Практическая работа № 6

Знакомство с NumPy

Сокр. Numerical Python - специализированный инструмент языка Python для обработки больших числовых массивов. Массивы NumPy походжи на тип list, но обеспечивают более эффективное хранение и операции с данными. Разница тем ощутимее, чем больше массив.

За счет чего это происходит? Мы знаем, что стандартная реализация Python написана на C, то есть каждый объект Python - это ловко замаскированная конструкция на языке C, содержащая не только значение, но и другую информацию. Например, целое число в Python, это не только само значение, а указатель на структуру с несколькими полями. Т.е. если на C целое число - это ярлык для места в памяти, байты которого кодируют целочисленное значение, то в Python - это указатель на место в памяти, где хранится не только само целочисленное значение, но и информация об объекте int языка Python. Это плата за динамическую типизацию. Если собрать данные в список, каждый элемент будет, помимо значения, содержать еще некоторое количество метаданных о типе, счетчик ссылок и др. Если список однороден, эта информация избыточна. Т.е. главное отличие массива NumPy от списка в том, что он содержит один указатель на непрерывный блок данных.

Массивы

In [2]:

```
import numpy as np
```

In [3]:

```
np.array([1,2,3,4])
```

Out[3]:

```
array([1, 2, 3, 4])
```

In [4]:

```
np.array([2.73, 3.14, 1, 2])
```

Out[4]:

```
array([2.73, 3.14, 1.  , 2.  ])
```

Можно задать тип данных явно:

In [5]:

```
np.array([1,2,3,4,5], dtype = float)
```

Out[5]:

```
array([1., 2., 3., 4., 5.])
```

In [7]:

```
x = np.array([1,2,3,4,5], dtype = int)
x
```

Out[7]:

```
array([1, 2, 3, 4, 5])
```

In [8]:

```
x[0]=3.14
x
```

Out[8]:

```
array([3, 2, 3, 4, 5])
```

Массивы NumPy могут быть многомерными:

In [10]:

```
np.array([range(i,i+2) for i in [2,4,6]])
```

Out[10]:

```
array([[2, 3],
       [4, 5],
       [6, 7]])
```

Вложенные списки тут рассматриваются как строки

Сгенерируем еще несколько массивов

In [11]:

```
np.zeros(19, dtype = int)
```

Out[11]:

```
array([0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0])
```

In [12]:

```
np.ones((3,5))
```

Out[12]:

```
array([[1., 1., 1., 1., 1.],
       [1., 1., 1., 1., 1.],
       [1., 1., 1., 1., 1.]])
```

In [13]:

```
np.full((3,5), 3.14)
```

Out[13]:

```
array([[3.14, 3.14, 3.14, 3.14, 3.14],
       [3.14, 3.14, 3.14, 3.14, 3.14],
       [3.14, 3.14, 3.14, 3.14, 3.14]])
```

In [14]:

```
np.ones((3,5)) *3.14
```

Out[14]:

```
array([[3.14, 3.14, 3.14, 3.14, 3.14],
       [3.14, 3.14, 3.14, 3.14, 3.14],
       [3.14, 3.14, 3.14, 3.14, 3.14]])
```

In [15]:

```
np.arange(0,20,4)
```

Out[15]:

```
array([ 0,  4,  8, 12, 16])
```

In [16]:

```
np.linspace(0,1,5)
```

Out[16]:

```
array([0.  , 0.25, 0.5 , 0.75, 1.  ])
```

In [20]:

```
np.random.random((3,3))*10 + 5
```

Out[20]:

```
array([[ 5.24768866,  5.52834975,  9.79919724],
       [11.9189688 ,  6.08320723,  5.70362594],
       [12.99318033, 12.03376935,  8.2072786 ]])
```

In [21]:

```
np.random.normal(5,2,(5,5))
```

Out[21]:

```
array([[5.33136562, 4.9571746 , 3.57687469, 4.22154532, 1.47374   ],
       [6.74655662, 4.86721425, 5.57085208, 4.17676773, 5.35469298],
       [5.23259885, 3.32290858, 8.27510218, 7.61567186, 1.26837904],
       [8.82314281, 5.30523994, 7.06317837, 3.96893508, 6.18402403],
       [5.57672841, 7.56706833, 1.54703043, 6.12009909, 7.79077211]])
```

In [22]:

```
np.random.randint(0,10,(7,7))
```

Out[22]:

```
array([[5, 2, 7, 3, 2, 5, 7],
       [7, 1, 6, 6, 6, 9, 1],
       [9, 2, 0, 0, 8, 2, 1],
       [1, 1, 4, 7, 7, 5, 4],
       [5, 6, 3, 4, 1, 3, 7],
       [5, 6, 6, 9, 9, 8, 1],
       [9, 4, 7, 3, 9, 4, 0]])
```

