

## 1. Примеры из методических указаний

```
Ввод [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]]
```

```
Ввод [3]: print(m[1, 0])
```

```
5
```

```
Ввод [4]: print(m[1, :])
```

```
[[5 6 7 8]]
```

```
Ввод [5]: print(m[:, 2])
```

```
[[3]  
 [7]  
 [5]]
```

```
Ввод [6]: print(m[1, 2:])
```

```
[[7 8]]
```

```
Ввод [7]: print(m[0:2, 1])
```

```
[[2]  
 [6]]
```

```
Ввод [8]: print(m[0:2, 1:3])
```

```
[[2 3]  
 [6 7]]
```

```
Ввод [9]: cols = [0, 1, 3]  
print(m[:, cols])
```

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

Расчет статистик по данным в массиве

Расчет статистик по данным в массиве

```
Ввод [11]: print(m)  
m.shape # Размерность массива
```

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

```
Out[11]: (3, 4)
```

```
Ввод [12]: np.max(m) # Максимальный элемент
```

```
Out[12]: 9
```

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

```
Ввод [14]: print(m.max(axis=1))
```

```
[[4]  
 [8]  
 [9]]
```

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

```
Ввод [15]: print(m.max(axis=0))
```

```
[[9 6 7 8]]
```

```
Ввод [16]: print(m.mean())  
print(m.mean(axis=1))  
print(m.sum())  
print(m.sum(axis=0))
```

```
4.833333333333333  
[[2.5]  
 [6.5]  
 [5.5]]  
58  
[[15 9 15 19]]
```

Использование boolean массива для доступа к ndarray

#### Использование boolean массива для доступа к ndarray

```
Ввод [17]: nums = np.array([1, 2, 3, 4, 5, 6, 7, 8, 9, 10])
letters = np.array(['a', 'b', 'c', 'd', 'a', 'e', 'b'])

Ввод [18]: less_than_5 = nums < 5
print(less_than_5)

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

Ввод [19]: pos_a = letters == 'a'
print(pos_a)

[ True False False False  True False False]

Ввод [20]: print(nums[less_than_5])

[1 2 3 4]

Ввод [21]: 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 [22]: mod_m = np.logical_and(m>=3, m<=7)
print(mod_m)
print(m[mod_m])

[[False False  True  True]
 [ True  True  True False]
 [False False  True  True]]
[[3 4 5 6 7 5 7]]

Ввод [23]: nums = np.array([1, 2, 3, 4, 5, 6, 7, 8, 9, 10])
print(nums[nums < 5])

[1 2 3 4]

Ввод [24]: nums[nums < 5] = 10
print(nums)

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

```
Ввод [25]: m[m > 7] = 25
print(m)

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

#### Дополнительные функции

np.arange()

```
Ввод [26]: print(np.arange(10))

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

```
Ввод [27]: print(np.arange(5, 12))

[ 5  6  7  8  9 10 11]
```

```
Ввод [28]: print(np.arange(1, 5, 0.5))

[1.  1.5 2.  2.5 3.  3.5 4.  4.5]
```

np.matrix()

Вариант со списком Python.

```
Ввод [29]: a = [[1, 2], [3, 4]]
print(np.matrix(a))

[[1 2]
 [3 4]]
```

Вариант с массивом Numpy.

```
Ввод [30]: b = np.array([[5, 6], [7, 8]])
print(np.matrix(b))

[[5 6]
 [7 8]]
```

Вариант в Matlab стиле.

Вариант в Matlab стиле.

Ввод [31]: `print(np.matrix('[1, 2; 3, 4]'))`

```
[[1 2]
 [3 4]]
```

`np.zeros(), np.eye()`

Ввод [32]: `print(np.zeros((3, 4)))`

```
[[0. 0. 0. 0.]
 [0. 0. 0. 0.]
 [0. 0. 0. 0.]]
```

Ввод [33]: `print(np.eye(5))`

