Mineração de Dados
Aula 1 – parte 3

Especialização em Ciência de Dados e suas Aplicações

UTFPR
UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ

Coleta

Utilizando APIs

Permite obter dados facilmente

- **Dois tipos básicos**:
  - *Streaming*
  - Requisições (requisição-resposta)

- **Disponíveis:** Twitter, Foursquare, etc

Utilizando APIs

Exemplo: Usando a API do Twitter

Passo1: Instalar a biblioteca TwitterAPI:

- pip install TwitterAPI
- ou fazer um clone no https://github.com/geduldig/TwitterAPI

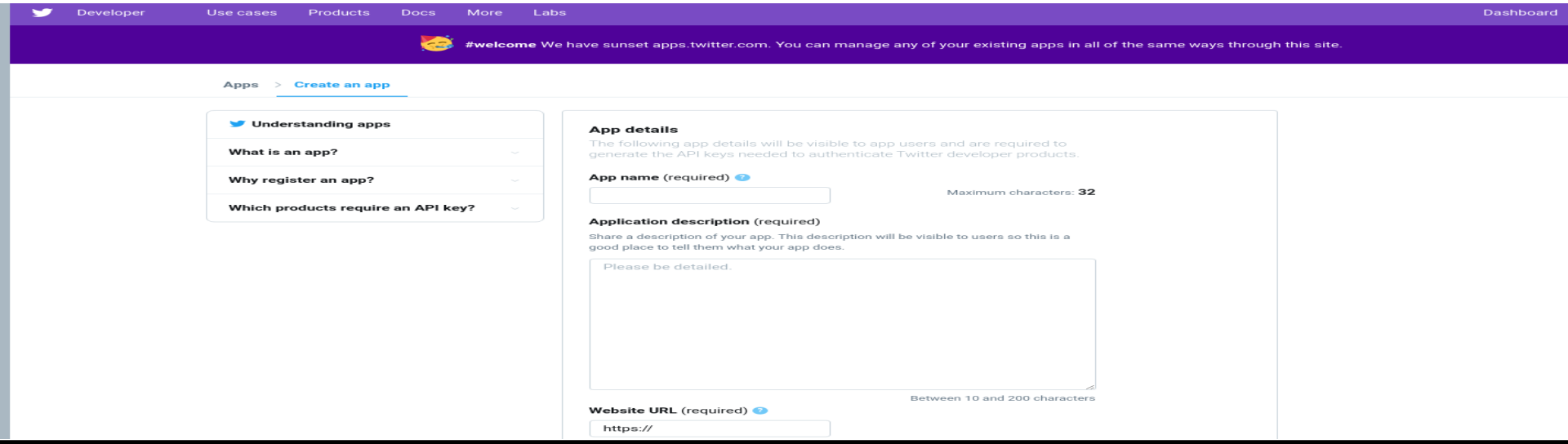
Utilizando APIs

Exemplo: Usando a API do Twitter

Passo 2. Criar uma aplicação no Twitter

a. Necessário ter cadastro no Twitter

b. Acessar https://developer.twitter.com/en/apps/create e clicar em “Create an App”



Utilizando APIs

Exemplo: Usando a API do Twitter

Passo 3. Gerar os *tokens* da app

a. Na tela de detalhes da aplicação, clicar na aba "*Keys and Access Token*"

b. Para gerar o *token*, nesta mesma aba clique em "*Create Access Token*"

As opções podem ter mudado, mas a ideia continua a mesma.

Utilizando APIs

Exemplo: Usando a API do Twitter

```
from TwitterAPI import TwitterAPI

twitter_api = TwitterAPI(consumer_key='XXX
                         consumer_secret='XXXX',
                         access_token_key='XXXX',
                         access_token_secret='XXXX')

for item in twitter_api.request('statuses/filter', {'track':‘csbc'}):
    print item['text']
```

Facilita a comunicação com a API do Twitter

Utilizando APIs

Exemplo: Usando a API do Twitter

```
from TwitterAPI import TwitterAPI

twitter_api = TwitterAPI(consumer_key='XXXX',
                         consumer_secret='XXXX',
                         access_token_key='XXXX',
                         access_token_secret='XXXX')

for item in twitter_api.request('statuses/filter', {'track':‘csbc'}):
    print item['text']
```

