

Министерство науки и высшего образования Российской Федерации  
Федеральное государственное автономное образовательное учреждение  
высшего образования  
«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт перспективной инженерии  
Департамент цифровых, робототехнических систем и электроники

**ОТЧЕТ**  
**ПО ЛАБОРАТОРНОЙ РАБОТЕ №2**  
**дисциплины**  
**«Искусственный интеллект и машинное обучение»**  
**Вариант-3**

Выполнил:  
Пугачев Кирилл Дмитриевич  
2 курс, группа ИТС-б-о-23-1,  
11.03.02 Инфокоммуникационные  
технологии и системы связи,  
очная форма обучения

---

(подпись)

Проверила:  
Ассистент департамента цифровых,  
робототехнических систем и электроники  
Хацукова А.И

---

(подпись)

Отчет защищен с оценкой \_\_\_\_\_ Дата защиты \_\_\_\_\_

Ставрополь, 2025 г.

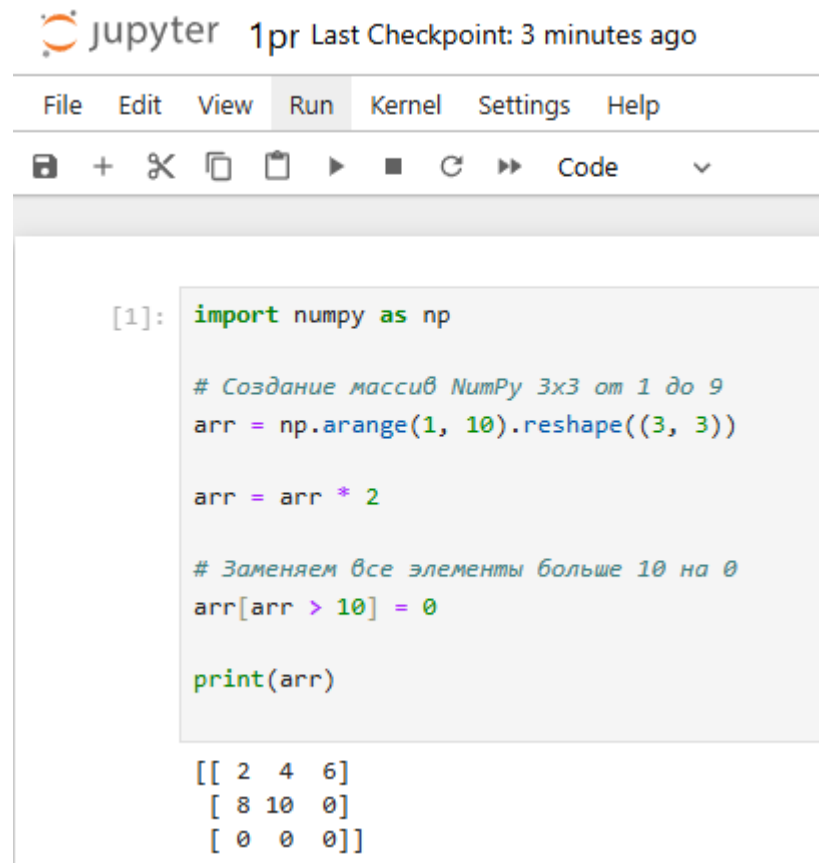
## Тема: Основы работы с библиотекой NumPy

**Цель работы:** исследовать базовые возможности библиотеки NumPy языка программирования Python.

**Ссылка на репозиторий:** <https://github.com/chillkirill/LABA2AI>

**Ход работы:**

### 1. Выполнение практических заданий.



The screenshot shows a Jupyter Notebook interface. At the top, the Jupyter logo and '1pr' are visible, along with 'Last Checkpoint: 3 minutes ago'. Below this is a menu bar with 'File', 'Edit', 'View', 'Run', 'Kernel', 'Settings', and 'Help'. Under the menu bar is a toolbar with icons for saving, adding, deleting, copying, pasting, running, and other actions. The main area contains a code cell with the following Python code:

```
[1]: import numpy as np

# Создание массив NumPy 3x3 от 1 до 9
arr = np.arange(1, 10).reshape((3, 3))

arr = arr * 2

# Заменяем все элементы больше 10 на 0
arr[arr > 10] = 0

print(arr)
```