```
[[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.]]
```

`np.ravel()`

Ввод [34]: `A = np.array([[1, 2, 3], [4, 5, 6], [7, 8, 9]])`  
`print(A)`

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

Ввод [35]: `np.ravel(A)`

Out[35]: `array([1, 2, 3, 4, 5, 6, 7, 8, 9])`

Ввод [37]: `>>> np.ravel(A, order='C')`

Out[37]: `array([1, 2, 3, 4, 5, 6, 7, 8, 9])`

Ввод [38]: `>>> np.ravel(A, order='F')`

Out[38]: `array([1, 4, 7, 2, 5, 8, 3, 6, 9])`

`np.where()`

Ввод [39]: `>>> a = np.array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9])`  
`>>> np.where(a % 2 == 0, a * 10, a / 10)`

Out[39]: `array([ 0. , 0.1, 20. , 0.3, 40. , 0.5, 60. , 0.7, 80. , 0.9])`

Ввод [44]: `a = np.random.rand(10)`  
`print(a)`  
`print(np.where(a > 0.5, True, False))`  
`np.where(a > 0.5, 1, -1)`

```
[0.79534005 0.92829777 0.02056662 0.8153471  0.96491854 0.07281233
 0.88941926 0.57618181 0.35753835 0.71187427]
[ True  True False  True  True False  True  True False  True]
```

Out[44]: `array([ 1, 1, -1, 1, 1, -1, 1, 1, -1, 1])`

`np.meshgrid()`

Ввод [46]: `>>> x = np.linspace(0, 1, 5)`  
`print(x)`  
`>>> y = np.linspace(0, 2, 5)`  
`>>> y`

```
[0.   0.25 0.5  0.75 1.   ]
```

Out[46]: `array([0. , 0.5, 1. , 1.5, 2. ])`

Ввод [47]: `>>> xg, yg = np.meshgrid(x, y)`  
`print(xg)`  
`>>> yg`

```
[[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.   ]]
```

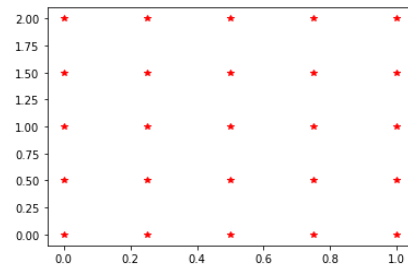
Out[47]: `array([[0. , 0. , 0. , 0. , 0. ],
 [0.5, 0.5, 0.5, 0.5, 0.5],
 [1. , 1. , 1. , 1. , 1. ],
 [1.5, 1.5, 1.5, 1.5, 1.5],
 [2. , 2. , 2. , 2. , 2. ]])`

Ввод [48]: `import matplotlib.pyplot as plt`

```
Ввод [48]: import matplotlib.pyplot as plt
           %matplotlib inline

Ввод [49]: plt.plot(xg, yg, color="r", marker="*", linestyle="none")
```

```
Out[49]: [<matplotlib.lines.Line2D at 0x25d44454520>,
          <matplotlib.lines.Line2D at 0x25d44454610>,
          <matplotlib.lines.Line2D at 0x25d444545e0>,
          <matplotlib.lines.Line2D at 0x25d44454670>,
          <matplotlib.lines.Line2D at 0x25d44454760>]
```



```
np.random.permutation()
```

```
Ввод [50]: print(np.random.permutation(7))
           >>> a = ['a', 'b', 'c', 'd', 'e']
           >>> np.random.permutation(a)
```

```
[2 6 0 3 1 4 5]
```

```
Out[50]: array(['d', 'c', 'b', 'e', 'a'], dtype='<U1'))
```

```
Ввод [51]: >>> arr = np.linspace(0, 10, 5)
           >>> arr
```

```
Out[51]: array([ 0. ,  2.5,  5. ,  7.5, 10. ])
```

```
Ввод [53]: >>> arr_mix = np.random.permutation(arr)
           >>> arr_mix
```

```
Out[53]: array([10. ,  0. ,  2.5,  5. ,  7.5])
```

## 2. Решение задач

### 2.1 Знакомство с NumPy.

Выполнение задания представлено на рисунках 1-5.

Создайте массив с 5 любыми числами:

