

CAPTURA E VISUALIZAÇÃO DE DADOS

Diénert Vieira

dienertalencar@gmail.com

(83) 9 8182-1478

CAPTURA – SCRAPY



<https://www.anaconda.com/distribution/#download-section>

► Python 3+

Several thin, parallel diagonal lines in a light gray color extending from the bottom right corner towards the center of the slide.

CAPTURA – SCRAPY



<https://scrapy.org/>

Framework de código aberto e colaborativo para extração de dados de sites de uma forma rápida, simples e extensível.

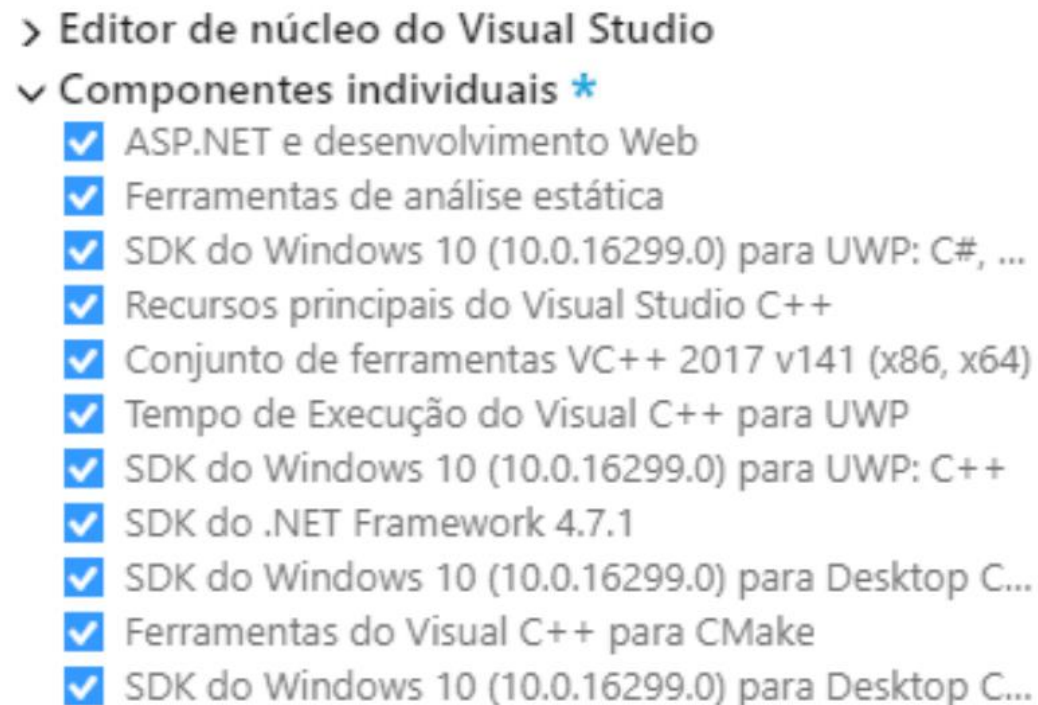
- Instalação: depois de ter o Anaconda instalado, executar no terminal:

```
conda install -c conda-forge scrapy
```

CAPURA – SCRAPY

- ▶ Também se pode instalar via pip, mas (no windows) é preciso instalar Visual Studio C++ através do instalador do Visual Studio Community
- ▶ (Windows) Selecionar os seguintes componentes:
- ▶ Executar o seguinte comando:

```
pip install scrapy
```

- 
- > Editor de núcleo do Visual Studio
- ✓ Componentes individuais *
- ✓ ASP.NET e desenvolvimento Web
 - ✓ Ferramentas de análise estática
 - ✓ SDK do Windows 10 (10.0.16299.0) para UWP: C#, ...
 - ✓ Recursos principais do Visual Studio C++
 - ✓ Conjunto de ferramentas VC++ 2017 v141 (x86, x64)
 - ✓ Tempo de Execução do Visual C++ para UWP
 - ✓ SDK do Windows 10 (10.0.16299.0) para UWP: C++
 - ✓ SDK do .NET Framework 4.7.1
 - ✓ SDK do Windows 10 (10.0.16299.0) para Desktop C...
 - ✓ Ferramentas do Visual C++ para CMake
 - ✓ SDK do Windows 10 (10.0.16299.0) para Desktop C...

CAPURA – SCRAPY

► O que é preciso para criar o meu primeiro Spider?

1. Criar um arquivo: terra_spider.py

2. Importar o pacote do scrapy:

`import scrapy`

3. Criar uma classe que herde de Spider

`class TerraSpider (scrapy.Spider):`

4. Configurar um nome e um conjunto de URLs a serem capturadas como propriedades dessa classe

```
name = 'Terra'
start_urls = {
    'https://www.terra.com.br/'
}
```

CAPURA – SCRAPY

- ▶ O que é preciso para criar o meu primeiro Spider?
 - ▶ Implementar o método parse
 - ▶ Extrair das páginas os dados de interesse, através da variável **response**:

```
10     def parse(self, response):
11         titulos = response.css(".main-url::text")
12         print("titulos: {}".format(len(titulos)))
13         for titulo in titulos:
14             conteudo = titulo.extract().strip()
15             if conteudo != "":
16                 yield {
17                     'titulo': conteudo
18                 }
```

CAPURA – SCRAPY

- ▶ Executar o spider no terminal de comandos:

```
scrapy runspider terra_spider.py -o noticias.csv
```

```
scrapy runspider terra_spider.py -o noticias.json
```

- ▶ A extensão do arquivo é levada em consideração para o formato resultante, podem ser:

JSON, JSON Lines (*.jl), CSV, XML, Pickle, Marshal



Apropriado para
Streaming

CAPTURA – SCRAPY

Vamos testar:

- ▶ Implementar um Spider para capturar os títulos das notícias do Terra
- ▶ Testar diferentes tipos de saída



CAPTURA – SCRAPY

Você está usando um Proxy?

