



Raspagem de Dados para mulheres



26 de novembro,
das 9h às 17h

Inscrições pelo
sympa.com.br/PyLadiesSP

Apoio:



O que é?

Raspagem de dados ou web scraping é a coleta automatizada de dados na Internet.

Há uma variedade de técnicas e tecnologias para fazer isso.

- Para coleta de dados **não estruturados** como **HTML** que misturam dados e apresentação:
 - Uso de bibliotecas para extrair os dados do HTML (BeautifulSoup, Scrapy, ...)
- Para coleta de dados **estruturados** em formato **XML ou JSON**:
 - Uso de APIs disponibilizadas pelos provedores dos dados
 - Uso de biblioteca para manipular json ou xml

O que fazer com os dados coletados?

1. armazenar em arquivos, banco de dados, ...
2. analisar
3. apresentar (graficamente)

HTML

(escreva em um arquivo: ws_html.py)

```
from urllib.request import urlopen

# url de algum site
url = 'http://www.bbc.com/mundo'

# acessando a url e atribuindo a uma variável
# response é do tipo http.client.HTTPResponse (<http.client.HTTPResponse object at
0x0316F470>)
response = urlopen(url)

# resultado atribuido a resultado
resultado = response.read().decode('utf-8')

# gravar o resultado em um arquivo
open('resultado_html.txt', 'w', encoding='utf-8').write(resultado)

# gravar o resultado em um arquivo
open('resultado_html.html', 'w', encoding='utf-8').write(resultado)
```

Analisar o Resultado da Coleta de HTML

- Abra o arquivo resultado_html.html
- Abra o arquivo resultado_html.txt

O resultado_html.html mostrará o conteúdo visual do que foi coletado, pois o HTML é um formato de apresentação.

O resultado_html.txt mostrará o código HTML do mesmo conteúdo.

O resultado não contém dados estruturados, pois o HTML mistura dados + apresentação.

Desta forma fica difícil extrair os dados.

BeautifulSoup

biblioteca para manipular o código HTML

é necessário instalá-la:

```
pip install beautifulsoup4
```

Como importar:

```
from bs4 import BeautifulSoup
```

BeautifulSoup - exemplo

```
from urllib.request import urlopen
from bs4 import BeautifulSoup

html = urlopen('http://www.python.org/')
bsObj =
BeautifulSoup(html.read(), 'html.parser')
print(bsObj.h1)
```

```
$ terminal: python arquivo.py
<h1 class="site-headline">
<a href="/"></a>
</h1>
```


Scrapy

“[...] biblioteca Python que manipula grande parte da complexidade de busca e avaliação dos links de um site, rastreando domínios ou lista de domínios com facilidade.”

Para instalar:

```
pip install scrapy
```

Como importar:

```
import scrapy
```

Scrapy - Exemplo

```
import scrapy

class SpiderSimples(scrapy.Spider):
    name = 'meuspider'
    start_urls = ['http://example.com']

    def parse(self, response):
        self.log('Visitei o site: %s' % response.url)
        yield 'url': response.url, 'tamanho' len(response.body)}
        proxima_url = 'http://www.google.com.br'
        self.log('Agora vou para: %s' % proxima_url)
        yield scrapy.Request(proxima_url, self.handle_google)

    def handle_google(self, response):
        self.log('Visitei o google via URL: %s' % response.url)
```

Scrapy - Exemplo



Para rodar, utilize o comando:

```
$ terminal:  
scrapy runspider arquivo.py
```

RSS (XML)

(escreva em um arquivo: ws_rss.py)

```
from urllib.request import urlopen

# url de algum site
url = 'http://feeds.bbc.co.uk/news/world/rss.xml'

# acessando a url e atribuindo a uma variável
# response é do tipo http.client.HTTPResponse (<http.client.HTTPResponse object at
0x0316F470>)
response = urlopen(url)

# resultado atribuido a resultado
resultado = response.read().decode('utf-8')

# gravar o resultado em um arquivo
open('resultado_rss.txt', 'w', encoding='utf-8').write(resultado)
# gravar o resultado em um arquivo
open('resultado_rss.xml', 'w', encoding='utf-8').write(resultado)
```

Analisar o Resultado da Coleta de rss.xml

- Abra o arquivo resultado_rss.xml
- Abra o arquivo resultado_rss.txt

O resultado_rss.xml mostrará o conteúdo visual do que foi coletado, pois o rss.xml é um padrão que pode ser “renderizado” pelo navegador.

O resultado_rss.txt mostrará o código rss.xml do mesmo conteúdo.