```
Ввод [5]: c = np.array([23, 78, 95, 12, 1])
print("c = ", c)

c = [23 78 95 12 1]
```

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

```
Ввод [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

```
Ввод [7]: c = np.array([23, 78, 95, 12, 1])
print("c = ", c)
print('\tc ** 3:', c ** 3)

c = [23 78 95 12 1]
c ** 3: [12167 474552 857375 1728 1]
```

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

```
Ввод [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]
```

## Рисунок 1

```
Ввод [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.

```
Ввод [10]: arr1 = np.array([10, 45, 21, 7, 5])
arr2 = np.array([2, 3, 6, 25, 9])
print("arr1 / arr2 =", arr1 / arr2)

arr1 / arr2 = [ 5.      15.      3.5      0.28      0.55555556]
```

### Л — логика

К элементам массива можно применять логические операции.

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

```
Ввод [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]
```

## Рисунок 2

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"

```
Ввод [14]: arr1 = np.array([10, 45, 21, 7, 5])
arr2 = np.array([2, 3, 6, 25, 9])
print("arr1 =", arr1)
print("arr2 =", arr2)
print("\t(arr1 < 6) & (arr2 % 3 == 0): ", (arr1<6) & (arr2 % 3 == 0))

arr1 = [10 45 21  7  5]
arr2 = [ 2  3  6 25  9]
      (arr1 < 6) & (arr2 % 3 == 0): [False False False False  True]
```

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

```
Ввод [16]: print("(arr1 % 2 == 0) | (arr2 > 2): ", (arr1%2==0) | (arr2>2))

(arr1 % 2 == 0) | (arr2 > 2): [ True  True  True  True  True]
```

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

```
Ввод [17]: 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.

```
Ввод [25]: arr3 = np.arange(1,21)
print("arr3 =", arr3)
print("arr3[arr3 > 5 & arr3 % 2 != 0]:", arr3[np.logical_and(arr3 > 5, arr3 % 2 != 0)])

arr3 = [ 1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20]
arr3[arr3 > 5 & arr3 % 2 != 0]: [ 7  9 11 13 15 17 19]
```

## Рисунок 3

### 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:

```
Ввод [26]: # 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', 'male']))

# 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:

```
Ввод [27]: 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(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: [0.05 0.2 0.45 0.8 1.25]
```

Рисунок 4

```
Ввод [27]: 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(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: [0.05 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

Ввод [28]: arr = np.array([5, 2, 3, 12, 4, 5])
letters = np.array(['f', 'o', 'o', 'b', 'a', 'r'])
print(letters[np.logical_and(arr > 1, arr < 5, arr % 2 == 0)])

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

Рисунок 5

## 2.2 Домашнее задание (рис 6-7).

Задание №1 представлено на рисунке 6.

## 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.

```
Ввод [3]: import numpy as np
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.

```
Ввод [5]: 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)

```
Ввод [7]: print((a % 4 == 0) & (b < 14))

[False  True False False False False]
```

## Рисунок 6 – Задачи для закрепления пройденного материала

Задание №2 представлено на рисунке 7.

