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 – 380305.2022.13. ПЗ КР

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

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

«\_\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_2019г. «\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_\_2019г.

Нормоконтролёр, \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Автор, студент группы ЭУ-142

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_/\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_/ Еловсков Л.В./\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_/

«\_\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_2019г.

Работа защищена с оценкой

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

«\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_2019г.

Челябинск 2019

# АННОТАЦИЯ

Еловсков Л.В.

Программирование на языке Python (курс молодого бойца)

Челябинск: ЮУрГУ, ЭУ-142, 2019

В своей работе было рассказано о Python, о его библиотеках, и автоматизация некоторых задач при помощи Python.

Оглавление

[ВВЕДЕНИЕ 3](#_Toc10695676)

[МЕТОДЫ СОРТИРОВКИ 5](#_Toc10695677)

[ПАКЕТ NUMPY 13](#_Toc10695678)

[РЕШЕНИЕ СЛАУ МЕТОДОМ ГАУССА 44](#_Toc10695679)

[ЗАКЛЮЧЕНИЕ 47](#_Toc10695680)

[СПИСОК ЛИТЕРАТУРЫ 48](#_Toc10695681)

# ВВЕДЕНИЕ

Python - это интерпретируемый, объектно-ориентированный, тьюринг-полный язык программирования высокого уровня, предназначенный для решения самого широкого круга задач. С его помощью можно обрабатывать числовую и текстовую информацию, создавать изображения, работать с базами данных, разрабатывать WеЬ-сайты и приложения с графическим интерфейсом. Python - язык кроссплатформенный, он позволяет создавать программы, которые будут работать во всех операционных системах. В этой книге мы рассмотрим базовые возможности Python версии 3.4 применительно к операционной системе Windows.

Согласно официальной версии, название языка произошло вовсе не от змеи. Создатель языка Гвидо ван Россум (Guido van Rossum) назвал свое творение в честь британского комедийного телешоу ВВС «Летающий цирк Монти Пайтона» (Monty Python's Flying Circus). Поэтому правильное произношение названия этого замечательного языка - Пайтон.

Программа на языке Python представляет собой обычный текстовый файл с расширением ру (консольная программа) или pyw (программа с графическим интерфейсом). Все инструкции из этого файла выполняются интерпретатором построчно. Для ускорения работы при первом импорте модуля создается промежуточный байт-код, который сохраняется в одноименном файле с расширением рус. При последующих запусках, если модуль не был изменен, исполняется именно байт-код. Для выполнения низкоуровневых операций и задач, требующих высокой скорости работы, можно написать модуль на языке С или С++, скомпилировать его, а затем подключить к основной программе.

Python относится к категории языков объектно-ориентированных. Это означает, что практически все данные в нем являются объектами, даже значения, относящиеся к элементарным типам, наподобие чисел и строк, а также сами типы данных. В переменной всегда сохраняется только ссылка на объект, а не сам объект. Например, можно создать функцию, сохранить ссылку на нее в переменной, а затем вызвать функцию через эту переменную. Такое обстоятельство делает язык Python идеальным инструментом для создания программ, использующих функции обратного вызова, - например, при разработке. графического интерфейса. Тот факт, что язык является объектно-ориентированным, отнюдь не означает, что и объектно-ориентированный стиль программирования (ООП) является при его использовании обязательным. На языке Python можно писать программы как в стиле ООП, так и в процедурном стиле, - как того требует конкретная ситуация или как предпочитает программист.

# МЕТОДЫ СОРТИРОВКИ

Существует огромное количество методов сортировки, но в данной работе разберём всего три.

Сортировка включением

Одним из наиболее простых и естественных методов внутренней сортировки является сортировка простыми включениями. Идея алгоритма очень проста. Пусть имеется массив ключей Arr0, Arr1, ..., ArrN‑1. Для каждого элемента массива, начиная со второго, производится сравнение с элементами с меньшим индексом. Элемент Arri последовательно сравнивается с элементами Arrj, где jЄ[i‑1;0], т.е. изменяется от i‑1 до 0. До тех пор, пока для очередного элемента Arrj выполняется соотношение Arrj > Arri, Arri и Arrj меняются местами. Если удается встретить такой элемент Arrj, что Arrj ≤ Arri, или если достигнута нижняя граница массива, производится переход к обработке элемента Arri+1. Так продолжается до тех пор, пока не будет достигнута верхняя граница массива.

Легко видеть, что в лучшем случае, когда массив уже упорядочен для выполнения алгоритма с массивом из N элементов потребуется N‑1 сравнение и 0 пересылок. В худшем случае, когда массив упорядочен в обратном порядке потребуется N(N‑1)/2 сравнений и столько же пересылок. Таким образом, можно оценивать сложность метода простых включений как O(N2).

Можно сократить число сравнений, применяемых в методе простых включений, если воспользоваться тем, что при обработке элемента Arri массива элементы Arr0, Arr1, ..., Arri‑1 уже упорядочены, и воспользоваться для поиска элемента, с которым должна быть произведена перестановка, методом двоичного деления. В этом случае оценка числа требуемых сравнений становится O(N\*Log(N)). Заметим, что поскольку при выполнении перестановки требуется сдвижка на один элемент нескольких элементов, то оценка числа пересылок остается O(N2). Алгоритм сортировки включением, оформленный в виде функции приведен ниже.

Обменная сортировка

Простая обменная сортировка, называемая «методом пузырька», для массива Arr0, Arr2, ..., ArrN‑1 работает следующим образом. Начиная с конца массива сравниваются два соседних элемента ArrN‑1 и ArrN‑2. Если выполняется условие ArrN‑2 > ArrN‑1, то они меняются местами. Процесс продолжается для ArrN‑2 и ArrN‑3 и т.д., пока не будет произведено сравнение Arr1 и Arr0. Понятно, что после этого на месте Arr0 окажется элемент с наименьшим значением. На втором шаге процесс повторяется, но последними сравниваются Arr2 и Arr1. И так далее. На последнем шаге будут сравниваться только текущие значения ArrN‑1 и ArrN‑2. Понятна аналогия с пузырьком, поскольку наименьшие элементы, самые «легкие», постепенно «всплывают» к верхней границе массива.

Для метода обменной сортировки требуется число сравнений N(N‑1)/2, минимальное число пересылок 0, а среднее и максимальное число пересылок − O(N2).

Метод пузырька допускает три простых усовершенствования. Во-первых, если на некотором шаге не было произведено ни одного обмена, то выполнение алгоритма можно прекращать. Во-вторых, можно запоминать наименьшее значение индекса массива, для которого на текущем шаге выполнялись перестановки. Очевидно, что верхняя часть массива до элемента с этим индексом уже отсортирована, и на следующем шаге можно прекращать сравнения значений соседних элементов при достижении такого значения индекса. В-третьих, метод пузырька работает неравноправно для «легких» и «тяжелых» значений. Легкое значение попадает на нужное место за один шаг, а тяжелое на каждом шаге опускается по направлению к нужному месту на одну позицию.

Сортировка выбором

При сортировке массива Arr0, Arr2, ..., ArrN‑1 методом простого выбора среди всех элементов находится элемент с наименьшим значением Arri, и Arr0 и Arri обмениваются значениями. Затем этот процесс повторяется для получаемого подмассива Arr1, Arr2, ..., ArrN‑1, ... Arrj, Arrj+1, ..., ArrN‑1 до тех пор, пока мы не дойдем до подмассива ArrN‑1, содержащего к этому моменту наибольшее значение.

Для метода сортировки простым выбором оценка требуемого числа сравнений – N(N‑1)/2. Порядок требуемого числа пересылок, которые требуются для выбора минимального элемента, в худшем случае составляет O(N2). Однако порядок среднего числа пересылок есть O(N\*Lg(N)), что в ряде случаев делает этот метод предпочтительным.

