МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ
ВЫСШЕГО ОБРАЗОВАНИЯ
«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»
ИНСТИТУТ ЦИФРОВОГО РАЗВИТИЯ

Отчет о лабораторной работе № 2 по дисциплине «Технологии распознавания образов»

Выполнил студент
2 курса, группы ПИЖ-б-о-20-1
Тотубалина С.С.
Проверил:
Доцент кафедры инфокоммуникаций,
Воронкин Р.А.

Ставрополь, 2022 г.

1. Отработка примеров работы с Jupyter Notebook

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]:
```
print(m[1, 0])
```

```
5
```

In [4]:
```
print(m[1, :])
```

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

In [5]:
```
print(m[:, 2])
```

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

In [6]:
```
print(m[1, 2:])
```

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

In [7]:
```
print(m[0:2, 1])
```

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

In [8]:
```
print(m[0:2, 1:3])
```

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

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

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

Расчёт статистик по данным в массиве

In [10]:
```
print(m)
m.shape # Размерность массива
```

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

Out[10]: (3, 4)

In [11]:
```
np.max(m) # Максимальный элеемент
```

Out[11]: 9

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

In [12]:
```
print(m.max(axis=1))
```

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

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

In [13]:
```
print(m.max(axis=0))
```

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

In [14]:
```
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

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

In [18]:
```
less_then_5 = nums < 5
print(less_then_5)
```

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

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

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

In [20]:
```
print(nums[less_then_5])
```

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

In [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]]
```

In [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]]
```

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

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

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

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

In [25]:
```
m[m > 7] = 25
print(m)
```

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

Дополнительные функции

np.arange()

In [26]:
```
print(np.arange(10))
```

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

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

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

In [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.

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

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

Вариант с массивом Numpy.

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

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

Вариант в Matlab стиле.

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

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

np.zeros(), np.eye()

In [32]:
```
print(np.zeros((3, 4)))
```

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

In [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()

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

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

In [35]:
```
np.ravel(A)
```

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

In [36]:
```
>>> np.ravel(A, order='C')
```

Out[36]:
```
array([1, 2, 3, 4, 5, 6, 7, 8, 9])
```

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

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

np.where()

In [38]:
```
>>> a = np.array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9])
>>> np.where(a % 2 == 0, a * 10, a / 10)
```

Out[38]:
```
array([ 0. ,  0.1, 20. ,  0.3, 40. ,  0.5, 60. ,  0.7, 80. ,  0.9])
```

In [39]:
```
a = np.random.rand(10)
print(a)
print(np.where(a > 0.5, True, False))
np.where(a > 0.5, 1, -1)
```

```
[0.07275365 0.56570807 0.90090954 0.90675735 0.14903458 0.80734187
 0.22755264 0.82028806 0.68569889 0.21028175]
[False  True  True  True False  True False  True  True False]
```

Out[39]:
```
array([-1,  1,  1,  1, -1,  1, -1,  1,  1, -1])
```

np.meshgrid()

In [40]:
```
>>> x = np.linspace(0, 1, 5)
print(x)
>>> y = np.linspace(0, 2, 5)
>>> y
```

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

Out[40]:
```
array([0. , 0.5, 1. , 1.5, 2. ])
```

In [41]:
```
>>> xg, yg = np.meshgrid(x, y)
print(xg)
>>> yg
```

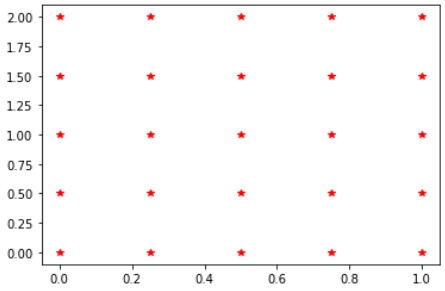
```
[[0.   0.25 0.5  0.75 1.  ]
 [0.   0.25 0.5  0.75 1.  ]
 [0.   0.25 0.5  0.75 1.  ]
 [0.   0.25 0.5  0.75 1.  ]
 [0.   0.25 0.5  0.75 1.  ]]
```

Out[41]:
```
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 [42]:
```
import matplotlib.pyplot as plt
%matplotlib inline
```

In [43]:
```
plt.plot(xg, yg, color="r", marker="*", linestyle="none")
```

Out[43]:
```
[<matplotlib.lines.Line2D at 0x215261c8ac0>,
 <matplotlib.lines.Line2D at 0x215261c8b20>,
 <matplotlib.lines.Line2D at 0x215261c8c10>,
 <matplotlib.lines.Line2D at 0x215261c8d30>,
 <matplotlib.lines.Line2D at 0x215261c8e50>]
```



np.random.permutation()

In [44]:
```
print(np.random.permutation(7))
>>> a = ['a', 'b', 'c', 'd', 'e']
>>> np.random.permutation(a)
```

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

Out[44]:
```
array(['b', 'c', 'e', 'a', 'd'], dtype='<U1')
```

In [45]:
```
>>> arr = np.linspace(0, 10, 5)
>>> arr
```

Out[45]:
```
array([ 0. ,  2.5,  5. ,  7.5, 10. ])
```

In [46]:
```
>>> arr_mix = np.random.permutation(arr)
>>> arr_mix
```

Out[46]:
```
array([ 2.5,  7.5,  0. , 10. ,  5. ])
```

In [47]:
```
index_mix = np.random.permutation(len(arr_mix))
print(index_mix)
arr[index_mix]
```