O resultado contém dados estruturados no padrão rss.



```
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet title="XSL formatting" type="text/xsl" href="/shared/bsp/xsl/rss/nolsol.xsl"?>
<rss xmlns:dc="http://purl.org/dc/elements/1.1/" xmlns:content="http://purl.org/rss/1.0/modules/content/" xmlns:atom="http://www.w3.org/2005/Atom" version="2.0" x
<channel>
  <title><![CDATA[BBC News - World]]></title>
  <description><![CDATA[BBC News - World]]></description>
  <link>http://www.bbc.co.uk/news/</link>
  <image>
    <url>http://news.bbcimg.co.uk/nol/shared/img/bbc_news_120x60.gif</url>
    <title>BBC News - World</title>
    <link>http://www.bbc.co.uk/news/</link>
  </image>
  <generator>RSS for Node</generator>
  <lastBuildDate>Sat, 19 Nov 2016 19:43:07 GMT</lastBuildDate>
  <copyright><![CDATA[Copyright: (C) British Broadcasting Corporation, see http://news.bbc.co.uk/2/hi/help/rss/4498287.stm for terms and conditions of reuse
  <language><![CDATA[en-gb]]></language>
  <ttl>15</ttl>
  <item>
    <title><![CDATA[Syria conflict: Aleppo hospitals 'knocked out by bombardment']]></title>
    <description><![CDATA[Days of strikes on Aleppo leave rebel areas virtually without a functioning hospital, reports say.]]></description>
    <link>http://www.bbc.co.uk/news/world-middle-east-38039282</link>
    <guid isPermaLink="true">http://www.bbc.co.uk/news/world-middle-east-38039282</guid>
    <pubDate>Sat, 19 Nov 2016 19:10:20 GMT</pubDate>
    <media:thumbnail width="976" height="549" url="http://c.files.bbc.co.uk/3166/production/_92564621_036476169-1.jpg"/>
  </item>
  <item>
    <title><![CDATA[Pope Francis names 17 new cardinals of Roman Catholic Church]]></title>
    <description><![CDATA[Pope Francis names 17 new cardinals from around the world, many of whom will help name his successor.]]></description>
    <link>http://www.bbc.co.uk/news/world-europe-38038501</link>
    <guid isPermaLink="true">http://www.bbc.co.uk/news/world-europe-38038501</guid>
    <pubDate>Sat, 19 Nov 2016 13:00:42 GMT</pubDate>
    <media:thumbnail width="976" height="549" url="http://c.files.bbc.co.uk/1350C/production/_92561197_mediaitem92561195.jpg"/>
  </item>
  <item>
    <title><![CDATA[Trump locks horns with Hamilton musical cast over Pence booing]]></title>
    <description><![CDATA[Donald Trump wades into a row over the booing of the US vice-president elect at a Broadway musical.]]></description>
    <link>http://www.bbc.co.uk/news/world-us-canada-38039286</link>
    <guid isPermaLink="true">http://www.bbc.co.uk/news/world-us-canada-38039286</guid>
```

Refatorar `ws_html.py` e `ws_rss.py`

Temos dois arquivos com conteúdo muito parecido.

Podemos melhorar o código.

- transformar as instruções em funções
- juntar em um módulo o que é em comum entre os dois arquivos
- separar o que é diferente


```
# arquivo com instruções em comum: ws.py
```



```
from urllib.request import urlopen
```

```
def coletar_dados(url):  
    # acessando a url e atribuindo a um variável  
    # response é do tipo http.client.HTTPResponse  
(<http.client.HTTPResponse object at 0x0316F470>)  
    response = urlopen(url)  
  
    # resultado atribuido a resultado  
    return response.read().decode('utf-8')
```

```
def guardar_dados(nome_arquivo, resultado):  
    # gravar o resultado em um arquivo  
    open(nome_arquivo, 'w', encoding='utf-8').write(resultado)
```

```
# arquivo com instruções em comum: ws.py (parte 2)
def atualizar_arquivo(nome_arquivo, resultado):
    # gravar o resultado em um arquivo
    open(nome_arquivo, 'a', encoding='utf-8').write(resultado)
```

```
# arquivo com instruções específicas: ws_html.py
import ws

url = 'http://www.bbc.com/mundo'

dados = ws.coletar_dados(url)
ws.guardar_dados('resultado_html.html', dados)
```

```
# arquivo com instruções específicas: ws_rss.py
```

```
import ws
```

```
url =
```

```
'http://feeds.bbc.co.uk/news/world/rss.xml'
```

```
dados = ws.coletar_dados(url)
```

```
ws.guardar_dados('resultado_rss.html', dados)
```

APIs



Os fornecedores dos dados disponibilizam APIs (Application Programming Interfaces) para facilitar a coleta dos dados.

