Мухаметшин А.Р. ББМО-01-25

```
!git clone https://github.com/neuralcomputer/ML_School.git
```

```
Cloning into 'ML_School'...
remote: Enumerating objects: 94, done.
remote: Counting objects: 100% (15/15), done.
remote: Compressing objects: 100% (15/15), done.
remote: Total 94 (delta 5), reused 0 (delta 0), pack-reused 79 (from 1)
Receiving objects: 100% (94/94), 33.83 MiB | 20.95 MiB/s, done.
Resolving deltas: 100% (29/29), done.
```

```
a=5 # зададим число a
b=0.3 # зададим число b
a
```

```
5
```

```
c=a+7 # сложение чисел
c
```

```
12
```

```
a-b # вычитание чисел
```

```
4.7
```

```
a*b #Умножение чисел
```

```
1.5
```

```
a/b # деление чисел
```

```
16.666666666666668
```

```
b=0 # а если поделим на 0?
a/b # возникнет ошибка, на 0 делить нельзя
```

```
---------------------------------------------------------------------------
ZeroDivisionError                         Traceback (most recent call last)
/tmp/ipython-input-2965453836.py in <cell line: 0>()
      1 b=0 # а если поделим на 0?
----> 2 a/b # возникнет ошибка, на 0 делить нельзя

ZeroDivisionError: division by zero
```

```
import math
```

```
# Округление
a=3.2
b=math.ceil(a) # округление к ближайшему большему целому.
c=math.floor(a) #округление вниз
b, c
```

```
(4, 3)
```

```
a=-7
d=math.fabs(a)
d
```

```
7.0
```

```
a=7
b=2
math.fmod(a,b)
```

```
1.0
```

```
math.pi
```

```
3.141592653589793
```

```
math.e
```

```
2.718281828459045
```

```
a=3
b=4
math.pow(a, b)
```

```
81.0
```

```
a=16
math.sqrt(a)
# Корень кубический.
a=8
math.pow(a,1/3)
```

```
2.0
```

```
b=10
math.exp(b)
```

```
22026.465794806718
```

```
math.pow(math.e,b)
```

```
22026.465794806703
```

```
c=81
a=3
#math.log(x[, base])
math.log(c, a)
```

```
4.0
```

```
c=4
math.log2(c)
```

```
2.0
```

```
c=100
math.log10(c)
```

```
2.0
```

```
c=math.e*math.e
math.log(c)
```

```
2.0
```

```
x = math.pi/4
```

```
math.cos(x)
```

```
0.7071067811865476
```

```
math.sin(x)
```

```
0.7071067811865475
```

```
#x = math.pi/2 #
math.tan(x) # а если x=math.pi/2?
```

```
0.9999999999999999
```

```
x = math.pi/2 #
math.tan(x) # а если x=math.pi/2?
```

```
1.633123935319537e+16
```

```
x=1
math.atan(x)/math.pi
```

```
0.25
```

```
x=1000000000000
math.tanh(x)
```

```
1.0
```

```
a=2
math.factorial(a)
```

```
2
```

```
# Функция для перевода градусов в радианы
def degrees_to_radians(degrees):
    return degrees * (math.pi / 180)

# Функция для перевода радианов в градусы
def radians_to_degrees(radians):
    return radians * (180 / math.pi)

# Примеры использования
degrees = 180
radians = degrees_to_radians(degrees)
print(f"{degrees} градусов = {radians} радиан")

radians = math.pi
degrees = radians_to_degrees(radians)
print(f"{radians} радиан = {degrees} градусов")
```

```
180 градусов = 3.141592653589793 радиан
3.141592653589793 радиан = 180.0 градусов
```

```
import numpy as np
```

```
a = np.array([1, 2, 3])
type(a)
```

```
numpy.ndarray
```

```
a.shape
```

```
(3,)
```

```
b = np.array([[1.5, 2, 3], [4, 5, 6]])
b
```

```
array([[1.5, 2. , 3. ],
       [4. , 5. , 6. ]])
```

```
b.shape
```

```
(2, 3)
```

```
np.zeros((3, 5)) # двумерный массив из нулей, размером 3 на 5
```

```
array([[0., 0., 0., 0., 0.],
       [0., 0., 0., 0., 0.],
       [0., 0., 0., 0., 0.]])