Практика: Дано задние. Сравнить данные методы сортировки и выявить из них самый эффективный алгоритм. Для был написан данный код:

|  |
| --- |
| def **select**(arr, dim):  k = 0  alg\_count = [0, 0]  for k in **range**(0, dim - 1):  m = k  i = k + 1  for i in **range**(i, dim):  alg\_count[0] += 1  if arr[i] < arr[m]:  m = i  if k != m:  t = arr[k]  arr[k] = arr[m]  arr[m] = t  alg\_count[1] += 1  return alg\_count  def **insert**(arr, dim):  alg\_count = [0, 0]  for i in **range**(1, dim):  temp = arr[i]  j = i - 1  while j >= 0:  alg\_count[0] += 1  if arr[j] > temp:  alg\_count[1] += 1  arr[j + 1] = arr[j]  arr[j] = temp  j -= 1  return alg\_count  def **bubble**(arr, dim):  n = 1  alg\_count = [0, 0]  while n < dim:  for i in **range**(dim - n):  alg\_count[0] += 1  if arr[i] > arr[i + 1]:  arr[i], arr[i + 1] = arr[i + 1], arr[i]  alg\_count[1] += 1  n += 1  return alg\_count  import random  DIM = 40  bubble\_arr = []  insert\_arr = []  select\_arr = []  CTotal = [0, 0, 0]  MTotal = [0, 0, 0]  for i in **range**(1, DIM+1):  select\_arr.append(i)  bubble\_arr.append(i)  insert\_arr.append(i)  myfile = **open**("sort\_methods.txt", "w")  **print**("\nУПОРЯДОЧЕННАЯ ПОСЛЕДОВАТЕЛЬНОСТЬ: Исходный массив")  **print**(select\_arr)  count = [0, 0]  count = select(select\_arr, DIM)  **print**("\nУПОРЯДОЧЕННАЯ ПОСЛЕДОВАТЕЛЬНОСТЬ: Результирующий массив")  **print**(select\_arr)  CTotal[0] = count[0]  MTotal[0] = count[1]  count = [0, 0]  count = insert(insert\_arr, DIM)  CTotal[1] = count[0]  MTotal[1] = count[1]  count = [0, 0]  count = bubble(bubble\_arr, DIM)  CTotal[2] = count[0]  MTotal[2] = count[1]  **print**("УПОРЯДОЧЕННАЯ ПОСЛЕДОВАТЕЛЬНОСТЬ:\n")  **print**("Размер массива: ", DIM)  **print**("Сравнений: ", CTotal[0], " ", CTotal[1], " ", CTotal[2])  **print**("Перестановок: ", MTotal[0], " ", MTotal[1], " ", MTotal[2])  select\_arr.clear()  bubble\_arr.clear()  insert\_arr.clear()  for i in **range**(DIM, 0, -1):  select\_arr.append(i)  bubble\_arr.append(i)  insert\_arr.append(i)  **print**("\nОБРАТНО УПОРЯДОЧЕННАЯ ПОСЛЕДОВАТЕЛЬНОСТЬ: Исходный массив")  **print**(select\_arr)  count = [0, 0]  count = select(select\_arr, DIM)  **print**("\nОБРАТНО УПОРЯДОЧЕННАЯ ПОСЛЕДОВАТЕЛЬНОСТЬ: Результирующий массив")  **print**(select\_arr)  CTotal[0] = count[0]  MTotal[0] = count[1]  count = [0, 0]  count = insert(insert\_arr, DIM)  CTotal[1] = count[0]  MTotal[1] = count[1]  count = [0, 0]  count = bubble(bubble\_arr, DIM)  CTotal[2] = count[0]  MTotal[2] = count[1]  **print**("Размер массива: ", DIM)  **print**("Сравнений: ", CTotal[0], " ", CTotal[1], " ", CTotal[2])  **print**("Перестановок: ", MTotal[0], " ", MTotal[1], " ", MTotal[2])  NUM = 1500  CTotal.clear()  MTotal.clear()  CTotal = [0, 0, 0]  MTotal = [0, 0, 0]  for n in **range**(0, NUM):  select\_arr.clear()  bubble\_arr.clear()  insert\_arr.clear()  select\_arr = [random.randint(0, 100) for i in **range**(DIM)]  for i in **range**(0, DIM):  bubble\_arr.append(select\_arr[i])  insert\_arr.append(select\_arr[i])  count = [0, 0]  count = select(select\_arr, DIM)  CTotal[0] += count[0]  MTotal[0] += count[1]  count = [0, 0]  count = insert(insert\_arr, DIM)  CTotal[1] += count[0]  MTotal[1] += count[1]  count = [0, 0]  count = bubble(bubble\_arr, DIM)  CTotal[2] += count[0]  MTotal[2] += count[1]  **print**("\nСЛУЧАЙНАЯ РЕАЛИЗАЦИЯ:")  **print**("Проведено экспериментов: ", NUM)  **print**("Размер массива: ", DIM)  **print**("Сравнений: ", CTotal[0]/NUM, " ", CTotal[1]/NUM, " ", CTotal[2]/NUM)  **print**("Перестановок: ", MTotal[0]/NUM, " ", MTotal[1]/NUM, " ", MTotal[2]/NUM) |

Результат: Самый эффективный метод сортировки - это select sort, т. к. количество сравнений и перемещений меньше чем у остальных.

|  |
| --- |
| УПОРЯДОЧЕННАЯ ПОСЛЕДОВАТЕЛЬНОСТЬ: Исходный массив  [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40]  УПОРЯДОЧЕННАЯ ПОСЛЕДОВАТЕЛЬНОСТЬ: Результирующий массив  [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40]  УПОРЯДОЧЕННАЯ ПОСЛЕДОВАТЕЛЬНОСТЬ:  Размер массива: 40  Сравнений: 780 780 780  Перестановок: 0 0 0  ОБРАТНО УПОРЯДОЧЕННАЯ ПОСЛЕДОВАТЕЛЬНОСТЬ: Исходный массив  [40, 39, 38, 37, 36, 35, 34, 33, 32, 31, 30, 29, 28, 27, 26, 25, 24, 23, 22, 21, 20, 19, 18, 17, 16, 15, 14, 13, 12, 11, 10, 9, 8, 7, 6, 5, 4, 3, 2, 1]  ОБРАТНО УПОРЯДОЧЕННАЯ ПОСЛЕДОВАТЕЛЬНОСТЬ: Результирующий массив  [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40]  Размер массива: 40  Сравнений: 780 780 780  Перестановок: 20 780 780 |

# ПАКЕТ NUMPY

NumPy это open-source модуль для python, который предоставляет общие математические и числовые операции в виде пре-скомпилированных, быстрых функций. Они объединяются в высокоуровневые пакеты. Они обеспечивают функционал, который можно сравнить с функционалом MatLab. NumPy (Numeric Python) предоставляет базовые методы для манипуляции с большими массивами и матрицами. SciPy (Scientific Python) расширяет функционал numpy огромной коллекцией полезных алгоритмов, таких как минимизация, преобразование Фурье, регрессия, и другие прикладные математические техники. Сообщество NumPy и SciPy поддерживает онлайн руководство, включающие гайды и туториалы, тут: docs.scipy.org/doc.

Подключение пакета Numpy:

Есть несколько путей импорта. Стандартный метод это — использовать простое выражение:

import numpy

Тем не менее, для большого количества вызовов функций numpy, становиться утомительно писать numpy.X снова и снова. Вместо этого намного легче сделать это так:

import numpy as np

Это выражение позволяет нам получать доступ к numpy объектам используя np.X вместо numpy.X. Также можно импортировать numpy прямо в используемое пространство имен, чтобы вообще не использовать функции через точку, а вызывать их напрямую:

from numpy import \*

Практика: Дано 31 задача, которые необходимо решить при помощи пакета 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.

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

3. 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.

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

5. 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.

6. 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.

7. 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.

8. 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.

9. 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.

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

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

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

13. 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.

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

15. 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.

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

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

18. Создать прямоугольную матрицу A, имеющую N строк и M столбцов со случайными элементами. Найти сумму расположенных параллельно побочной диагонали (ближайшие к побочной). Элементы побочной диагонали имеют индексы от [N,0] до [0,N].

19. 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].

20. 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].

21. 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]).

22. 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.

23. 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]).

