

**РОССИЙСКОЙ ФЕДЕРАЦИИ**  
**Федеральное государственное автономное**  
**образовательное учреждение высшего образования**  
**«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»**

**Кафедра инфокоммуникаций**  
**«Основы работы с библиотекой NumPy»**

**Отчет по лабораторной работе № 3.2**  
**по дисциплине «Программирование на Python»**

Выполнил студент группы ИВТ-б-о-21-1

Харченко Богдан.

«10» марта 2023г.

Подпись студента

---

Работа защищена «    » \_\_\_\_\_ 20\_\_ г.

Проверил Воронкин Р.А. \_\_\_\_\_  
(подпись)

Ставрополь 2023

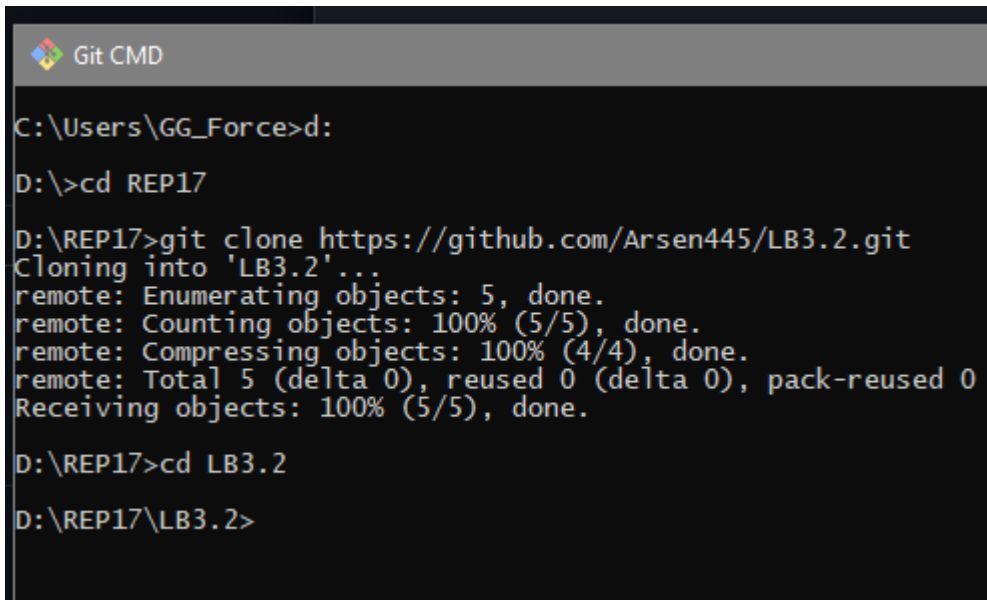
**Цель работы:** исследовать базовые возможности библиотеки NumPy языка программирования Python.

**Порядок выполнения работы:**

1. Создал общедоступный репозиторий на GitHub, в котором использована лицензия MIT и язык программирования Python.

Рисунок 1 - Создание репозитория

2. Выполните клонирование созданного репозитория.



```
Git CMD
C:\Users\GG_Force>d:
D:\>cd REP17
D:\REP17>git clone https://github.com/Arsen445/LB3.2.git
Cloning into 'LB3.2'...
remote: Enumerating objects: 5, done.
remote: Counting objects: 100% (5/5), done.
remote: Compressing objects: 100% (4/4), done.
remote: Total 5 (delta 0), reused 0 (delta 0), pack-reused 0
Receiving objects: 100% (5/5), done.
D:\REP17>cd LB3.2
D:\REP17\LB3.2>
```

Рисунок 2 - Клонирование репозитория

3. Организуйте свой репозиторий в соответствие с моделью ветвления git-flow.

```
C:\Users\Asus\Desktop\Учеба\4 семестр\Анализ данных\lw_3.2>git checkout -b develop
Switched to a new branch 'develop'

C:\Users\Asus\Desktop\Учеба\4 семестр\Анализ данных\lw_3.2>
```

Рисунок 3 - Ветвление по модели git-flow

4. Проработать примеры лабораторной работы.

Пример 1.

