Funcionalidades essenciais Indexação, seleção e filtragem



Funcionalidades essenciais – Indexação, seleção e filtragem

A indexação de séries funciona como a indexação de arrays NumPy, porém, podemos utilizar os valores de índice da Series em vez de utilizar

somente inteiros.

```
In [1]: import pandas as pd
In [4]: obj = pd.Series([0.,1.,2.,3.], index=['a','b','c','d'])
In [5]: obj
Out[5]: a
        dtype: float64
```

```
In [4]: obj[1]
Out[4]: 1.0
In [5]: obj['b']
Out[5]: 1.0
```

O fatiamento usando rótulos, inclui o final, diferente do fatiamento com número des índices.

```
In [12]:
         obj[1:4]
Out[12]: b
              1.0
              2.0
               3.0
         dtype: float64
         obj['b':'d']
In [14]:
               2.0
               3.0
          dtype: float64
```

Funcionalidades essenciais – Indexação, seleção e filtragem

Podemos também retornar as linhas com base em seu valor.

```
In [27]: obj = pd.Series([0.,1.,2.,3.,2.,2.,1.], index=['a','b','c','d','e','f','g'])
In [25]: obj[obj == 2]
Out[25]: c
         dtype: float64
         obj[obj < 3]
In [26]:
Out[26]: a
              1.0
              2.0
              2.0
         dtype: float64
```





Funcionalidades essenciais – Indexação, seleção e filtragem

Podemos atribuir um único valor a um intervalo conforme exemplo abaixo.



Funcionalidades essenciais – Indexação, seleção e filtragem

Vamos agora ver a indexação em um DataFrame.

Primeiro vamos criar o nosso DataFrame.

```
import numpy as np
dados = pd.DataFrame(np.arange(16).reshape((4, 4)),
                     index=["Vila Velha", "Vitória", "Viana", "Cariacica"],
                     columns=["um", "dois", "três", "quatro"])
dados
         um dois três quatro
Vila Velha
   Vitória
   Viana
                        11
Cariacica 12
```



Funcionalidades essenciais – Indexação, seleção e filtragem

Podemos indexar utilizando fatiamento e a seleção de dados com um array booleano, retornando linhas do DataFrame:

dados[:2]				
	um	dois	três	quatro
Vila Velha	0	1	2	3
Vitória	4	5	6	7

dados[dados["um"] < 8]					
	um	dois	três	quatro	
Vila Velha	0	1	2	3	
Vitória	4	5	6	7	

Funcionalidades essenciais – Indexação, seleção e filtragem

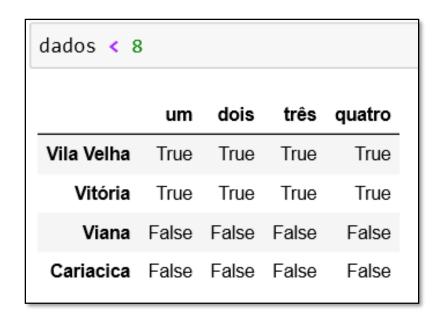
Passar um único elemento ou uma lista para o operador [] seleciona colunas, veja:

```
dados["um"]
Vila Velha
Vitória
Viana
Cariacica
Name: um, dtype: int32
```



Funcionalidades essenciais – Indexação, seleção e filtragem

Outro caso de uso está na indexação com um DataFrame booleano, como aquele gerado por uma comparação escalar:



dados[dad	ados[dados < 8] = 0				
dados					
	um	dois	três	quatro	
Vila Velha	0	0	0	0	
Vitória	0	0	0	0	
Viana	8	9	10	11	
Cariacica	12	13	14	15	

Funcionalidades essenciais – Indexação, seleção e filtragem

Seleção com loc e iloc

Os operadores loc e iloc permitem selecionar um subconjunto de linhas e colunas de um DataFrame com uma notação semelhante àquela do NumPy, usando rótulos de eixo (loc) ou inteiros (iloc).

Funcionalidades essenciais – Indexação, seleção e filtragem

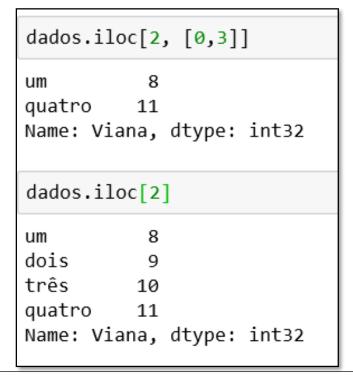
Vamos selecionar uma linha com duas colunas de nosso DataFrame:

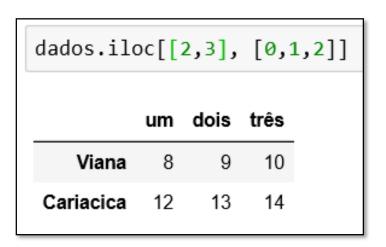
```
dados.loc['Viana', ['um', 'quatro']]

um 8
quatro 11
Name: Viana, dtype: int32
```

Funcionalidades essenciais – Indexação, seleção e filtragem

Agora vamos realizar seleções usando inteiros com iloc:





Funcionalidades essenciais – Indexação, seleção e filtragem

Tanto loc quanto iloc trabalham com fatias, além de rótulos únicos ou listas de rótulos:

dados.loc[:'Viana', 'um':'três']				
	um	dois	três	
Vila Velha	0	1	2	
Vitória	4	5	6	
Viana	8	9	10	



FIM