The output of the code cell is a 3x3 NumPy array:

```
[[ 2  4  6]
 [ 8 10  0]
 [ 0  0  0]]
```

Рисунок 1. Практическое задание 1

```
[1]: import numpy as np

# Создание массива NumPy из 20 случайных целых чисел от 1 до 100
arr = np.random.randint(1, 101, 20)

print("Исходный массив:", arr)

# делятся на 5 без остатка
divisible_by_5 = arr[arr % 5 == 0]

# элементы делящиеся на 5
print("Элементы, делящиеся на 5:", divisible_by_5)

# Заменяем элементы делящиеся на 5 и на -1
arr[arr % 5 == 0] = -1

print("Обновленный массив:", arr)
```

Исходный массив: [13 45 4 11 35 67 4 35 72 3 50 18 6 89 97 36 10 98 72 84]  
 Элементы, делящиеся на 5: [45 35 35 50 10]  
 Обновленный массив: [13 -1 4 11 -1 67 4 -1 72 3 -1 18 6 89 97 36 -1 98 72 84]

Рисунок 2. Практическое задание 2

```
[1]: import numpy as np

# два массива со случайными числами от 0 до 50
arr1 = np.random.randint(0, 51, (1, 5))
arr2 = np.random.randint(0, 51, (1, 5))

print("Массив 1:", arr1)
print("Массив 2:", arr2)

# Объединение по строкам
combined_arr = np.concatenate((arr1, arr2), axis=0)


print("Объединенный массив:\n", combined_arr)

# Разделение каждый из которых содержит 5 элементов
arr1_split, arr2_split = np.split(combined_arr, 2)

print("Первый разделенный массив:", arr1_split)
print("Второй разделенный массив:", arr2_split)
```










Массив 1: [[ 8 9 14 40 21]]  
 Массив 2: [[18 47 18 50 27]]  
 Объединенный массив:  
 [[ 8 9 14 40 21]  
 [18 47 18 50 27]]  
 Первый разделенный массив: [[ 8 9 14 40 21]]  
 Второй разделенный массив: [[18 47 18 50 27]]

Рисунок 3. Практическое задание 3

 jupyter

pr4 Last Checkpoint: 2 minutes ago

FileEditViewRunKernelSettingsHelp

         Code

```
[1]: import numpy as np

# массив из 50 чисел, равномерно распределенных от -10 до 10
arr = np.linspace(-10, 10, 50)

print("Массив:", arr)

# Сумму всех элементов
total_sum = np.sum(arr)

# Сумму полож элементов
positive_elements = arr[arr > 0] # Выбираем только положительные элементы
positive_sum = np.sum(positive_elements)

# Сумма отриц элементов
negative_elements = arr[arr < 0] # здесь только отриц
negative_sum = np.sum(negative_elements)

print("Сумма всех элементов:", total_sum)
print("Сумма положительных элементов:", positive_sum)
print("Сумма отрицательных элементов:", negative_sum)
```

Массив: [-10. -9.59183673 -9.18367347 -8.7755102 -8.36734694  
-7.95918367 -7.55102041 -7.14285714 -6.73469388 -6.32653061  
-5.91836735 -5.51020408 -5.10204082 -4.69387755 -4.28571429  
-3.87755102 -3.46938776 -3.06122449 -2.65306122 -2.24489796  
-1.83673469 -1.42857143 -1.02040816 -0.6122449 -0.20408163  
0.20408163 0.6122449 1.02040816 1.42857143 1.83673469  
2.24489796 2.65306122 3.06122449 3.46938776 3.87755102  
4.28571429 4.69387755 5.10204082 5.51020408 5.91836735  
6.32653061 6.73469388 7.14285714 7.55102041 7.95918367  
8.36734694 8.7755102 9.18367347 9.59183673 10. ]

