



Aula 4
Introdução ao
Pandas - Parte I

Introdução



- Segundo a própria descrição em seu *site* (<https://pandas.pydata.org/>): *"Pandas ferramenta de análise de dados rápida, poderosa, flexível, open source, e construída sobre a linguagem Python"*
- Dado que o Pandas é construído na linguagem Python e faz uso não só dos tipos básicos quando da biblioteca Numpy, a troca de informações é natural uma vez que se sabe tais conceitos
- Popularmente conhecida como *"A biblioteca do cientista de dados"*

Introdução

- Principais características:
 - Estruturas de dados rápidas (construídas em Cython ou C) e indexadas para a manipulação de dados
 - Leitura e gravação em diferentes fontes: memória, rede, arquivos
 - Leitura e gravação em diferentes formatos: CSV, Excel, Bancos de Dados Relacionais, JSON, dentre muitos outros
 - Facilidades para seleções, junções e sumarização dos dados

Introdução

- Principais características:
 - Facilidades para obtenção de subconjuntos dos dados (ex: *slicing*, amostragem, etc.)
 - Utilizado academicamente e comercialmente (ex: finanças, neurosciência, economia, estatística, *web analytics*, etc.)
 - Integração natural com outras bibliotecas de aprendizado de máquina

Introdução

- Para utilizar o pandas, basta importá-lo com a instrução:
`import pandas`
- É comum o uso do alias `pd` para se referir ao pandas



```
import pandas as pd
```

Estruturas Básicas

- O pandas possui duas estruturas básicas para a manipulação de dados:
 - Series: para dados unidimensionais
 - DataFrame: para dados multidimensionais
- Ambas as estruturas são indexadas, i.e., é possível acessar um dado ou uma tupla por meio dos índices

Series

- Para se criar um Series, o comando completo é:
`pd.Series(data=None, index=None, dtype=None, name=None, copy=False, fastpath=False)`
 - **data**: array-like, iterável, dicionário ou valor escalar
 - **index**: array-like
 - **dtype**: tipo de dados
 - **name**: nome para a série
 - **copy**: copia os dados de entrada (bool)

Series

Criando uma Series utilizando todos os argumentos

```
[4] #Criando uma série com todas as informações do método construtor  
series1 = pd.Series(data=['Rafael','Ricardo','Ronaldo'], index=['000.000.000-11', '010.030.040.80', '858.987.98-10'], dtype=str, name='Cadastro', copy=False)
```

```
[5] series1
```

```
000.000.000-11    Rafael  
010.030.040.80    Ricardo  
858.987.98-10     Ronaldo  
Name: Cadastro, dtype: object
```

```
[7] # Acessando o primeiro elemento da series  
series1[0]
```

```
'Rafael'
```

```
[8] #Uma series recuperar conjuntos de dados utilizando a operação de slice  
series1[1:3]
```

```
010.030.040.80    Ricardo  
858.987.98-10     Ronaldo  
Name: Cadastro, dtype: object
```

```
9 #Utilizando o índice para acessar os objetos  
series1.loc['000.000.000-11']
```

```
'Rafael'
```


Series

Criando uma Series apenas com uma lista

```
[11] #Criando uma series apenas com os dados fornecidos por uma lista (índices são criados automaticamente)
      series2 = pd.Series(data=['Rafael','Ricardo','Ronaldo'])
```

▶ series2

```
0    Rafael
1    Ricardo
2    Ronaldo
dtype: object
```

Criando uma Series com um dicionário

```
[16] #Criando uma series com dicionário (chaves são os índices)
      series3 = pd.Series({'000.000.000-11':'Rafael', '010.030.040.80':'Ricardo', '858.987.98-10':'Ronaldo'})
```

▶ series3

```
000.000.000-11    Rafael
010.030.040.80    Ronaldo
858.987.98-10     Ronaldo
dtype: object
```

Series

Uma Series é no mínimo tão boa quanto a combinação de um dicionário e um *array* Numpy

```
[27] #Uma series combina o poder de um dicionário com o poder dos arrays Numpy  
      series4 = pd.Series(np.arange(10))
```

```
▶ series4.max()
```

```
↪ 9
```

DataFrame

- O construtor de um DataFrame `pd.DataFrame(data=None, index=None, columns=None, dtype=None, copy=False)`
 - **data**: array-like, iterável, dicionário, ou um outro DataFrame
 - **index**: array-like
 - **columns**: array-like
 - **dtype**: tipo de dados
 - **copy**: copia os dados de entrada (bool)

DataFrame

Criando DataFrames com *arrays*

```
[43] #Criando um DataFrame utilizando todos os campos do construtor e indexando pelo CPF
      dados = np.array([[ 'Rafael', '000.000.000-00', 'SI'],
                        [ 'Ricardo', '000.000.123-89', 'Crochê'],
                        [ 'Ricardo', '000.000.456-12', 'SI' ]
                        ])

      df1 = pd.DataFrame(data=dados, index=dados[:,1], columns=[ 'Nome', 'CPF', 'Curso'], copy=False)
```

[44] df1

	Nome	CPF	Curso
000.000.000-00	Rafael	000.000.000-00	SI
000.000.123-89	Ricardo	000.000.123-89	Crochê
000.000.456-12	Ricardo	000.000.456-12	SI

```
[45] #Criando um DataFrame apenas informando os dados
      df2 = pd.DataFrame(dados)
```

df2

	0	1	2
0	Rafael	000.000.000-00	SI
1	Ricardo	000.000.123-89	Crochê
2	Ricardo	000.000.456-12	SI

DataFrame

Criando DataFrames um dicionário

```
[47] #Criando um DataFrame com um dicionário (as chaves correspondem às colunas)
      dados_dict = {'Nome': ['Rafael', 'Ricardo', 'Ronaldo'],
                    'CPF': ['000.000.000-11', '010.030.040.80', '858.987.98-10'],
                    'Curso': ['SI', 'Crochê', 'SI']}
      df3 = pd.DataFrame(dados_dict)
```

▶ df3

	Nome	CPF	Curso
0	Rafael	000.000.000-11	SI
1	Ricardo	010.030.040.80	Crochê
2	Ronaldo	858.987.98-10	SI

Propriedades Básicas das Estruturas

- O pandas possui um série de atributos que armazenam informações do DataFrame quanto as suas dimensões, índices e colunas

```
[197] df1
```

	Nome	CPF	Curso
000.000.000-00	Rafael	000.000.000-00	SI
000.000.123-89	Ricardo	000.000.123-89	Crochê
000.000.456-12	Ricardo	000.000.456-12	SI

```
[199] # Shape de um DataFrame (número de linhas x número de colunas)  
df1.shape
```

```
(3, 3)
```

[+ Código](#)[+ Texto](#)

```
[201] # Número de tuplas (linhas) de um DataFrame  
len(df1)
```

```
3
```

```
[203] # Retornando as colunas de um DataFrame  
df1.columns
```

```
Index(['Nome', 'CPF', 'Curso'], dtype='object')
```

```
df1.index
```

```
Index(['000.000.000-00', '000.000.123-89', '000.000.456-12'], dtype='object')
```

Diferentes Formatos de Entrada e Saída

- Até o momento vimos como criar manualmente as estruturas do Pandas: Series e DataFrames
- Entretanto, muitas vezes os dados que vamos manipular encontram-se em arquivos de **diferentes formatos**, como CSV, JSON, planilhas do Excel, páginas web, e Banco de Dados
- O pandas possui funcionalidades para ler dados nos formatos citados acima (dentre outros)
- Os dados a serem lidos podem estar **armazenados localmente** ou **remotamente**
- Também há funcionalidade para **salvar** os arquivos em **diferentes formatos**

Lendo um CSV

- O pandas possui o método `read_csv()` para fazer a leitura de arquivos em tal formato
- Deve-se obrigatoriamente especificar o caminho do arquivo CSV
- Pode-se especificar uma série de outros parâmetros, como o caractere separador, definir uma coluna para ser utilizada com índice, especificar quais colunas do arquivo devem ser carregadas, etc.

