

# Ciência de Dados

---

## Aula 2.2 - Engenharia de Dados - Introdução

Prof. Wellington Franco



UNIVERSIDADE  
FEDERAL DO CEARÁ  
CAMPUS DE CRATEÚS



# Agenda

1. Introdução
2. Extração;
3. Tratamento;
4. Limpeza;
5. Manipulação de Dados;

# Lendo arquivos CSV e inserindo em DataFrame

- O Pandas permite upload de arquivos (txt, CSV, xlsx etc) para manipulá-los em DataFrame;
- A seguir, iremos ler um arquivo em CSV e colocá-lo em um DataFrame;
- Os passos apresentados anteriormente são fundamentais para manipular os elementos dentro do arquivo dentro do DataFrame :)

# Lendo arquivos CSV e inserindo em DataFrame

- Vamos fazer o upload de um arquivo CSV chamado “aula2\_dataManipulation.csv”
  - Esse arquivo é um pedaço de um CSV maior que contém informações sobre Tempo médio de resposta a incidentes por ano, mês, classificação e incidente registrado pelo Corpo de Bombeiros de Nova York.
  - O arquivo completo encontra-se em:  
<https://data.cityofnewyork.us/Social-Services/FDNY-Monthly-Response-Times/j34j-vqvt>

# Lendo arquivos CSV e inserindo em DataFrame

- Para ler o arquivo CSV em questão, basta usar a função do Pandas “*read\_csv*”:

```
In [42]: file = pd.read_csv('aula2_dataManipulation.csv')
```

```
In [43]: type(file)
```

```
Out[43]: pandas.core.frame.DataFrame
```

No trecho acima, a variável `file` recebe os dados do CSV já inseridos dentro do DataFrame, como mostra a célula seguinte.

# Lendo arquivos CSV e inserindo em DataFrame

- Se quisermos saber o conteúdo das primeiras linhas de *file*, basta chamar o método *head()*;
- O método *head* exibe as cinco primeiras linhas do *DataFrame*, mas se quiséssemos ver mais linhas, basta colocar o valor desejado entre parênteses.

```
In [44]: file.head(10)
```

```
Out[44]:
```

	YEARMONTH	INCIDENTCLASSIFICATION	INCIDENTBOROUGH	INCIDENTCOUNT	AVERAGERESPONSETIME
0	2009/07	All Fire/Emergency Incidents	Citywide	40850	04:27
1	2009/07	All Fire/Emergency Incidents	Manhattan	10709	04:32
2	2009/07	All Fire/Emergency Incidents	Bronx	8137	04:37
3	2009/07	All Fire/Emergency Incidents	Staten Island	2205	04:45
4	2009/07	All Fire/Emergency Incidents	Brooklyn	11505	04:01
5	2009/07	All Fire/Emergency Incidents	Queens	8294	04:43
6	2009/07	False Alarm	Citywide	2655	04:07
7	2009/07	False Alarm	Manhattan	474	04:07
8	2009/07	False Alarm	Bronx	755	04:26
9	2009/07	False Alarm	Staten Island	192	03:49

# Descrição dos Dados

- O DataFrame possui um método chamado *describe()* que serve para dar uma visão geral de colunas numéricas. Se dermos `file.describe()`, será apresentado apenas um resumo estatístico:

```
In [45]: file.describe()
```

```
Out[45]:
```

INCIDENTCOUNT	
count	24.000000
mean	5209.916667
std	8945.296111
min	10.000000
25%	354.000000
50%	1569.000000
75%	6042.250000
max	40850.000000

# Descrição dos Dados

- Para checarmos os tipos de dados, basta chamarmos o método *info()*:

```
In [46]: file.info()
```

```
<class 'pandas.core.frame.DataFrame'>  
RangeIndex: 24 entries, 0 to 23  
Data columns (total 5 columns):  
#   Column                                Non-Null Count  Dtype  
---  ---                                -  
0   YEARMONTH                            24 non-null     object  
1   INCIDENTCLASSIFICATION              24 non-null     object  
2   INCIDENTBOROUGH                     24 non-null     object  
3   INCIDENTCOUNT                      24 non-null     int64  
4   AVERAGERESPONSETIME                 24 non-null     object  
dtypes: int64(1), object(4)  
memory usage: 1.1+ KB
```



# Iterando sobre DataFrames

- Como já havíamos visto antes, para iterar em DataFrame, para iteramos sob linhas basta usar `iloc[]` ou `loc[]` e, sob colunas, o nome da coluna entre colchetes:

```
In [47]: for i in range(len(file)):
         print(file.loc[i])
```

```
YEARMONTH                2009/07
INCIDENTCLASSIFICATION    All Fire/Emergency Incidents
INCIDENTBOROUGH            Citywide
INCIDENTCOUNT             40850
AVERAGERESPONSETIME        04:27
Name: 0, dtype: object
YEARMONTH                2009/07
INCIDENTCLASSIFICATION    All Fire/Emergency Incidents
INCIDENTBOROUGH            Manhattan
INCIDENTCOUNT             10709
AVERAGERESPONSETIME        04:32
Name: 1, dtype: object
YEARMONTH                2009/07
INCIDENTCLASSIFICATION    All Fire/Emergency Incidents
INCIDENTBOROUGH            Bronx
INCIDENTCOUNT             8137
AVERAGERESPONSETIME        04:37
Name: 2, dtype: object
YEARMONTH                2009/07
```

```
In [45]: len(file)
```

```
Out[45]: 24
```

```
In [46]: range(len(file))
```

```
Out[46]: range(0, 24)
```

Range vai variar de [0,24)

# Iterando sobre DataFrames

- Mas DataFrame também possui um método chamado `iterrows()` que pode tornar a iteração mais legível:

```
In [48]: for index, row in file.iterrows() :  
         print(index, row['YEARMONTH'], row['INCIDENTBOROUGH'])
```

```
0 2009/07 Citywide  
1 2009/07 Manhattan  
2 2009/07 Bronx  
3 2009/07 Staten Island  
4 2009/07 Brooklyn  
5 2009/07 Queens  
6 2009/07 Citywide  
7 2009/07 Manhattan  
8 2009/07 Bronx  
9 2009/07 Staten Island  
10 2009/07 Brooklyn  
11 2009/07 Queens  
12 2009/07 Citywide  
13 2009/07 Manhattan  
14 2009/07 Bronx  
15 2009/07 Staten Island  
16 2009/07 Brooklyn  
17 2009/07 Queens  
18 2009/07 Citywide  
19 2009/07 Manhattan  
20 2009/07 Bronx  
21 2009/07 Staten Island  
22 2009/07 Brooklyn  
23 2009/07 Queens
```

# Agrupamentos de linhas no DataFrame

- Em DataFrame, se for preciso fazer agrupamentos (como o famoso GroupBy nos banco de dados), podemos fazer da seguinte forma:

```
In [51]: file.groupby('INCIDENTCLASSIFICATION').groups
```

```
Out[51]: {'All Fire/Emergency Incidents': Int64Index([0, 1, 2, 3, 4, 5], dtype='int64'),  
         'False Alarm': Int64Index([6, 7, 8, 9, 10, 11], dtype='int64'),  
         'Medical Emergencies': Int64Index([12, 13, 14, 15, 16, 17], dtype='int64'),  
         'Medical False Alarm': Int64Index([18, 19, 20, 21, 22, 23], dtype='int64')}
```

