NUMPY

Numpy ≡ Массивы с математикой

Официальный сайт последней стабильной версии

NumPy (Numerical Python) - это фундаментальная библиотека для научных вычислений в Python. Она предоставляет:

- Многомерные массивы (ndarray) высокой производительности
- Математические функции для работы с массивами
- Инструменты для линейной алгебры, преобразования Фурье и случайных чисел

Основные преимущества:

- Быстродействие: Оптимизированные реализации на C/Fortran
- Компактность: Меньший объем памяти по сравнению со списками Python
- Удобство: Векторные операции без циклов

Создание массивов:

```
import numpy as np

# Из списка

arr = np.array([1, 2, 3, 4, 5])


# Специальные массивы

zeros = np.zeros(5)          # [0., 0., 0., 0., 0.]

ones = np.ones(5)            # [1., 1., 1., 1., 1.]

range_arr = np.arange(0, 10) # [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]

arr = np.full(4, 3) # [3,3,3,3]
```

Задача 1: Создание массивов

Создайте:

1. Массив из 10 нулей

2. Массив из 7 единиц

3. Массив чисел от 0 до 20 с шагом 2

4. Массив из 5 случайных чисел от 0 до 1

Задача 2: Операции с массивами

Дан массив: [2, 4, 6, 8, 10]

Выполните операции:

1. Умножьте каждый элемент на 3

2. Прибавьте к каждому элементу 5

3. Возведите каждый элемент в квадрат

4. Найдите сумму всех элементов

5. Найдите среднее значение

Задача 3: Индексация и срезы

Дан массив: [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]

Получите:

1. Первые 3 элемента

2. Последние 3 элемента

3. Каждый второй элемент

4. Элементы с индексами 2, 4, 6

5. Все элементы больше 5

Задача 4: Двумерные массивы

Создайте матрицу 3x3 из чисел от 1 до 9

Выполните:

1. Найдите сумму всех элементов

2. Найдите сумму по столбцам

3. Найдите сумму по строкам

4. Транспонируйте матрицу

5. Умножьте матрицу на себя (поэлементно)

Необходимо создать массив из 5 одинаковых значений 7. В ответе напишите соответствующую команду. Способ решения должен быть корректным с помощью NumPy.

Типы данных в массивах

Массивы в numpy имеют не только размер, но и конкретный тип данных, которые хранятся внутри массива. При создании массива почти все функции создания имеют аргумент dtype, который означает, с каким типом создать данным массив. Также, класс ndarray имеет атрибут dtype, который означает тип хранимых данных.

```
arr = np.array([[1.5, 2, 3, 4], [5, 6, 7, 8], [9, 10, 11, 12]])

print(arr.dtype)

# Один из элементов float, поэтому весь массив будет float
```

Что тут происходит?

```
arr = np.array([[1.1, 1.6, 2.4], [-1.7, 2.6, -1.2]], dtype=int)
print(arr)
print(arr.dtype)

arr = np.array([[1.1, 1.6, 2.4], [-1.7, 2.6, -1.2]])
print(arr.dtype)
arr = arr.astype(int)
print(arr)
print(arr.dtype)
```

Изменение размеров массива (Reshape)

Ещё одна интересная тема, которую стоит рассмотреть. Мы уже касались вопроса приведения массива к выпрямленному виду - это один из видов изменения размерности массива.

По сути, размерность массива - это порядок выстроенных элементы в нём. Изменение размерности - это изменение структуры без изменения элементов или порядка.

```
arr = np.arange(20)

print(arr)

print(arr.reshape((4, 5)))
```