

**МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное
образовательное учреждение высшего образования
«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»**

**Кафедра инфокоммуникаций
Отчет по лабораторной работе № 3.2
«Основы работы с библиотекой NumPy»
по дисциплине «Основы программной инженерии»**

Выполнил студент группы

ПИЖ-б-о-21-1

Зиберов Александр

« » марта 2023 г.

Подпись студента _____

Работа защищена

« » _____ 20__ г.

Проверил Воронкин Р.А. _____
(подпись)

Ставрополь, 2023

Цель работы:

Исследовать базовые возможности библиотеки NumPy языка программирования Python.

Выполнение работы:

Создать общедоступный репозиторий на GitHub, в котором будет использована лицензия MIT, рисунок 1.

Ссылка: https://github.com/afk552/trolab_2

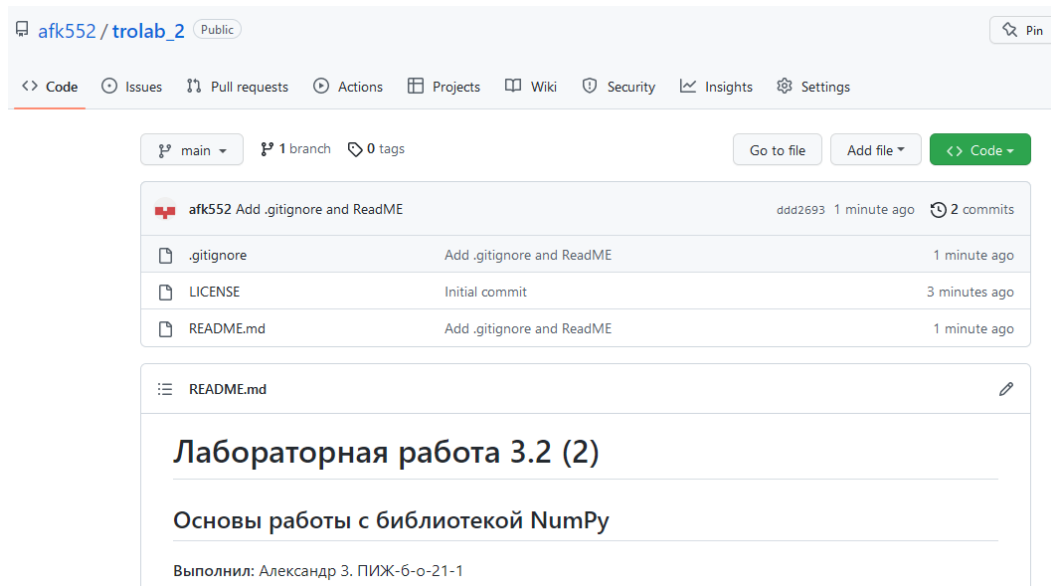


Рисунок 1 – Удаленный репозиторий на GitHub

Дополните файл .gitignore необходимыми правилами для работы с IDE PyCharm, рисунок 2.