Este método agrupa e retorna os índices de cada elemento de um grupo

```
In [52]: file['INCIDENTCLASSIFICATION'].value_counts()
```

```
Out[52]: Medical False Alarm      6  
         False Alarm             6  
         All Fire/Emergency Incidents 6  
         Medical Emergencies      6  
         Name: INCIDENTCLASSIFICATION, dtype: int64
```

Este método agrupa e calcula a quantidade de elementos por grupo

# Função de Agregação

- As funções de agregação são aquelas que reduzem a dimensão dos *objects* retornados. Isso significa que a saída do DataFrame tem menos ou as mesmas linhas, como o original. Algumas funções comuns de agregação estão tabuladas abaixo:
  - max: retorna o valor máximo do grupo
  - min: retorna o valor mínimo do grupo
  - mean: retorna a média do grupo
  - std: retorna o desvio padrão do grupo
  - sum: retorna a soma do grupo

# Função de Agregação

```
In [53]: file['INCIDENTCOUNT'].agg(np.max)
```

```
Out[53]: 40850
```

```
In [54]: file['INCIDENTCOUNT'].agg(np.min)
```

```
Out[54]: 10
```

```
In [55]: file['INCIDENTCOUNT'].agg(np.mean)
```

```
Out[55]: 5209.916666666667
```

```
In [56]: file['INCIDENTCOUNT'].agg(np.std)
```

```
Out[56]: 8945.296111057072
```

```
In [57]: file['INCIDENTCOUNT'].agg(np.sum)
```

```
Out[57]: 125038
```

# Função de Agregação

Podemos usar função *lambda* como argumento, se for preciso:

```
In [58]: file['INCIDENTCOUNT'].agg(lambda x: (x%2 != 0))
```

```
Out[58]: 0      False
         1      True
         2      True
         3      True
         4      True
         5     False
         6      True
         7     False
         8      True
         9     False
        10      True
        11      True
        12     False
        13      True
        14      True
        15      True
        16     False
        17      True
        18     False
        19     False
        20      True
        21     False
        22      True
        23     False
        Name: INCIDENTCOUNT, dtype: bool
```

Neste exemplo, aplicamos a função *lambda* para checar os elementos da coluna se são ímpares.

# Alterando um elemento do DataFrame

Suponha que você queira alterar um elemento do DataFrame. A maneira mais “imediata” a se fazer, seria atribuindo um valor diretamente ao DataFrame especificando a linha e coluna como a seguir:

```
file["INCIDENTCOUNT"].iloc[8] = 50
```

Neste exemplo, queremos alterar o DataFrame file, na coluna *IncidentCount*, na linha de index 8 para o valor 50

# Alterando um elemento do DataFrame

Ao executar a célula, o valor da linha de índice 8 será alterado, mas apresentará o seguinte *warning*:

```
In [59]: file["INCIDENTCOUNT"].iloc[8] = 50
```

```
C:\ProgramData\Anaconda3\lib\site-packages\pandas\core\indexing.py:670: SettingWithCopyWarning:  
A value is trying to be set on a copy of a slice from a DataFrame
```

```
See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user\_guide/indexing.html#returning-a-view-versus-a-copy
```

```
self._setitem_with_indexer(indexer, value)
```

```
In [60]: file["INCIDENTCOUNT"]
```

```
Out[60]: 0      40850  
1      10709  
2       8137  
3       2205  
4      11505  
5       8294  
6       2655  
7       474  
8        50  
9       192  
10      565
```



# Alterando um elemento do DataFrame

Para alterar o elemento de um DataFrame, de acordo com a documentação do Pandas, usando **LOC**, devemos fazer da seguinte forma:

Rótulo da Linha

Rótulo da Coluna

```
In [61]: file.loc[8, "INCIDENTCOUNT"] = 60
```

```
In [62]: file["INCIDENTCOUNT"]
```

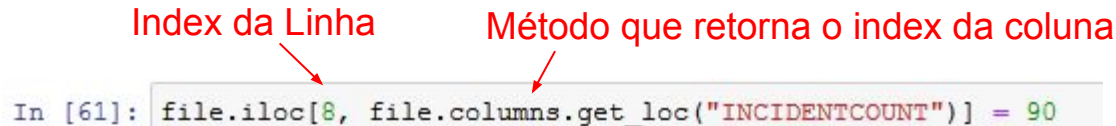
```
Out[62]: 0      40850  
         1      10709  
         2       8137  
         3       2205  
         4      11505  
         5       8294  
         6       2655  
         7        474  
         8        60  
         9       192  
        10       565
```

# Alterando um elemento do DataFrame

Para alterar o elemento de um DataFrame, de acordo com a documentação do Pandas, usando **ILOC**, devemos fazer da seguinte forma:

Index da Linha      Método que retorna o index da coluna

```
In [61]: file.iloc[8, file.columns.get_loc("INCIDENTCOUNT")] = 90
```



```
In [62]: file["INCIDENTCOUNT"]
```

```
Out[62]: 0    40850
         1    10709
         2     8137
         3     2205
         4    11505
         5     8294
         6     2655
         7      474
         8       90
         9     192
```

Se olharmos esse método isoladamente, veremos que ele retorna o index da coluna associada.

```
In [63]: file.columns.get_loc("INCIDENTCOUNT")
```

```
Out[63]: 3
```

# Checando linhas/colunas inválidas (NaN)

- Quando estamos trabalhando com DataFrames, para checar o tipo de uma coluna, basta chamarmos o método:

```
In [64]: file.INCIDENTCOUNT.dtype
```

```
Out[64]: dtype('int64')
```

# Checando linhas/colunas inválidas (NaN)

- Contudo, para nossa experimentação, vamos alterar um elemento do DataFrame para uma string e ver qual será o novo tipo da coluna:

```
In [65]: file.iloc[8, file.columns.get_loc("INCIDENTCOUNT")] = "casa"
```

Alteramos o elemento de index 8 que era um inteiro para uma string

```
In [66]: file["INCIDENTCOUNT"]
```

```
Out[66]: 0    40850
         1    10709
         2     8137
         3     2205
         4    11505
         5     8294
         6     2655
         7      474
         8     casa
         9      192
        10      565
        11      669
        12    18606
        13     4895
        14     3429
        15      933
        16     5344
        17     4005
        18      408
```

← Checando a alteração

```
In [67]: file.INCIDENTCOUNT.dtype
```

```
Out[67]: dtype('O')
```

Verificando o novo tipo. 'O' é do tipo objeto, que pode ter tanto números (int, float) quanto texto

# Checando linhas/colunas inválidas (NaN)

- Precisamos checar sempre que possível os tipos de dados que temos presente nas colunas para que possamos fazer a manipulação dos dados de forma correta e não se deparar com “surpresas” mais na frente.
  - Por exemplo: Querer fazer contas algébricas em uma coluna que possui um elemento textual em alguma linha ou valores NaN.
- É importante também checar elementos do tipo NaN (Not a Number):
  - NaN geralmente são valores que não são números (e nem string);
  - Geralmente pode aparecer depois de erros na hora de inserir valores inválidos;
  - Pode aparecer em operações com resultados indeterminados:

```
In [78]: float("inf") - float("inf")
```

```
Out[78]: nan
```

# Checando linhas/colunas inválidas (NaN)

- Para checarmos se uma coluna possui valores *NaN*, basta usarmos a seguinte função:

```
In [68]: pd.isnull(file["INCIDENTCOUNT"])
```

```
Out[68]: 0    False
         1    False
         2    False
         3    False
         4    False
         5    False
         6    False
         7    False
         8    False
         9    False
        10    False
        11    False
        12    False
        13    False
        14    False
        15    False
        16    False
        17    False
        18    False
        19    False
```