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

Для анализа выбран csv файл, содержащий информацию о росте мужчин и женщин по странам мира. Анализовались данные для топ 10 стран, в которых вычислялись статистические характеристики для мужчин и для женщин, а также в конце вычислен коэффициент корреляции для мужчин из топ 10 и мужчин из 10 последних стран по росту.

```
Ввод [4]: import csv
import numpy as np
```

```
Ввод [5]: with open('MF_height.csv', 'r', newline='') as csvfile:
    data = csv.reader(csvfile, delimiter=',')
    men_height = []
    women_height = []
    last10_men = []
    for row in data:
        if row[0] != "Rank" and int(row[0]) <= 10:
            men_height.append(float(row[2]))
            women_height.append(float(row[3]))
        if row[0] != "Rank" and 180 < int(row[0]) <= 190:
            last10_men.append(float(row[2]))
```

```
Ввод [6]: men_arr = np.array(men_height)
women_arr = np.array(women_height)
print(f"Среднее значение роста мужчин из топ 10 стран: {np.mean(men_arr)}")
print(f"Среднее значение роста женщин из топ 10 стран: {np.mean(women_arr)}")
print(f"Среднее отклонение для мужчин: {np.std(men_arr)}")
print(f"Среднее отклонение для женщин: {np.std(women_arr)}")
print(f"Коэффициент парной корреляции: {np.corrcoef(men_arr, women_arr)}")
print(
    "Коэффициент парной корреляции для мужчин из топ с 10-ю последними странами:"
    f"{np.corrcoef(men_arr, last10_men)}"
)

Среднее значение роста мужчин из топ 10 стран: 182.06900000000002
Среднее значение роста женщин из топ 10 стран: 168.59199999999998
Среднее отклонение для мужчин: 0.9499310501294319
Среднее отклонение для женщин: 1.080932930389304
Коэффициент парной корреляции: [[1.          0.73824827]
 [0.73824827 1.          ]]
Коэффициент парной корреляции для мужчин из топ с 10-ю последними странами: [[1.          0.98096368]
 [0.98096368 1.          ]]
```

## Рисунок 7 – Исследование датасета



## 2.3 Индивидуальное задание (рис. 8-9)

5. Дана целочисленная квадратная матрица. Определить: сумму элементов в тех столбцах, которые не содержат отрицательных элементов; минимум среди сумм модулей элементов диагоналей, параллельных побочной диагонали матрицы.

Инициализирую матрицу и импортирую NumPy

```
Ввод [2]: import numpy as np
```

```
Ввод [5]: n = int(input("Enter the size of the matrix: "))
```

Enter the size of the matrix: 5

```
Ввод [14]: matrix = np.random.randint(-20, 100, (n, n))  
print(matrix)
```

```
[[ 13  59  83  69  -4]  
 [ 32  33   4  20  33]  
 [ 49  34  25 -17  96]  
 [ -5   7  90  13  76]  
 [ 53  82  49  94  63]]
```

### 1. Сумма элементов в тех столбцах, которые не содержат отрицательных элементов

Вывожу массив boolean по столбцам, в которых минимальный элемент больше нуля:

```
Ввод [20]: print(matrix.min(axis=0) > 0)
```

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

Здесь я уже попытался вывести только столбцы матрицы, которые не содержали отрицательных элементов:

```
Ввод [25]: print(matrix[:, matrix.min(axis=0) > 0])
```

```
[[59 83]  
 [33  4]  
 [34 25]  
 [ 7 90]  
 [82 49]]
```

Узнав как вывести только столбцы матрицы, которые не содержали отрицательных элементов, можно рассчитать сумму элементов в этих столбцах.

Рисунок 8 – Выполнение первой части задания

Узнав как получить доступ к нужным мне столбцам, посчитал их сумму:

```
Ввод [26]: summa = np.sum(matrix[:, matrix.min(axis=0) > 0])  
print(summa)
```

466

## 2. Минимум среди сумм модулей элементов диагоналей, параллельных побочной диагонали матрицы

```
Ввод [2]: import numpy as np
```

```
Ввод [3]: n = int(input("Enter the size of the matrix: "))
```

Enter the size of the matrix: 5

```
Ввод [4]: matrix = np.random.randint(-20, 100, (n, n))  
print(matrix)
```

```
[[ 72  57 -11  22  32]  
 [ 90  93  99 -18  20]  
 [ 93  -1  84  57 -18]  
 [ 83  96   5  43  -2]  
 [ 16  23  24  86 -10]]
```

```
Ввод [5]: old_main_diagonal = matrix.diagonal().copy()  
np.fill_diagonal(matrix, np.flipud(matrix).diagonal())  
np.fill_diagonal(np.flipud(matrix), old_main_diagonal)  
print(matrix)
```

```
[[ 16  57 -11  22 -10]  
 [ 90  96  99  43  20]  
 [ 93  -1  84  57 -18]  
 [ 83  93   5 -18  -2]  
 [ 72  23  24  86  32]]
```

```
Ввод [11]: print(np.tril(matrix, -1))  
print(np.triu(matrix, 1))
```

```
[[ 0  0  0  0  0]  
 [90  0  0  0  0]  
 [93 -1  0  0  0]  
 [83 93  5  0  0]]
```