24. 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.

25. 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.

26. 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.

27. 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.

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

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

30. 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.

31. 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.

Решение:

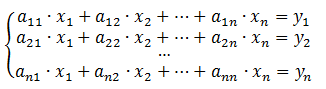
|  |
| --- |
| import numpy as np  import random  **print**("=============================ЗАДАЧА 1=============================")  N = random.randint(5, 10)  M = random.randint(5, 10)  A = np.random.randint(-10, 10, (N, M))  **print**("Матрица:\r\n{}".format(A))  sum\_cols = np.sum(np.absolute(A), axis=0)  max\_sum = np.max(sum\_cols)  max\_sum\_col = np.argmax(sum\_cols)  col = A[:, max\_sum\_col]  max\_el = np.max(col)  **print**("Наибольший элемент: {}".format(max\_el))  **print**("=============================ЗАДАЧА 2=============================")  N = random.randint(5, 10)  M = random.randint(5, 10)  A = np.random.randint(-10, 10, (N, M))  **print**("Матрица:\r\n{}".format(A))  average = np.average(A, axis=1)  max\_average = np.max(average)  **print**("Наибольшее среднее значение: {}".format(max\_average))  **print**("=============================ЗАДАЧА 3=============================")  N = random.randint(5, 10)  M = random.randint(5, 10)  A = np.random.randint(-10, 10, (N, M))  **print**("Матрица:\r\n{}".format(A))  sum\_cols = np.sum(np.absolute(A), axis=0)  max\_sum = np.max(sum\_cols)  max\_sum\_col = np.argmax(sum\_cols)  col = A[:, max\_sum\_col]  min\_el = np.min(col)  **print**("Наименьший элемент: {}".format(min\_el))  **print**("=============================ЗАДАЧА 4=============================")  N = random.randint(5, 10)  M = random.randint(5, 10)  A = np.random.randint(-10, 10, (N, M))  **print**("Матрица:\r\n{}".format(A))  average = np.average(A, axis=1)  min\_average = np.min(average)  **print**("Наименьшее среднее значение: {}".format(min\_average))  **print**("=============================ЗАДАЧА 5=============================")  N = random.randint(5, 10)  M = random.randint(5, 10)  A = np.random.randint(-10, 10, (N, M))  **print**("Матрица:\r\n{}".format(A))  average\_rows = np.average(A, axis=1)  average\_cols = np.average(A, axis=0)  A = np.vstack((A, [average\_cols]))  A = np.column\_stack((A, np.append(average\_rows, 0)))  **print**("Полученная матрица:\r\n {}".format(A))  **print**("=============================ЗАДАЧА 6=============================")  N = random.randint(5, 10)  M = random.randint(5, 10)  A = np.random.randint(-10, 10, (N, M))  **print**("Матрица:\r\n{}".format(A))  sum\_elements = np.sum(A)  sum\_cols = np.sum(A, axis=0)  A = np.vstack((A, [sum\_cols/sum\_elements]))  **print**("Полученная матрица:\r\n {}".format(A))  **print**("=============================ЗАДАЧА 7=============================")  N = random.randint(5, 10)  M = random.randint(5, 10)  A = np.random.randint(-10, 10, (N, M))  **print**("Матрица:\r\n{}".format(A))  sum\_elements = np.sum(A)  sum\_cols = np.sum(A, axis=1)  A = np.column\_stack((A, sum\_cols/sum\_elements))  **print**("Полученная матрица:\r\n {}".format(A))  **print**("=============================ЗАДАЧА 8=============================")  N = random.randint(5, 10)  M = random.randint(5, 10)  A = np.random.randint(-10, 10, (N, M))  **print**("Матрица:\r\n{}".format(A))  A\_bool = A < 0  num\_cols = np.sum(A\_bool, axis=0)  num\_rows = np.sum(A\_bool, axis=1)  A = np.vstack((A, [num\_cols]))  A = np.column\_stack((A, np.append(num\_rows, 0)))  **print**("Полученная матрица:\r\n {}".format(A))  **print**("=============================ЗАДАЧА 9=============================")  N = random.randint(5, 10)  M = random.randint(5, 10)  L = random.randint(1, 5)  K = random.randint(1, 5)  A = np.random.randint(-10, 10, (N, M))  **print**("Матрица:\r\n{}".format(A))  A\_slice = A[:L, :K]  slice\_bool = A\_slice == 0  count\_zero\_elements = slice\_bool.sum()  **print**("Количество нулевых элементов в срезе: {}".format(count\_zero\_elements))  **print**("=============================ЗАДАЧА 10=============================")  N = random.randint(5, 10)  M = random.randint(5, 10)  K = random.randint(1, 5)  A = np.random.randint(-10, 10, (N, M))  **print**("Матрица:\r\n{}".format(A))  col = A[:, K]  col = col.reshape(**len**(col), 1)  A = A \* col  **print**("Полученная матрица:\r\n {}".format(A))  **print**("=============================ЗАДАЧА 11=============================")  N = random.randint(5, 10)  M = random.randint(5, 10)  L = random.randint(1, 5)  A = np.random.randint(-10, 10, (N, M))  **print**("Матрица:\r\n{}".format(A))  A = A + A[L]  **print**("Полученная матрица:\r\n {}".format(A))  **print**("=============================ЗАДАЧА 12=============================")  N = random.randint(5, 10)  M = random.randint(5, 10)  A = np.random.randint(-10, 10, (N, M))  **print**("Матрица:\r\n{}".format(A))  row\_max = np.max(A, axis=1)  A = A / row\_max.reshape(**len**(row\_max), 1)  **print**("Полученная матрица:\r\n {}".format(A))  **print**("=============================ЗАДАЧА 13=============================")  N = random.randint(5, 10)  M = random.randint(5, 10)  A = np.random.randint(-10, 10, (N, M))  **print**("Матрица:\r\n{}".format(A))  col\_max = np.max(A, axis=0)  A = A / col\_max  **print**("Полученная матрица:\r\n {}".format(A))  **print**("=============================ЗАДАЧА 14=============================")  N = random.randint(5, 10)  M = random.randint(5, 10)  A = np.random.randint(-10, 10, (N, M))  **print**("Матрица:\r\n{}".format(A))  max\_el = np.max(A)  A = A / max\_el  **print**("Полученная матрица:\r\n {}".format(A))  **print**("=============================ЗАДАЧА 15=============================")  N = random.randint(5, 10)  M = random.randint(5, 10)  H = random.randint(1, 5)  A = np.random.randint(-10, 10, (N, M))  **print**("Матрица:\r\n{}".format(A))  A\_bool = A == H  row\_sum = np.sum(A\_bool, axis=0)  **print**("Столбцы в которых встречается значение {}:".format(H))  **print**(np.argwhere(row\_sum).flatten())  **print**("Столбцы в которых нет значения {}:".format(H))  **print**(np.argwhere(row\_sum == 0).flatten())  **print**("=============================ЗАДАЧА 16=============================")  N = random.randint(5, 10)  M = random.randint(5, 10)  L = random.randint(1, 5)  A = np.random.randint(-10, 10, (N, M))  **print**("Матрица:\r\n{}".format(A))  A = np.delete(A, L, axis=0)  **print**("Полученная матрица:\r\n {}".format(A))  **print**("=============================ЗАДАЧА 17=============================")  N = random.randint(5, 10)  M = random.randint(5, 10)  L = random.randint(1, 5)  A = np.random.randint(-10, 10, (N, M))  **print**("Матрица:\r\n{}".format(A))  row = np.random.randint(-9, 10, M)  A = np.insert(A, L, row, axis=0)  **print**("Полученная матрица:\r\n {}".format(A))  **print**("=============================ЗАДАЧА 18=============================")  N = random.randint(5, 10)  M = random.randint(5, 10)  A = np.random.randint(-10, 10, (N, M))  **print**("Матрица:\r\n{}".format(A))  diagonal\_main = np.diagonal(A)  sum\_diagonal\_main = np.sum(diagonal\_main)  **print**("Cумма элементов главной диагонали:\r\n{}".format(sum\_diagonal\_main))  diagonal\_side = np.diagonal(A[::-1])  sum\_diagonal\_side = np.sum(diagonal\_side)  **print**("сумму элементов побочной диагонали:\r\n{}".format(sum\_diagonal\_side))  **print**("=============================ЗАДАЧА 19=============================")  N = random.randint(5, 10)  A = np.random.randint(-10, 10, (N, N))  **print**("Матрица:\r\n{}".format(A))  diagonal\_elements = np.array([np.diagonal(A, i) for i in [1, -1]]).flatten()  **print**("Элементы, расположенные параллельно главной диагонали:")  **print**(diagonal\_elements)  **print**("Сумма элементов, расположенные параллельно главной диагонали:")  **print**(np.sum(diagonal\_elements))  **print**("=============================ЗАДАЧА 20=============================")  N = random.randint(5, 10)  A = np.random.randint(-10, 10, (N, N))  **print**("Матрица:\r\n{}".format(A))  diagonal\_elements = np.array([np.diagonal(A[::-1], i) for i in [1, -1]]).flatten()  **print**("Сумма элементов, расположенные параллельно побочной диагонали:")  **print**(np.prod(diagonal\_elements))  **print**("=============================ЗАДАЧА 21=============================")  N = random.randint(5, 10)  A = np.random.randint(-10, 10, (N, N))  **print**("Матрица:\r\n{}".format(A))  diagonal\_elements = [np.diagonal(A, i) for i in [1, -1]]  values = (diagonal\_elements[0] + diagonal\_elements[1])/2  rng = np.arange(N-1)  A[rng, rng+1] = values[rng]  A[rng+1, rng] = values[rng]  **print**("Полученная матрица:\r\n{}".format(A))  **print**("=============================ЗАДАЧА 22=============================")  N = random.randint(5, 10)  M = random.randint(5, 10)  A = np.random.randint(0, 2, (N, M))  **print**("Матрица:\r\n{}".format(A))  col = [i % 2 for i in np.sum(A, axis=1)]  A = np.insert(A, M, col, axis=1)  **print**("Полученная матрица:\r\n {}".format(A))  **print**("=============================ЗАДАЧА 23=============================")  N = random.randint(5, 10)  A = np.random.randint(-10, 10, (N, N))  **print**("Матрица:\r\n{}".format(A))  elements = np.diagonal(A, 1)  **print**("Сумма элементов, расположенных выше главной диагонали:")  **print**(np.sum(elements))  elements = np.diagonal(A[::-1], -1)  **print**("Произведение элементов, расположенных выше побочной диагонали:")  **print**(np.prod(elements))  **print**("=============================ЗАДАЧА 24=============================")  N = random.randint(5, 10)  M = random.randint(5, 10)  L = random.randint(1, 5)  K = random.randint(1, 5)  A = np.random.randint(-10, 10, (N, M))  **print**("Матрица:\r\n{}".format(A))  parts = [  A[:L, :K],  A[:L, K:],  A[L:, :K],  A[L:, K:],  ]  for i in **range**(**len**(parts)):  **print**("Cумма элементов {} части: {}".format(i+1, np.sum(parts[i])))  **print**("=============================ЗАДАЧА 25=============================")  N = random.randint(5, 10)  M = random.randint(5, 10)  L = random.randint(1, 5)  K = random.randint(1, 5)  A = np.random.randint(-10, 10, (N, M))  **print**("Матрица:\r\n{}".format(A))  A\_bool = A == 0  col = np.sum(A\_bool, axis=1)  A = np.insert(A, M, col, axis=1)  row = np.append(np.sum(A\_bool, axis=0), 0)  A = np.insert(A, N, row, axis=0)  **print**("Полученная матрица:\r\n {}".format(A))  **print**("=============================ЗАДАЧА 26=============================")  N = random.randint(5, 10)  M = random.randint(5, 10)  L = random.randint(1, 5)  K = random.randint(1, 5)  A = np.random.randint(-10, 10, (N, M))  **print**("Матрица:\r\n{}".format(A))  parts = [  A[:L, :K],  A[:L, K:],  A[L:, :K],  A[L:, K:],  ]  for i in **range**(**len**(parts)):  **print**("Cреднее арифметическое {} части: {}".format(i+1, np.average(parts[i])))  **print**("=============================ЗАДАЧА 27=============================")  N = random.randint(5, 10)  M = random.randint(5, 10)  H = random.randint(1, 5)  A = np.random.randint(-10, 10, (N, M))  **print**("Матрица:\r\n{}".format(A))  A\_bool = A == H  col\_sum = np.sum(A\_bool, axis=1)  **print**("Строки в которых встречается значение {}:".format(H))  **print**(np.argwhere(col\_sum).flatten())  **print**("Строки в которых нет значения {}:".format(H))  **print**(np.argwhere(col\_sum == 0).flatten())  **print**("=============================ЗАДАЧА 28=============================")  N = random.randint(5, 10)  M = random.randint(5, 10)  K = random.randint(1, 5)  A = np.random.randint(-10, 10, (N, M))  **print**("Матрица:\r\n{}".format(A))  A = np.delete(A, K, axis=1)  **print**("Полученная матрица:\r\n {}".format(A))  **print**("=============================ЗАДАЧА 29=============================")  N = random.randint(5, 10)  M = random.randint(5, 10)  K = random.randint(1, 5)  A = np.random.randint(-10, 10, (N, M))  **print**("Матрица:\r\n{}".format(A))  col = np.random.randint(-9, 10, N)  **print**("Столбец для вставки: {}".format(col))  A = np.insert(A, K, col, axis=1)  **print**("Полученная матрица:\r\n {}".format(A))  **print**("=============================ЗАДАЧА 30=============================")  N = random.randint(5, 10)  M = random.randint(5, 10)  A = np.random.randint(-10, 10, (N, M))  **print**("Матрица:\r\n{}".format(A))  row = np.sum(A, axis=0) \* -1  A = np.insert(A, N, row, axis=0)  **print**("Полученная матрица:\r\n {}".format(A))  **print**("=============================ЗАДАЧА 31=============================")  N = random.randint(5, 10)  M = random.randint(5, 10)  A = np.random.randint(-10, 10, (N, M))  **print**("Матрица:\r\n{}".format(A))  col = np.sum(A, axis=1) \* -1  A = np.insert(A, M, col, axis=1)  **print**("Полученная матрица:\r\n {}".format(A)) |