# Checando linhas/colunas inválidas (NaN)

```
In [64]: file.INCIDENTCOUNT.dtype
```

```
Out[64]: dtype('int64')
```

```
In [65]: file.iloc[9, file.columns.get_loc("INCIDENTCOUNT")] = np.nan
```

```
In [66]: file["INCIDENTCOUNT"]
```

```
Out[66]: 0    40850.0  
1    10709.0  
2     8137.0  
3     2205.0  
4    11505.0  
5     8294.0  
6     2655.0  
7      474.0  
8        90.0  
9         NaN  
10     565.0  
11     669.0  
12    18606.0  
13     4895.0  
14     3429.0  
15     933.0  
16     5344.0  
17     4005.0  
18      408.0  
19     550.0
```

```
In [67]: file.INCIDENTCOUNT.dtype
```

```
Out[67]: dtype('float64')
```

- NaN presente nas colunas não a tornam do tipo “O”!

setando NaN manualmente para exemplo

# Célula em Branco

- Imagine que o arquivo que vc vai fazer upload contém um campo que não está preenchido.
  - Vamos fazer o carregamento do arquivo *aula2\_dataManipulation\_2.csv*

```
2009/07,Medical Emergencies,Bronx,3429,04:17  
2009/07,Medical Emergencies,Staten Island,933,04:19  
2009/07,Medical Emergencies,Brooklyn,,03:51  
2009/07,Medical Emergencies,Queens,4005,04:30  
2009/07,Medical False Alarm,Citywide,408,04:13
```

Este campo entre vírgulas não está preenchido!





# Célula em Branco

- Fazendo a leitura do CSV:

```
In [73]: file_2 = pd.read_csv('aula2_dataManipulation_2.csv')
```

```
In [74]: file_2
```

```
Out[74]:
```

	YEARMONTH	INCIDENTCLASSIFICATION	INCIDENTBOROUGH	INCIDENTCOUNT	AVERAGERESPONSETIME
0	2009/07	All Fire/Emergency Incidents	Citywide	40850.0	04:27
1	2009/07	All Fire/Emergency Incidents	Manhattan	10709.0	04:32
2	2009/07	All Fire/Emergency Incidents	Bronx	8137.0	04:37
3	2009/07	All Fire/Emergency Incidents	Staten Island	2205.0	04:45
4	2009/07	All Fire/Emergency Incidents	Brooklyn	11505.0	04:01
5	2009/07	All Fire/Emergency Incidents	Queens	8294.0	04:43
6	2009/07	False Alarm	Citywide	2655.0	04:07
7	2009/07	False Alarm	Manhattan	474.0	04:07
8	2009/07	False Alarm	Bronx	755.0	04:26
9	2009/07	False Alarm	Staten Island	192.0	03:49
10	2009/07	False Alarm	Brooklyn	565.0	03:42

# Célula em Branco

- Se chamarmos o método para detectar se há valores NaN no DataFrame, ele irá retornar:

```
In [75]: file_2.isnull().values.any()
```

```
Out[75]: True
```

- Se passarmos o DataFrame como argumento da função isNull, pode ser demorado a encontrar manualmente:

```
In [76]: pd.isnull(file_2)
```

```
Out[76]:
```

	YEARMONTH	INCIDENTCLASSIFICATION	INCIDENTBOROUGH	INCIDENTCOUNT	AVERAGERESPONSETIME
0	False	False	False	False	False
1	False	False	False	False	False
2	False	False	False	False	False
3	False	False	False	False	False
4	False	False	False	False	False
5	False	False	False	False	False
6	False	False	False	False	False
7	False	False	False	False	False
8	False	False	False	False	False
9	False	False	False	False	False
10	False	False	False	False	False

# Célula em Branco

- Assim, vamos definir uma função que, ao encontrar um valor null (NaN) em uma célula, ele irá nos retornar a linha e coluna:

```
In [77]: def find_for_null_on_DataFrame(df):  
  
    if df.isnull().values.any():  
  
        for i in range(len(df)):  
            for j in range(len(df.columns)):  
                if pd.isnull(df.loc[i][j]):  
                    print("LINE:", i, "COLUMN:", j, "(" + df.columns.values[j] + ")")  
  
    else:  
        print("Não há valor null no DataFrame")
```

# Célula em Branco

- Por fim, chamando o método, temos:

```
In [78]: find_for_null_on_DataFrame(file_2)
```

```
LINE: 16 COLUMN: 3 (INCIDENTCOUNT)
```

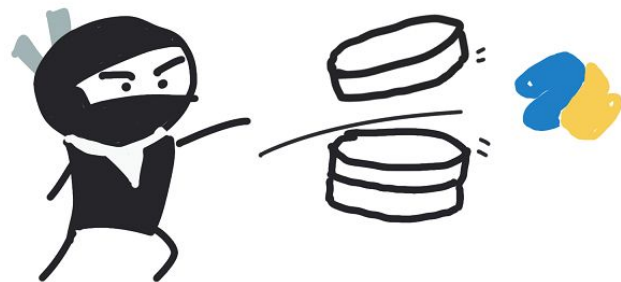
	YEARMONTH	INCIDENTCLASSIFICATION	INCIDENTBOROUGH	INCIDENTCOUNT	AVERAGERESPONSETIME
0	False	False	False	False	False
1	False	False	False	False	False
2	False	False	False	False	False
3	False	False	False	False	False
4	False	False	False	False	False
5	False	False	False	False	False
6	False	False	False	False	False
7	False	False	False	False	False
8	False	False	False	False	False
9	False	False	False	False	False
10	False	False	False	False	False
11	False	False	False	False	False
12	False	False	False	False	False
13	False	False	False	False	False
14	False	False	False	False	False
15	False	False	False	False	False
16	False	False	False	True	False
17	False	False	False	False	False
18	False	False	False	False	False
19	False	False	False	False	False
20	False	False	False	False	False
21	False	False	False	False	False
22	False	False	False	False	False
23	False	False	False	False	False



# **Visualizando Banco de Dados Relacional com *DataFrame***

# Introdução

- Até agora aprendemos a fazer manipulação de arquivos em DataFrames no intuito de tornar nossa visualização de dados mais intuitiva e fácil de mexer;
- Sabemos que é possível fazer upload de arquivos como csv, xlsx para melhor manipular e salvar as alterações nesses tipos de arquivos;
- É possível, também, fazer manipulação e visualização de tabelas de banco de dados com DataFrames.



# Configurando ambiente

- Primeiramente, iremos importar as seguintes bibliotecas:

```
In [1]: import numpy as np  
import pandas as pd  
import psycopg2
```

1. **Numpy** para realização de operações estatísticas;
2. **Pandas** para manipular tabelas em DataFrames;
3. **Psycopg2** para conectar o banco de dados com o python.

