03

Lendo nomes com Json

Transcrição

Nos treinamentos anteriores, geralmente disponibilizávamos um arquivo `.csv` para realizarmos nossas análises. Porém, a biblioteca Pandas também nos permite importar e exportar diferentes tipos de arquivos, que é o que faremos nesse treinamento.

Começaremos analisando as informações de uma escola de programação que possui muitos alunos e alunas em diferentes cursos. Abriremos o Google Colaboratory (https://colab.research.google.com/) e criaremos um novo notebook Python 3, que renomearemos para `Pandas IO.ipynb`.

> Na atividade anterior, "Preparando o ambiente", temos um passo a passo do que é necessário para utilizar o Colaboratory, uma ferramenta disponibilidade pelo Google para programação em Python.

Vamos buscar os nomes femininos e masculinos com os seguintes links:

https://guilhermeonrails.github.io/nomes_ibge/nomes-f.json (https://guilhermeonrails.github.io/nomes_ibge/nomes-f.json)

https://guilhermeonrails.github.io/nomes_ibge/nomes-m.json (https://guilhermeonrails.github.io/nomes_ibge/nomes-m.json)

Voltando ao Colaboratory, importaremos o Pandas em nosso projeto com o conhecido comando `import pandas as pd` (utilizando o apelido convencional `pd`).

```
import pandas as pd
```

COPIAR CÓDIGO

Agora gostaríamos de ler a API contendo os nomes masculinos e femininos. Para isso, chamaremos a função `pd.read_json()` - ou seja, uma função que lê arquivos JSON. Como parâmetro, entre aspas duplas, passaremos a URL dos nomes de sexo feminino.

```
pd.read_json('https://guilhermeonrails.github.io/nomes_ibge/nor
```

COPIAR CÓDIGO

Ao executarmos, receberemos uma tabela contendo os 20 nomes da nossa consulta - ou seja, do índice `0` ao `19`.

	nome	regiao	freq	rank	sexo
0	MARIA	0	11694738	1	F
1	ANA	0	3079729	2	F
2	FRANCISCA	0	721637	3	F
3	ANTONIA	0	588783	4	F
4	ADRIANA	0	565621	5	F
5	JULIANA	0	562589	6	F
6	MARCIA	0	551855	7	F
7	FERNANDA	0	531607	8	F
8	PATRICIA	0	529446	9	F
9	ALINE	0	509869	10	F
10	SANDRA	0	479230	11	F
11	CAMILA	0	469851	12	F
12	AMANDA	0	464624	13	F
13	BRUNA	0	460770	14	F
14	JESSICA	0	456472	15	F
15	LETICIA	0	434056	16	F
16	JULIA	0	430067	17	F
17	LUCIANA	0	429769	18	F
18	VANESSA	0	417512	19	F
19	MARIANA	0	381778	20	F

Vamos armazenar esse resultado em uma variável? Vou chamá-la de `nomes_f` :

```
nomes_f = pd.read_json('https://guilhermeonrails.github.io/nome
```

COPIAR CÓDIGO

Vamos criar uma variável chamada `nomes_m` para armazenar os nomes masculinos também:

```
nomes_f = pd.read_json('https://guilhermeonrails.github.io/nome
nomes_m = pd.read_json('https://guilhermeonrails.github.io/nome
```

COPIAR CÓDIGO

Podemos sortear nomes através da função `sample` :

```
nomes_m.sample(5)
```

COPIAR CÓDIGO

	nome	regiao	freq	rank	sexo
149	ISRAEL	0	11694738	1	F
117	WAGNER	0	3079729	2	F
45	ADRIANO	0	721637	3	F
120	EDILSON	0	588783	4	F
22	LEANDRO	0	565621	5	F

Podemos identificar a quantidade de nomes em nossos conjuntos imprimindo (`print()`) o tamanho de cada um deles. Para isso, concatenaremos a string "Quantidade de nomes: " com a soma `len(nomes_f) + len(nomes_m))` .

```
print("Quantidade de nomes: " + len(nomes_f) + len(nomes_m))
```

COPIAR CÓDIGO

A execução dessa instrução resultará em um erro, afinal o retorno da soma é um número inteiro, e não uma string que possa ser concatenada com nosso

texto. Corrigiremos isso utilizando a função conversora `str()`.

```
print("Quantidade de nomes: " + str(len(nomes_f) + len(nomes_m
```

COPIAR CÓDIGO

Quantidade de nomes: 400

Agora queremos juntar esses dois conjuntos em um único dataframe contendo os nomes de todos os alunos e alunas da escola. Para isso, criaremos uma variável `frames` que receberá uma lista contendo nossas informações.

```
frames = [nomes_f, nomes_m]
```

COPIAR CÓDIGO

Em seguida, poderíamos usar a função `pd.concat()` do Pandas para concatenarmos os dados em um único dataframe. Entretanto, se observamos nossa lista `frames`, veremos que existem diversas colunas desnecessárias para o projeto, como `freq` (a frequência do nome no país), nem o `rank` (de *ranking*), a região ou o sexo.

Na função `pd.concat()`, é possível especificar quais colunas queremos concatenar, nesse caso apenas a `nome`. Ao final, usaremos a função `to_frame()` para transformarmos o conjunto resultante em um dataframe.

```
pd.concat(frames)['nome'].to_frame()
```

COPIAR CÓDIGO

Como resultado, teremos:

	nome
0	MARIA
1	ANA
2	FRANCISCA
3	ANTONIA
4	ADRIANA
...	...
195	WALTER
196	ARLINDO
197	MICHAEL
198	ALVARO
199	GEOVANE

Armazenaremos esse retorno em uma variável `nomes`:

```
nomes = pd.concat(frames)['nome'].to_frame()
```

COPIAR CÓDIGO

Com a função `sample()`, podemos conseguir um determinado número de amostras aleatórias dentro deste dataframe. Nesse caso, pediremos cinco:

```
nomes.sample(5)
```

COPIAR CÓDIGO

	nome
30	ANDERSON
5	PAULO
31	RICARDO
85	FABRICIO
17	FELIPE

Executando novamente, teremos outro resultado.

	nome
156	VALDEMAR
178	DENILSON
13	BRUNA
187	NAIARA
82	SOLANGE

Com o comando `pd.read_json()`, criamos um dataframe com 400 nomes, entre eles 200 femininos e 200 masculinos, que foram reunidos utilizando a função `concat()`.