Результат:

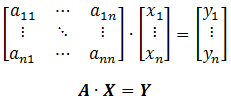
|  |
| --- |
| =============================ЗАДАЧА 1=============================  Матрица:  [[ -5 -6 7 4 0 -6 -9 -1 3 -8]  [ -7 -9 -10 -6 7 5 -3 -7 -6 3]  [ -1 -7 -10 5 6 8 -4 -4 9 -2]  [ -2 3 8 -8 -1 -6 9 9 -8 -9]  [ -5 -2 4 6 -8 6 -7 2 0 6]  [ 0 -2 4 -2 8 -10 -3 -6 1 0]  [ -6 9 2 8 2 -8 -10 -9 -10 -1]  [ -5 -9 -6 5 -9 7 -2 5 2 5]  [ 6 9 -6 -2 -2 2 -7 -5 3 7]  [ 4 -1 -4 1 -2 1 -9 9 -1 -6]]  Наибольший элемент: 9  =============================ЗАДАЧА 2=============================  Матрица:  [[ 6 6 2 -1 -8 2]  [ 9 9 -5 7 -5 -3]  [-2 8 -2 -8 -7 -5]  [ 2 8 7 1 -5 1]  [-9 -8 7 -8 4 9]  [-7 7 -6 -4 -7 2]  [ 2 -6 -9 2 1 2]]  Наибольшее среднее значение: 2.3333333333333335  =============================ЗАДАЧА 3=============================  Матрица:  [[ 5 -1 4 -10 3 -10 -1 -10 7 1]  [ 8 -7 0 -1 9 -3 -10 8 9 -10]  [ -3 -2 -3 -10 5 1 -2 -4 0 3]  [ 6 8 3 3 -4 -10 3 7 9 -10]  [ -3 -3 -8 -10 5 -3 7 -10 3 -10]  [ 7 -10 -8 6 -7 -1 6 5 -9 0]  [ 3 -1 4 -10 5 4 1 -5 9 -4]  [ -1 -8 3 -8 -3 -3 8 0 6 -2]  [ -2 7 -1 -9 2 -8 -3 -4 -8 4]]  Наименьший элемент: -10  =============================ЗАДАЧА 4=============================  Матрица:  [[ -2 8 5 -4 7 -10 8 7 -10 3]  [-10 8 -7 1 4 3 0 3 -3 -3]  [ -6 -6 -10 9 -9 -7 -9 1 7 1]  [ -9 -9 0 -7 -8 6 7 -6 -4 -9]  [ 0 -3 -2 -3 -7 -7 8 -4 6 6]  [ 8 5 -6 3 -1 8 -7 6 8 -3]  [ -6 -5 -9 -3 4 4 -4 6 -9 2]]  Наименьшее среднее значение: -3.9  =============================ЗАДАЧА 5=============================  Матрица:  [[ -8 -1 6 8 9 8 -9 7]  [-10 4 -8 -8 5 1 -1 -2]  [ -7 -8 0 -3 3 -6 4 -7]  [ 8 2 5 -10 -1 2 5 -7]  [ 0 3 -5 9 6 -1 -4 5]  [ -5 -6 -8 0 -10 6 -6 4]  [ 2 -8 5 0 -3 0 -10 8]]  Полученная матрица:  [[ -8. -1. 6. 8. 9.  8. -9. 7. 2.5 ]  [-10. 4. -8. -8. 5.  1. -1. -2. -2.375 ]  [ -7. -8. 0. -3. 3.  -6. 4. -7. -3. ]  [ 8. 2. 5. -10. -1.  2. 5. -7. 0.5 ]  [ 0. 3. -5. 9. 6.  -1. -4. 5. 1.625 ]  [ -5. -6. -8. 0. -10.  6. -6. 4. -3.125 ]  [ 2. -8. 5. 0. -3.  0. -10. 8. -0.75 ]  [ -2.85714286 -2. -0.71428571 -0.57142857 1.28571429  1.42857143 -3. 1.14285714 0. ]]  =============================ЗАДАЧА 6=============================  Матрица:  [[ 4 7 -2 5 6 9 8 -3 -9 3]  [ 3 -5 -8 2 9 7 1 -9 -9 -5]  [ 7 2 6 2 4 -2 6 -3 -3 -2]  [ -1 -10 -9 -1 5 1 -2 -7 -8 -4]  [ -5 2 8 -1 -10 7 -10 0 4 -9]  [ -6 -6 5 0 -3 -10 -2 2 -2 -7]  [ -9 -3 -4 -1 -6 6 4 1 9 4]  [ -5 1 -7 -6 -6 -8 -9 9 -7 9]  [ 5 4 4 5 -4 -2 6 0 -7 9]]  Полученная матрица:  [[ 4. 7. -2. 5. 6.  9. 8. -3. -9. 3. ]  [ 3. -5. -8. 2. 9.  7. 1. -9. -9. -5. ]  [ 7. 2. 6. 2. 4.  -2. 6. -3. -3. -2. ]  [ -1. -10. -9. -1. 5.  1. -2. -7. -8. -4. ]  [ -5. 2. 8. -1. -10.  7. -10. 0. 4. -9. ]  [ -6. -6. 5. 0. -3.  -10. -2. 2. -2. -7. ]  [ -9. -3. -4. -1. -6.  6. 4. 1. 9. 4. ]  [ -5. 1. -7. -6. -6.  -8. -9. 9. -7. 9. ]  [ 5. 4. 4. 5. -4.  -2. 6. 0. -7. 9. ]  [ 0.125 0.14285714 0.125 -0.08928571 0.08928571  -0.14285714 -0.03571429 0.17857143 0.57142857 0.03571429]]  =============================ЗАДАЧА 7=============================  Матрица:  [[ -8 -9 7 3 -8 2 0 4 -3 -7]  [ 2 1 7 -9 7 3 1 3 2 8]  [ -2 1 5 9 -5 -10 0 -6 7 -10]  [ 2 -10 6 5 6 3 7 8 2 0]  [ -9 -2 5 -7 8 -2 4 9 2 2]  [ -5 -1 -4 -10 -7 1 2 -4 -6 3]  [ 1 6 -6 5 9 -2 -7 8 -10 -5]  [ 7 1 9 -1 -5 2 2 1 -4 -4]  [ 0 3 -10 0 -9 -1 -2 7 3 -6]  [ -8 2 4 0 4 4 -9 -7 1 0]]  Полученная матрица:  [[ -8. -9. 7. 3. -8.  2. 0. 4. -3. -7.  1.35714286]  [ 2. 1. 7. -9. 7.  3. 1. 3. 2. 8.  -1.78571429]  [ -2. 1. 5. 9. -5.  -10. 0. -6. 7. -10.  0.78571429]  [ 2. -10. 6. 5. 6.  3. 7. 8. 2. 0.  -2.07142857]  [ -9. -2. 5. -7. 8.  -2. 4. 9. 2. 2.  -0.71428571]  [ -5. -1. -4. -10. -7.  1. 2. -4. -6. 3.  2.21428571]  [ 1. 6. -6. 5. 9.  -2. -7. 8. -10. -5.  0.07142857]  [ 7. 1. 9. -1. -5.  2. 2. 1. -4. -4.  -0.57142857]  [ 0. 3. -10. 0. -9.  -1. -2. 7. 3. -6.  1.07142857]  [ -8. 2. 4. 0. 4.  4. -9. -7. 1. 0.  0.64285714]]  =============================ЗАДАЧА 8=============================  Матрица:  [[ -9 -2 -5 -3 0 8 -1]  [ -1 1 1 -4 -4 -6 6]  [ -1 -10 -2 -10 -2 7 -5]  [ -1 7 -10 6 1 7 -6]  [ 1 6 -9 -6 3 2 8]  [ 5 4 -2 -7 7 -9 4]  [ 9 8 9 -7 9 5 4]  [ -1 -3 -6 -5 2 3 -5]]  Полученная матрица:  [[ -9 -2 -5 -3 0 8 -1 5]  [ -1 1 1 -4 -4 -6 6 4]  [ -1 -10 -2 -10 -2 7 -5 6]  [ -1 7 -10 6 1 7 -6 3]  [ 1 6 -9 -6 3 2 8 2]  [ 5 4 -2 -7 7 -9 4 3]  [ 9 8 9 -7 9 5 4 1]  [ -1 -3 -6 -5 2 3 -5 5]  [ 5 3 6 7 2 2 4 0]]  =============================ЗАДАЧА 9=============================  Матрица:  [[ 5 -5 -1 8 -9 1]  [ -6 4 -8 -4 -8 7]  [ -1 -7 -4 -1 -5 -6]  [ 7 -10 -8 5 3 7]  [ 1 1 1 -1 7 -6]]  Количество нулевых элементов в срезе: 0  =============================ЗАДАЧА 10=============================  Матрица:  [[ -7 5 -5 -1 -1 -8 -6 -2 -9]  [ 0 5 5 7 9 0 2 9 -9]  [ 6 7 9 -8 -6 -8 3 3 -7]  [ -4 4 3 -10 4 -2 -4 4 -8]  [ 6 -1 -8 -7 1 -5 9 4 -2]  [ 0 3 -6 4 1 -7 -9 0 -7]  [ 9 -10 -8 9 -5 -9 7 7 1]  [ -7 -7 -3 -4 -5 -7 -8 9 -9]  [ -2 -9 5 1 6 6 7 4 4]]  Полученная матрица:  [[-35 25 -25 -5 -5 -40 -30 -10 -45]  [ 0 25 25 35 45 0 10 45 -45]  [ 42 49 63 -56 -42 -56 21 21 -49]  [-16 16 12 -40 16 -8 -16 16 -32]  [ -6 1 8 7 -1 5 -9 -4 2]  [ 0 9 -18 12 3 -21 -27 0 -21]  [-90 100 80 -90 50 90 -70 -70 -10]  [ 49 49 21 28 35 49 56 -63 63]  [ 18 81 -45 -9 -54 -54 -63 -36 -36]]  =============================ЗАДАЧА 11=============================  Матрица:  [[ -9 2 0 -7 -5 -4 -7 -2 8]  [ -1 1 -2 3 -10 -4 4 4 3]  [ 9 2 -8 -6 -9 -1 -1 2 -3]  [ 6 -1 -2 5 -9 -10 -4 -1 2]  [ -8 2 -5 1 2 7 -7 0 -5]  [ -8 -3 3 -1 1 -9 -6 9 3]  [ -7 -7 -7 -3 -8 -10 -8 -7 5]]  Полученная матрица:  [[ -3 1 -2 -2 -14 -14 -11 -3 10]  [ 5 0 -4 8 -19 -14 0 3 5]  [ 15 1 -10 -1 -18 -11 -5 1 -1]  [ 12 -2 -4 10 -18 -20 -8 -2 4]  [ -2 1 -7 6 -7 -3 -11 -1 -3]  [ -2 -4 1 4 -8 -19 -10 8 5]  [ -1 -8 -9 2 -17 -20 -12 -8 7]]  =============================ЗАДАЧА 12=============================  Матрица:  [[ 7 -4 -3 -10 1 -8 -5 7 -8]  [ 9 -9 -4 5 7 0 8 6 -4]  [ -3 -5 9 1 0 2 1 -9 -1]  [-10 -3 -5 -3 -6 8 -2 4 5]  [ -6 1 0 -1 4 5 2 4 6]  [-10 5 -9 6 8 1 1 -4 1]  [ 6 -3 2 8 -2 9 -6 1 -3]  [ 5 5 2 3 -6 -10 6 -5 4]  [ -8 4 5 3 -6 -10 -3 -8 -4]  [ -7 3 -1 7 -1 -1 -6 6 -8]]  Полученная матрица:  [[ 1. -0.57142857 -0.42857143 -1.42857143 0.14285714 -1.14285714  -0.71428571 1. -1.14285714]  [ 1. -1. -0.44444444 0.55555556 0.77777778 0.  0.88888889 0.66666667 -0.44444444]  [-0.33333333 -0.55555556 1. 0.11111111 0. 0.22222222  0.11111111 -1. -0.11111111]  [-1.25 -0.375 -0.625 -0.375 -0.75 1.  -0.25 0.5 0.625 ]  [-1. 0.16666667 0. -0.16666667 0.66666667 0.83333333  0.33333333 0.66666667 1. ]  [-1.25 0.625 -1.125 0.75 1. 0.125  0.125 -0.5 0.125 ]  [ 0.66666667 -0.33333333 0.22222222 0.88888889 -0.22222222 1.  -0.66666667 0.11111111 -0.33333333]  [ 0.83333333 0.83333333 0.33333333 0.5 -1. -1.66666667  1. -0.83333333 0.66666667]  [-1.6 0.8 1. 0.6 -1.2 -2.  -0.6 -1.6 -0.8 ]  [-1. 0.42857143 -0.14285714 1. -0.14285714 -0.14285714  -0.85714286 0.85714286 -1.14285714]]  =============================ЗАДАЧА 13=============================  Матрица:  [[ 1 -3 7 0 1 9 0 5 4]  [-3 7 1 6 -9 -2 -8 2 7]  [-6 3 2 -2 -8 8 -2 9 -2]  [ 3 -2 7 -4 -5 -7 0 6 -8]  [-6 5 5 1 4 -7 7 -5 3]  [ 1 -9 7 -5 7 -5 -6 -4 4]]  Полученная матрица:  [[ 0.33333333 -0.42857143 1. 0. 0.14285714 1.  0. 0.55555556 0.57142857]  [-1. 1. 0.14285714 1. -1.28571429 -0.22222222  -1.14285714 0.22222222 1. ]  [-2. 0.42857143 0.28571429 -0.33333333 -1.14285714 0.88888889  -0.28571429 1. -0.28571429]  [ 1. -0.28571429 1. -0.66666667 -0.71428571 -0.77777778  0. 0.66666667 -1.14285714]  [-2. 0.71428571 0.71428571 0.16666667 0.57142857 -0.77777778  1. -0.55555556 0.42857143]  [ 0.33333333 -1.28571429 1. -0.83333333 1. -0.55555556  -0.85714286 -0.44444444 0.57142857]]  =============================ЗАДАЧА 14=============================  Матрица:  [[ -3 6 -7 0 2 4 1 5 -1 5]  [ -4 8 -9 -7 1 -1 -6 -8 -7 -6]  [ -3 -9 1 -7 7 3 2 3 -7 -4]  [ -1 -4 3 0 8 -3 -10 -7 -9 -4]  [ 2 3 3 3 4 -5 4 -7 -4 6]  [ 6 -1 8 -4 -4 -1 -3 4 3 3]  [-10 6 -7 -8 -1 -1 5 3 3 -8]]  Полученная матрица:  [[-0.375 0.75 -0.875 0. 0.25 0.5 0.125 0.625 -0.125 0.625]  [-0.5 1. -1.125 -0.875 0.125 -0.125 -0.75 -1. -0.875 -0.75 ]  [-0.375 -1.125 0.125 -0.875 0.875 0.375 0.25 0.375 -0.875 -0.5 ]  [-0.125 -0.5 0.375 0. 1. -0.375 -1.25 -0.875 -1.125 -0.5 ]  [ 0.25 0.375 0.375 0.375 0.5 -0.625 0.5 -0.875 -0.5 0.75 ]  [ 0.75 -0.125 1. -0.5 -0.5 -0.125 -0.375 0.5 0.375 0.375]  [-1.25 0.75 -0.875 -1. -0.125 -0.125 0.625 0.375 0.375 -1. ]]  =============================ЗАДАЧА 15=============================  Матрица:  [[ 2 5 -8 -8 -9 -6 -9]  [ -3 9 9 8 9 3 -4]  [ -4 -5 0 2 -3 5 -4]  [ -6 -9 4 7 -6 -10 -2]  [ -5 9 -1 8 9 5 1]  [ 0 -9 -5 5 3 7 2]  [ -7 -1 3 -3 1 -6 6]  [ 5 0 -9 -5 3 7 6]  [ 7 -7 -4 -3 9 -1 8]  [ 2 4 9 -10 2 -2 -9]]  Столбцы в которых встречается значение 5:  [0 1 3 5]  Столбцы в которых нет значения 5:  [2 4 6]  =============================ЗАДАЧА 16=============================  Матрица:  [[ 9 3 -7 0 5 -5 -8 7 1 -7]  [ 4 5 8 4 7 2 2 2 4 7]  [ 0 -8 -9 -7 -7 -9 5 8 9 8]  [ 4 -5 -7 -7 6 -4 -5 -9 7 -3]  [ 6 5 -6 0 8 -10 6 -5 -3 -5]  [ 3 9 4 -10 -9 -8 5 -9 -4 -9]  [ 5 8 -9 -9 0 -7 1 6 -4 -1]  [ -2 2 9 9 0 5 -8 7 -9 3]  [ 1 1 -1 5 0 -1 -5 8 -10 7]  [ -5 4 -1 4 3 0 0 -6 -9 -2]]  Полученная матрица:  [[ 9 3 -7 0 5 -5 -8 7 1 -7]  [ 4 5 8 4 7 2 2 2 4 7]  [ 0 -8 -9 -7 -7 -9 5 8 9 8]  [ 4 -5 -7 -7 6 -4 -5 -9 7 -3]  [ 6 5 -6 0 8 -10 6 -5 -3 -5]  [ 5 8 -9 -9 0 -7 1 6 -4 -1]  [ -2 2 9 9 0 5 -8 7 -9 3]  [ 1 1 -1 5 0 -1 -5 8 -10 7]  [ -5 4 -1 4 3 0 0 -6 -9 -2]]  =============================ЗАДАЧА 17=============================  Матрица:  [[-10 3 -3 0 -2 1]  [ 9 -7 4 6 -8 -3]  [ 2 -8 -9 -6 -2 -6]  [ 0 8 5 -2 9 -6]  [ 8 5 -3 3 -1 -10]  [ -6 -10 -7 8 8 2]  [ -7 2 8 3 -10 8]  [ -3 -2 5 9 -7 -1]]  Полученная матрица:  [[-10 3 -3 0 -2 1]  [ 9 -7 4 6 -8 -3]  [ -5 2 7 9 -8 -6]  [ 2 -8 -9 -6 -2 -6]  [ 0 8 5 -2 9 -6]  [ 8 5 -3 3 -1 -10]  [ -6 -10 -7 8 8 2]  [ -7 2 8 3 -10 8]  [ -3 -2 5 9 -7 -1]]  =============================ЗАДАЧА 18=============================  Матрица:  [[ -5 8 5 -3 8 6 7]  [ 3 -1 5 0 9 -3 -9]  [ -8 -3 5 -9 1 2 -8]  [ -6 -10 -2 -2 -7 -5 4]  [ -2 1 -2 5 -5 -7 -4]  [ 6 7 3 -1 7 4 -6]  [ 6 -3 -2 -6 -7 7 3]  [ -9 5 3 4 -2 -10 -1]  [ -8 -2 8 1 4 4 4]  [-10 -3 -10 -2 -10 5 -5]]  Cумма элементов главной диагонали:  -1  сумму элементов побочной диагонали:  -11  =============================ЗАДАЧА 19=============================  Матрица:  [[-10 -5 -4 9 -9]  [ 6 -3 -2 2 3]  [ 0 -5 7 -9 -10]  [-10 3 9 3 4]  [ 1 9 -6 3 1]]  Элементы, расположенные параллельно главной диагонали:  [-5 -2 -9 4 6 -5 9 3]  Сумма элементов, расположенные параллельно главной диагонали:  1  =============================ЗАДАЧА 20=============================  Матрица:  [[ -9 -1 -6 -9 -7 -9 -7 8]  [ -1 1 -5 -3 8 9 -3 6]  [ -3 -9 9 -5 3 -8 5 -3]  [ 2 5 0 0 -7 -5 -9 6]  [ 4 -6 -4 6 -4 4 9 -3]  [ 5 -1 3 4 -2 -10 3 6]  [ 0 2 9 -7 -2 -9 8 5]  [ -7 7 -3 0 5 7 -1 4]]  Сумма элементов, расположенные параллельно побочной диагонали:  0  =============================ЗАДАЧА 21=============================  Матрица:  [[ 3 2 4 6 1]  [ -6 4 3 0 3]  [ 5 8 -4 4 0]  [ 7 -10 -1 -2 0]  [ 3 8 9 5 4]]  Полученная матрица:  [[ 3 -2 4 6 1]  [ -2 4 5 0 3]  [ 5 5 -4 1 0]  [ 7 -10 1 -2 2]  [ 3 8 9 2 4]]  =============================ЗАДАЧА 22=============================  Матрица:  [[1 0 0 1 0 0 0 1]  [0 0 1 0 0 0 1 0]  [1 0 1 0 0 0 1 1]  [0 0 0 1 0 0 1 1]  [0 1 0 1 1 0 1 1]  [0 0 1 1 0 0 1 1]  [0 1 0 1 1 0 0 1]  [0 1 1 0 0 1 0 1]]  Полученная матрица:  [[1 0 0 1 0 0 0 1 1]  [0 0 1 0 0 0 1 0 0]  [1 0 1 0 0 0 1 1 0]  [0 0 0 1 0 0 1 1 1]  [0 1 0 1 1 0 1 1 1]  [0 0 1 1 0 0 1 1 0]  [0 1 0 1 1 0 0 1 0]  [0 1 1 0 0 1 0 1 0]]  =============================ЗАДАЧА 23=============================  Матрица:  [[ 2 1 6 4 -9 8 4 -8 -8 -6]  [ -9 3 -3 -4 7 -4 -4 4 -7 -7]  [ -2 2 -1 8 -6 -10 -2 6 7 6]  [ -2 4 -6 1 -2 6 -9 3 -3 -2]  [ -7 -2 3 -2 -6 4 6 6 -1 -8]  [ -7 -4 9 2 -3 0 -7 -2 -8 -4]  [ 5 3 8 -9 8 3 -10 5 -4 -8]  [ 3 3 -5 8 7 2 -10 -5 -6 1]  [ 2 0 -9 -8 -4 9 -2 8 -6 4]  [ -3 4 -7 3 -5 -9 -9 -7 4 6]]  Сумма элементов, расположенных выше главной диагонали:  4  Произведение элементов, расположенных выше побочной диагонали:  -221184  =============================ЗАДАЧА 24=============================  Матрица:  [[ -4 -9 0 3 9 -3 -5 -9]  [ -8 7 8 -8 -7 -7 7 6]  [ 0 7 -8 -1 -2 6 -5 -4]  [ -8 -2 3 -6 -7 -3 5 4]  [ -3 -6 1 -7 2 -9 8 8]  [ 2 9 -10 -5 3 -10 3 -4]  [ -3 -6 -9 0 -7 5 -9 3]  [ 6 -6 6 2 -5 3 3 -1]  [ -1 -9 -6 0 -1 -4 5 3]  [ -2 8 -6 -1 -1 -8 0 0]]  Cумма элементов 1 части: -12  Cумма элементов 2 части: -15  Cумма элементов 3 части: -9  Cумма элементов 4 части: -64  =============================ЗАДАЧА 25=============================  Матрица:  [[ -2 -1 -6 2 -5 4 -1 4]  [ 5 -1 8 -7 -6 -4 -10 -8]  [ 3 -7 8 3 -4 -4 -5 -10]  [ -7 2 6 1 -10 3 8 -7]  [ -1 1 -3 -7 -4 -2 -7 1]  [ 9 -7 -3 -3 7 -9 2 -9]  [ 9 2 5 7 1 -10 3 -5]  [ -6 -1 3 5 -3 -3 -10 4]  [ 2 0 1 -5 -6 2 -4 1]  [ -2 3 -1 -10 3 -2 6 0]]  Полученная матрица:  [[ -2 -1 -6 2 -5 4 -1 4 0]  [ 5 -1 8 -7 -6 -4 -10 -8 0]  [ 3 -7 8 3 -4 -4 -5 -10 0]  [ -7 2 6 1 -10 3 8 -7 0]  [ -1 1 -3 -7 -4 -2 -7 1 0]  [ 9 -7 -3 -3 7 -9 2 -9 0]  [ 9 2 5 7 1 -10 3 -5 0]  [ -6 -1 3 5 -3 -3 -10 4 0]  [ 2 0 1 -5 -6 2 -4 1 1]  [ -2 3 -1 -10 3 -2 6 0 1]  [ 0 1 0 0 0 0 0 1 0]]  =============================ЗАДАЧА 26=============================  Матрица:  [[ 0 6 4 -4 -7 -10 -4]  [ 5 -5 -9 2 -4 -9 2]  [ -7 -5 -7 0 -8 9 3]  [ 8 3 -4 8 6 7 6]  [-10 -6 6 -7 9 7 -4]  [ 8 0 9 -8 9 -9 9]]  Cреднее арифметическое 1 части: 0.16666666666666666  Cреднее арифметическое 2 части: -4.25  Cреднее арифметическое 3 части: -0.4166666666666667  Cреднее арифметическое 4 части: 2.3125  =============================ЗАДАЧА 27=============================  Матрица:  [[ 3 4 -9 3 -7]  [ -2 7 4 9 -7]  [ 4 1 3 6 7]  [ 5 -9 -9 7 -2]  [ 1 -6 -9 -4 -3]  [ 3 -1 3 9 -7]  [ -7 -8 -9 -2 -10]  [ 3 5 3 3 1]  [ -6 -6 -8 8 7]  [ -5 -6 -9 -5 6]]  Строки в которых встречается значение 2:  []  Строки в которых нет значения 2:  [0 1 2 3 4 5 6 7 8 9]  =============================ЗАДАЧА 28=============================  Матрица:  [[ 6 -6 2 6 6 5 -7 -8]  [ 1 7 2 5 -9 8 -6 -7]  [ -8 -2 -9 -7 -3 -7 -4 -10]  [ 1 -2 3 2 -1 -1 8 -1]  [ -5 -8 -7 2 -2 -3 -10 -1]  [ 7 -2 -4 1 -6 1 -1 1]  [ -8 9 -1 -3 -1 5 9 -8]  [ -4 5 -9 4 9 -7 2 -5]  [ -2 -10 8 2 9 8 7 5]  [ 3 7 -7 -10 6 -7 3 -2]]  Полученная матрица:  [[ 6 -6 2 6 6 -7 -8]  [ 1 7 2 5 -9 -6 -7]  [ -8 -2 -9 -7 -3 -4 -10]  [ 1 -2 3 2 -1 8 -1]  [ -5 -8 -7 2 -2 -10 -1]  [ 7 -2 -4 1 -6 -1 1]  [ -8 9 -1 -3 -1 9 -8]  [ -4 5 -9 4 9 2 -5]  [ -2 -10 8 2 9 7 5]  [ 3 7 -7 -10 6 3 -2]]  =============================ЗАДАЧА 29=============================  Матрица:  [[ 5 2 -6 0 2 -4 -6 -8 -2]  [-10 9 -7 -7 -3 9 -10 -3 7]  [ -8 1 -10 0 0 2 0 -5 -10]  [ -8 3 5 9 -5 7 0 6 2]  [ 6 6 8 6 6 -2 7 8 8]  [ 8 -4 8 -10 -5 -6 9 -1 -2]  [ 8 1 -6 -1 3 9 6 -9 -9]]  Столбец для вставки: [-6 5 -6 -5 5 -8 -5]  Полученная матрица:  [[ 5 2 -6 0 2 -6 -4 -6 -8 -2]  [-10 9 -7 -7 -3 5 9 -10 -3 7]  [ -8 1 -10 0 0 -6 2 0 -5 -10]  [ -8 3 5 9 -5 -5 7 0 6 2]  [ 6 6 8 6 6 5 -2 7 8 8]  [ 8 -4 8 -10 -5 -8 -6 9 -1 -2]  [ 8 1 -6 -1 3 -5 9 6 -9 -9]]  =============================ЗАДАЧА 30=============================  Матрица:  [[ 5 -2 6 -10 -10]  [ -9 -2 3 9 -6]  [ -8 1 5 8 2]  [ 4 -7 4 -10 -5]  [ 0 -6 9 8 -2]  [ 3 -6 -1 -8 -8]  [ -8 -3 5 4 1]  [ 7 -3 6 -6 -6]  [ -4 6 -2 -7 -10]  [ -1 -8 6 6 -6]]  Полученная матрица:  [[ 5 -2 6 -10 -10]  [ -9 -2 3 9 -6]  [ -8 1 5 8 2]  [ 4 -7 4 -10 -5]  [ 0 -6 9 8 -2]  [ 3 -6 -1 -8 -8]  [ -8 -3 5 4 1]  [ 7 -3 6 -6 -6]  [ -4 6 -2 -7 -10]  [ -1 -8 6 6 -6]  [ 11 30 -41 6 50]]  =============================ЗАДАЧА 31=============================  Матрица:  [[ 0 -6 5 9 3]  [-4 -4 -4 9 0]  [-2 -5 8 -2 -6]  [ 7 0 -1 -7 8]  [-9 -7 8 1 5]  [-7 -3 2 -6 7]]  Полученная матрица:  [[ 0 -6 5 9 3 -11]  [ -4 -4 -4 9 0 3]  [ -2 -5 8 -2 -6 7]  [ 7 0 -1 -7 8 -7]  [ -9 -7 8 1 5 2]  [ -7 -3 2 -6 7 7]] |