In [23]:

```
np.eye(5)
```

Out[23]:

```
array([[1., 0., 0., 0., 0.],
       [0., 1., 0., 0., 0.],
       [0., 0., 1., 0., 0.],
       [0., 0., 0., 1., 0.],
       [0., 0., 0., 0., 1.]])
```

Еще можно создать неинициированный массив, просто для выделения памяти. Значения в нем будут произвольные, случайно оказавшиеся в соответствующих ячейках памяти

In [26]:

```
np.empty((7,6))
```

Out[26]:

```
array([[8.53183415e-312, 8.53137747e-312, 1.35566986e+131,
        5.36793230e+242, 3.92880998e+160, 5.50409570e+257],
       [2.59345414e+161, 5.56206154e+180, 8.17434450e+141,
        1.09918035e+155, 4.26209394e+180, 1.34452472e+161],
       [5.04620658e+180, 3.57885821e+246, 6.01121703e+175,
        4.73544978e+223, 8.90302838e+247, 9.77146823e-153],
       [1.96097649e+243, 6.99405043e-009, 2.93494529e+222,
        1.03063392e-113, 1.72731111e+097, 6.99416121e-009],
       [2.93494530e+222, 1.03063392e-113, 1.05894612e-153,
        7.61402229e-010, 8.98752578e-153, 4.83245960e+276],
       [8.03704345e-095, 1.95575364e-109, 1.05190656e-153,
        1.65980221e-307, 8.53183922e-312, 8.53137686e-312],
       [1.05177053e-153, 7.04134408e-009, 1.43178810e-065,
        1.35717430e+131, 1.05176086e-153, 1.74910380e+243]])
```

Рассмотрим несколько категорий простых операций с массивами:

- атрибуты массивов
- индексация массивов
- срезы массивов
- изменение формы массивов
- слияние и разбиение массивов

Атрибуты

In [ ]:

```
np.random.seed(0)
```

In [36]:

```
#x = np.random.randint(10, size = 6)
#x = np.random.randint(10, size = (3,5))
x = np.random.randint(10, size = (3,5,2))
x
```

Out[36]:

```
array([[[8, 4],
        [1, 6],
        [4, 6],
        [3, 7],
        [4, 0]],

       [[0, 9],
        [3, 3],
        [9, 5],
        [7, 4],
        [6, 9]],

       [[6, 2],
        [8, 1],
        [0, 1],
        [2, 3],
        [8, 1]]])
```

In [37]:

```
print('x.ndim ', x.ndim)
print('x.shape ', x.shape)
print('x.size ', x.size)
```

```
x.ndim  3
x.shape  (3, 5, 2)
x.size  30
```

In [38]:

```
print(x.dtype)
```

```
int32
```

In [39]:

```
x.itemsize
```

Out[39]:

```
4
```

In [40]:

```
x.nbytes
```

Out[40]:

```
120
```

In [41]:

```
x.size * x.itemsize
```

Out[41]:

```
120
```

Индексация

In [42]:

```
#x = np.random.randint(10, size = 6)
#x = np.random.randint(10, size = (3,5))
x = np.random.randint(10, size = (3,5,2))
x
```

Out[42]:

```
array([[[0, 6],
        [7, 7],
        [1, 3],
        [9, 5],
        [0, 5]],

       [[4, 6],
        [4, 2],
        [4, 7],
        [2, 6],
        [0, 7]],

       [[9, 3],
        [9, 9],
        [9, 0],
        [4, 8],
        [3, 3]]])
```

In [46]:

```
x[1,0,1]
```

Out[46]:

```
6
```

In [50]:

```
x[0,0,:]
```

Out[50]:

```
array([0, 6])
```

Срезы

In [53]:

```
x = np.arange(10)
x
```

Out[53]:

```
array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9])
```

Для одномерных массивов срезы работают так же, как для списков.

Самостоятельно

- первые 5 элементов
- элементы после индекса 5
- каждый второй элемент
- каждый второй элемент, начиная с индекса 2
- все элементы в обратном порядке
- каждый второй элемент в обратном порядке

In [55]:

```
x[:5]
```

Out[55]:

```
array([0, 1, 2, 3, 4])
```

In [56]:

```
x[5:]
```

Out[56]:

```
array([5, 6, 7, 8, 9])
```

In [57]:

```
x[::2]
```

Out[57]:

```
array([0, 2, 4, 6, 8])
```

In [60]:

```
x[2::2]
```

Out[60]:

```
array([2, 4, 6, 8])
```

In [58]:

```
x[::-1]
```

Out[58]:

```
array([9, 8, 7, 6, 5, 4, 3, 2, 1, 0])
```

In [59]:

```
x[::-2]
```

Out[59]:

```
array([9, 7, 5, 3, 1])
```

Для многомерных массивов все работает похожим образом, срезы разделяются запятыми