В приведенной записи, в квадратных скобках указывается номер строки – первой цифрой и номер столбца – второй.

Двоеточие означает “все элементы”, в приведенном примере, первый элемент – это номер строки, второй – указание на то, что необходимо взять элементы всех столбцов матрицы.

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

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

```
Ввод [2]: m[1, 0]
```

```
Out[2]: 5
```

```
Ввод [3]: m[1, :]
```

```
Out[3]: matrix([[5, 6, 7, 8]])
```

```
Ввод [4]: m[:, 2]
```

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

```
Ввод [5]: m[1, 2:]
```

```
Out[5]: matrix([[7, 8]])
```

```
Ввод [6]: m[0:2, 1]
```

```
Out[6]: matrix([[2],
               [6]])
```

```
Ввод [7]: m[0:2, 1:3]
```

```
Out[7]: matrix([[2, 3],
               [6, 7]])
```

```
Ввод [8]: cols = [0, 1, 3]
```

```
Ввод [9]: m[:, cols]
```

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

Рисунок 4 - Результат выполнения примера 1

## Пример 2.

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

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

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

```
Ввод [2]: type(m)  
Out[2]: numpy.matrix
```

```
Ввод [3]: m = np.array(m)  
type(m)  
Out[3]: numpy.ndarray
```

```
Ввод [4]: m.shape  
Out[4]: (3, 4)
```

```
Ввод [5]: m.max()  
Out[5]: 9
```

```
Ввод [6]: np.max(m)  
Out[6]: 9
```

```
Ввод [7]: m = np.matrix(m)  
m.max(axis=1)  
Out[7]: matrix([[4],  
 [8],  
 [9]])
```

```
Ввод [8]: m.max(axis=0)  
Out[8]: matrix([[9, 6, 7, 8]])
```

```
Ввод [9]: m.mean()  
Out[9]: 4.833333333333333
```

```
Ввод [10]: m.mean(axis=1)  
Out[10]: matrix([[2.5],  
 [6.5],  
 [5.5]])
```

```
Ввод [11]: m.sum(axis=0)  
Out[11]: matrix([[15, 9, 15, 19]])
```

Рисунок 5 - Результат выполнения примера 2

## Пример 3.

Использование `boolean` массивов для доступа к данным порой является более лучшим вариантом, чем использование численных индексов.

Как вы знаете, в Python есть такой тип данных – `boolean`. Переменные этого типа принимают одно из двух значений: `True` или `False`. Такие

переменные можно создать самостоятельно, либо они могут являться результатом какого-то выражения. Используя второй подход, можно построить на базе созданных нами в самом начале ndarray массивов массивы с элементами типа boolean.

Самым замечательным в использовании boolean массивов при работе с ndarray является то, что их можно применять для построения выборок.

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.

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

Ввод [2]: less_than_5 = nums < 5
          less_than_5

Out[2]: array([ True,  True,  True,  True, False, False, False, False, False,
               False])

Ввод [3]: pos_a = letters == 'a'
          pos_a

Out[3]: array([ True, False, False, False,  True, False, False])

Ввод [4]: nums[less_than_5]

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

Ввод [5]: nums[nums < 5] = 10
          print(nums)

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

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

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

Ввод [7]: mod_m = np.logical_and(m >= 3, m <= 7)
          mod_m

Out[7]: matrix([[False, False,  True,  True],
               [ True,  True,  True, False],
               [False, False,  True,  True]])

Ввод [8]: m[mod_m]

Out[8]: matrix([[3, 4, 5, 6, 7, 5, 7]])

Ввод [9]: m[m > 7] = 25
          print(m)

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

Рисунок 6 - Результат выполнения примера 3

#### Пример 4.

Функция `arange()` аналогична по своему назначению функции `range()` из стандартной библиотеки Python. Ее основное отличие заключается в том, что `arange()` позволяет строить вектор с указанием шага в виде десятичной дроби.

Функция `np.ravel()` используется для того, чтобы преобразовать матрицу в одномерный вектор.

