



Curso de Python

28-01-2021

<https://github.com/tsbressan/CursoPythonUnisinos>

Importação de arquivos

- **Datasets disponíveis**

Muitos sites fornecem datasets livres para utilização:

- <https://www.kaggle.com/datasets>
- <http://web.iodp.tamu.edu/LORE/> <- <- <- <- <-
- <http://dados.gov.br/>
- <http://www.portaldatransparencia.gov.br/>
- <https://www.usgs.gov/products/data-and-tools/science-datasets>

Bibliotecas para processamento

- **Baixar material da IODP**
- <http://web.iodp.tamu.edu/LORE/>
- **Vamos utilizar a propriedade GRA, Expedition 362, site U1480**
- **Arquivo: *.csv**
- **Separador: ,**

Bibliotecas para processamento

- **Dataset – leitura IODP**

Resolução – Intervalo de leitura



Bibliotecas para processamento

- **Baixar material da IODP**
- **Importante:**
- Antes de importar, ajustar nome das colunas. Eliminar espaços ou caracteres especiais.

Bibliotecas para processamento

- **Pandas:** <https://pandas.pydata.org/>
- Biblioteca muito utilizada para análise, manipulação e formação de dados
- Necessita ser instalado no Python (pode ser instalado no ambiente selecionado): pip install pandas ou conda install pandas ou pelo gerenciador: Anaconda Navigator
- Necessita ser importada antes da utilização:
- `import pandas as pd`

Bibliotecas para processamento

- **Pandas: consultar instalação**

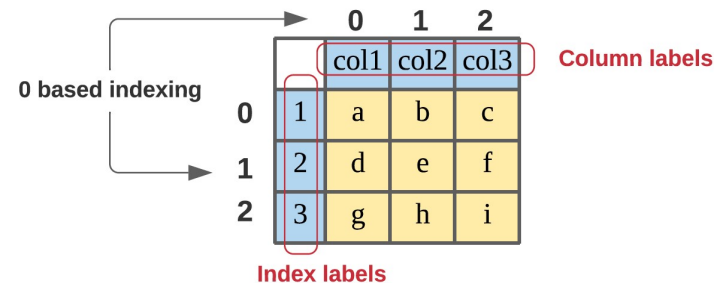
```
import pandas as pd
```

```
pd.__version__
```

Bibliotecas para processamento

- **Pandas: dataframe**

Forma uma estrutura de dados bidimensional, suporte a dados heterogêneos, com eixo rotulados (linhas e colunas). Os dados são tabulados conforme organização entre linhas e colunas formando uma tabela de dados.



	Name	Team	Number	Position	Age
0	Avery Bradley	Boston Celtics	0.0	PG	25.0
1	John Holland	Boston Celtics	30.0	SG	27.0
2	Jonas Jerebko	Boston Celtics	8.0	PF	29.0
3	Jordan Mickey	Boston Celtics	NaN	PF	21.0
4	Terry Rozier	Boston Celtics	12.0	PG	22.0
5	Jared Sullinger	Boston Celtics	7.0	C	NaN
6	Evan Turner	Boston Celtics	11.0	SG	27.0

<https://media.geeksforgeeks.org/wp-content/uploads/finallpandas.png>



Bibliotecas para processamento

- **Pandas: dataframe**

Por exemplo:

```
import pandas as pd
```

```
#lista de cidades
```


```
idades = ['Porto Alegre', 'Curitiba', 'Fortaleza', 'Maceio',  
          'Santiago', 'Brasília', 'São Paulo']
```

```
#DataFrame constructor
```

```
df = pd.DataFrame(cidades)
```

```
print(df)
```

Saída:



	0
0	Porto Alegre
1	Curitiba
2	Fortaleza
3	Maceio
4	Santiago
5	Brasília
6	São Paulo

Bibliotecas para processamento

- **Pandas: dataframe**

Por exemplo:

```
import pandas as pd
```

```
#lista de cidades
```


```
idades = {"Cidade": ['Porto Alegre', 'Curitiba', 'Fortaleza', 'Maceio',  
                    'Santiago', 'Brasília', 'São Paulo'], "Estado": ['RS','PR','CE','AL','RS','DF','SP']}
```

```
#DataFrame constructor
```

```
df = pd.DataFrame(cidades)
```

```
print(df)
```

Saída:



	Cidade	Estado
0	Porto Alegre	RS
1	Curitiba	PR
2	Fortaleza	CE
3	Maceio	AL
4	Santiago	RS
5	Brasília	DF
6	São Paulo	SP

Bibliotecas para processamento

- **Pandas: dataframe**

Importante sempre consulta a documentação:

<https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.html>

```
class pandas.DataFrame(data=None, index=None, columns=None,
dtype=None, copy=False)
```

```
>>> d = {'col1': [1, 2], 'col2': [3, 4]}
>>> df = pd.DataFrame(data=d)
>>> df
   col1  col2
0     1     3
1     2     4
```

Bibliotecas para processamento

- **Pandas: dataframe**

```
>>>type(df)
```

```
-> pandas.core.frame.DataFrame
```

```
>>>df.dtypes
```

```
Cidade    object
```

```
Estado    object
```

```
dtype: object
```