Сумма всех элементов: 7.105427357601002e-15  
Сумма положительных элементов: 127.55102040816328  
Сумма отрицательных элементов: -127.55102040816327

Рисунок 4. Практическое задание 4

```
[1]: import numpy as np

# единич матрица 4x4
identity_matrix = np.identity(4)

print("Единичная матрица:\n", identity_matrix)

# диаг матрица 4x4 с заданными диагональными элементами
diagonal_matrix = np.diag([5, 10, 15, 20])

print("\nДиагональная матрица:\n", diagonal_matrix)

# сумма всех элементов единич матрицы
identity_sum = np.sum(identity_matrix)

# сумма всех элементов диаг матрицы
diagonal_sum = np.sum(diagonal_matrix)

print("\nСумма элементов единичной матрицы:", identity_sum)
print("Сумма элементов диагональной матрицы:", diagonal_sum)

# Сравнение
if identity_sum > diagonal_sum:
    print("\nСумма элементов единичной матрицы больше суммы элементов диагональной матрицы.")
elif identity_sum < diagonal_sum:
    print("\nСумма элементов единичной матрицы меньше суммы элементов диагональной матрицы.")
else:
    print("\nСуммы элементов обеих матриц равны.")
```

Единичная матрица:

```
[[1. 0. 0. 0.]
 [0. 1. 0. 0.]
 [0. 0. 1. 0.]
 [0. 0. 0. 1.]]
```

Диагональная матрица:

```
[[ 5  0  0  0]
 [ 0 10  0  0]
 [ 0  0 15  0]
 [ 0  0  0 20]]
```

Сумма элементов единичной матрицы: 4.0

Сумма элементов диагональной матрицы: 50

Сумма элементов единичной матрицы меньше суммы элементов диагональной матрицы.

Рисунок 5. Практическое задание 5

jupyter

pr6 Last Checkpoint: 2 minutes ago

File Edit View Run Kernel Settings Help

+

✂

▶

■

↺

▶▶

Code

▼

```
[1]: import numpy as np

matrix1 = np.random.randint(1, 21, (3, 3))
matrix2 = np.random.randint(1, 21, (3, 3))

print("Матрица 1:\n", matrix1)
print("\nМатрица 2:\n", matrix2)

sum_matrix = matrix1 + matrix2

# разность
diff_matrix = matrix1 - matrix2

# поэлементное умножение
elementwise_product = matrix1 * matrix2

print("\nСумма :\n", sum_matrix)
print("\nРазность :\n", diff_matrix)
print("\nПоэлементное произведение :\n", elementwise_product)
```

Матрица 1:

```
[[19 16 14]
 [14 16 17]
 [ 4  8  7]]
```

Матрица 2:

```
[[ 4  8 15]
 [10 13 16]
 [18 16  8]]
```

Сумма :

```
[[23 24 29]
 [24 29 33]
 [22 24 15]]
```

Разность :

```
[[ 15  8 -1]
 [  4  3  1]
 [-14 -8 -1]]
```

Поэлементное произведение :

```
[[ 76 128 210]
 [140 208 272]
 [ 72 128  56]]
```

Рисунок 6. Практическое задание 6

```
[1]: import numpy as np

# 2x3, со случайными числами от 1 до 10
matrix1 = np.random.randint(1, 11, (2, 3))

# 3x2, со случайными числами от 1 до 10
matrix2 = np.random.randint(1, 11, (3, 2))

print("Матрица 1:\n", matrix1)
print("\nМатрица 2:\n", matrix2)

# умножение с оператором @
product_matrix = matrix1 @ matrix2

print("\nРезультат умножения:\n", product_matrix)
```

