

# Engenharia de Dados – Parte 04

**Wellington Franco**  
**Universidade Federal do Ceará – UFC**  
**Campus da UFC em Crateús**  
[wellington@crateus.ufc.br](mailto:wellington@crateus.ufc.br)

# Extração de Dados

# 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

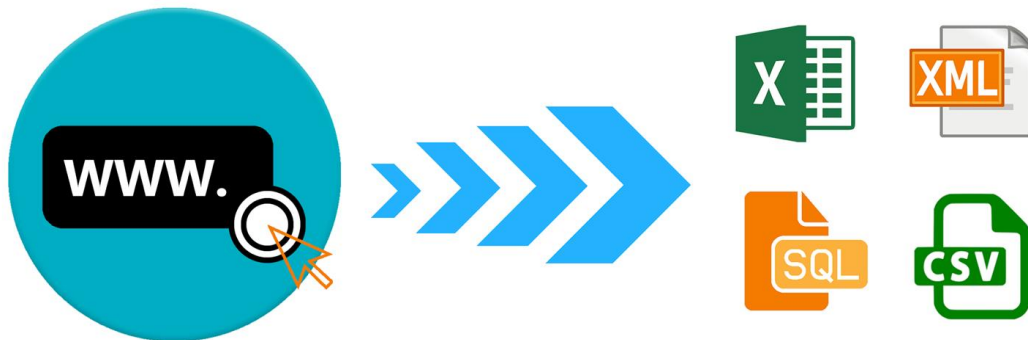
**Web Scraping** é uma estratégia de baixar automaticamente os dados de uma página web. Os dados baixados são geralmente armazenados em um índice ou banco de dados para facilitar sua busca.