Bibliotecas para processamento

- **Pandas: dataframe**

```
>>>df["Cidade"]
```

```
0    Porto Alegre
```

```
1      Curitiba
```

```
2    Fortaleza
```

```
3      Maceio
```

```
4    Santiago
```

```
5    Brasília
```

```
6    São Paulo
```

```
Name: Cidade, dtype: object
```

Conversão de tipo:

```
df["A"] =pd.to_numeric(df["A"])
```

Bibliotecas para processamento

- **Pandas: dataframe**

>>>df.head() #retorna um número n de linhas (resumo do dataset).
Padrão: 5

Out[18]:

	Cidade	Estado
0	Porto Alegre	RS
1	Curitiba	PR
2	Fortaleza	CE
3	Maceio	AL
4	Santiago	RS

>>> df #listar todo o dataset – cuidado
 #com o tamanho!!!

Out[21]:

	Cidade	Estado
0	Porto Alegre	RS
1	Curitiba	PR
2	Fortaleza	CE
3	Maceio	AL
4	Santiago	RS
5	Brasília	DF
6	São Paulo	SP

Bibliotecas para processamento

- **Pandas: dataframe**

```
>>> df.rename(columns = {'Cidade': 'Cid'})
```

```
>>> df.drop(['Cidade'], axis=1)
```

```
>>> df.drop([0], axis=0)
```

```
>> df.iloc[0:2] #selecionar intervalo de linhas no dataset
```

```
>> df.sample(frac=0.5) #acesso aleatório a uma quantidade de dados
```

Importação de arquivos .xlsx e .csv

- Criando o dataset de dados com o Pandas DataFrame
- **Dataset a partir de arquivo *.xlsx**

```
import pandas as pd
```

```
#nome do arquivo
```

```
file = "... "
```

```
df = pd.read_excel(file, sheet_name = '...'))
```


Importação de arquivos .xlsx e .csv

- Criando o dataset de dados com o Pandas DataFrame
- **Dataset a partir de arquivo *.csv com delimitador: ,:**

```
import pandas as pd
```

```
#nome do arquivo
```

```
file = "..."
```

```
#ler arquivos .csv com delimitador informado, neste caso a ,
```

```
df = pd.read_csv(file,sep=",")
```

Importação de arquivos .xlsx e .csv

- Atividades sobre os dados importados:

- `df.describe()`

Resumo da estatística com média, maior, menor,...

- `df["<column>"].max()`
- `df["<column>"].min()`
- `df["<column>"].mean()`
- `df.count()`
- `df.value_counts()`

Importação de arquivos .xlsx e .csv

- Atividades sobre os dados importados:

- `df.sort_values(by= "<column>", ascending=...)`

Ordenar os dados por ordem crescente ou decrescente.

Seleção/Consulta :

- `df["<column>"]` #column
- `df[...:..]` #line por exemplo: `df[0:2]` # usa o index

Importação de arquivos .xlsx e .csv

- Atividades sobre os dados importados:

Seleção/Consulta :

- `df[df["<column>"] > 0]`
- Consulta por coluna. Retorna dataframe completo pelo parâmetro da consulta.
- `df.filter(["Exp", "Site"])`
- Mostar na tela apenas as colunas selecionadas.

Importação de arquivos .xlsx e .csv

- Filtro nos dados:

#dados em branco ou ausentes - consultar usando os operadores

Logic in Python (and pandas)			
<	Less than	!=	Not equal to
>	Greater than	df.column.isin(values)	Group membership
==	Equals	pd.isnull(obj)	Is NaN
<=	Less than or equals	pd.notnull(obj)	Is not NaN
>=	Greater than or equals	&, , ~, ^, df.any(), df.all()	Logical and, or, not, xor, any, all

Importação de arquivos .xlsx e .csv

- Filtro nos dados:

#dados em branco ou ausentes

None ou NaN (Not a Number) # numpy nan (np.nan)

0 ou “ ” ou ‘ ’ não são dados em branco.

Importação de arquivos .xlsx e .csv

- Filtro nos dados:

#dados outliers

Valores fora do padrão, dados com ruídos ou desproporcionais

- Visualizado através de um box plot, ou um gráfico de pontos simples
- Ajustar esses valores com uma função específica, por exemplo, `scipy.stats.zscore`)
- Uma forma simples é comparar os valores com um intervalo conhecido.

```
dataset[(dataset["valor"] > 1) & (dataset["valor"] < 2)]
```


Bibliotecas para processamento

- **Pandas: dataframe**

Acessar o conteúdo do dataframe via estrutura de repetição (for):

...

```
for line in range(0,len(dataset)):
    print (dataset["..."][line])
```

...

Bibliotecas para processamento

- **Criar arquivos .csv**
- Utiliza a biblioteca pandas

```
import pandas as pd
```

```
df = pd.DataFrame({'nome': ['João', 'Carlos'],  
                  'Cor': ['red', 'Blue'],  
                  'Filmes': ['Fast and Furious', 'Bruxa de Blair']})
```

```
df.to_csv("teste1.csv")
```