In [61]:

```
x = np.random.randint(10, size = (3,5))
x
```

Out[61]:

```
array([[2, 5, 6, 8, 1],
       [0, 9, 9, 7, 6],
       [2, 2, 6, 1, 8]])
```

In [62]:

```
x[:2,:3]
```

Out[62]:

```
array([[2, 5, 6],
       [0, 9, 9]])
```

In [63]:

```
x[:3,::2]
```

Out[63]:

```
array([[2, 6, 1],
       [0, 9, 6],
       [2, 6, 8]])
```

In [64]:

```
x[::1,::-1]
```

Out[64]:

```
array([[1, 8, 6, 5, 2],
       [6, 7, 9, 9, 0],
       [8, 1, 6, 2, 2]])
```

In [65]:

```
x
```

Out[65]:

```
array([[2, 5, 6, 8, 1],
       [0, 9, 9, 7, 6],
       [2, 2, 6, 1, 8]])
```

In [66]:

```
x[:,0]
```

Out[66]:

```
array([2, 0, 2])
```

In [67]:

```
x[1,:]
```

Out[67]:

```
array([0, 9, 9, 7, 6])
```

Срезы массивов возвращают представления, а не копии (!)

In [68]:

```
x
```

Out[68]:

```
array([[2, 5, 6, 8, 1],
       [0, 9, 9, 7, 6],
       [2, 2, 6, 1, 8]])
```

In [69]:

```
x_sub = x[:2,:2]
x_sub
```

Out[69]:

```
array([[2, 5],
       [0, 9]])
```

In [70]:

```
x_sub[0] = 100
x_sub
```

Out[70]:

```
array([[100, 100],
       [  0,   9]])
```

In [71]:

```
x
```

Out[71]:

```
array([[100, 100,   6,   8,   1],
       [  0,   9,   9,   7,   6],
       [  2,   2,   6,   1,   8]])
```

Если нужно создать копию:

In [72]:

```
x_sub_copy = x[:2,:2].copy()
x_sub_copy[0,0] = 10000
print(x_sub_copy)
print(x)
```

```
[[10000   100]
 [    0     9]]
[[100 100   6   8   1]
 [  0   9   9   7   6]
 [  2   2   6   1   8]]
```

Изменение формы

In [78]:

```
grid = np.arange(1,13).reshape((4,3))
grid
```

Out[78]:

```
array([[ 1,  2,  3],
       [ 4,  5,  6],
       [ 7,  8,  9],
       [10, 11, 12]])
```

In [79]:

```
grid.reshape((2,6))
```

Out[79]:

```
array([[ 1,  2,  3,  4,  5,  6],
       [ 7,  8,  9, 10, 11, 12]])
```

In [80]:

```
grid.reshape((12,1))
```

Out[80]:

```
array([[ 1],
       [ 2],
       [ 3],
       [ 4],
       [ 5],
       [ 6],
       [ 7],
       [ 8],
       [ 9],
       [10],
       [11],
       [12]])
```

In [81]:

```
grid.reshape((1,12))
```

Out[81]:

```
array([[ 1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12]])
```

Слияние и разбиение массивов

В NumPy есть следующие функции для объединения массивов:

- concatenate
- vstack
- hstack

In [82]:

```
x = np.array([1,2,3])
y = np.array([4,5,6])
```

In [83]:

```
np.concatenate((x,y))
```

Out[83]:

```
array([1, 2, 3, 4, 5, 6])
```

In [96]:

```
x = np.random.randint(10, size = (5,3))
y = np.random.randint(10, size = (5,3))
print(x)
print(y)
```

```
[[8 0 5]
 [8 6 8]
 [6 3 7]
 [9 7 1]
 [3 6 8]]
[[3 7 1]
 [0 7 7]
 [3 1 2]
 [7 2 3]
 [4 1 2]]
```

In [97]:

```
np.concatenate((x,y),axis = 1)
```

Out[97]:

```
array([[8, 0, 5, 3, 7, 1],
       [8, 6, 8, 0, 7, 7],
       [6, 3, 7, 3, 1, 2],
       [9, 7, 1, 7, 2, 3],
       [3, 6, 8, 4, 1, 2]])
```

In [98]:

```
np.hstack([x,y])
```

Out[98]:

```
array([[8, 0, 5, 3, 7, 1],
       [8, 6, 8, 0, 7, 7],
       [6, 3, 7, 3, 1, 2],
       [9, 7, 1, 7, 2, 3],
       [3, 6, 8, 4, 1, 2]])
```