Siga as instruções em <https://github.com/Dienert/proxy>

Para Windows, Linux ou Mac

Se mesmo assim você recebe a seguinte mensagem:

Could not open CONNECT tunnel with proxy localhost:3128
[{'status': 407, 'reason': b'Proxy Authentication Required'}]

Configure as variáveis de ambiente:

`http_proxy=http://usuário:senha@localhost:3128`

`https_proxy=http://usuário:senha@localhost:3128`

CAPTURA – SCRAPY

Está ocorrendo o seguinte problema ao capturar algo?

```
from cryptography.hazmat.bindings._openssl import ffi, lib
ImportError: DLL load failed: The operating system cannot run %1
```

1. Crie um no ambiente conda (ex.:
`conda create -n scrapy-env python=3`)
2. Ative o ambiente (`activate scrapy-env`)
3. Execute: `conda install -c conda-forge scrapy`

CAPTURA – SCRAPY

Localização do Python

```
python  
import sys, os  
os.path.dirname(sys.executable)
```



CAPTURA – SCRAPY

- Como descobrir o trecho de código HTML adequado para o meu propósito?

CSS Selectors ou XPath



CAPURA – SCRAPY

- Como descobrir o trecho de código HTML adequado para o meu propósito?

```
titulos = response.css(".main-url::text")
```

```
titulos = response.xpath(  
'//*[contains(@class, "main-url")]/text()'  
)
```

CAPURA – SCRAPY

- ▶ Como descobrir o trecho de código HTML adequado para o meu propósito?
- ▶ Ferramenta de Exploração Visual do código de uma página

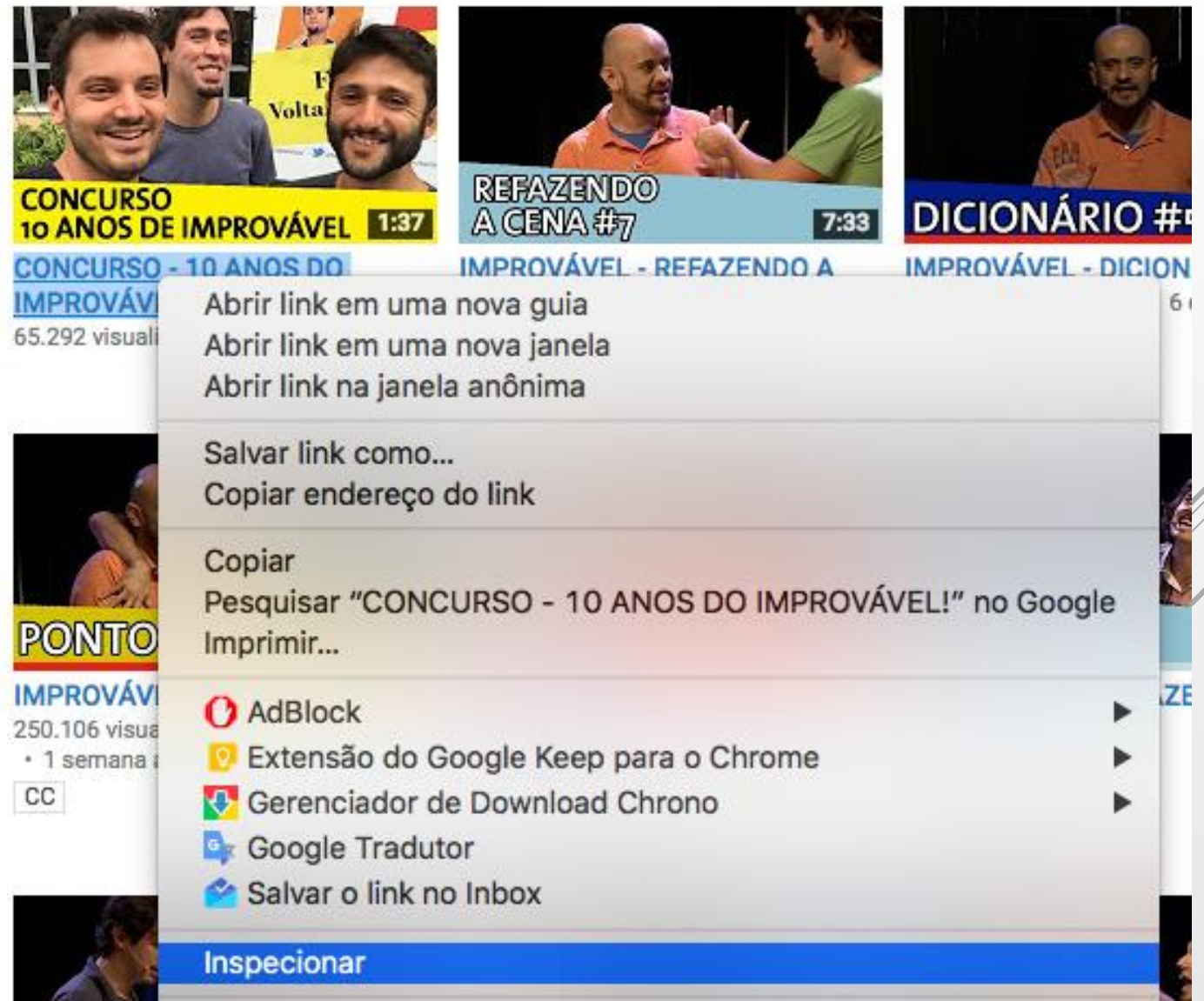
<http://selectorgadget.com/>

- ▶ Pode ser usado como uma extensão do chrome ou adicionar um atalho aos favoritos
- ▶ Conferir portal do Terra

CAPTURA - YOUTUBE

- ▶ Inspecionando elementos de uma página:
 - ▶ Botão direito sobre o Título do Vídeo, por exemplo
 - ▶ Inspecionar

Ferramentas para Desenvolvimento Web



CAPTURA - YOUTUBE

- Mostrar Inspeção de Código

The screenshot displays a YouTube video player interface with four video thumbnails. The first thumbnail is titled "CONCURSO - 10 ANOS DO IMPROVÁVEL!" with 65.292 visualizações and 22 horas atrás. The second is "IMPROVÁVEL - REFAZENDO A CENA #7" with 120.364 visualizações and 1 dia atrás. The third is "IMPROVÁVEL - DICIONÁRIO #5" with 181.665 visualizações and 6 dias atrás. The fourth is "IMPROVÁVEL - MANEQUIM #2" with 196.881 visualizações and 1 semana atrás. Below the thumbnails, the Chrome DevTools 'Elements' panel is open, showing the HTML structure of the video player. The selected element is a link with the class 'yt-ui-ellipsis-2' and the title 'CONCURSO - 10 ANOS DO IMPROVÁVEL!'. The 'Styles' panel on the right shows the default styles for this element, including 'display: -webkit-box' and 'line-clamp: 2'.