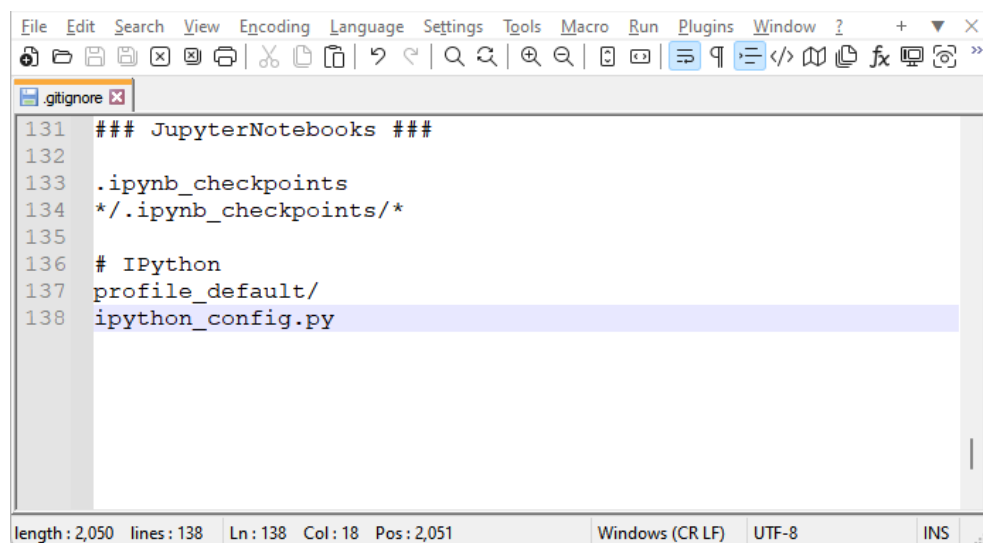


Рисунок 2 – Окно блокнота

Организируйте свой репозиторий в соответствие с моделью ветвления git-flow, рисунок 3.

```
C:\git\trolab_2>git branch
* develop
  main

C:\git\trolab_2>
```

Рисунок 3 – Окно командной строки

Проработать примеры лабораторной работы.

```
In [1]: import numpy as np
m = np.matrix('1 2 3 4; 5 6 7 8; 9 1 5 7')
print(m)

print(m[1, 0])
print(m[1, :])
print(m[:, 2])
print(m[1, 2:])
print(m[0:2, 1])
print(m[0:2, 1:3])

cols = [0, 1, 3]
m[:, cols]

[[1 2 3 4]
 [5 6 7 8]
 [9 1 5 7]]
5
[[5 6 7 8]]
[[3]
 [7]
 [5]]
[[7 8]]
[[2]
 [6]]
[[2 3]
 [6 7]]

Out[1]: matrix([[1, 2, 4],
               [5, 6, 8],
               [9, 1, 7]])
```

Рисунок 4 – Пример индексации

```
In [2]: m = np.matrix('1 2 3 4; 5 6 7 8; 9 1 5 7')
print(m)
type(m)

n = np.array(m)
type(n)

m.shape

m.max()
np.max(m)

m.max(axis=1)
m.max(axis=0)

m.mean()
m.sum(axis=0)

[[1 2 3 4]
 [5 6 7 8]
 [9 1 5 7]]

Out[2]: matrix([[15,  9, 15, 19]])

In [3]: nums = np.array([1, 2, 3, 4, 5, 6, 7, 8, 9, 10])
letters = np.array(['a', 'b', 'c', 'd', 'a', 'e', 'b'])

less_than_5 = nums < 5
print(less_than_5)

pos_a = letters == 'a'
pos_a

nums[less_than_5]

[ True  True  True  True False False False False False False]

Out[3]: array([1, 2, 3, 4])
```

Рисунок 5 – Примеры функций numpy

```
In [4]: m = np.matrix('1 2 3 4; 5 6 7 8; 9 1 5 7')
print(m)
mod_m = np.logical_and(m>=3, m<=7)
print(mod_m)

m[mod_m]

[[1 2 3 4]
 [5 6 7 8]
 [9 1 5 7]]
[[False False  True  True]
 [ True  True  True False]
 [False False  True  True]]

Out[4]: matrix([[3, 4, 5, 6, 7, 5, 7]])

In [5]: nums = np.array([1, 2, 3, 4, 5, 6, 7, 8, 9, 10])
nums[nums < 5]

Out[5]: array([1, 2, 3, 4])

In [6]: nums[nums < 5] = 10
print(nums)

[10 10 10 10  5  6  7  8  9 10]

In [7]: m[m > 7] = 25
print(m)

[[ 1  2  3  4]
 [ 5  6  7 25]
 [25  1  5  7]]
```

Рисунок 6 – Примеры логических операций

Решить задания в ноутбуках, выполненных преподавателем.

