A, для которого сумма абсолютных значений элементов максимальна.

****

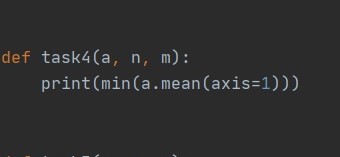
1. Создать прямоугольную матрицу A, имеющую N строк и M столбцов со случайными элементами. Найти наибольшее значение среди средних значений для каждой строки матрицы.

****

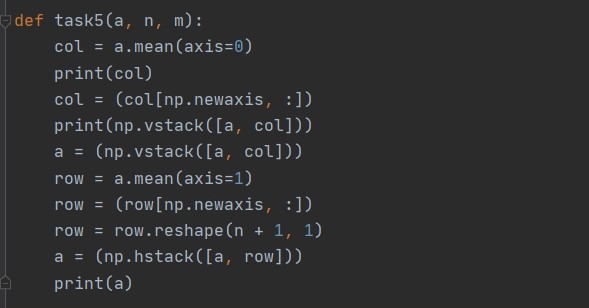
1. Создать прямоугольную матрицу A, имеющую N строк и M столбцов со случайными элементами. Найти наименьший элемент столбца матрицы A, для которого сумма абсолютных значений элементов максимальна.

****

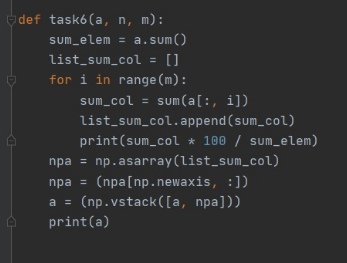
1. Создать прямоугольную матрицу A, имеющую N строк и M столбцов со случайными элементами. Найти наименьшее значение среди средних значений для каждой строки матрицы.



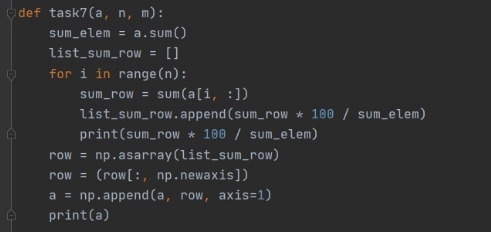
1. Создать прямоугольную матрицу A, имеющую N строк и M столбцов со случайными элементами. Определить средние значения по всем строкам и столбцам матрицы. Результат оформить в виде матрицы из N + 1 строк и M + 1 столбцов.



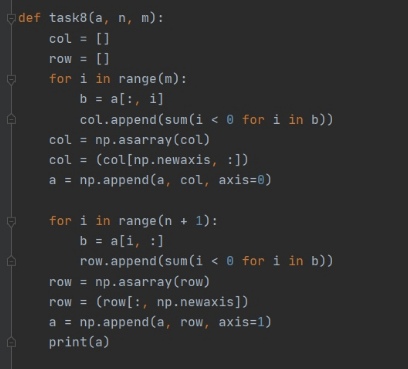
1. Создать прямоугольную матрицу A, имеющую N строк и M столбцов со случайными элементами. Найти сумму элементов всей матрицы. Определить, какую долю в этой сумме составляет сумма элементов каждого столбца. Результат оформить в виде матрицы из N + 1 строк и M столбцов.



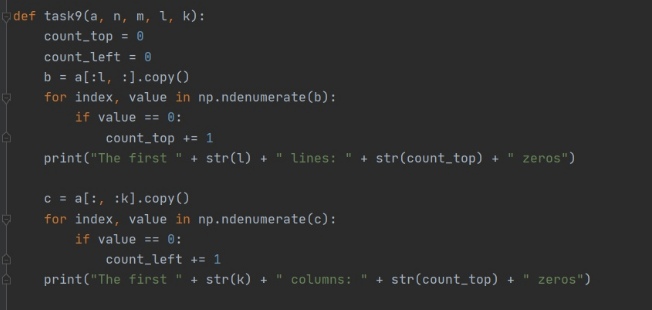
1. Создать прямоугольную матрицу A, имеющую N строк и M столбцов со случайными элементами. Найти сумму элементов всей матрицы. Определить, какую долю в этой сумме составляет сумма элементов каждой строки. Результат оформить в виде матрицы из N строк и M+1 столбцов.



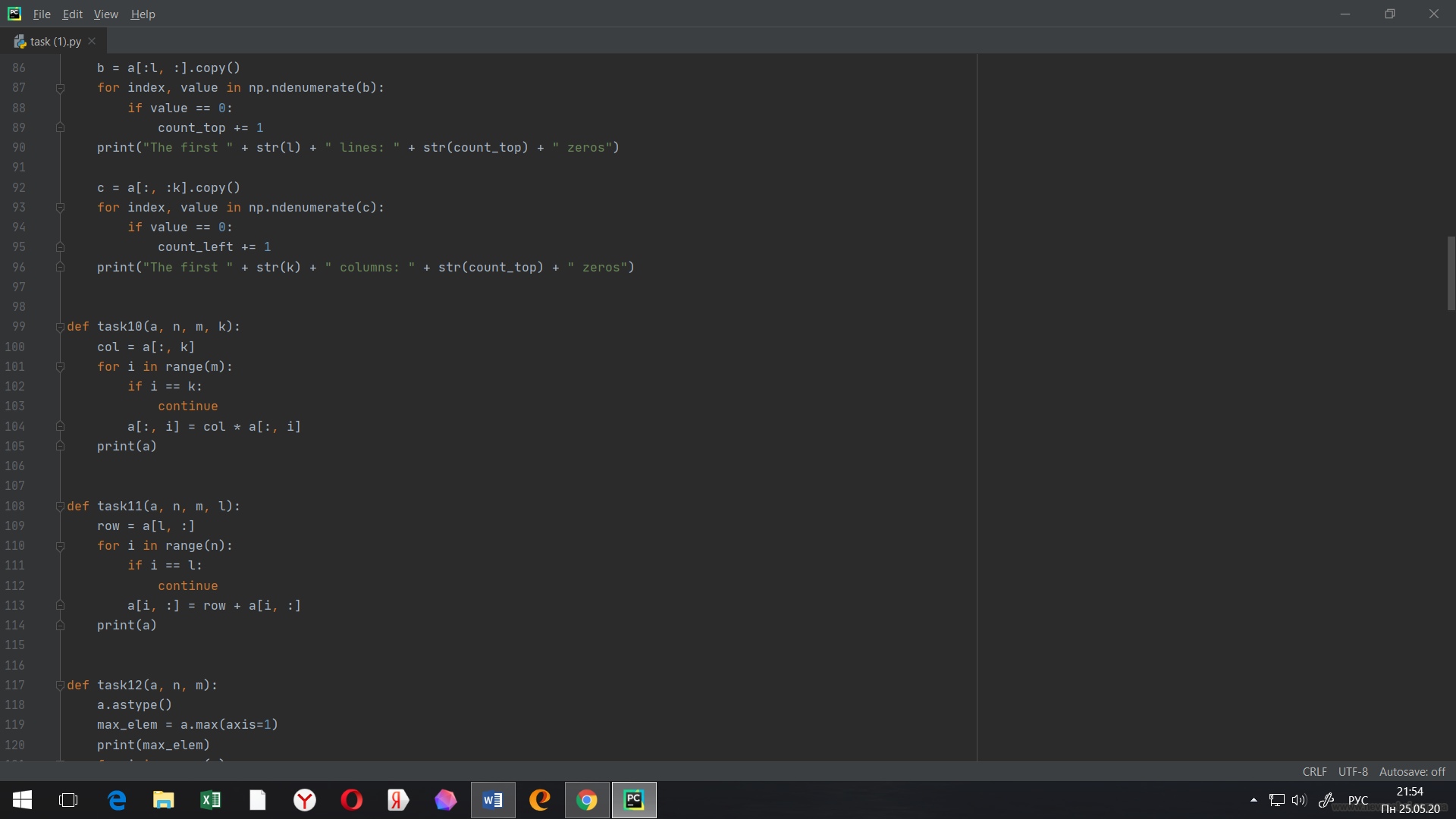
1. Создать прямоугольную матрицу A, имеющую N строк и M столбцов со случайными элементами. Определить, сколько отрицательных элементов содержится в каждом столбце и в каждой строке матрицы. Результат оформить в виде матрицы из N + 1 строк и M + 1 столбцов.



