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 – 380305.2022.10. ПЗ КР

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

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

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

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

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_/\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_/ Попова Э.А./\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_/

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

Работа защищена с оценкой

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

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

Челябинск 2020

# АННОТАЦИЯ

Попова Э.А.

Программирование на языке Python (Курс молодого бойца)

Челябинск: ЮУрГУ, ЭУ-120, 2020

Работа состоит из трёх частей:

* В первой части говориться о методах сортировках
* Во второй части говориться о библиотеке Numpy
* В третьей части говориться немного о линейной алгебре

[**ВВЕДЕНИЕ** 4](#_Toc11621898)

[**ГЛАВА 1. Алгоритмы сортировки** 5](#_Toc11621899)

[**1.1 Сортировка выбором (Selection sort)** 5](#_Toc11621900)

[**1.2 Сортировка вставками (Сортировка включениями) (Insertion sort)** 6](#_Toc11621901)

[**1.3 Сортировка “Методом Пузырька” (Обменная сортировка) (Bubble sort)** 7](#_Toc11621902)

[**1.4 Анализ алгоритмов сортировки** 8](#_Toc11621903)

[**ГЛАВА 2. Библиотека NumPy.** 12](#_Toc11621904)

[**2.1 Что такое NumPy?** 12](#_Toc11621905)

[**2.2 NumPy на практике (31 задача)** 13](#_Toc11621906)

[**ГЛАВА 3. Решение СЛАУ методом Гаусса на Python с помощью NumPy (5 персональных задач)** 18](#_Toc11621907)

**ЗАКЛЮЧЕНИЕ**

**ВВЕДЕНИЕ**

На данный момент один из самых востребованных языков программирования является Python. На нём можно делать почти всё: игры, веб-приложения, бизнесс-приложения, анализировать данные и т.д., без ощутимых проблем.

Более того, порог вхождения низкий, а код во многом лаконичный и понятный даже тому, кто никогда на нём не писал. За счёт простоты кода, дальнейшее сопровождение программ, написанных на Python, становится легче и приятнее по сравнению с Java или C++. А с точки зрения бизнеса это влечёт за собой сокращение расходов и увеличение производительности труда сотрудников.

Далее будут раскрыты основные возможности языка, и решено пару задач с помощью пакета Numpy.

**ГЛАВА 1. Алгоритмы сортировки**

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.

## 1.1 Сортировка выбором (Selection sort)

Этот алгоритм сегментирует список на две части: отсортированную и неотсортированную. Наименьший элемент удаляется из второго списка и добавляется в первый.

На практике не нужно создавать новый список для отсортированных элементов. В качестве него используется крайняя левая часть списка. Находится наименьший элемент и меняется с первым местами.

Теперь, когда нам известно, что первый элемент списка отсортирован, находим наименьший элемент из оставшихся и меняем местами со вторым. Повторяем это до тех пор, пока не останется последний элемент в списке.

Код:

def select\_sort(array):

for i in range(len(array)):

indxMin = i

for j in range(i+1, len(array)):

if array[j] < array[indxMin]:

indxMin = j

tmp = array[indxMin]

array[indxMin] = array[i]

array[i] = tmp

return a

## 1.2 Сортировка вставками (Сортировка включениями) (Insertion sort)

Как и сортировка выборкой, этот алгоритм сегментирует список на две части: отсортированную и неотсортированную. Алгоритм перебирает второй сегмент и вставляет текущий элемент в правильную позицию первого сегмента.

Предполагается, что первый элемент списка отсортирован. Переходим к следующему элементу, обозначим его х. Если х больше первого, оставляем его на своём месте. Если он меньше, копируем его на вторую позицию, а х устанавливаем, как первый элемент.

Переходя к другим элементам несортированного сегмента, перемещаем более крупные элементы в отсортированном сегменте вверх по списку, пока не встретим элемент меньше x или не дойдём до конца списка. В первом случае x помещается на правильную позицию.