Матрица 1:

```
[[7 6 2]
 [5 7 2]]
```

Матрица 2:

```
[[ 4  8]
 [ 1  6]
 [10  1]]
```

Результат умножения:

```
[[54 94]
 [47 84]]
```

Рисунок 7. Практическое задание 7

```
[1]: import numpy as np
import numpy.linalg as la

# 3x3
matrix = np.random.rand(3, 3)

print("Исходная матрица:\n", matrix)

# определитель
determinant = la.det(matrix)

print("\nОпределитель матрицы:", determinant)

# Проверка, матрица вырожденная или не (определитель равен 0)если 0 то ничего не будет
if np.isclose(determinant, 0):
    print("\nМатрица вырождена, обратной матрицы не существует.")
else:
    # обратная матрицу
    inverse_matrix = la.inv(matrix)

    print("\nОбратная матрица:\n", inverse_matrix)
```

Исходная матрица:

```
[[0.1261026  0.36782876 0.2696788 ]
 [0.00192422 0.00896292 0.5537988 ]
 [0.44902583 0.83598093 0.79023471]]
```

Определитель матрицы: 0.03276915463449463

Обратная матрица:

```
[[-1.39119372e+01 -1.99043042e+00  6.14254520e+00]
 [ 7.54213473e+00 -6.54337733e-01 -2.11529857e+00]
 [-7.37270767e-02  1.82321594e+00  1.28921596e-02]]
```

Рисунок 8. Практическое задание 8



```
[1]: import numpy as np

# 4x4, со случайными целыми числами от 1 до 50
matrix = np.random.randint(1, 51, (4, 4))

print("Исходная матрица:\n", matrix)

# транспонированная (вычисление)
transposed_matrix = matrix.T

print("\nТранспонированная матрица:\n", transposed_matrix)

# след матрицы
trace = np.trace(matrix)

print("\nСлед матрицы:", trace)
```

Исходная матрица:

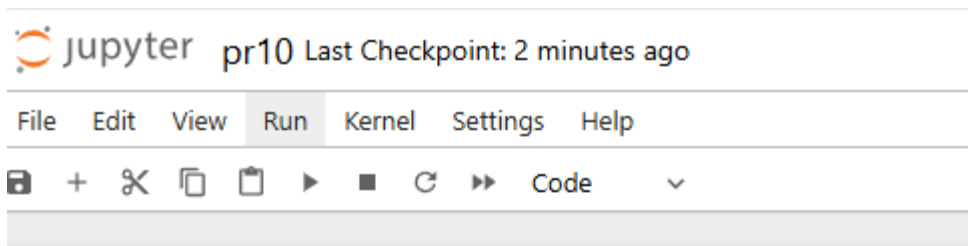
```
[[45  9 46 15]
 [48  8 20 47]
 [30 21 16 42]
 [35 34 24  3]]
```

Транспонированная матрица:

```
[[45 48 30 35]
 [ 9  8 21 34]
 [46 20 16 24]
 [15 47 42  3]]
```

След матрицы: 72

Рисунок 9. Практическое задание 9



```
[1]: import numpy as np
import numpy.linalg as la

# создал матрицу из коэф
A = np.array([[2, 3, -1],
              [4, -1, 2],
              [-3, 5, 4]])

# вектор правой части B
B = np.array([5, 6, -2])

# решение системы линейных уравнений Ax = B
try:
    x = la.solve(A, B)

    print("Решение системы уравнений:")
    print("x =", x[0])
    print("y =", x[1])
    print("z =", x[2])

except la.LinAlgError:
    print("Система не имеет решений.")

Решение системы уравнений:
x = 1.6396396396396398
y = 0.5765765765765767
z = 0.009009009009009008993
```

Рисунок 10. Практическое задание 10

3. **Маркетинговый анализ.** Три рекламные кампании дали суммарно 500 новых клиентов. Вторая кампания привлекла на 50 клиентов больше, чем первая, а третья привлекла в 1.5 раза больше клиентов, чем первая. Сколько клиентов привлекла каждая кампания?

