МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ 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 «Технологии распознавания образов»

Выполнил студент группы ПИЖ-б-о-21-1

Кучеренко С. Ю. « » 2023 г.

Подпись студента_____

Работа защищена « »_____2023 г.

Проверил Воронкин Р.А. _____
(подпись)

Ставрополь 2023

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

Выполнение работы:

1. Изучить теоретический материал работы.

2. Создать общедоступный репозиторий на GitHub, в котором будет использована лицензия MIT и выбранный Вами язык программирования (выбор языка программирования будет доступен после установки флажка Add .gitignore).

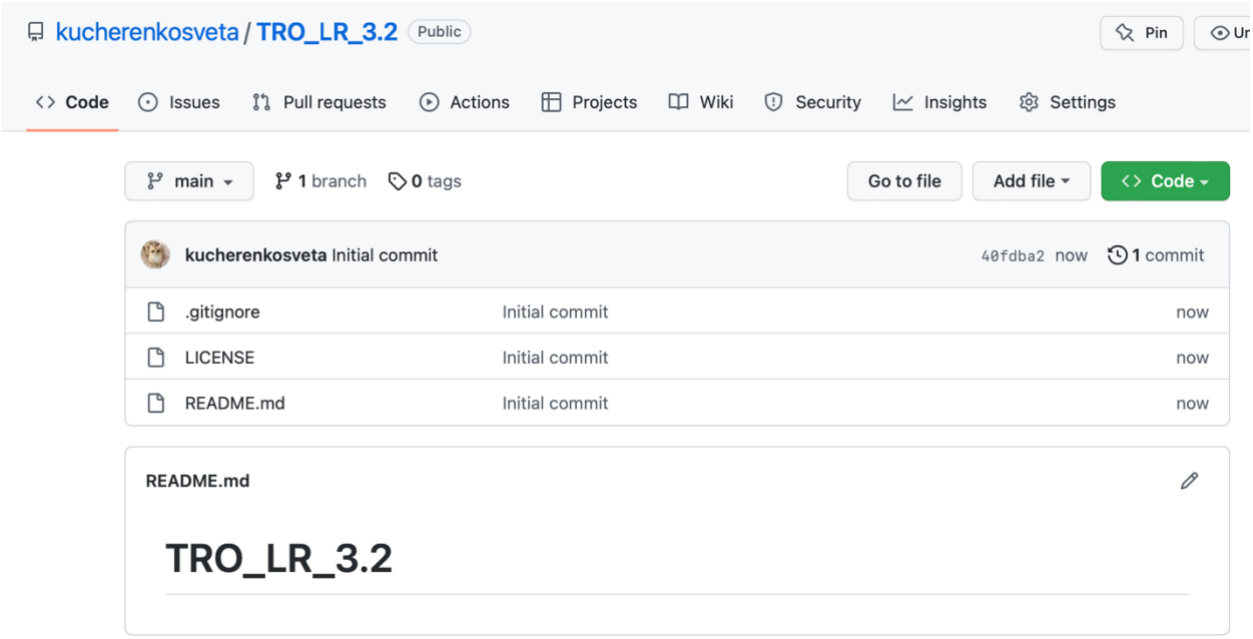


Рисунок 1 – Создание репозитория

3. Выполните клонирование созданного репозитория на рабочий компьютер.

```
Last login: Mon Feb 20 01:03:08 on ttys000
(base) svetik@MacBook-Air-Svetik ~ % cd ~/Desktop/TRO/
(base) svetik@MacBook-Air-Svetik TRO % git clone https://github.com/kucherenkosv
eta/TRO_LR_3.2.git
Cloning into 'TRO_LR_3.2'...
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.
(base) svetik@MacBook-Air-Svetik TRO %
```

Рисунок 2 – Клонирование репозитория

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

```
(base) svetik@MacBook-Air-Svetik TRO % cd TRO_LR_3.2
(base) svetik@MacBook-Air-Svetik TRO_LR_3.2 % git flow init

Which branch should be used for bringing forth production releases?
   - main
Branch name for production releases: [main]
Branch name for "next release" development: [develop]

How to name your supporting branch prefixes?
Feature branches? [feature/]
Bugfix branches? [bugfix/]
Release branches? [release/]
Hotfix branches? [hotfix/]
Support branches? [support/]
Version tag prefix? []
Hooks and filters directory? [/Users/svetik/Desktop/TRO/TRO_LR_3.2/.git/hooks]
```

Рисунок 3 – Организация в соответствие с моделью ветвления git-flow.