Функция `np.where()` возвращает один из двух заданных элементов в зависимости от условия.

```
Ввод [1]: import numpy as np
          np.arange(10)

Out[1]: array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9])

Ввод [2]: np.arange(5, 12)

Out[2]: array([ 5,  6,  7,  8,  9, 10, 11])

Ввод [3]: np.arange(1, 5, 0.5)

Out[3]: array([1. , 1.5, 2. , 2.5, 3. , 3.5, 4. , 4.5])

Ввод [4]: A = np.array([[1, 2, 3], [4, 5, 6], [7, 8, 9]])
          A

Out[4]: array([[1, 2, 3],
               [4, 5, 6],
               [7, 8, 9]])

Ввод [5]: np.ravel(A)

Out[5]: array([1, 2, 3, 4, 5, 6, 7, 8, 9])

Ввод [6]: np.ravel(A, order='C')

Out[6]: array([1, 2, 3, 4, 5, 6, 7, 8, 9])

Ввод [7]: np.ravel(A, order='F')

Out[7]: array([1, 4, 7, 2, 5, 8, 3, 6, 9])

Ввод [8]: a = np.array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9])
          np.where(a % 2 == 0, a * 10, a / 10)

Out[8]: array([ 0. ,  0.1, 20. ,  0.3, 40. ,  0.5, 60. ,  0.7, 80. ,  0.9])

Ввод [9]: a = np.random.rand(10)
          a

Out[9]: array([0.55078842, 0.83322535, 0.50401979, 0.41202144, 0.53401922,
               0.81203636, 0.86304881, 0.75363088, 0.22908859, 0.04878729])

Ввод [10]: np.where(a > 0.5, True, False)

Out[10]: array([ True,  True,  True, False,  True,  True,  True,  True, False,
                False])

Ввод [11]: np.where(a > 0.5, 1, -1)

Out[11]: array([ 1,  1,  1, -1,  1,  1,  1,  1, -1, -1])
```

Рисунок 7 - Результат выполнения примера 4

### Пример 5.

Функция `meshgrid()` позволяет получить матрицу координат из координатных векторов. Если, например, у нас есть два одномерных вектора координат, то передав их в качестве аргументов в `meshgrid()` мы получим две матрицы, в которой элементы будут составлять пары, заполняя все пространство, определяемое этими векторами.

Каждому элементу `xg[i,j]` соответствует свой элемент `yg[i,j]`. Можно визуализировать эти данные.

Строка `%matplotlib inline` строка нужна, если вы работаете в Jupyter Notebook, чтобы графики рисовались “по месту”.

```
Ввод [1]: import numpy as np

Ввод [2]: x = np.linspace(0, 1, 5)
x
Out[2]: array([0. , 0.25, 0.5 , 0.75, 1. ])

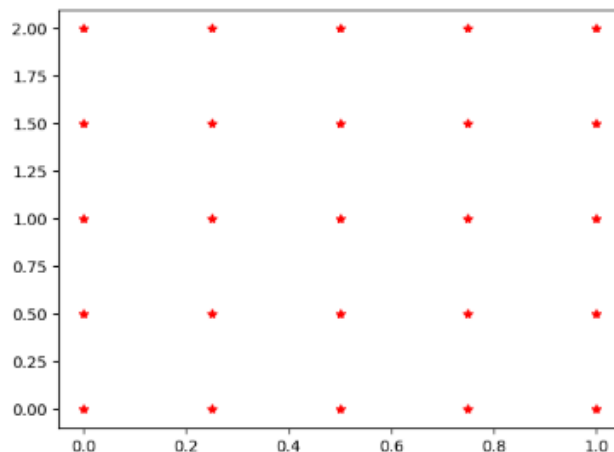
Ввод [3]: y = np.linspace(0, 2, 5)
y
Out[3]: array([0. , 0.5, 1. , 1.5, 2. ])

Ввод [4]: xg, yg = np.meshgrid(x, y)
xg
Out[4]: array([[0. , 0.25, 0.5 , 0.75, 1. ],
               [0. , 0.25, 0.5 , 0.75, 1. ],
               [0. , 0.25, 0.5 , 0.75, 1. ],
               [0. , 0.25, 0.5 , 0.75, 1. ],
               [0. , 0.25, 0.5 , 0.75, 1. ]])

Ввод [5]: yg
Out[5]: array([[0. , 0. , 0. , 0. , 0. ],
               [0.5, 0.5, 0.5, 0.5, 0.5],
               [1. , 1. , 1. , 1. , 1. ],
               [1.5, 1.5, 1.5, 1.5, 1.5],
               [2. , 2. , 2. , 2. , 2. ]])

Ввод [6]: import matplotlib.pyplot as plt
%matplotlib inline
plt.plot(xg, yg, color="r", marker="*", linestyle="none")

Out[6]: [<matplotlib.lines.Line2D at 0x2d9a2a20ca0>,
<matplotlib.lines.Line2D at 0x2d9a2a11be0>,
<matplotlib.lines.Line2D at 0x2d9a2a20d60>,
<matplotlib.lines.Line2D at 0x2d9a2a20eb0>,
<matplotlib.lines.Line2D at 0x2d9a2a20fd0>]
```