# Configurando ambiente

- Em seguida, faremos a conexão do banco com o Python através da biblioteca Psycopg2;
  - Explicitar: host, nome do banco, usuário e senha;
  - Explicitar o nível de isolamento.

```
In [2]: conn_str = "host={} dbname={} user={} password={}".format('localhost', 'sefaz', 'postgres', 'admin')
```

```
In [3]: conn_str
```

```
Out[3]: 'host=localhost dbname=sefaz user=postgres password=admin'
```

```
In [4]: conn = psycopg2.connect(conn_str)
```

```
In [5]: # Nível zero de isolamento: READ UNCOMMITTED
# Veja mais na documentação para outros níveis de isolamento: https://www.psycopg.org/docs/connection.html
# Para saber mais sobre nível de isolamento: https://docs.microsoft.com/pt-br/sql/connect/jdbc/understanding-isolation-levels
conn.set_isolation_level(0)
```

<

>



# Importando Tabelas

- Obtendo a tabela desejada do banco e associando ao DataFrame:

```
In [6]: # Iremos colocar a tabela ESTABELECIMENTO dentro do DataFrame
df = pd.read_sql('select * from estabelecimento', con=conn)
```

```
In [7]: # Visualizando tabela no DataFrame
df
```

Out[7]:

	res_id	res_nom_estabelecimento	res_endereco	res_numero	res_complemento	res_cep	res_bairro	res_cidade	res_uf	res_telefone	res_tip_estabelec
0	1	Amika Coffee House	Rua Ana Bilhar	1136	B	60160110	Meireles	Fortaleza	Ce	(85)3031-0351	
1	2	Torra Café	Rua Marcos Macêdo	827		60150190	Aldeota	Fortaleza	Ce		
2	3	Urbici Café	R. Barbosa de Freitas	951	Loja 01	60170021	Aldeota	Fortaleza	Ce		
3	4	Blend Coffee House	R. Sabino Pires			60150090	Aldeota	Fortaleza	Ce	(85) 3121-6455	

# Visão geral da Tabela

- Checando informações gerais do DataFrame:
  - Colunas disponíveis
  - Informações gerais (tipo da coluna, valores nulos)
  - Dimensionalidade do DataFrame (quantidade de linhas e colunas)
- Obter essas informações por DataFrame é mais prático e rápido que por consultas SQL :)

```
In [8]: # Imprimindo as colunas presentes no DataFrame
df.columns
```

```
Out[8]: Index(['res_id', 'res_nom_estabelecimento', 'res_endereco', 'res_numero',
              'res_complemento', 'res_cep', 'res_bairro', 'res_cidade', 'res_uf',
              'res_telefone', 'res_tip_estabelecimento'],
              dtype='object')
```

```
In [9]: # Checando informações gerais da tabela
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4 entries, 0 to 3
Data columns (total 11 columns):
#   Column                                Non-Null Count  Dtype
---  -
0   res_id                                4 non-null      int64
1   res_nom_estabelecimento              4 non-null      object
2   res_endereco                        4 non-null      object
3   res_numero                          4 non-null      object
4   res_complemento                     4 non-null      object
5   res_cep                             4 non-null      object
6   res_bairro                          4 non-null      object
7   res_cidade                          4 non-null      object
8   res_uf                              4 non-null      object
9   res_telefone                        4 non-null      object
10  res_tip_estabelecimento              4 non-null      int64
dtypes: int64(2), object(9)
memory usage: 480.0+ bytes
```

```
In [10]: # Olhando a dimensionalidade do DataFrame (linhas x colunas)
df.shape
```

```
Out[10]: (4, 11)
```

# Consultas via DataFrame

- Consulta por coluna (ou por valor específico):

```
In [11]: # Checando uma coluna específica do DataFrame
df["res_bairro"]
```

```
Out[11]: 0    Meireles
1     Aldeota
2     Aldeota
3     Aldeota
Name: res_bairro, dtype: object
```

```
In [12]: # Checando uma coluna específica do DataFrame com uma visualização melhor :)
df[df["res_bairro"] == "Aldeota"]
```

```
Out[12]:
```

	res_id	res_nom_estabelecimento	res_endereco	res_numero	res_complemento	res_cep	res_bairro	res_cidade	res_uf	res_telefone	res_tip_estabelecim
1	2	Torra Café	Rua Marcos Macêdo	827		60150190	Aldeota	Fortaleza	Ce		
2	3	Urbici Café	R. Barbosa de Freitas	951	Loja 01	60170021	Aldeota	Fortaleza	Ce		
3	4	Blend Coffee House	R. Sabino Pires			60150090	Aldeota	Fortaleza	Ce	(85) 3121-6455	

# Consultas via DataFrame

- Checando valores únicos por coluna:

```
In [13]: # Obtendo os valores unitários de uma coluna específica
df["res_bairro"].unique()

Out[13]: array(['Meireles', 'Aldeota'], dtype=object)
```

- Obtendo o COUNT(\*) através de consulta por DataFrame:

```
In [14]: # Fazendo busca por valor e retornando sua quantidade
# Usamos o index [0] para retornar apenas a quantidade de linhas em que nosso valor aparece
df[df["res_nom_estabelecimento"] == "Amika Coffee House"].shape[0]

Out[14]: 1
```

# Consultas via DataFrame

- Utilizando HEAD e TAIL na tabela (DataFrame):

```
In [15]: # checando as n primeiras instâncias do DataFrame
n = 3
df.head(n)
```

Out[15]:

	res_id	res_nom_estabelecimento	res_endereco	res_numero	res_complemento	res_cep	res_bairro	res_cidade	res_uf	res_telefone	res_tip_estabelec
0	1	Amika Coffee House	Rua Ana Bilhar	1136	B	60160110	Meireles	Fortaleza	Ce	(85)3031-0351	
1	2	Torra Café	Rua Marcos Macêdo	827		60150190	Aldeota	Fortaleza	Ce		
2	3	Urbici Café	R. Barbosa de Freitas	951	Loja 01	60170021	Aldeota	Fortaleza	Ce		

```
In [16]: # checando as n últimas instâncias do DataFrame
n = 2
df.tail(n)
```

Out[16]:

	res_id	res_nom_estabelecimento	res_endereco	res_numero	res_complemento	res_cep	res_bairro	res_cidade	res_uf	res_telefone	res_tip_estabelecir
2	3	Urbici Café	R. Barbosa de Freitas	951	Loja 01	60170021	Aldeota	Fortaleza	Ce		
3	4	Blend Coffee House	R. Sabino Pires			60150090	Aldeota	Fortaleza	Ce	(85) 3121-6455	

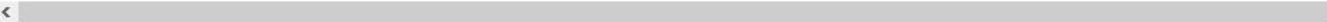
# Consultas via DataFrame

- Busca condicional

```
In [17]: # Efetuando busca condicional
condition = (df["res_bairro"] == "Aldeota") & (df["res_numero"] != "")
df[condition]
```

```
Out[17]:
```

	res_id	res_nom_estabelecimento	res_endereco	res_numero	res_complemento	res_cep	res_bairro	res_cidade	res_uf	res_telefone	res_tip_estabelecir
1	2	Torra Café	Rua Marcos Macêdo	827		60150190	Aldeota	Fortaleza	Ce		
2	3	Urbici Café	R. Barbosa de Freitas	951	Loja 01	60170021	Aldeota	Fortaleza	Ce		

<  >

```
In [18]: # Visualizando apenas as colunas de interesse da busca anterior
columns_of_interest = ["res_nom_estabelecimento", "res_endereco", "res_numero"]
df[condition][columns_of_interest]
```

```
Out[18]:
```

	res_nom_estabelecimento	res_endereco	res_numero
1	Torra Café	Rua Marcos Macêdo	827
2	Urbici Café	R. Barbosa de Freitas	951

```
In [19]: # Checando a dimensionalidade da consulta
# Caso deseje retornar apenas a quantidade de linhas, inserir [0] ao final de shape
df[condition][columns_of_interest].shape
```

```
Out[19]: (2, 3)
```

