Язык Python. Часть 2

Лекция 11

Массированные операции

- Правильный шаблон использования numpy - массированные операции
- Чем меньше явных циклов, тем лучше
- Простейшее - присваивание значения всему ndarray
- Инкремент/декремент

Пример

```
 1 import numpy as np
 2
 3 data = np.array([100, 200, 20], dtype=np.int16)
 4 print(data)
 5 print(data + 1)
 6 data += 2
 7 print(data)
 8 data = 11
 9 print(data)
10 data = 1.23
11 print(data)
```

Многомерные индексы и вырезки

- В квадратных скобках может идти последовательность индексов или вырезок
- Через запятую
- Элементов в списке - не больше, чем размерность ndarray
- Может быть меньше, но не 0

Многомерные индексы и вырезки

- Если меньше, то недостающие понимаются как :
- Размерность результата зависит от количества индексов и вырезок
- Каждый индекс уменьшает размерность
- Каждая непустая вырезка сохраняет размерность

Пример: одномерный случай

```
1 import numpy as np
2
3 data = np.array([100, 12, 23, 200, 20, 10, 12, 8], dtype=np.
4 print(data[1])   # элемент
5 print(data[2:5]) # вектор
6 print(data[2:5].shape) # (3,)
7 print(data[:].shape) # (8,)
8 print(data[2:2].shape) #  (0,)
```

Пример: двумерный случай

```
 1 import numpy as np
 2
 3 data = np.array([[100, 12, 23, 200, 20, 10, 12, 8],
 4                  [1, 2, 3, 4, 11, 22, 33, 44],
 5                  [9, 8, 7, 6, 99, 88, 77, 66],
 6                 ], dtype=np.int16)
 7
 8 print(data[1, 2])   # элемент
 9 print(data[1, 2:5]) # вектор
10 print(data[1, 2:5].shape) # (3,)
11 print(data[1:2, 2]) # вектор
12 print(data[1:2, 2].shape) # (1,)
13 print(data[1:, 2:5].shape) # (2, 3)
```

Пример: пустые вырезки

```
 1 import numpy as np
 2
 3 data = np.array([[100, 12, 23, 200, 20, 10, 12, 8],
 4                  [1, 2, 3, 4, 11, 22, 33, 44],
 5                  [9, 8, 7, 6, 99, 88, 77, 66],
 6                 ], dtype=np.int16)
 7
 8 print(data[1, 2:2]) # пустой ndarray
 9 print(data[1, 2:2].shape) # (0,)
10 print(data[1:, 2:2]) # пустой ndarray
11 print(data[1:, 2:2].shape) # (2, 0)
12 print(data[1:1, 2]) # пустой ndarray
13 print(data[1:1, 2].shape) # (0,)
14 print(data[1:1, 2:]) # пустой ndarray
15 print(data[1:1, 2:].shape) # (0, 6)
```

Присваивание многомерной вырезке

- В левой части присваивания может стоять многомерная вырезка
- Тогда ее элементы не "вынимаются" из исходного массива
- Просходит присваивание на их позиции
- В правой части может быть все, что подходит под shape вырезки

reshape

- Можно придать новую форму данным ndarray
- Не копируя их
- Например, вектор длиной 20 превратить в матрицу 5x4
- Или матрицу 12 x 8 преобразовать в 6 x 16

reshape: детали

- Вектор занимает память "слева направо"
- Двумерный ndarray занимает память построчно
- Сначала 0-я строчка
- Потом 1-я, и так далее
- Каждая строчка - слева направо

reshape: детали

- Трехмерный (и более) случай - по аналогии
- Сначала идем по первому измерению
- И выкладываем элементы по очереди
- Каждый элемент обрабатываем рекурсивно
- Метод reshape накладывает новые размерности, не меняя данных

Пример: многомерные случаи

```
 1 import numpy as np
 2
 3 data = np.array(range(24), dtype=np.int8)
 4
 5 print(data.reshape((2, 12)))
 6 print(data.reshape((3, 8)))
 7 print(data.reshape((4, 6)))
 8 print(data.reshape((2, 3, 4)))
 9 print(data.reshape((1, 2, 3, 4)))
10 print(data.reshape((2, 1, 3, 4)))
```

Пример: присваивание вырезке

```
1 import numpy as np
2
3 data = np.array(range(36), dtype=np.int8).reshape((4, 3, 3))
4
5 data[1 : 3, 2, 2 : 3] = 55
6 print(data)
```

Пример: изменение вырезки

```
1 import numpy as np
2
3 data = np.array(range(36), dtype=np.int8).reshape((4, 9))
4
5 print(data)
6 print()
7 data[1 : 3, 2 : 4] += 55
8 print(data)
```

Поэлементная арифметика

- Предполагается, что операнды - ndarray
- У обоих операндов - один shape
- Операция выполняется поэлементно
- Может быть обычная операция (+ и т.п.) или операция с присваиванием (+= и т.п.)