Ввод [ ]:



## Рисунок 8 - Результат выполнения примера 5

### Пример 6.

Функция `permutation()` либо генерирует список заданной длины из натуральных чисел от нуля до указанного числа, либо перемешивает переданный ей в качестве аргумента массив.

Основное практическое применение эта функция находит в задачах машинного обучения, где довольно часто требуется перемешать выборку данных перед тем, как передавать ее в алгоритм.

```
Ввод [1]: import numpy as np

Ввод [2]: np.random.permutation(7)
Out[2]: array([1, 2, 3, 5, 6, 4, 0])

Ввод [3]: a = ['a', 'b', 'c', 'd', 'e']
          np.random.permutation(a)
Out[3]: array(['a', 'd', 'c', 'b', 'e'], dtype='<U1')

Ввод [4]: arr = np.linspace(0, 10, 5)
          arr
Out[4]: array([ 0. ,  2.5,  5. ,  7.5, 10. ])

Ввод [5]: arr_mix = np.random.permutation(arr)
          arr_mix
Out[5]: array([ 5. , 10. ,  2.5,  7.5,  0. ])

Ввод [6]: index_mix = np.random.permutation(len(arr_mix))
          index_mix
Out[6]: array([3, 2, 0, 4, 1])

Ввод [7]: arr[index_mix]
Out[7]: array([ 7.5,  5. ,  0. , 10. ,  2.5])
```

Рисунок 9 - Результат выполнения примера 6

5. 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.

## Вариант 9.

Характеристикой столбца целочисленной матрицы назовем сумму модулей его отрицательных нечетных элементов. Переставляя столбцы заданной матрицы, расположить их в соответствии с ростом характеристик. Найти сумму элементов в тех столбцах, которые содержат хотя бы один отрицательный элемент.

### Индивидуальное задание

Элемент матрицы называется локальным минимумом, если он строго меньше всех имеющихся у него соседей. Подсчитать количество локальных минимумов заданной матрицы размером 10 на 10. Найти сумму модулей элементов, расположенных выше главной диагонали.

```
Ввод [1]: import numpy as np

# Создаем матрицу размером 10 на 10
matrix = np.random.randint(0, 100, (10, 10))
executed in 142ms, finished 10:16:33 2023-04-28

Ввод [2]: # Определяем локальные минимумы
is_local_min = np.logical_and(matrix[1:] > matrix[:-1], matrix[:-1] < matrix[1:])
local_min_count = np.count_nonzero(is_local_min)
executed in 5ms, finished 10:16:51 2023-04-28

Ввод [3]: # Считаем сумму модулей элементов выше главной диагонали
sum_above_diag = np.sum(np.abs(np.triu(matrix, k=1)))
executed in 18ms, finished 10:17:01 2023-04-28

Ввод [4]: print(f"Количество локальных минимумов: {local_min_count}")
print(f"Сумма модулей элементов выше главной диагонали: {sum_above_diag}")
executed in 13ms, finished 10:18:29 2023-04-28

Количество локальных минимумов: 39
Сумма модулей элементов выше главной диагонали: 2139
```

```
Ввод [5]: print(matrix)
executed in 7ms, finished 10:19:32 2023-04-28

[[11 61 83 76 65  0 49 51 62 95]
 [60 33 41 52 25 88 27 41 17 54]
 [70 96 30 12  3 34 48 23 28  9]
 [61 87 13 62 84 94 55 73 42 78]
 [62 32 24  1 45 12 40 26 92  4]
 [37 11 75 73 92  7 31 44  7 63]
 [29 47 78  1 55 11 98 55 91 48]
 [78 78 77 59 24 44 40 65 74 44]
 [39 96 43 98  8 94 64 59 94 38]
 [39 29 30 36 50 91 68 21 42 32]]
```