Lendo um CSV



..
sample_data
carros.csv

```
[67] df4 = pd.read_csv('carros.csv', sep=';')
```

df4


	Nome	Motor	Ano	Quilometragem	Zero_km	Acessórios	Valor
0	Jetta Variant	Motor 4.0 Turbo	2003	44410.0	False	['Rodas de liga', 'Travas elétricas', 'Piloto ...	88078.64
1	Passat	Motor Diesel	1991	5712.0	False	['Central multimídia', 'Teto panorâmico', 'Fre...	106161.94
2	Crossfox	Motor Diesel V8	1990	37123.0	False	['Piloto automático', 'Controle de estabilidade...	72832.16
3	DS5	Motor 2.4 Turbo	2019	NaN	True	['Travas elétricas', '4 X 4', 'Vidros elétrico...	124549.07
4	Aston Martin DB4	Motor 2.4 Turbo	2006	25757.0	False	['Rodas de liga', '4 X 4', 'Central multimídia...	92612.10
...
253	Phantom 2013	Motor V8	2014	27505.0	False	['Controle de estabilidade', 'Piloto automátic...	51759.58
254	Cadillac Ciel concept	Motor V8	1991	29981.0	False	['Bancos de couro', 'Painel digital', 'Sensor ...	51667.06
255	Classe GLK	Motor 5.0 V8 Bi-Turbo	2002	52637.0	False	['Rodas de liga', 'Controle de tração', 'Câmbi...	68934.03
256	Aston Martin DB5	Motor Diesel	1996	7685.0	False	['Ar condicionado', '4 X 4', 'Câmbio automátic...	122110.90
257	Macan	Motor Diesel V6	1992	50188.0	False	['Central multimídia', 'Teto panorâmico', 'Vid...	90381.47

258 rows x 7 columns

Lendo um JSON

- O Pandas também possui um método para ler um arquivo em formato JSON e gerar um DataFrame: `read_json()`
- Em seu uso mais básico, pode-se apenas passar o caminho do arquivo como argumento do método

Exemplo da leitura de um arquivo JSON local



The screenshot displays a Jupyter Notebook interface. On the left, a file explorer shows a directory with files: `sample_data`, `carros.csv`, and `matriculas.json`. The main notebook area shows the execution of the following code:

```
[69] df5 = pd.read_json('matriculas.json')  
  
[70] df5
```

The output of the code is a DataFrame with the following structure:

	Nome	CPF	Curso
0	Rafael	000.012.212-10	Sistemas de Informação
1	Ricardo	789.123.987-50	Corte e Costura
2	Ronaldo	555.444.333-22	Geografia

To the right of the notebook, the content of the `matriculas.json` file is shown. It is a JSON array containing three objects, each representing a person with their name, CPF, and course.

```
1 {  
2   {  
3     "Nome": "Rafael",  
4     "CPF": "000.012.212-10",  
5     "Curso": "Sistemas de Informação"  
6   },  
7   {  
8     "Nome": "Ricardo",  
9     "CPF": "789.123.987-50",  
10    "Curso": "Corte e Costura"  
11  },  
12  {  
13    "Nome": "Ronaldo",  
14    "CPF": "555.444.333-22",  
15    "Curso": "Geografia"  
16  }  
17 }
```

Lendo um JSON

Exemplo da leitura de um arquivo JSON remoto

```
[73] df7 = pd.read_json('https://servicodados.ibge.gov.br/api/v1/censos/nomes/ranking?qtd=200')
```

df7

	nome	regiao	freq	rank	sexo
0	MARIA	0	11734129	1	
1	JOSE	0	5754529	2	
2	ANA	0	3089858	3	
3	JOAO	0	2984119	4	
4	ANTONIO	0	2576348	5	
...
195	FABIANO	0	159150	196	
196	MILENA	0	159042	197	
197	WESLEY	0	157205	198	
198	DIOGO	0	156119	199	
199	ADILSON	0	155430	200	

200 rows × 5 columns

Lendo Dados Armazenados em Tabelas HTML

- O Pandas consegue fazer a leitura de tabelas em páginas HTML e convertê-las para um DataFrame
- Para isso, em sua forma mais básica, basta utilizar o método `read_html()` e informar o caminho da página
- A diferença em relação aos métodos anteriores, é que como uma página pode possuir mais de uma tabela, o método retorna uma lista de DataFrames, e cada elemento da lista corresponde a um DataFrame de uma tabela da página
- O Pandas trata automaticamente os *spans*, repetindo automaticamente os valores em linhas ou colunas expandidas

Lendo Dados Armazenados em Tabelas HTML

```
[27] df8 = pd.read_html('https://www.ufms.br/cursos/graduacao/')
```

```
[28] type(df8)
```

```
list
```

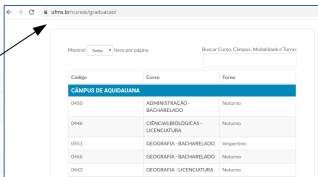
```
[29] len(df8)
```

```
1
```

```
df8[0]
```

	Unidade	Código	Curso	Turno
0	CÂMPUS DE AQUIDAUANA	450	ADMINISTRAÇÃO - BACHARELADO	Noturno
1	CÂMPUS DE AQUIDAUANA	446	CIÊNCIAS BIOLÓGICAS - LICENCIATURA	Noturno
2	CÂMPUS DE AQUIDAUANA	451	GEOGRAFIA - BACHARELADO	Vespertino
3	CÂMPUS DE AQUIDAUANA	466	GEOGRAFIA - BACHARELADO	Noturno
4	CÂMPUS DE AQUIDAUANA	443	GEOGRAFIA - LICENCIATURA	Noturno
...
144	INSTITUTO DE QUÍMICA	2304	ENGENHARIA QUÍMICA - BACHARELADO	Integral
145	INSTITUTO DE QUÍMICA	2302	QUÍMICA - BACHARELADO EM QUÍMICA TECNOLÓGICA	Integral (Matutino e Vespertino)
146	INSTITUTO DE QUÍMICA	2301	QUÍMICA - LICENCIATURA	Noturno
147	INSTITUTO INTEGRADO DE SAÚDE	2801	ENFERMAGEM - BACHARELADO	Integral (Matutino e Vespertino)
148	INSTITUTO INTEGRADO DE SAÚDE	2802	FISIOTERAPIA - BACHARELADO	Integral (Matutino e Vespertino)

149 rows x 4 columns



Código	Curso	Turno
CÂMPUS DE AQUIDAUANA		
0450	ADMINISTRAÇÃO - BACHARELADO	Noturno
0446	CIÊNCIAS BIOLÓGICAS - LICENCIATURA	Noturno
0451	GEOGRAFIA - BACHARELADO	Vespertino
0466	GEOGRAFIA - BACHARELADO	Noturno
0443	GEOGRAFIA - LICENCIATURA	Noturno

Lendo Dados de Planilhas Excel

- O Pandas também possui uma função para ler dados em planilha no formato aceito pelo Microsoft Excel que é a `read_csv()`
- Por padrão, apenas a primeira folha planilha é retornada em um `DataFrame`
- Entretanto, é possível especificar um conjunto de planilhas a serem retornadas por meio da especificação do índice ou do nome das planilhas

Lendo Dados de Planilhas Excel

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S
1	RGa	Q1 (0,5)	Q2 (0,5)	Q3 (0,5)	Q4 (0,5)	Q5 (0,5)	Q6 (0,5)	Q7 (0,5)	Q8 (0,5)	Q9 (0,5)	Q10 (0,5)	Q11 (0,5)	Q12 (0,5)	Q13 (0,5)	Q14 (2,5)	Q15 (1,0)	Q16 (1,0)	Q17 (1,0)	Total
2	2018.0743.001-0	0,50	0,50	0,50	0,50	0,50	0,50	0,50	0,50	0,50	0,50	0,50	0,50	0,50	2,25	0,50	0,50	0,50	6,00
3	2018.0743.003-8	0,50	0,50	0,50	0,50	0,50	0,50	0,50	0,50	0,50	0,50	0,50	0,50	0,50	2,50	1,00	0,50	0,50	8,50
4	2018.0743.006-7	0,50	0,50	0,50	0,50	0,50	0,50	0,50	0,50	0,50	0,50	0,50	0,50	0,50	2,50	1,00	0,50	0,50	8,50
5	2018.0743.034-0	0,50	0,50	0,50	0,50	0,50	0,50	0,50	0,50	0,50	0,50	0,50	0,50	0,50	2,50	1,00	0,50	0,50	8,50
6	2018.0743.036-7	0,50	0,50	0,50	0,50	0,50	0,50	0,50	0,50	0,50	0,50	0,50	0,50	0,50	2,50	1,00	0,50	0,50	8,50
7	2018.0743.051-0	0,50	0,50	0,50	0,50	0,50	0,50	0,50	0,50	0,50	0,50	0,50	0,50	0,50	2,50	1,00	0,50	0,50	8,50
8	2018.0743.051-0	0,50	0,50	0,50	0,50	0,50	0,50	0,50	0,50	0,50	0,50	0,50	0,50	0,50	2,50	1,00	0,50	0,50	8,50
9	2018.0743.054-7	0,50	0,50	0,50	0,50	0,50	0,50	0,50	0,50	0,50	0,50	0,50	0,50	0,50	2,50	1,00	0,50	0,50	8,50
10	2018.0743.054-7	0,50	0,50	0,50	0,50	0,50	0,50	0,50	0,50	0,50	0,50	0,50	0,50	0,50	2,50	1,00	0,50	0,50	8,50
11	Média	0,50	0,48	0,48	0,48	0,48	0,48	0,48	0,48	0,48	0,48	0,48	0,48	0,48	0,48	0,48	0,48	0,48	0,48
12																			
13																			
14																			

```
[34] df9 = pd.read_excel('notas_IA.xlsx',)
```

```
[35] df9
```