Код:

def insert\_sort(array):

for i in range(len(array)):

v = array[i]

j = i

while (array[j-1] > v) and (j > 0):

array[j] = array[j-1]

j = j - 1

array[j] = v

print array

return array

## 1.3 Сортировка “Методом Пузырька” (Обменная сортировка) (Bubble sort)

Этот простой алгоритм выполняет итерации по списку, сравнивая элементы попарно и меняя их местами, пока более крупные элементы не «всплывут» в начало списка, а более мелкие не останутся на «дне».

Сначала сравниваются первые два элемента списка. Если первый элемент больше, они меняются местами. Если они уже в нужном порядке, оставляем их как есть. Затем переходим к следующей паре элементов, сравниваем их значения и меняем местами при необходимости. Этот процесс продолжается до последней пары элементов в списке.

При достижении конца списка процесс повторяется заново для каждого элемента. Это крайне неэффективно, если в массиве нужно сделать, например, только один обмен. Алгоритм повторяется n² раз, даже если список уже отсортирован.

Код:

def bubble\_sort(array):

a = array

for i in range(len(a),0,-1):

for j in range(1, i):

if a[j-1] > a[j]:

tmp = a[j-1]

a[j-1] = a[j]

a[j] = tmp

print a

return a

## 1.4 Анализ алгоритмов сортировки

В этой части представлен код программы, определяющей эффективность трех алгоритмов методов сортировки (вставками (insertion), обменная (bubble), выбором (selection)).

Код:

import select\_sort

import bubble\_sort

import insert\_sort

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\_sort.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\_sort.insert(insert\_arr, DIM)  
CTotal[1] = count[0]  
MTotal[1] = count[1]  
count = [0, 0]  
count = bubble\_sort.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\_sort.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\_sort.insert(insert\_arr, DIM)  
CTotal[1] = count[0]  
MTotal[1] = count[1]  
count = [0, 0]  
count = bubble\_sort.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\_sort.select(select\_arr, DIM)  
 CTotal[0] += count[0]  
 MTotal[0] += count[1]  
 count = [0, 0]  
 count = insert\_sort.insert(insert\_arr, DIM)  
 CTotal[1] += count[0]  
 MTotal[1] += count[1]  
 count = [0, 0]  
 count = bubble\_sort.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)

myfile.write("\nСЛУЧАЙНАЯ РЕАЛИЗАЦИЯ:\n")

wr\_str = "Проведено экспериментов: " + str(NUM) + "\n"

myfile.write(wr\_str)

wr\_str = "Размер массива: " + str(DIM) + "\n"

myfile.write(wr\_str)

wr\_str = "Сравнений: " + str(CTotal[0]/NUM) + " " + str(CTotal[1]/NUM) + " " + str(CTotal[2]/NUM) + "\n"

myfile.write(wr\_str)

wr\_str = "Перестановок: " + str(MTotal[0]/NUM) + " " + str(MTotal[1]/NUM) + " " + str(MTotal[2]/NUM) + "\n"

myfile.write(wr\_str)

myfile.close()

Результат:

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

Таким образом, можно сделать вывод о том, что самым эффективным методом сортировки для массива является метод выбора (selection).

# **ГЛАВА 2. Библиотека NumPy.**

## 2.1 Что такое NumPy?

NumPy — библиотека с открытым исходным кодом для языка программирования Python.

Возможности:

* поддержка многомерных массивов (включая матрицы);
* поддержка высокоуровневых математических функций, предназначенных для работы с многомерными массивами.

Математические алгоритмы, реализованные на интерпретируемых языках (например, Python), часто работают гораздо медленнее тех же алгоритмов, реализованных на компилируемых языках (например, Фортран, Си, Java). Библиотека NumPy предоставляет реализации вычислительных алгоритмов (в виде функций и операторов), оптимизированные для работы с многомерными массивами. В результате любой алгоритм, который может быть выражен в виде последовательности операций над массивами (матрицами) и реализованный с использованием NumPy, работает так же быстро, как эквивалентный код, выполняемый в MATLAB.

NumPy можно рассматривать как свободную альтернативу MATLAB. Язык программирования MATLAB внешне напоминает NumPy: оба они интерпретируемые, оба позволяют выполнять операции над массивами (матрицами), а не над скалярами. Преимущество MATLAB в наличии большого количества пакетов («тулбоксов»), например, Simulink. Для NumPy тоже существуют подобные «пакеты», например, библиотека SciPy предоставляет больше MATLAB-подобной функциональности, библиотека Matplotlib позволяет создавать графики в стиле MATLAB. И MATLAB, и NumPy для решения основных задач линейной алгебры используют код, основанный на коде библиотеки LAPACK.

