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

Толубаев Рамиль Ахметович

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.

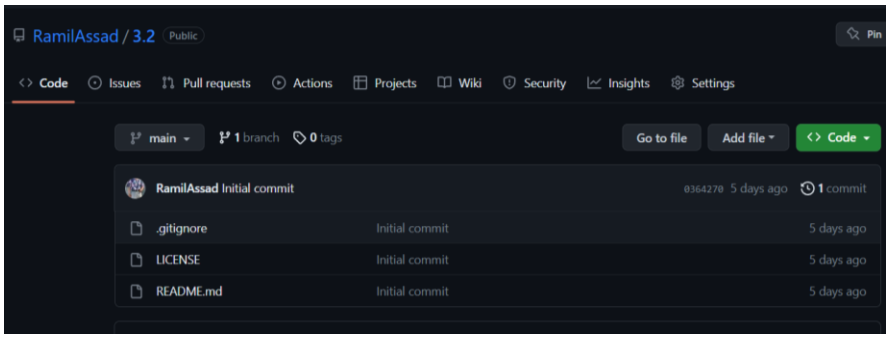

Рисунок 1 - Создание репозитория

2. Выполните клонирование созданного репозитория.

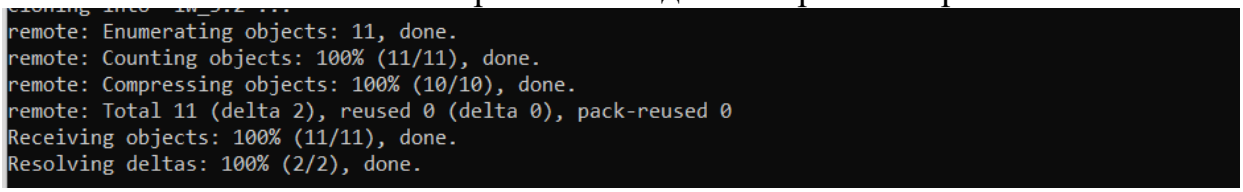

Рисунок 2 - Клонирование репозитория

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.

Пример 1

```
In [2]: 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]]

In [3]: m[1, 0]
Out[3]: 5

In [4]: m[1, :]
Out[4]: matrix([[5, 6, 7, 8]])

In [5]: m[:, 2]
Out[5]: matrix([[3],
                [7],
                [5]])

In [6]: m[1, 2:]
Out[6]: matrix([[7, 8]])

In [7]: m[0:2, 1]
Out[7]: matrix([[2],
                [6]])

In [8]: m[0:2, 1:3]
Out[8]: matrix([[2, 3],
                [6, 7]])

In [9]: cols = [0, 1, 3]
        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.

Пример 2

```
In [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]]

In [2]: type(m)

Out[2]: numpy.matrix

In [3]: m = np.array(m)
        type(m)

Out[3]: numpy.ndarray

In [4]: m.shape

Out[4]: (3, 4)

In [5]: m.max()

Out[5]: 9

In [6]: np.max(m)

Out[6]: 9

In [8]: m = np.matrix(m)
        m.max(axis=1)

Out[8]: matrix([[4],
                [8],
                [9]])

In [9]: m.max(axis=0)

Out[9]: matrix([[9, 6, 7, 8]])

In [10]: m.mean()

Out[10]: 4.833333333333333

In [11]: m.mean(axis=1)

Out[11]: matrix([[2.5],
                 [6.5],
                 [5.5]])

In [13]: m.sum(axis=0)

Out[13]: 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.

Пример 3

```
In [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'])

In [2]: less_then_5 = nums < 5
        less_then_5

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

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

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

In [4]: nums[less_then_5]

Out[4]: 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 = 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]]

In [8]: mod_m = np.logical_and(m >= 3, m <= 7)
        mod_m

Out[8]: matrix([[False, False,  True,  True],
                [ True,  True,  True, False],
                [False, False,  True,  True]])

In [9]: m[mod_m]

Out[9]: matrix([[3, 4, 5, 6, 7, 5, 7]])

In [10]: 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.

Пример 4

```
In [1]: import numpy as np
        np.arange(10)
Out[1]: array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9])

In [2]: np.arange(5, 12)
Out[2]: array([ 5,  6,  7,  8,  9, 10, 11])

In [3]: np.arange(1, 5, 0.5)
Out[3]: array([1. , 1.5, 2. , 2.5, 3. , 3.5, 4. , 4.5])

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

In [6]: np.ravel(A)
Out[6]: array([1, 2, 3, 4, 5, 6, 7, 8, 9])

In [7]: np.ravel(A, order='C')
Out[7]: array([1, 2, 3, 4, 5, 6, 7, 8, 9])