```

```
np.ones((2, 3, 4)) # трехмерный массив из единиц размером 2 на 3 на 4
```

```
array([[[1., 1., 1., 1.],
        [1., 1., 1., 1.],
        [1., 1., 1., 1.]],

       [[1., 1., 1., 1.],
        [1., 1., 1., 1.],
        [1., 1., 1., 1.]]])
```

```
np.eye(5) # создаем единичную матрицу
```

```
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.]])
```

```
np.empty((3, 3))
```

```
array([[2.01162260e-315, 0.00000000e+000, 6.77272909e-310],
       [6.77272909e-310, 2.78653024e-321, 1.63041663e-322],
       [1.99838123e-315, 6.77277642e-310, 1.18575755e-321]])
```

```
np.empty((3, 2))
```

```
array([[1.5, 2. ],
       [3. , 4. ],
       [5. , 6. ]])
```

```
a=np.arange(20, 30, 5)
np.arange(1, 0, -0.1)
```

```
array([1. , 0.9, 0.8, 0.7, 0.6, 0.5, 0.4, 0.3, 0.2, 0.1])
```

```
a=np.arange(1,10,0.001)
print(a)
```

```
[1.    1.001 1.002 ... 9.997 9.998 9.999]
```

```
np.set_printoptions(threshold=10000, precision=2) # будем выводить до 10 десяти
print(a)
```

```
np.set_printoptions(threshold=1000, precision=8) # вернем параметры по умолчани
print(a)
```

```
  9.99  9.99  9.99  9.99  9.99  9.99  9.99  9.99 10.   10.   10.   10.  ]
[1.    1.001 1.002 ... 9.997 9.998 9.999]
```

```
a = np.array([20, 30, 40, 50]) # создадим один массив
a
```

```
array([20, 30, 40, 50])
```

```
b = np.arange(4) # создадим другой массив
b
```

```
array([0, 1, 2, 3])
```

```
c=a+b # сложим их
c
```

```
array([20, 31, 42, 53])
```

```
d=np.array([2,5])
a+d # размер не подходит для расширения
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
/tmp/ipython-input-1564389262.py in <cell line: 0>()
      1 d=np.array([2,5])
----> 2 a+d # размер не подходит для расширения

ValueError: operands could not be broadcast together with shapes (4,) (2,)
```

```
c=a - b # вычтем
c
```

```
array([20, 29, 38, 47])
```

```
c=a*b # умножим поэлементно
c
```

```
array([  0,  30,  80, 150])
```

```
c=a/b # поделим поэлементно
c
```

```
/tmp/ipython-input-3468064263.py:1: RuntimeWarning: divide by zero encountered i
  c=a/b # поделим поэлементно
array([       inf, 30.        , 20.        , 16.66666667])
```

```
np.inf + np.inf
```

```
inf
```

```
-1*np.inf
```

```
-inf
```

```
np.inf * np.inf
```

```
inf
```

```
np.inf - np.inf
```

```
nan
```

```
np.nan*(-1)
```

```
nan
```

```
np.nan+np.nan
```

```
nan
```

```
np.nan+5
```

```
nan
```

```
a ** b # возведение в степень, первый аргумент - основание, второй - степень.
```

```
array([     1,     30,   1600, 125000])
```

```
a % b # Взятие остатка от деления (при взятии остатка от деления на 0 возвращае
```

```
/tmp/ipython-input-900318961.py:1: RuntimeWarning: divide by zero encountered in
  a % b # Взятие остатка от деления (при взятии остатка от деления на 0 возвраща
array([0, 0, 0, 2])
```

```
a + 1
```

```
array([21, 31, 41, 51])
```

```
1 + b
```

```
array([1, 2, 3, 4])
```

```
a ** 3
```

```
array([  8000,  27000,  64000, 125000])
```

```
2 ** b
```

```
array([1, 2, 4, 8])
```

```
a=np.ones((3,1))
b=2*np.transpose(a)
print(a.shape, b.shape)
```

```
(3, 1) (1, 3)
```

```
a*b # ?????