5. Дополните файл .gitignore необходимыми правилами для выбранного языка программирования, интерактивной оболочки Jupyter notebook и интегрированной среды разработки.

6. Проработать примеры лабораторной работы.

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

In [2]: 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]: type(m)
Out[3]: numpy.matrix

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

In [5]: m.max()
Out[5]: 9

In [6]: m.max(axis=1)
Out[6]: matrix([[4],
                [8],
                [9]])

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

In [8]: m.mean(axis=1)
Out[8]: matrix([[2.5],
                [6.5],
                [5.5]])
```

In [9]:
```
m.sum()
```

Out[9]:
```
58
```

In [10]:
```
nums = np.array([1, 2, 3, 4, 5, 6, 7, 8, 9, 10])
less_then_5 = nums < 5
less_then_5
```

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

In [11]:
```
nums[less_then_5]
```

Out[11]:
```
array([1, 2, 3, 4])
```

In [12]:
```
m = np.matrix('1 2 3 4; 5 6 7 8; 9 1 5 7')
mod_m = np.logical_and(m>=3, m <=7)
mod_m
```

Out[12]:
```
matrix([[False, False,  True,  True],
        [ True,  True,  True, False],
        [False, False,  True,  True]])
```

In [14]:
```
nums = np.array([1,2,3,4,5,6,7,8,9,10])
nums[nums<5]
```

Out[14]:
```
array([1, 2, 3, 4])
```

In [15]:
```
nums[nums<5] = 10
print(nums)
```

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

In [16]:
```
m = np.matrix('1 2 3 4; 5 6 7 8; 9 1 5 7')
m[m > 7] = 25
print(m)
```

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

In [17]:
```
np.arange(10)
```

Out[17]:
```
array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9])
```

In [18]:
```
np.arange(5,12)
```

Out[18]:
```
array([ 5,  6,  7,  8,  9, 10, 11])
```

In [19]:
```
np.arange(1,5,0.5)
```

Out[19]:
```