## 2.2 NumPy на практике (31 задача)

Для начала понимания, как работать с матрицами, решим следующие задачи, количеством 31(Приведу пример решения первых 5-ти задач):

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.

Код:

import numpy as np

N = 3

M = 5

A = np.random.randint(0, 100, (N, M))

print("Матрица:\n" + str(A))

sum = A.sum(axis=0)

i = sum.argmax(axis=0)

max = A.max(axis=0)

max = max[i]

print("Наибольшее значение: " + str(max))

Результат:

Матрица:

[[92 29 87 61 96]

[60 1 51 67 39]

[56 0 28 65 7]]

Наибольшее значение: 92

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

Код:

import numpy as np

N = 3

M = 5

A = np.random.randint(0, 100, (N, M))

print("Матрица:\n" + str(A))

Average = A.mean(axis=1)

i = Average.argmax(axis=0)

max = Average.max(axis=0)

print("Наибольшее среднее значение: " + str(max))

Результат:

Матрица:

[[47 79 55 76 85]

[90 39 16 35 56]

[35 93 62 57 19]]

Наибольшее среднее значение: 68.4

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.

Код:

import numpy as np

N = 3

M = 5

A = np.random.randint(0, 100, (N, M))

print("Матрица:\n" + str(A))

sum = A.sum(axis=0)

i = sum.argmin(axis=0)

min = A.min(axis=0)

min = min[i]

print("Наименьшее значение: " + str(min))

Результат:

Матрица:

[[64 88 42 47 53]

[75 27 35 71 52]

[35 99 29 50 74]]

Наименьшее значение: 29

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

Код:

import numpy as np

N = 3

M = 5

A = np.random.randint(0, 100, (N, M))

print("Матрица:\n" + str(A))

Average = A.mean(axis=1)

i = Average.argmin(axis=0)

min = Average.min(axis=0)

print("Наименьшее среднее значение: " + str(min))

Результат:

Матрица:

[[57 70 22 65 71]

[43 56 66 89 51]

[62 16 66 11 14]]

Наименьшее среднее значение: 33.8

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.

Код:

import numpy as np

N = 3

M = 5

A = np.random.randint(0, 100, (N, M))

print("Матрица:\n" + str(A))

Average\_line = A.mean(axis=1)

Average\_column = A.mean(axis=0)

Average\_line = Average\_line[: , np.newaxis]

A = np.hstack((A, Average\_line))

Average\_column = np.hstack((Average\_column, [0.]))

A = np.vstack((A, Average\_column))

print("Новая матрица:\n" + str(A))

Результат:

Матрица:

[[13 87 69 20 39]

[23 71 15 1 91]

[38 85 62 72 20]]

Новая матрица:

[[ 13. 87. 69. 20. 39. 45.6 ]

[ 23. 71. 15. 1. 91. 40.2 ]

[ 38. 85. 62. 72. 20. 55.4 ]

[ 24.66666667 81. 48.66666667 31. 50. 0. ]]

На основе решения 31 задачи, можно сделать вывод, что библиотека NumPy хорошая альтернатива MatLab и отлично подходит для работы с математическими и числовыми операциями, которые используются с большими матрицами и массивами.

**ГЛАВА 3. Решение СЛАУ методом Гаусса на Python с помощью NumPy (5 персональных задач)**

Метод Гаусса — классический метод решения системы линейных алгебраических уравнений (СЛАУ). Назван в честь немецкого математика Карла Фридриха Гаусса. Это метод последовательного исключения переменных, когда с помощью элементарных преобразований система уравнений приводится к равносильной системе треугольного вида, из которой последовательно, начиная с последних (по номеру), находятся все переменные системы.

Алгоритм решения СЛАУ методом Гаусса подразделяется на два этапа.