CONCURSO - 10 ANOS DO IMPROVÁVEL!
65.292 visualizações • 22 horas atrás

IMPROVÁVEL - REFAZENDO A CENA #7
120.364 visualizações • 1 dia atrás

IMPROVÁVEL - DICIONÁRIO #5
181.665 visualizações • 6 dias atrás

IMPROVÁVEL - MANEQUIM #2
196.881 visualizações • 1 semana atrás

Elements Console Sources Network Timeline Profiles Application Security Audits AdBlock

```
grid vve-check" data-context-item-id="co4bwQaHhPI" data-visibility-tracking="CCwQLDUYASITCIj2xeK579ECFQNSkAodTCQECyibHEDyiZ60kPiGx3I=">::before<div class="yt-lockup-dismissable">><div class="yt-lockup-thumbnail">...</div><div class="yt-lockup-content"><h3 class="yt-lockup-title">...<a class="yt-uix-sessionlink yt-uix-tile-link spf-link yt-ui-ellipsis yt-ui-ellipsis-2" dir="ltr" title="CONCURSO - 10 ANOS DO IMPROVÁVEL!" aria-describedby="description-id-613713" data-sessionlink="ei=qB6SWIiCKo0kwQTMjJBY&feature=c4-videos-u&ved=CDQQLx4iEwiI9sXiue_RAHUDUpAKHUwkBAasomxw" href="/watch?v=co4bwQaHhPI"> = $0"CONCURSO - 10 ANOS DO IMPROVÁVEL!"::after</a><span class="accessible-description" id="description-id-613713"> - Duração: 97...li div div div h3 a.yt-uix-sessionlink.yt-uix-tile-link.spf-link.yt-ui-ellipsis.yt-ui-ellipsis-2
```

Styles Computed >>

Filter :hov .cls +

```
element.style {  
}  
. www-player-webp...fldqjIP.css:1  
webkit .yt-ui-ellipsis-2 {  
  -webkit-line-clamp: 2;  
}  
. www-player-webp...fldqjIP.css:1  
webkit .yt-ui-ellipsis {  
  display: -webkit-box;  
  -webkit-box-orient: vertical;  
}  
.w www-core-webp-vfl2EV8mG.css:1  
ebkit .yt-ui-ellipsis-2 {  
  -webkit-line-clamp: 2;  
}
```


CAPTURA - DEPURAÇÃO

► Depuração executando um Spider

1. Abrir no browser

- A extensão do Chrome SelectorGadget não funciona
- Mas o seu link nos favoritos funciona em qualquer browser

```
import scrapy
from scrapy.utils.response import open_in_browser

class YoutubeSpider(scrapy.Spider):
    name = "Youtube"

    canal = 'videosimprovaveis'
    # canal = 'portadosfundos'

    start_urls = {
        'https://www.youtube.com/user/{}/videos'.format(canal)
    }

    def parse(self, response):
        open_in_browser(response)
```

CAPTURA - DEPURAÇÃO

► Usando o Scrapy Shell

1. A partir de uma URL

```
scrapy shell <url>
```

2. A partir de Arquivos locais

```
scrapy shell ./caminho_relativo/index.html
```

```
scrapy shell ../outro_caminho/arquivo.htm
```

```
scrapy shell /caminho_absoluto/teste.html
```

3. A partir de uma URI

```
scrapy shell file:///caminho_absoluto/teste.html
```

CAPTURA - DEPURAÇÃO

► Usando o Scrapy Shell

► Comandos

```
fetch(url[, redirect=True])  
view(response)
```

Submete outra
Requisição

Equivalente ao
`open_in_browser(response)`

► Exemplo

```
scrapy shell https://www.youtube.com/user/videosimprovaveis/videos
```

CAPTURA - DEPURAÇÃO

► Usando o Scrapy Shell

► Comandos

Objeto com
Tags sujeito a
novas seleções

```
In [30]: response.css('title')
Out[30]: [<Selector xpath='descendant-or-self::title' data='<title>
Barbixas\n - YouTube</title>'>]

In [31]: response.css('title').extract()
Out[31]: ['<title> Barbixas\n - YouTube</title>']

In [32]: response.css('title').extract_first()
Out[32]: '<title> Barbixas\n - YouTube</title>'

In [33]: response.css('title::text').extract_first()
Out[33]: ' Barbixas\n - YouTube'
```

Lista dos
Elementos
Title com
suas Tags

Primeiro
Elemento
Title com
suas Tags

Não captura
conteúdo de
elementos filhos

CAPTURA - DEPURAÇÃO

► Usando o Scrapy Shell

► Iniciando Shell a partir da execução de um Spider

```
import scrapy
from scrapy.shell import inspect_response

class YotubeSpider(scrapy.Spider):
    name = "Youtube"

    canal = 'videosimprovaveis'
    # canal = 'portadosfundos'

    start_urls = {
        'https://www.youtube.com/user/{}/videos'.format(canal)
    }

    def parse(self, response):
        inspect_response(response, self)
```

CAPTURA – CSS SELECTORS

- ▶ Seletores CSS são padrões usados para selecionar elementos HTML para aplicar algum estilo (CSS)

Referências:

https://www.w3schools.com/cssref/css_selectors.asp

https://developer.mozilla.org/en-US/docs/Web/CSS/CSS_Selectors

Funções: https://www.w3schools.com/cssref/css_functions.asp

CAPTURA – CSS SELECTORS

Seletor Universal

```
* {  
  color: green;  
  font-size: 20px;  
  line-height: 25px;  
}
```

