



A criação da biblioteca



- A biblioteca Pandas foi criada para facilitar a realização de análises e visualização de dados;
- Possui estruturas de dados diferentes, a Series e o Dataframe.





Importando a biblioteca



Com o ambiente Python devidamente configurado, a biblioteca Pandas pode ser importada dentro de um código da seguinte forma:

import pandas as pd



Series



A Series é uma das estruturas oferecidas pela biblioteca Pandas e sua organização é similar a um array unidimensional ou uma lista de valores.

Os objetos do tipo Series possuem um índice para identificar cada valor, o index.

Para criar um objeto do tipo Series contendo quatro elementos, basta:

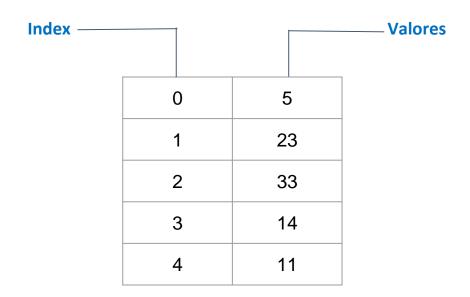
```
notas = pd.Series([5,23,33,14,11])
```

Como não especificamos rótulos para os valores inseridos, a biblioteca os atribui de forma automática, indo de 0 a 4.



Series







Series



Também é possível criar uma Series determinando uma chave para cada um dos valores que estiverem sendo inseridos:

```
notas = pd.Series([5,23,33,14,11], index=['a', 'b', 'c', 'd', 'e'])
```



Series



Os valores de uma Series podem ser acessados através da referenciação de seu rótulo. Acessando a nota **'b'** teremos o retorno do valor **23**.

```
notas = pd.Series([5,23,33,14,11], index=['a', 'b', 'c', 'd', 'e'])
notas['b']
```



Series



Casos os valores dentro da Series sejam numéricos, é possível acessar a média e o desvio padrão de forma simples.

Média:

notas.mean()

Desvio padrão:

notas.std()



DataFrame



Os DataFrame são estruturas bidimensionais e podem comportar diferentes tipos de atributo, formando uma estrutura com múltiplas informações.



DataFrame



A estrutura de um DataFrame é semelhante a uma planilha:

	Nome	Idade	Tamanho
0	Ana	23	36
1	Camila	35	44
2	Saulo	42	50
3	Raphael	21	48
4	David	25	40



DataFrame



É possível verificar todas as informações sobre a estrutura de um objeto do tipo DataFrame:

clientes.dtypes



DataFrame



Também podem ser verificadas apenas as colunas presentes no objeto DataFrame e acessados todos os valores de uma coluna, respectivamente.

```
clientes.columns
clientes["Nomes"]
```



DataFrame



É possível ordenar os dados de um DataFrame por alguma coluna específica. A seguir é apresentada a ordenação do objeto pelo nome e depois pela idade.

```
clientes.sort_values(by="Nomes")
clientes.sort_values(by="Idade")
```



DataFrame



No caso dos DataFrame, se o objetivo for acessar um valor a partir do index é necessário utilizar uma função específica, chamada *loc*.

clientes.loc[2]



DataFrame



Em um objeto DataFrame é possível fazer uma filtragem direta nos dados retornados, informando uma determinada condição. No exemplo a seguir é realizado um filtro nos dados para retornar apenas os clientes que tenha idade inferior a 40 anos.

clientes[clientes["Idade"]<40]</pre>



Leitura de dados



A biblioteca dispõe de algumas funcionalidades para a leitura de arquivos, uma funcionalidade muito importante para quem vai trabalhar com dados que serão importados de dentro de arquivos. A seguir são apresentadas as funções para leitura de arquivos, respectivamente, do tipo csv, xlsx e html.

- pd.read_csv
- pd.read_xlsx
- pd.read_html



Selecionar registros não NaN



Em diversas situações pode ser necessário selecionar os registros de um DataFrame que não contenham atributos nulos.

clientes.dropna()



Preenchendo atributos NaN



A biblioteca oferece um recurso importante para preencher todos os registros NaN com um determinado valor. Isso pode ser feito de forma simples através da função *fillna*.

```
clientes.fillna()
```



Verificando valores NaN



Em muitas situações é interessante verificar se no Dataframe existem valores NaN ou não. O comando a seguir irá retornar True ou False para cada um dos atributos dos elementos do DataFrame.

clientes.isna()







Obrigada!

Ana Laurentino