На первом этапе осуществляется так называемый прямой ход, когда путём элементарных преобразований над строками систему приводят к ступенчатой или треугольной форме, либо устанавливают, что система несовместна. А именно, среди элементов первого столбца матрицы выбирают ненулевой, перемещают его на крайнее верхнее положение перестановкой строк и вычитают получившуюся после перестановки первую строку из остальных строк, домножив её на величину, равную отношению первого элемента каждой из этих строк к первому элементу первой строки, обнуляя тем самым столбец под ним. После того, как указанные преобразования были совершены, первую строку и первый столбец мысленно вычёркивают и продолжают пока не останется матрица нулевого размера. Если на какой-то из итераций среди элементов первого столбца не нашёлся ненулевой, то переходят к следующему столбцу и проделывают аналогичную операцию.

На втором этапе осуществляется так называемый обратный ход, суть которого заключается в том, чтобы выразить все получившиеся базисные переменные через небазисные и построить фундаментальную систему решений, либо, если все переменные являются базисными, то выразить в численном виде единственное решение системы линейных уравнений. Эта процедура начинается с последнего уравнения, из которого выражают соответствующую базисную переменную (а она там всего одна) и подставляют в предыдущие уравнения, и так далее, поднимаясь по «ступенькам» наверх. Каждой строчке соответствует ровно одна базисная переменная, поэтому на каждом шаге, кроме последнего (самого верхнего), ситуация в точности повторяет случай последней строки.

**Вариант 10**

Задача №1

| 1.03 1.77 1.08 1.62 1.27 1.69 | \* | X1| = | 0.41|

| 1.40 1.35 1.32 1.83 1.65 1.75 | \* | X2| = | 0.44|

| 1.07 1.27 1.17 1.00 1.92 1.82 | \* | X3| = | 0.05|

| 1.27 1.34 1.52 1.80 1.12 1.84 | \* | X4| = | 0.31|

| 1.87 1.49 1.30 1.31 1.46 1.75 | \* | X5| = | 0.41|

| 1.16 1.58 1.52 1.92 1.26 1.97 | \* | X6| = | 0.16|

Задача №2

| 1.30 1.50 | \* | X1| = | 0.54|

| 1.76 1.24 | \* | X2| = | 0.66|

Задача №3

| 1.46 1.34 1.41 1.90 1.33 1.27 | \* | X1| = | 0.00|

| 1.84 1.76 1.28 1.86 1.25 1.34 | \* | X2| = | 0.23|

| 1.42 1.64 1.59 1.49 1.77 1.56 | \* | X3| = | 0.95|

| 1.07 1.56 1.46 1.28 1.03 1.43 | \* | X4| = | 0.13|

| 1.17 1.73 1.95 1.92 1.96 1.46 | \* | X5| = | 0.19|

| 1.76 1.65 1.50 1.35 1.12 1.57 | \* | X6| = | 0.72|

Задача №4

| 1.02 1.09 1.10 1.10 1.96 1.10 | \* | X1| = | 0.11|

| 1.05 1.88 1.28 1.43 1.78 1.85 | \* | X2| = | 0.07|

| 1.90 1.82 1.62 1.52 1.73 1.89 | \* | X3| = | 0.21|

| 1.71 1.14 1.53 1.23 1.42 1.87 | \* | X4| = | 0.89|

| 1.50 1.23 1.92 1.62 1.52 1.49 | \* | X5| = | 0.40|

| 1.41 1.78 1.28 1.30 1.48 1.68 | \* | X6| = | 0.41|

Задача №5

| 1.39 1.80 1.85 1.35 1.73 1.42 | \* | X1| = | 0.71|

| 1.75 1.30 1.41 1.95 1.54 1.94 | \* | X2| = | 0.21|

| 1.29 1.87 1.36 1.74 1.03 1.00 | \* | X3| = | 0.50|

| 1.06 1.17 1.01 1.13 1.91 1.21 | \* | X4| = | 0.14|

| 1.05 1.70 1.95 1.95 1.45 1.99 | \* | X5| = | 0.68|

| 1.25 1.68 1.29 1.43 1.69 1.77 | \* | X6| = | 0.56|

**Код:**

import numpy as np

file = open('nympy-gauss-slv.csv', 'wb+')

file.truncate()

for i in range(1, 6):

task\_file = "task\_" + str(i) + ".csv"

m = np.genfromtxt(task\_file, delimiter=';')

