Python

numpy

Модуль numpy

- NumPy это open-source модуль для python, который предоставляет общие математические и числовые операции. NumPy (Numeric Python) предоставляет базовые методы для манипуляции с большими массивами и матрицами. SciPy (Scientific Python) расширяет функционал numpy огромной коллекцией полезных алгоритмов, таких как минимизация, преобразование Фурье, регрессия, и другие прикладные математические техники.

Установка

- Для того, чтобы установить модуль numpy, необходимо открыть консоль
- Win + R cmd Enter
- Далее, в консоли необходимо прописать pip install numpy
- Установка завершена

Что дальше?

- Далее открываем снова jupyter notebook (Инструкция в файле Jupyter Notebook.pdf)
- В первой строке прописываем import numpy as np
  - Import – подключение модуля
  - Numpy – модуль
  - As np – используется для сокращенного пользования модулем

NUMPY

Массивы

Массивы

- В программировании очень часто встречаются массивы.
- Они очень похожи на списки, но у них есть различия. В основном эти различия затрагивают память, но нам это не особо нужно.
- Массивы бывают 1-мерные (вектора) 2-мерные (матрицы) и многомерные (условно тензоры).
- Все элементы массива должны принадлежать к одному типу данных.

Создание массивов

- В программировании очень часто встречаются массивы.
- Они очень похожи на списки, но у них есть различия. В основном эти различия затрагивают память, но нам это не особо нужно.
- Массивы бывают 1-мерные (вектора) 2-мерные (матрицы) и многомерные (условно тензоры).
- Все элементы массива должны принадлежать к одному типу данных.
- Каждый модуль имеет порядка 50-100 различных методов и функций.

Создание массивов

- Итак, первое, что нам необходимо рассмотреть – функция array()
- Данная функция позволяет создать массив из имеющихся данных
- На данных скринах Вы видите создание одномерного и двумерного массивов
- Двумерный массив – это как массив массивов

```
import numpy as np
```

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

```
print(mass)
```

```
[1 2 3]
```

```
mass2 = np.array([[1, 2, 3], [4, 5, 6], [7, 8, 9]])
```

```
print(mass2)
```

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

Создание массивов

- Важный момент. Если мы заходим создать массив из тысячи или, даже, из 100 элементов, нам не очень удобно будет прописывать каждый.
- На помощь приходит функция arange()
- И да, как вы заметили, тут можно не printовать данные

```
In [11]: mass3 = np.arange(1,10)
         mass3

Out[11]: array([1, 2, 3, 4, 5, 6, 7, 8, 9])

In [12]: a = np.arange(0,100, 5)
         a.shape = (4,5)
         a

Out[12]: array([[ 0,  5, 10, 15, 20],
                [25, 30, 35, 40, 45],
                [50, 55, 60, 65, 70],
                [75, 80, 85, 90, 95]])
```

Создание массивов

- Метод .linspace(a,b,c) создает арифметическую прогрессию, где
- a, b – промежуток (от a до b)
- c – количество элементов в одномерном массиве

```
In [30]: np.linspace(10, 100, 30)

Out[30]: array([ 10.        ,  13.10344828,  16.20689655,  19.31034483,
                 22.4137931 ,  25.51724138,  28.62068966,  31.72413793,
                 34.82758621,  37.93103448,  41.03448276,  44.13793103,
                 47.24137931,  50.34482759,  53.44827586,  56.55172414,
                 59.65517241,  62.75862069,  65.86206897,  68.96551724,
                 72.06896552,  75.17241379,  78.27586207,  81.37931034,
                 84.48275862,  87.5862069 ,  90.68965517,  93.79310345,
                 96.89655172, 100.        ])
```

Создание массивов

- К тому же есть 2 специальных метода .ones() и .zeros(), которые создают массивы из 1 и 0 соответственно

```
In [19]: np.zeros(5)
Out[19]: array([0., 0., 0., 0., 0.])

In [20]: np.ones(5)
Out[20]: array([1., 1., 1., 1., 1.])

In [21]: np.ones([5,5])
Out[21]: 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., 1.]])

In [22]: np.zeros([5,5])
Out[22]: array([[0., 0., 0., 0., 0.],
                [0., 0., 0., 0., 0.],
                [0., 0., 0., 0., 0.],
                [0., 0., 0., 0., 0.],
                [0., 0., 0., 0., 0.]])
```

Создание массивов

- Ну, и конечно, создание массива с помощью модуля random

```
In [31]: import random

In [39]: np.array([random.randint(-10,10) for x in range(0,10)])
Out[39]: array([ 10,   1,   0, -10,  -4,   0,   9,   2,   4,  -4])

In [42]: temp = np.array([random.randint(-10,10) for x in range(0,10)])
         temp.shape = (2,5)
         temp
Out[42]: array([[ -3,  -4,  -2, -10,  -9],
                [  2,  -9, -10, -10,   1]])
```

Закрепление

- В массиве найти максимальный элемент с четным индексом. Другая формулировка задачи: среди элементов массива с четными индексами, найти тот, который имеет максимальное значение.
- Найти в массиве те элементы, значение которых меньше среднего арифметического, взятого от всех элементов массива.
- В одномерном массиве целых чисел определить два наименьших элемента. Они могут быть как равны между собой (оба являться минимальными), так и различаться.
- Найти сумму всех цифр целочисленного массива. Например, если дан массив [12, 104, 81], то сумма всех его цифр будет равна 1 + 2 + 1 + 0 + 4 + 8 + 1 = 17.
- Найти среднее арифметическое положительных элементов массива.
- В массиве случайных целых чисел поменять местами минимальный и максимальный элементы.
- Найти сумму, произведение и среднее арифметическое элементов матрицы (двумерного массива).
- Найти сумму элементов главной диагонали матрицы

Домашняя работа

- Заполнить одномерный массив случайными числами. Найти и вывести на экран наибольший его элемент и порядковый номер этого элемента.
- В массиве, содержащем положительные и отрицательные целые числа, вычислить сумму четных положительных элементов.