Задание 1

Задания из основного ноутбука

```
In [1]: # подключение модуля numpy под именем np
import numpy as np
```

```
In [2]: # основная структура данных - массив
a = np.array([1, 2, 3, 4, 5])
b = np.array([0.1, 0.2, 0.3, 0.4, 0.5])

print("a =", a)
print("b =", b)

a = [1 2 3 4 5]
b = [0.1 0.2 0.3 0.4 0.5]
```

Создайте массив с 5 любыми числами:

```
In [3]: m = np.matrix('20 74 43 97 35')
print(m)

[[20 74 43 97 35]]
```

Арифметические операции, в отличие от операций над списками, применяются поэлементно:

```
In [4]: list1 = [1, 2, 3]
array1 = np.array([1, 2, 3])

print("list1:", list1)
print('\tlist1 * 3:', list1 * 3)
print('\tlist1 + [1]:', list1 + [1])

print('array1:', array1)
print('\tarray1 * 3:', array1 * 3)
print('\tarray1 + 1:', array1 + 1)

list1: [1, 2, 3]
list1 * 3: [1, 2, 3, 1, 2, 3, 1, 2, 3]
list1 + [1]: [1, 2, 3, 1]
array1: [1 2 3]
array1 * 3: [3 6 9]
array1 + 1: [2 3 4]
```

Рисунок 7 – Решение заданий 1

Создайте массив из 5 чисел. Возведите каждый элемент массива в степень 3

```
In [5]: m = np.matrix('20 74 43 97 35')
m_array = np.array(m)
print(m_array ** 3)

[[ 8000 405224 79507 912673 42875]]
```

Если в операции участвуют 2 массива (по умолчанию – одинакового размера), операции считаются для соответствующих пар:

```
In [6]: print("a + b =", a + b)
print("a * b =", a * b)

a + b = [1.1 2.2 3.3 4.4 5.5]
a * b = [0.1 0.4 0.9 1.6 2.5]
```

```
In [7]: # вот это разность
print("a - b =", a - b)

# вот это деление
print("a / b =", a / b)

# вот это целочисленное деление
print("a // b =", a // b)

# вот это квадрат
print("a ** 2 =", a ** 2)

a - b = [0.9 1.8 2.7 3.6 4.5]
a / b = [10. 10. 10. 10. 10.]
a // b = [ 9.  9. 10.  9. 10.]
a ** 2 = [ 1  4  9 16 25]
```

Создайте два массива одинаковой длины. Выведите массив, полученный делением одного массива на другой.

```
In [8]: m1 = np.matrix('24 12 54 76 44 88')
m2 = np.matrix('76 44 42 54 12 6')

print(m1 / m2)

[[ 0.31578947  0.27272727  1.28571429  1.40740741  3.66666667 14.66666667]]
```

Рисунок 8 – Решение заданий 2

Л — логика

К элементам массива можно применять логические операции.

Возвращаемое значение -- массив, содержащий результаты вычислений для каждого элемента (True -- "да" или False -- "нет").

```
In [9]: print("a =", a)
print("\ta > 1: ", a > 1)
print("\nb =", b)
print("\tb < 0.5: ", b < 0.5)

print("\nОдновременная проверка условий:")
print("\t(a > 1) & (b < 0.5): ", (a>1) & (b < 0.5))
print("\tA вот это проверяет, что a > 1 ИЛИ b < 0.5: ", (a > 1) | (b < 0.5))

a = [1 2 3 4 5]
a > 1: [False True True True True]

b = [0.1 0.2 0.3 0.4 0.5]
b < 0.5: [ True True True True False]

Одновременная проверка условий:
(a > 1) & (b < 0.5): [False True True True False]
A вот это проверяет, что a > 1 ИЛИ b < 0.5: [ True True True True True]
```

Создайте 2 массива из 5 элементов. Проверьте условие "Элементы первого массива меньше 6, элементы второго массива делятся на 3"

```
In [10]: n1 = np.matrix('1 3 2 97 5')
n2 = np.matrix('90 56 23 66 33')

print((n1 < 6) & (n2%3==0))

[[ True False False False  True]]
```