Рисунок 11. Индивидуально задание к 3 варианту



```
# 3. Матричный метод (нахождение обратной матрицы и умножение)

A_matrix = np.array([[1, 1, 1],
                     [-1, 1, 0],
                     [-1.5, 0, 1]])
B_matrix = np.array([500, 50, 0])

A_inv = la.inv(A_matrix)
solution_matrix = A_inv @ B_matrix

print("\nРешение матричным методом:")
print("Кампания 1:", solution_matrix[0])
print("Кампания 2:", solution_matrix[1])
print("Кампания 3:", solution_matrix[2])

# Сравнение
print("\nСравнение результатов:")
print("np.linalg.solve - x: {:.2f}, y: {:.2f}, z: {:.2f}".format(solution_solve[0], solution_solve[1], solution_solve[2]))
print("Метод Крамера - x: {:.2f}, y: {:.2f}, z: {:.2f}".format(x_kramer, y_kramer, z_kramer))
print("Матричный метод - x: {:.2f}, y: {:.2f}, z: {:.2f}".format(solution_matrix[0], solution_matrix[1], solution_matrix[2]))

Решение с помощью np.linalg.solve:
Кампания 1: 128.57142857142858
Кампания 2: 178.57142857142858
Кампания 3: 192.8571428571429

Решение методом Крамера:
Кампания 1: 128.5714285714286
Кампания 2: 178.57142857142853
Кампания 3: 192.85714285714295

Решение матричным методом:
Кампания 1: 128.5714285714286
Кампания 2: 178.57142857142858
Кампания 3: 192.8571428571429

Сравнение результатов:
np.linalg.solve - x: 128.57, y: 178.57, z: 192.86
Метод Крамера - x: 128.57, y: 178.57, z: 192.86
Матричный метод - x: 128.57, y: 178.57, z: 192.86
```

Рисунок 13. Индивидуальное задание (часть 2)

## 2. Создание репозитория и работа с ним.

Создайте новый репозиторий

Репозиторий содержит все файлы проекта, включая историю изменений. У вас уже есть репозиторий проекта в другом месте? [Импортируйте репозиторий.](#)

Обязательные поля отмечены звездочкой (\*).

Владелец \*

охладите курилла

/

LABA2AI

Доступен LABA2AI

Хорошие названия репозитория должны быть короткими и запоминающимися. Нужно вдохновение? Как насчёт `curly-octo-engine` ?

Описание (необязательно)

Публичный

любой пользователь интернета может увидеть этот репозиторий. Вы выбираете, кто может совершать коммиты.

Личное

Вы сами выбираете, кто может просматривать этот репозиторий и фиксировать его в нем.

Инициализируйте этот репозиторий с помощью:

Добавьте файл README.

Здесь вы можете написать подробное описание своего проекта. [Узнайте больше о файлах README.](#)

Добавить .gitignore

шаблон gitignore: 

Отсутствует

Выберите, какие файлы не отслеживать, из списка шаблонов. [Узнайте больше об игнорировании файлов.](#)

Выберите лицензию

Лицензия: 

Отсутствует

Лицензия сообщает другим пользователям, что они могут и чего не могут делать с вашим кодом. [Узнайте больше о лицензиях.](#)

Это сделает веткой по умолчанию. Измените имя по умолчанию в ваших [настройках](#).