# 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

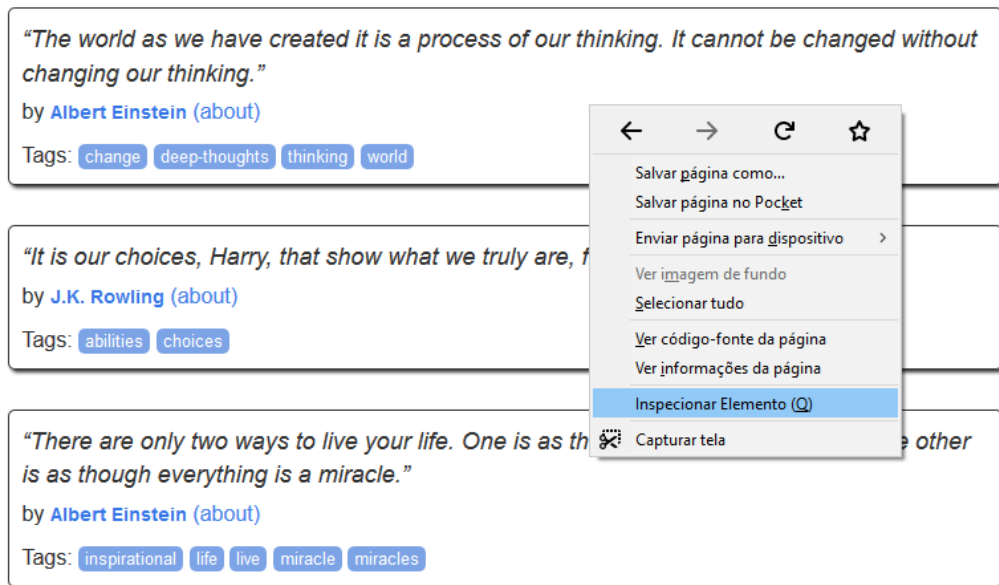
humor

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 Inspeccionar 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>
    <small>whitespace</small>
    <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>
    <small>whitespace</small>
    <a class="tag" href="/tag/deep-thoughts/page/1/">deep-thoughts</a>
    <small>whitespace</small>
    <a class="tag" href="/tag/thinking/page/1/">thinking</a>
    <small>whitespace</small>
    <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 - Extraindo 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.httppauth.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.downloadermiddlewareshttpproxy.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 thinkin
g."', '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 everyth
ing 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."', 'autho
r': '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': ['adul
thood', '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', 'tag
s': ['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': ['ediso
n', '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 Roosev
elt', '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', 'simi
le']}
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()
```

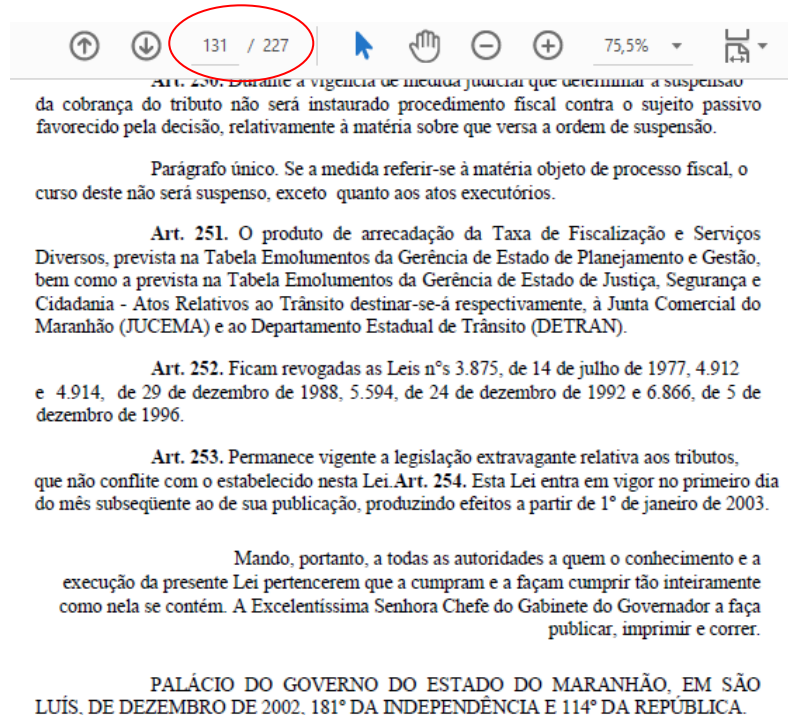
```
Out[3]: 227
```

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



# Crawler: PDF - Biblioteca para manipulação de PDF

- Se dermos uma rápida olhada no PDF original, veremos que só há conteúdo textual até a página 131.



# Crawler: PDF - Biblioteca para manipulação de PDF

- Da página 132 em diante, só haverá a seção Anexo, na qual só contém tabelas.
- Como extração de tabelas de PDF pode não ser o caso de nosso interesse agora (além de ser um processo trabalhoso e nem sempre possível), dropar essas páginas de anexos seja uma boa alternativa.



## ANEXO I REGIME DE SUBSTITUIÇÃO TRIBUTÁRIA

	MERCADORIAS SUJEITAS AO REGIME DE SUBSTITUIÇÃO TRIBUTÁRIA
I	Açúcar de qualquer tipo;
II	Água mineral ou potável e gelo.
III	Alcool hidratado e anidro;
IV	Bebidas alcoólicas;



# Crawler: PDF - Biblioteca para manipulação de PDF

- Assim, iremos gerar um PDF de saída contendo apenas o conteúdo textual:

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

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

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

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

```
In [4]: output = PdfFileWriter()

        # it will keep the required pages from PDF to generate the output
        for i in range(input1.getNumPages()):
            if i < 131: # it starts to accumulate from the beginning the document to the attachment
                output.addPage(input1.getPage(i))
```

