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

Институт цифрового развития
Кафедра инфокоммуникаций

«Основы работы с библиотекой NumPy»

ОТЧЕТ
по лабораторной работе №2
дисциплины
«Технологии распознавания образов»

Выполнил:
Борсуков Владислав Олегович
2 курс, группа ПИЖ-б-о-21-1,
011.03.04 «Программная инженерия»,
направленность (профиль) «Разработка
и сопровождение программного
обеспечения», очная форма обучения

_____
(подпись)

Проверил:

_____
(подпись)

Отчет защищен с оценкой_____Дата защиты_____

Ставрополь, 2022 г.

Проработка примеров из методических указаний:

```
In [1]: import numpy as np
        # Теперь создадим матрицу, с которой будем работать.
        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 [12]: # Элемент матрицы с заданными координатами
         m[1, 0]

Out[12]: 5

In [13]: # Строка матрицы
         m[1, :]

Out[13]: matrix([[5, 6, 7, 8]])

In [14]: # Столбец матрицы
         m[:, 2]

Out[14]: matrix([[3],
                 [7],
                 [5]])

In [15]: # Часть строки матрицы
         m[1, 2:]

Out[15]: matrix([[7, 8]])

In [7]: # Часть столбца матрицы
        m[0:2, 1]

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

In [8]: # Непрерывная часть матрицы
        m[0:2, 1:3]

Out[8]: matrix([[2, 3],
                [6, 7]])

In [9]: # Произвольные столбцы / строки матрицы
        cols = [0, 1, 3]
        m[:, cols]
```

Рисунок 1 – Проработка примеров

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

In [10]: # Расчет статистик по данным в массиве
         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 [11]: type(m)

Out[11]: numpy.matrix

In [16]: m = np.array(m)
         type(m)

Out[16]: numpy.ndarray

In [17]: # Размерность массива
         m.shape

Out[17]: (3, 4)

In [18]: # Вызов функции расчета статистики
         m.max()

Out[18]: 9

In [21]: np.max(m)

Out[21]: 9

In [22]: # Расчет статистик по строкам или столбцам массива
         m.max()

Out[22]: 9

In [23]: m.max(axis=1)

Out[23]: array([4, 8, 9])

In [24]: m.max(axis=0)

Out[24]: array([9, 6, 7, 8])
```

Рисунок 2 – Проработка примеров

```
In [25]: # Функции (методы) для расчета статистик в Numpy
         m.mean()
Out[25]: 4.833333333333333

In [26]: m.mean(axis=1)
Out[26]: array([2.5, 6.5, 5.5])

In [27]: # Использование boolean массива для доступа к ndarray
         nums = np.array([1, 2, 3, 4, 5, 6, 7, 8, 9, 10])
         letters = np.array(['a', 'b', 'c', 'd', 'a', 'e', 'b'])

In [28]: b = 5 > 7
         print(b)
         False

In [29]: less_then_5 = nums < 5
         less_then_5
Out[29]: array([ True,  True,  True,  True, False, False, False, False, False,
                False])

In [30]: pos_a = letters == 'a'
         pos_a
Out[30]: array([ True, False, False, False,  True, False, False])

In [31]: less_then_5 = nums < 5
         less_then_5
         nums[less_then_5]
Out[31]: array([1, 2, 3, 4])

In [32]: 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 [33]: mod_m = np.logical_and(m>=3, m<=7)
         mod_m
Out[33]: matrix([[False, False,  True,  True],
                 [ True,  True,  True, False],
                 [False, False,  True,  True]])
```

Рисунок 3 – Проработка примеров

```
In [33]: mod_m = np.logical_and(m>=3, m<=7)
         mod_m
Out[33]: matrix([[False, False,  True,  True],
                 [ True,  True,  True, False],
                 [False, False,  True,  True]])

In [34]: m[mod_m]
Out[34]: matrix([[3, 4, 5, 6, 7, 5, 7]])

In [35]: nums[nums < 5] = 10
         print(nums)
         [10 10 10 10  5  6  7  8  9 10]

In [36]: m[m > 7] = 25
         print(m)
         [[ 1  2  3  4]
          [ 5  6  7 25]
          [25  1  5  7]]

In [37]: # Дополнительные функции
         # np.arange()
         np.arange(10)
Out[37]: array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9])

