

**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

Эсеналиев Арсен.

«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.

**Create a new repository**

A repository contains all project files, including the revision history. Already have a project repository elsewhere? [Import a repository.](#)

---

**Owner \*** **Repository name \***

Arsen445 / LB3.2

LB3.2 is available.

Great repository names are short and memorable. Need inspiration? How about [automatic-potato?](#)

**Description** (optional)

---

☒ **Public**  
Anyone on the internet can see this repository. You choose who can commit.

☐ **Private**  
You choose who can see and commit to this repository.

---

**Initialize this repository with:**

☒ **Add a README file**  
This is where you can write a long description for your project. [Learn more about READMEs.](#)

**Add .gitignore**

**.gitignore template:** Python

Choose which files not to track from a list of templates. [Learn more about ignoring files.](#)

**Choose a license**

**License:** MIT License

A license tells others what they can and can't do with your code. [Learn more about licenses.](#)

This will set **main** as the default branch. Change the default name in your [settings](#).

---

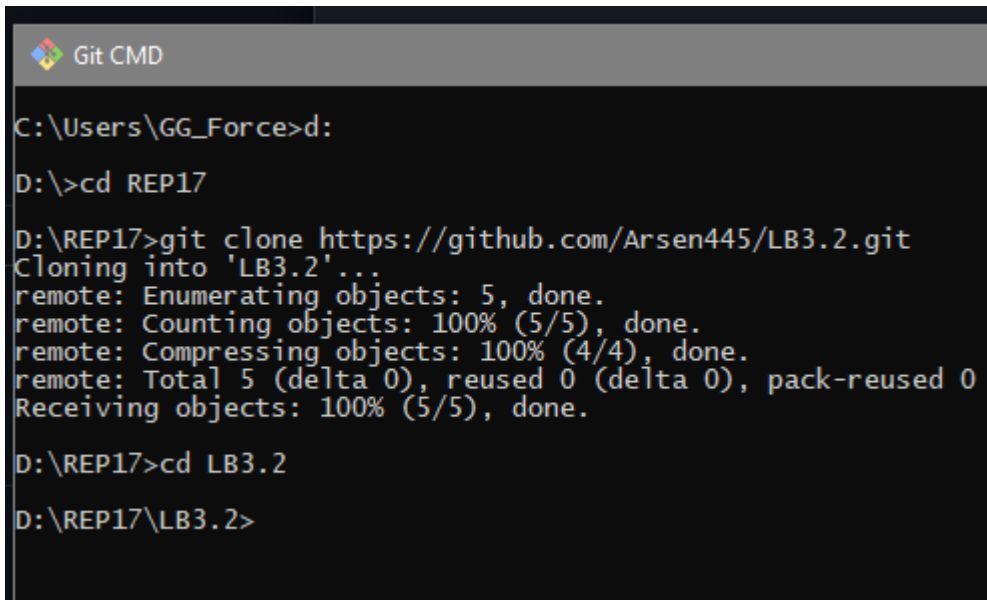
You are creating a public repository in your personal account.

---

[Create repository](#)

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`

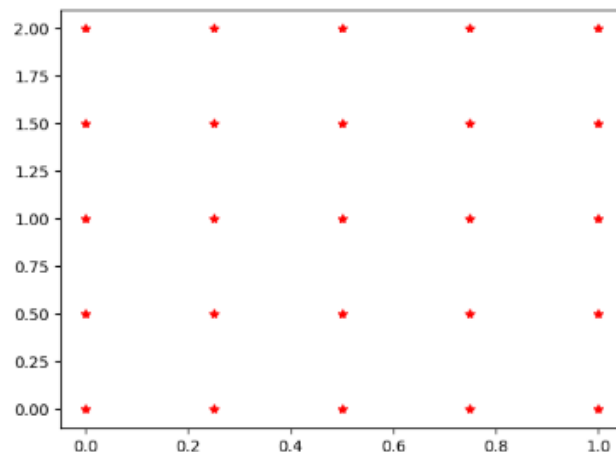
Out[4]: `array([[0. , 0.25, 0.5 , 0.75, 1. ],  
[0. , 0.25, 0.5 , 0.75, 1. ],  
[0. , 0.25, 0.5 , 0.75, 1. ],  
[0. , 0.25, 0.5 , 0.75, 1. ],  
[0. , 0.25, 0.5 , 0.75, 1. ]])`

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 27(7).