Пример: поэлементное сложение

```
1 import numpy as np
2
3 data_1 = np.array(range(36), dtype=np.int8).reshape((4, 9))
4 data_2 = np.array(range(36), dtype=np.int8).reshape((4, 9))
5
6 data = data_1 + data_2
7 print(data)
```

Пример: поэлементное сложение с присваиванием

```
1 import numpy as np
2
3 data_1 = np.array(range(36), dtype=np.float64).reshape((4, 9
4 data_2 = np.array(range(36), dtype=np.float64).reshape((4, 9
5
6 data_1 += data_2
7 print(data_1)
```

Поэлементная арифметика

- Так работают сложение, вычитание
- Умножение и деление
- Умножение - тоже поэлементное
- Есть умножение матриц, но обозначается по-другому

view и copy

- Можно создать новый ndarray методом copy
- С копией данных
- Можно создать новый объект над данными старого
- С помощью метода view
- Стандартная вырезка дает view
- Вырезка через массив индексов дает copy

Пример с view

```
 1 import numpy as np
 2
 3 data_1 = np.array(range(36), dtype=np.int8).reshape((4, 9))
 4 data_2 = data_1[2:4]
 5 data_3 = data_1[[2, 3]]
 6
 7 data_1[2, 0] = 77
 8 data_2[0, 1] = 88
 9 data_3[0, 2] = 99
10
11 print(data_1)
12 print()
13 print(data_2)
14 print()
15 print(data_3)
```

view

- Иногда важно понять, является ли ndarray самостоятельным
- Или это view
- Можно проверить через атрибут base
- None для самостоятельного
- Ссылка на базовый для view

Еще пример с view

```
 1 import numpy as np
 2
 3 data_1 = np.array(range(36), dtype=np.int16).reshape((4, 9)
 4 data_2 = data_1[2:4]
 5 data_3 = data_1[[2, 3]]
 6
 7 print(data_1.base)
 8 print(data_2.base)
 9 print(data_3.base)
10 print(np.array(range(36), dtype=np.int16).base)
```

Пример с view посложнее

```
 1 import numpy as np
 2
 3 data_1 = np.array(range(48), dtype=np.int16).reshape((6, 8)
 4 data_2 = data_1[1:-1, 1:-1]
 5 data_3 = data_2[1:-1, 1:-1]
 6 data_3[0, 0] = 111
 7 data_2[0, 0] = 222
 8 data_1[0, 0] = 333
 9
10 print(data_1)
11 print()
12 print(data_2)
13 print()
14 print(data_3)
```

broadcasting

- Мы можем складывать массив с массивом
- Если у них одинаковый shape
- Можем умножать вектор на число
- Хотим делать нечто промежуточное
- Например, складывать матрицу со строкой

broadcasting

- Как можно представить умножение вектора на число
- Берем число
- Прикладываем к каждому элементу вектора
- Складываем
- Получаем результат

broadcasting

- Повысим размерность
- Возьмем двумерный ndarray
- И строку с числами
- Прикладываем ее к каждой строке
- Складываем и получаем результат

Пример на broadcasting

```
1 import numpy as np
2
3 data_1 = np.array(range(48), dtype=np.int8).reshape((6, 8))
4 data_2 = np.array(range(8), dtype=np.int8).reshape((8,))
5
6 print(data_1 + data_2)
```

broadcasting: общие правила

- Если оба операнда - ndarray
- Сопоставляем их размерности
- Начиная с конца
- Размерности считаем совместимыми
- Если они равны или одна из них равна 1

broadcasting: общие правила

- Сюда можно вписать и скаляр
- Если считать, что размерность скаляра - пустой кортеж
- И это даже формально верно
- Можно попробовать: np.array(5).shape

broadcasting: общие правила

- Если ndim одного меньше, чем ndim другого
- Добавляем к меньшему единички
- Для каждой пары ndarray возможны два варианта
  - Размерности совместимы
  - Размерности несовместимы