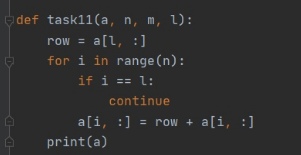
1. Создать прямоугольную матрицу A, имеющую N строк и M столбцов со случайными элементами. Определить, сколько нулевых элементов содержится в верхних L строках матрицы и в левых К столбцах матрицы.



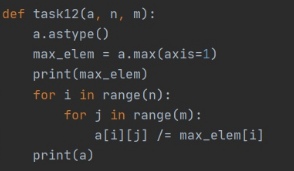
1. Создать прямоугольную матрицу A, имеющую N строк и M столбцов со случайными элементами. Перемножить элементы каждого столбца матрицы с соответствующими элементами K-го столбца.



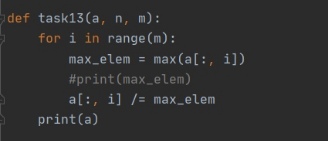
1. Создать прямоугольную матрицу A, имеющую N строк и M столбцов со случайными элементами. Просуммировать элементы каждой строки матрицы с соответствующими элементами L-й строки.



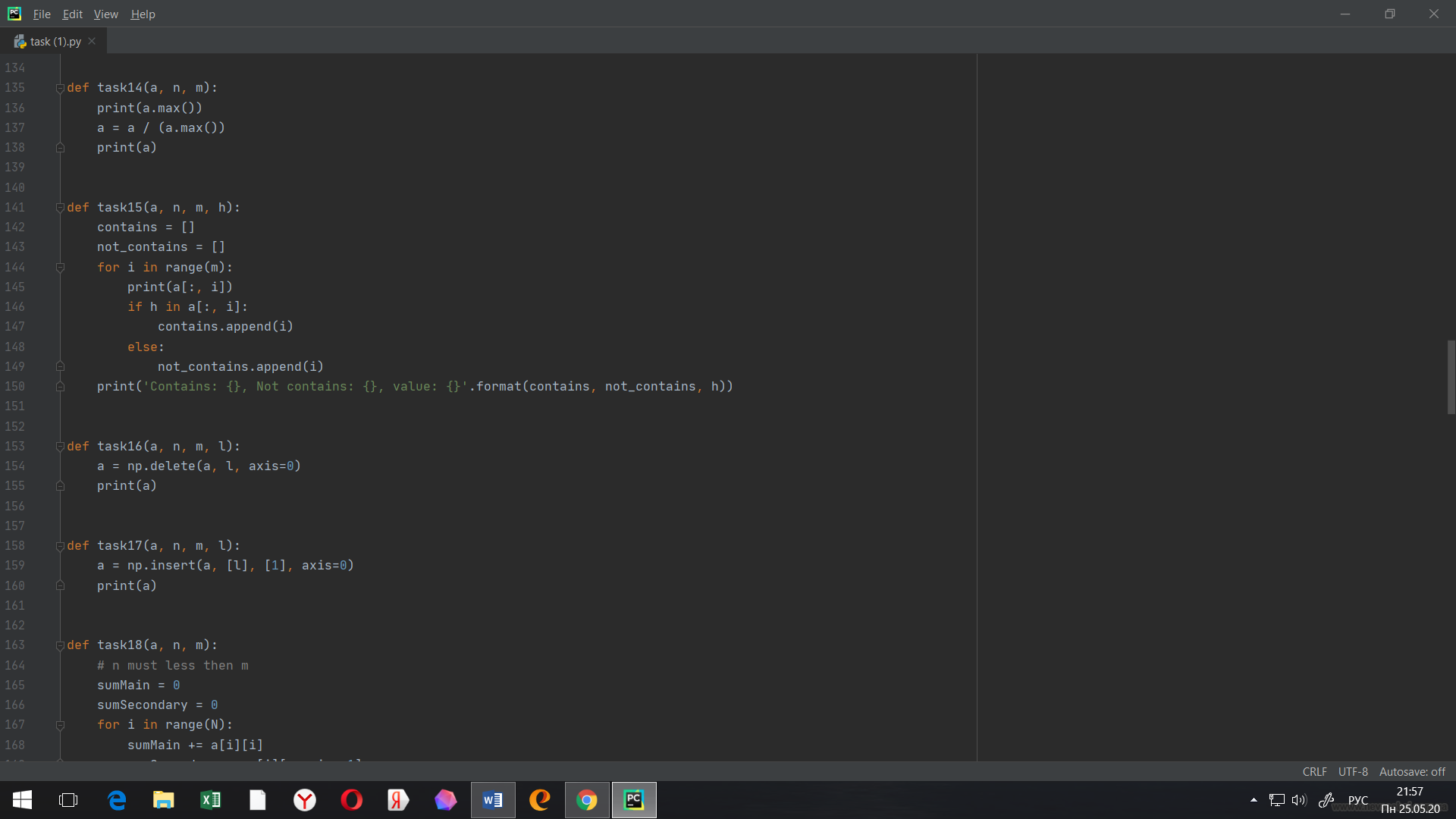
1. Создать прямоугольную матрицу A, имеющую N строк и M столбцов со случайными элементами. Разделить элементы каждой строки на элемент этой строки с наибольшим значением.



