Funcionalidades essenciais Reindexação e descarte de entradas de eixo



Funcionalidades essenciais – Reindexação

O método *reindex* cria um novo objeto com os dados de acordo com um novo índice. Primeiro vamos criar um objeto Series.

```
import pandas as pd
dados = pd.Series([100.0, 75.85, 10.18, -5.4], index=['um', 'dois', 'três', 'quatro'])
dados
          100.00
um
dois
           75.85
           10.18
três
           -5.40
quatro
dtype: float64
```



Funcionalidades essenciais – Reindexação

Agora vamos aplicar o método *reindex* ao objeto Series criado.

```
dados_reindexados = dados.reindex(['quatro','três','dois','um','zero'])
dados reindexados
                                          Se um índice não estava
         -5.40
quatro
                                          presente, é exibido o
três
           10.18
                                          indicador de dados ausentes,
dois
     75.85
                                          como por exemplo o índice
      100.00
um
                                          'zero'.
zero
              NaN
dtype: float64
```





Funcionalidades essenciais – Reindexação

Podemos reindexar também as linhas e/ou colunas de um DataFrame. Vamos criar um objeto DataFrame.

```
frame = pd.DataFrame(np.arange(9).reshape((3,3)), index=['a','b','c'], columns=['Vitória','Vila Velha','Viana'])
frame
   Vitória Vila Velha Viana
```



Funcionalidades essenciais – Reindexação

Para reindexar as linhas, basta passar apenas uma sequência.

frame_reindexado = frame.reindex(['d', 'c', 'b', 'a'						
frame_reindexado						
	Vitória	Vila Velha	Viana			
d	NaN	NaN	NaN			
С	6.0	7.0	8.0			
b	3.0	4.0	5.0			
а	0.0	1.0	2.0			



Funcionalidades essenciais – Reindexação

As colunas podem ser reindexadas com a palavra reservada columns.

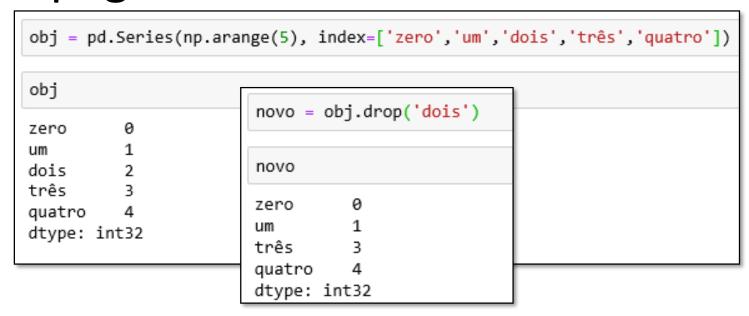
cio	cidades = ['Viana','Vitória','Vila Velha','Cariacica']							
fra	frame.reindex(columns=cidades)							
	Viana	Vitória	Vila Velha	Cariacica				
a	2	0	1	NaN				
b	5	3	4	NaN				
С	8	6	7	NaN				





Funcionalidades essenciais – Descartando entradas de um eixo

Podemos utilizar o método drop para devolver um objeto com o valor ou os valores indicados apagados de um eixo.





Funcionalidades essenciais – Descartando entradas de um eixo

Com DataFrame, podemos apagar os valores dos índices de qualquer eixo. Vamos criar um objeto DataFrame.

```
dados = pd.DataFrame(np.arange(9).reshape((3,3)), index=['Viana','Vila Velha','Vitória'], columns=['um','dois','três'])

dados

um dois três

Viana 0 1 2

Vila Velha 3 4 5

Vitória 6 7 8
```

Funcionalidades essenciais – Descartando entradas de um eixo

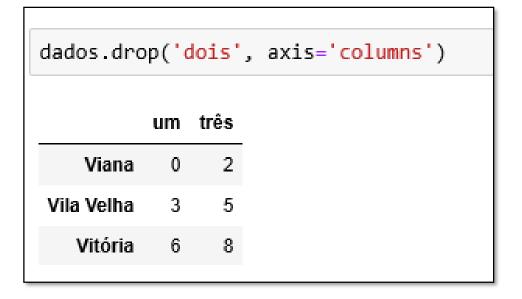
Podemos usar o método drop passando uma sequência de rótulos para descartar valores das linhas (eixo 0).



Funcionalidades essenciais – Descartando entradas de um eixo

Para descartar valores das colunas usamos axis=1 ou axis='columns'.

dados.dro	lados.drop('um', axis=1)					
	dois	três				
Viana	1	2				
Vila Velha	4	5				
Vitória	7	8				



Funcionalidades essenciais – Descartando entradas de um eixo

O drop, assim como muitas outras funções que modificam o tamanho ou o formato de uma *Series* ou um *DataFrame*, podem alterar um objeto in-place, ou seja, alterar o objeto original sem retornar um novo objeto. Muito cuidado, pois os dados serão realmente descartados do objeto.

dados.dro	dados.drop('dois', axis='columns', inplace=True)				
dados	dados				
	um	três			
Viana	0	2			
Vila Velha	3	5			
Vitória	6	8			



FIM