Tokens obtidos

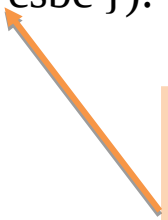
Utilizando APIs

Exemplo: Usando a API do Twitter

```
from TwitterAPI import TwitterAPI

twitter_api = TwitterAPI(consumer_key='XXXX',
                         consumer_secret='XXXX',
                         access_token_key='XXXX',
                         access_token_secret='XXXX')

for item in twitter_api.request('statuses/filter', {'track':‘csbc'}):
    print item['text']
```



Requisição dos *tweets* com a termo “csbc”

Exemplo: Usando a API do Twitter

Demonstração – Coleta Streaming

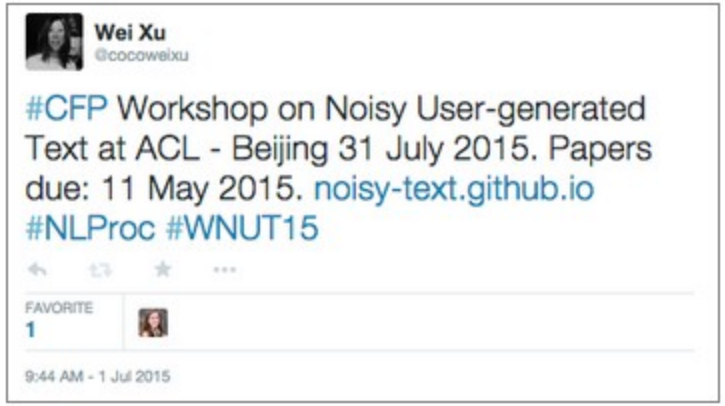
coletaAPIs/streamingTwitter/exemplo1

coletaAPIs/streamingTwitter/exemplo2/coletaTweetsStreaming.py
(Exemplo anterior com pequenas novidades)

Utilizando APIs

Exemplo: Usando a API do Twitter

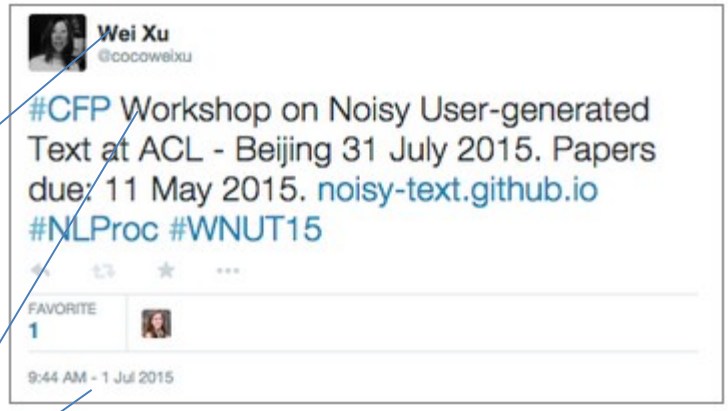
Exemplo de saída



Slide derivado de Wei Xu

Utilizando APIs

Exemplo: Usando a API do Twitter




```
{"favorited": false, "contributors": null, "truncated": false, "text": "#CFP Workshop on Noisy User-generated Text at ACL - Beijing 31 July 2015. Papers due: 11 May 2015. http://t.co/rcygyEowqH   #NLProc #WNUT15", "possibly_sensitive": false, "in_reply_to_status_id": null, "user": {"follow_request_sent": null, "profile_use_background_image": true, "default_profile_image": false, "id": 237918251, "verified": false, "profile_image_url_https": "https://pbs.twimg.com/profile_images/527088456967544832/DnclpoZO_normal.jpeg", "profile_sidebar_fill_color": "DDEEF6", "profile_text_color": "333333", "followers_count": 226, "profile_sidebar_border_color": "C0DEED", "id_str": "237918251", "profile_background_color": "C0DEED", "listed_count": 13, "profile_background_image_url_https": "https://abs.twimg.com/images/themes/theme1/bg.png", "utc_offset": null, "statuses_count": 120, "description": "I am a postdoctoral researcher @PennCIS, studying Natural Language Processing and Social Media.", "friends_count": 166, "location": "Philadelphia PA", "profile_link_color": "0084B4", "profile_image_url": "http://pbs.twimg.com/profile_images/527088456967544832/DnclpoZO_normal.jpeg", "following": null, "geo_enabled": true, "profile_background_image_url": "http://abs.twimg.com/images/themes/theme1/bg.png", "name": "Wei Xu", "lang": "en", "profile_background_tile": false, "favourites_count": 88, "screen_name": "cocoweixu", "notifications": null, "url": "http://www.cis.upenn.edu/~xwe/", "created_at": "Thu Jan 13 23:15:12 +0000 2011", "contributors_enabled": false, "time_zone": null, "protected": false, "default_profile": true, "is_translator": false}, "filter_level": "low", "geo": null, "id": 616333141884674048, "favorite_count": 0, "lang": "en", "entities": {"user_mentions": [], "symbols": [], "trends": [], "hashtags": [{"indices": [0, 4], "text": "CFP"}, {"indices": [124, 131], "text": "NLProc"}, {"indices": [132, 139], "text": "WNUT15"}], "urls": [{"url": "http://t.co/rcygyEowqH", "indices": [99, 121], "expanded_url": "http://noisy-text.github.io", "display_url": "noisy-text.github.io"}]}, "in_reply_to_user_id_str": null, "retweeted": false, "coordinates": null, "timestamp_ms": "1435780246598", "source": "<a href=\"http://twitter.com\" rel=\"nofollow\">Twitter Web Client</a>", "in_reply_to_status_id_str": null, "in_reply_to_screen_name": null, "id_str": "616333141884674048", "place": null, "retweet_count": 0, "created_at": "Wed Jul 01 19:50:46 +0000 2015", "in_reply_to_user_id": null}
```