Теперь проверьте условие "Элементы первого массива делятся на 2 или элементы второго массива больше 2"

```
In [11]: n1 = np.matrix('1 3 2 97 5')
n2 = np.matrix('90 56 23 66 33')

print((n1 % 2 == 0) | (n2 > 2))

n1 = np.matrix('2 4 6 8 10')
n2 = np.matrix('2 2 2 2 2')

print((n1 % 2 == 0) | (n2 > 2))

[[ True True True True True]]
[[ True True True True True]]
```

Рисунок 9 – Решение заданий 3

Зачем это нужно? Чтобы выбирать элементы массива, удовлетворяющие какому-нибудь условию:

```
In [12]: print("a =", a)
print("a > 2:", a > 2)
# индексация - выбираем элементы из массива в тех позициях, где True
print("a[a > 2]:", a[a > 2])

a = [1 2 3 4 5]
a > 2: [False False True True True]
a[a > 2]: [3 4 5]
```

Создайте массив с элементами от 1 до 20. Выведите все элементы, которые больше 5 и не делятся на 2

Подсказка: создать массив можно с помощью функции np.arange(), действие которой аналогично функции range, которую вы уже знаете.

```
In [13]: num_arr = np.arange(1, 60, 3)
print(num_arr)
type(num_arr)
# print(num_arr[num_arr > 5])
condition = (num_arr > 5) & (num_arr % 2 != 0)
num_arr[condition]

[ 1  4  7 10 13 16 19 22 25 28 31 34 37 40 43 46 49 52 55 58]

Out[13]: array([ 7, 13, 19, 25, 31, 37, 43, 49, 55])
```

А ещё NumPy умеет...

Все операции NumPy оптимизированы для быстрых вычислений над целыми массивами чисел и в методах np.array реализовано множество функций, которые могут вам понадобиться.

```
In [14]: # теперь можно считать средний размер котиков в одну строку!
print("np.mean(a) =", np.mean(a))
# минимальный элемент
print("np.min(a) =", np.min(a))
# индекс минимального элемента
print("np.argmin(a) =", np.argmin(a))
# вывести значения массива без дубликатов
print("np.unique(['male', 'male', 'female', 'female', 'male']) =", np.unique(['male', 'male', 'female', 'female', 'male']))

# и ещё много всяких методов
# Google в помощь

np.mean(a) = 3.0
np.min(a) = 1
np.argmin(a) = 0
np.unique(['male', 'male', 'female', 'female', 'male']) = ['female' 'male']
```

Рисунок 10 – Решение заданий 4

Пора еще немного потренироваться с NumPy.

Выполните операции, перечисленные ниже:

```
In [15]: print(a)
print(b)

print("Разность между a и b:", a - b
      )
print("Квадраты элементов b:", b ** 2
      )
print("Половины произведений элементов массивов a и b:", (a * b) / 2
      )

print()
print("Максимальный элемент b:", max(b)
      )
print("Сумма элементов массива b:", sum(b)
      )
print("Индекс максимального элемента b:", b.argmax()
      )

[1 2 3 4 5]
[0.1 0.2 0.3 0.4 0.5]
Разность между a и b: [0.9 1.8 2.7 3.6 4.5]
Квадраты элементов b: [0.01 0.04 0.09 0.16 0.25]
Половины произведений элементов массивов a и b: [0.05 0.2 0.45 0.8 1.25]

Максимальный элемент b: 0.5
Сумма элементов массива b: 1.5
Индекс максимального элемента b: 4
```

Задайте два массива: [5, 2, 3, 12, 4, 5] и ['f', 'o', 'o', 'b', 'a', 'r']