Geralmente elas requerem autenticação, ou seja, uma API key e access token.

Dados abertos: educação no Brasil

Exemplo: <http://educacao.dadosabertosbr.com/api>

<http://educacao.dadosabertosbr.com/api/docs/%2Fapi%2Fescolas%2Fbuscaavancada>

<http://educacao.dadosabertosbr.com/api/escolas/buscaavancada?situacaoFuncionamento=1>

```
[ ▼ 2 items, 34 KB
195656,
[ ▼ 100 items, 34 KB
{ ▼ 14 properties, 357 bytes
  "anoCenso": 2013,
  "cod": 31315001,
  "nome": "-ASSOCOACAO ASSISTENCIAL CRECHE JARDIM SANTA CECILIA",
  "codCidade": 3136702,
  "cidade": "JUIZ DE FORA",
  "estado": "MG",
  "regiao": "Sudeste",
  "situacaoFuncionamento": 1,
  "dependenciaAdministrativa": 4,
  "idebAI": 0.0,
  "idebAF": 0.0,
  "enemMediaGeral": 0.0,
  "dependenciaAdministrativaTxt": "Privada",
  "situacaoFuncionamentoTxt": "Em atividade"
},
{ ▼ 14 properties, 371 bytes
  "anoCenso": 2013,
  "cod": 33062501,
  "nome": "0101001 ESCOLA MUNICIPAL VICENTE LICINIO CARDOSO",
  "codCidade": 3304557,
  "cidade": "RIO DE JANEIRO",
  "estado": "RJ",
  "regiao": "Sudeste",
  "situacaoFuncionamento": 1,
  "dependenciaAdministrativa": 3,
  "idebAI": 0.0,
  "idebAF": 4.900000095367432,
  "enemMediaGeral": 0.0,
  "dependenciaAdministrativaTxt": "Municipal",
  "situacaoFuncionamentoTxt": "Em atividade"
},
```

```
import csv
```

```
url =
```

```
'http://educacao.dadosabertosbr.com/api/escolas/buscaavancada?s  
ituacaoFuncionamento=1'
```

```
dados = ws.coletar_dados(url)
```

```
dados_escolas = json.loads(dados)
```

```
quantidade = dados_escolas [0]
```

```
escolas = dados_escolas [1]
```



```
import csv
with open('escolas.csv', 'w', newline= '') as csvfile:
    fieldnames = ['nome', 'cidade', 'estado',
'dependenciaAdministrativa', 'enemMediaGeral']
    writer = csv.DictWriter(csvfile, fieldnames=fieldnames)
    writer.writeheader()
    for escola in escolas:
        spamwriter.writerow(escola)
```

Facebook API - Preparação

<https://developers.facebook.com/>

Entrar

Adicionar um aplicativo

Configurar o aplicativo

Obter um ID do aplicativo

Sobre a API:

<https://developers.facebook.com/docs/graph-api>

Facebook - Configurações

Browser address bar: <https://developers.facebook.com/apps/1410506405633782/settings/k>

Search bar: :ebook api bad request

Left sidebar:

- wsaula - Test1
- Painel
- Configurações
 - Básico
 - Avançado
- Funções

Top navigation:

- ID do APLICATIVO: 1410506405633782
- Visualizar o Analytics
- Ferramentas e suporte
- Documentos

Main content area:

Info Você está editando uma versão de teste do **wsaula**

ID do Aplicativo	Chave Secreta do Aplicativo

Right sidebar menu:

- Ver todos os aplicativos
- Adicionar um novo aplicativo
- Solicitações
- Configurações do desenvolvedor
- Configurações da empresa
- Sair do Facebook

Facebook - Configurações



Configurações
Básico
Avançado

ID do Aplicativo
1522873017729918

Chave Secreta do Aplicativo
b051937827b5a8de756121263bd76f08
[Redefinir](#)

Facebook - Configurações

Configurações

Básico

Avançado

☐ Não

Aplicativo nativo ou para desktop?

Ative se seu aplicativo for nativo ou um aplicativo para desktop

ws_facebook_v1.py

Script para:

- obtenção do token de acesso
- consulta de posts

```
# arquivo instruções específicas: ws_facebook_v1.py (parte 1)
```

```
import ws
```

```
# configurado no facebook
```

```
app_id = '1522873017729918'
```

```
app_secret = 'b051937827b5a8de756121263bd76f08'
```

```
url_para_obter_access_token =
```

```
'https://graph.facebook.com/oauth/access_token?client_id={app_id}&client_secret={app_secret}&grant_type=client_credentials'.format(
```

```
    app_id=app_id,
```

```
    app_secret=app_secret)
```

```
# coletar os dados
```

```
access_token = ws.coletar_dados(url_para_obter_access_token)
```

```
print(access_token)
```

```
# arquivo instruções específicas: ws_facebook_v1.py (parte 2)
api_url = 'https://graph.facebook.com/'
page_id = 'PyLadiesSP'
url = api_url + page_id + '?fields=posts&' + access_token

print(url)
dados = ws.coletar_dados(url)
ws.guardar_dados('resultado_facebook.json', dados)
```


ws_facebook_v2.py

- criar uma função para a obtenção do token
- criar uma função para gerar a url para consultar posts de qualquer usuário do Facebook

```
# arquivo ws_facebook_v2.py (parte 1)
import ws
```

```
def obter_access_token():
    # configurado no facebook
    app_id = '1522873017729918'
    app_secret = 'b051937827b5a8de756121263bd76f08'

    url_para_obter_access_token =
    'https://graph.facebook.com/oauth/access_token?client_id={app_id}
    &client_secret={app_secret}&grant_type=client_credentials'.format(
        app_id=app_id,
        app_secret=app_secret)
    # coletar os dados
    return ws.coletar_dados(url_para_obter_access_token)
```

```
# ws_facebook_v2.py (parte 2)
api_url = 'https://graph.facebook.com/'

def url_para_obter_posts(page_id, access_token):
    return api_url + page_id + '?fields=posts&' + access_token

access_token = obter_access_token()

# posts de PyLadiesSP
url = url_para_obter_posts('PyLadiesSP', access_token)
dados = ws.coletar_dados(url)
ws.guardar_dados('resultado_facebook_pyladiessp.json', dados)

# posts de Ideias Incríveis
url = url_para_obter_posts('ideias.incriveis', access_token)
dados = ws.coletar_dados(url)
ws.guardar_dados('resultado_facebook_ideiasincriveis.json',
dados)
```

```
{
  "posts": {
    "data": [
      {
        "message": "As meninas do PyLadies dominaram a Vila das Minas na Roadsec esse ano com workshops, dojo e palestras \u003C3\u003E\n#souPyLadiesSP #roadsec",
        "story": "PyLadies S\u003E Paulo shared PyLadies Brasil's album.",
        "created_time": "2016-11-18T22:22:35+0000",
        "id": "1469656506672137_1615174665453653"
      },
      {
        "message": "J\u003E come\u003E o Workshop de Introdu\u003E Python aqui na Roadsec.\nE s\u003E chegar aqui na \u003E de atividades.);\n#souPyLadiesSP #roadsec",
        "story": "PyLadies S\u003E Paulo at Roadsec.",
        "created_time": "2016-11-18T13:09:35+0000",
        "id": "1469656506672137_1615021138802339"
      },
      {
        "message": "Amanh\u003E tem Roadsec e estamos com v\u003E atividades na Vila das Minas. Confira a programa\u003E\nintrodu\u003E a Python com as PyLadies S\u003E Paulo\n11h - Mesa Redonda: Quem s\u003E as mulheres na computa\u003E com as PyLadies Brasil\n13h - Dojo com as PyLadies S\u003E Carlos\n15h - Workshop de introdu\u003E a Machine Learning com as PyLadies Campinas\n15h - Palestra PyLadies S\u003E Paulo e o que \u003E Python\nE ai quem vamos? \u003E",
        "story": "PyLadies S\u003E Paulo shared Roadsec's photo.",
        "created_time": "2016-11-18T01:21:43+0000",
        "id": "1469656506672137_161486838817614"
      },
      {
        "message": "O Grupy-SP est\u003E completando 9 anos com um mega meetup!!!\nSer\u003E dia 19\/11 e tem muito espa\u003E pra todo mundo poder ir. Inscr\u003E no link abaixo.\nhttp:\/\/\/meetu.ps\/31YNpm",
        "created_time": "2016-11-10T23:54:47+0000",
        "id": "1469656506672137_1612428422394944"
      },
      {
        "message": "Estaremos na Vila das Minas da Roadsec dia 18\/11 com um Workshop de introdu\u003E a Python.\nE ai quem vamos? \u003E",
        "story": "PyLadies S\u003E Paulo shared Roadsec's photo.",
        "created_time": "2016-11-09T17:36:01+0000",
        "id": "1469656506672137_1611899782447808"
      }
    ]
  }
}
```



JSON

`json.loads` converte uma string que contém formato JSON em dicionário.

