Saída em diferentes formatos

Transcrição

Nós criamos vários dataframes diferentes por meio de leituras possíveis do Pandas, como a função `read_json()`, que utilizamos para buscar os alunos, e o `read_html()`, com a qual conseguimos os cursos. A partir disso, juntamos tabelas e criamos dataframes auxiliares que nos trouxeram mais informações, por exemplo `matriculas_por_curso`, que mostra a quantidade de alunos matriculados em cada curso, ou `matriculas`, mostrando que alunos estão inscritos em quais cursos.

Nosso objetivo agora é exportar o dataframe `matriculas_por_curso` para um novo formato, como o CSV, um dos formatos mais comuns em análise de dados. Para iniciarmos essa nova etapa do projeto, criaremos a seção "Saída em diferentes formatos".

A exportação de um dataframe para um arquivo CSV pode ser feita por meio da função `to_csv()`, passando como parâmetro, entre aspas simples, o nome que será atribuído ao arquivo - nesse caso, `matriculas_por_curso.csv`. Além dele, passaremos o parâmetro `index=False` para determinarmos que o índice da tabela não será exportado.

```
matriculas_por_curso.to_csv('matriculas_por_curso.csv', index=False
```

COPIAR CÓDIGO

Ao executarmos, teremos a impressão de que nada aconteceu. Entretanto, o arquivo `matriculas_por_curso.csv` estará disponível para download na aba "Files" à esquerda da tela. Como aprendemos nos outros cursos de Pandas aqui da plataforma, tal arquivo pode ser lido a partir da função `pd.read_csv()`.

```
pd.read_csv('matriculas_por_curso.csv')
```

COPIAR CÓDIGO

	quantidade_de_alunos	nome_do_curso
0	14	Lógica de programação
1	2	Java para Web
2	47	C# para Web
3	34	Ruby on Rails
4	1	Cursos de Python

Começamos esse projeto lendo um arquivo JSON, e também é possível criarmos um arquivo desse tipo. Para isso, criaremos a variável `matriculas_json` recebendo a chamada de `matriculas_por_curso.to_json()`. Ao exibirmos o conteúdo da nova variável, veremos nossos dados no formato escolhido.

```
matriculas_json = matriculas_por_curso.to_json()
matriculas_json
```

COPIAR CÓDIGO

{"quantidade_de_alunos":{"1":14,"2":2,"3":47,"4":34,"5":1,"6":81,"7":28, ...},"nome_do_curso":{"1":"L\u00f3gica de programa\u00e7\u00e3o","2":"Java para Web","3":"C# para Web", ...}}

Também lemos uma tabela a partir de uma página HTML usando a função `read_html()`. Para transformarmos o dataframe `matriculas_por_curso` em HTML, usaremos a função `to_html`. Armazenaremos o seu retorno em uma nova variável `matriculas_html`.

```
matriculas_html = matriculas_por_curso.to_html()
matriculas_html
```

COPIAR CÓDIGO

Será trabalhoso lermos o conteúdo dessa variável (uma `<table>`), pois ele será exibido em uma única linha. Uma maneira de melhorarmos essa exibição é imprimirmos o conteúdo na tela com o comando `print()`.

```
print(matriculas_html)
```

COPIAR CÓDIGO

Como retorno, teremos uma tabela formatada como um código HTML padrão.

```
<table border="1" class="dataframe">
  <thead>
    <tr style="text-align: right;">
      <th></th>
      <th>quantidade_de_alunos</th>
      <th>nome_do_curso</th>
    </tr>
    <tr>
      <th>id_curso</th>
      <th></th>
      <th></th>
    </tr>
  </thead>
  <tbody>
    <tr>
      <th>1</th>
      <td>58</td>
      <td>Lógica de programação</td>
    </tr>
    <tr>
      <th>2</th>
      <td>31</td>
      <td>Java para Web</td>
    </tr>
    <tr>
      <th>3</th>
      <td>53</td>
      <td>C# para Web</td>
    </tr>
    <tr>
      <th>4</th>
      <td>4</td>
```

```
  <td>Ruby on Rails</td>
</tr>
<tr>
  <th>5</th>
  <td>12</td>
  <td>Cursos de Python</td>
</tr>
<tr>
  <th>6</th>
  <td>74</td>
  <td>PHP com MySql</td>
</tr>
<tr>
  <th>7</th>
  <td>32</td>
  <td>.NET para web</td>
</tr>
<tr>
  <th>8</th>
  <td>18</td>
  <td>Novas integrações com Java</td>
</tr>
<tr>
  <th>9</th>
  <td>59</td>
  <td>TDD com Java</td>
</tr>
<tr>
  <th>10</th>
  <td>39</td>
  <td>Código limpo com C#</td>
</tr>
<tr>
  <th>11</th>
  <td>61</td>
  <td>Preparatório para certificação Java</td>
</tr>
<tr>
```

```
    <th>12</th>
    <td>65</td>
    <td>Hardware básico</td>
  </tr>
  <tr>
    <th>13</th>
    <td>14</td>
    <td>Persistência com .NET</td>
  </tr>
  <tr>
    <th>14</th>
    <td>27</td>
    <td>Desenvolvendo jogos</td>
  </tr>
  <tr>
    <th>15</th>
    <td>46</td>
    <td>Análise de dados</td>
  </tr>
  <tr>
    <th>16</th>
    <td>43</td>
    <td>Estatística básica</td>
  </tr>
  <tr>
    <th>17</th>
    <td>56</td>
    <td>Internet das coisas</td>
  </tr>
  <tr>
    <th>18</th>
    <td>53</td>
    <td>Programação funcional</td>
  </tr>
  <tr>
    <th>19</th>
    <td>74</td>
    <td>Boas práticas em Java</td>
```

```
    </tr>
    <tr>
      <th>20</th>
      <td>35</td>
      <td>Orientação objetos com Java</td>
    </tr>
  </tbody>
</table>
```

COPIAR CÓDIGO

Para visualizarmos esse conteúdo em uma página, abriremos um editor de texto (como o Visual Studio Code), colaremos todo o código e salvaremos o arquivo como `matriculas.html` . Ao abrirmos o arquivo em um navegador, como o Chrome ou o Firefox, a tabela será exibida, com todos os cursos e quantidade de alunos.

	quantidade_de_alunos	nome_do_curso
id_curso		
1	58	Lógica de programação
2	31	Java para Web
3	53	C# para Web
4	4	Ruby on Rails
5	12	Cursos de Python
6	74	PHP com MySql
7	32	.NET para web
8	18	Novas integrações com Java
9	59	TDD com Java
10	39	Código limpo com C#
11	61	Preparatório para certificação Java
12	65	Hardware básico
13	14	Persistência com .NET
14	27	Desenvolvendo jogos
15	46	Análise de dados
16	43	Estatística básica
17	56	Internet das coisas
18	53	Programação funcional
19	74	Boas práticas em Java
20	35	Orientação objetos com Java

Exemplo de tabela em HTML (https://caelum-online-public.s3.amazonaws.com/1077-pandasIO/Transcri%C3%A7%C3%A3o/imagens/matriculas.html).

Com isso, aprendemos a exportar um dataframe para três formatos diferentes: CSV, JSON e HTML!