Slide derivado de Wei Xu

Utilizando APIs

Formato JSON

```
{ "Alunos" : [
        { "nome": "João", "notas": [ 8, 9, 7 ]  },
        { "nome": "Maria", "notas": [ 8, 10, 7 ] },
        { "nome": "Pedro", "notas": [ 10, 10, 9 ] }
    ]
}
```

Visualização da hierarquia

Utilizando APIs

Outras opções para o Twitter, por exemplo:

http://www.tweepy.org/

Utilizando APIs

Processando um JSON

Utilizando APIs

Exemplo: JSON referente a um *tweet*

pip install simplejson

```
import simplejson as json

# Arquivo com Tweets
tweets_file = open('tweet.txt', "r")

#le a linha do arquivo
tweet_json = tweets_file.readline()

#imprime a linha lida
print  tweet_json

#remove espacos em branco
strippedJson = tweet_json.strip()

#converte uma string json em um objeto python
tweet = json.loads(strippedJson)

print tweet['id'] # ID do tweet
print tweet['created_at'] # data de postagem
print tweet['text'] # texto do tweet
>>        >>        >>
print tweet['user']['id'] # id do usuario que postou
print tweet['user']['name'] # nome do usuario
print tweet['user']['screen_name'] # nome da conta do usuario
```

coletaAPIs/streamingTwitter/processaExemplo1/

Slide derivado de Wei Xu

Utilizando APIs

Exemplo: JSON referente a um *tweet*

```
import simplejson as json

# Arquivo com Tweets
tweets_file = open('tweet.txt', "r")

#le a linha do arquivo
tweet_json = tweets_file.readline()

#imprime a linha lida
print  tweet_json

#remove espacos em branco
strippedJson = tweet_json.strip()

#converte uma string json em um objeto python
tweet = json.loads(strippedJson)

print tweet['id'] # ID do tweet
print tweet['created_at'] # data de postagem
print tweet['text'] # texto do tweet
>>        >>        >>
print tweet['user']['id'] # id do usuario que postou
print tweet['user']['name'] # nome do usuario
print tweet['user']['screen_name'] # nome da conta do usuario
```

coletaAPIs/streamingTwitter/processaExemplo1/

Acessando campos do json

Slide derivado de Wei Xu

Utilizando APIs

Exemplo: JSON referente a um *tweet*


```
import simplejson as json

# Arquivo com Tweets
tweets_file = open('tweet.txt', "r")

#le a linha do arquivo
tweet_json = tweets_file.readline()

#imprime a linha lida
print  tweet_json

#remove espacos em branco
strippedJson = tweet_json.strip()

#converte uma string json em um objeto python
tweet = json.loads(strippedJson)

print tweet['id'] # ID do tweet
print tweet['created_at'] # data de postagem
print tweet['text'] # texto do tweet
>>        >>        >>
print tweet['user']['id'] # id do usuario que postou
print tweet['user']['name'] # nome do usuario
print tweet['user']['screen_name'] # nome da conta do usuario
```

```
616333141884674048
Wed Jul 01 19:50:46 +0000 2015
#CFP Workshop on Noisy User-generated Text
at ACL - Beijing 31 July 2015. Papers due:
11 May 2015. http://t.co/rcygyEowqH
#NLProc #WNUT15
237918251
Wei Xu
cocoweixu
```