Выведите буквы из второго массива, индексы которых соответствуют индексам чисел из первого массива, которые больше 1, меньше 5 и делятся на 2

```
In [16]: arr = np.array([5, 2, 3, 12, 4, 5])
letters = np.array(['f', 'o', 'o', 'b', 'a', 'r'])
print(letters[np.logical_and(arr > 1, arr < 5, arr % 2 == 0)])

['o' 'o' 'a']
```

Рисунок 11 – Решение заданий 5

Задание 2

Задания из ноутбука с домашним заданием

Лабораторная работа 3.2. Домашнее задание

Задание №1

Создайте два массива: в первом должны быть четные числа от 2 до 12 включительно, а в другом числа 7, 11, 15, 18, 23, 29.

1. Сложите массивы и возведите элементы получившегося массива в квадрат:

```
In [13]: import numpy as np

In [14]: #n1 = np.arange(2, 13, 2)
n1 = np.array([2, 4, 6, 8, 10, 12])
n2 = np.array([7, 11, 15, 18, 23, 29])

print((n1 + n2) ** 2)

[ 81 225 441 676 1089 1681]
```

2. Выведите все элементы из первого массива, индексы которых соответствуют индексам тех элементов второго массива, которые больше 12 и дают остаток 3 при делении на 5.

```
In [15]: condition = (n2 > 12) & (n2 % 5 == 3)
print(n1[condition])

[ 8 10]
```

3. Проверьте условие "Элементы первого массива делятся на 4, элементы второго массива меньше 14". (Подсказка: в результате должен получиться массив с True и False)

```
In [16]: print((n1 % 4 == 0) & (n2 < 14))

[False  True False False False False]
```

Рисунок 12 – Решение домашнего задания 1

Задание №2

- Найдите интересный для вас датасет. Например, можно выбрать датасет тут: <http://data.un.org/Explorer.aspx> (выбираете датасет, жмете на view data, потом download, выбирайте csv формат)
- Рассчитайте подходящие описательные статистики для признаков объектов в выбранном датасете
- Проанализируйте и прокомментируйте содержательно полученные результаты
- Все комментарии оформляйте строго в ячейках формата markdown

В выбранном наборе данных представлены данные о популярности языков программирования (в процентном значении) в период с 2004 по 2022. В качестве объектов для анализа были выбраны языки C# и Python.

```
In [5]: import csv
import numpy as np

data = []
x, y = [], []
with open("p_langs.csv", encoding='utf-8') as f:
    reader = csv.DictReader(f, dialect='excel')
    for row in reader:
        data.append(row)
    for i in data:
        x.append(float(i.get('C#')))
        y.append(float(i.get('Python')))
x_np = np.array(x)
y_np = np.array(y)

print(f"Дисперсия для C#: {np.var(x_np)}")
print(f"Дисперсия для Python: {np.var(y_np)}")
print(f"Среднее значение популярности языка C#: {np.mean(x_np)}")
print(f"Среднее значение популярности языка Python: {np.mean(y_np)}")
print(f"Среднее отклонение для C#: {np.std(x_np)}")
print(f"Среднее отклонение для Python: {np.std(y_np)}")
print(f"Коэффициент парной корреляции: {np.corrcoef(x_np, y_np)}")

print(f"Значение медианы для C#: {np.median(x_np)}")
print(f"Значение медианы для Python: {np.median(y_np)}")

q3, q1 = np.percentile(x_np, [75, 25])
iqr = q3 - q1
print(f"Межквартильный диапазон для C#: {iqr}")
q3, q1 = np.percentile(y_np, [75, 25])
iqr = q3 - q1
print(f"Межквартильный диапазон для Python: {iqr}")
```

```
Дисперсия для C#: 1.637885688914625
Дисперсия для Python: 85.98154273264382
Среднее значение популярности языка C#: 7.589241786161137
Среднее значение популярности языка Python: 12.613696682464454
Среднее отклонение для C#: 1.2797998783379338
Среднее отклонение для Python: 9.268388515184582
Коэффициент парной корреляции: [[1.         0.8851394]
 [0.8851394 1.         ]]
Значение медианы для C#: 7.2900000000000001
Значение медианы для Python: 9.17
Межквартильный диапазон для C#: 2.1899999999999994
Межквартильный диапазон для Python: 12.19
```