	RGa	Q1 (0,5)	Q2 (0,5)	Q3 (0,5)	Q4 (0,5)	Q5 (0,5)	Q6 (0,5)	Q7 (0,5)	Q8 (0,5)	Q9 (0,5)	Q10 (0,5)	Q11 (0,5)	Q12 (0,5)	Q13 (0,5)	Q14 (2,5)	Q15 (1,0)	Q16 (1,0)	Q17 (1,0)	Total
0	2018.0743.001-0	0.5	0.500000	0.000000	0.000000	0.0	0.000000	0.500000	0.5	0.000000	0.000000	0.500000	0.500000	0.000000	2.250000	0.800000	0.0	0.000000	6.000000
1	2018.0743.003-8	0.5	0.500000	0.500000	0.500000	0.0	0.500000	0.500000	0.5	0.000000	0.000000	0.500000	0.500000	0.000000	2.500000	1.000000	0.0	0.000000	8.000000
2	2018.0743.006-7	0.5	0.500000	0.500000	0.000000	0.0	0.500000	0.000000	0.5	0.500000	0.500000	0.500000	0.500000	0.500000	2.500000	1.000000	1.0	0.000000	9.500000
3	2018.0743.034-0	0.5	0.000000	0.500000	0.500000	0.0	0.500000	0.500000	0.5	0.000000	0.000000	0.500000	0.500000	0.500000	2.500000	1.000000	0.5	0.000000	8.500000
4	2018.0743.036-7	0.5	0.500000	0.500000	0.500000	0.0	0.000000	0.500000	0.5	0.000000	0.500000	0.500000	0.500000	0.500000	2.000000	0.750000	0.3	0.000000	8.050000
5	2018.0743.051-0	0.5	0.500000	0.500000	0.500000	0.0	0.500000	0.500000	0.5	0.000000	0.000000	0.000000	0.000000	0.000000	2.500000	1.000000	0.0	0.000000	7.500000
6	2018.0743.051-0	0.5	0.500000	0.500000	0.500000	0.0	0.500000	0.500000	0.5	0.000000	0.000000	0.500000	0.500000	0.500000	2.500000	1.000000	1.0	1.000000	10.500000
7	2019.0743.054-7	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
8	2018.0743.019-7	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
9	Média	0.5	0.428571	0.428571	0.357143	0.0	0.257143	0.428571	0.5	0.071429	0.214286	0.428571	0.428571	0.285714	2.385714	0.928571	0.4	0.142857	8.292857

Lendo Dados de Planilhas Excel

- Ao especificar mais de uma folha da planilha a ser retornada, o formato de retorno é um dict, em que o índice ou nome da planilha será a chave do dicionário

```
[39] df10 = pd.read_excel('notas_IA.xlsx', sheet_name=[0,1])
```

```
[41] type(df10)
```

```
dict
```

```
[42] df10.keys()
```

```
dict_keys([0, 1])
```

```
df10[0]
```

	RGA	Q1 (0,5)	Q2 (0,5)	Q3 (0,5)	Q4 (0,5)	Q5 (0,5)	Q6 (0,5)	Q7 (0,5)	Q8 (0,5)	Q9 (0,5)	Q10 (0,5)	Q11 (0,5)	Q12 (0,5)	Q13 (0,5)	Q14 (2,5)	Q15 (1,0)	Q16 (1,0)	Q17 (1,0)	Total
0	2016.0743.001-0	0.5	0.500000	0.000000	0.000000	0.0	0.000000	0.500000	0.5	0.000000	0.000000	0.500000	0.500000	0.000000	2.200000	0.800000	0.0	0.000000	6.000000
1	2018.0743.061-8	0.5	0.500000	0.500000	0.500000	0.0	0.500000	0.500000	0.5	0.000000	0.000000	0.500000	0.500000	0.000000	2.500000	1.000000	0.0	0.000000	8.000000
2	2018.0743.036-7	0.5	0.500000	0.500000	0.000000	0.0	0.500000	0.000000	0.5	0.500000	0.500000	0.500000	0.500000	0.500000	2.500000	1.000000	1.0	0.000000	9.500000
3	2018.0743.034-0	0.5	0.000000	0.500000	0.500000	0.0	0.500000	0.500000	0.5	0.000000	0.000000	0.500000	0.500000	0.500000	2.500000	1.000000	0.5	0.000000	8.500000
4	2018.0743.006-7	0.5	0.500000	0.500000	0.500000	0.0	0.000000	0.500000	0.5	0.000000	0.500000	0.500000	0.500000	0.500000	2.000000	0.750000	0.3	0.000000	8.050000
5	2018.0743.051-0	0.5	0.500000	0.500000	0.500000	0.0	0.500000	0.500000	0.5	0.000000	0.500000	0.000000	0.000000	0.000000	2.500000	1.000000	0.0	0.000000	7.500000
6	2018.0743.020-0	0.5	0.500000	0.500000	0.500000	0.0	0.500000	0.500000	0.5	0.000000	0.000000	0.500000	0.500000	0.500000	2.500000	1.000000	1.0	0.000000	10.500000
7	2019.0743.054-7	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
8	2018.0743.019-7	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
9	Meda	0.5	0.428571	0.428571	0.357143	0.0	0.357143	0.428571	0.5	0.071429	0.214286	0.428571	0.428571	0.285714	2.385714	0.935714	0.4	0.142857	8.292857

Lendo Dados de um Banco de Dados

- O Pandas permite ler e gravar em um banco de dados
- Nativamente o pandas suporta o SGBD SQLite3, porém, pode-se facilmente fazer a leitura de outras bases de dados
- Nesta aula utilizaremos o MySQL

Lendo Dados de um Banco de Dados

- Para habilitar a leitura de um banco de dados MySQL, será necessária a instalação do:
 - sqlalchemy: para criar uma engine de conexão com o banco de dados através da função `create_engine`
 - pymysql: para estabelecer a conexão com o banco de dados
- A sintaxe da string de conexão com o banco de dados utilizando o pymysql é dada por:
`'mysql+pymysql://mysql_user:mysql_password@mysql_host/mysql_db'`
- Para fazer uma consulta, basta utilizar o método `read_sql()` e informar a consulta e um objeto da conexão

Lendo Dados de um Bando de Dados

```
[2] In [ ]:
from sqlalchemy import create_engine
import pymysql
import pandas as pd

[10] In [ ]:
#Criando a engine de conexão com a string de conexão
sqlEngine = create_engine('mysql+pymysql://root:root@localhost/progweb2')

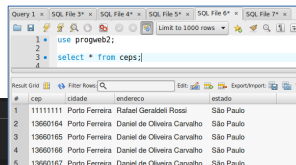
#Criando a conexão com o banco
dbConnection = sqlEngine.connect()

#Fazendo uma consulta passando a string de consulta e a conexão com o banco
dados = pd.read_sql("select * from ceps", dbConnection);

#Fechando a conexão com o banco
dbConnection.close()

dados
```

	cep	cidade	endereço	estado
0	11111111	Porto Ferreira	Rafael Geraldelli Rossi	São Paulo
1	13660164	Porto Ferreira	Daniel de Oliveira Carvalho	São Paulo
2	13660165	Porto Ferreira	Daniel de Oliveira Carvalho	São Paulo
3	13660166	Porto Ferreira	Daniel de Oliveira Carvalho	São Paulo
4	13660167	Porto Ferreira	Daniel de Oliveira Carvalho	São Paulo



The screenshot shows a SQL query editor with a query window and a results grid. The query is: `use progweb2; select * from ceps;`. The results grid displays 5 rows of data with columns: cep, cidade, endereço, and estado.