# Função de Agregação

- **Funções de Agregação:** como já vimos anteriormente neste curso, basta aplicar as funções presentes na biblioteca Numpy sob uma coluna (ou linha) do DataFrame:
  - std: desvio padrão
  - mean: média
  - median: mediana
  - max: valor máximo
  - min: valor mínimo

```
In [20]: # obter desvio padrão de uma coluna  
np.std(df["res_id"])
```

```
Out[20]: 1.118033988749895
```

```
In [21]: # obter média de uma coluna  
np.mean(df["res_id"])
```

```
Out[21]: 2.5
```

```
In [22]: # obter mediana de uma coluna  
np.median(df["res_id"])
```

```
Out[22]: 2.5
```

```
In [23]: # obter o máximo de uma coluna  
np.max(df["res_id"])
```

```
Out[23]: 4
```

```
In [24]: # obter o mínimo de uma coluna  
np.min(df["res_id"])
```

```
Out[24]: 1
```

# Join via DataFrame

- **Junção de Tabelas:** Dada duas (ou mais tabelas), é possível fazer operações como o JOIN por DataFrames; basta fazer a junção de duas tabelas ligando-as através do comando **ON**:

```
In [25]: # Iremos colocar a tabela TIPO_ESTABELECIMENTO dentro do DataFrame
df2 = pd.read_sql('select * from TIPO_ESTABELECIMENTO', con=conn)
```

```
In [26]: df2
```

```
Out[26]:
```

	res_tip_estabelecimento	tipo
0	1	Comercial
1	2	Privado
2	3	Empresarial

```
In [27]: df2.shape
```

```
Out[27]: (3, 2)
```

```
In [28]: # Faremos o JOIN das duas tabelas através da coluna "RES_TIP_ESTABELECIMENTO" e armazenaremos o resultado em um novo DF
df3 = pd.merge(df, df2, on = "res_tip_estabelecimento")
```



# Join via DataFrame

## Resultado:

In [29]: df3

Out[29]:

n_estabelecimento	res_endereco	res_numero	res_complemento	res_cep	res_bairro	res_cidade	res_uf	res_telefone	res_tip_estabelecimento	tipo
umika Coffee House	Rua Ana Bilhar	1136	B	60160110	Meireles	Fortaleza	Ce	(85)3031-0351	1	Comercial
Torra Café	Rua Marcos Macêdo	827		60150190	Aldeota	Fortaleza	Ce		1	Comercial
Urbici Café	R. Barbosa de Freitas	951	Loja 01	60170021	Aldeota	Fortaleza	Ce		2	Privado
Blend Coffee House	R. Sabino Pires			60150090	Aldeota	Fortaleza	Ce	(85) 3121-6455	3	Empresarial

Coluna da  
tabela Tipo de  
Estabelecimento

In [30]: df3.shape

Out[30]: (4, 12)

Coluna responsável por juntar as duas tabelas

# Cursor via DataFrame

- **Cursor:** Utilizaremos o recurso de cursor para fazer inserção e remoção de linhas na tabela:

```
In [31]: # Inicializando um cursor
         cursor = conn.cursor()
         cursor.execute("select * from ESTABELECIMENTO")
         rows = cursor.fetchall()
```

```
In [32]: print(rows)

[(1, 'Amika Coffee House', 'Rua Ana Bilhar', '1136', 'B', '60160110', 'Meireles', 'Fortaleza', 'Ce', '(85)3031-0351', 1),
 (2, 'Torra Café', 'Rua Marcos Macêdo', '827', '', '60150190', 'Aldeota', 'Fortaleza', 'Ce', '', 1), (3, 'Urbici Café', 'R.
Barbosa de Freitas', '951', 'Loja 01', '60170021', 'Aldeota', 'Fortaleza', 'Ce', '', 2), (4, 'Blend Coffee House', 'R. Sabi
no Pires', '', '', '60150090', 'Aldeota', 'Fortaleza', 'Ce', '(85) 3121-6455', 3)]
```

# Cursor via DataFrame

- **Cursor:** Inserindo valores na tabela:

```
In [33]: # Vamos inserir uma nova instância na tabela ESTABELECIMENTO
cursor.execute("INSERT INTO estabelecimento VALUES(5,'Ânimo Café', 'Av Pontes Vieira', '417', '', '60013523', 'São João do Tauape')")
```

```
In [34]: # Verificando inserção
pd.read_sql('select * from estabelecimento', con=conn)
```

Out[34]:

	res_id	res_nom_estabelecimento	res_endereco	res_numero	res_complemento	res_cep	res_bairro	res_cidade	res_uf	res_telefone	res_tip_estabelec
0	1	Amika Coffee House	Rua Ana Bilhar	1136	B	60160110	Meireles	Fortaleza	Ce	(85)3031-0351	
1	2	Torra Café	Rua Marcos Macêdo	827		60150190	Aldeota	Fortaleza	Ce		
2	3	Urbici Café	R. Barbosa de Freitas	951	Loja 01	60170021	Aldeota	Fortaleza	Ce		
3	4	Blend Coffee House	R. Sabino Pires			60150090	Aldeota	Fortaleza	Ce	(85) 3121-6455	
4	5	Ânimo Café	Av Pontes Vieira	417		60013523	São João do Tauape	Fortaleza	Ce		

# Cursor via DataFrame

- **Cursor:** Removendo valores na tabela:

```
In [35]: # Removendo instância
cursor.execute("DELETE FROM estabelecimento WHERE RES_NOM_ESTABELECIMENTO = 'Ânimo Café'")
```

```
In [36]: # Verificando remoção
pd.read_sql('select * from estabelecimento', con=conn)
```

Out[36]:

	res_id	res_nom_estabelecimento	res_endereco	res_numero	res_complemento	res_cep	res_bairro	res_cidade	res_uf	res_telefone	res_tip_estabelec
0	1	Amika Coffee House	Rua Ana Bilhar	1136	B	60160110	Meireles	Fortaleza	Ce	(85)3031-0351	
1	2	Torra Café	Rua Marcos Macêdo	827		60150190	Aldeota	Fortaleza	Ce		
2	3	Urbici Café	R. Barbosa de Freitas	951	Loja 01	60170021	Aldeota	Fortaleza	Ce		
3	4	Blend Coffee House	R. Sabino Pires			60150090	Aldeota	Fortaleza	Ce	(85) 3121-6455	

# ***Commit()* e *Close()***

- **Cursor:** *commitando* alterações e encerrando a conexão com o banco

```
In [37]: conn.commit()
```

```
In [38]: conn.close()
```

# Motivação

- Com o crescente volume de informações na web, a extração de dados manuais pode se tornar uma tarefa inviável (em alguns casos impossível).



# Estratégias

**Web Scraping** é uma estratégia de baixar automaticamente os dados de uma página web. Os dados baixados são geralmente armazenados em um índice ou banco de dados para facilitar sua busca.