```

```
array([[2., 2., 2.],
       [2., 2., 2.],
       [2., 2., 2.]])
```

```
a=3*np.ones((4,4))
b=-1*np.ones((4,1))
c=a+b
c
```

```
array([[2., 2., 2., 2.],
       [2., 2., 2., 2.],
       [2., 2., 2., 2.],
       [2., 2., 2., 2.]])
```

```
np.cos(a) # косинус
```

```
array([[-0.9899925, -0.9899925, -0.9899925, -0.9899925],
       [-0.9899925, -0.9899925, -0.9899925, -0.9899925],
       [-0.9899925, -0.9899925, -0.9899925, -0.9899925],
       [-0.9899925, -0.9899925, -0.9899925, -0.9899925]])
```

```
np.arctan(a) # арктангенс, название отличается от модуля math
```

```
array([[1.24904577, 1.24904577, 1.24904577, 1.24904577],
       [1.24904577, 1.24904577, 1.24904577, 1.24904577],
       [1.24904577, 1.24904577, 1.24904577, 1.24904577],
       [1.24904577, 1.24904577, 1.24904577, 1.24904577]])
```

```
np.tanh(a) # гипертангенс
```

```
array([[0.99505475, 0.99505475, 0.99505475, 0.99505475],
       [0.99505475, 0.99505475, 0.99505475, 0.99505475],
       [0.99505475, 0.99505475, 0.99505475, 0.99505475],
       [0.99505475, 0.99505475, 0.99505475, 0.99505475]])
```

```
c=5*np.cos(a) #
c
```

```
array([[-4.94996248, -4.94996248, -4.94996248, -4.94996248],
       [-4.94996248, -4.94996248, -4.94996248, -4.94996248],
       [-4.94996248, -4.94996248, -4.94996248, -4.94996248],
       [-4.94996248, -4.94996248, -4.94996248, -4.94996248]])
```

```
np.round(c,2)
```

```
array([[-4.95, -4.95, -4.95, -4.95],
       [-4.95, -4.95, -4.95, -4.95],
       [-4.95, -4.95, -4.95, -4.95],
       [-4.95, -4.95, -4.95, -4.95]])
```

```
np.ceil(c)
```

```
array([[-4., -4., -4., -4.],
       [-4., -4., -4., -4.],
       [-4., -4., -4., -4.],
       [-4., -4., -4., -4.]])
```

```
np.fix(c)
```

```
array([[-4., -4., -4., -4.],
       [-4., -4., -4., -4.],
       [-4., -4., -4., -4.],
       [-4., -4., -4., -4.]])
```

```
a = np.array([[1, 2, 3], [4, 5, 6]]) # создадим массив
a
```

```
array([[1, 2, 3],
       [4, 5, 6]])
```

```
np.sum(a) # суммирование всех элементов массива между собой
```

```
np.int64(21)
```

```
a.sum() # другой способ вызова той же функции суммирования
a.min() # минимальный элемент массива
```

```
np.int64(1)
```

```
a.max() # максимальный элемент массива
a.max() # максимальный элемент массива
```

```
np.int64(6)
```

```
a.min(axis=1) # минимум по строкам для каждого столбца
```

```
array([1, 4])
```

```
a.min(axis=0) # минимум по столбцам для каждой строки
```

```
array([1, 2, 3])
```

```
a = np.arange(10) ** 3 # создадим массив. Какая у него форма?
a
```

```
array([  0,   1,   8,  27,  64, 125, 216, 343, 512, 729])
```

```
a[4] # Посмотрим на первый элемент этого массива, помним что первый элемент име
```

```
np.int64(64)
```

```
a[3:5] # Посмотрим сразу на четвертый и пятый элементы. Помним что в срезах кон
```

```
array([27, 64])
```

```
a[[3,7]]
```

```
array([ 27, 343])
```

```
a[[3,3]]
```

```
array([27, 27])
```

```
a[3:5] = 2 # изменяем
a
```

```
array([  0,   1,   8,   2,   2, 125, 216, 343, 512, 729])
```

```
a[0.5] # такие индексы невозможны
```

```
---------------------------------------------------------------------------
IndexError                                Traceback (most recent call last)
/tmp/ipython-input-1672302078.py in <cell line: 0>()
----> 1 a[0.5] # такие индексы невозможны

IndexError: only integers, slices (`:`), ellipsis (`...`), numpy.newaxis
(`None`) and integer or boolean arrays are valid indices
```

```
a[::-1] # Что здесь происходит?