#	cep	cidade	endereço	estado
1	11111111	Porto Ferreira	Rafael Geraldelli Rossi	São Paulo
2	13660164	Porto Ferreira	Daniel de Oliveira Carvalho	São Paulo
3	13660165	Porto Ferreira	Daniel de Oliveira Carvalho	São Paulo
4	13660166	Porto Ferreira	Daniel de Oliveira Carvalho	São Paulo
5	13660167	Porto Ferreira	Daniel de Oliveira Carvalho	São Paulo

- O pandas possui métodos para converter ou gravar os DataFrames nos formatos apresentados anteriormente e em muitos outros formatos
- Basicamente há um método “to” para cada método “read”, além de métodos para gerar outros tipos de formatos
- Vale ressaltar que ao gravar um arquivo, o índice do DataFrame, por padrão, é gravado junto, o que pode causar resultados indesejados
- Para isso, é comum utilizar o parâmetro `include_index = False` quando for gravar um arquivo

Gravação

Exemplo de gravação de um DataFrame em um arquivo CSV

Arquivos

-
- sample_data
- cadastro.csv
- carros.csv
- matriculas.json
- matriculas_zuadas.json
- notas_xlsx

[133] df10

Nota
0 10
1 4
2 6

[134] df1

	Nome	CPF	Curso	Idade	Cidade	Ano_Nascimento
000.000.000-00	rafael	000.000.000-00	SI	76	Porto Ferreira	1944
000.000.123-89	ricardo	000.000.123-89	Crochê	4	Jales	2016
000.000.456-12	ronaldo	000.000.456-12	SI	97	Rindpolis	1923

[136] df1.to_csv('cadastro.csv', index=False)

cadastro.csv

De 1 a 3 de 3 entradas

Nome	CPF	Curso	Idade	Cidade	Ano_Nascimento
rafael	000.000.000-00	SI	76	Porto Ferreira	1944
ricardo	000.000.123-89	Crochê	4	Jales	2016
ronaldo	000.000.456-12	SI	97	Rindpolis	1923

Mostrar 10 por página

Gravação

Convertendo um DataFrame para uma tabela Latex

[140] df1

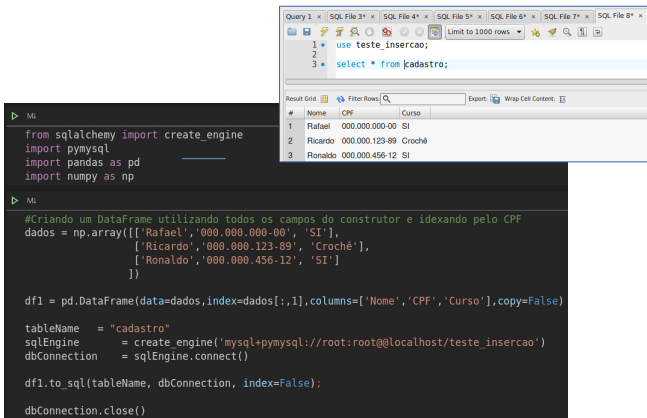
	Nome	CPF	Curso	Idade	Cidade	Ano_Nascimento
000.000.000-00	rafael	000.000.000-00	SI	76	Porto Ferreira	1944
000.000.123-89	ricardo	000.000.123-89	Crochê	4	Jales	2016
000.000.456-12	ronaldo	000.000.456-12	SI	97	Rinópolis	1923

print(df1.to_latex())

```
\begin{tabular}{lllllr}
\toprule
{} & Nome & CPF & Curso & Idade & Cidade & Ano_Nascimento \\
\midrule
000.000.000-00 & raphael & 000.000.000-00 & SI & 76 & Porto Ferreira & 1944 \\
000.000.123-89 & ricardo & 000.000.123-89 & Crochê & 4 & Jales & 2016 \\
000.000.456-12 & ronaldo & 000.000.456-12 & SI & 97 & Rinópolis & 1923 \\
\bottomrule
\end{tabular}
```

Gravação

Exemplo de gravação de um DataFrame em um Banco de Dados MySQL



The image shows a Jupyter Notebook cell with Python code and a MySQL query window. The Python code creates a DataFrame with three rows of data and saves it to a MySQL database. The MySQL window shows the SQL query used to insert the data and the resulting table.

```
from sqlalchemy import create_engine
import pymysql
import pandas as pd
import numpy as np

#Criando um DataFrame utilizando todos os campos do construtor e indexando pelo CPF
dados = np.array([[['Rafael','000.000.000-00', 'SI'],
                  ['Ricardo','000.000.123-89', 'Crochê'],
                  ['Ronaldo','000.000.456-12', 'SI']
                 ]])

df1 = pd.DataFrame(data=dados,index=dados[:,1],columns=['Nome','CPF','Curso'],copy=False)

tableName = "cadastro"
sqlEngine = create_engine('mysql+pymysql://root:root@localhost/teste_insercao')
dbConnection = sqlEngine.connect()

df1.to_sql(tableName, dbConnection, index=False);

dbConnection.close()
```

MySQL Query Window:

```
Query 1 x SQL File 3 x SQL File 4 x SQL File 5 x SQL File 6 x SQL File 7 x SQL File 8 x
Limit to 1000 rows
1 use teste_insercao;
2
3 select * from cadastro;
```

#	Nome	CPF	Curso
1	Rafael	000.000.000-00	SI
2	Ricardo	000.000.123-89	Crochê
3	Ronaldo	000.000.456-12	SI

Exibindo os Dados

- Pode-se utilizar o próprio identificador de um `DataFrame` ou `Series` como último comando de uma célula para exibir o seu conteúdo, ou ainda pode-se utilizar o comando `print` e passar o `DataFrame` ou `Series` como argumento
- Entretanto, essa exibição normal geralmente “trunca” o conjunto de dados, sendo exibida apenas um número máximo de linhas cujo valor é pré-configurado
- Pode-se alterar o número máximo de linhas utilizando a função `pd.set_option('display.max_rows', [qtd_linhas])`

Exibindo os Dados

Exemplo da alteração da quantidade máximo de linhas exibidas pelo pandas. Quando utilizado o valor `None`, todas as linhas são exibidas.

```
[259] df4
```

	Nome	Motor	Ano	Quilometragem	Zero_km	Acessórios	Valor
0	Jetta Variant	Motor 4.0 Turbo	2003	44410.0	Falso	[Rodas de liga, 'Travas elétricas', 'Piloto...	80079.64
1	Passat	Motor Diesel	1991	5712.0	Falso	['Central multimídia', 'Teto panorâmico', 'Fre...	106161.94
2	Crossfox	Motor Diesel V8	1990	37123.0	Falso	['Piloto automático', 'Controle de estabilidade...	72832.16
3	D55	Motor 2.4 Turbo	2019	NaN	True	['Travas elétricas', '4 X 4', 'Vidros elétricos...	124549.07
4	Aston Martin DB4	Motor 2.4 Turbo	2006	25757.0	Falso	['Rodas de liga', '4 X 4', 'Central multimídia...	82612.10
...
253	Phantom 2013	Motor V8	2014	27505.0	Falso	['Controle de estabilidade', 'Piloto automático...	51759.58
254	Cadillac Ciel concept	Motor V8	1991	29981.0	Falso	['Bancos de couro', 'Painel digital', 'Sensor...	51697.06
255	Claasse GLK	Motor 5.0 V8 Bi-Turbo	2002	52637.0	Falso	['Rodas de liga', 'Controle de tração', 'Câmbi...	48994.03
256	Aston Martin DB5	Motor Diesel	1999	7695.0	Falso	['Ar condicionado', '4 X 4', 'Câmbio automati...	122110.90
257	Macan	Motor Diesel V6	1992	50188.0	Falso	['Central multimídia', 'Teto panorâmico', 'Vid...	80381.47

258 rows x 7 columns

```
[264] pd.set_option('display.max_rows', None)
```

```
df4
```

	Nome	Motor	Ano	Quilometragem	Zero_km	Acessórios	Valor
0	Jetta Variant	Motor 4.0 Turbo	2003	44410.0	Falso	[Rodas de liga, 'Travas elétricas', 'Piloto...	88070.64
1	Passat	Motor Diesel	1991	5712.0	Falso	['Central multimídia', 'Teto panorâmico', 'Fre...	106161.94
2	Crossfox	Motor Diesel V8	1990	37123.0	Falso	['Piloto automático', 'Controle de estabilidade...	72832.16
3	D55	Motor 2.4 Turbo	2019	NaN	True	['Travas elétricas', '4 X 4', 'Vidros elétricos...	124549.07
4	Aston Martin DB4	Motor 2.4 Turbo	2006	25757.0	Falso	['Rodas de liga', '4 X 4', 'Central multimídia...	82612.10
5	Palo Weekend	Motor 1.8 16v	2012	10728.0	Falso	['Sensor de estacionamento', 'Teto panorâmico...	97497.73
6	A5	Motor 4.0 Turbo	2019	NaN	True	['Câmbio automático', 'Câmera de estacionamento...	56445.20
7	Serie 3 Cabrio	Motor 1.0 8v	2009	77599.0	Falso	['Controle de estabilidade', 'Sensor creepcon...	112330.44
8	Dodge Journey	Motor 3.0 32v	2010	96197.0	Falso	['Vidros elétricos', 'Piloto automático', 'Tet...	120716.27
9	Carens	Motor 5.0 V8 Bi-Turbo	2011	37978.0	Falso	['Ar condicionado', 'Painel digital', 'Central...	74566.49
...