Seletor de Tipo de Elemento

```
ul {  
  list-style: none;  
  border: solid 1px #ccc;  
}
```

CAPTURA – CSS SELECTORS

Seletor de ID

```
#container {  
  width: 960px;  
  margin: 0 auto;  
}
```

Qualquer elemento com
atributo `id="container"`

Seletor de Classe

```
.box {  
  padding: 20px;  
  margin: 10px;  
  width: 240px;  
}
```

Qualquer elemento com
atributo `class="box"`

SELECTORS

```
<div id="container">
  <span class="something">
    <div class="box">
      <p>Teste 1</p>
      <span><p>Teste 2</p></span>
    </div>
  </span>
</div>
<p>Teste 3</p>
<div class="box">
  <p>Teste 4</p>
</div>
```

Combinador de descendência

```
#container .box {
  background-color: #83D0F2;
}
```

Qualquer elemento com atributo `class="box"` dentro de um elemento com atributo `id="container"` (não precisa ser filho diretamente)

Combinador de filho

```
* {
  /* texto sublinhado */
  text-decoration: underline;
}
```

```
div > p {
  /* texto tachado */
  text-decoration: line-through;
}
```

Qualquer parágrafo filho diretamente de uma `div`

CAPTURA – CSS SELECTORS

Combinador de Irmão

```
<div>
  <h2>Title</h2>
  <p>Teste 1</p>
  <p>Teste 2</p>
  <p>Teste 3</p>
```

```
  <div>
    <p>Teste 4</p>
    <p>Teste 5</p>
  </div>
</div>
```

```
h2 ~ p {
  /* texto sublinhado */
  text-decoration: underline;
}
```

2

Qualquer **p** filho de um elemento que contenha um **h2**

Combinação de irmão adjacente

```
p + p {
  /* texto tachado */
  text-decoration: line-through;
}
```

1

Qualquer **p** que antes dele tenha um **p** irmão

CAPTURA – CSS SELECTORS

Combinador de atributo com valor exato

```
input[type="text"] {  
  background-color: #444;  
  width: 200px;  
}
```

Qualquer **input** com
atributo **type** exatamente
igual a "test"

Combinação de atributo contendo a palavra específica

```
[title~="flower"] {  
  border: 5px solid yellow;  
}
```

Qualquer **elemento** com
atributo **title** contendo
"flower" separado por
espaço

CAPTURA – CSS SELECTORS

Combinador de atributo somente com a palavra ou separada por hífen

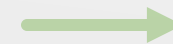
```
[class]="top" {  
  background: yellow;  
}
```



class="top" ou
class="top-
qualquerCoisa"

Combinação de atributo com valor que começa com os caracteres

```
[class^="top"] {  
  background: yellow;  
}
```



class="topeira"

CAPTURA – CSS SELECTORS

Combinação de atributo com valor que termina com os caracteres

```
[class$="test"] {  
    background: yellow;  
}
```



class="test" ou
class="algumaCoisatest"

Combinação de atributo com valor que contém os caracteres

```
[class*="te"] {  
    background: yellow;  
}
```



class="tomate"
ou
class="teste"

CAPTURA – CSS SELECTORS

Vamos Testar:

- ▶ Implementar um Spider que capture os Títulos e os Links dos vídeos de um canal do Youtube
- ▶ Utilizar SelectorGadget para descobrir o CSS Selector apropriado para o título
 - ▶ Utilizar alguma visualização do response no browser e utilizar novamente o SelectorGadget (a partir dos favoritos) para ver o CSS Selector, houve diferença?
- ▶ `youtube_spider_03_css_titulo_link.py`

CAPTURA – XPATH

- ▶ XPath é útil para identificar e navegar através de elementos e atributos em um documento XML
- ▶ Significa XML Path Language
- ▶ Utiliza sintaxe parecida com caminhos de diretórios
- ▶ Possui mais de 200 funções incorporadas
- ▶ XPath é o principal elemento do padrão XSLT (Transformações em Linguagem de Estilo Extensível)
- ▶ É uma recomendação do W3C (Comunidade Internacional que desenvolve padrões abertos para garantir o crescimento a longo prazo da Web)
- ▶ Referências: <https://developer.mozilla.org/en-US/docs/Web/XPath>
<https://developer.mozilla.org/en-US/docs/Web/XPath/Functions>
<https://developer.mozilla.org/en-US/docs/Web/XPath/Axes>

CAPTURA – XPATH – EXEMPLOS

- ▶ Pegar esse XML em:
<https://goo.gl/ntojjC>
- ▶ Executar Scrapy Shell para esse arquivo

```
scrapy shell ./catalog.xml
```

- ▶ CTRL + \ ou exit sai do shell

```
1  <?xml version="1.0" encoding="ISO-8859-1"?>
2  <catalog>
3    <cd country="USA">
4      <title>Empire Burlesque</title>
5      <artist>Bob Dylan</artist>
6      <price>10.90</price>
7    </cd>
8    <cd country="UK">
9      <title>Hide your heart</title>
10     <artist>Bonnie Tyler</artist>
11     <price>9.90</price>
12   </cd>
13   <cd country="USA">
14     <title>Greatest Hits</title>
15     <artist>Dolly Parton</artist>
16     <price>9.90</price>
17   </cd>
18 </catalog>
```


CAPURA – XPATH

- ▶ Selecionar o elemento RAIZ **catalog**:

/catalog

- ▶ Selecionar todos os elementos cd do elemento catalog:

/catalog/cd

- ▶ Selecionar todos os elementos price de todos os elementos cd do elemento catalog:

/catalog/cd/price

- ▶ **Nota:** Se o caminho começa com uma barra (/) ele representa um caminho absoluto para um elemento!

```
1  <?xml version="1.0" encoding="ISO-8859-1"?>
2  <catalog>
3      <cd country="USA">
4          <title>Empire Burlesque</title>
5          <artist>Bob Dylan</artist>
6          <price>10.90</price>
7      </cd>
8      <cd country="UK">
9          <title>Hide your heart</title>
10         <artist>Bonnie Tyler</artist>
11         <price>9.90</price>
12     </cd>
13     <cd country="USA">
14         <title>Greatest Hits</title>
15         <artist>Dolly Parton</artist>
16         <price>9.90</price>
17     </cd>
18 </catalog>
```