array([1. , 1.5, 2. , 2.5, 3. , 3.5, 4. , 4.5])
```

In [20]:
```
a = [[1, 2], [3, 4]]
np.matrix(a)
```

Out[20]:
```
matrix([[1, 2],
        [3, 4]])
```

In [21]:
```
np.zeros((3,4))
```

Out[21]:
```
array([[0., 0., 0., 0.],
       [0., 0., 0., 0.],
       [0., 0., 0., 0.]])
```

In [22]:
```
np.eye(5)
```

Out[22]:
```
array([[1., 0., 0., 0., 0.],
       [0., 1., 0., 0., 0.],
       [0., 0., 1., 0., 0.],
       [0., 0., 0., 1., 0.],
       [0., 0., 0., 0., 1.]])
```

In [23]:
```
A = np.array([[1, 2, 3], [4, 5, 6], [7, 8, 9]])
A
```

Out[23]:
```
array([[1, 2, 3],
       [4, 5, 6],
       [7, 8, 9]])
```

In [24]:
```
np.ravel(A)
```

Out[24]:
```
array([1, 2, 3, 4, 5, 6, 7, 8, 9])
```

In [25]:
```
np.ravel(A, order='C')
```

Out[25]:
```
array([1, 2, 3, 4, 5, 6, 7, 8, 9])
```

In [26]:
```
np.ravel(A, order='F')
```

Out[26]:
```
array([1, 4, 7, 2, 5, 8, 3, 6, 9])
```

In [27]:
```
a = np.array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9])
np.where(a%2 ==0, a*10, a/10)
```

Out[27]:
```
array([ 0. ,  0.1, 20. ,  0.3, 40. ,  0.5, 60. ,  0.7, 80. ,  0.9])
```

In [28]:
```
a = np.random.rand(10)
a
```

Out[28]:
```
array([0.76215256, 0.06758131, 0.38671262, 0.12612126, 0.89707673,
       0.39262836, 0.15868646, 0.12567919, 0.55760012, 0.45380386])
```

In [29]:
```
x = np.linspace(0, 1, 5)
x
```

Out[29]:
```
array([0.  , 0.25, 0.5 , 0.75, 1.  ])
```

In [30]:
```
y = np.linspace(0, 2, 5)
y
```

Out[30]:
```
array([0. , 0.5, 1. , 1.5, 2. ])
```

In [31]:
```
xg, yg = np.meshgrid(x, y)
xg
```

Out[31]:
```
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.  ]])
```

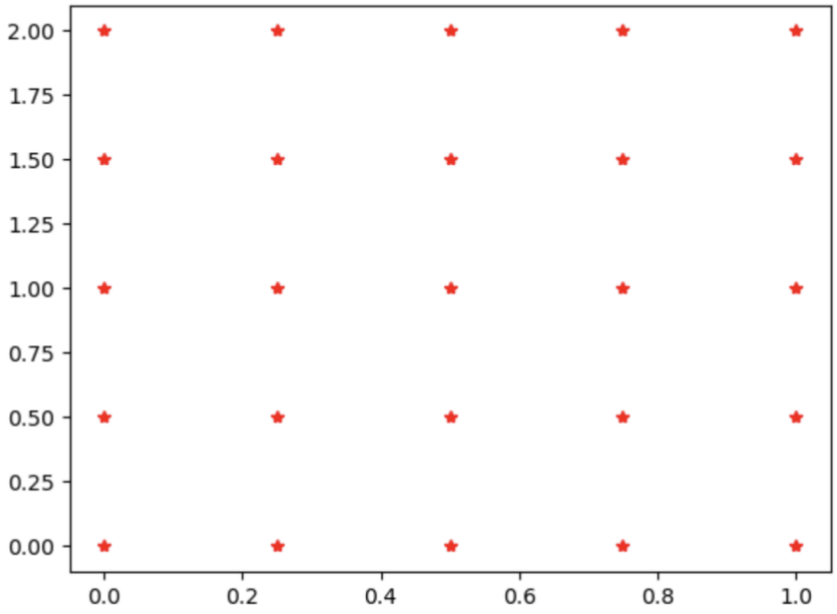
In [32]:
```
yg
```

Out[32]:
```
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 [33]: import matplotlib.pyplot as plt
         %matplotlib inline