Exibindo os Dados

- Caso não queira exibir todas as linhas, pode-se utilizar a função `head()` para exibir as n primeiras linhas de um `DataFrame` ou `Series`

```
[88] df4
```

	Nome	Motor	Ano	Quilometragem	Zero_km	Acessórios	Valor
0	Jetta Variant	Motor 4.0 Turbo	2003	44410.0	False	[Rodas de liga, 'Travas elétricas', 'Piloto ...	88078.64
1	Passat	Motor Diesel	1991	5712.0	False	['Central multimídia', 'Teto panorâmico', 'Fre...	106161.94
2	Crossfox	Motor Diesel V8	1990	37123.0	False	['Piloto automático', 'Controle de estabilidade...	72832.16
3	DSS	Motor 2.4 Turbo	2019	NaN	True	['Travas elétricas', '4 X 4', 'Vidros elétric...	124549.07
4	Aston Martin DB4	Motor 2.4 Turbo	2006	25757.0	False	[Rodas de liga, '4 X 4', 'Central multimídia...	82612.10
...
253	Phantom 2013	Motor V8	2014	27505.0	False	['Controle de estabilidade', 'Piloto automático...	51759.58
254	Cadillac Ciel concept	Motor V8	1991	29382.0	False	['Bancos de couro', 'Painel digital', 'Sensor ...	51667.06
255	Classe GLK	Motor 5.0 V8 Bi-Turbo	2002	52637.0	False	[Rodas de liga, 'Controle de tração', 'Câmbi...	68934.03
256	Aston Martin DB5	Motor Diesel	1996	7685.0	False	['Ar condicionado', '4 X 4', 'Câmbio automático...	122110.90
257	Macan	Motor Diesel V6	1992	50188.0	False	['Central multimídia', 'Teto panorâmico', 'Vid...	90383.47

258 rows x 7 columns

```
df4.head(n=10)
```

	Nome	Motor	Ano	Quilometragem	Zero_km	Acessórios	Valor
0	Jetta Variant	Motor 4.0 Turbo	2003	44410.0	False	[Rodas de liga, 'Travas elétricas', 'Piloto ...	88078.64
1	Passat	Motor Diesel	1991	5712.0	False	['Central multimídia', 'Teto panorâmico', 'Fre...	106161.94
2	Crossfox	Motor Diesel V8	1990	37123.0	False	['Piloto automático', 'Controle de estabilidade...	72832.16
3	DSS	Motor 2.4 Turbo	2019	NaN	True	['Travas elétricas', '4 X 4', 'Vidros elétric...	124549.07
4	Aston Martin DB4	Motor 2.4 Turbo	2006	25757.0	False	[Rodas de liga, '4 X 4', 'Central multimídia...	82612.10
5	Pallo Weekend	Motor 1.8 16v	2012	10728.0	False	['Sensor de estacionamento', 'Teto panorâmico...	97497.73
6	AS	Motor 4.0 Turbo	2019	NaN	True	['Câmbio automático', 'Câmera de estacionamento...	96445.20
7	Série 3 Cabrio	Motor 1.0 8v	2009	77599.0	False	['Controle de estabilidade', 'Sensor espessur...	112310.44
8	Dodge Journey	Motor 3.0 32v	2010	96197.0	False	['Vidros elétricos', 'Piloto automático', 'Tel...	120718.27
9	Carens	Motor 5.0 V8 Bi-Turbo	2011	37978.0	False	['Ar condicionado', 'Painel digital', 'Centra...	76966.49

Exibindo os Dados

- Da mesma forma, pode-se utilizar a função `tail()` para exibir as últimas n linhas

[91] df4

	Nome	Motor	Ano	Quilometragem	Zero_km	Acessórios	Valor
0	Jetta Variant	Motor 4.0 Turbo	2003	44410.0	False	[Rodas de liga', 'Travas elétricas', 'Piloto ...	89078.64
1	Passat	Motor Diesel	1991	5712.0	False	['Central multimídia', 'Teto panorâmico', 'Fre...	106161.94
2	Crossfox	Motor Diesel V8	1990	37123.0	False	['Piloto automático', 'Controle de estabilidade...	72832.16
3	DS5	Motor 2.4 Turbo	2019	NaN	True	['Travas elétricas', '4 X 4', 'Vidros elétricos...	124549.07
4	Aston Martin DB4	Motor 2.4 Turbo	2006	25757.0	False	['Rodas de liga', '4 X 4', 'Central multimídia...	92612.10
...
253	Phantom 2013	Motor V8	2014	27505.0	False	['Controle de estabilidade', 'Piloto automátic...	51759.58
254	Cadillac Ciel concept	Motor V8	1991	29981.0	False	['Bancos de couro', 'Painel digital', 'Sensor ...	51667.06
255	Classe GLK	Motor 5.0 V8 Bi-Turbo	2002	52637.0	False	['Rodas de liga', 'Controle de tração', 'Câmbi...	68934.03
256	Aston Martin DB5	Motor Diesel	1996	7685.0	False	['Ar condicionado', '4 X 4', 'Câmbio automátic...	122110.90
257	Macan	Motor Diesel V6	1992	50188.0	False	['Central multimídia', 'Teto panorâmico', 'Vid...	90381.47

258 rows x 7 columns

df4.tail(n=10)

	Nome	Motor	Ano	Quilometragem	Zero_km	Acessórios	Valor
248	XC60	Motor 4.0 Turbo	2019	NaN	True	['Painel digital', 'Piloto automático', 'Centr...	77675.79
249	LS 460L	Motor Diesel V8	2010	89685.0	False	['Rodas de liga', 'Freios ABS', 'Controle de L...	58881.67
250	Lamborghini Sesto Elemento	Motor V8	2007	85384.0	False	['Sensor crepuscular', 'Freios ABS', 'Ar condi...	55081.99
251	Lamborghini Huracán	Motor V8	1994	98108.0	False	['Sensor de chuva', 'Sensor de estacionamento'...	118826.44
252	A7	Motor 1.0 8v	2007	71280.0	False	['Sensor de chuva', 'Vidros elétricos', 'Senso...	137627.62
253	Phantom 2013	Motor V8	2014	27505.0	False	['Controle de estabilidade', 'Piloto automátic...	51759.58
254	Cadillac Ciel concept	Motor V8	1991	29981.0	False	['Bancos de couro', 'Painel digital', 'Sensor ...	51667.06
255	Classe GLK	Motor 5.0 V8 Bi-Turbo	2002	52637.0	False	['Rodas de liga', 'Controle de tração', 'Câmbi...	68934.03
256	Aston Martin DB5	Motor Diesel	1996	7685.0	False	['Ar condicionado', '4 X 4', 'Câmbio automátic...	122110.90
257	Macan	Motor Diesel V6	1992	50188.0	False	['Central multimídia', 'Teto panorâmico', 'Vid...	90381.47

Acessando Colunas

- Para retornar uma coluna de um DataFrame, existem duas possibilidades:
 - `[identificador_df]['nome_da_coluna']`
 - `[identificador_df].nome_da_coluna` caso não haja espaços no nome da coluna
- Pode-se também retornar várias colunas passando uma lista com o nome das colunas de acordo com a seguinte sintaxe:
`[identificador_df][lista_nome_das_colunas]`
- No caso do acesso à uma única coluna, o tipo de retorno é uma `Series`

Acessando Colunas

```
[44] df1
```

```
Nome      CPF      Curso
000.000.000-00  Rafael  000.000.000-00  SI
000.000.123-89  Ricardo 000.000.123-89  Crochê
000.000.456-12  Ricardo 000.000.456-12  SI
```

```
[45] df1['Nome']
```

```
000.000.000-00    Rafael
000.000.123-89    Ricardo
000.000.456-12    Ricardo
Name: Nome, dtype: object
```

```
[48] type(df1['Nome'])
```

```
pandas.core.series.Series
```

```
[47] df1[['Nome', 'CPF']]
```

```
Nome      CPF
000.000.000-00  Rafael  000.000.000-00
000.000.123-89  Ricardo 000.000.123-89
000.000.456-12  Ricardo 000.000.456-12
```

```
type(df1[['Nome', 'CPF']])
```

```
pandas.core.frame.DataFrame
```

Acessando Coluna

[97] df1

	Nome	CPF	Curso
000.000.000-00	Rafael	000.000.000-00	SI
000.000.123-89	Ricardo	000.000.123-89	Crochê
000.000.456-12	Ricardo	000.000.456-12	SI

df1.Nome

000.000.000-00 Rafael
000.000.123-89 Ricardo
000.000.456-12 Ricardo
Name: Nome, dtype: object

Acessando Linhas

- Pode-se acessar / retornar as linhas de um DataFrame ou Series utilizando as funções `loc` ou `iloc`
- A função `iloc` recebe o índice posicional como argumento
- Já a função `loc` recebe o índice que identifica as linhas
- A função `loc` também pode receber um *array* de *booleans* em que todos os elementos com o valor `True` serão selecionados