```

```
array([729, 512, 343, 216, 125,   2,   2,   8,   1,   0])
```

```
a
a[-1]
```

```
np.int64(729)
```

```
b = np.array([[ 0, 1, 2, 3],
... [10, 11, 12, 13],
... [20, 21, 22, 23],
... [30, 31, 32, 33],
... [40, 41, 42, 43]]) # создадим двумерный массив
b
```

```
array([[ 0,  1,  2,  3],
       [10, 11, 12, 13],
       [20, 21, 22, 23],
       [30, 31, 32, 33],
       [40, 41, 42, 43]])
```

```
b[2,3] # обратимся к элементу на третьей строке в четвертом столбце
```

```
np.int64(23)
```

```
b[[2,3],3]
```

```
array([23, 33])
```

```
b[(2,3)]
```

```
np.int64(23)
```

```
b[2]
```

```
array([20, 21, 22, 23])
```

```
b[2][3]
```

```
np.int64(23)
```

```
b[:,2] # это третий столбец целиком (все строки)
```

```
array([ 2, 12, 22, 32, 42])
```

```
b[:]
```

```
array([[ 0,  1,  2,  3],
       [10, 11, 12, 13],
       [20, 21, 22, 23],
       [30, 31, 32, 33],
       [40, 41, 42, 43]])
```

```
b[: 2] # это первая и вторая строки массива
```

```
array([[ 0,  1,  2,  3],
       [10, 11, 12, 13]])
```

```
b[1:3, : : ] # что это за смайлик?
```

```
array([[10, 11, 12, 13],
       [20, 21, 22, 23]])
```

```
b
b[2:3][0] #????
```

```
array([20, 21, 22, 23])
```

```
e=np.ones((2,3,4,2,3,4))
```

```
e[...,2] # даже такое работает, опустили все предыдущие индексы
```

```
[[1., 1., 1.],
 [1., 1., 1.]],

[[1., 1., 1.],
 [1., 1., 1.]]],


[[[1., 1., 1.],
  [1., 1., 1.]],

 [[1., 1., 1.],
  [1., 1., 1.]],

 [[1., 1., 1.],
  [1., 1., 1.]],

 [[1., 1., 1.],
  [1., 1., 1.]]],



[[[[1., 1., 1.],
   [1., 1., 1.]],

  [[1., 1., 1.],
   [1., 1., 1.]],

  [[1., 1., 1.],
   [1., 1., 1.]],

  [[1., 1., 1.],
   [1., 1., 1.]]],


 [[[1., 1., 1.],
   [1., 1., 1.]],

  [[1., 1., 1.],
```

```
       [1., 1., 1.]]]])