# РЕШЕНИЕ СЛАУ МЕТОДОМ ГАУССА

Метод Гаусса — классический метод решения системы линейных алгебраических уравнений (СЛАУ).Рассмотрим систему линейных уравнений с действительными постоянными коэффициентами:



или в матричной форме



Метод Гаусса решения системы линейных уравнений включает в себя 2 стадии:

* последовательное (прямое) исключение;
* обратная подстановка.

Практика: Дано решить СЛАУ методом Гаусса. Для этого был написан код, который открывает csv-файл c СЛАУ и решает его.

Исходные данные:

|  |
| --- |
| Вариант №13  Задача №1  | 1,97 1,48 | \* | X1| = | 0,64|  | 1,97 1,60 | \* | X2| = | 0,10|  Задача №2  | 1,40 1,17 1,45 1,85 1,45 | \* | X1| = | 0,17|  | 1,85 1,04 1,43 1,98 1,83 | \* | X2| = | 0,84|  | 1,40 1,09 1,53 1,27 1,35 | \* | X3| = | 0,95|  | 1,21 1,43 1,78 1,00 1,34 | \* | X4| = | 0,80|  | 1,04 1,94 1,97 1,20 1,90 | \* | X5| = | 0,17|  Задача №3  | 1,05 1,85 1,85 1,57 1,22 | \* | X1| = | 0,22|  | 1,95 1,65 1,39 1,65 1,38 | \* | X2| = | 0,81|  | 1,80 1,83 1,61 1,24 1,84 | \* | X3| = | 0,98|  | 1,01 1,57 1,62 1,47 1,51 | \* | X4| = | 0,54|  | 1,37 1,60 1,26 1,30 1,25 | \* | X5| = | 0,47|  Задача №4  | 1,98 1,16 1,35 1,92 | \* | X1| = | 0,48|  | 1,34 1,16 1,19 1,62 | \* | X2| = | 0,37|  | 1,11 1,40 1,26 1,77 | \* | X3| = | 0,95|  | 1,72 1,41 1,25 1,94 | \* | X4| = | 0,94|  Задача №5  | 1,08 1,21 1,97 | \* | X1| = | 0,26|  | 1,64 1,14 1,78 | \* | X2| = | 0,89|  | 1,54 1,63 1,97 | \* | X3| = | 0,85| |

Решение:

|  |
| --- |
| import numpy as np  for i in **range**(1,6):  file = "SLAU" + str(i) + ".csv"  m = np.genfromtxt(file, delimiter=';')  A = np.genfromtxt(file, delimiter=';', usecols=(**range**(**len**(m-1))))  B = np.genfromtxt(file, delimiter=';', usecols=(**len**(m)))  slau = np.linalg.solve(A, B)  **print**("ответ к задаче " + str(i), slau) |

Результат:

|  |
| --- |
| ответ к задаче 1 [ 3.70558376 -4.5 ]  ответ к задаче 2 [ 0.67204028 -2.00177278 1.6178842 -0.96592738 0.69810436]  ответ к задаче 3 [ 0.51790645 -0.74739316 0.25259695 -0.10339057 0.61794625]  ответ к задаче 4 [-2.00669117 -2.59740997 -4.46982499 7.03151449]  ответ к задаче 5 [ 0.96159761 0.35158357 -0.61113784] |

# ЗАКЛЮЧЕНИЕ

Python-разработчик умеет автоматизировать задачи «здесь и сейчас». Знание Python позволяет создавать как совсем простые вещи вроде написания мелких утилит, так и сложные, например, проводить научные вычисления и работать с большими данными.

Что может Python-разработчик:

* Создавать интерактивные веб-интерфейсы, применяя все возможности современных браузеров для своего проекта
* Описывать алгоритмы для автоматизации рутинных задач. Оптимизировать процессы и экономить время для решения стандартных задач
* Обрабатывать большие данные, анализировать и систематизировать их

Это всё звучит достаточно неплохо, и многое говорит о настоящей значимости языка программирования Python в мире IT.

# СПИСОК ЛИТЕРАТУРЫ

1. Programming Python (4th Edition), Mark Lutz
2. Learning Python, Mark Lutz (5th Edition)
3. python.org