Acessando Linhas

- Em casa do retorno de uma linha, o tipo de retorno é uma `Series`
- No caso do retorno de múltiplas linhas, o tipo de retorno é um `DataFrame`
- Vale ressaltar que é possível também selecionar colunas também com as funções `loc` e `iloc`

Acessando Linhas

```
[58] df1
```

	Nome	CPF	Curso
000.000.000-00	Rafael	000.000.000-00	SI
000.000.123-89	Ricardo	000.000.123-89	Crochê
000.000.456-12	Ricardo	000.000.456-12	SI

```
[59] # Acessando a primeira tupla do DataFrame df1  
df1.iloc[0]
```

Nome	Rafael
CPF	000.000.000-00
Curso	SI
Name: 000.000.000-00, dtype: object	

```
[60] # Quando retornado um único elemento, o tipo de dado é um Series  
type(df1.iloc[0])
```

```
pandas.core.series.Series
```

```
[61] # Pode-se utilizar slicing com o iloc igual ao Numpy  
df1.iloc[0:2]
```

	Nome	CPF	Curso
000.000.000-00	Rafael	000.000.000-00	SI
000.000.123-89	Ricardo	000.000.123-89	Crochê

```
# Quando retornado mais de uma tupla, o tipo de dado é um DataFrame  
type(df1.iloc[0:2])
```

```
pandas.core.frame.DataFrame
```

Acessando Linhas

```
[64] df1
```

	Nome	CPF	Curso
000.000.000-00	Rafael	000.000.000-00	SI
000.000.123-89	Ricardo	000.000.123-89	Crochê
000.000.456-12	Ricardo	000.000.456-12	SI

```
[65] # Retornando uma única tupla passando o índice da tupla entre colchetes  
df1.loc['000.000.000-00']
```

```
Nome      Rafael  
CPF      000.000.000-00  
Curso      SI  
Name: 000.000.000-00, dtype: object
```

```
# Retornando mais de uma tupla passando os índices em uma lista  
df1.loc[['000.000.000-00', '000.000.123-89']]
```

	Nome	CPF	Curso
000.000.000-00	Rafael	000.000.000-00	SI
000.000.123-89	Ricardo	000.000.123-89	Crochê

Acessando Linhas

```
[93] df1
```

	Nome	CPF	Curso
000.000.000-00	Rafael	000.000.000-00	SI
000.000.123-89	Ricardo	000.000.123-89	Crochê
000.000.456-12	Ricardo	000.000.456-12	SI

```
[94] # Utilizando uma listas de booleans para selecionar tuplas com o loc  
df1.loc[[True,False,True]]
```

	Nome	CPF	Curso
000.000.000-00	Rafael	000.000.000-00	SI
000.000.456-12	Ricardo	000.000.456-12	SI

```
# Utilizando uma listas de booleans para selecionar tuplas diretamente no DataFrame  
df1[[True,False,True]]
```

	Nome	CPF	Curso
000.000.000-00	Rafael	000.000.000-00	SI
000.000.456-12	Ricardo	000.000.456-12	SI

Acessando Elementos

- Quando se retorna uma única tupla em formato de `Series`, e pode-se acessar uma `Series` pelo índice
- Como o acesso à uma única coluna ou o acesso à uma única linha retorna uma `Series`, podemos combinar os dois tipos de acesso para acessar uma célula do `DataFrame`

Acessando Elementos

```
[79] df1
```

	Nome	CPF	Curso
000.000.000-00	Rafael	000.000.000-00	SI
000.000.123-89	Ricardo	000.000.123-89	Crochê
000.000.456-12	Ricardo	000.000.456-12	SI

```
[80] # Extraindo uma tupla e acessando o elemento pelo índice  
df1.iloc[0]['Nome']
```

```
'Rafael'
```

```
[81] df1.loc['000.000.000-00']['Nome']
```

```
'Rafael'
```

```
[82] # Extraindo uma coluna e acessando o elemento pelo índice  
df1['Nome'].iloc[0]
```

```
'Rafael'
```

```
df1['Nome'].loc['000.000.000-00']
```

```
'Rafael'
```

Acessando Elementos

```
[84] df1
```

	Nome	CPF	Curso
000.000.000-00	Rafael	000.000.000-00	SI
000.000.123-89	Ricardo	000.000.123-89	Crochê
000.000.456-12	Ricardo	000.000.456-12	SI

```
[86] # Acessando linhas e colunas apenas utilizando índices numéricos  
df1.iloc[0:2,[0,2]]
```

	Nome	Curso
000.000.000-00	Rafael	SI
000.000.123-89	Ricardo	Crochê

```
#Acessando linhas e colunas utilizando índices  
df1.loc[['000.000.000-00', '000.000.456-12'], ['Nome', 'Curso']]
```

	Nome	Curso
000.000.000-00	Rafael	SI
000.000.456-12	Ricardo	SI

Consultas

- Ao aplicar um teste lógico sobre os dados de uma coluna, é retornado um *array* booleano, o qual contém o valor `True` nas respectivas células em que o teste é verdadeiro, e `False` nas respectivas células em que o teste é falso
- Uma vez que é possível selecionar elementos de um `DataFrame` ou `Series` por meio de um *array* booleano, é possível utilizar o recurso do teste lógico para fazer consultas

Consultas

```
[181] df4.head(n=5)
```

	Nome	Motor	Ano	Quilometragem	Zero_kn	Acessórios	Valor
0	Jetta Variant	Motor 4.0 Turbo	2003	44410.0	False	['Rodas de liga', 'Travas elétricas', 'Piloto ...	88078.64
1	Passat	Motor Diesel	1991	5712.0	False	['Central multimídia', 'Teto panorâmico', 'Fre...	106161.94
2	Crossfox	Motor Diesel V8	1990	37123.0	False	['Piloto automático', 'Controle de estabilidade...	72832.16
3	DS5	Motor 2.4 Turbo	2019	NaN	True	['Travas elétricas', '4 X 4', 'Vidros elétrico...	124549.07
4	Aston Martin DB4	Motor 2.4 Turbo	2006	25757.0	False	['Rodas de liga', '4 X 4', 'Central multimídia...	92612.10

```
[180] # Um teste lógico irá retornar uma lista (Series) de boolean  
df4['Ano'] >= 2010
```

```
0    False  
1    False  
2    False  
3     True  
4    False  
...  
253    True  
254   False  
255   False  
256   False  
257   False  
Name: Ano, Length: 258, dtype: bool
```

```
• # É possível retornar os elementos utilizando a lista de boolean  
df4[df4['Ano'] >= 2010].head(n=5)
```

	Nome	Motor	Ano	Quilometragem	Zero_kn	Acessórios	Valor
3	DS5	Motor 2.4 Turbo	2019	NaN	True	['Travas elétricas', '4 X 4', 'Vidros elétrico...	124549.07
5	Palo Weekend	Motor 1.8 16v	2012	10728.0	False	['Sensor de estacionamento', 'Teto panorâmico'...	97497.73
6	A5	Motor 4.0 Turbo	2019	NaN	True	['Câmbio automático', 'Câmera de estacionamento...	56445.20
8	Dodge Journey	Motor 3.0 32v	2010	99197.0	False	['Vidros elétricos', 'Piloto automático', 'Tet...	120716.27
9	Carens	Motor 5.0 V8 Bi-Turbo	2011	37978.0	False	['Ar condicionado', 'Painel digital', 'Central...	76566.49

Consultas

- Pode-se também fazer uso dos operadores lógicos nas consultas:
 - OU: |
 - E: &
 - NEGAÇÃO: ~
 - OU EXCLUSIVO: ^

Consultas

```
[101] df4.head(n=5)
```

	Nome	Motor	Ano	Quilometragem	Zero_km	Acessórios	Valor
0	Jetta Variant	Motor 4.0 Turbo	2003	44410.0	False	['Rodas de liga', 'Travas elétricas', 'Piloto ...	88078.64
1	Passat	Motor Diesel	1991	5712.0	False	['Central multimídia', 'Teto panorâmico', 'Fre...	106161.94
2	Crossfox	Motor Diesel V8	1990	37123.0	False	['Piloto automático', 'Controle de estabilidade...	72832.16
3	DS5	Motor 2.4 Turbo	2019	NaN	True	['Travas elétricas', '4 X 4', 'Vidros elétrico...	124549.07
4	Aston Martin DB4	Motor 2.4 Turbo	2006	25757.0	False	['Rodas de liga', '4 X 4', 'Central multimídia...	92612.10

```
[105] # Usando um teste lógico composto (é necessário utilizar o parênteses em cada teste)
(df4['Ano'] <= 2010) | (df4['Zero_km'] == True)
```

```
0      True
1      True
2      True
3      True
4      True
...
253  False
254    True
255    True
256    True
257    True
Length: 258, dtype: bool
```

```
# É possível retornar os elementos utilizando a lista de boolean
df4[(df4['Ano'] <= 2010) | (df4['Zero_km'] == True)].head(n=5)
```