In [99]:

```
x = np.random.randint(10, size = (4,3))
y = np.random.randint(10, size = (5,3))
print(x)
print(y)
```

```
[[8 0 6]
 [8 7 0]
 [3 0 2]
 [5 7 2]]
[[5 3 1]
 [7 2 1]
 [5 6 7]
 [3 4 3]
 [8 9 2]]
```

In [100]:

```
np.vstack([x,y])
```

Out[100]:

```
array([[8, 0, 6],
       [8, 7, 0],
       [3, 0, 2],
       [5, 7, 2],
       [5, 3, 1],
       [7, 2, 1],
       [5, 6, 7],
       [3, 4, 3],
       [8, 9, 2]])
```

Есть еще аналогичная функция np.dstack для объединения массивов по третье оси

Функции для разбиения массивов:

- split
- hsplit
- vsplit

Этим функциям необходимо передать список идексов, задающих точки раздела

In [101]:

```
x = np.arange(10)
x
```

Out[101]:

```
array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9])
```

In [104]:

```
nodes = [3,5]
x1, x2, x3 = np.split(x,nodes)
```

In [105]:

```
print(x1)
print(x2)
print(x3)
```

```
[0 1 2]
[3 4]
[5 6 7 8 9]
```

In [106]:

```
x = np.arange(100).reshape([10,10])
x
```

Out[106]:

```
array([[ 0,  1,  2,  3,  4,  5,  6,  7,  8,  9],
       [10, 11, 12, 13, 14, 15, 16, 17, 18, 19],
       [20, 21, 22, 23, 24, 25, 26, 27, 28, 29],
       [30, 31, 32, 33, 34, 35, 36, 37, 38, 39],
       [40, 41, 42, 43, 44, 45, 46, 47, 48, 49],
       [50, 51, 52, 53, 54, 55, 56, 57, 58, 59],
       [60, 61, 62, 63, 64, 65, 66, 67, 68, 69],
       [70, 71, 72, 73, 74, 75, 76, 77, 78, 79],
       [80, 81, 82, 83, 84, 85, 86, 87, 88, 89],
       [90, 91, 92, 93, 94, 95, 96, 97, 98, 99]])
```

In [107]:

```
upper, lower = np.vsplit(x,[5])
```

In [108]:

```
upper
```

Out[108]:

```
array([[ 0,  1,  2,  3,  4,  5,  6,  7,  8,  9],
       [10, 11, 12, 13, 14, 15, 16, 17, 18, 19],
       [20, 21, 22, 23, 24, 25, 26, 27, 28, 29],
       [30, 31, 32, 33, 34, 35, 36, 37, 38, 39],
       [40, 41, 42, 43, 44, 45, 46, 47, 48, 49]])
```

In [109]:

```
lower
```

Out[109]:

```
array([[50, 51, 52, 53, 54, 55, 56, 57, 58, 59],
       [60, 61, 62, 63, 64, 65, 66, 67, 68, 69],
       [70, 71, 72, 73, 74, 75, 76, 77, 78, 79],
       [80, 81, 82, 83, 84, 85, 86, 87, 88, 89],
       [90, 91, 92, 93, 94, 95, 96, 97, 98, 99]])
```

In [110]:

```
left, right = np.hsplit(x,[2])
```

In [111]:

```
left
```

Out[111]:

```
array([[ 0,  1],
       [10, 11],
       [20, 21],
       [30, 31],
       [40, 41],
       [50, 51],
       [60, 61],
       [70, 71],
       [80, 81],
       [90, 91]])
```

In [112]:

```
right
```

Out[112]:

```
array([[ 2,  3,  4,  5,  6,  7,  8,  9],
       [12, 13, 14, 15, 16, 17, 18, 19],
       [22, 23, 24, 25, 26, 27, 28, 29],
       [32, 33, 34, 35, 36, 37, 38, 39],
       [42, 43, 44, 45, 46, 47, 48, 49],
       [52, 53, 54, 55, 56, 57, 58, 59],
       [62, 63, 64, 65, 66, 67, 68, 69],
       [72, 73, 74, 75, 76, 77, 78, 79],
       [82, 83, 84, 85, 86, 87, 88, 89],
       [92, 93, 94, 95, 96, 97, 98, 99]])
```

для трехмерных массивов аналогично есть функция dsplit

Самостоятельно

- Создать массивы случайных чисел размерностью 20 на 10 и 20 на 40
- Объединить эти массивы по первому измерению
- Разбить полученный массив на 2 равные половины, размерностью 20 на 25
- Преобразовать первый полученный массив в вектор-строку, а второй в вектор-столбец

In [113]:

```

```

In [ ]:

```

```

In [ ]:

```

```

In [ ]:

```
```

Выводы

In [ ]:

```
```