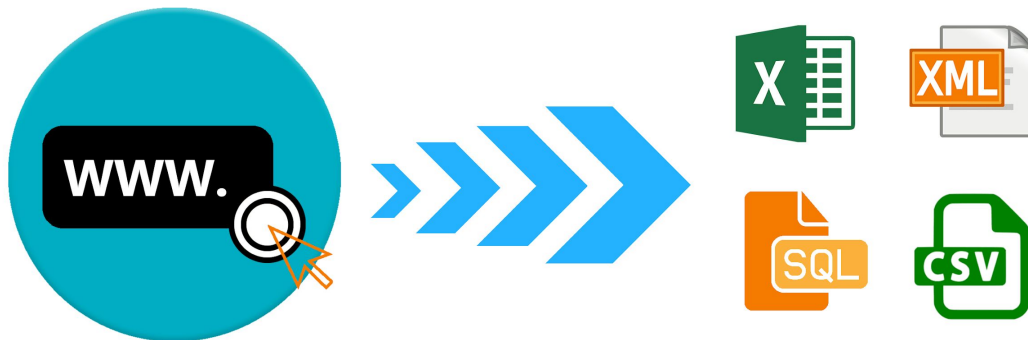
# **Extração de Dados**



# Estratégias

As estratégias variam de acordo com a necessidade da extração de dados:

- **Web Crawler:** retirar informações a partir de links URL.
  - Outros termos para *web crawler*: *bots*, *web spiders*, *web robot* e *web scutter*.
- **Scraping:** retirar informações a partir de arquivos específicos.



# Crawler: Web

- Uma das maneiras de extração mais usual é fazer um *crawler* de dados da Web de forma automatizada;
- Para isso, iremos ilustrar o seguinte cenário:  
Desejamos gerar um arquivo de saída (*json*, *csv*,...) retirando todas as citações, nome dos autores e tags do site <http://quotes.toscrape.com/>

## Quotes to Scrape

Login

*"The world as we have created it is a process of our thinking. It cannot be changed without changing our thinking."*

by [Albert Einstein](#) (about)

Tags: [change](#) [deep-thoughts](#) [thinking](#) [world](#)

*"It is our choices, Harry, that show what we truly are, far more than our abilities."*

by [J.K. Rowling](#) (about)

Tags: [abilities](#) [choices](#)

## Top Ten tags

love

inspirational

life

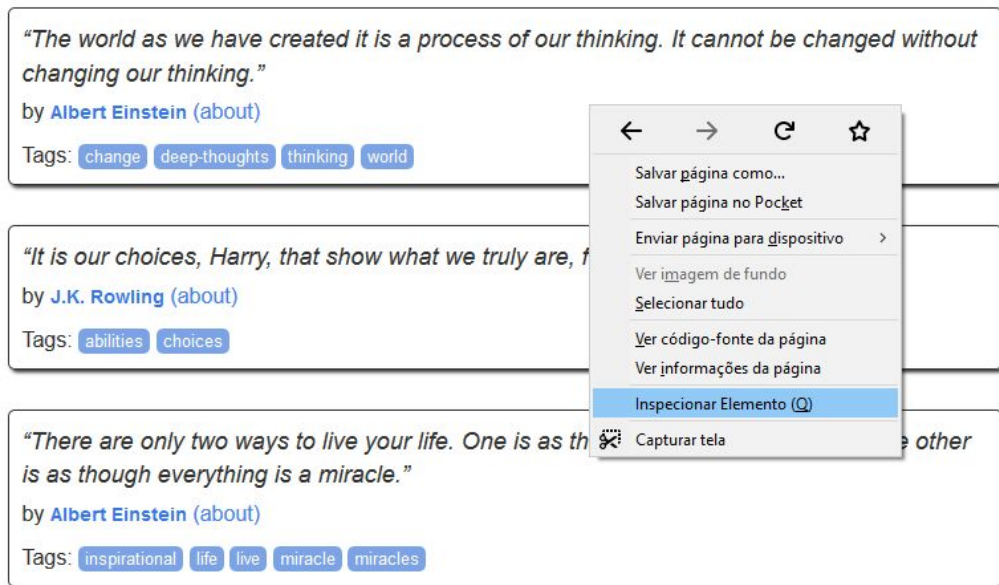
humor

books

reading

# Crawler: Web

Inicialmente, vamos dar uma rápida olhada em como a informação está estruturada. Ao entrar no site, clique com o botão direito na página e vá em Inspecionar Elemento



# Crawler: Web

Será exibido um código contendo informações estruturadas em HTML. Observe que o texto que desejamos está contido em DIVs e SPANs:

```
▼ <div class="quote" itemscope="" itemtype="http://schema.org/CreativeWork">
  ▼ <span class="text" itemprop="text">
    "The world as we have created it is a process of our thinking. It cannot be changed without changing our thinking."
  </span>
  ▼ <span>
    by
    <small class="author" itemprop="author">Albert Einstein</small>
    whitespace
    <a href="/author/Albert-Einstein">(about)</a>
  </span>
  ▼ <div class="tags">
    Tags:
    <meta class="keywords" itemprop="keywords" content="change,deep-thoughts,thinking,world">
    <a class="tag" href="/tag/change/page/1/">change</a>
    whitespace
    <a class="tag" href="/tag/deep-thoughts/page/1/">deep-thoughts</a>
    whitespace
    <a class="tag" href="/tag/thinking/page/1/">thinking</a>
    whitespace
    <a class="tag" href="/tag/world/page/1/">world</a>
  </div>
</div>
```



**<div>** é utilizada para criar uma divisão ou uma seção em um documento HTML.

**<span>** geralmente é usada para agrupar elementos em linha em um documento.

# Crawler: Web

Estamos interessados em obter os conteúdos que estão dentro dos retângulos vermelhos:

```
<div class="quote" itemscope="" itemtype="http://schema.org/CreativeWork">
```

← **<div> principal**

```
<span class="text" itemprop="text">
```

```
"The world as we have created it is a process of our thinking. It cannot be changed without changing our thinking."
```

```
</span>
```

```
<span>
```

```
by
```

```
<small class="author" itemprop="author">Albert Einstein</small>
```

```
 whitespace
```

```
<a href="/author/Albert-Einstein">(about)</a>
```

```
</span>
```

```
<div class="tags">
```

```
Tags:
```

```
<meta class="keywords" itemprop="keywords" content="change,deep-thoughts,thinking,world">
```

```
<a class="tag" href="/tag/change/page/1/">change</a>
```

```
 whitespace
```

```
<a class="tag" href="/tag/deep-thoughts/page/1/">deep-thoughts</a>
```

```
 whitespace
```

```
<a class="tag" href="/tag/thinking/page/1/">thinking</a>
```

```
 whitespace
```

```
<a class="tag" href="/tag/world/page/1/">world</a>
```

```
</div>
```

```
</div>
```

"The world as we have created it is a process of our thinking. It cannot be changed without changing our thinking."

by [Albert Einstein](#) (about)

Tags: [change](#) [deep-thoughts](#) [thinking](#) [world](#)

# Crawler: Web - Extraíndo os dados no Jupyter

- Usaremos a biblioteca Scrapy;
  - <https://docs.scrapy.org/en/latest/intro/tutorial.html>
- A documentação disponibiliza um framework que já resolve bastante coisa;
- Os códigos que utilizaremos aqui foram todos retirados da documentação, precisando apenas de uma adaptação pequena ou outra.