	Nome	Motor	Ano	Quilometragem	Zero_km	Acessórios	Valor
0	Jetta Variant	Motor 4.0 Turbo	2003	44410.0	False	['Rodas de liga', 'Travas elétricas', 'Piloto ...	88078.64
1	Passat	Motor Diesel	1991	5712.0	False	['Central multimídia', 'Teto panorâmico', 'Fre...	106161.94
2	Crossfox	Motor Diesel V8	1990	37123.0	False	['Piloto automático', 'Controle de estabilidade...	72832.16
3	DS5	Motor 2.4 Turbo	2019	NaN	True	['Travas elétricas', '4 X 4', 'Vidros elétrico...	124549.07
4	Aston Martin DB4	Motor 2.4 Turbo	2006	25757.0	False	['Rodas de liga', '4 X 4', 'Central multimídia...	92612.10

Consultas

- Pode-se também fazer uso da função `query` para realizar as consultas
- Essa função recebe como parâmetro uma *string* contendo os testes lógicos sobre as colunas do `DataFrame`

Consultas

```
[113] df4.head(n=5)
```

		None	Motor	Ano	Quilometragem	Zero_km		Acessórios	Valor
0	Jetta Variant	Motor 4.0 Turbo	2003	44410.0	False	[Rodas de liga, 'Travas elétricas', 'Piloto ...		88078.64	
1	Passat	Motor Diesel	1991	5712.0	False	['Central multimídia', 'Teto panorâmico', 'Fre...		106161.94	
2	Crossfox	Motor Diesel V8	1990	37123.0	False	['Piloto automático', 'Controle de estabilidade...		72832.16	
3	DS5	Motor 2.4 Turbo	2019	NaN	True	['Travas elétricas', '4 X 4', 'Vidros elétrico...		124549.07	
4	Aston Martin DB4	Motor 2.4 Turbo	2006	25757.0	False	['Rodas de liga', '4 X 4', 'Central multimídia...		92612.10	

```
df4.query('Ano > 2010 & Valor < 60000')
```

		None	Motor	Ano	Quilometragem	Zero_km		Acessórios	Valor
6	A5	Motor 4.0 Turbo	2019	NaN	True	['Câmbio automático', 'Câmera de estacionamento...		56445.20	
13	J5	Motor V6	2019	NaN	True	['Sensor crepuscular', 'Painel digital', 'Roda...		53183.38	
15	RAM	Motor Diesel V8	2016	115607.0	False	['Sensor crepuscular', 'Câmbio automático', 'S...		59910.40	
32	Linea	Motor 1.0 8v	2018	17720.0	False	['Piloto automático', 'Painel digital', 'Vidro...		59358.69	
36	Pajero TR4	Motor 2.4 Turbo	2019	NaN	True	['Controle de tração', 'Bancos de couro', 'Câmb...		51606.59	
79	Livina	Motor 2.0 16v	2019	NaN	True	['Sensor crepuscular', 'Câmera de estacionamento...		50742.10	
94	Cadillac CTS	Motor 1.8 16v	2018	80661.0	False	['Travas elétricas', 'Sensor de chuva', 'Freio...		53894.61	
113	J3	Motor 1.8 16v	2013	15196.0	False	['Controle de estabilidade', '4 X 4', 'Câmera ...		56979.30	
130	Classe CL	Motor V8	2019	NaN	True	['Central multimídia', 'Câmbio automático', '4...		58525.41	
174	Lamborghini Jalpa	Motor 1.8 16v	2018	9146.0	False	['Piloto automático', 'Freios ABS', 'Sensor de...		54388.11	
177	C3 Picasso	Motor 3.0 32v	2019	NaN	True	['Teto panorâmico', 'Sensor crepuscular', 'Sen...		56453.94	
186	T6	Motor 3.0 32v	2019	NaN	True	['Sensor de chuva', 'Câmera de estacionamento...		56842.79	
204	Sedan	Motor V8	2019	NaN	True	['Bancos de couro', 'Central multimídia', 'Fre...		57888.10	
253	Phantom 2013	Motor V8	2014	27505.0	False	['Controle de estabilidade', 'Piloto automátic...		51759.58	

Consultas

- Pode-se comparar o conteúdo exato de uma coluna com campos textuais utilizando o operador '=='
- Porém, para outros tipos de textos envolvendo o conteúdo textual, como o `contains` ou `startswith`, deve-se fazer uso do campo `str` de uma coluna

Consultas

```
df4[df4['Motor'] == 'Motor V6']
```

	Nome	Motor	Ano	Quilometragem	Zero_km	Acessórios	Valor
12	S-18	Motor V6	2001	89773.0	False	['Bancos de couro', 'Sensor de chuva', 'Rodas ...	112732.99
13	J5	Motor V6	2019	NaN	True	['Sensor crepuscular', 'Painel digital', 'Roda...	53183.38
37	Polo Sedan	Motor V6	2019	NaN	True	['Sensor de chuva', 'Sensor crepuscular', 'Ar ...	123281.87
41	HB20	Motor V6	2003	40762.0	False	['Sensor de chuva', 'Travas elétricas', 'Contr...	135235.29
56	Aston Martin DB7 Vantage	Motor V6	2007	119513.0	False	['Sensor crepuscular', 'Piloto automático', 'T...	128694.23
81	March	Motor V6	2007	79607.0	False	['Bancos de couro', 'Piloto automático', 'Sens...	54332.87
83	Lamborghini Centenario	Motor V6	1995	42733.0	False	['Central multimídia', 'Vidros elétricos', 'Se...	63578.50
92	Star	Motor V6	2016	8356.0	False	['Câmera de estacionamento', '4 X 4', 'Central...	102241.54
101	Idea	Motor V6	2001	70641.0	False	['Central multimídia', 'Piloto automático', 'T...	139549.67
104	Edge	Motor V6	2018	26544.0	False	['Freios ABS', 'Controle de tração', 'Sensor c...	88416.54
131	500 Abarth	Motor V6	1994	56944.0	False	['4 X 4', 'Teto panorâmico', 'Controle de traç...	90345.05
136	Aston Martin One-77	Motor V6	1993	30705.0	False	['Sensor de estacionamento', 'Rodas de liga', ...	141982.93
155	RX 350	Motor V6	2013	2314.0	False	['Piloto automático', 'Sensor crepuscular', 'A...	138823.81
200	J3 Turin	Motor V6	2019	NaN	True	['Sensor crepuscular', 'Câmera de estacionamen...	127024.75
217	R8	Motor V6	1997	96100.0	False	['4 X 4', 'Controle de estabilidade', 'Câmera ...	128565.61
224	Boxer	Motor V6	2005	53369.0	False	['Travas elétricas', 'Piloto automático', 'Vid...	112146.80
235	A3 Sedan	Motor V6	2019	NaN	True	['Ar condicionado', 'Sensor de chuva', 'Travas...	90141.47
241	Elfa Hafei Pícape Cabine Dupla	Motor V6	2008	112787.0	False	['Sensor crepuscular', 'Controle de tração', '...	141645.08
246	RS5	Motor V6	1996	55083.0	False	['Painel digital', 'Câmbio automático', 'Vidro...	89536.82
247	Cerato	Motor V6	2011	48796.0	False	['Sensor de estacionamento', 'Câmera de estaci...	87975.30

Consultas

```
df4[df4['Motor'].str.contains('16v')].head(n=10)
```

	Nome	Motor	Ano	Quilometragem	Zero_km	Acessórios	Valor
5	Palio Weekend	Motor 1.8 16v	2012	10728.0	False	['Sensor de estacionamento', 'Teto panorâmico'...	97497.73
31	Gol G4	Motor 2.0 16v	1995	53332.0	False	['Sensor crepuscular', '4 X 4', 'Vidros elêtri...	69105.18
33	Ford F100	Motor 2.0 16v	1991	33808.0	False	['Controle de tração', 'Sensor crepuscular', '...	65955.40
40	Tucson	Motor 2.0 16v	2013	93415.0	False	['Vidros elétricos', 'Sensor crepuscular', 'Fr...	67997.19
42	Optima	Motor 1.8 16v	2019	NaN	True	['Central multimídia', 'Ar condicionado', 'Tet...	86641.34
59	EXTRA 103EX	Motor 2.0 16v	2019	NaN	True	['Sensor de estacionamento', 'Central multimed...	112033.27
64	Lamborghini Ankonian	Motor 2.0 16v	2001	100912.0	False	['Bancos de couro', 'Câmera de estacionamento'...	142623.02
72	Doblo	Motor 1.8 16v	1997	54395.0	False	['Câmera de estacionamento', 'Teto panorâmico'...	78283.49
75	C5	Motor 2.0 16v	1994	118236.0	False	['Painel digital', 'Teto panorâmico', 'Piloto ...	60813.92
79	Livina	Motor 2.0 16v	2019	NaN	True	['Sensor crepuscular', 'Câmera de estacionamen...	50742.10