Assim podemos manipular os dados.

```
import json  
json.loads('["foo", {"bar":["baz", null, 1.0, 2]}]')
```

```
['foo', {'bar': ['baz', None, 1.0, 2]}]
```

```
{ ▼ 2 properties, 11 KB
  "posts": { ▼ 2 properties, 11 KB
    "data": [ ▼ 25 items, 11 KB
      { ▼ 4 properties, 297 bytes
        "message": "As meninas do Pyladies dominaram a Vila da
        "story": "PyLadies S\u00e3o Paulo shared PyLadies Bras
        "created_time": "2016-11-18T22:22:35+0000",
        "id": "1469656506672137_1615174665453653"
      },
      { ▼ 4 properties, 310 bytes
        "message": "J\u00e1 come\u00e7ou o Workshop de Introdu
        ;)\n\n#souPyLadiesSP #roadsec",
        "story": "PyLadies S\u00e3o Paulo at Roadsec.",
        "created_time": "2016-11-18T13:09:35+0000",
        "id": "1469656506672137_1615021138802339"
      },
    ],
  },
}
```

```
print(dados['posts']['data'][0]['message'])
As meninas do Pyladies dominaram a Vila da ...
```

```
print(dados['posts']['data'][1]['id'])
1469656506672137_1615021138802339
```

ws_facebook_v3.py

- incluir no início import json
- converter json para dict
- manipular os dados no dicionário
- criar um arquivo que contenha os textos dos posts

```
# arquivo ws_facebook_v3.py (parte 1)
```

```
import json
```

```
import ws
```

```
def obter_access_token():
```

```
    # configurado no facebook
```

```
    app_id = '1522873017729918'
```

```
    app_secret = 'b051937827b5a8de756121263bd76f08'
```

```
    url_para_obter_access_token =
```

```
    'https://graph.facebook.com/oauth/access_token?client_id={app_id}'  
&client_secret={app_secret}&grant_type=client_credentials'.format(  
    app_id=app_id,  
    app_secret=app_secret)
```

```
    # coletar os dados
```

```
    return ws.coletar_dados(url_para_obter_access_token)
```




```
# ws_facebook_v3.py (parte 2)
api_url = 'https://graph.facebook.com/'

def url_para_obter_posts(page_id, access_token):
    return api_url + page_id + '?fields=posts&' + access_token

access_token = obter_access_token()

# posts de PyLadiesSP
url = url_para_obter_posts('PyLadiesSP', access_token)
dados = ws.coletar_dados(url)
ws.guardar_dados('resultado_facebook_pyladiessp.json', dados)

# converte json em dict
posts = json.loads(dados)
```

ws_facebook_v3.py (parte 3)

```
messages = []
for post in posts['posts']['data']:
    messages.append(post.get('message', post.get('story', '')))

ws.guardar_dados('posts.txt', '\n'.join(messages))
```

```
{ ▼ 2 properties, 11 KB
  "posts": { ▼ 2 properties, 11 KB
    "data": [ ▼ 25 items, 11 KB
      { ▼ 4 properties, 297 bytes
        "message": "As meninas do PyLadies dominaram a Vila das Minas na Roadsec esse ano com workshops, dojo e palestras \u003C3\u003E\n#souPyLadiesSP #roadsec",
        "story": "PyLadies S\u003E Paulo shared PyLadies Brasil's album.",
        "created_time": "2016-11-18T22:22:35+0000",
        "id": "1469656506672137_1615174665453653"
      },
      { ▼ 4 properties, 310 bytes
        "message": "J\u003E come\u003E ou o Workshop de Introdu\u003E \u003E Python aqui na Roadsec.\nE s\u003E chegar aqui na \u003E de atividades.
        ;)\n#souPyLadiesSP #roadsec",
        "story": "PyLadies S\u003E Paulo at Roadsec.",
        "created_time": "2016-11-18T13:09:35+0000",
        "id": "1469656506672137_1615021138802339"
      },
      { ▼ 4 properties, 661 bytes
        "message": "Amanh\u003E tem Roadsec e estamos com v\u003E atividades na Vila das Minas. Confira a programa\u003E\n10h - Workshop de
        introdu\u003E a Python com as PyLadies S\u003E Paulo\n11h - Mesa Redonda: Quem s\u003E as mulheres na computa\u003E com as PyLadies
        Brasil\n13h - Dojo com as PyLadies S\u003E Carlos\n15h - Workshop de introdu\u003E a Machine Learning com as PyLadies Campinas\n15h - Palestra
        PyLadies S\u003E Paulo a a sua \u003E Python\u003E e a sua \u003E
      }
    ]
  }
}
```

ws_facebook_v4.py

- criar uma função para obter apenas as mensagens dos posts

```
# ws_facebook_v4.py (parte 2)
```

```
api_url = 'https://graph.facebook.com/'
```

```
def url_para_obter_posts(page_id, access_token):  
    return api_url + page_id + '?fields=posts&' + access_token
```

```
def obter_mensagens(posts):  
    messages = []  
    for post in posts['posts']['data']:  
        messages.append(post.get('message', post.get('story',  
''))))  
    return messages
```

```
# ws_facebook_v4.py (parte 3)

access_token = obter_access_token()

# posts de PyLadiesSP
url = url_para_obter_posts('PyLadiesSP', access_token)

dados = ws.coletar_dados(url)
posts = json.loads(dados)
mensagens = obter_mensagens(posts)
ws.guardar_dados('posts.txt', '\n'.join(mensagens))
```