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

Out[47]:
```
array([ 0. ,  2.5,  5. , 10. ,  7.5])
```

2. Решение задач

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 [6]:
```
c = np.array([8, 12, 96, 45, 3])
print("c = ", c)
```

```
c =  [ 8 12 96 45  3]
```

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

In [7]:
```
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 [8]:
```
c = np.array([8, 12, 96, 45, 3])
print("c = ", c)
print('\tc ** 3:', c ** 3)
```

```
c =  [ 8 12 96 45  3]
        c ** 3: [   512   1728 884736  91125     27]
```

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

In [9]:
```
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 [10]:
```
# 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 [11]:
```
arr1 = np.array([8, 12, 96, 45, 3])
arr2 = np.array([2, 9, 6, 11, 3])
print("arr1 / arr2 =", arr1 / arr2)
```

```
arr1 / arr2 = [ 4.          1.33333333 16.          4.09090909  1.        ]
```

Л — логика

К элементам массива можно применять логические операции.

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

In [12]:
```
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 [13]:
```
arr1 = np.array([8, 12, 96, 45, 3])
arr2 = np.array([2, 9, 6, 11, 3])
print("arr1 =", arr1)
print("arr2 =", arr2)
print("\t(arr1 < 6) & (arr2 % 3 == 0):  ", (arr1<6) & (arr2 % 3 == 0))
```

```
arr1 = [ 8 12 96 45  3]
arr2 = [ 2  9  6 11  3]
        (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"

In [14]:
```
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:

In [15]:
```
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 [16]:
```
arr3 = np.arange(9, 22)
print("arr3 =", arr3)
print("arr3[arr3 > 5 & arr3 % 2 != 0]:", arr3[np.logical_and(arr3 > 5, arr3 % 2 != 0)])
```

```
arr3 = [ 9 10 11 12 13 14 15 16 17 18 19 20 21]
arr3[arr3 > 5 & arr3 % 2 != 0]: [ 9 11 13 15 17 19 21]
```

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 [17]:
```
# 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:

In [20]:
```
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

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

3. Выполнение индивидуального задания

Индивидуальное задание

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

In [2]:
```
import numpy as np
```

In [8]:
```
n = int(input("Enter the size of the matrix: "))
```

```
Enter the size of the matrix: 3
```

In [9]:
```
matrix = np.random.randint(-20, 100, (n, n))
print(matrix)
```

```
[[ -4  95  52]
 [ 96  31  54]
 [-11  64  87]]
```

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

In [10]:
```
print(matrix.min(axis=0) > 0)
```

```
[False  True  True]
```

In [11]:
```
print(matrix[:, matrix.min(axis=0) > 0])
```

```
[[95 52]
 [31 54]
 [64 87]]
```

In [12]:
```
summa = np.sum(matrix[:, matrix.min(axis=0) > 0])
print(summa)
```

```
383
```

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

In [13]:
```
import numpy as np
```

In [14]:
```
n = int(input("Enter the size of the matrix: "))
```

```
Enter the size of the matrix: 6
```

In [15]:
```
matrix = np.random.randint(-20, 300, (n, n))
matrix = np.absolute(matrix)
print(matrix)
```

```
[[110 206 179 293 249 125]
 [235 182 227 114 142  14]
 [ 76 190  52 298 294 215]
 [258  28  75 224 226 172]
 [134 107 257  81 275 233]
 [ 88 199 236 122 239 188]]
```

In [16]:
```
min_sum = min([np.fliplr(matrix).trace(offset=i) for i in range(-(n-1), n)])
print(min_sum)
```

```
110
```

4. Решение индивидуальной задачи

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:

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

Полученная формула является решением задачи.

Анализ. Представим решение (11) в безразмерном виде:

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

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:

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

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:

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

Итак, выражение (9) записывается в виде

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

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

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

In [2]:
```
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= 0.049999999999998934
```

In [3]:
```
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()
```

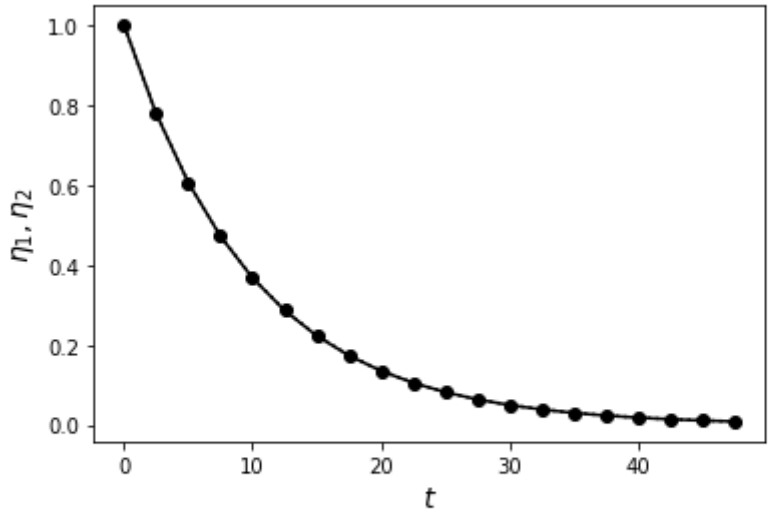


Рисунок 1 - Зависимость приведённого давления от времени откачки t

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 вызвать 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

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)

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 массивов 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_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.