```

```
import numpy as np
a = np.array(([[0, 1, 2], [10, 12, 13]], [[100, 101, 102], [110, 112, 113]]))
a
```

```
array([[[  0,   1,   2],
        [ 10,  12,  13]],

       [[100, 101, 102],
        [110, 112, 113]]])
```

```
a.shape
```

```
(2, 2, 3)
```

```
for row in a:
  print(row)
  print(' ')
```

```
[[ 0  1  2]
 [10 12 13]]

[[100 101 102]
 [110 112 113]]
```

```
a.flat
```

```
<numpy.flatiter at 0x18b9d8a0>
```

```
for el in a.flat:
  print(el)
```

```
0
1
2
10
12
13
100
101
102
110
112
113
```

```
a.ravel() # Делает массив плоским, но сам массив не изменяется
```

```
array([  0,   1,   2,  10,  12,  13, 100, 101, 102, 110, 112, 113])
```

```
a
```

```
array([[[  0,   1,   2],
        [ 10,  12,  13]],

       [[100, 101, 102],
        [110, 112, 113]]])
```

```
a.shape = (6, 2) # Изменение формы, сам массив изменяется
a
```

```
array([[  0,   1],
       [  2,  10],
       [ 12,  13],
       [100, 101],
       [102, 110],
       [112, 113]])
```

```
a.transpose() # Транспонирование, сам массив не изменяется
```

```
array([[  0,   2,  12, 100, 102, 112],
       [  1,  10,  13, 101, 110, 113]])
```

```
a
a.reshape((3, 4)) # Изменение формы, но сам массив не изменяется
```

```
array([[  0,   1,   2,  10],
       [ 12,  13, 100, 101],
       [102, 110, 112, 113]])
```

```
a
```

```
array([[  0,   1],
       [  2,  10],
       [ 12,  13],
       [100, 101],
       [102, 110],
       [112, 113]])
```

```
a = np.array([[1, 2], [3, 4]])
a
```

```
array([[1, 2],
       [3, 4]])
```

```
b = np.array([[5, 6], [7, 8]])
b
```

```
array([[5, 6],
       [7, 8]])
```

```
np.vstack((a, b))
```

```
array([[1, 2],
       [3, 4],
       [5, 6],
       [7, 8]])
```

```
np.hstack((a, b))
```

```
array([[1, 2, 5, 6],
       [3, 4, 7, 8]])
```

```
np.column_stack((a, b))
```

```
array([[1, 2, 5, 6],
       [3, 4, 7, 8]])
```

```
np.row_stack((a, b))
```

```
/tmp/ipython-input-596025742.py:1: DeprecationWarning: `row_stack` alias is depr
  np.row_stack((a, b))
array([[1, 2],
       [3, 4],
       [5, 6],
       [7, 8]])
```

```
a = np.arange(12).reshape((2, 6))
a
```

```
array([[ 0,  1,  2,  3,  4,  5],
       [ 6,  7,  8,  9, 10, 11]])
```

```
z1,z2,z3=np.hsplit(a, 3) # Разбить на 3 части по столбцам
z1
```

```
array([[0, 1],
       [6, 7]])
```

```
np.hsplit(a, (3, 4)) # Разрезать a после третьего и четвёртого столбца
```

```
[array([[0, 1, 2],
        [6, 7, 8]]),
 array([[3],
```

```
       [9]]),
 array([[ 4,  5],
       [10, 11]])]
```

```
np.vsplit(a, 2) # Разбить на 2 части по строкам
```

```
[array([[0, 1, 2, 3, 4, 5]]), array([[ 6,  7,  8,  9, 10, 11]])]
```

```
a = np.arange(12)
a
```

```
array([ 0,  1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11])
```

```
b = a
b
```

```
array([ 0,  1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11])
```

```
b is a # проверим что это один и тот же объект
```

```
True
```

```
b.shape
```

```
(12,)
```

```
b.shape = (3,4) # изменим массив b
a.shape #
```

```
(3, 4)
```

```
c = a.view()
c
```

```
array([[ 0,  1,  2,  3],
       [ 4,  5,  6,  7],
       [ 8,  9, 10, 11]])
```

```
c is a
```

```
False
```

```
a[0,0]=100500 # изменим массив a
c # изменилось и его представление c
```

```
array([[100500,      1,      2,      3],
       [     4,      5,      6,      7],
       [     8,      9,     10,     11]])
```

```
c[1,1]=-100500 # поменяем с
c # оно конечно изменилось
```

```
array([[ 100500,       1,       2,       3],
       [      4, -100500,       6,       7],
       [      8,       9,      10,      11]])
```

```
a # и a тоже изменилось
```

```
array([[ 100500,       1,       2,       3],
       [      4, -100500,       6,       7],
       [      8,       9,      10,      11]])
```

```
d = a.copy() # создается новый объект массива
d # данные в нем такие же
```

```
array([[ 100500,       1,       2,       3],
       [      4, -100500,       6,       7],
       [      8,       9,      10,      11]])
```

```
d is a # Это тот же объект? Нет
```

```
False
```

```
a[0,0]=-1 # изменим массив a
a
```

```
array([[     -1,       1,       2,       3],
       [      4, -100500,       6,       7],
       [      8,       9,      10,      11]])
```

```
d # изменился ли массив d? Нет.