In [38]: np.arange(5, 12)
Out[38]: array([ 5,  6,  7,  8,  9, 10, 11])

In [39]: np.arange(1, 5, 0.5)
Out[39]: array([1. , 1.5, 2. , 2.5, 3. , 3.5, 4. , 4.5])

In [40]: # np.matrix()
         a = [[1, 2], [3, 4]]
         np.matrix(a)
Out[40]: matrix([[1, 2],
                 [3, 4]])
```

Рисунок 4 – Проработка примеров

```
In [41]: # np.zeros(), np.eye()
         np.zeros((3, 4))

Out[41]: array([[0., 0., 0., 0.],
                [0., 0., 0., 0.],
                [0., 0., 0., 0.]])

In [42]: np.eye(3)

Out[42]: array([[1., 0., 0.],
                [0., 1., 0.],
                [0., 0., 1.]])

In [43]: # np.ravel()
         A = np.array([[1, 2, 3], [4, 5, 6], [7, 8, 9]])
         A

Out[43]: array([[1, 2, 3],
                [4, 5, 6],
                [7, 8, 9]])

In [44]: np.ravel(A)

Out[44]: array([1, 2, 3, 4, 5, 6, 7, 8, 9])

In [45]: np.ravel(A, order='C')

Out[45]: array([1, 2, 3, 4, 5, 6, 7, 8, 9])

In [46]: np.ravel(A, order='F')

Out[46]: array([1, 4, 7, 2, 5, 8, 3, 6, 9])

In [49]: # np.where()
         a = np.array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9])
         np.where(a % 2 == 0, a * 10, a / 10)

Out[49]: array([ 0. ,  0.1, 20. ,  0.3, 40. ,  0.5, 60. ,  0.7, 80. ,  0.9])

In [50]: a = np.random.rand(10)
         a

Out[50]: array([0.42098051, 0.13510814, 0.31091368, 0.23236793, 0.51940724,
                0.80800519, 0.07819001, 0.64569304, 0.18739326, 0.71199996])
```

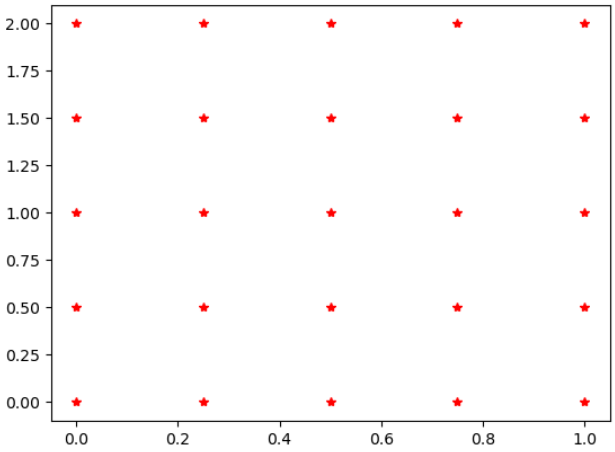
Рисунок 5 – Проработка примеров

```
In [51]: np.where(a > 0.5, True, False)

Out[51]: array([False, False, False, False,  True,  True, False,  True, False,
                True])

In [52]: np.where(a > 0.5, 1, -1)

Out[52]: array([-1, -1, -1, -1,  1,  1, -1,  1, -1,  1])

In [53]: # np.meshgrid()
         x = np.linspace(0, 1, 5)
         x

Out[53]: array([0.  , 0.25, 0.5 , 0.75, 1.  ])

In [54]: y = np.linspace(0, 2, 5)
         y

Out[54]: array([0. , 0.5, 1. , 1.5, 2. ])

In [55]: xg, yg = np.meshgrid(x, y)
         xg

Out[55]: 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 [56]: yg

Out[56]: 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. ]])
```

Рисунок 6 – Проработка примеров

```
In [57]: import matplotlib.pyplot as plt
         %matplotlib inline
         plt.plot(xg, yg, color="r", marker="*", linestyle="none")
```

```
Out[57]: [<matplotlib.lines.Line2D at 0x22390208430>,
          <matplotlib.lines.Line2D at 0x223901f9370>,
          <matplotlib.lines.Line2D at 0x223902084f0>,
          <matplotlib.lines.Line2D at 0x22390208640>,
          <matplotlib.lines.Line2D at 0x22390208760>]
```