Рисунок 14. Создание репозитория

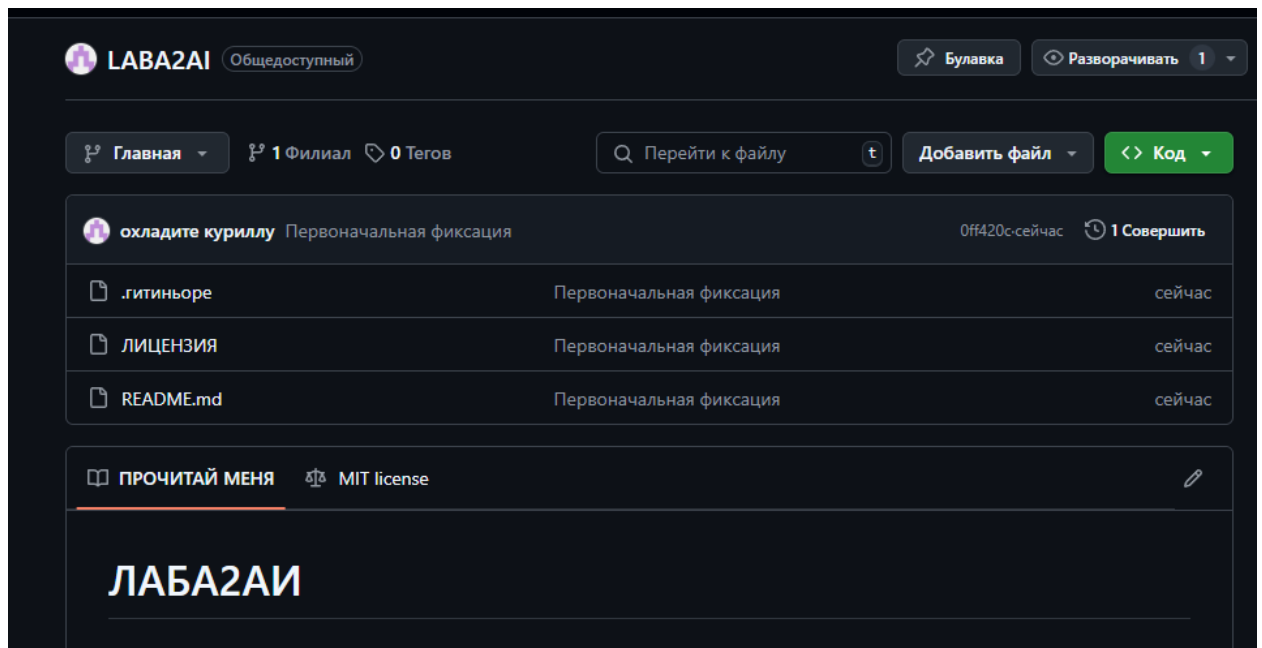


Рисунок 15. Созданный репозиторий

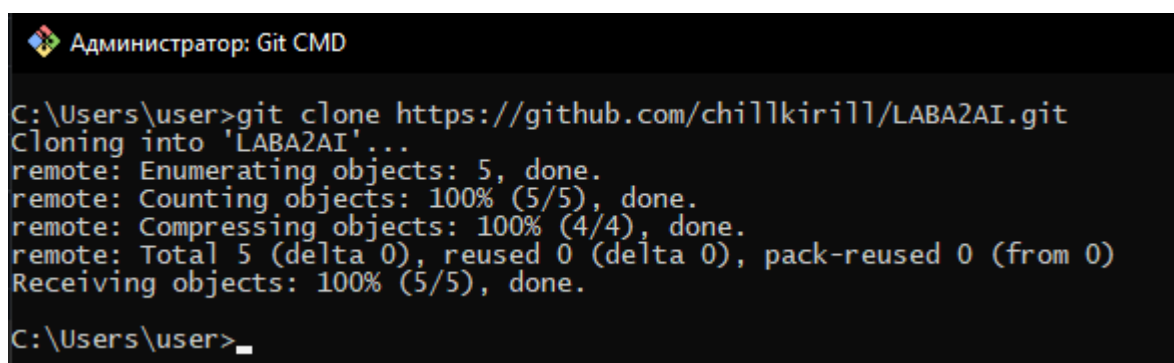


Рисунок 16. Клонирование репозитория

```

C:\Users\user\LABA2AI>git add .

C:\Users\user\LABA2AI>git commit -m "Добавление файлов, с которыми работал"
[main aae7055] Добавление файлов, с которыми работал
11 files changed, 934 insertions(+)
create mode 100644 1pr.ipynb
create mode 100644 2pr.ipynb
create mode 100644 individ11.ipynb
create mode 100644 pr10.ipynb
create mode 100644 pr3.ipynb
create mode 100644 pr4.ipynb
create mode 100644 pr5.ipynb
create mode 100644 pr6.ipynb
create mode 100644 pr7.ipynb
create mode 100644 pr8.ipynb
create mode 100644 pr9.ipynb

C:\Users\user\LABA2AI>git push
Enumerating objects: 14, done.