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 1

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.

```
Ввод [1]: import numpy as np

Ввод [2]: # Создаем матрицу размером 5x5 со случайными целыми числами от -9 до 9
matrix = np.random.randint(-9, 10, size=(5, 5))
print("Исходная матрица:\n", matrix)

Исходная матрица:
[[-2  4  7 -6  2]
 [ 6  5  2  8 -9]
 [ 4 -9 -6  5  9]
 [-3 -8  6  1 -7]
 [-5 -7 -6  3  1]]

Ввод [3]: # Вычисляем характеристики для каждого столбца
characteristics = np.sum(np.abs(matrix[matrix < 0][::2]), axis=0)
print("Характеристики столбцов:\n", characteristics)

Характеристики столбцов:
36

Ввод [4]: # Сортируем столбцы матрицы по возрастанию характеристик
sorted_matrix = matrix[:, np.argsort(characteristics)]
print("Отсортированная матрица:\n", sorted_matrix)

Отсортированная матрица:
[[-2]
 [ 6]
 [ 4]
 [-3]
 [-5]]

Ввод [5]: # Находим сумму элементов в столбцах, содержащих хотя бы один отрицательный элемент
sum_negative_columns = np.sum(sorted_matrix[:, np.any(sorted_matrix < 0, axis=0)], axis=0)
print("Сумма элементов в столбцах с отрицательными элементами:", sum_negative_columns.sum())

Сумма элементов в столбцах с отрицательными элементами: 0
```

Рисунок 10 - Результат выполнения индивидуального задания 1

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

Допустим, что у нас есть маятник длиной  $l$ , который начинает колебаться под действием гравитации с ускорением свободного падения  $g$ . Масса маятника  $m$  известна. Необходимо найти собственную частоту маятника, получить кинематическое уравнение гармонических колебаний и построить график колебаний.

## Индивидуальное задание 2

Допустим, что у нас есть маятник длиной  $l$ , который начинает колебаться под действием гравитации с ускорением свободного падения  $g$ . Масса маятника  $m$  известна. Необходимо найти собственную частоту маятника, получить кинематическое уравнение гармонических колебаний и построить график колебаний.

```
Ввод [5]: import numpy as np
import matplotlib.pyplot as plt

#Задаем параметры маятника
l = 1.5 # Длина маятника, м
g = 9.81 # Ускорение свободного падения, м/с^2
m = 0.2 # Масса маятника, кг

#Вычисляем собственную частоту маятника
omega = np.sqrt(g / l)

print(f"Собственная частота маятника: {omega:.2f} рад/с")

#Вычисляем период колебаний
T = 2 * np.pi / omega

print(f"Период колебаний: {T:.2f} с")

#Генерируем данные для графика
t = np.linspace(0, 5 * T, 1000)
theta = np.cos(omega * t)

#Строим график колебаний
plt.plot(t, theta)
plt.xlabel("Время, с")
plt.ylabel("Угол отклонения, рад")
plt.title("Гармонические колебания маятника")
plt.show()
```

Собственная частота маятника: 2.56 рад/с  
Период колебаний: 2.46 с

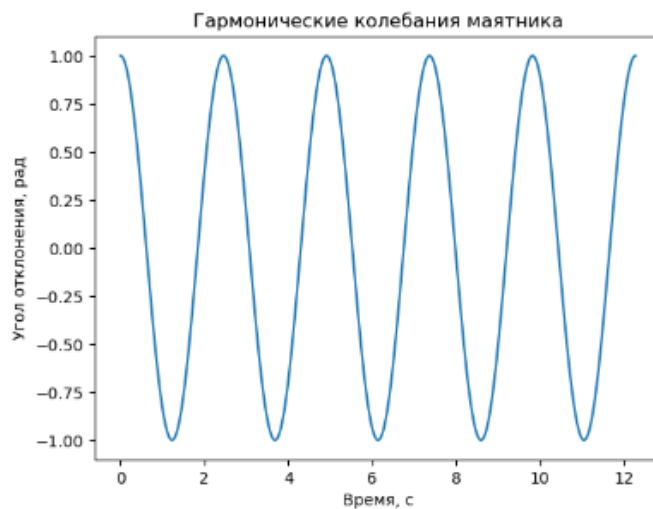


Рисунок 11 - Результат выполнения индивидуального задания 2

### 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 `pums`, то получим массив, в котором будут содержаться элементы из `pums` с индексами равными индексам `True` позиций массива `less_than_5`.

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