Facebook API - paginação

A cada consulta de uma API, além dos dados, é também informado quais são os endereços para buscar os resultados anteriores e os posteriores, como se fosse a paginação dos dados.

```
{
  "message": "J\u00e1 come\u00e7ou o Curso de Python Intermedi\u00e1rio \u00c3nHoje \u00e9 nosso 13\u00ba encontro e quem est\u00e1 min\u00e9 a Roberta :D\u00e9nvem tambem.py",
  "created_time": "2016-09-24T14:51:41+0000",
  "id": "1469656506672137_1593778900926563"
},
{
  "paging": {
    "previous": "https://graph.facebook.com/v2.8/1469656506672137/posts?since=1479507755&access_token=1522873017729918|pum50zN48gJTrwzqZ\u0026__paging_token=enc_AdAxiPgNiW0ahSxlvEEosTDbwaamvpHDSOZBcnUZAhf5BDcz2IuBvernKLeVAfe3b50su2B0WKy80KHnzZAWn0WpNpWVUq8DT2vugfYeSUzoPdFKzgZD2D",
    "next": "https://graph.facebook.com/v2.8/1469656506672137/posts?access_token=1522873017729918|pum50zN48gJTrwzqZbF5qE9UEc0&limit=25&__paging_token=enc_AdBWZBp7Qi3zBgkhWkVKCaG6FpUpfpWHe0moLtiIvHJR1ZAFvgNXcQPrfErUGTUCJZACnB2AZCJh4F0ZBkh19I9m27a9O5ZAM39E1uEF3qujFa38TrBjgZ"
  },
  "id": "1469656506672137"
}
```

ws_facebook_v5.py

- repetidas consultas



```
{ ▼ 2 properties, 11 KB
  "posts": { ▼ 2 properties, 11 KB
    "data": [ ▼ 25 items, 11 KB
      { ▼ 4 properties, 297 bytes
        "message": "As meninas do PyLadies dominaram a Vila das Minas na Roadsec esse ano com workshops, dojo e palestras \u003C3\n#souPyLadiesSP #roadsec",
        "story": "PyLadies S\u00e3o Paulo shared PyLadies Brasil's album.",
        "created_time": "2016-11-18T22:22:35+0000",
        "id": "1469656506672137_1615174665453653"
      },
      { ▶ 4 properties, 310 bytes },
      { ▶ 4 properties, 661 bytes },
      { ▶ 3 properties, 293 bytes },
      { ▶ 4 properties, 280 bytes },
      { ▶ 3 properties, 709 bytes },
      { ▶ 4 properties, 767 bytes },
      { ▶ 3 properties, 252 bytes },
      { ▶ 4 properties, 238 bytes },
      { ▶ 4 properties, 774 bytes },
      { ▶ 3 properties, 293 bytes },
      { ▶ 4 properties, 240 bytes },
      { ▶ 3 properties, 150 bytes },
      { ▶ 4 properties, 252 bytes },
      { ▶ 4 properties, 1 KB },
      { ▶ 3 properties, 483 bytes },
      { ▶ 4 properties, 350 bytes },
      { ▶ 3 properties, 181 bytes },
      { ▶ 4 properties, 448 bytes },
      { ▶ 4 properties, 267 bytes },
      { ▶ 4 properties, 575 bytes },
      { ▶ 3 properties, 252 bytes },
      { ▶ 4 properties, 412 bytes },
      { ▶ 4 properties, 660 bytes },
      { ▶ 3 properties, 270 bytes }
    ],
    "paging": { ▼ 2 properties, 603 bytes
      "previous": "https://graph.facebook.com/v2.8/1469656506672137/posts?since=1479507755&access_token=1522873017729918|pum50zN48gJTrwzqZbF5qE9UEc0&limit=25&__paging_token=enc_AdAxiPgniW0ahSxlvEEosTDbwaamvpHDSO2BcnUZAhf5BDcz2IuBvernKLeVAfe3b50suZB0WKy80KHnzZAWnOWpNpWVUq8DT2vugfyYeSUzoPdFKzg2DZD&__previous=1",
      "next": "https://graph.facebook.com/v2.8/1469656506672137/posts?access_token=1522873017729918|pum50zN48gJTrwzqZbF5qE9UEc0&limit=25&until=14747287016
```



```
# ws_facebook_v5.py (parte 3)
access_token = obter_access_token()

# posts de PyLadiesSP
url = url_para_obter_posts('PyLadiesSP', access_token)
condicao = True
while condicao is True:
    dados = ws.coletar_dados(url)
    posts = json.loads(dados)
    mensagens = obter_mensagens(posts)
    ws.atualizar_arquivo('posts.txt', '\n'.join(mensagens))
    url = posts['posts']['paging']['next']
    condicao = ???
```

