Series

January 16, 2019

1 Series

O primeiro tipo de dado que aprenderemos é a Serie. Vamos importar Pandas e explorar tal objeto.

A Serie é muito semelhante a uma matriz NumPy (na verdade, ela é construída em cima do objeto de matriz NumPy). O que diferencia a matriz NumPy de uma Série, é que uma Serie pode ter rótulos de eixos, o que significa que pode ser indexado por um rótulo, em vez de apenas uma localização numérica. Também não precisa manter dados numéricos, ele pode conter qualquer objeto Python arbitrário.

Vamos explorar este conceito através de alguns exemplos:

```
In [6]: import numpy as np
        import pandas as pd
```

1.0.1 Criando uma Serie

Você pode converter uma lista, numpy array ou dicionário para uma série:

```
In [18]: labels = ['a', 'b', 'c'] #lista
```

```
In [12]: minha_lista = [10,20,30] #lista
```

```
In [13]: arr = np.array([10,20,30]) #array
```

```
In [15]: d = {'a':10, 'b':20, 'c':30} #dicionario
```

** Transformando em Series **

```
In [19]: pd.Series(data=minha_lista, index=labels)
```

```
Out[19]: a    10
         b    20
         c    30
         dtype: int64
```

```
In [20]: series = pd.Series(data=minha_lista, index=labels)
```

```
In [22]: series['a']
```

```
Out[22]: 10
```

```
In [24]: series['b']

Out[24]: 20

In [25]: pd.Series([sum,print,len])

Out[25]: 0      <built-in function sum>
         1    <built-in function print>
         2      <built-in function len>
         dtype: object

In [26]: pd.Series(minha_lista,labels)

Out[26]: a    10
         b    20
         c    30
         dtype: int64
```

** NumPy Arrays **

```
In [6]: pd.Series(arr) #usando arrays

Out[6]: 0    10
        1    20
        2    30
        dtype: int32

In [7]: pd.Series(arr,labels) #arrays e numeros

Out[7]: a    10
        b    20
        c    30
        dtype: int32
```

** Dicionários **

```
In [8]: pd.Series(d) #dicionario

Out[8]: a    10
        b    20
        c    30
        dtype: int64
```

1.0.2 Dados nas Series

Uma série de pandas pode conter uma variedade de tipos de objeto:

```
In [9]: pd.Series(data=labels)
```

```
Out[9]: 0    a
        1    b
        2    c
        dtype: object
```

```
In [10]: # Mesmo funções (embora seja improvável que você use isso)
         pd.Series([sum,print,len])
```

```
Out[10]: 0      <built-in function sum>
         1    <built-in function print>
         2      <built-in function len>
         dtype: object
```

1.1 Usando um Índice

A chave para usar uma Serie é entender seu índice. O Pandas faz uso desses nomes ou números de índice, permitindo pesquisas rápidas de informações (funciona como uma tabela de hash ou dicionário).

Vamos ver alguns exemplos de como pegar informações de uma Serie. Vamos criar duas Series, ser1 e ser2:

```
In [31]: serie1 = pd.Series([1,2,3,4], ['EUA', 'Alemanha', 'URSS', 'Japao'])
```

```
In [32]: serie1
```

```
Out[32]: EUA         1
         Alemanha    2
         URSS        3
         Japao       4
         dtype: int64
```

```
In [33]: serie2 = pd.Series([1,2,3,4], ['EUA', 'Alemanha', 'Italia', 'Japao'])
```

```
In [34]: serie2
```

```
Out[34]: EUA         1
         Alemanha    2
         Italia      3
         Japao       4
         dtype: int64
```

```
In [35]: serie1+serie2 #operacoes baseada no indice(o indice eh a primeira coluna)
```

```
Out[35]: Alemanha    4.0
         EUA         2.0
         Italia      NaN
         Japao       8.0
         URSS        NaN
         dtype: float64
```

As operações também são feitas com base no índice:
Vamos parar aqui por enquanto e passar para a DataFrames, que expandirá o conceito da Serie!