```

```
array([[ 100500,       1,       2,       3],
       [      4, -100500,       6,       7],
       [      8,       9,      10,      11]])
```

```
import random
import numpy as np
import numpy.random as rand
```

```
np.random.sample() # одно случайное число
```

```
0.23664264573977645
```

```
a=-3
b=5
a+(b-a)*np.random.sample() # одно случайное число из диапазона [a,b)
```

```
-0.33980587129550166
```

```
np.random.sample(5) # Массив из 5 случайных чисел
```

```
array([0.17070599, 0.31451864, 0.33420286, 0.40893845, 0.81362903])
```

```
np.random.sample((1, 1, 4)) # 4 случайных числа в трехмерном массиве
```

```
array([[[0.83821064, 0.48549386, 0.08870214, 0.32630805]]])
```

```
np.random.randint(0, 3, 10)# массив из 10 случайных целых чисел от 0 до 2 (3 не
```

```
array([1, 1, 0, 1, 0, 2, 2, 1, 0, 1])
```

```
np.random.random_integers(0, 3, 10)# массив из 10 случайных целых чисел от 0 дс
```

```
/tmp/ipython-input-1533306391.py:1: DeprecationWarning: This function is deprecа
  np.random.random_integers(0, 3, 10)# массив из 10 случайных целых чисел от 0 д
array([1, 3, 2, 1, 2, 1, 1, 1, 3, 2])
```

```
np.random.randint(0, 3, (2, 10))# двумерный массив случайных чисел от 0 до 2.
```

```
array([[0, 1, 2, 2, 1, 2, 0, 0, 2, 2],
       [1, 2, 2, 0, 2, 1, 0, 1, 0, 1]])
```

```
np.random.uniform(2, 8, (2, 10)) # двумерный массив со случайными числами от 2
```

```
array([[2.13395795, 5.24393853, 3.83535093, 6.39256987, 2.81612218,
        6.30650815, 3.02862155, 3.04103694, 6.17170842, 5.69535534],
       [7.6394254 , 4.84270285, 2.10892391, 7.85654095, 5.65442691,
        5.93657585, 7.34997729, 3.21114078, 5.45762166, 2.00759772]])
```

```
a = np.arange(10)
a=np.random.randint(0,5,(2,3))
```

```
array([[3, 0, 4],
       [1, 3, 3]])
```

```
np.random.shuffle(a)
a
```

```
array([[1, 3, 3],
       [3, 0, 4]])
```

```
np.random.seed(42) # начинаем с места №1000
np.random.random(10) # берем 10 случайных чисел
```

```
array([0.37454012, 0.95071431, 0.73199394, 0.59865848, 0.15601864,
       0.15599452, 0.05808361, 0.86617615, 0.60111501, 0.70807258])
```

```
np.random.seed(100) # Начинаем с другого места №100
np.random.random(10) # Берем 10 случайных чисел, они другие
```

```
array([0.54340494, 0.27836939, 0.42451759, 0.84477613, 0.00471886,
       0.12156912, 0.67074908, 0.82585276, 0.13670659, 0.57509333])
```

```
om.seed(1000) # Начинаем с того же места №1000
om.random(10) # Берем 10 случайных чисел, они такие же как в первом варианте
```

```
array([0.65358959, 0.11500694, 0.95028286, 0.4821914 , 0.87247454,
       0.21233268, 0.04070962, 0.39719446, 0.2331322 , 0.84174072])
```

```
batchsize, maps, h, w = 1, 1, 3, 3
data = (np.arange(batchsize * maps * h * w).reshape(batchsize, maps, h, w).as
data
```

```
array([[[[0., 1., 2.],
         [3., 4., 5.],
         [6., 7., 8.]]]], dtype=float32)
```