**NUMPY & MatPlotLib**

|  |  |
| --- | --- |
| **Команда** | **Описание** |
| arr = np.ones((3, 3) np.uint8) | массив нулей |
| np.zeros\_like(array) | массив единиц |
| np.empty([0], dtype=np.float32) | пустой массив(здесь это вектор) нулевой длины |
| if type(arr) == np.ndarray: | проверка на то, что это массив numpy |
| arr.dtype | тип элементов массива |
| a.shape == (355, 64, 8)  a[0, :, :] .shape == (64, 8)  a[:, :7, :] .shape == (355, 7, 8) | операции над измерениями |
| np.sum(arr, axis =0) | суммировать элементы массива по оси 0 |
| np.mean(arr) | среднее. Можно добавить axis |
| arr.reshape(m, n) | преобразовать массив к размеру m\*n |
| arr.flatten() | векторизовать |
| arr = np.asarray(ls)  ls = arr.tolist() | из списка в массив и наоборот |
| indx\_max = np.argmax(arr) | индекс максимального элемента (можно указать axis) |
| indxs = np.where[arr == 0) | массив индексов, где arr[i] = 0 |
| arr = np.where(**arr < 10**, **0**, **arr\*10**) | все элементы массива, которые **<10** **обнулятся**, а остальные будут **умножены на 10**. |
| np.random.rand(rows, columns) | массив случайных чисел |
| np.random.normal(0.0, pow(k, -0.5), (rows, columns)) | нормальное отклонение |
| np.random.randint(a, b, n) | n-вектор рандомных чисел между a и b - int |
| np.random.random(n) | n-вектор рандомных чисел между 0.0 и 1.0 |
| np.random.uniform(a, b, n) | n-вектор рандомных чисел между a и b - float |
| a.shape == (3, 18, 16)  a.size == 3\*18\*16=864  a.ndim = 3 | размерность тензора  размер всего тензора  кол-во измерений в тензоре |
| np.add(W1, W2) == W1+W2 | суммирование элементов |
| np.subtract(W1, W2)==W1-W2 | вычитание элементов |
| np.dot(W1, W2)==W1@W2 | произведение матриц |
| np.dot(a, b) | скалярное произведение векторов |
| prodW = W1\*W2 | произведение элементов массивов |
| np.prod(A) | произведение всех элементов массива |
| res = np.concatenate( (a1, a2) ) | склейка двух массивов a1 и а2 (можно указать axis) |
| matrix.diaginal() | вектор диагональных элементов |
| matrix.diagonal(offset = **+1**) | вектор элементов **выше** диагонали |
| matrix.trace() | след матрицы – сумма диагональных элементов |
| np.det(arr) | определитель |
| np.linalg.norm(arr) | Евклидова норма |
| Wt = W.T | транспонирование |
| np.linalg.inv(M) | обратная матрица |
| Eigenvals, eigenvecs =  np.linalg.eig(arr) | Собств числа и собств вектора |
| if (np.all(A > 0)) | если все элементы матрицы А > 0 |
| if (np.any(A > 0)) | если какой-то элементы матрицы А > 0 |
| np.digitize(arr, bins=[10, 20, 30]) | в массиве arr все значения меньше 10 ->**0**, значения [10, 20) -> **1**, значения [20, 30) -> **2**, значения >= 30 -> **3**. Чтобы сравнивать по правому краю, добавить параметр **right=True** |
| np.swapaxes(arr, m, n) | замена измерение (осей) |
| np.save(‘arr.npy’, np.array([1,2,3]) ) | сохранение массива в файл |
| arr = np.load(‘arr.npy’) | загрузка массива из файла |
|  | |
| plt.plot(X.numpy(), Y.numpy, c=’**r**’, ‘o’, **label**=’name of line’, linewidth=3, alpha=0.2) | построение графика. alpha-степень прозрачности |
| plt.title(‘$ y = sinX $’) | название графика |
| plt.axis(xmin, xmax, ymin, ymax) | устанавливаем диапазон |
| plt.scatter(x, y, **label**=’point’, s=2) | отобразить одну точку в системе координат. s – размер точки |
| plt.savefig(‘name.jpg’, fmt=’jpg’) | сохранение графика в файл |
| fig = plt.figure()  fig.add\_axis([0, 0, 1, 1]**, polar=’True’**) | создание полярной системы координат |
| plt.title(‘name graphic’) | название графика |
| plt.xlabel(‘axis name’) | подпись оси Х |
| plt.axis(‘off’) | отключить оси |
| plt.**legend**(loc=’upper left’) | добавление легенды |
| plt.subplot(m, n, index)  plt.imshow(imgRGB) | несколько графиков в одной картинке  m,n – кол-во строк и столбцов. index – номер текущего графика |
| plt.figure(1) # draw plt.figure(2) # draw plt.show() | нарисуем графики в двух разных окнах |
| matplotlib.rcParams['figure.figsize'] = (x.0, y.0) | задаем размер окна. x, y = 80 pixels |
| **plt.show()** | **вывод графика в окне** |
|  |  |
|  |  |
|  |  |
|  |  |