In [8]: np.ravel(A, order='F')
Out[8]: array([1, 4, 7, 2, 5, 8, 3, 6, 9])

In [9]: a = np.array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9])
        np.where(a % 2 == 0, a * 10, a / 10)
Out[9]: array([ 0. ,  0.1, 20. ,  0.3, 40. ,  0.5, 60. ,  0.7, 80. ,  0.9])

In [10]: a = np.random.rand(10)
         a
Out[10]: array([0.55642332, 0.37526446, 0.24589265, 0.95861707, 0.18235878,
                0.44916279, 0.67206023, 0.48754707, 0.09085441, 0.14104925])

In [11]: np.where(a > 0.5, True, False)
Out[11]: array([ True, False, False,  True, False, False,  True, False, False,
                False])

In [12]: np.where(a > 0.5, 1, -1)
Out[12]: array([ 1, -1, -1,  1, -1, -1,  1, -1, -1, -1])
```

Рисунок 7 - Результат выполнения примера 4

Пример 5.

Функция meshgrid() позволят 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”.

Пример 5

```
In [1]: import numpy as np

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

In [4]: y = np.linspace(0, 2, 5)
        y
Out[4]: array([0. , 0.5, 1. , 1.5, 2. ])

In [5]: xg, yg = np.meshgrid(x, y)
        xg
Out[5]: 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.  ]])

In [6]: yg
Out[6]: 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. ]])

In [7]: import matplotlib.pyplot as plt
        %matplotlib inline
        plt.plot(xg, yg, color="r", marker="*", linestyle="none")
Out[7]: [<matplotlib.lines.Line2D at 0x14b74807fd0>,
         <matplotlib.lines.Line2D at 0x14b7482a160>,
         <matplotlib.lines.Line2D at 0x14b7482a1c0>,
         <matplotlib.lines.Line2D at 0x14b7482a2e0>,
         <matplotlib.lines.Line2D at 0x14b7482a400>]
```

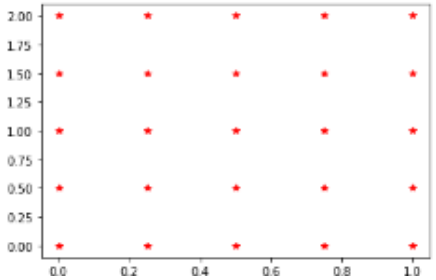

Рисунок 8 - Результат выполнения примера 5

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.

Пример 6

```
In [1]: import numpy as np

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

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

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

In [6]: arr_mix = np.random.permutation(arr)
        arr_mix
Out[6]: array([ 2.5, 10. ,  5. ,  0. ,  7.5])

In [8]: index_mix = np.random.permutation(len(arr_mix))
        index_mix
Out[8]: array([4, 1, 0, 3, 2])

In [9]: arr[index_mix]
Out[9]: array([10. ,  2.5,  0. ,  7.5,  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.

Дана целочисленная квадратная матрица.

Определить: произведение элементов в тех строках, которые не содержат отрицательных элементов; максимум среди сумм элементов диагоналей, параллельных главной диагонали матрицы.

Индивидуальное задание 1

Дана целочисленная квадратная матрица.
Определить:
• произведение элементов в тех строках, которые не содержат отрицательных элементов;
• максимум среди сумм элементов диагоналей, параллельных главной диагонали матрицы.
Не должны быть использованы условный оператор if, а также операторы циклов while и for, а только средства библиотеки NumPy.

In [1]:
```
import numpy as np
```

In [2]:
```
n = 5
a = np.random.randint(-10, 100, (n, n))
print(a)
```

```
5
[[65 12 86 89 62]
 [81 61 12 14 58]
 [46  9 67 77 77]
 [34 31 63 56 36]
 [-2  3 34  7 19]]
```

In [3]:
```
pos_a = a[np.where(a.min(axis=1) > 0)].prod(axis=1)
print(f"Произведение в строках без отрицательных элементов: {pos_a}")
```

```
Произведение в строках без отрицательных элементов: [370147440  48145104 164458602 133866432]
```

In [7]:
```
maximum = max([
    sum(np.diag(a, k=-3)),
    sum(np.diag(a, k=-2)),
    sum(np.diag(a, k=-1)),
    sum(np.diag(a, k=1)),
    sum(np.diag(a, k=2)),
    sum(np.diag(a, k=3))])
print(f"Максимум среди сумм элементов диагоналей, параллельных главной диагонали: {maximum}")
```

```
Максимум среди сумм элементов диагоналей, параллельных главной диагонали: 177
```

Рисунок 10 - Результат выполнения индивидуального задания 1

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

Рассчитать методом контурных токов токи в ветвях для заданной электрической цепи с источниками ЭДС.

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_then_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_then_5.

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