In [34]: plt.plot(xg, yg, color="r", marker="*", linestyle="none")

Out[34]: [<matplotlib.lines.Line2D at 0x11605c700>,
          <matplotlib.lines.Line2D at 0x11605c850>,
          <matplotlib.lines.Line2D at 0x11605c970>,
          <matplotlib.lines.Line2D at 0x11605ca90>,
          <matplotlib.lines.Line2D at 0x11605cbb0>]
```




```
In [35]: np.random.permutation(7)

Out[35]: array([1, 5, 4, 0, 3, 6, 2])

In [36]: a = ['a', 'b', 'c', 'd', 'e']
         np.random.permutation(a)

Out[36]: array(['e', 'b', 'c', 'd', 'a'], dtype='<U1')

In [37]: arr = np.linspace(0, 10, 5)
         arr

Out[37]: array([ 0. ,  2.5,  5. ,  7.5, 10. ])

In [38]: arr_mix = np.random.permutation(arr)
         arr_mix

Out[38]: array([ 2.5,  7.5, 10. ,  5. ,  0. ])

In [39]: index_mix = np.random.permutation(len(arr_mix))
         index_mix

Out[39]: array([3, 4, 2, 0, 1])

In [40]: arr[index_mix]

Out[40]: array([ 7.5, 10. ,  5. ,  0. ,  2.5])
```

Рисунки 4 – 10 – Примеры лабораторной работы

7. Решить задания в ноутбуках, выданных преподавателем.

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 [5]:
```
c = np.array([4, 8, 15, 16, 23])
print('c =', c)
```

```
c = [ 4  8 15 16 23]
```

Арифметические операции, в отличие от операций над списками, применяются поэлементно:

In [6]:
```
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 [7]:
```
c = np.array([4, 8, 15, 16, 23])
print('c ** 3 =', c ** 3)
```

```
c ** 3 = [   64   512  3375  4096 12167]
```

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

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

In [9]:
```
# 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 [10]:
```
m1 = np.array([4, 8, 15, 16, 23])
m2 = np.array([5, 6, 55, 2, 1])
print("a / b =", m1 / m2)
```

```
a / b = [ 0.8         1.33333333  0.27272727  8.         23.        ]
```

Л — логика

К элементам массива можно применять логические операции.

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

In [11]:
```
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]

b = [0.1 0.2 0.3 0.4 0.5]
        b < 0.5:   [ True  True  True  True False]

Одновременная проверка условий:
        (a > 1) & (b < 0.5):   [False  True  True  True False]
А вот это проверяет, что a > 1 ИЛИ b < 0.5:   [ True  True  True  True  True]
```

Создайте 2 массива из 5 элементов. Проверьте условие "Элементы первого массива меньше 6, элементы второго массива делятся на 3"

In [14]:
```
print("Проверка 1-го массива: ", m1 < 6)
print("Проверка 2-го массива: ", m2 % 3 == 0)
```

```
Проверка 1-го массива:  [ True False False False False]
Проверка 2-го массива:  [False  True False False False]
```

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

In [15]:
```
print((m1 % 2 == 0) & (b > 2))
```

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

In [ ]:
```
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])
```

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 [18]:
```
m3 = np.arange(1, 21)
print(m3[np.logical_and((m3 > 5), (m3 % 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:

In [ ]:
```
# 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(['male', 'male', 'female', 'female', 'm

# и ещё много всяких методов
# Google в помощь
```

Пора еще немного потренироваться с NumPy.

Выполните операции, перечисленные ниже:

In [19]:
```
print("Разность между a и b:", (a - b)
      )
print("Квадраты элементов b:", b**2
      )
print("Половины произведений элементов массивов a и b:", a * b / 2
      )

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

```
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: [0.05 0.2  0.45 0.8  1.25]

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

Задайте два массива: [5, 2, 3, 12, 4, 5] и ['f', 'o', 'o', 'b', 'a', 'r']

Выведите буквы из второго массива, индексы которых соответствуют индексам чисел из первого массива, которые больше 1, меньше 5 и делятся на 2

In [20]:
```
q = np.array([5, 2, 3, 12, 4, 5])
w = np.array(['f', 'o', 'o', 'b', 'a', 'r'])

print(w[np.logical_and(q > 1, q < 5, q % 2 == 0)])
```

```
['o' 'o' 'a']
```

Рисунки 11 – 16 – Выполнение заданий ноутбука lab3.2

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
```

In [3]:
```
a = np.arange(2,13,2)
b = np.array([7,11,15,18,23,29])
print((a + b))
print((a + b)**2)
```

```
[ 9 15 21 26 33 41]
[  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]:
```
print(a[np.logical_and(b > 12, b % 5 == 3)])
```

```
[ 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 [5]:
```
c = np.logical_and(a % 4 == 0, b < 14)
c
```

Out[5]:
```
array([False,  True, False, False, False, False])
```

Рисунок 17 – Выполнение задания 1 ноутбука lab3.2hw

- 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 [12]:
```
import csv
import numpy as np
```

In [13]:
```
with open('organizations_gdp_hist.csv', 'r', newline='') as csvfile:
    data = csv.reader(csvfile, delimiter=',')
    total = []
    years = []
    for row in data:
        if row[4] == "Ingreso mediano alto":
            total.append(float(row[6]))
            years.append(float(row[5]))
```

In [14]:
```
arr_t = np.array(total)
arr_y = np.array(years)
```

In [16]:
```
print(f"Среднее значение ВВП в Ингресо медиано альт: {np.mean(arr_t)}")
print(f"Средний год исследования: {np.mean(arr_y)}")
```