Slide derivado de Wei Xu

Utilizando APIs

Exemplo: JSON referente a um *tweet*


```
import simplejson as json

# Arquivo com Tweets
tweets_file = open('tweet.txt', "r")

#le a linha do arquivo
tweet_json = tweets_file.readline()

#imprime a linha lida
print  tweet_json

#remove espacos em branco
strippedJson = tweet_json.strip()

#converte uma string json em um objeto python
tweet = json.loads(strippedJson)

print tweet['id'] # ID do tweet
print tweet['created_at'] # data de postagem
print tweet['text'] # texto do tweet
>>          >>          >>
print tweet['user']['id'] # id do usuario que postou
print tweet['user']['name'] # nome do usuario
print tweet['user']['screen_name'] # nome da conta do usuario
```

```
616333141884674048
Wed Jul 01 19:50:46 +0000 2015
#CFP Workshop on Noisy User-generated Text
at ACL - Beijing 31 July 2015. Papers due:
11 May 2015. http://t.co/rcygyEowqH
#NLProc #WNUT15
237918251
Wei Xu
cocoweixu
```

Processando todos os *tweets* retornados em saidaColetaStream.txt.

coletaAPIs/streamingTwitter/processaExemplo2/

Slide derivado de Wei Xu

Utilizando APIs

Personalização da coleta

```
r = api.request('statuses/filter', {'locations':'-74,40,-73,41'})
for item in r:
        print item
```

Longitude e latitude da área: NYC
Tweets geolocalizados nessa área

Parameter value	Tracks Tweets from...
-122.75,36.8,-121.75,37.8	San Francisco
-74,40,-73,41	New York City
-122.75,36.8,-121.75,37.8,-74,40,-73,41	San Francisco OR New York City

Utilizando APIs

Personalização da coleta

```
r = api.request('statuses/filter', {'locations':'-74,40,-73,41'})
for item in r:
        print item
```

Longitude e latitude da área: NYC
Tweets geolocalizados nessa área

Parameter value	Tracks Tweets from...
-122.75,36.8,-121.75,37.8	San Francisco
-74,40,-73,41	New York City
-122.75,36.8,-121.75,37.8,-74,40,-73,41	San Francisco OR New York City

Combinação

```
track=twitter&locations=-122.75,36.8,-121.75,37.8
```

Palavra “twitter” **ou** tweets de San Fran

Assim será retornado tweets fora da área com a palavra Twitter

Utilizando os dados

Exemplo de saída

id do usuário

tempo

latitude

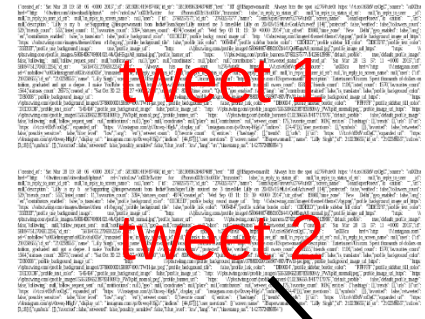
```
296298186       2014-05-01T18:15:09Z    40.87700034
29.23510051     I'm at Başkent Cafe»    4sq.com/1mhXj6u
```

longitude

texto informado

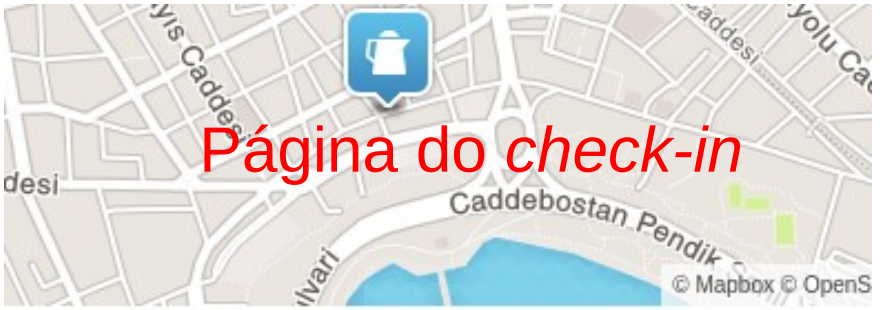
Utilizando os dados

Exemplo de saída



tweet 1

tweet 2



Página do *check-in*

Ümit Yıldız checked in at **Başkent Cafe**

Pendik, Istanbul | May 1, 2014 via Foursquare for Android

id do usuário

tempo

latitude

```
296298186       2014-05-01T18:15:09Z    40.87700034
29.23510051     I'm at Başkent Cafe»    4sq.com/1mhXj6u
```

