

NumPy

Introdução

Indexação básica e fatiamento

A indexação de arrays unidimensionais segue a mesma ideia de indexação de listas.

```
In [20]: import numpy as np

In [21]: array = np.array([0,1,2,3,4,5,6,7,8,9])

In [22]: array[8]
Out[22]: 8

In [23]: array[4:7]
Out[23]: array([4, 5, 6])

In [24]: array[4:7] = 66

In [25]: array
Out[25]: array([ 0,  1,  2,  3, 66, 66, 66,  7,  8,  9])
```

O valor escalar 66 é propagado para todos elementos da fatia (4:7) que são os elementos de índices 4, 5 e 6.

Indexação básica e fatiamento

Fatias são visualizações do array original, não são cópias. Sendo assim, qualquer alteração será refletida no array original.

```
In [12]: import numpy as np
```

```
In [13]: array = np.array([0,1,2,3,4,5,6,7,8,9])
```

```
In [14]: array
```

```
Out[14]: array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9])
```

```
In [15]: fatia = array[4:9]
```

```
In [16]: fatia
```

```
Out[16]: array([4, 5, 6, 7, 8])
```

```
In [17]: fatia[3] = 17
```

```
In [18]: fatia
```

```
Out[18]: array([ 4,  5,  6, 17,  8])
```

```
In [19]: array
```

```
Out[19]: array([ 0,  1,  2,  3,  4,  5,  6, 17,  8,  9])
```

A fatia `[:]` fará uma atribuição a todos os valores em um array.

```
In [1]: import numpy as np
In [2]: array = np.array([10,20,30,40,50,60,70,80])
In [3]: array
Out[3]: array([10, 20, 30, 40, 50, 60, 70, 80])
In [4]: fatia = array[2:6]
In [5]: fatia
Out[5]: array([30, 40, 50, 60])
In [6]: fatia[:] = 100
In [7]: fatia
Out[7]: array([100, 100, 100, 100])
In [8]: array
Out[8]: array([ 10,  20, 100, 100, 100, 100,  70,  80])
```

Se precisarmos alterar uma cópia mantendo o array original podemos criar a fatia usando `array[x:y].copy()`.

```
In [4]: import numpy as np

In [5]: array = np.array([0,1,2,3,4,5,6,7,8,9])

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

In [7]: fatia = array[4:9].copy()

In [8]: fatia
Out[8]: array([4, 5, 6, 7, 8])
```

```
In [9]: fatia[3] = 17

In [10]: fatia
Out[10]: array([ 4,  5,  6, 17,  8])

In [11]: array
Out[11]: array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9])
```

Indexação básica e fatiamento

Em um array bidimensional, os elementos em cada índice não são mais escalares, mas são arrays unidimensionais. Desta forma podemos acessar seus elementos recursivamente.

```
In [26]: import numpy as np
```

```
In [27]: array2d = np.array([[10, 20, 30],[40, 50, 60],[70, 80, 90]])
```

```
In [28]: array2d[1]
```

O elemento na posição 1 é um array unidimensional

```
Out[28]: array([40, 50, 60])
```

```
In [29]: array2d[1][1]
```

```
Out[29]: 50
```

```
In [30]: array2d[1,1]
```

```
Out[30]: 50
```

Existem duas formas de acessar um elemento do array interno.

		axis 1		
		0	1	2
axis 0	0	0,0	0,1	0,2
	1	1,0	1,1	1,2
	2	2,0	2,1	2,2

Pensar no eixo 0 como linhas e no eixo 1 como colunas pode ajudar.

NumPy

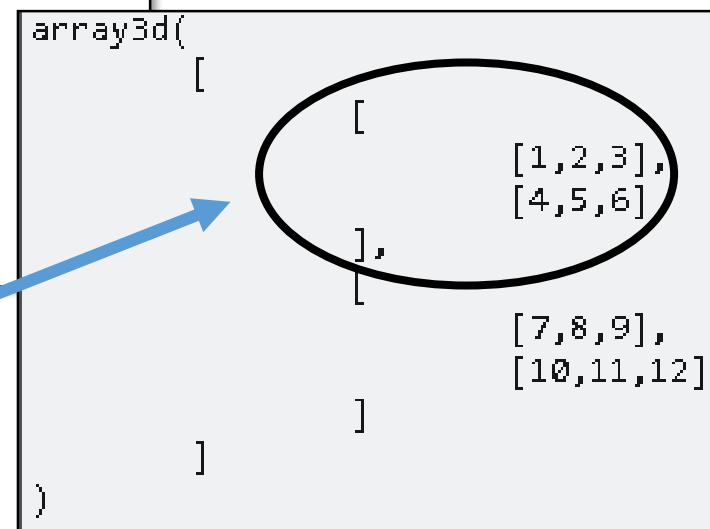
Indexação básica e fatiamento

Vamos ver agora a indexação de arrays multidimensionais. Veja a seguir o array de três dimensões (2 x 2 x 3).

```
In [1]: import numpy as np
In [2]: array3d = np.array([[[1,2,3],[4,5,6]], [[7,8,9],[10,11,12]]])
In [3]: array3d
Out[3]:
array([[[ 1,  2,  3],
        [ 4,  5,  6]],
       [[ 7,  8,  9],
        [10, 11, 12]]])
```