```
In [5]: with open("normas.pdf", "wb") as outputStream:
        output.write(outputStream) # generate the Norma output file
```

---

# Crawler: PDF - Biblioteca para manipulação de PDF

- Antes de partirmos para a próxima parte, vale destacar aqui que a biblioteca PyPDF ainda permite outros tipos de manipulação, como rotacionar a página em algum ângulo específico, adicionar imagem marca d'água nas páginas do documento de saída, etc.
- Como exemplo, vamos gerar um novo documento contendo apenas duas páginas, na qual a segunda está rotacionada em 90º para a direita.

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

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

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

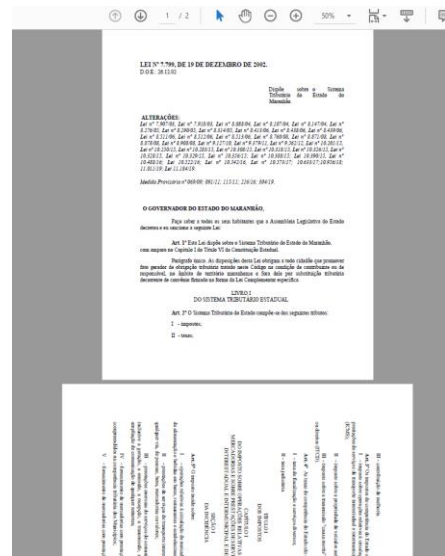
```
Out[3]: 227
```

```
In [4]: output2 = PdfFileWriter()
```

```
In [5]: output2.addPage(input1.getPage(0))
```

```
In [6]: output2.addPage(input1.getPage(1).rotateClockwise(90))
```

```
In [7]: with open("saida2.pdf", "wb") as outputStream:
        output2.write(outputStream) # generate the Norma output file
```



# Crawler: Biblioteca para manipulação de conteúdo em PDF

- Agora que já editamos o PDF para trabalhar somente com o conteúdo que desejamos manipular, vamos agora usar a biblioteca PDFMiner para fazer a extração do conteúdo e biblioteca Spacy, para manipulação de NLP (Processamento de Linguagem Natural):

```
In [5]: import spacy

import io
from pdfminer.converter import TextConverter
from pdfminer.pdfinterp import PDFPageInterpreter
from pdfminer.pdfinterp import PDFResourceManager
from pdfminer.pdfpage import PDFPage
```

# Crawler: Biblioteca para manipulação de conteúdo em PDF

- Seja a seguinte função que iremos definir como `extract_text_from_pdf()`, no qual ela recebe como argumento o *path* do arquivo PDF e retorna o texto extraído para uma variável (do tipo `String`):

```
def extract_text_from_pdf(pdf_path):  
    resource_manager = PDFResourceManager()  
    fake_file_handle = io.StringIO()  
    converter = TextConverter(resource_manager, fake_file_handle)  
    page_interpreter = PDFPageInterpreter(resource_manager, converter)  
  
    with open(pdf_path, 'rb') as fh:  
        for page in PDFPage.get_pages(fh,  
                                       caching=True,  
                                       check_extractable=True):  
            page_interpreter.process_page(page)  
  
            text = fake_file_handle.getvalue()  
  
    # close open handles  
    converter.close()  
    fake_file_handle.close()  
  
    if text:  
        return text
```

Em linhas gerais, esse algoritmo basicamente irá receber o arquivo em PDF e checar cada sentença (letra, palavra, número) e colocá-la na variável *text*.

# Crawler: Biblioteca para manipulação de conteúdo em PDF

- Ao chamarmos a função anterior, ela irá retornar todo o texto para a variável:

```
In [8]: all_text = extract_text_from_pdf('normas.pdf')
```

```
In [9]: all_text
```

```
Out[9]: ' LEI Nº 7.799, DE 19 DE DEZEMBRO DE 2002. D.O.E.: 26.12.02 Dispõe sobre o Sistema Tributário do Estado do Maranhão. ALTERAÇÕES: Lei nº 7.907/03, Lei nº 7.918/03, Lei nº 8.088/04, Lei nº 8.107/04, Lei nº 8.147/04, Lei nº 8.276/05, Lei nº 8.290/05, Lei nº 8.314/05, Lei nº 8.413/06, Lei nº 8.438/06, Lei nº 8.439/06, Lei nº 8.511/06, Lei nº 8.512/06, Lei nº 8.513/06, Lei nº 8.760/08, Lei nº 8.871/08, Lei nº 8.878/08, Lei nº 8.908/08, Lei nº 9.127/10, Lei nº 9.379/11, Lei nº 9.562/12, Lei nº 10.201/15, Lei nº 10.250/15, Lei nº 10.283/15, Lei nº 10.308/15, Lei nº 10.318/15, Lei nº 10.326/15, Lei nº 10.328/15, Lei nº 10.329/15, Lei nº 10.356/15; Lei nº 10.388/15; Lei nº 10.390/15, Lei nº 10.488/16; Lei nº 10.522/16; Lei nº 10.542/16, Lei nº 10.573/17; 10.633/17;10.956/18; 11.011/19; Lei nº 11.184/19. Medida Provisória nº 069/09; 091/11; 115/11; 216/16; 304/19. O GOVERNADOR DO ESTADO DO MARANHÃO, Faço saber a todos os seus habitantes que a Assembleia Legislativa do Estado decretou e eu sanciono a seguinte Lei: Art. 1º Esta Lei dispõe sobre o Sistema Tributário do Estado do Maranhão, com amparo no Capítulo I do Título VI da Constituição Estadual. Parágrafo único. As disposições desta Lei obrigam a todo cidadão que promover fato gerador de obrigação tributária tratado neste Código na condição de contribuinte ou de responsável, no âmbito do território maranhense e fora dele por substituição tributária decorrente de convênio firmado na forma da Lei Complementar específica. LIVRO I DO SISTEMA TRIBUTÁRIO ESTADUAL Art. 2º O Sistema Tributário do Estado compõe-se dos seguintes tributos: I - impostos; II - taxas; \x0c III - contribuição de melhoria. Art. 3º Os impostos de competência do Estado são os seguintes: I - imposto sobre operações relativas à circulação de mercadorias e sobre prestações de serviços de transporte interestadual e intermunicipal e de comunicação (ICMS); II - imposto sobre a propriedade de veículos automotores (IPVA); III - imposto sobre a transmissão "causa mortis" e doação de quaisquer bens ou direitos (ITCD). Art. 4º As taxas de competência do Estado são as seguintes: I - taxa de fiscalização e serviços diversos; II - taxa judiciária. TÍTULO I DOS IMPOSTOS CAPÍTULO I DO IMPOSTO SOBRE OPERAÇÕES RELATIVAS À CIRCULAÇÃO DE MERCADORIAS E SOBRE PRESTAÇÕES DE SERVIÇOS DE TRANSPORTE INTERESTADUAL E INTERMUNICIPAL E DE COMUNICAÇÃO
```

# Crawler: Biblioteca para manipulação de conteúdo em PDF

- A variável string anterior conterá todo o texto tal qual o PDF, entretanto:
  - A variável não sabe reconhecer uma palavra. Embora todo o texto esteja na íntegra, a manipulação textual se dará em forma de token (menor unidade da palavra; no nosso caso, letra a letra).

```
In [10]: len(all_text)
```

```
Out[10]: 259548
```

```
In [11]: all_text[1]
```

```
Out[11]: 'L'
```

- Trabalhar com texto extraído de PDF pode ser um desafio um tanto trabalhoso, visto que o texto não possui uma estrutura que nos auxilie na hora de retirarmos a informação desejada.

# Crawler: Biblioteca para manipulação de conteúdo em PDF

- Uma estratégia para reconhecimento de palavra - ao invés de manipulação token a token - é o uso da biblioteca *Spacy*, conhecida para lidar com Processamento de Linguagem Natural;
- Basta configurar um idioma e depois fazê-la percorrer toda a String para que ela possa identificar o que for de palavra que esteja em seu dicionário;
- Para textos em português, a configuração do idioma tem que ser por português de Portugal.