```
Среднее значение ВВП в Ингресо медиано альт: 1531038886.3048372
Средний год исследования: 1990.5
```

In [18]:
```
print(f"Минимальное значение ВВП в Ингресо медиано альт: {np.min(arr_t)}")
print(f"Минимальный год исследования: {np.min(arr_y)}")
```

```
Минимальное значение ВВП в Ингресо медиано альт: 0.0
Минимальный год исследования: 1960.0
```

In [19]:
```
print(f"Максимальное значение ВВП в Ингресо медиано альт: {np.max(arr_t)}")
print(f"Максимальный год исследования: {np.max(arr_y)}")
```

```
Максимальное значение ВВП в Ингресо медиано альт: 9412034299.23122
Максимальный год исследования: 2021.0
```

Рисунок 18 – Выполнение задания 2 ноутбука lab3.2hw

8. 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. Привести в ноутбуке обоснование принятых решений. Номер варианта индивидуального задания необходимо уточнить у преподавателя.

Вариант 9.

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.

In [17]:
```
import numpy as np
```

Создаем квадратную матрицу, размерностью 10 на 10

In [18]:
```
array = np.random.randint(-10, 10,(10, 10))
array
```

Out[18]:
```
array([[ -6,   6,   7,  -3,   9,   1,  -1,   9,  -6,   7],
       [ -5,  -3,   8,  -6,   7,  -1,   9,   1,   9,  -3],
       [ -7,  -3,   2,   8,   9,  -5,   3,  -9,  -4,  -9],
       [ -5,  -6,   1,   3, -10,  -2,  -5,  -1,   6,   3],
       [  5,   7,  -1,   2,  -5,  -2,  -1,   9,   3, -10],
       [  7,   0,   7,   3,  -1,  -4,   8,   0, -10,   5],
       [  3,   6,  -5,   9,   2,  -5,  -4,  -3,   6,   3],
       [ -3,  -3,   9, -10,   6,   2,  -7,  -8,  -1,   4],
       [ -7,   6,   1,  -7,  -8,  -3,   4,   1,   0,   5],
       [ -9,  -8,   7,   8,   3,  -8,   3,   7,   0,  -8]])
```

Подсчитать количество локальных минимумов заданной матрицы размером 10 на 10

Используем функцию для поиска локальных минимумов

In [19]:
```
def local_minima(array_m):
    return ((array_m < np.roll(array_m,  1, 0)) &
            (array_m < np.roll(array_m, -1, 0)) &
            (array_m < np.roll(array_m,  1, 1)) &
            (array_m < np.roll(array_m, -1, 1)))

b = local_minima(array)
print(b)
```

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

Считаем количество локальных минимумов

In [20]:
```
count = np.where((b == True), 1, 0).sum()
print(count)
```

```
20
```

Найти сумму модулей элементов, расположенных выше главной диагонали.

Находим модули элементов матрицы

In [21]:
```
modul = np.absolute(array)
print(modul)
```