???? = vamos pensar quais poderiam ser as condições de parada?

Facebook APIs


Tudo tem um id para conseguir obter os dados associados.

A partir do id de um post podemos obter os dados do post, como por exemplo likes:

https://graph.facebook.com/1469656506672137_1615021138802339?fields=likes&access_token=...

Para saber quais são as várias possibilidades, consulte o Graph API Explorer:

<https://developers.facebook.com/tools/explorer>

Token de acesso:  EAACEdEose0cBAOe3Puzo3FFfQWOwmgVZBwYY7ABaLhZAdgOOIUENo4zkqQc1GmLTmgaLAnYSGvqGtBSxkqVZCDAOHMrfYIk2H9 Get Token ▾

GET ▾ → /v2.8 ▾ /1469656506672137_1615021138802339?fields=likes



▶ Submit

Saiba mais sobre a sintaxe da Graph API

Node: 1469656506672137_1615021

☒ likes

+ Search for a field

+ Search for a field

```
{
  "likes": {
    "data": [
      {
        "id": "764628623592259",
        "name": "Leila Aparecida da Silva"
      },
      {
        "id": "840362745983320",
        "name": "Caroline Dantas"
      },
      {
        "id": "770530686319690",
        "name": "Eliza Ramos"
      },
      {
        "id": "761379187251469",
        "name": "Débora Fernandes"
      },
      {
        "id": "1106573342750500",
        "name": "Ariadyne Oliveira"
      },
      {
        "id": "893251710693095",
        "name": "Patricia Novais"
      }
    ]
  }
}
```

ws_facebook_v6.py

- criar uma função para fazer outras consultas a partir dos posts, por exemplo: likes de cada post.

```
# ws_facebook_v6.py (parte 3)
```

```
def url_para_coletar_likes_mais_recentes_de_um_post(postid):  
    return  
    'https://graph.facebook.com/{postid?fields=likes&{access_token}  
'  
    .format(postid=postid, access_token=access_token)
```

```
def coletar_likes_mais_recentes_de_um_post(post):  
    url = url_para_coletar_likes_mais_recentes_de_um_post(  
        post['id'])  
    resultado_likes = coletar_dados(url)  
    return json.loads(resultado_likes)
```

```
def obter_likes(likes):  
    nomes = []  
    for like in likes['likes']['data']:  
        nomes.append(like['name'])  
    return nomes
```

```
# ws_facebook_v6.py (parte 4)
def post_e_seus_likes(post):
    message = post.get('message', post.get('story', ''))
    likes = coletar_likes_mais_recentes_de_um_post(post)
    return (message, obter_likes(likes))
```

```
# ws_facebook_v6.py (parte 4)
access_token = obter_access_token()

# posts de PyLadiesSP
url = url_para_obter_posts('PyLadiesSP', access_token)
condicao = True
while condicao is True:
    dados = ws.coletar_dados(url)
    posts = json.loads(dados)
    for post in posts['posts']['data']:
        msg, likes = post_e_seus_likes(post)
        msg_e_likes = msg+'\n'+'\n'.join(likes)+'\n'+'-'*3+'\n'
        ws.atualizar_arquivo('posts.txt', msg_e_likes)
    url = posts['posts']['paging']['next']
    condicao = ????
```

???? = vamos pensar quais poderiam ser as condições de parada?