```
In [11]: nlp = spacy.load('pt') # setting the language of the words
```

```
In [12]: doc = nlp(all_text)
```

- Por fim, teremos a seguinte situação:

```
all_text[1]
```

```
'L'
```

```
doc[1]
```

```
LEI
```

Observe que na primeira variável só lidamos com token, enquanto que na segunda lidamos com a palavra.

# Crawler: Biblioteca para manipulação de conteúdo em PDF

- Agora vamos analisar um contexto em que gostaríamos de ter uma informação particular do deste documento e gostaríamos de retirá-la do PDF para utilizá-la em alguma aplicação (como banco de dados, por exemplo).
- O documento da Lei 7799 possui um conjunto de hierarquias que constituem a lei:
  - Parte, Livro, Título, Capítulo, Seção, Subseção, Artigo, Parágrafo, Inciso, Alínea e Item.

## TÍTULO I DOS IMPOSTOS

### CAPÍTULO I DO IMPOSTO SOBRE OPERAÇÕES RELATIVAS À CIRCULAÇÃO DE MERCADORIAS E SOBRE PRESTAÇÕES DE SERVIÇOS DE TRANSPORTE INTERESTADUAL E INTERMUNICIPAL E DE COMUNICAÇÃO

#### SEÇÃO I DA INCIDÊNCIA

**Art. 5º** O imposto incide sobre:

I - operações relativas à circulação de mercadorias, inclusive o fornecimento de alimentação e bebidas em bares, restaurantes e estabelecimentos similares;



# Crawler: Biblioteca para manipulação de conteúdo em PDF

- Suponha que queremos percorrer todo o documento e retirar apenas informações referente aos TÍTULOS do documento. O seguinte trecho de código resolveria nosso problema:

```
In [14]: def extrac_titles(doc):

    string = list() # it will keep the piece of text of interest
    law7799 = list() # a list which contains all elements from Norma

    start_titulo = 0
    stop_titulo = 1

    nullify = 0 # this variable nullify key words such as Livro, Capítulo, Artigo (etc) which is INSIDE a text

    for index, token in enumerate(doc):

        # LOOKING FOR "TÍTULO"
        if token.text == "TÍTULO" and nullify == 0: # when the word 'TÍTULO' is identified
            start_titulo = 1
            stop_titulo = 0
            nullify = 1
        if start_titulo == 1 and stop_titulo == 0:
            string.append(token)
            if str(doc[index+1]) == "CAPÍTULO": # Titulo ends when a Capítulo starts
                stop_titulo = 1
                start_livro = 0
                temp = string.copy()
                law7799.append(temp)
                nullify = 0
                string.clear()

    return law7799
```

observe o critério de  
parada!

# Crawler: Biblioteca para manipulação de conteúdo em PDF

- Como podemos observar no documento, o título sempre termina quando o capítulo começa

TÍTULO I  
DOS IMPOSTOS

CAPÍTULO I  
DO IMPOSTO SOBRE OPERAÇÕES RELATIVAS À CIRCULAÇÃO DE  
MERCADORIAS E SOBRE PRESTAÇÕES DE SERVIÇOS DE TRANSPORTE  
INTERESTADUAL E INTERMUNICIPAL E DE COMUNICAÇÃO

SEÇÃO I  
DA INCIDÊNCIA

**Art. 5º** O imposto incide sobre:

I - operações relativas à circulação de mercadorias, inclusive o fornecimento de alimentação e bebidas em bares, restaurantes e estabelecimentos similares;

# Crawler: Biblioteca para manipulação de conteúdo em PDF

- Com a definição da função anterior, teríamos a seguinte saída:

```
In [16]: titles = extrac_titles(doc)
```

```
In [17]: for text in titles:  
         print(' '.join(str(i) for i in text))
```

```
TÍTULO I    DOS IMPOSTOS  
TÍTULO II   DAS TAXAS  
TÍTULO III   DA CONTRIBUIÇÃO DE MELHORIA  
TÍTULO I  
TÍTULO II
```