CAPURA – XPATH

- **Nota:** Se o caminho começa com duas barras (**//**) então todos os elementos no documento que se encaixam no critério serão selecionados (mesmo que eles estejam em níveis diferentes da árvore XML)!
- Selecionar todos os elementos **cd** no documento:

//cd

```
1  <?xml version="1.0" encoding="ISO-8859-1"?>
2  <catalog>
3      <cd country="USA">
4          <title>Empire Burlesque</title>
5          <artist>Bob Dylan</artist>
6          <price>10.90</price>
7      </cd>
8      <cd country="UK">
9          <title>Hide your heart</title>
10         <artist>Bonnie Tyler</artist>
11         <price>9.90</price>
12     </cd>
13     <cd country="USA">
14         <title>Greatest Hits</title>
15         <artist>Dolly Parton</artist>
16         <price>9.90</price>
17     </cd>
18 </catalog>
```

CAPTURA – XPATH

Selecionando elementos desconhecidos

- ▶ Curingas (*) podem ser usados para selecionar elementos XML desconhecidos.
- ▶ Selecionar todos os elementos filhos de todos os elementos **cd** do elemento **catalog**:

/catalog/cd/*

- ▶ Selecionar todos os elementos **price** que são elementos netos do elemento **catalog**:

/catalog/*/price

```
1  <?xml version="1.0" encoding="ISO-8859-1"?>
2  <catalog>
3      <cd country="USA">
4          <title>Empire Burlesque</title>
5          <artist>Bob Dylan</artist>
6          <price>10.90</price>
7      </cd>
8      <cd country="UK">
9          <title>Hide your heart</title>
10         <artist>Bonnie Tyler</artist>
11         <price>9.90</price>
12     </cd>
13     <cd country="USA">
14         <title>Greatest Hits</title>
15         <artist>Dolly Parton</artist>
16         <price>9.90</price>
17     </cd>
18 </catalog>
```

CAPURA – XPATH

Selecionando elementos desconhecidos

- Selecionar todos os elementos **price** que têm dois ancestrais:

`/*/*/price`

- Selecionar todos os elementos no documento:

`//*`

```
1  <?xml version="1.0" encoding="ISO-8859-1"?>
2  <catalog>
3      <cd country="USA">
4          <title>Empire Burlesque</title>
5          <artist>Bob Dylan</artist>
6          <price>10.90</price>
7      </cd>
8      <cd country="UK">
9          <title>Hide your heart</title>
10         <artist>Bonnie Tyler</artist>
11         <price>9.90</price>
12     </cd>
13     <cd country="USA">
14         <title>Greatest Hits</title>
15         <artist>Dolly Parton</artist>
16         <price>9.90</price>
17     </cd>
18 </catalog>
```

CAPTURA – XPATH

Selecionando Seções

- ▶ Usando-se colchetes numa expressão XPath você pode especificar um elemento adiante.
- ▶ Selecionar o primeiro elemento **cd** filho do elemento catalog:

/catalog/cd[1]

- ▶ Selecionar o último elemento **cd** filho do elemento **catalog** (Nota: não existe a função first()):

/catalog/cd[last()]

- ▶ Selecionar todos os elementos **cd** do elemento **catalog** que tem um elemento **price**:

/catalog/cd[price]

```
1  <?xml version="1.0" encoding="ISO-8859-1"?>
2  <catalog>
3      <cd country="USA">
4          <title>Empire Burlesque</title>
5          <artist>Bob Dylan</artist>
6          <price>10.90</price>
7      </cd>
8      <cd country="UK">
9          <title>Hide your heart</title>
10         <artist>Bonnie Tyler</artist>
11         <price>9.90</price>
12     </cd>
13     <cd country="USA">
14         <title>Greatest Hits</title>
15         <artist>Dolly Parton</artist>
16         <price>9.90</price>
17     </cd>
18 </catalog>
```

CAPURA – XPATH

Selecionando Seções

- Selecionar todos os elementos **cd** do elemento **catalog** que tem um elemento **price** com valor de **10.90**:

`/catalog/cd[price=10.90]`

- Selecionar todos os elementos **price** de todos os elementos **cd** do elemento **catalog** que tem um elemento **price** com valor de **10.90**:

`/catalog/cd[price=10.90]/price`

```
1  <?xml version="1.0" encoding="ISO-8859-1"?>
2  <catalog>
3      <cd country="USA">
4          <title>Empire Burlesque</title>
5          <artist>Bob Dylan</artist>
6          <price>10.90</price>
7      </cd>
8      <cd country="UK">
9          <title>Hide your heart</title>
10         <artist>Bonnie Tyler</artist>
11         <price>9.90</price>
12     </cd>
13     <cd country="USA">
14         <title>Greatest Hits</title>
15         <artist>Dolly Parton</artist>
16         <price>9.90</price>
17     </cd>
18 </catalog>
```


CAPURA – XPATH

Selecionando vários caminhos

- ▶ Usando o operador "|" numa expressão XPath você pode selecionar vários caminhos.
- ▶ Selecionar todos os elementos title e artist do elemento cd do elemento catalog:

/catalog/cd/title | /catalog/cd/artist

- ▶ Selecionar todos os elementos title e artist do documento:

//title | //artist

```
1  <?xml version="1.0" encoding="ISO-8859-1"?>
2  <catalog>
3      <cd country="USA">
4          <title>Empire Burlesque</title>
5          <artist>Bob Dylan</artist>
6          <price>10.90</price>
7      </cd>
8      <cd country="UK">
9          <title>Hide your heart</title>
10         <artist>Bonnie Tyler</artist>
11         <price>9.90</price>
12     </cd>
13     <cd country="USA">
14         <title>Greatest Hits</title>
15         <artist>Dolly Parton</artist>
16         <price>9.90</price>
17     </cd>
18 </catalog>
```