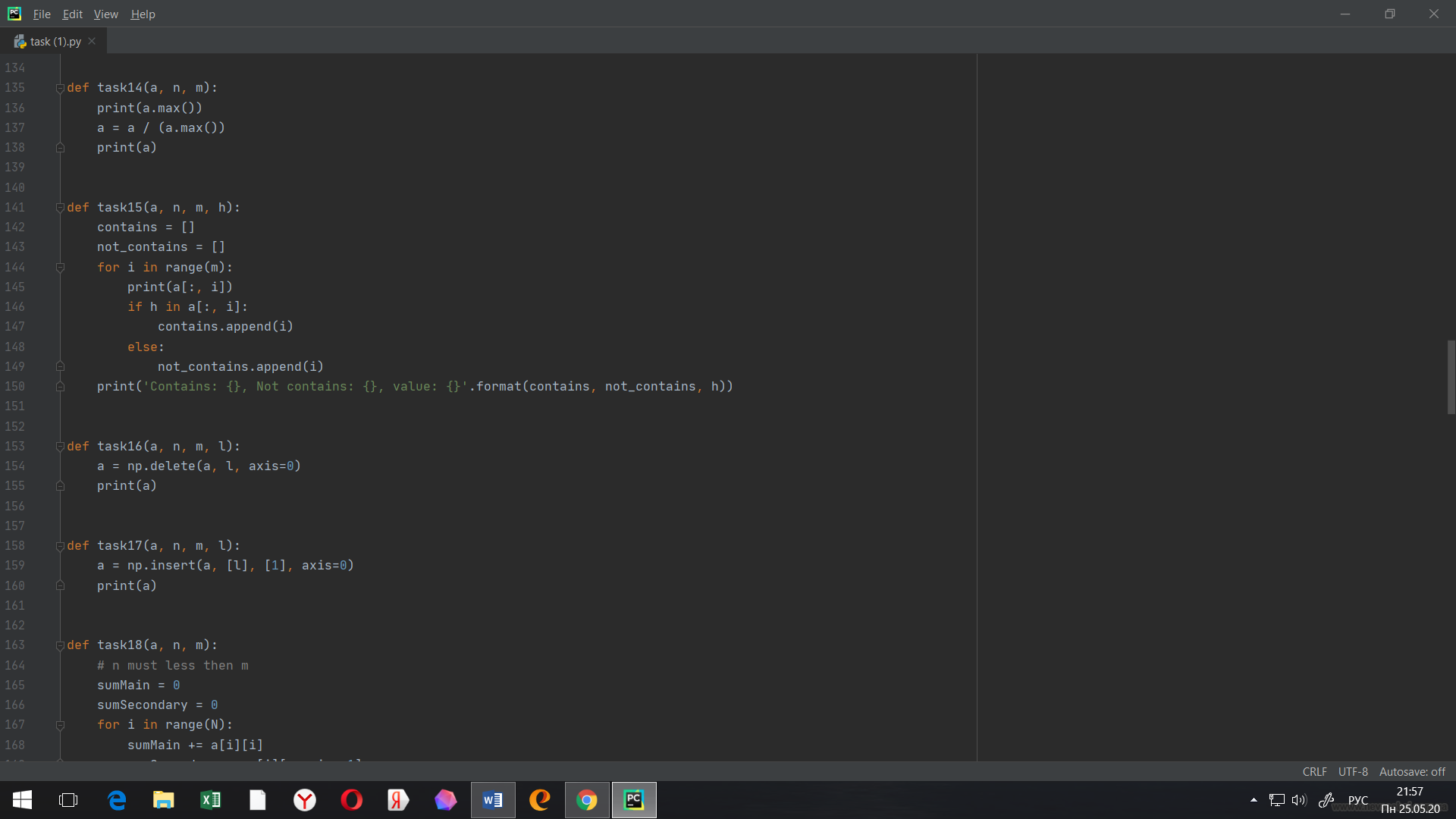
1. Создать прямоугольную матрицу A, имеющую N строк и M столбцов со случайными элементами. Разделить элементы каждого столбца матрицы на элемент этого столбца с наибольшим значением.



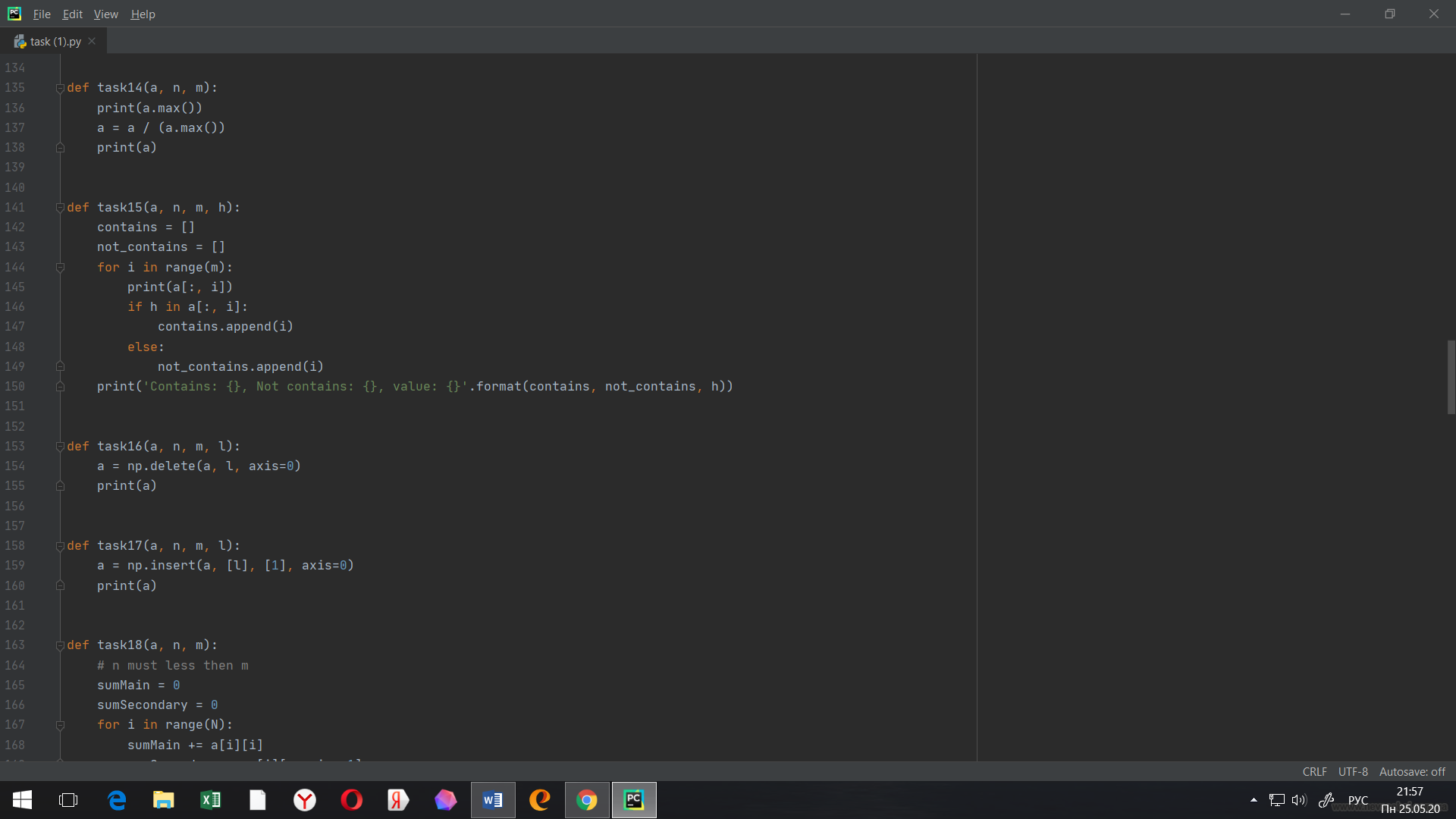
1. Создать прямоугольную матрицу A, имеющую N строк и M столбцов со случайными элементами. Разделить элементы матрицы на элемент матрицы с наибольшим значением.