По полученным результатам можно сделать вывод, что Python был гораздо популярнее C#, что показывает среднее значение. Коэффициенты среднего отклонения же показывают, что Python по популярности сильно обгоняет C#, что так же заметно по дисперсиям. Коэффициенты парной корреляции показывают, что значения мало связаны между собой. Межквартильные диапазоны показали сильный размах в значениях у Python.

Рисунок 13 – Решение домашнего задания 2

Создать ноутбук, в котором выполнить решение индивидуального задания. Ноутбук должен содержать условие индивидуального задания. При решении индивидуального задания не должны быть использованы условный оператор if, а также операторы циклов while и for, а только средства библиотеки NumPy. Привести в ноутбуке обоснование принятых решений

Номер варианта индивидуального задания необходимо уточнить у преподавателя.

Задание.

Дана целочисленная квадратная матрица. Определить: сумму элементов в тех столбцах, которые не содержат отрицательных элементов; минимум среди сумм модулей элементов диагоналей, параллельных побочной диагонали матрицы.

Задание: Дана целочисленная квадратная матрица. Определить: сумму элементов в тех столбцах, которые не содержат отрицательных элементов; минимум среди сумм модулей элементов диагоналей, параллельных побочной диагонали матрицы.

```
In [1]: import numpy as np
import random
n = 5
random.seed(10)
arr = np.random.randint(-3, 30, size=(n, n))

print(arr)
summ = arr[:, (arr>=0).all(axis=0)].sum()
minn = min([np.fliplr(arr).diagonal(i).sum() for i in range(-(n-1), n)])

print("Сумма элементов в столбцах, не содержащих отрицательных элементов:", summ)
print("Минимум среди сумм модулей элементов диагоналей, параллельных побочной: ", minn)

[[18  6 -3  0 -3]
 [ 8 11 13 -3  8]
 [12  4  8 25  8]
 [ 1 24 21  4 10]
 [-1  1  8 17  5]]
Сумма элементов в столбцах, не содержащих отрицательных элементов: 46
Минимум среди сумм модулей элементов диагоналей, параллельных побочной: 5
```

In []:

Рисунок 14 – Решение индивидуального задания

Создать ноутбук, в котором выполнить решение вычислительной задачи (например, задачи из области физики, экономики, математики, статистики и т. д.), условие которой предварительно необходимо согласовать с преподавателем.

Задание.

Известно, что вклад, находящийся в банке с начала года, возрастает к концу года на определенный процент (свой для каждого банка). В начале года $\frac{3}{8}$ вклада, который составляет 800 тыс. руб., вложили в первый банк, $\frac{1}{8}$ во второй банк и оставшуюся часть вклада в третий банк. К концу года сумма этих вкладов стала равна 907 тыс. руб. Если бы первоначально $\frac{1}{8}$ вклада положили в первый банк, $\frac{4}{8}$ вклада — во второй банк, оставшуюся часть вклада — в третий банк, то к концу года сумма этих вкладов стала бы равна 894 тыс. руб. Если бы $\frac{4}{8}$ вклада вложили в первый банк, $\frac{3}{8}$ вклада — во второй банк, оставшуюся часть вклада — в третий банк, то к концу года сумма этих вкладов была бы равна 903 тыс. руб. Какой процент начисляет каждый банк?

