NumPy

NumPy (Numeric Python) – это сторонний пакет, предоставляющий методы для работы с большими массивами.

Когда мы разбирали работу с пакетами, мы уже установили NumPy через командную строку с помощью строки:

pip install numpy

Для работы с NumPy импортируем модуль «numpy»:

import numpy as np

Так как мы будем много использовать данный модуль, мы даем ему короткое название «np».

# Массивы

Главной особенностью NumPy является объект array (массив). Массив в NumPy схож с листом Python, за исключением того, что в нем можно хранить элементы только одинакового типа, а также он является неизменяемы по длине (т.е. мы не можем добавлять или удалять из него элементы). Работа с массивами реализована гораздо быстрее и эффективнее.

Объявим массив в NumPy с помощью функции array и листа:

import numpy as np  
  
a = np.array([1, 2, 3.0])  
print('a =', a)  
  
b = np.array([1, 2, 3])  
print('b =', b)  
  
c = np.array([1., 2., 3.0], dtype=int)  
print('c =', c)

Вывод на консоль:

a = [1. 2. 3.]

b = [1 2 3]

c = [1 2 3]

Как мы видим, если не указать тип функции «array», то тип данных определяется автоматически, иначе – если указать, то все элементы приводятся к указанному типу.

Также массивы могут быть многомерными. Многомерный массив можно определить с помощью листа, элементы которого тоже являются листами одинаковой длины. Array() трансформирует вложенные последовательности в массивы, в итоге мы получаем массив массивов:

import numpy as np  
  
a = np.array([[1, 2], [3, 4]])  
print(a)

Вывод на консоль:

[[1 2]

[3 4]]

Если мы указываем листы разной длины, то получаем массив, тип элементов которого является «list», т.е. мы получим не массив массивов, а массив листов:

import numpy as np  
  
b = np.array([[1, 2], [3, 4, 5]])  
print(b)

Вывод на консоль:

[list([1, 2]) list([3, 4, 5])]

Кроме функции array(), массив можно создать еще некоторыми способами, например:

import numpy as np  
  
a = np.zeros((2, 3))  
print(a)  
  
b = np.ones((2, 3))  
print(b)

Вывод на коносоль:

[[0. 0. 0.]

[0. 0. 0.]]

[[1. 1. 1.]

[1. 1. 1.]]

Функции np.zeros() и np.ones() создают массив указанной размерности, состоящий из нулей и единиц соответственно. Размерность (2, 3) означает, что в массиве будет два элемента, которые являются массивами, содержащими 3 элемента.

К элементам массива NumPy можно обращаться также, как и к элементам листа:

import numpy as np  
  
a = np.array([1, 2, 3])  
print(a[1])  
print(a[1:])

Вывод на консоль:

2

[2 3]

К элементам многомерного массива можно обращаться также, используя индекс или срез для каждого измерения, указывая их через запятую или несколько квадратных скобок:

import numpy as np  
  
a = np.array([[1, 2, 3], [4, 5, 6]])  
print(a[1, 1]) # аналогично a[1][1]  
print(a[1:, 1:]) # аналогично a[1:][1:]

Вывод на консоль:

5

[[5 6]]

Массивы поддерживают in и len:

import numpy as np  
  
a = np.array([1, 2, 3])  
print(2 in a)  
print(len(a))

Вывод на консоль:

True

3

Однако, если применить функцию len() к многомерному массиву, то она вернет длину его первого измерения:

import numpy as np  
  
a = np.array([[1, 2, 3], [4, 5, 6]])  
print(len(a))

Вывод на консоль:

2

Чтобы узнать реальный размер массива используется метод shape:

import numpy as np  
  
a = np.array([[1, 2, 3], [4, 5, 6]])  
print(a.shape)

Вывод на консоль:

(2, 3)

Мы можем транспонировать массив, используя метод transpose() – т.е. поменять строки и столбцы местами:

import numpy as np  
  
print("Исходный: ")  
a = np.array([[1, 2, 3], [4, 5, 6]])  
print(a)  
print(a.shape)  
  
print("Транспонированный: ")  
a = a.transpose()  
print(a)  
print(a.shape)

Вывод на консоль:

Исходный:

[[1 2 3]

[4 5 6]]

(2, 3)

Транспонированный:

[[1 4]

[2 5]

[3 6]]

(3, 2)

Также, мы можем изменить размерность массива с помощью функции reshape(), которая возвращает указанный массив приведенный к указанной размерности. Только при условии, что

shape\_1 \* shape\_2 \* … \* shape\_n = new\_shape\_1 \* new\_shape\_2 \* … \* new\_shape\_m

import numpy as np  
  
print("Исходный: ")  
a = np.array([[1, 2, 3, 4, 5, 6]])  
print(a)  
print(a.shape)  
  
print("С измененным размером: ")  
a = np.reshape(a, (2, 3))  
print(a)  
print(a.shape)

Вывод на консоль:

Исходный:

[[1 2 3 4 5 6]]

(1, 6)

С измененным размером:

[[1 2 3]

[4 5 6]]

(2, 3)

# Операции над массивами

## Математические операции

Мы можем использовать стандартные математические операции, такие как сложение, умножение и т.д., которые будут применяться к массивам поэлементно, поэтому, чтобы они работали необходимо применять их к массивам одинаковой размерности (Когда для массивов мы используем стандартные математические операции, должен соблюдаться принцип: элемент--элемент. Это означает, что массивы должны быть одинакового размера во время сложения, вычитания и тому подобных операций:):