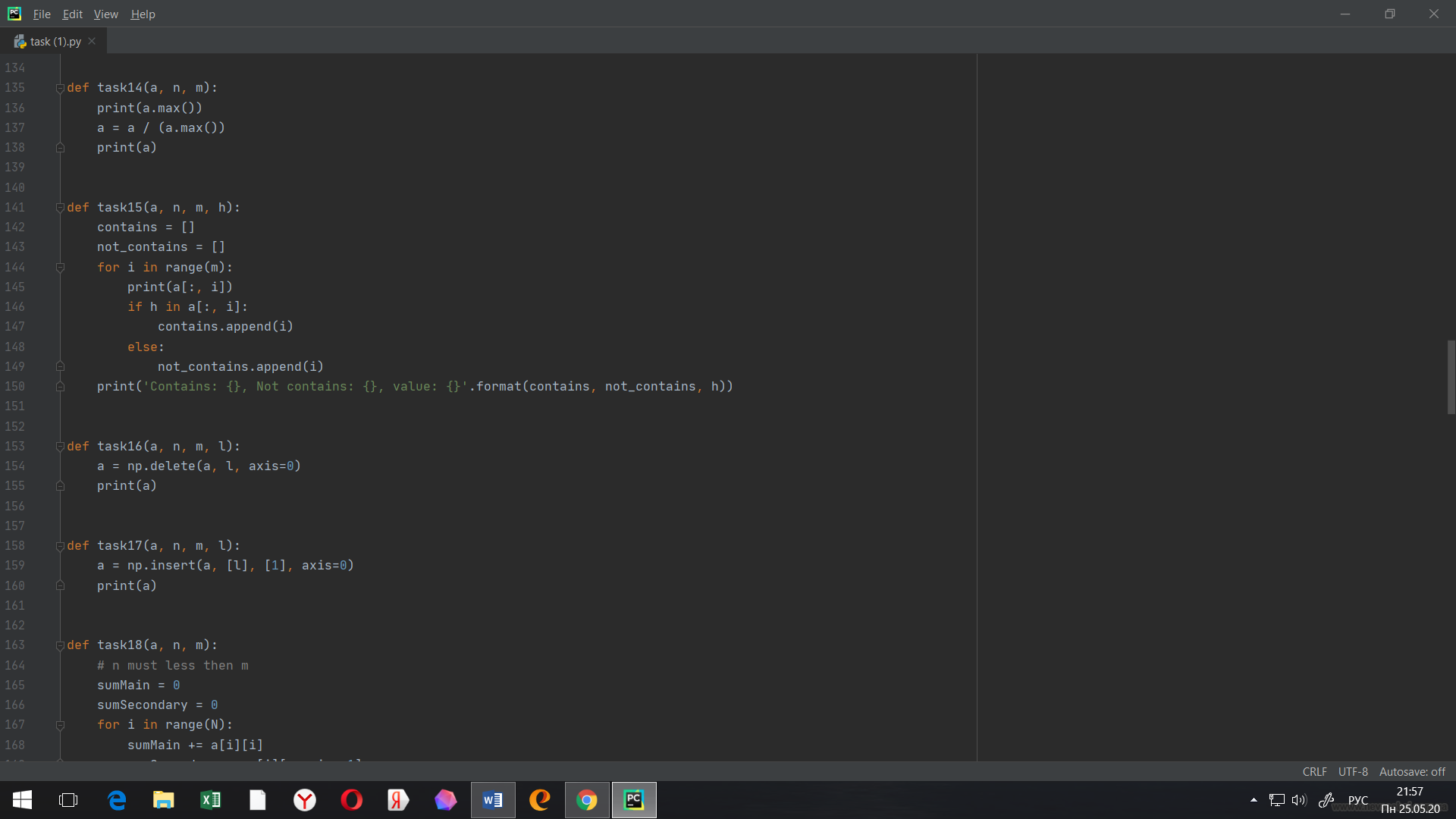
1. Создать прямоугольную матрицу A, имеющую N строк и M столбцов со случайными элементами. Все элементы имеют целый тип. Дано целое число H. Определить, какие столбцы имеют хотя бы одно такое число, а какие не имеют.