```
In [58]: # np.random.permutation()
         np.random.permutation(7)
```

```
Out[58]: array([5, 2, 6, 4, 0, 3, 1])
```

```
In [59]: a = ['a', 'b', 'c', 'd', 'e']
         np.random.permutation(a)
```

```
Out[59]: array(['d', 'a', 'c', 'e', 'b'], dtype='<U1')
```

```
In [61]: arr = np.linspace(0, 10, 5)
         arr
```

```
Out[61]: array([ 0. ,  2.5,  5. ,  7.5, 10. ])
```

```
In [62]: arr_mix = np.random.permutation(arr)
         arr_mix
```

```
Out[62]: array([ 7.5,  5. , 10. ,  0. ,  2.5])
```

```
In [63]: index_mix = np.random.permutation(len(arr_mix))
         index_mix
```

```
Out[63]: array([3, 2, 0, 1, 4])
```

```
In [64]: arr[index_mix]
```

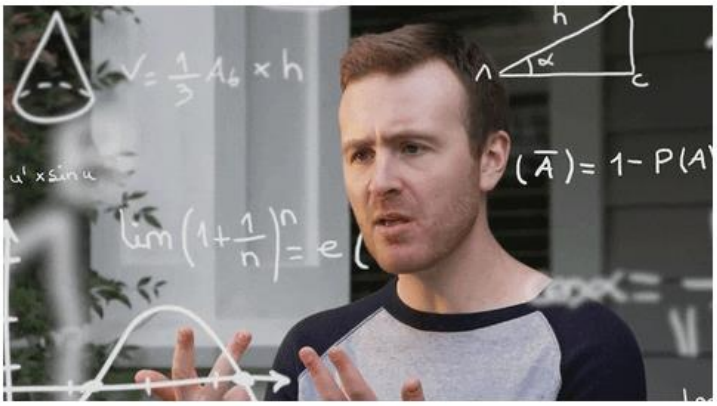
```
Out[64]: array([ 7.5,  5. ,  0. ,  2.5, 10. ])
```

Рисунок 7 – Проработка примеров

Выполнение заданий из файла lab3.2:

Лабораторная работа 3.2. Знакомство с NumPy

Библиотека NumPy -- быстрая библиотека для математики в Python, основная структура данных -- массив `numpy.array` :



In [2]:
```
# подключение модуля numpy под именем np
import numpy as np
```

In [3]:
```
# 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 [7]:
```
arr = np.random.randint(-6, 6, 5)
arr
```

Out[7]:
```
array([-5, -2,  4,  0, -1])
```

Рисунок 8 – Проработка lab3.2

In [8]:
```
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]
```

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

In [18]:
```
arr = np.random.randint(-6, 6, 5)
print(arr)
arr ** 3
```

```
[-6  3  0 -5 -5]
```

Out[18]:
```
array([-216,   27,    0, -125, -125], dtype=int32)
```

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

In [19]:
```
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]
```

Рисунок 9 – Проработка lab3.2

In [20]:
```
# 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 [37]:
```
arr_1 = np.random.randint(-6, 6, 5)
arr_2 = np.random.randint(-6, 6, 5)
result = arr_1 // arr_2
print(result)
```

```
[ 0  5 -1 -1  4]
```

Л — логика

К элементам массива можно применять логические операции.

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

In [38]:
```
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("А 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]
```

Рисунок 10 – Проработка lab3.2

In [39]:
```
arr_1 = np.random.randint(-10, 10, 5)
arr_2 = np.random.randint(-10, 10, 5)
print(arr_1 < 6)
print(arr_2 % 3 == 0)
```

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

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

In [59]:
```
print((arr_1 % 2 == 0) | (arr_2 > 2))
```

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

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

In [60]:
```
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 [62]:
```
arr = np.arange(1,21)
print(arr[(arr > 5) & (arr % 2 != 0)])
```

```
[ 7  9 11 13 15 17 19]
```

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:

Рисунок 11 – Проработка lab3.2

In [63]:
```
# 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(['m

# 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']
```

Пора еще немного потренироваться с NumPy.