longitude

texto informado

Utilizando os dados

Exemplo de saída

id do usuário

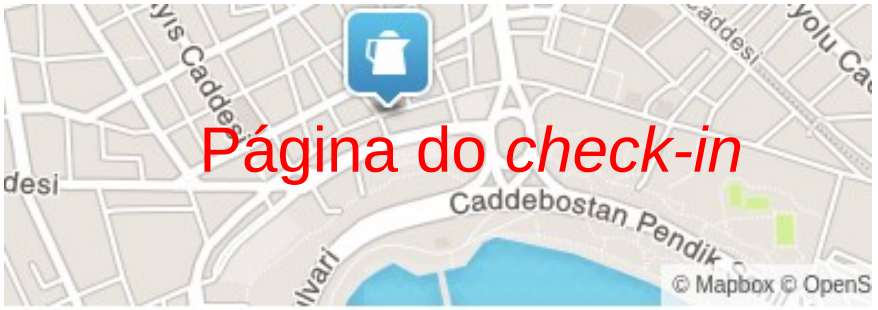
tempo

latitude

```
296298186       2014-05-01T18:15:09Z    40.87700034
29.23510051     I'm at Başkent Cafe     4sq.com/1mhXj6u
```

longitude

texto informado



Página do *check-in*

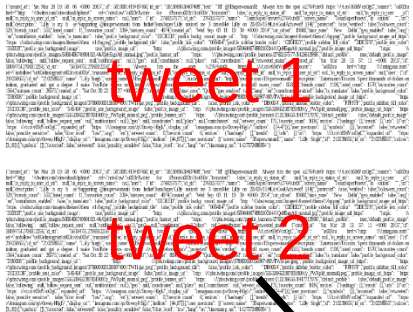
Ümit Yıldız checked in at **Başkent Cafe**

Pendik, Istanbul | May 1, 2014 via Foursquare for Android

Local (pertence à categoria café)

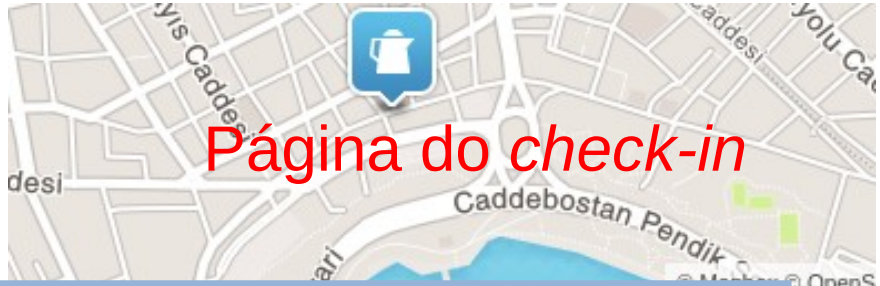
Utilizando os dados

Exemplo de saída



tweet 1

tweet 2



Página do *check-in*

Web *crawler*

id do usuário

tempo

latitude

296298186 2014-05-01T18:15:09Z 40.87700034

29.23510051 I'm at Başkent Cafe 4sq.com/1mhXj6u

longitude

texto informado

Local (pertence à categoria café)

Web *crawler*

Programas que analisam páginas Web em busca do conteúdo desejado

- Alternativa para quando não existe API
- Existem várias estratégias

Coleta com Web *Crawler*

```
# -*- encoding: utf-8 -*-

import urllib

#url obtida através do tweet
url = "http://www.utfpr.edu.br"

pagina = urllib.urlopen(url).read()

print(pagina)

fsaida = open('paginaColetada.html','w')
fsaida.write(str(pagina))
```

Demonstração
coletaCrawler/ColetaPaginaSimples.py

Coleta com Web *Crawler*

```
# -*- encoding: utf-8 -*-

import urllib

#url obtida através do tweet
url = "http://www.utfpr.edu.br"

pagina = urllib.urlopen(url).read()

print(pagina)

fsaida = open('paginaColetada.html','w')
fsaida.write(str(pagina))
```