 Scrapy

latest

# Crawler: Web - Extraíndo os dados no Jupyter

- Primeiro, vamos configurar o ambiente:

```
In [1]: from IPython.core.interactiveshell import InteractiveShell
InteractiveShell.ast_node_interactivity = "all"
# Exibir versão do Python
import platform
platform.python_version()
```

Out[1]: '3.7.6'

```
In [2]: try: # Checando se Scrapy está instalado
import scrapy
except:
    !pip install scrapy
import scrapy
from scrapy.crawler import CrawlerProcess
from scrapy.utils.project import get_project_settings
```

# Crawler: Web

- Vamos definir o **Parser** que será utilizado para extrair as informações desejadas:

```
In [3]: class QuotesSpider(scrapy.Spider):
        name = "quotes"
        # URLs
        start_urls = [
            'http://quotes.toscrape.com/page/1/'

        ]

        # Parse da página principal a ser crawleada
        def parse(self, response):

            for quote in response.css('div.quote'):
                yield {
                    'text': quote.css('span.text::text').extract()[0],
                    'author': quote.css('span small::text').extract()[0],
                    'tags': quote.css('div.tags a.tag::text').extract()
                }
```

URL DESEJADA

div principal que contém os campos que desejamos extrair as informações

::text extrai apenas o conteúdo textual dos campos



# Crawler: Web

- Por fim, daremos *start* na função:

```
In [4]: process = CrawlerProcess(get_project_settings())  
  
# Iniciando processo  
process.crawl(QuotesSpider)  
process.start()
```

← Instanciar o processo de Crawler;

↑  
classe definida anteriormente com o parser

# Crawler: Web

- Repare que a saída do *crawler* é bem “poluída” de *warnings*.

```
process = CrawlerProcess(get_project_settings())

# Iniciando processo
process.crawl(QuotesSpider)
process.start()

2020-07-28 09:39:13 [scrapy.utils.log] INFO: Scrapy 2.2.1 started (bot: scrapybot)
2020-07-28 09:39:13 [scrapy.utils.log] INFO: Versions: lxml 4.5.0.0, libxml2 2.9.9, cssselect 1.1.0, parsel 1.6.0, w3lib 1.
22.0, Twisted 20.3.0, Python 3.7.6 (default, Jan 8 2020, 20:23:39) [MSC v.1916 64 bit (AMD64)], pyOpenSSL 19.1.0 (OpenSSL
1.1.1d 10 Sep 2019), cryptography 2.8, Platform Windows-10-10.0.18362-SP0
2020-07-28 09:39:13 [scrapy.utils.log] DEBUG: Using reactor: twisted.internet.selectreactor.SelectReactor
2020-07-28 09:39:13 [scrapy.crawler] INFO: Overridden settings:
{}
2020-07-28 09:39:13 [scrapy.extensions.telnet] INFO: Telnet Password: ebd5d5cc8854e398
2020-07-28 09:39:13 [scrapy.middleware] INFO: Enabled extensions:
['scrapy.extensions.corestats.CoreStats',
 'scrapy.extensions.telnet.TelnetConsole',
 'scrapy.extensions.logstats.LogStats']
2020-07-28 09:39:13 [scrapy.middleware] INFO: Enabled downloader middlewares:
['scrapy.downloadermiddlewares.httppath.HttpAuthMiddleware',
 'scrapy.downloadermiddlewares.downloadtimeout.DownloadTimeoutMiddleware',
 'scrapy.downloadermiddlewares.defaultheaders.DefaultHeadersMiddleware',
 'scrapy.downloadermiddlewares.useragent.UserAgentMiddleware',
 'scrapy.downloadermiddlewares.retry.RetryMiddleware',
 'scrapy.downloadermiddlewares.redirect.MetaRefreshMiddleware',
 'scrapy.downloadermiddlewares.httpcompression.HttpCompressionMiddleware',
 'scrapy.downloadermiddlewares.redirect.RedirectMiddleware',
 'scrapy.downloadermiddlewares.cookies.CookiesMiddleware',
 'scrapy.downloadermiddlewares.httpproxy.HttpProxyMiddleware',
 'scrapy.downloadermiddlewares.stats.DownloaderStats']
2020-07-28 09:39:13 [scrapy.middleware] INFO: Enabled spider middlewares:
['scrapy.spidermiddlewares.httperror.HttpErrorMiddleware',
 'scrapy.spidermiddlewares.offsite.OffsiteMiddleware',
 'scrapy.spidermiddlewares.referrer.RefererMiddleware',
 'scrapy.spidermiddlewares.urllength.UrlLengthMiddleware',
 'scrapy.spidermiddlewares.depth.DepthMiddleware']
2020-07-28 09:39:13 [scrapy.middleware] INFO: Enabled item pipelines:
[]
2020-07-28 09:39:13 [scrapy.core.engine] INFO: Spider opened
2020-07-28 09:39:13 [scrapy.extensions.logstats] INFO: Crawled 0 pages (at 0 pages/min), scraped 0 items (at 0 items/min)
2020-07-28 09:39:13 [scrapy.extensions.telnet] INFO: Telnet console listening on 127.0.0.1:6023
```

# Crawler: Web

- A saída das informações *crawleadas* também não está em um bom formato visual:

```
Out[4]: <Deferred at 0x1f47765d208>
```

```
2020-07-28 09:39:14 [scrapy.core.engine] DEBUG: Crawled (200) <GET http://quotes.toscrape.com/page/1/> (referer: None)
2020-07-28 09:39:14 [scrapy.core.scrapers] DEBUG: Scraped from <200 http://quotes.toscrape.com/page/1/>
{'text': '"The world as we have created it is a process of our thinking. It cannot be changed without changing our thinking."', 'author': 'Albert Einstein', 'tags': ['change', 'deep-thoughts', 'thinking', 'world']}
2020-07-28 09:39:14 [scrapy.core.scrapers] DEBUG: Scraped from <200 http://quotes.toscrape.com/page/1/>
{'text': '"It is our choices, Harry, that show what we truly are, far more than our abilities."', 'author': 'J.K. Rowling', 'tags': ['abilities', 'choices']}
2020-07-28 09:39:14 [scrapy.core.scrapers] DEBUG: Scraped from <200 http://quotes.toscrape.com/page/1/>
{'text': '"There are only two ways to live your life. One is as though nothing is a miracle. The other is as though everything is a miracle."', 'author': 'Albert Einstein', 'tags': ['inspirational', 'life', 'live', 'miracle', 'miracles']}
2020-07-28 09:39:14 [scrapy.core.scrapers] DEBUG: Scraped from <200 http://quotes.toscrape.com/page/1/>
{'text': '"The person, be it gentleman or lady, who has not pleasure in a good novel, must be intolerably stupid."', 'author': 'Jane Austen', 'tags': ['aliteracy', 'books', 'classic', 'humor']}
2020-07-28 09:39:14 [scrapy.core.scrapers] DEBUG: Scraped from <200 http://quotes.toscrape.com/page/1/>
{'text': '"Imperfection is beauty, madness is genius and it's better to be absolutely ridiculous than absolutely boring."', 'author': 'Marilyn Monroe', 'tags': ['be-yourself', 'inspirational']}
2020-07-28 09:39:14 [scrapy.core.scrapers] DEBUG: Scraped from <200 http://quotes.toscrape.com/page/1/>
{'text': '"Try not to become a man of success. Rather become a man of value."', 'author': 'Albert Einstein', 'tags': ['adulthood', 'success', 'value']}
2020-07-28 09:39:14 [scrapy.core.scrapers] DEBUG: Scraped from <200 http://quotes.toscrape.com/page/1/>
{'text': '"It is better to be hated for what you are than to be loved for what you are not."', 'author': 'André Gide', 'tags': ['life', 'love']}
2020-07-28 09:39:14 [scrapy.core.scrapers] DEBUG: Scraped from <200 http://quotes.toscrape.com/page/1/>
{'text': '"I have not failed. I've just found 10,000 ways that won't work."', 'author': 'Thomas A. Edison', 'tags': ['edison', 'failure', 'inspirational', 'paraphrased']}
2020-07-28 09:39:14 [scrapy.core.scrapers] DEBUG: Scraped from <200 http://quotes.toscrape.com/page/1/>
{'text': '"A woman is like a tea bag; you never know how strong it is until it's in hot water."', 'author': 'Eleanor Roosevelt', 'tags': ['misattributed-eleanor-roosevelt']}
2020-07-28 09:39:14 [scrapy.core.scrapers] DEBUG: Scraped from <200 http://quotes.toscrape.com/page/1/>
{'text': '"A day without sunshine is like, you know, night."', 'author': 'Steve Martin', 'tags': ['humor', 'obvious', 'simile']}
2020-07-28 09:39:14 [scrapy.core.engine] INFO: Closing spider (finished)
2020-07-28 09:39:14 [scrapy.statscollectors] INFO: Dumping Scrapy stats:
```