CAPURA – XPATH

Selecionando vários caminhos

- Selecionar todos os elementos title, artist e price do documento:

`//title | //artist | //price`

- Selecionar todos os elementos title do elemento cd do elemento catalog, e todos os elementos artist no documento:

`/catalog/cd/title | //artist`

```
1  <?xml version="1.0" encoding="ISO-8859-1"?>
2  <catalog>
3      <cd country="USA">
4          <title>Empire Burlesque</title>
5          <artist>Bob Dylan</artist>
6          <price>10.90</price>
7      </cd>
8      <cd country="UK">
9          <title>Hide your heart</title>
10         <artist>Bonnie Tyler</artist>
11         <price>9.90</price>
12     </cd>
13     <cd country="USA">
14         <title>Greatest Hits</title>
15         <artist>Dolly Parton</artist>
16         <price>9.90</price>
17     </cd>
18 </catalog>
```


CAPURA – XPATH

Selecionando atributos

- ▶ Em XPath todos os atributos são especificados pelo prefixo "@".
- ▶ Selecionar todos os atributos chamados **country**:

//@country

- ▶ Selecionar todos os elementos **cd** que tem um atributo chamado **country**:

//cd[@country]

- ▶ Selecionar todos os elementos **cd** que tem algum atributo:

//cd[@*]

```
1  <?xml version="1.0" encoding="ISO-8859-1"?>
2  <catalog>
3      <cd country="USA">
4          <title>Empire Burlesque</title>
5          <artist>Bob Dylan</artist>
6          <price>10.90</price>
7      </cd>
8      <cd country="UK">
9          <title>Hide your heart</title>
10         <artist>Bonnie Tyler</artist>
11         <price>9.90</price>
12     </cd>
13     <cd country="USA">
14         <title>Greatest Hits</title>
15         <artist>Dolly Parton</artist>
16         <price>9.90</price>
17     </cd>
18 </catalog>
```

CAPTURA – XPATH

Selecionando atributos

- Selecionar todos os elementos **cd** que tem um atributo chamado **country** com valor 'UK':

//cd[@country='UK']

- Selecionar atributo country do segundo elemento cd

//cd[2]/@country

```
1  <?xml version="1.0" encoding="ISO-8859-1"?>
2  <catalog>
3      <cd country="USA">
4          <title>Empire Burlesque</title>
5          <artist>Bob Dylan</artist>
6          <price>10.90</price>
7      </cd>
8      <cd country="UK">
9          <title>Hide your heart</title>
10         <artist>Bonnie Tyler</artist>
11         <price>9.90</price>
12     </cd>
13     <cd country="USA">
14         <title>Greatest Hits</title>
15         <artist>Dolly Parton</artist>
16         <price>9.90</price>
17     </cd>
18 </catalog>
```

CAPURA – XPATH++

► Funções e caminhos relativos

Exemplo	Resultado
cd	Seleciona todos os elementos cd que são filhos do nó atual
*	Seleciona todos os elementos filhos do nó atual
text()	Seleciona todos os nós textos filhos do nó atual
@src	Seleciona o atributo src do nó atual
@*	Seleciona todos os atributos do nó atual
*/cd	Seleciona todos os cds netos do nó atual
.	Seleciona o nó atual
./cd	Seleciona os elementos cds descendentes do nó atual
..	Seleciona o pai do nó atual
../@src	Seleciona o atributo src do pai do nó atual
contains(e, str)	Testa se o elemento e contém a string str

CAPTURA – CSS SELECTORS

Vamos Testar:

- ▶ Atualize o seu YoutubeSpider para capturar os Títulos e os Links dos vídeos usando XPath
- ▶ `youtube_spider_04_xpath_titulo_link`

CAPTURA – CSS SELECTORS

Vamos Testar:

- ▶ Atualize o seu YoutubeSpider para capturar também o número de visualizações, a duração do vídeo e há quantos dias o vídeo foi postado.
- ▶ Pode usar CSS Selector, XPath ou os dois (escolha o mais conveniente)
- ▶ `youtube_spider_05_todas_informacoes.py`

CAPURA – SCRAPY

- ▶ Como descobrir quais são as requisições submetidas ao servidor? Quais parâmetros foram utilizados?
- ▶ E se for uma requisição AJAX, é possível capturar esse tipo de conteúdo?

Ferramentas para
Desenvolvimento Web



Chrome DevTools



Firefox Developer Tools

CAPTURA – EXEMPLO 1 - YOUTUBE

► Problema: Contém paginação

The screenshot displays a grid of 18 video thumbnails from a YouTube search results page. Each thumbnail includes a video preview, a title bar with the video title and duration, a subtitle with the series name and episode number, view count, upload time, and a Creative Commons license icon.