As meninas do PyLadies dominaram a Vila das Minas na Roadsec esse ano com workshops, dojo e palestras <3

#souPyLadiesSP #roadsec

Naomi Ceder; Daniele Marighetti; Greyce Carla; Fátima Sousa; Patricia Leite; Raquel Barros; Priscila Viana; Eliza Ramos; Juliana Neres; Mazinho Web; Luiz Menezes Filho; Camila Lopes; Patty Vader; Marcelo Yamaki; PyLadies São Paulo; Bruno Rocha;

Já começou o Workshop de Introdução à Python aqui na Roadsec.
E só chegar aqui na área de atividades. ;)

#souPyLadiesSP #roadsec

Leila Aparecida da Silva; Caroline Dantas; Eliza Ramos; Débora Fernandes; Ariadyne Oliveira; Patricia Novais; Mazinho Web; Marcelo Yamaki; Débora Oliveira; Juliana Neres; Natalia Zaramelo de Campos;

Amanhã tem Roadsec e estamos com várias atividades na Vila das Minas. Confira a programação:

10h - Workshop de introdução a Python com as PyLadies São Paulo

11h - Mesa Redonda: Quem são as mulheres na computação com as PyLadies Brasil

13h - Dojo com as PyLadies São Carlos

15h - Workshop de introdução a Machine Learning com as PyLadies Campinas

15h - Palestra PyLadies São Paulo e o que é Python

E aí quem vamos? <3

Vanessa de Moraes; Patricia Novais; Marcelo Yamaki; Leila Aparecida da Silva; Stefanie Souza; Júlia de Almeida; PyLadies Campinas; Mazinho Web; Isis Ramos;

Twitter APIs



- REST (tweets recentes, últimos 7 dias)
- Streaming (tempo real)

```
$ terminal: pip install TwitterAPI
```

Twitter APIs - Preparação



- Criar um arquivo chamado: pizza.py
- Entrar em: <https://apps.twitter.com>
- Clicar em Create New App
- Criar o app (preencher as informações)

Twitter APIs - Preparação

Create an application



Application Details

Name *

Your application name. This is used to attribute the source of a tweet and in user-facing authorization screens. 32 characters max.

Description *

Your application description, which will be shown in user-facing authorization screens. Between 10 and 200 characters max.

Website *

Your application's publicly accessible home page, where users can go to download, make use of, or find out more information about your application. This fully-qualified URL is used in the source attribution for tweets created by your application and will be shown in user-facing authorization screens.
(If you don't have a URL yet, just put a placeholder here but remember to change it later.)

Callback URL

Where should we return after successfully authenticating? [OAuth 1.0a](#) applications should explicitly specify their `oauth_callback` URL on the request token step, regardless of the value given here. To restrict your application from using callbacks, leave this field blank.

Developer Agreement

☒ Yes, I have read and agree to the [Twitter Developer Agreement](#).

Twitter APIs - Preparação

- Clicar em 'Manage keys and access tokens'
- Pegar o `consumer_key` e `consumer_secret`
- Mais pra baixo da página terá um botão para gerar o Token, gerar tokens
- Pegar o `access_token` e o `access_token_secret`
- No arquivo `pizza.py` coloque as variáveis acima com seus respectivos valores.

Twitter APIs - REST



```
from TwitterAPI import TwitterAPI, TwitterRestPager
```

```
search_term = 'pizza'
```

```
consumer_key = # sua consumer key
```

```
consumer_secret = # sua consumer secret
```

```
access_token = # sua access token
```

```
access_token_secret = # sua access token secret
```

```
api = TwitterAPI(consumer_key, consumer_secret, access_token,  
access_token_secret)
```

```
resultado = TwitterRestPager(api, 'search/tweets', {'q': search_term})
```

```
for item in resultado.get_iterator():
```

```
    r = item['text'] if 'text' in item else item
```

```
    print(item['created_at'])
```

```
    print(r.encode(encoding=sys.getdefaultencoding()))
```

Twitter APIs - Streaming



- Crie um outro arquivo chamado: mais_pizza.py

Twitter APIs - Streaming



```
from TwitterAPI import TwitterAPI
```

```
search_term = 'pizza'
```

```
consumer_key = # sua consumer key
```

```
consumer_secret = # sua consumer secret
```

```
access_token = # sua access token
```

```
access_token_secret = # sua access token secret
```

```
api = TwitterAPI(consumer_key, consumer_secret, access_token, access_token_secret)
```

```
resultado = api.request('statuses/filter', {'track': search_term})
```

```
for item in resultado:
    try:
        print(item['created_at'])
        print(item['user']['location'])
        print(item['user']['name'])
        print(r)
        print('-')
    except:
        pass
```

<https://gist.githubusercontent.com/hrp/900964/raw/2bbee4c296e6b54877b537144be89f19beff75f4/twitter.json>

Virtualenv - O que é?

Virtualenv é uma ferramenta para criar ambientes Python isolados.

Virtualenv cria uma pasta que contém todos os executáveis necessários para um projeto Python.

Virtualenv - Criação e Ativação (Windows)

1. Cria uma pasta para seu projeto
2. Entre na pasta
3. Crie o ambiente virtual

```
python -m venv .venv
```

4. Ative o ambiente virtual

```
.venv\scripts\activate.bat
```

<https://docs.python.org/3/library/venv.html>