# Crawler: Web

- Para isso, vamos gerar um *output* para que possamos trabalhar com eles via *DataFrame*.
- Seja a seguinte classe:

```
In [2]: import json

class JsonWriterPipeline(object):

    # Função para gerar/abrir arquivo JSON
    def open_spider(self, spider):
        self.file = open('quoteresult.json', 'w')

    # Fechar arquivo após escrita
    def close_spider(self, spider):
        self.file.close()

    # Inserir itens coletados da página WEB no arquivo JSON criado
    def process_item(self, item, spider):
        line = json.dumps(dict(item)) + "\n"
        self.file.write(line)
        return item
```

Toda essa classe pode ser obtida através da Documentação<sup>1</sup> do scrapy para geração de arquivos de saída.

# Crawler: Web

In [3]: `import logging`

```
class QuotesSpider(scrapy.Spider):
    name = "quotes"
    # URLs
    start_urls = [
        'http://quotes.toscrape.com/page/1/',
        'http://quotes.toscrape.com/page/2/',
    ]

    # Configuração obrigatória de pipeline para geração de arquivo de saída
    custom_settings = {
        'LOG_LEVEL': logging.WARNING,
        'ITEM_PIPELINES': {'__main__.JsonWriterPipeline': 1},
        'FEED_FORMAT': 'json',
        'FEED_URI': 'quoteresult.json'
    }

    # Parse da página principal a ser crawleada
    def parse(self, response):
        for quote in response.css('div.quote'):
            yield {
                'text': quote.css('span.text::text').extract()[0],
                'author': quote.css('span small::text').extract()[0],
                'tags': quote.css('div.tags a.tag::text').extract()
            }
```

Adicionamos esses trechos no código para que a função de geração de *output* seja efetuada.

# Crawler: Web

Dê `start()` novamente. Observe que a quantidade de *Warnings* agora diminuiu consideravelmente :)

```
In [4]: process = CrawlerProcess(get_project_settings())
```

```
# Iniciando processo
process.crawl(QuotesSpider)
process.start()
```

```
2020-07-28 10:55:25 [scrapy.utils.log] INFO: Scrapy 2.2.1 started (bot: scrapybot)
2020-07-28 10:55:25 [scrapy.utils.log] INFO: Versions: lxml 4.5.0.0, libxml2 2.9.9, cssselect 1.1.0, parsel 1.6.0, w3lib 1.
22.0, Twisted 20.3.0, Python 3.7.6 (default, Jan 8 2020, 20:23:39) [MSC v.1916 64 bit (AMD64)], pyOpenSSL 19.1.0 (OpenSSL
1.1.1d 10 Sep 2019), cryptography 2.8, Platform Windows-10-10.0.18362-SP0
2020-07-28 10:55:25 [scrapy.utils.log] DEBUG: Using reactor: twisted.internet.selectreactor.SelectReactor
2020-07-28 10:55:25 [scrapy.crawler] INFO: Overridden settings:
{'LOG_LEVEL': 30}
2020-07-28 10:55:25 [py.warnings] WARNING: C:\ProgramData\Anaconda3\lib\site-packages\scrapy\extensions\feedexport.py:210:
ScrapyDeprecationWarning: The `FEED_URI` and `FEED_FORMAT` settings have been deprecated in favor of the `FEEDS` setting. P
lease see the `FEEDS` setting docs for more details
    exporter = cls(crawler)
```

```
Out[4]: <Deferred at 0x1d7e7fddbc8>
```



# Crawler: Web

Finalmente, vamos abrir o *output* gerado em um DataFrame:

**lines** : *bool, default False*

Read the file as a json object per line.

```
In [5]: import pandas as pd
# Carregando JSON criado para visualizar saída
output = pd.read_json('quoteresult.json', lines=True)
output
```

Out[5]:

	text	author	tags
0	"The world as we have created it is a process ...	Albert Einstein	[change, deep-thoughts, thinking, world]
1	"This life is what you make it. No matter what...	Marilyn Monroe	[friends, heartbreak, inspirational, life, lov...
2	"It takes a great deal of bravery to stand up ...	J.K. Rowling	[courage, friends]
3	"If you can't explain it to a six year old, yo...	Albert Einstein	[simplicity, understand]
4	"You may not be her first, her last, or her on...	Bob Marley	[love]
5	"I like nonsense, it wakes up the brain cells....	Dr. Seuss	[fantasy]
6	"I may not have gone where I intended to go, b...	Douglas Adams	[life, navigation]

# Crawler: PDF

- Em alguns cenários podemos encontrar o armazenamento de dados em forma de PDF.
  - É comum em domínios como órgãos públicos que possuem dados antigos que estão sendo digitalizados;
- Armazenamento de dados em PDF é um dos meios mais trabalhosos de lidar com manipulação de informação, visto que ele não possui uma estrutura de armazenamento (como dados armazenados em *html*, *javascript*) e, por isso, a extração dessa informação deve ser feita de forma manual.



# Crawler: PDF - Biblioteca para manipulação de PDF

- Antes de extrairmos os dados do PDF para manipulá-lo, talvez seja necessário manipular o arquivo em si, como por exemplo:
  - Retirar do PDF somente as páginas de interesse: isso acontece quando você vai trabalhar apenas com um subconjunto do arquivo;
  - Juntar vários arquivos PDF em um só: quando você lida com vários arquivos e resolve juntar todos em um para que a leitura dos dados seja feita de forma única.

# Crawler: PDF - Biblioteca para manipulação de PDF

- Uma indicação de biblioteca para manipulação de arquivo em PDF é a PyPDF4;
  - <https://pypi.org/project/PyPDF4/#files>
- Como exemplo, vamos pegar o PDF da Lei 7799 do estado do Maranhão.

```
In [1]: import os  
        from PyPDF4 import PdfFileReader, PdfFileWriter
```

```
In [2]: input1 = PdfFileReader("lei7799.pdf", "rb")
```

```
In [3]: input1.getNumPages()
```

```
Out[3]: 227
```

Este documento é um PDF que contém toda a lei 7799, totalizando 227 páginas



# Dúvidas?

Email: [wellington@crateus.ufc.br](mailto:wellington@crateus.ufc.br)



UNIVERSIDADE  
FEDERAL DO CEARÁ  
CAMPUS DE CRATEÚS