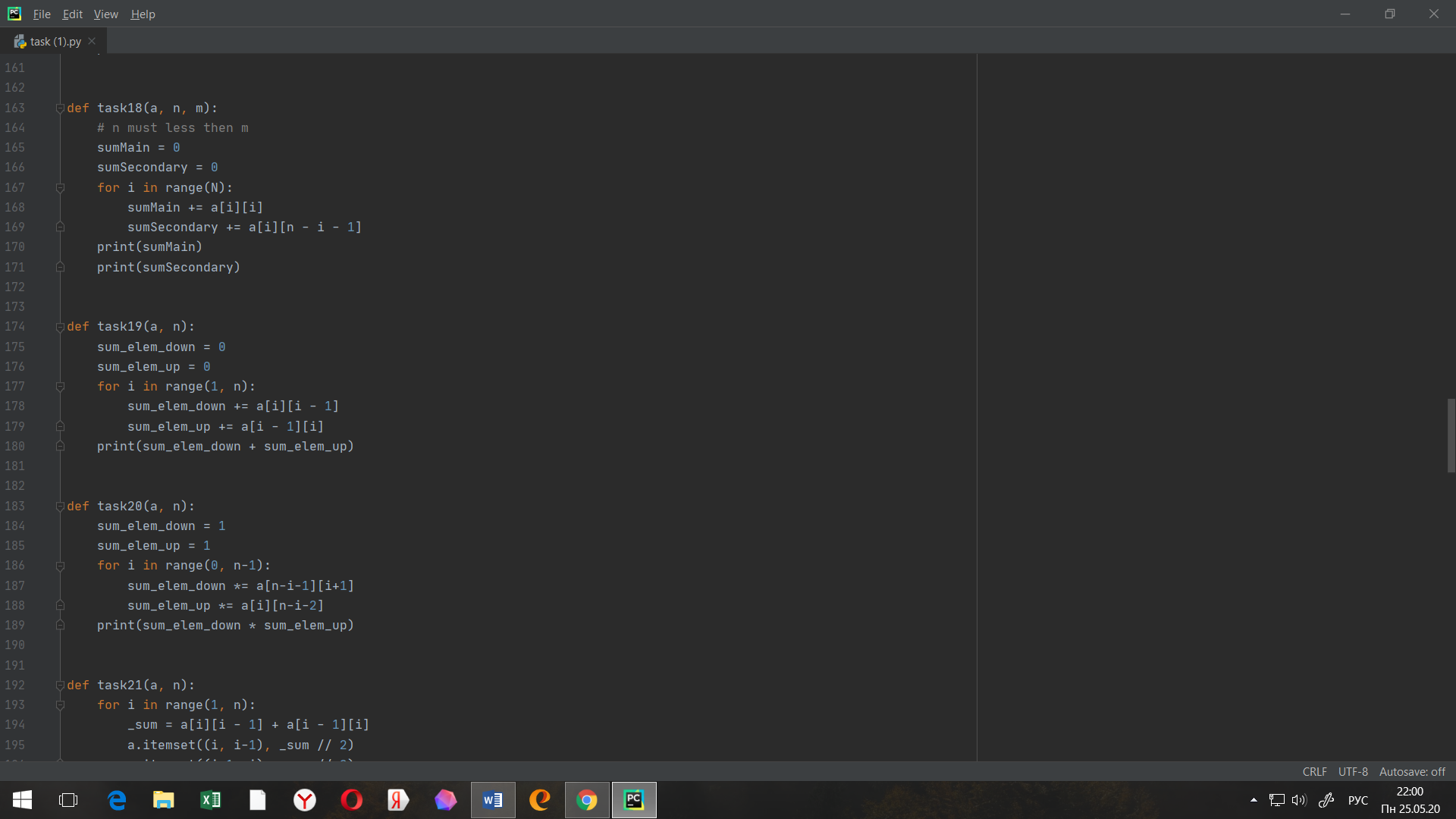
1. Создать прямоугольную матрицу A, имеющую N строк и M столбцов со случайными элементами. Исключить из матрицы строку с номером L. Сомкнуть строки матрицы.



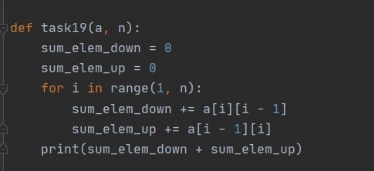
1. Создать прямоугольную матрицу A, имеющую N строк и M столбцов со случайными элементами. Добавить к матрице строку и вставить ее под номером L.



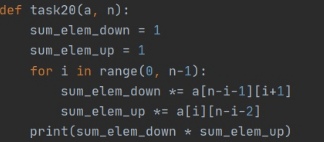
1. Создать прямоугольную матрицу A, имеющую N строк и M столбцов со случайными элементами. Найти сумму элементов, стоящих на главной диагонали, и сумму элементов, стоящих на побочной диагонали (элементы главной диагонали имеют индексы от [0,0] до [N,N], а элементы побочной диагонали — от [N,0] до [0,N]).



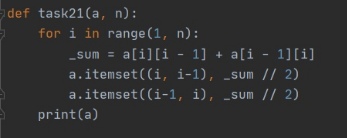
1. Создать квадратную матрицу A, имеющую N строк и N столбцов со случайными элементами. Определить сумму элементов, расположенных параллельно главной диагонали (ближайшие к главной). Элементы главной диагонали имеют индексы от [0,0] до [N,N].



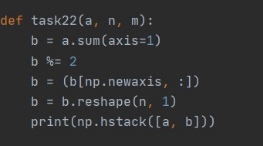
1. Создать квадратную матрицу A, имеющую N строк и N столбцов со случайными элементами. Определить произведение элементов, расположенных параллельно побочной диагонали (ближайшие к побочной). Элементы побочной диагонали имеют индексы от [N,0] до [0,N].



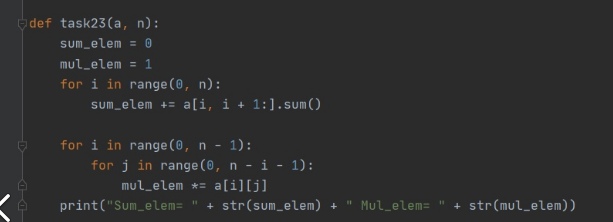
1. Создать квадратную матрицу A, имеющую N строк и N столбцов со случайными элементами. Каждой паре элементов, симметричных относительно главной диагонали (ближайшие к главной), присвоить значения, равные полусумме этих симметричных значений (элементы главной диагонали имеют индексы от [0,0] до [N,N]).



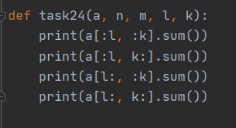
1. Создать прямоугольную матрицу A, имеющую N строк и M столбцов со случайными элементами. Исходная матрица состоит из нулей и единиц. Добавить к матрице еще один столбец, каждый элемент которого делает количество единиц в каждой строке чётным.



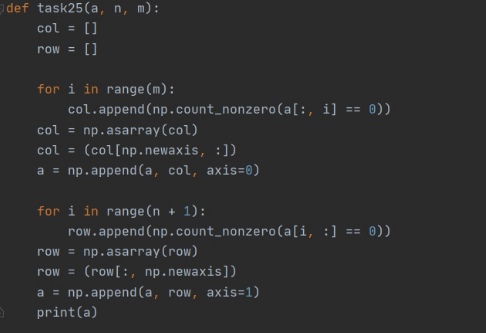
1. Создать квадратную матрицу A, имеющую N строк и N столбцов со случайными элементами. Найти сумму элементов, расположенных выше главной диагонали, и произведение элементов, расположенных выше побочной диагонали (элементы главной диагонали имеют индексы от [0,0] до [N,N], а элементы побочной диагонали — от [N,0] до [0,N]).