Thumbnail	Title	Duration	Series	Views	Upload Time	License
1	OBJETIVOS #3	7:02	IMPROVÁVEL - OBJETIVOS #3	355.418	1 mês atrás	CC
2	CENAS #110	8:29	IMPROVÁVEL - CENAS IMPROVÁVEIS #110	538.092	1 mês atrás	CC
3	QUADRADO #22	7:13	IMPROVÁVEL - QUADRADO IMPROVÁVEL #22	253.907	1 mês atrás	CC
4	RECONSTITUIÇÃO #7	9:56	IMPROVÁVEL - RECONSTITUIÇÃO IMPROVÁVEL #7	243.734	1 mês atrás	CC
5	CENAS #109	5:06	IMPROVÁVEL - CENAS IMPROVÁVEIS #109	461.836	1 mês atrás	CC
6	REALIDADE PARALELA #12	5:49	IMPROVÁVEL - REALIDADE PARALELA #12	212.146	2 meses atrás	CC
7	REFAZENDO A CENA #5	5:50	IMPROVÁVEL - REFAZENDO A CENA #5	203.293	2 meses atrás	CC
8	ENTRA-SAI #7	7:26	IMPROVÁVEL - ENTRA-SAI #7	281.915	2 meses atrás	CC
9	QUADRADO #21	8:16	IMPROVÁVEL - QUADRADO IMPROVÁVEL #21	286.310	2 meses atrás	CC
10	CENAS #108	7:33	IMPROVÁVEL - CENAS IMPROVÁVEIS #108	565.535	2 meses atrás	CC
11	REALIDADE PARALELA #11	7:15	IMPROVÁVEL - REALIDADE PARALELA #11	272.747	2 meses atrás	CC
12	MORTES #8	7:08	IMPROVÁVEL - MORTES IMPROVÁVEIS #8	473.273	2 meses atrás	CC
13	CENAS #107	6:36	IMPROVÁVEL - CENAS IMPROVÁVEIS #107	504.247	2 meses atrás	CC
14	REFAZENDO A CENA #4	5:13	IMPROVÁVEL - REFAZENDO A CENA #4	237.223	2 meses atrás	CC
15	MUSICAL #26	12:11	IMPROVÁVEL - MUSICAL IMPROVÁVEL #26	294.780	3 meses atrás	CC
16	ENTRA-SAI #6	6:01	IMPROVÁVEL - ENTRA-SAI #6	291.537	3 meses atrás	CC
17	TRAILER #4	6:54	IMPROVÁVEL - TRAILER IMPROVÁVEL #4	287.920	3 meses atrás	CC
18	CENAS #106	5:22	IMPROVÁVEL - CENAS IMPROVÁVEIS #106	440.597	3 meses atrás	CC

A large pink arrow points to a button labeled "Carregar mais" (Load more) at the bottom center of the page.

CAPTURA – EXEMPLO 1 - YOUTUBE

► Explicação do Código

```
def parse(self, response):  
    link = response.css('button.load-more-button::attr("data-uix-load-more-href")').extract_first()  
    if bool(link and link.strip()):  
        for item in self.parse_page(response):  
            yield item  
    else:  
        obj = json.loads(response.text)  
        page = Selector(text=obj["content_html"])  
        for item in self.parse_page(page):  
            yield item  
        if "load_more_widget_html" in obj:  
            botao = Selector(text=obj["load_more_widget_html"])  
            link = botao.css('button.load-more-button::attr("data-uix-load-more-href")').extract_first()  
    if bool(link and link.strip()):  
        yield response.follow(link)
```