Рисунок 9 – Выполнение второй части задания

### 2.4 Вычислительная задача (рис. 10-12).

Найти давление воздуха в откачиваемом сосуде как функцию времени откачки  $t$ . Объем сосуда  $V = 100$  л. Процесс считать изотермическим и скорость откачки независимой от давления и равной  $C = 0.01$  л/с. Скоростью откачки называют объем газа, откачиваемый за единицу времени, причем этот объем измеряется при давлении газа в данный момент времени.

#### Решение

Рассмотрим изотермический процесс при изменении объема на  $dV$  и давления на  $dp$ , тогда  $pV = (p + dp)(V + dV)$ , или  $pV = pV + pdV + Vdp + dpdV$ ; пренебрегая последним слагаемым, мы получаем дифференциальное уравнение первого порядка с разделяющимися переменными  $dp/p = -dV/V$ . Учитывая, что изменение объема  $dV = Cdt$ , уравнение можно проинтегрировать:

$$\ln p - \ln p_0 = -\frac{C}{V}t, \text{ или } \frac{p}{p_0} = \exp\left(-\frac{C}{V}t\right). \quad (11)$$

Полученная формула является решением задачи.

**Анализ.** Представим решение (11) в безразмерном виде:

$$\eta = \exp(-\beta t), \quad (12)$$

где  $\eta = p/p_0$  и  $\beta = C/V$ . Это решение отличается от формулы (9). Для того чтобы сравнить (9) и (12), необходимо привести оба решения к одной области определения, т. е. перейти в (9) от безразмерного номера  $n$  цикла к соответствующему моменту времени  $t_n$ , где  $t_n = n\Delta t$ .

Для произвольного  $t_n$  выпишем два решения  $\eta_{n-1} = q^{n-1}$  и  $\eta_n = q^n$ , затем составим разностное уравнение  $\Delta\eta_n \equiv \eta_n - \eta_{n-1} = q^n - q^{n-1} = q^{n-1}(q - 1)$ , или  $\Delta\eta_n / \eta_n = 1 - q^{-1}$ . Найдем сумму от 0 до  $n$ :

$$\sum_{n=0}^n \frac{\Delta\eta_n}{\eta_n} = \sum_{n=0}^n (1 - q^{-1}).$$

## Рисунок 10 – Условие задания + словесное решение

Правая часть выражения есть  $(1 - q^{-1})n = (1 - q^{-1})t_n / \Delta t$ , а левую часть вычислим, учитывая, что  $\eta_0 = 1$ , и, переходя от суммирования к интегрированию, запишем:

$$\sum_{n=0}^n \frac{\Delta\eta_n}{\eta_n} \rightarrow \int_0^{\eta_n} \frac{d\eta}{\eta} = \ln \eta_n - \ln \eta_0 = \ln \eta_n.$$

Итак, выражение (9) записывается в виде

$$\ln \eta_n = (1 - q^{-1})t_n / \Delta t \text{ или } \eta_n = \exp(-\gamma t_n), \quad (13)$$

где  $\gamma_n = (q^{-1} - 1) / \Delta t$ , и совпадает с (12) в узлах  $t_n$  дискретной временной сетки.

```
Ввод [1]: import numpy as np
import matplotlib.pyplot as plt
import math
```

```
Ввод [5]: t=[]; p1=[]; p2=[]
N=20; C=0.01; tmax=50.0; V=0.1; p0=1.0
dV=0.0005
dt=tmax/N
qV=V/(V+dV)
b1=C/V
dt1=V*(1.0/qV-1.0)/C
print (" q =",qV," b1=",b1," \Delta t=",dt1)
```

```
q = 0.9950248756218906 b1= 0.09999999999999999 \Delta t= 0.04999999999999998934
```

```
Ввод [6]: t.append(0); p1.append(p0); p2.append(p0)
for i in range(1,N):
    t1=i*dt; t.append(t1)
    p1.append(math.exp(-b1*t1))
    p2.append(qV**(t1/dt1))
plt.plot(t,p1,'k-')
plt.plot(t,p2,'ko:')
plt.xlabel('$t$', fontsize=14)
plt.ylabel('$\eta_1$, $\eta_2$', fontsize=14)
plt.show()
```