broadcasting: совместимость размерностей

- Скаляр совместим с чем угодно
- Вектор несовместим с вектором другого размера
- Вектор совместим с любым ndarray, чья последняя размерность равна его размеру
- Например, вектор размером 5 совместим с матрицей (10, 5)

broadcasting: совместимость размерностей

- Вектор неединичного размера несовместим с любым ndarray, чья последняя размерность не равна его размеру и не равна 1
- Например, вектор размером 5 несовместим с матрицей (8, 4)
- Но совместим с матрицей (8, 1)
- Или с ndarray, чей shape - (100, 200, 300, 5)

broadcasting: совместимость размерностей

- Поэлементные операции проводятся между ndarray с совпадающими shape
- Или с совместимым
- Если совместимы, но не совпадают, то в каких-то позициях shape есть отличия
- В одном 1, в другом что-то другое

broadcasting: совместимость размерностей

- В таком случае при поэлементной операции
- Там, где не 1, перебираются все индексы
- А там, где 1 - берется значение с индексом 0
- (А других и нет)

broadcasting: разберем на примерах

- Вектор размером 5 и скаляр
- shape-ы: (5,) и ()
- Второй превращается в (1,)
- Скаляр воспринимаем как 1-мерный вектор

broadcasting: разберем на примерах

- Поэлементно складываем
- От второго операнда всегда берем 0-й элемент
- Получили операцию вектора с числом
- Осознали как частный случай broadcasting-а

broadcasting: разберем на примерах

- Возьмем матрицу, shape (5, 6)
- Возьмем число
- Осознаем число как ndarray, shape ()
- Дополним shape до (1, 1)

broadcasting: разберем на примерах

- Начинаем поэлементное сложение
- По обоим измерениям во втором аргументе берем нулевые элементы
- Получаем поэлементое добавление числа
- Осознали как частный случай broadcasting-a

broadcasting: разберем на примерах

- Возьмем матрицу, shape (5, 6)
- Возьмам вектор, shape (6,)
- Дополним shape до (1, 6)
- Начинаем поэлементное сложение

broadcasting: разберем на примерах

- По второму измерению работаем поэлементно
- По первому - берез позицию 0 во втором аргументе
- Проходим по строчкам матрицы
- Каждую строчку поэлементно складываем со вторым аргументом
- Тоже частный случай broadcasting-а

Пример

```
1 import numpy as np
2
3 print(np.array([1, 2, 10, 15, 20]) + 52)
4 print()
5 print(np.array([[1, 2, 10, 15, 20], [43, 23, 2, 9, -3]]) + 5
6 print()
7 print(np.array([[1, 2], [10, 15], [20, -9]]) + np.array([5,
```

broadcasting: разберем на примерах

- Возьмем матрицу, shape (5, 6)
- Возьмем вектор, shape (5,)
- Дополним shape до (1, 5)
- Получаем несовместимость

broadcasting: разберем на примерах

- Возьмем матрицу, shape (5, 6)
- Возьмем вектор, shape (5, 1)
- По первому измерению работаем поэлементно
- По второму - берем позицию 0 во втором аргументе

Пример

```
1 import numpy as np
2
3 print(np.array([[1, 2], [10, 15], [20, -9]]) + np.array([5,
4
5 print(np.array([[1, 2], [10, 15], [20, -9]]) + np.array([[5,
6
7 print(np.array([[1, 2], [10, 15], [20, -9]]) + np.array([[5]
```

broadcasting: разберем на примерах

- Возьмем матрицу, shape (600, 400, 3)
- Картинка 600x400 в RGB-формате
- Хотим все значения умножить на определенный коэффициент
- Для каждой точки - свой
- Для всех каналов - одинаковый

broadcasting: разберем на примерах

- Коэффициенты хранятся в матрице
- shape - (600, 400)
- Просто сложить shape (600, 400, 3) и shape (600, 400) нельзя
- Но можно сделать reshape в (600, 400, 1)
- И потом сложить

Пример

```
 1 import numpy as np
 2
 3 data = np.array([
 4    [(123, 234, 120), (123, 45, 34), (34, 21, 12), (45, 54,
 5    [(32, 123, 222), (213, 123, 33), (12, 23, 34), (54, 34,
 6 ], dtype=np.float32)
 7 coeffs = np.array([
 8    [0.2, 0.3, 0.45, 0.56],
 9    [0.21, 0.32, 0.43, 0.54],
10 ], dtype=np.float32)
11
12 print(data * coeffs) # так не заработает
13
14 print(data * coeffs.reshape((2, 4, 1)))
```