Выполните операции, перечисленные ниже:

In [64]:
```
print("Разность между a и b:", a - b
      )
print("Квадраты элементов b:", b ** 2
      )
print("Половины произведений элементов массивов a и b:", a * b
      )

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

```
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: [0.1 0.4 0.9 1.6 2.5]

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

Рисунок 12 – Проработка lab3.2

Выполнить задания из файла lab3.2hw:

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 [2]:
```
import numpy as np

arr_1 = np.arange(1,7) * 2
arr_2 = np.array([7, 11, 15, 18, 23, 29])
print((arr_1 + arr_2) ** 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 [4]:
```
arr_1[np.where ((arr_2 > 12) & (arr_2 % 5 == 3))]
```

Out[4]:
```
array([ 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 [6]:
```
(arr_1 % 4 == 0) & (arr_2 < 14)
```

Out[6]:
```
array([False,  True, False, False, False, False])
```

Рисунок 13 – Выполнение заданий из lab3.2hw

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 получившиеся результаты
- Все комментарии оформляйте строго в ячейках формата markdown

In [8]:
```
import csv
import numpy as np
with open('topSubscribed.csv', 'r', newline='', encoding='utf-8') as csvfile:
    reader = csv.reader(csvfile, delimiter=';')
    views = []
    video = []
    next(reader)
    for i in reader:
        views.append(int(i[3]))
        video.append(int(i[4]))
    arr_views = np.array(views)
    arr_video = np.array(video)
    print(f"Среднее количество просмотров: {np.mean(arr_views)}")
    print(f"Среднее количество видео: {np.mean(arr_video)}" )

    print(f"Среднее отклонение количества просмотров {np.std(arr_views)}")
    print(f"Среднее отклонение количества видео: {np.std(arr_video)}")

    print(f"Дисперсия количества просмотров {np.var(arr_views)}")
    print(f"Дисперсия количества видео: {np.var(arr_video)}" )
```

```
Среднее количество просмотров: 9994912409.734
Среднее количество видео: 8536.892
Среднее отклонение количества просмотров 12998953101.280153
Среднее отклонение количества видео: 30775.865282300936
Дисперсия количества просмотров 1.6897278172928092e+20
Дисперсия количества видео: 947153883.874336
```

Рисунок 14 – Выполнение заданий из lab3.2hw

Индивидуальное задание:

1. Дана целочисленная прямоугольная матрица. Определить:

- количество строк, не содержащих ни одного нулевого элемента;
- максимальное из чисел, встречающихся в заданной матрице более одного раза.

In [43]:
```
import numpy as np

matrix = np.random.randint (-10, 11, (5, 5))
print(matrix)
```

```
[[  1   4   8  -6  -1]
 [  4 -10  -5  -8   9]
 [ -7  -2  -5   0   2]
 [  8  -5  -7  -9 -10]
 [ -2  -1   6  -1  -9]]
```

In [33]:
```
k = 0
for i in range(5):
    if 0 not in matrix[i, :]:
        k += 1
print(f"Количество строк без нулей: {k}")
```

```
Количество строк без нулей: 4
```

In [44]:
```
max_elem = matrix.max()
while True:
    if np.count_nonzero(matrix == max_elem) > 1:
        print(f"Максимальное из чисел в матрице, встречающееся более одного раза: {max_elem}")
        break
    else:
        max_elem -= 1
```

```
Максимальное из чисел в матрице, встречающееся более одного раза: 8
```

Рисунок 15 – Индивидуальное задание

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?

Основной элемент библиотеки NumPy — объект ndarray (что значит N-размерный массив). Этот объект является многомерным однородным массивом с заранее заданным количеством элементов.

3. Как осуществляется доступ к частям многомерного массива?

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

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

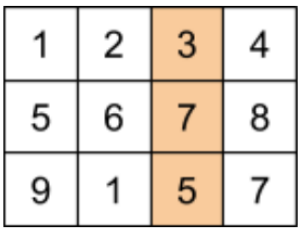
```
>>> m[1, 0]
5
```

Получим вторую строчку матрицы.

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

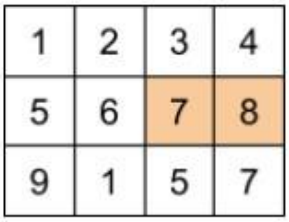
```
>>> m[1, :]
matrix([[5, 6, 7, 8]])
```

Извлечем третий столбец матрицы.




```
>>> m[:, 2]
matrix([[3],
        [7],
        [5]])
```

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.