Задача

Известно, что вклад, находящийся в банке с начала года, возрастает к концу года на определенный процент (свой для каждого банка). В начале года $\frac{3}{8}$ вклада, который составляет 800 тыс. руб., вложили в первый банк, $\frac{1}{8}$ во второй банк и оставшуюся часть вклада в третий банк. К концу года сумма этих вкладов стала равна 907 тыс. руб. Если бы первоначально $\frac{1}{8}$ вклада положили в первый банк, $\frac{4}{8}$ вклада — во второй банк, оставшуюся часть вклада — в третий банк, то к концу года сумма этих вкладов стала бы равна 894 тыс. руб. Если бы $\frac{4}{8}$ вклада вложили в первый банк, $\frac{3}{8}$ вклада — во второй банк, оставшуюся часть вклада — в третий банк, то к концу года сумма этих вкладов была бы равна 903 тыс. руб. Какой процент начисляет каждый банк?

Решение:

Введем обозначения:

Пусть x_1 - процент, начисляемый вкладчику в первом банке, тогда x_2, x_3 проценты во втором и третьем банке соответственно. Рассчитаем вклады, умножив части вклада на общую его сумму:

$$\frac{3}{8} \cdot 800 = 300, \frac{1}{8} \cdot 800 = 100, \frac{4}{8} \cdot 800 = 400$$

```
In [97]: cost = 800 # тыс.руб
cost1 = 0.375 * cost
cost2 = 0.125 * cost
cost3 = 0.5 * cost
print(f'Суммы вкладов в банки 1, 2 и 3: {cost1} тыс.руб, {cost2} тыс.руб, {cost3} тыс.руб.')
```

Суммы вкладов в банки 1, 2 и 3: 300.0 тыс.руб, 100.0 тыс.руб, 400.0 тыс.руб.

Рассчитаем сумму с начисленными процентами за год в каждом из банков по формуле: $cost_i \cdot \frac{x_i}{100}$

Начислено в первом банке: $300 \cdot \frac{x_1}{100} = 3x_1$

Начислено во втором банке: $100 \cdot \frac{x_2}{100} = x_2$

Начислено в третьем банке: $400 \cdot \frac{x_3}{100} = 4x_3$

Всего на вклад в 800 тыс. руб., сделанный в три банка (в первый — 300 тыс. руб., во второй — 100 тыс. руб., в третий — 400 тыс. руб.), было начислено за год: $907 - 800 = 107$ тыс. руб. Следовательно, можем составить систему уравнений.

Первое уравнение системы: $3x_1 + x_2 + 4x_3 = 107$

Следующие уравнения составляем аналогично:

$$x_1 + 4x_2 + 3x_3 = 94, 4x_1 + 3x_2 + x_3 = 103$$

Получим систему трех линейных уравнений:

$$\begin{cases} 3x_1 + x_2 + 4x_3 = 107 \\ x_1 + 4x_2 + 3x_3 = 94 \\ 4x_1 + 3x_2 + x_3 = 103 \end{cases}$$

Рисунок 15 – Решение задачи 1

Запишем уравнение в виде матрицы A и матрицы столбца свободных членов B:

$$\left(\begin{array}{ccc|c} 3 & 1 & 4 & 107 \\ 1 & 4 & 3 & 94 \\ 4 & 3 & 1 & 103 \end{array} \right)$$

Решим при помощи библиотеки numpy:

```
In [2]: import numpy as np
A = np.array([[3., 1., 4.], [1., 4., 3.], [4., 3., 1.]]) # Коэффициенты
B = np.array([107., 94., 103.]) # Свободные члены
result = np.linalg.solve(A, B)
print(f'Решение системы библиотечным методом: {result}')
```

Решение системы библиотечным методом: [15. 10. 13.]

Решение методом Гаусса, проверка результата:

Для начала, создаем расширенную матрицу A^* . Затем приводим её к ступенчатой матрице, у которой все ненулевые строки имеют первый элемент, равный единице. Полученную в результате преобразований систему уравнений называется обратным ходом метода Гаусса. Обратный ход может быть выполнен форме последовательного определения неизвестных, начиная с последнего.

```
In [3]: def gaussian(A, b):
# Расширенная матрица
reshaped_b = b.reshape((len(b), 1))
A = np.hstack((A, reshaped_b))

# Приведение матрицы к треугольному виду
for i, i_val in enumerate(A):
    for j in range(i + 1, len(A)):
        A[j] -= A[i] * A[j][i] / A[i][i]

# Подстановка (обратный ход)
x = np.array([0] * len(b), dtype=float)
i = len(A) - 1
while i >= 0:
    x[i] = (A[i][-1] - sum(x * A[i][0:-1])) / A[i][i]
    i -= 1
return x

print("Решение системы методом Гаусса: ", gaussian(A, B))
```

Решение системы методом Гаусса: [15. 10. 13.]