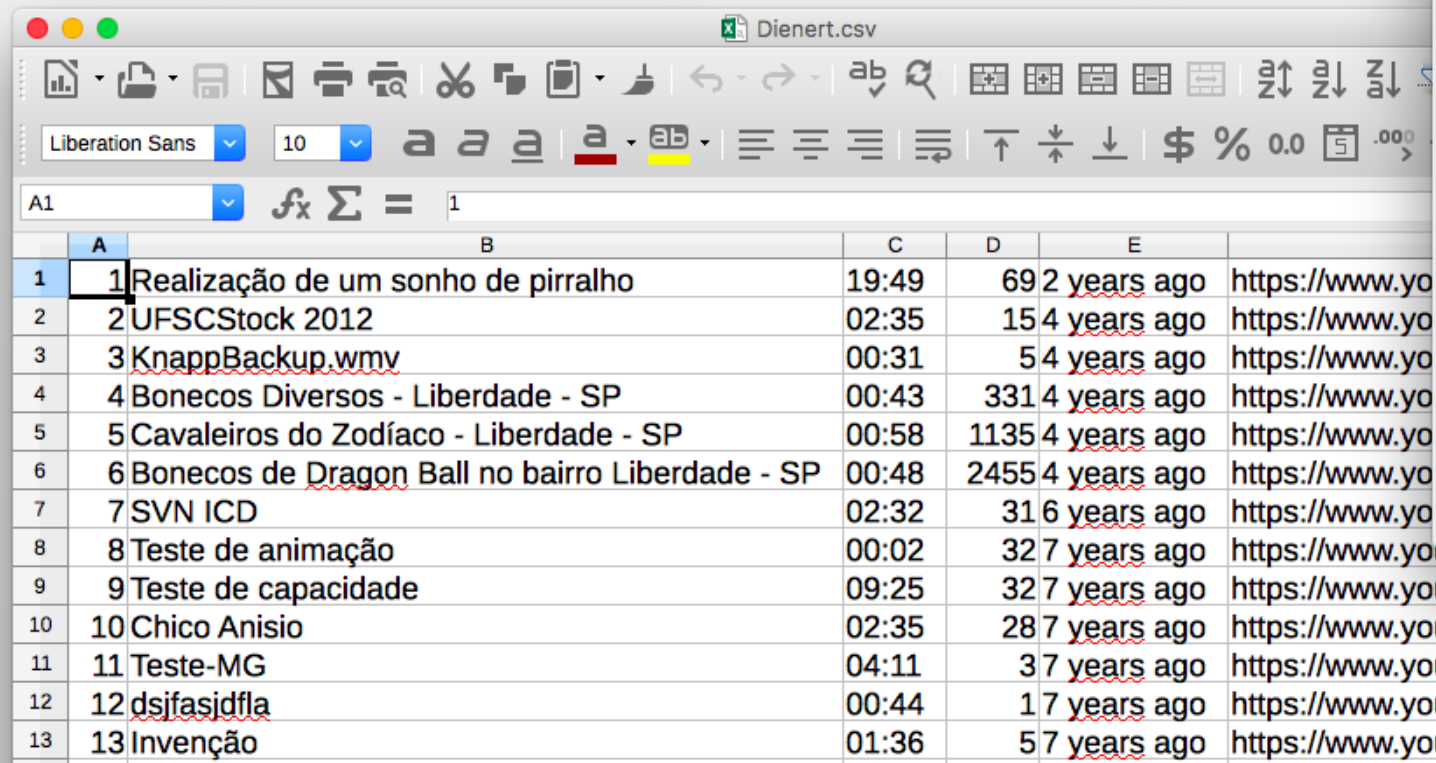
Link para carregar os próximos vídeos

Transformando a resposta em JSON em um dicionário Python

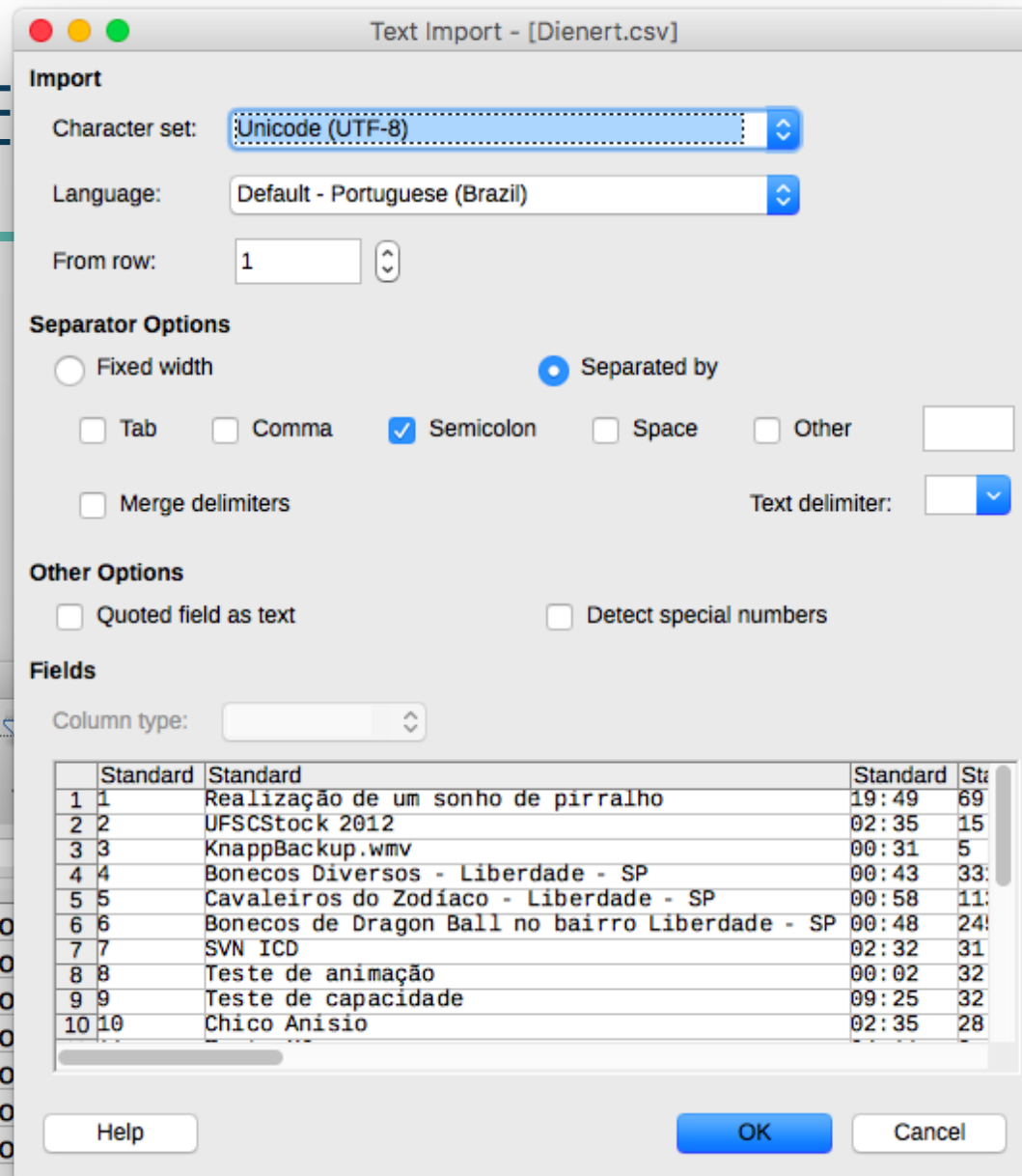
Link para carregar os próximos vídeos

CAPTURA – EXEMPLO 1 - YOUTUBE

- ▶ Abrindo o arquivo de saída
- ▶ LibreOffice
 - ▶ Permite escolher a codificação
 - ▶ Separador de colunas



	A	B	C	D	E
1	1	Realização de um sonho de pirralho	19:49	69	2 years ago
2	2	UFSCStock 2012	02:35	15	4 years ago
3	3	KnappBackup.wmv	00:31	5	4 years ago
4	4	Bonecos Diversos - Liberdade - SP	00:43	331	4 years ago
5	5	Cavaleiros do Zodíaco - Liberdade - SP	00:58	1135	4 years ago
6	6	Bonecos de Dragon Ball no bairro Liberdade - SP	00:48	2455	4 years ago
7	7	SVN ICD	02:32	31	6 years ago
8	8	Teste de animação	00:02	32	7 years ago
9	9	Teste de capacidade	09:25	32	7 years ago
10	10	Chico Anisio	02:35	28	7 years ago
11	11	Teste-MG	04:11	3	7 years ago
12	12	dsjfasjdfla	00:44	1	7 years ago
13	13	Invenção	01:36	5	7 years ago



Text Import - [Dienert.csv]

Import

Character set: Unicode (UTF-8)

Language: Default - Portuguese (Brazil)

From row: 1

Separator Options

☐ Fixed width ☒ Separated by

☐ Tab ☐ Comma ☒ Semicolon ☐ Space ☐ Other

☐ Merge delimiters

Text delimiter:

Other Options

☐ Quoted field as text ☐ Detect special numbers

Fields

Column type:

	Standard	Standard	Standard	Standard
1	1	Realização de um sonho de pirralho	19:49	69
2	2	UFSCStock 2012	02:35	15
3	3	KnappBackup.wmv	00:31	5
4	4	Bonecos Diversos - Liberdade - SP	00:43	331
5	5	Cavaleiros do Zodíaco - Liberdade - SP	00:58	1135
6	6	Bonecos de Dragon Ball no bairro Liberdade - SP	00:48	2455
7	7	SVN ICD	02:32	31
8	8	Teste de animação	00:02	32
9	9	Teste de capacidade	09:25	32
10	10	Chico Anisio	02:35	28

Help OK Cancel