## Рисунок 11 – Часть кода программы

```
Ввод [6]: t.append(0); p1.append(p0); p2.append(p0)
for i in range(1,N):
    t1=i*dt; t.append(t1)
    p1.append(math.exp(-b1*t1))
    p2.append(qv**(t1/dt1))
plt.plot(t,p1,'k-')
plt.plot(t,p2,'ko:')
plt.xlabel('$t$', fontsize=14)
plt.ylabel('$\eta_1, \eta_2$', fontsize=14)
plt.show()
```

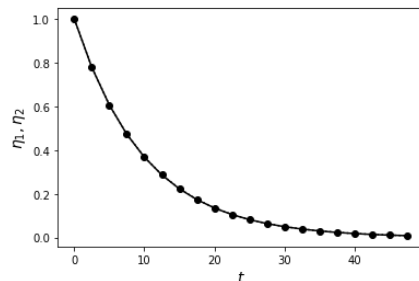


Рисунок 1 - Зависимость приведенного давления от времени откачки t

## Рисунок 12 – Заключительная часть кода программы + график решений

### 3. Ответы на 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?

Ndarray - это (обычно фиксированный размер) многомерный контейнер элементов одного типа и размера. Количество измерений и элементов в массиве определяется его формой, которая является кортежем из N натуральных чисел, которые определяют размеры каждого измерения.

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

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

`m[1, 0]`

Строка матрицы

`m[1, :]`

Столбец матрицы

`m[:, 2]`

Часть строки матрицы

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:]`

Часть столбца матрицы

`>>> m[0:2, 1]`

Непрерывная часть матрицы

`m[0:2, 1:3]`

Произвольные столбцы / строки матрицы

`cols = [0, 1, 3]`

`m[:, cols]`

4. Как осуществляется расчет статистик по данным?

Размерность массива

`m.shape`

В результате мы получим кортеж из двух элементов, первый из них – это количество строк, второй – столбцов.

Вызов функции расчета статистики

Для расчета той или иной статистики, соответствующую функцию можно вызывать как метод объекта, с которым вы работаете. Для нашего массива это будет выглядеть так.

```
m.max()
```

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

```
m.max(axis=1)
```

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

```
m.max(axis=0)
```

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
------------	----------

<code>argmax</code>	Индексы элементов с максимальным значением (по осям)
---------------------	--

<code>argmin</code>	Индексы элементов с минимальным значением (по осям)
---------------------	---

<code>max</code>	Максимальные значения элементов (по осям)
------------------	---

<code>min</code>	Минимальные значения элементов (по осям)
------------------	--

<code>mean</code>	Средние значения элементов (по осям)
-------------------	--------------------------------------

<code>prod</code>	Произведение всех элементов (по осям)
-------------------	---------------------------------------

<code>std</code>	Стандартное отклонение (по осям)
------------------	----------------------------------

<code>sum</code>	Сумма всех элементов (по осям)
------------------	--------------------------------

<code>var</code>	Дисперсия (по осям)
------------------	---------------------

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

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. Для иллюстрации данной возможности воспользуемся массивом `nums`. Используя второй подход, можно построить на базе созданных нами в самом начале ndarray массивы с элементами типа `boolean`. В этом примере мы создали boolean массив, в котором на месте элементов из `nums`, которые меньше пяти стоит `True`, в остальных случаях – `False`. Построим массив, в котором значение `True` будут иметь элементы, чей индекс совпадает с индексами, на которых стоит символ 'a' в массиве `letters`. 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_than_5 = nums < 5
```

```
less_than_5
```

```
array([ True,  True,  True,  True, False, False, False, False, False, False])
```

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`.