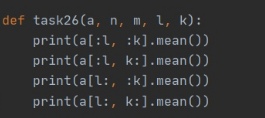
1. Создать прямоугольную матрицу A, имеющую N строк и M столбцов со случайными элементами. Дан номер строки L и номер столбца K, при помощи которых исходная матрица разбивается на четыре части. Найти сумму элементов каждой части.



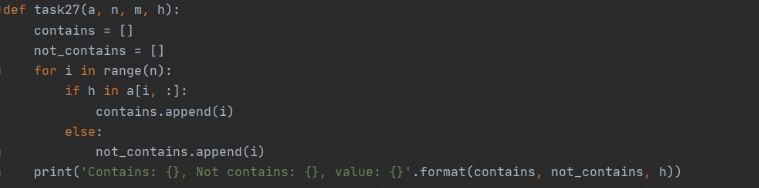
1. Создать прямоугольную матрицу A, имеющую N строк и M столбцов со случайными элементами. Определить, сколько нулевых элементов содержится в каждом столбце и в каждой строке матрицы. Результат оформить в виде матрицы из N + 1 строк и M + 1 столбцов.



1. Создать прямоугольную матрицу A, имеющую N строк и M столбцов со случайными элементами. Дан номер строки L и номер столбца K, при помощи которых исходная матрица разбивается на четыре части. Найти среднее арифметическое элементов каждой части.



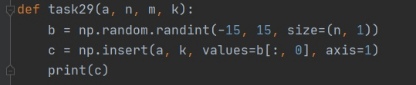
1. Создать прямоугольную матрицу A, имеющую N строк и M столбцов со случайными элементами. Все элементы имеют целый тип. Дано целое число H. Определить, какие строки имеют хотя бы одно такое число, а какие не имеют.



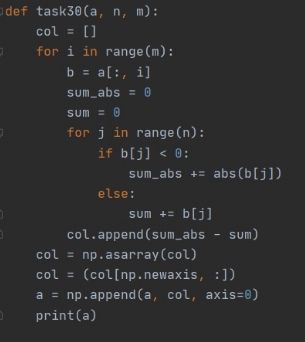
1. Создать прямоугольную матрицу A, имеющую N строк и M столбцов со случайными элементами. Исключить из матрицы столбец с номером K. Сомкнуть столбцы матрицы.

https://sun2.is74.userapi.com/gMLayhSDqI81WUDZKp0aU3VUYnvBmedCfQeEeg/jFVPI-VGzEw.jpg

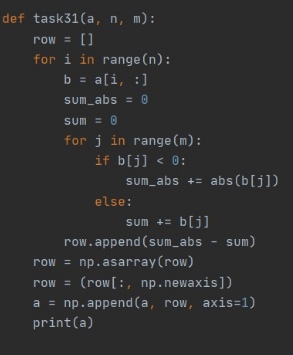
1. Создать прямоугольную матрицу A, имеющую N строк и M столбцов со случайными элементами. Добавить к матрице столбец чисел и вставить его под номером K.



1. Создать прямоугольную матрицу A, имеющую N строк и M столбцов со случайными элементами. Добавить к элементам каждого столбца такой новый элемент, чтобы сумма положительных элементов стала бы равна модулю суммы отрицательных элементов. Результат оформить в виде матрицы из N + 1 строк и M столбцов.



1. Создать прямоугольную матрицу A, имеющую N строк и M столбцов со случайными элементами. Добавить к элементам каждой строки такой новый элемент, чтобы сумма положительных элементов стала бы равна модулю суммы отрицательных элементов. Результат оформить в виде матрицы из N строк и M + 1 столбцов.



Подытожим данную часть тем, что как вы могли заметить скриншоты программ предоставлены из одного файла и для запуска необходимой задачи мне потребовался следующий код:

|  |
| --- |
|  |
| if \_\_name\_\_ =='\_\_main\_\_': |
|  | N = 5 |
|  | M = 6 |
|  |  |
|  | A = np.random.randint(-15, 15, size=(N, M)) |
|  | print("task1") |
|  | task1(A, N, M) |
|  | print("task2") |
|  | task2(A, N, M) |
|  | print("task3") |
|  | task3(A, N, M) |
|  | print("task4") |
|  | task4(A, N, M) |
|  | print("task5") |
|  | task5(A, N, M) |
|  | print("task8") |
|  | task8(A, N, M) |
|  | print("task9") |
|  | task9(A, N, M, 2, 3) |
|  | print("task10") |
|  | task10(A, N, M, 2) |
|  | print("task11") |
|  | task11(A, N, M, 2) |
|  |  |
|  | print("task15") |
|  | task15(A, N, M, 4) |
|  | print("task16") |
|  | task16(A, N, M, 1) |
|  | print("task17") |
|  | task17(A, N, M, 2) |
|  | print("task18") |
|  | task18(A, N, M) |
|  | print("task24") |
|  | task24(A, N, M, 2, 3) |
|  | print("task25") |
|  | task25(A, N, M) |
|  | print("task26") |
|  | task26(A, N, M, 2, 3) |
|  | print("task27") |
|  | task27(A, N, M, 3) |
|  | print("task28") |
|  | task28(A, N, M, 1) |
|  | print("task29") |
|  | task29(A, N, M, 2) |
|  | print("task30") |
|  | task30(A, N, M) |
|  | print("task31") |
|  | task31(A, N, M) |
|  |  |
|  | A = np.random.randint(1, 25, size=(N, M)) |
|  | print("task6") |
|  | task6(A, N, M) |
|  | print("task7") |
|  | task7(A, N, M) |
|  |  |
|  | B = np.random.random(size=(N, M)) |
|  | B \*= 100 |
|  | B = np.around(B, decimals=2) |
|  | print("task12") |
|  | task12(B, N, M) |
|  | print("task13") |
|  | task13(B, N, M) |
|  | print("task14") |
|  | task14(B, N, M) |
|  |  |
|  | A = np.random.randint(-15, 15, size=(N, N)) |
|  | print("task19") |
|  | task19(A, N) |
|  | print("task20") |
|  | task20(A, N) |
|  | print("task21") |
|  | task21(A, N) |
|  | print("task23") |
|  | task23(A, N) |
|  |  |
|  | A = np.random.randint(0, 2, size=(N, M)) |
|  | print("task22") |
|  | task22(A, N, M) |
|  |  |
|  |  |
|  |  |

ГЛАВА 3. РЕШЕНИЕ СЛАУ МЕТОДОМ ГАУССА НА ЯЗЫКЕ ПРОГРАММИРОВАНИЯ PYTHON

Метод Гаусса — классический метод решения системы линейных алгебраических уравнений (СЛАУ).