Система имеет единственное решение: $x_1 = 15, x_2 = 10, x_3 = 13$

Ответ: первый банк выплачивает 15% годовых, второй банк - 10% годовых, а третий банк - 13% годовых.

Рисунок 16 – Решение задачи 2

Вывод: В результате выполнения работы были исследованы базовые возможности библиотеки массивов NumPy языка Python.

Контрольные вопросы:

1. Каково назначение библиотеки NumPy?

numpy – это библиотека для языка программирования Python, которая предоставляет в распоряжение разработчика инструменты для эффективной работы с многомерными массивами и высокопроизводительные вычислительные алгоритмы.

2. Что такое массивы ndarray?

ndarray - это многомерный контейнер элементов одного типа и размера. Количество измерений и элементов в массиве определяется его формой, которая является кортежем из N натуральных чисел, которые определяют размеры каждого измерения.

3. Как осуществляется доступ к частям многомерного массива?

Извлечем элемент из нашей матрицы с координатами (1, 0), 1 – это номер строки, 0 – номер столбца.

`m[1, 0]`

Строка матрицы

`m[1, :]`

Столбец матрицы

`m[:, 2]`

Часть строки матрицы

Иногда возникает задача взять не все элементы строки, а только часть: рассмотрим пример, когда нам из второй строки нужно извлечь все элементы, начиная с третьего.

`m[1, 2:]`

Часть столбца матрицы

```
>>> m[0:2, 1]
```

Непрерывная часть матрицы

```
m[0:2, 1:3]
```

Произвольные столбцы / строки матрицы

```
cols = [0, 1, 3]
```

```
m[:, cols]
```

4. Как осуществляется доступ к частям многомерного массива?

Размерность массива: `m.shape`

Для расчета той или иной статистики, соответствующую функцию можно вызвать как метод объекта, с которым вы работаете. Для нашего массива это будет выглядеть так.

```
m.max()
```

Если необходимо найти максимальный элемент в каждой строке, то для этого нужно передать в качестве аргумента параметр `axis=1`.

```
m.max(axis=1)
```

Для вычисления статистики по столбцам, передайте в качестве параметра аргумент `axis=0`.

```
m.max(axis=0)
```

Функции (методы) для расчета статистик в NumPy

`argmax` Индексы элементов с максимальным значением (по осям)

`argmin` Индексы элементов с минимальным значением (по осям)

`max` Максимальные значения элементов (по осям)

`min` Минимальные значения элементов (по осям)

`mean` Средние значения элементов (по осям)

`prod` Произведение всех элементов (по осям)

`std` Стандартное отклонение (по осям)

`sum` Сумма всех элементов (по осям)

`var` Дисперсия (по осям)

5. Как выполняется выборка данных из массивов ndarray?

Boolean выражение в Numpy можно использовать для индексации, не создавая предварительно boolean массив. Получить соответствующую выборку можно, передав в качестве индекса для объекта ndarray, условное выражение. Самым замечательным в использовании boolean массивов при работе с ndarray является то, что их можно применять для построения выборок.

```
less_than_5 = nums < 5
```

```
less_than_5
```

```
array([ True,  True,  True,  True, False, False, False, False, False, False])
```

Если мы переменную less_than_5 передадим в качестве списка индексов для nums, то получим массив, в котором будут содержаться элементы из nums с индексами равными индексам True позиций массива less_than_5.