Counting objects: 100% (14/14), done.
Delta compression using up to 8 threads
Compressing objects: 100% (13/13), done.
Writing objects: 100% (13/13), 8.52 KiB | 8.52 MiB/s, done.
Total 13 (delta 5), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (5/5), done.
To https://github.com/chillkirill/LABA2AI.git
    0ff420c..aae7055  main -> main

C:\Users\user\LABA2AI>_

```

Рисунок 17. Создание коммита с несколькими рабочими файлами

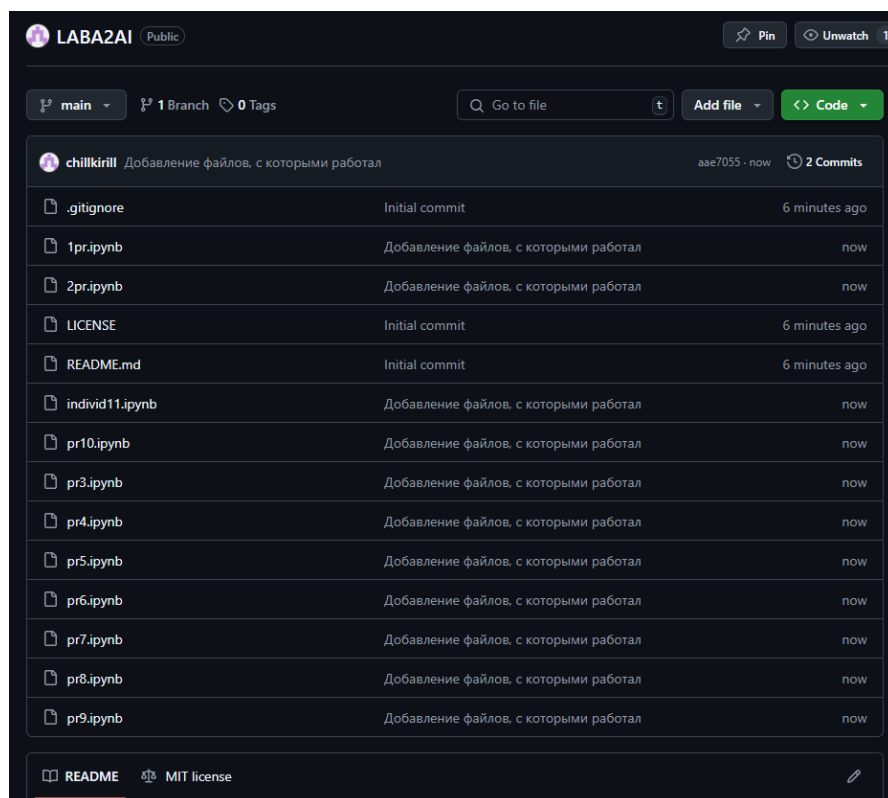


Рисунок 18. Результат коммита

**Вывод:** в ходе этой лабораторной работы были исследованы базовые возможности библиотеки NumPy языка программирования Python. Была проделана работа с обратными, транспонированными, единичными, нулевыми и обычными матрицами. Также были приобретены навыки с расчетом матриц.