Рисунок 10 - Результат выполнения индивидуального задания 1

6. Создать ноутбук, в котором выполнить решение вычислительной задачи.

Есть система из двух зарядов с зарядами  $Q_1$  и  $Q_2$ , размещенных на расстоянии  $d$  друг от друга. Необходимо найти потенциальную энергию системы.

Потенциальная энергия системы зарядов может быть найдена с помощью интеграла:

$$U = \frac{1}{4\pi\epsilon_0} \int \frac{Q_1 Q_2}{r} dr,$$

где  $r$  - расстояние между зарядами,  $\epsilon_0$  - электрическая постоянная.

## Индивидуальное задание 2

Есть система из двух зарядов с зарядами  $Q_1$  и  $Q_2$ , размещенных на расстоянии  $d$  друг от друга. Необходимо найти потенциальную энергию системы.

Потенциальная энергия системы зарядов может быть найдена с помощью интеграла:

$$U = \frac{1}{4\pi\epsilon_0} \int \frac{Q_1 Q_2}{r} dr,$$

где  $r$  - расстояние между зарядами,  $\epsilon_0$  - электрическая постоянная.

```
Ввод [1]: import numpy as np
          executed in 142ms, finished 10:33:59 2023-04-28

Ввод [3]: # определяем константы
          epsilon0 = 8.85e-12 # электрическая постоянная
          k = 1 / (4 * np.pi * epsilon0)
          executed in 5ms, finished 10:35:09 2023-04-28

Ввод [5]: # Задаём функцию, которая вычисляет подынтегральное выражение в точке r
          def integrand(r, q1, q2):
              return q1 * q2 / r
          executed in 5ms, finished 10:38:08 2023-04-28

Ввод [9]: # Вычисляем значение интеграла
          from scipy.integrate import quad

          q1 = 3 # Кл
          q2 = -5 # Кл
          d = 2 # м

          # игнорируем сообщение о возможной неточности
          import warnings
          warnings.filterwarnings('ignore')

          result, _ = quad(integrand, 0, d, args=(q1, q2), maxp1=100)
          U = k * result
          print(f"Потенциальная энергия системы: {U:.4e} Дж")
          executed in 13ms, finished 10:42:37 2023-04-28

          Потенциальная энергия системы: -5.6213e+12 Дж
```

Рисунок 11 - Результат выполнения индивидуального задания 2

### 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?

Этот объект является многомерным однородным массивом с заранее заданным количеством элементов.

### **3. Как осуществляется доступ к частям многомерного массива?**

В квадратных скобках указывается номер строки – первой цифрой и номер столбца – второй.

Двоеточие означает “все элементы”, первый элемент – это номер строки, второй – указание на то, что необходимо взять элементы всех столбцов матрицы.

### **4. Как осуществляется расчет статистик по данным?**

Для расчета той или иной статистики, соответствующую функцию можно вызвать как метод объекта, с которым вы работаете.

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

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

### **5. Как выполняется выборка данных из массивов ndarray?**

Использование `boolean` массивов для доступа к данным порой является более лучшим вариантом, чем использование численных индексов.

Как вы знаете, в Python есть такой тип данных – `boolean`. Переменные этого типа принимают одно из двух значений: `True` или `False`. Такие переменные можно создать самостоятельно, либо они могут являться результатом какого-то выражения.

Самым замечательным в использовании `boolean` массивов при работе с `ndarray` является то, что их можно применять для построения выборок.

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`.

**Вывод:** были исследованы базовые возможности библиотеки NumPy для языка программирования Python.