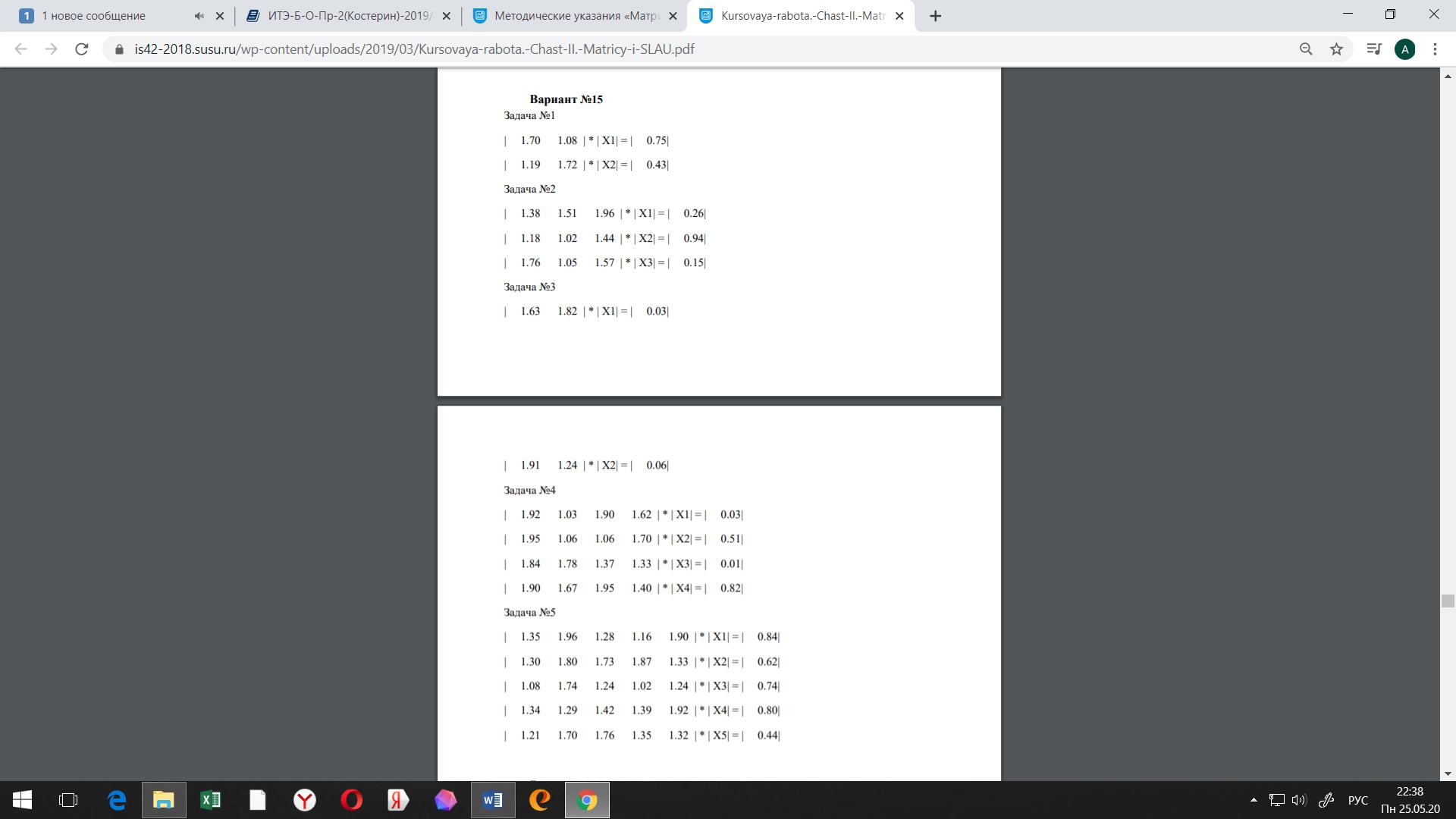
Метод Гаусса, называемый также методом последовательного исключения неизвестных, состоит в следующем. При помощи элементарных преобразований систему линейных уравнений приводят к такому виду, чтобы её матрица из коэффициентов оказалась трапециевидной (то же самое, что треугольной или ступенчатой) или близкой к трапециевидной (прямой ход метода Гаусса, далее - просто прямой ход).

