Pandas: Indexação, Seleção e atribuição

Professor: Nator Junior Carvalho da Costa

September 24, 2022

Sumário

Sumário

- 1 Pandas;
- 2 Series;
- 3 Atividade Prática.
- 4 DataFrame;
- 5 Atividade Prática;
- 6 LOC, ILOC, EX;
- 7 Atividade Prática.

Pandas

Pandas



Pandas é uma biblioteca Python para análise de dados.

Instalação da biblioteca via PyPI:

pip install pandas

Iniciando com Pandas

Series



Para utilizar a biblioteca basta realizar a importação Instalação da biblioteca via PyPI:

```
# Forma tradicional de importar a biblioteca pandas
import pandas as pd
```

Series

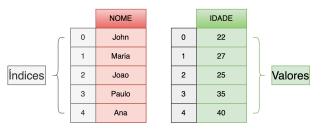


Figura 1: Classe Series do Pandas.

Series

Construindo séries a partir de um dicionário

```
1 >>> d = {'a': 1, 'b': 2, 'c': 3}
2 >>> ser = pd.Series(data=d)
3 >>> ser
4 a 1
5 b 2
6 c 3
7 dtype: int64
```

Operações com Series

Series

Construindo séries a partir de uma lista.

```
1 >>> d = [4,6,3,7]
2 >>> ser = pd.Series(data=d)
```

Operações:

- Para calcular a média, ser.mean();
- Para calcular a mediana, ser.median();
- Para calcular a variância, ser.var();
- Para calcular a desvio padrão, ser.std().

Series: Mesmo tipo de dados

```
1  >>> import pandas as pd
2  >>> serie = pd.Series([42, 99, -1])
3  >>> serie
4  0     42
5  1     99
6  2     -1
7  dtype: int64
```

Series: Tipos diferentes de dados

Acessando elementos

```
1 >>> serie[0]
2 42
3 >>> serie[2]
4 -1
```

Criando índices personalizados

```
>>> import pandas as pd
>>> serie = pd.Series(
    [200, 350, 550],
    index=['banana', 'prato feito', 'big mac']
>>> serie
banana
               200
prato feito 350
           550
big mac
dtype: int64
```

Acessando elementos

```
| >>> serie['big mac']
| 2 | 550
```

```
>>> import pandas as pd
  >>> serie = pd.Series(
      [200, 350, 550],
      index=[(0, 0), (0, 1), (0, 2)]
  >>> serie
  (0, 0) 200
  (0, 1) 350
9 (0, 2) 550
10 dtype: int64
11 >>> serie[(0,1)]
  350
12
```

Atividade 01

Series

Determine a média e variância das notas finais de 12 alunos de uma aula de Química. As notas são apresentadas na tabela abaixo e faça um programa que imprima o resultado das operações.

$$\begin{bmatrix} 8.0 & 9.1 & 7.5 & 7.4 & 7.6 & 7.8 & 8.0 & 9.3 & 7.0 & 7.0 \\ 7.2 & 7.5 & & & & & & & \end{bmatrix}$$
 (1)

Resolução

Series

Construindo séries a partir de um dicionário

```
1 >>> d =[8.0,9.1,7.5,7.4,7.6,7.8,8.0,9.3,7.0,7.0,7.2,7.5]
2 >>> ser = pd.Series(data=d)
3 >>> print("A média é:",ser.mean())
4 >>> print("A mediana é:",ser.median())
5 >>> print("A variância é:",ser.var())
6 >>> print("A desvio padão é:",ser.std())
```

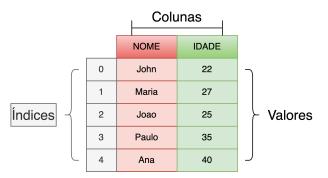


Figura 2: Classe DataFrame do Pandas.

DataFrame

Criando um DataFrame

```
>>> import pandas as pd
>>> df = pd.DataFrame({
    'calorias': [200, 350, 550],
    'gordura (%)':[0, 6, 15]
})
>>> df
   calorias gordura (%)
        200
0
        350
        550
                       15
```

DataFrame

Criando um DataFrame e definindo um índice.

```
>>> import pandas as pd
  >>> df = pd.DataFrame({
           'calorias': [200, 350, 550],
           'gordura (%)':[0, 6, 15]
       index=['banana', 'prato feito', 'big mac']
                calorias gordura (%)
                     200
  banana
                     350
  prato feito
                                    15
  big mac
                     550
12
```

DataFrame

Acessando os valores de uma coluna do DataFrame.

DataFrame

A forma de acessar um dado de um DataFrame por meio de índices é a seguinte:

```
dataframe[<coluna>][<linha>]
```

Exemplo:

DataFrame

Acessando os valores de uma linha do DataFrame.

```
>>> df = pd.DataFrame({
       'calorias': [200, 350, 550],
      'gordura (%)':[0, 6, 15]
  })
  >>> df
  banana
  prato feito
  big mac
              15
  Name: gordura (%), dtype: int64
  >>> df[1:2]
10
      calorias gordura (%)
11
           350
12
13
```

LOC

```
#podemos chamar uma linha pelo seu indice
df.loc[5]
#ou com um array de indices
df.loc[[0,1,2]]

# uma fatia, do quarto ao sétimo elemento
# (note que diferente do python puro, neste método
# a chave inicial e final estarão presente no resultado)
df.loc[4:8]
```

LOC

```
#busca simples
df.loc[(df['calorias']) >= 200]
#busca composta
df.loc[(df['calorias']) >= 10)&(df['calorias'] <= 350)]</pre>
```

LOC

```
#Redefinindo valores.

2 df.loc[(df['calorias']) >= 200, 'calorias'] = 210
```

ILOC

```
# Linhas:
df.iloc[0] # Selecionado a primeira linha do dataset
df.iloc[-1] # Selecionando a última linha
```

ILOC

```
# Colunas:

# Todos os dados da primeira coluna do dataset

df.iloc[:,0]

# Do primeiro ao quinto dado da última coluna

df.iloc[0:5,-1]
```

```
# Seleção de múltiplas linhas e colunas:
# resgatando as primeiras três linhas do dataset

df.iloc[0:3]
# todos os dados da segunda e terceira coluna

df.iloc[:, 1:3]
# 1º,3º e 5º elementos e 6ª a 8ª colunas

df.iloc[[0,2,4], 5:8]
```

Obrigado!