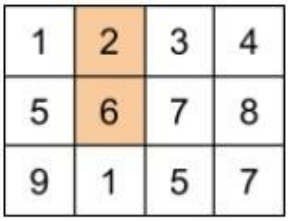



```
>>> m[1, 2:]
matrix([[7, 8]])
```

Запись **2:** означает, что начиная с третьего столбца включительно (т.к. нумерация начинается с 0, то третий элемент имеет индекс 2) взять все оставшиеся в ряду элементы .

Часть столбца матрицы

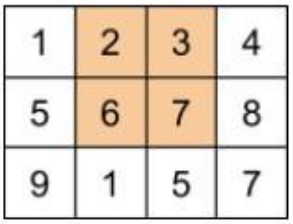
Аналогично предыдущему примеру, можно извлечь только часть столбца матрицы.




```
>>> m[0:2, 1]
matrix([[2],
        [6]])
```

Непрерывная часть матрицы

Извлечем из заданной матрицы матрицу, располагающуюся так как показано на рисунке ниже.




```
\>>> m[0:2, 1:3]
matrix([[2, 3],
        [6, 7]])
```

4. Как осуществляется расчет статистик по данным?

Функции (методы) для расчета статистик в *Numpy*

Ниже, в таблице, приведены методы объекта *ndarray* (или *matrix*), которые, как мы помним из раздела выше, могут быть также вызваны как функции библиотеки *Numpy*, для расчета статистик по данным массива.

Имя метода	Описание
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)

Вычислим некоторые из представленных выше статистик.

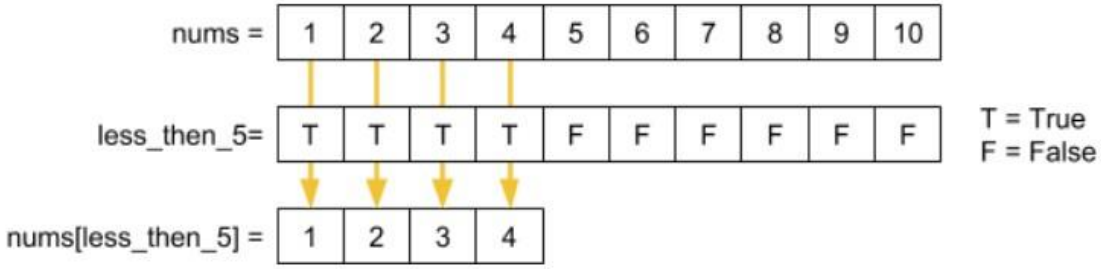
```
>>> m.mean()
4.833333333333333
>>> m.mean(axis=1)
matrix([[2.5],
        [6.5],
        [5.5]])
>>> m.sum()
58
>>> m.sum(axis=0)
matrix([[15,  9, 15, 19]])
```

5. Как выполняется выборка данных из массивов ndarray?

Самым замечательным в использовании *boolean* массивов при работе с *ndarray* является то, что их можно применять для построения выборок. Вернемся к рассмотренным выше примерам.

```
>>> less_then_5 = nums < 5
>>> less_then_5
array([ True,  True,  True,  True, False, False, False, False, False,
       False])
```

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*, графически это будет выглядеть так.




```
>>> nums[less_then_5]
array([1, 2, 3, 4])
```

Данный подход будет работать с массивами большей размерности. Возьмем уже знакомую нам по предыдущим урокам матрицу.

```
>>> 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]]
```

Построим логическую матрицу со следующим условием: *m>=3 and m<=7*, в *Numpy* нельзя напрямую записать такое условие, поэтому воспользуемся функцией *logical_and()*, ее и многие другие полезные функции вы сможете найти на странице *Logic functions*.

```
>>> mod_m = np.logical_and(m>=3, m<=7)
>>> mod_m
matrix([[False, False,  True,  True],
        [ True,  True,  True, False],
        [False, False,  True,  True]])
>>> m[mod_m]
matrix([[3, 4, 5, 6, 7, 5, 7]])
```

В результате мы получили матрицу с одной строкой, элементами которой являются все отмеченные как *True* элементы из исходной матрицы.