Saída do código

```
rt" class="primeiro"><a href="http://www.utfpr.edu.br">Reitoria</a></li>
ap"><a href="http://www.utfpr.edu.br/apucarana">Apucarana</a></li>
cm"><a href="http://www.utfpr.edu.br/campomourao">Campo Mourão</a></li>
cp"><a href="http://www.utfpr.edu.br/cornelioprocopio">Cornélio Procópio</a></
ct"><a href="http://www.utfpr.edu.br/curitiba">Curitiba</a></li>
dv"><a href="http://www.utfpr.edu.br/doisvizinhos">Dois Vizinhos</a></li>
fb"><a href="http://www.utfpr.edu.br/franciscobeltrao">Francisco Beltrão</a></
gp" class="primeiro"><a href="http://www.utfpr.edu.br/guarapuava">Guarapuava</
ld"><a href="http://www.utfpr.edu.br/londrina">Londrina</a></li>
md"><a href="http://www.utfpr.edu.br/medianeira">Medianeira</a></li>
pb"><a href="http://www.utfpr.edu.br/patobranco">Pato Branco</a></li>
pg"><a href="http://www.utfpr.edu.br/pontagrossa">Ponta Grossa</a></li>
sh"><a href="http://www.utfpr.edu.br/santahelena">Santa Helena</a></li>
td"><a href="http://www.utfpr.edu.br/toledo">Toledo</a></li>
```

Exemplo com a biblioteca BeautifulSoup

BeautifulSoup para copiar dados de páginas Web

Para instalá-la, execute o comando:

```
pip install beautifulsoup4
sudo apt-get install python-bs4
```

Coleta com Web *Crawler*

```
from bs4 import BeautifulSoup

pagina = '<html><body><p>Bom dia, Fafich!</p></body></html>'
soup = BeautifulSoup(pagina)

allP = soup.find_all('p')
print allP

print allP[0].text
```

Reconhece a *tag* do elemento desejado

Extrai o texto

```
[<p>Bom dia, Fafich!</p>]
Bom dia, Fafich!
```

coletaCrawler/beautifulSoup/example2.py

Coleta com Web *Crawler*

```
from bs4 import BeautifulSoup

pagina = "<table><tr><td>one</td><td>two</td></tr></table>"
soup = BeautifulSoup(pagina)

allTR = soup.find_all('tr')
print allTR
```

→ [<tr><td>one</td><td>two</td></tr>]

```
allTD = soup.find_all('td')
print allTD
```