O array3d na posição 0 é um array 2 x 3.

```
In [7]: array3d[0]
Out[7]:
array([[1, 2, 3],
       [4, 5, 6]])
```



NumPy

Indexação básica e fatiamento

Tanto valores escalares quanto arrays podem ser atribuídos a `array3d[0]`:

```
In [4]: valor_antigo = array3d[0].copy()
```

```
In [5]: array3d[0] = 77
```

```
In [6]: array3d
```

```
Out[6]:  
array([[[77, 77, 77],  
        [77, 77, 77]],  
       [[ 7,  8,  9],  
        [10, 11, 12]]])
```

```
In [7]: array3d[0] = valor_antigo
```

```
In [8]: array3d
```

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


NumPy

Indexação básica e fatiamento

Da mesma forma `array3d[1, 0]` contém todos os valores cujos índices começam com (1, 0), formando um array unidimensional.

```
In [9]: array3d[1, 0]
Out[9]: array([7, 8, 9])
```

```
array3d(
  [
    [
      [1,2,3],
      [4,5,6]
    ],
    [
      [7,8,9],
      [10,11,12]
    ]
  ]
)
```

```
In [10]: a = array3d[1]
```

```
In [11]: a
```

```
Out[11]: array([[ 7,  8,  9],
                [10, 11, 12]])
```

```
In [12]: a[0]
```

```
Out[12]: array([7, 8, 9])
```

Aqui nós temos o mesmo resultado, porém, fazendo por etapas (em dois passos).

NumPy

Indexação básica e fatiamento

Assim como os objetos unidimensionais, por exemplo, as listas, os ndarrays também podem ser fatiados:

```
In [16]: array = np.array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9])
```

```
In [17]: array
```

```
Out[17]: array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9])
```

```
In [18]: array[2:7]
```

```
Out[18]: array([2, 3, 4, 5, 6])
```

Indexação básica e fatiamento

Fatiar um array bidimensional é um pouco diferente de fatiar um array unidimensional, por exemplo.

```
In [29]: array2d = np.array([[1, 2, 3],[4, 5, 6],[7, 8, 9]])  
  
In [30]: array2d  
Out[30]:  
array([[1, 2, 3],  
       [4, 5, 6],  
       [7, 8, 9]])  
  
In [31]: array2d[:2]  
Out[31]:  
array([[1, 2, 3],  
       [4, 5, 6]])
```

Neste exemplo estamos fatiando à partir do eixo 0, que é o primeiro eixo, pegando as duas primeiras linhas.

Como se pegássemos estes dois resultados: `array2d[0]` e `array2d[1]`.

Indexação básica e fatiamento

Podemos passar várias fatias.

```
In [36]: array2d = np.array([[1, 2, 3],[4, 5, 6],[7, 8, 9]])  
In [37]: array2d  
Out[37]:  
array([[1, 2, 3],  
       [4, 5, 6],  
       [7, 8, 9]])  
In [38]: array2d[:2, 1:]  
Out[38]:  
array([[2, 3],  
       [5, 6]])
```

Neste exemplo estamos pegando as duas primeiras linhas (0 e 1) e a partir da primeira coluna (colunas 1 e 2).

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

Indexação básica e fatiamento

Podemos passar várias fatias.

```
In [36]: array2d = np.array([[1, 2, 3],[4, 5, 6],[7, 8, 9]])
```

```
In [37]: array2d
```

```
Out [37]:  
array([[1, 2, 3],  
       [4, 5, 6],  
       [7, 8, 9]])
```

Neste exemplo estamos pegando as duas primeiras linhas (0 e 1) e à partir da primeira coluna (colunas 1 e 2).

```
In [38]: array2d[:2, 1:]
```

```
Out [38]:  
array([[2, 3],  
       [5, 6]])
```

```
In [39]: array2d[1, :2]
```

```
Out [39]: array([4, 5])
```

Neste exemplo estamos pegando a segunda linha e as duas primeiras colunas

```
In [40]: array2d[:2, 2]
```

```
Out [40]: array([3, 6])
```

Neste exemplo estamos pegando as linhas 0 e 1 e a coluna 2.





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

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

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

Indexação básica e fatiamento

Veja uma ilustração detalhando o fatiamento:

	Expression	Shape
	<code>arr[:2, 1:]</code>	<code>(2, 2)</code>
	<code>arr[2]</code> <code>arr[2, :]</code> <code>arr[2:, :]</code>	<code>(3,)</code> <code>(3,)</code> <code>(1, 3)</code>
	<code>arr[:, :2]</code>	<code>(3, 2)</code>
	<code>arr[1, :2]</code> <code>arr[1:2, :2]</code>	<code>(2,)</code> <code>(1, 2)</code>

FIM