A = np.genfromtxt(task\_file, delimiter=';', usecols=(range(len(m-1))))

B = np.genfromtxt(task\_file, delimiter=';', usecols=(len(m)))

print("\nМатрица:")

print(m)

slv = np.linalg.solve(A, B)

print("\n Решение системы " + str(i) + " ", slv)

np.savetxt(file, np.array([slv]), delimiter=',')

file.close()

**Результат:**

Матрица №1:

[[1.03 1.77 1.08 1.62 1.27 1.69 0.41 nan]

[1.4 1.35 1.32 1.83 1.65 1.75 0.44 nan]

[1.07 1.27 1.17 1. 1.92 1.82 0.05 nan]

[1.27 1.34 1.52 1.8 1.12 1.84 0.31 nan]

[1.87 1.49 1.3 1.31 1.46 1.75 0.41 nan]

[1.16 1.58 1.52 1.92 1.26 1.97 0.16 nan]]

Решение:

[8.99971813 -22.46963345 -53.59254537 9.00452064 -13.13884729

53.78136281]

Матрица №2:

[[1.3 1.5 0.54 nan]

[1.76 1.24 0.66 nan]]

Решение:

[0.31167315 0.08988337]

Матрица №3:

[[1.46 1.34 1.41 1.9 1.33 1.27 0. nan]

[1.84 1.76 1.28 1.86 1.25 1.34 0.23 nan]

[1.42 1.64 1.59 1.49 1.77 1.56 0.95 nan]

[1.07 1.56 1.46 1.28 1.03 1.43 0.13 nan]

[1.17 1.73 1.95 1.92 1.96 1.46 0.19 nan]

[1.76 1.65 1.5 1.35 1.12 1.57 0.72 nan]]

Решение:

[0.78007425 -0.30513963 -0.44769356 -1.21730619 1.09061789 0.60124927]

Матрица №4:

[[1.02 1.09 1.1 1.1 1.96 1.1 0.11 nan]

[1.05 1.88 1.28 1.43 1.78 1.85 0.07 nan]

[1.9 1.82 1.62 1.52 1.73 1.89 0.21 nan]

[1.71 1.14 1.53 1.23 1.42 1.87 0.89 nan]

[1.5 1.23 1.92 1.62 1.52 1.49 0.4 nan]

[1.41 1.78 1.28 1.3 1.48 1.68 0.41 nan]]

Решение:

[-2.38267858 4.05225305 11.24228091 -15.04186524 1.1868965

-0.02127095]

Матрица №5:

[[1.39 1.8 1.85 1.35 1.73 1.42 0.71 nan]

[1.75 1.3 1.95 1.95 1.54 1.94 0.21 nan]

[1.29 1.87 1.36 1.74 1.03 1. 0.5 nan]

[1.06 1.17 1.01 1.13 1.91 1.21 0.14 nan]

[1.05 1.7 1.95 1.95 1.45 1.99 0.68 nan]

[1.25 1.68 1.29 1.43 1.69 1.77 0.56 nan]]

Решение:

[-0.24860265 0.65708781 0.29652515 -0.47899696 -0.22640873 0.25532473]

**ЗАКЛЮЧЕНИЕ**

Python невероятно эффективен: программы делают больше, чем многие другие языки, за меньшее количество кодов. Синтаксис Python также позволяет писать «чистый» код. Ваш код будет легко читать, у вас будет меньше проблем с отладкой и расширением программ по сравнению с другими языками. Python используется для различных целей: создания игр, создания веб-приложений, решения бизнес-задач и разработки внутренних инструментов для всех видов интересных проектов. Питон также широко используется в науке для теоретических исследований и решения проблем.

Тем не менее, одной из наиболее важных причин использования Python является сообщество Python, которое состоит из невероятно разнообразных и поддерживающих людей. Сообщество играет чрезвычайно важную роль в программировании, потому что программирование не является чисто индивидуальным вопросом. Дружелюбное и дружелюбное сообщество поможет вам решить проблемы, а сообщество Python готово помочь людям, чей Python является первым языком программирования.