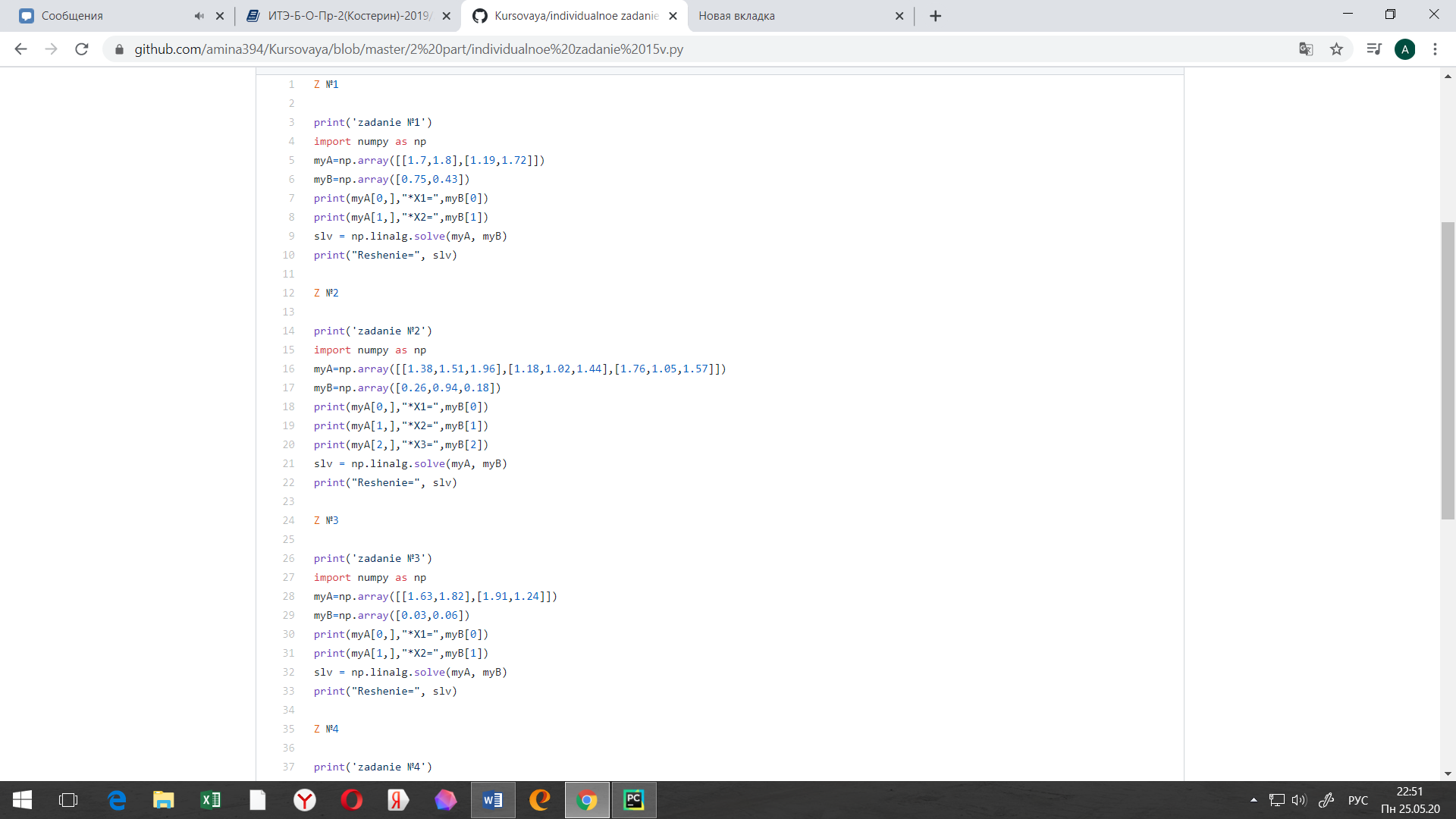
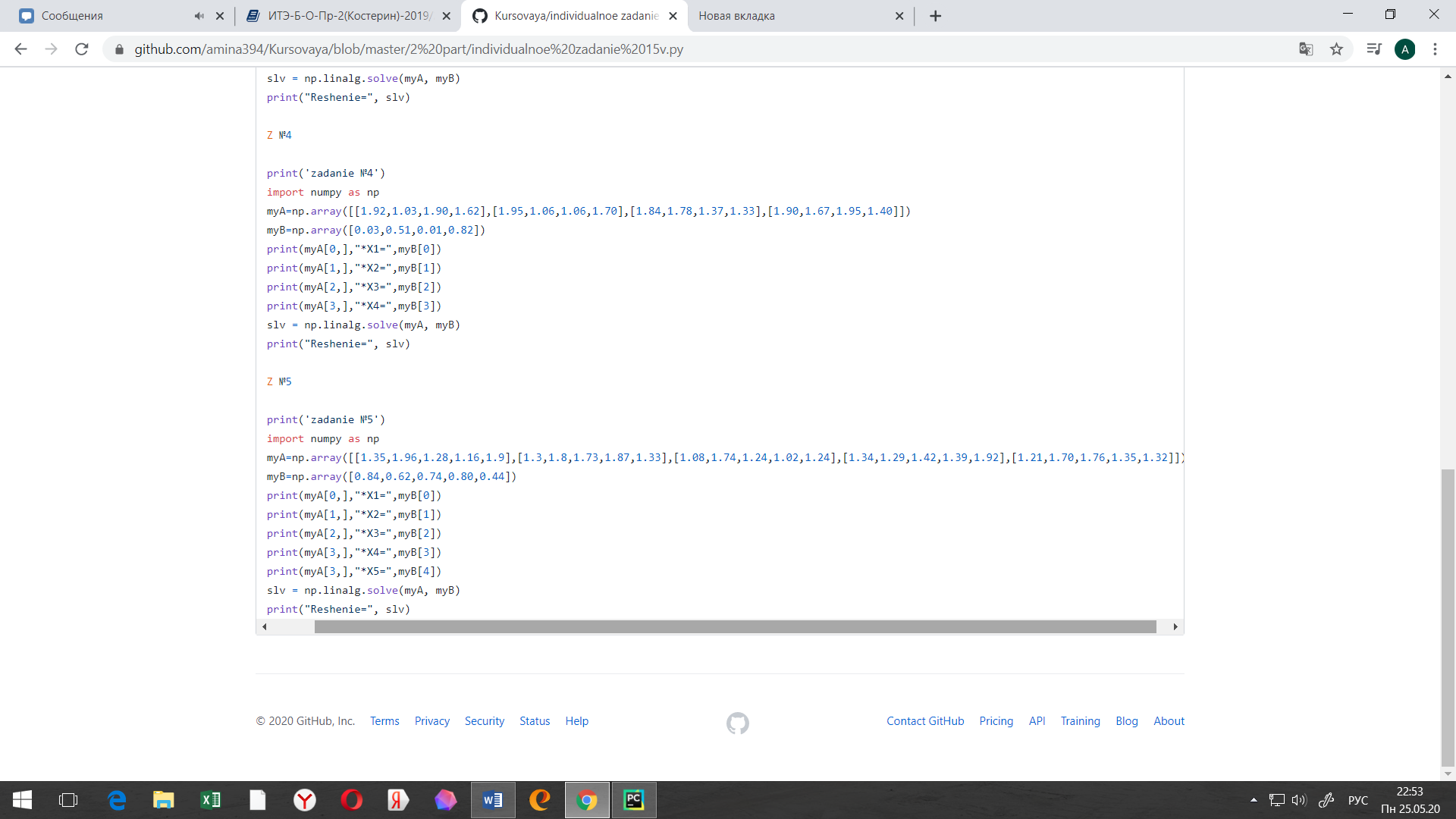
Метод Гаусса идеально подходит для решения систем содержащих больше трех линейных уравнений, для решения систем уравнений, которые не являются квадратными (чего не скажешь про метод Крамера иматричный метод). То есть метод Гаусса - наиболее универсальный метод для нахождения решения любой системы линейных уравнений, он работает в случае, когда система имеет бесконечно много решений или несовместна.

Теперь реализуем это при помощи Python и Numpy.

Нам дано 5 систем в индивидуальном варианте.

Вариант 15.





ЗАКЛЮЧЕНИЕ

В заключении стоит сказать, что благодаря языку программирования Python можно быстро, удобно и качественно решать различные задачи, существует множество библиотек, которые дают огромные возможности в решение и написании различных задач и программ.

Нельзя не отметить, что все большую роль начинают играть интерпретируемые языки, поскольку возрастающая мощь персональных компьютеров начинает обеспечивать достаточную скорость выполнения интерпретируемых программ. А единственным существенным преимуществом компилируемых языков программирования является создаваемый ими высокоскоростной код.

В конце хочется отметить, что Python – это язык программирования, востребованный сегодня и с большим потенциалом в будущее. Сегодня рынок труда нуждается в квалифицированных специалистах со знаниями Python.

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

* [https://ru.wikipedia.org](https://ru.wikipedia.org/wiki/%D0%97%D0%B0%D0%B3%D0%BB%D0%B0%D0%B2%D0%BD%D0%B0%D1%8F_%D1%81%D1%82%D1%80%D0%B0%D0%BD%D0%B8%D1%86%D0%B0)
* <https://is20-2019.susu.ru/blog/author/waksoft/>
* <https://pyprog.pro/reference_manual.html>
* <https://habr.com/ru/post/352678/>