```
[[ 6  6  7  3  9  1  1  9  6  7]
 [ 5  3  8  6  7  1  9  1  9  3]
 [ 7  3  2  8  9  5  3  9  4  9]
 [ 5  6  1  3 10  2  5  1  6  3]
 [ 5  7  1  2  5  2  1  9  3 10]
 [ 7  0  7  3  1  4  8  0 10  5]
 [ 3  6  5  9  2  5  4  3  6  3]
 [ 3  3  9 10  6  2  7  8  1  4]
 [ 7  6  1  7  8  3  4  1  0  5]
 [ 9  8  7  8  3  8  3  7  0  8]]
```

Находим сумму элементов, расположенных выше главной диагонали, при помощи функции trace(), которая позволяет получить сумму по главной диагонали со смещением offset

In [22]:
```
summa = sum([modul.trace(offset=i) for i in range(0,11)])
summa
```

Out[22]:
```
280
```

Рисунок 19 – Выполнение индивидуального задания

9. Зафиксируйте сделанные изменения в репозитории.

10. Создать ноутбук, в котором выполнить решение вычислительной задачи (например, задачи из области физики, экономики, математики, статистики и т. д.), условие которой предварительно необходимо согласовать с преподавателем.

Вычислить среднюю абсолютную ошибку

Дана модель с трендом и сезонностью, вычислить среднюю абсолютную ошибку в процентах за первые 8 периодов, для оценки точности прогноза.

$$MAE = \frac{1}{n}\sum_{i=1}^{n}\frac{|Y_i - \hat{Y}_i|}{Y_i}$$

Y – фактический объем продаж за анализируемый период;

Ŷ — значение прогнозной модели за аналазируемый период;

n — количество периодов.

In [24]:
```
import numpy as np
```

In [32]:
```
def mape(actual, pred):
    actual, pred = np.array(actual), np.array(pred)
    return np.mean(np.abs((actual - pred) / actual))

actual = [71389948, 72030972, 72671996, 73313020, 73954044, 74595068, 75236092, 75877116]
pred = [58062900, 69363777, 79020890, 77244336, 71054470, 74414780, 73881366, 74692113]

mape(actual, pred)
```

Out[32]:
```
0.05499298272449152
```

Рисунок 20 – Выполнение задания 10

11. Зафиксируйте сделанные изменения в репозитории.

12. Выполните слияние ветки для разработки с веткой main (master).

13. Отправьте сделанные изменения на сервер GitHub.

Ответы на вопросы:

1. Каково назначение библиотеки NumPy?

Библиотека NumPy предоставляет реализации вычислительных алгоритмов (в виде функций и операторов), оптимизированные для работы с многомерными массивами.

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. Однородный — потому что практически все объекты в нем одного размера или типа. На самом деле, тип данных определен другим объектом NumPy, который называется dtype (тип-данных). Каждый ndarray ассоциирован только с одним типом dtype.

3. Как осуществляется доступ к частям многомерного массива?

Элементы матрицы с заданными координатами: m[1,0]

Строка матрицы: m[1, :]

Столбец матрицы: m[:, 1]

Часть строки матрицы: m[1, 2:]

Часть столбца матрицы: m[0:2, 1]

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

Произвольные столбцы / строки матрицы: cols = [0, 1, 2]; m[:, cols]

4. Как осуществляется расчет статистик по данным?

Размерность массива: m.shape

Вызов функции расчёта статистики: m.max()

Расчёт статистики по строкам или столбцам массива: m.max(axis=1); m.max(axis=0)

Индексы элементов с максимальным значением (по осям): argmax

Индексы элементов с минимальными значением (по осям): argmin

Максимальные значения элементов (по осям): max

Минимальные значения элементов (по осям): min

Средние значения элементов (по осям): mean

Произведение всех элементов (по осям): prod

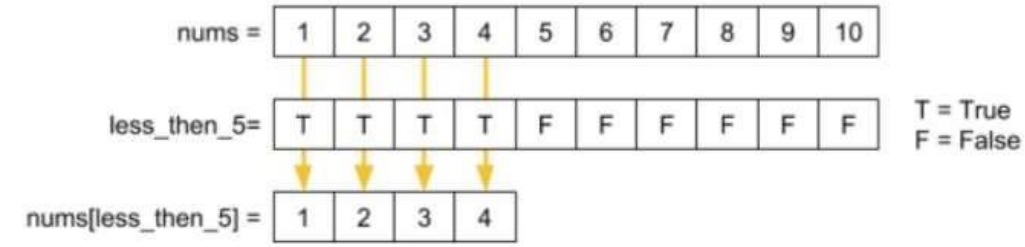
Стандартное отклонение (по осям): std

Сумма всех элементов (по осям): sum Дисперсия (по осям): var

5. Как выполняется выборка данных из массивов ndarray?

```
>>> 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])
```