→ [<td>one</td>, <td>two</td>]

```
for t in soup.find_all('td'):
  print t
```

→ <td>one</td>
<td>two</td>

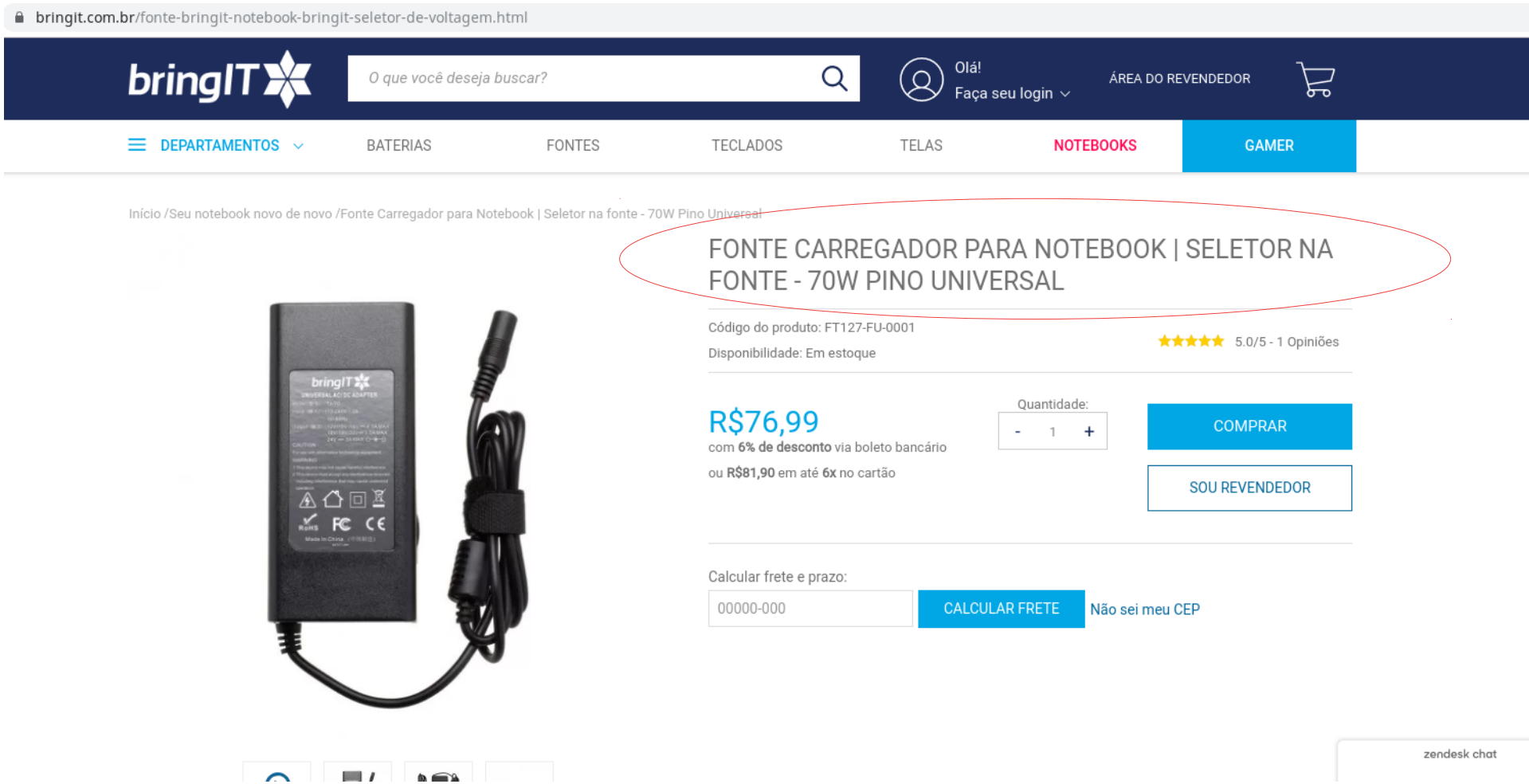
```
for t in soup.find_all('td'):
  print t.text
```

→ one
two

coletaCrawler/beautifulSoup/example3.py

Coleta com Web *Crawler*

https://www.bringit.com.br/fonte-bringit-notebook-bringit-seletor-de-voltagem.html



Pegar o nome do produto

Coleta com Web *Crawler*

```
import requests
from bs4 import BeautifulSoup


##pagina do produto
url = "https://www.bringit.com.br/fonte-bringit-notebook-bringit-seletor-de-voltagem.html"

##retorna o conteudo da pagina
req = requests.get(url)

##transforma o conteudo da pagina em um objeto BeautifulSoup
soup = BeautifulSoup(req.content,'html.parser')

nomeBruto = soup.find("div",{"class":"product-name"})


print nomeBruto.text
```

Faça um código para pegar o preço

coletaCrawler/beautifulSoup/bringit/

Coleta com Web *Crawler*

Algumas informações podem não ser coletáveis com o Beautifulsoup

O uso do Selenium pode ser demandado

Ver exemplo em: coletaCrawler/selenium/exemploSelenium.py

Coleta com Web *Crawler*

```
from selenium import webdriver
from selenium.webdriver.chrome.options import Options
from selenium.webdriver.support.ui import WebDriverWait
from selenium.webdriver.support import expected_conditions as ec
from selenium.webdriver.common.by import By

from bs4 import BeautifulSoup


driver = webdriver.Chrome()


driver.get("https://www.w3schools.com/xml/ajax_intro.asp")

#Clica no botao da pagina para ver o conteudo
botaolist = driver.find_elements_by_xpath('//*[@id="demo"]/button')
botaolist[0].click()

wait = WebDriverWait(driver, 10)
wait.until(ec.visibility_of_element_located((By.XPATH, '//*[@id="demo"]/h1')))

# get the page source
page_source = driver.page_source

#fecha o driver, mas quando estivermos coletando varias paginas podemos manter ativo pra nao precisar abrir o browser novamente
#driver.close()

soup = BeautifulSoup(page_source, "lxml") #grab the content with beautifulsoup for parsing

content= soup.find("div",{"id":"demo"})
print content
```