Consultas

```
df4[df4['Nome'].str.startswith('Lamborghini')].head(n=20)
```

	Nome	Motor	Ano	Quilometragem	Zero_km	Acessórios	Valor
22	Lamborghini Obvious	Motor Diesel V6	1994	98079.0	False	['Rodas de liga', 'Câmbio automático', 'Travas...	133529.84
24	Lamborghini Murciélago	Motor 5.0 V8 Bi-Turbo	2019	NaN	True	['Freios ABS', 'Câmbio automático', 'Ar condic...	121596.21
64	Lamborghini Ankonian	Motor 2.0 16v	2001	100912.0	False	['Bancos de couro', 'Câmera de estacionamento'...	142623.02
73	Lamborghini Egoista	Motor 3.0 32v	2006	26731.0	False	['Rodas de liga', 'Controle de tração', 'Centr...	83434.04
83	Lamborghini Centenario	Motor V6	1995	42733.0	False	['Central multimídia', 'Vidros elétricos', 'Se...	63578.50
106	Lamborghini Asterion LPI 910-4	Motor 5.0 V8 Bi-Turbo	1992	47503.0	False	['Ar condicionado', 'Painel digital', 'Control...	70709.34
109	Lamborghini LM002	Motor 1.8 16v	1995	110564.0	False	['Bancos de couro', 'Freios ABS', 'Central mul...	85052.26
117	Lamborghini Ferruccio	Motor 3.0 32v	2019	NaN	True	['Vidros elétricos', 'Piloto automático', 'Tet...	149157.17
137	Lamborghini Veneno	Motor 3.0 32v	2019	NaN	True	['4 X 4', 'Sensor de estacionamento', 'Câmbio ...	104038.79
174	Lamborghini Jalpa	Motor 1.8 16v	2018	9146.0	False	['Piloto automático', 'Freios ABS', 'Sensor de...	54388.11
238	Lamborghini Reventón	Motor 4.0 Turbo	2019	NaN	True	['Controle de tração', 'Ar condicionado', 'Cen...	67664.86
250	Lamborghini Sesto Elemento	Motor V8	2007	85384.0	False	['Sensor crepuscular', 'Freios ABS', 'Ar condi...	55081.99
251	Lamborghini Huracán	Motor V8	1994	98108.0	False	['Sensor de chuva', 'Sensor de estacionamento'...	118826.44

Percorrendo os Dados

- Podemos percorrer os elementos de um DataFrame acessando individualmente as linhas e colunas

```
[206] row_count, column_count = df1.shape
```

```
▶ for row in range(row_count):  
    for column in range(column_count):  
        print(df1.iloc[row,column], end='\t')  
    print()
```

```
↳ Rafael 000.000.000-00 SI  
Ricardo 000.000.123-89 Crochê  
Ricardo 000.000.456-12 SI
```

Percorrendo os Dados

- Podemos também percorrer os elementos utilizando a função `iterrows()`, o qual é um iterator, e a cada iteração retorna o índice de uma linha e a própria linha no formato de uma Series

```
[208] df1
```

	Nome	CPF	Curso
000.000.000-00	Rafael	000.000.000-00	SI
000.000.123-89	Ricardo	000.000.123-89	Crochê
000.000.456-12	Ricardo	000.000.456-12	SI

```
# iterrows retorna uma tupla - indice e series
for indice, linha in df1.iterrows():
    print(indice, linha)
    print('-----')
```

```
000.000.000-00 Nome          Rafael
CPF          000.000.000-00
Curso        SI
Name: 000.000.000-00, dtype: object
-----
000.000.123-89 Nome          Ricardo
CPF          000.000.123-89
Curso        crochê
Name: 000.000.123-89, dtype: object
-----
000.000.456-12 Nome          Ricardo
CPF          000.000.456-12
Curso        SI
Name: 000.000.456-12, dtype: object
-----
```

Percorrendo os Dados

- Já a função `items()` é um iterator em que a cada iteração retorna o índice/nome da coluna e uma Series com os dados da coluna

[216] df1

	Nome	CPF	Curso
000.000.000-00	Rafael	000.000.000-00	SI
000.000.123-89	Ricardo	000.000.123-89	Crochê
000.000.456-12	Ricardo	000.000.456-12	SI

```
# item retorna uma tupla - nome da coluna e dados da coluna (indexados)
for indice, coluna in df1.items():
    print(indice, coluna)
    print('-----')
```

```
Nome 000.000.000-00    Rafael
000.000.123-89    Ricardo
000.000.456-12    Ricardo
Name: Nome, dtype: object
-----
CPF 000.000.000-00    000.000.000-00
000.000.123-89    000.000.123-89
000.000.456-12    000.000.456-12
Name: CPF, dtype: object
-----
Curso 000.000.000-00    SI
000.000.123-89    Crochê
000.000.456-12    SI
Name: Curso, dtype: object
-----
```

Percorrendo os Dados

- O método `itertuples()` retorna uma tupla indexada para cada linha do DataFrame
- O índice corresponde aos nomes das colunas

```
#itertuples retorna uma tupla nomeada com o conteúdo de cada linha
for entrada in df1.itertuples():
    print(entrada)
```

Pandas(Index='000.000.000-00', Nome='Rafael', CPF='000.000.000-00', Curso='SI')
Pandas(Index='000.000.123-89', Nome='Ricardo', CPF='000.000.123-89', Curso='Crochê')
Pandas(Index='000.000.456-12', Nome='Ricardo', CPF='000.000.456-12', Curso='SI')

Percorrendo os Dados

```
[224] def lowercase(nome):  
      return nome.lower()
```

```
[225] df1
```



	Nome	CPF	Curso
000.000.000-00	Rafael	000.000.000-00	SI
000.000.123-89	Ricardo	000.000.123-89	Crochê
000.000.456-12	Ricardo	000.000.456-12	SI

```
[226] df1['Nome'] = df1['Nome'].apply(lowercase)
```

```
[226] df1
```



	Nome	CPF	Curso
000.000.000-00	rafael	000.000.000-00	SI
000.000.123-89	ricardo	000.000.123-89	Crochê
000.000.456-12	ricardo	000.000.456-12	SI

Percorrendo os Dados

- Quando a função *apply* é aplicada ao DataFrame, por padrão, a cada iteração será aplicada a função *callback* passando a Series de uma coluna
- Pode-se alterar esse comportamento para que a cada iteração seja passada a Series de cada linha → para isso deve-se utilizar o parâmetro `axis = 1`

Percorrendo os Dados

```
[269] df14 = pd.DataFrame(np.array([[1,2,3],  
                                  [4,5,6],  
                                  [7,8,9]]), columns=['Atr1', 'Atr2', 'Atr3'])
```

```
[270] df14
```

```
┌─ Atr1  Atr2  Atr3
```

0	1	2	3
1	4	5	6
2	7	8	9

```
[271] def padroniza(series):  
      return series / series.sum()
```

```
[273] #Padronizando Colunas  
df14.apply(padroniza)
```

```
┌─ Atr1    Atr2    Atr3
```

0	0.083333	0.133333	0.166667
1	0.333333	0.333333	0.333333
2	0.583333	0.533333	0.500000

```
➤ #Padronizando Linhas  
df14.apply(padroniza, axis = 1)
```

```
┌─ Atr1    Atr2    Atr3
```

0	0.166667	0.333333	0.500
1	0.266667	0.333333	0.400
2	0.291667	0.333333	0.375

Material Complementar

- Complete Python Pandas Data Science Tutorial! (Reading CSV/Excel files, Sorting, Filtering, Groupby)

<https://www.youtube.com/watch?v=vmEHCJofslg>

- `pandas.Series`

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

- `pandas.DataFrame`

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

Material Complementar

- Python Pandas : How to display full Dataframe i.e. print all rows & columns without truncation

<https://thispointer.com/>

`python-pandas-how-to-display-full-dataframe-i-e-print-all-rows-columns-w`

- Complete Python Pandas Data Science Tutorial! (Reading CSV/Excel files, Sorting, Filtering, Groupby)

<https://www.youtube.com/watch?v=vmEHCJofslg>

- LEARN PANDAS in about 10 minutes! A great python module for Data Science!

https://www.youtube.com/watch?v=iGFdh6_FePU

Imagem do Dia



Tópicos em Inteligência Artificial

<http://ava.ufms.br/>

Rafael Geraldeli Rossi
rafael.g.rossi@ufms.br

Slides baseados nos cursos