

Sprint 02

João Gabriel

02/05/2017

Resumo

Neste relatório se encontrará a ideia base transmitida durante o capítulo 11 do livro em estudo, com exemplos dos programas de maior relevância e conceitos base para entendimento do assunto abordado.

Parte II:

Web Scraping

1 Webbrowser

Nesta seção será apresentado alguns programas para uma ideia base da função *webbrowser*, que em sua essência é usada para abrir as páginas na web. Precisa ser instalada e pode ser muito bem usada com a *pyperclip*.



```
mapIt.py - C:\Users\João Gabriel\Desktop\Programs\mapIt.py (3.6.1)
File Edit Format Run Options Window Help
#!/python3
# mapIt.py - inicia uma coordenada no browser usando endereços a partir da
# command line ou do clipboard (área de transferência/trabalho)

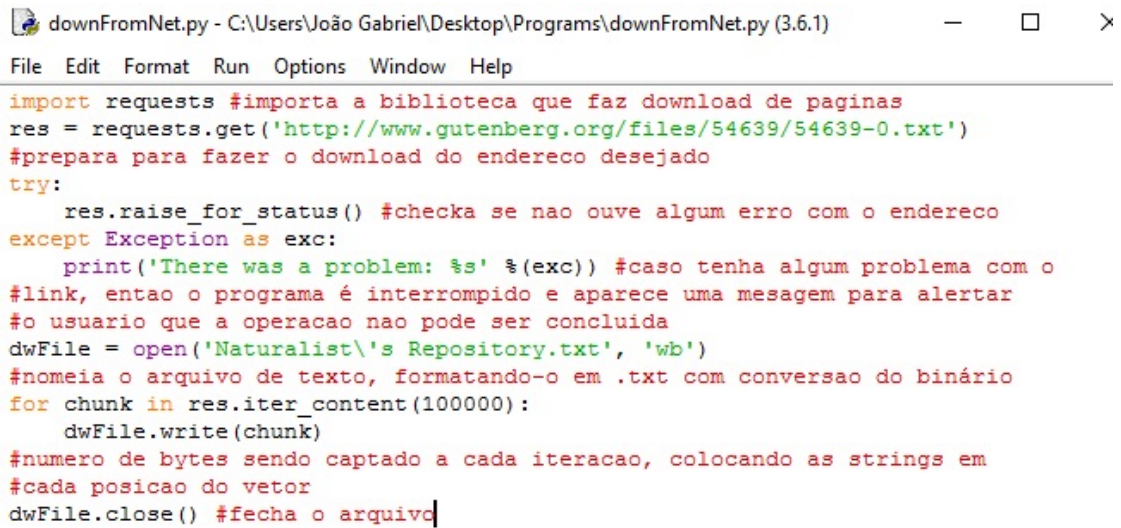
import webbrowser, sys, pyperclip #importando as bibliotecas necessarias
if len(sys.argv)>1: #se a var argv for >1 (na posicao 0 do vetor, esta mapIt)
    address = ' '.join(sys.argv[1:]) #pega o endereço da linha de comando
else:
    address = pyperclip.paste() #pega o endereço pelo clipboard - ctrl+c

webbrowser.open('https://www.google.com/maps/place/' + address) #abre o local
#no google maps
```

Figura 1: Código para encontrar um endereço a partir de um comando no executavel ou do ctrl+c

2 Requests

Agora, não só abrir páginas na web. Com a biblioteca *requests* podemos fazer o download de páginas da internet. Precisamos ter atenção na quantidade de bytes que iremos trabalhar para cada espaço no vetor, para termos certeza que toda informação poderá ser obtida.



```
import requests #importa a biblioteca que faz download de paginas
res = requests.get('http://www.gutenberg.org/files/54639/54639-0.txt')
#prepara para fazer o download do endereco desejado
try:
    res.raise_for_status() #checka se nao ouve algum erro com o endereco
except Exception as exc:
    print('There was a problem: %s' %(exc)) #caso tenha algum problema com o
#link, entao o programa é interrompido e aparece uma mensagem para alertar
#o usuario que a operacao nao pode ser concluida
dwFile = open('Naturalist\'s Repository.txt', 'wb')
#nomeia o arquivo de texto, formatando-o em .txt com conversao do binário
for chunk in res.iter_content(100000):
    dwFile.write(chunk)
#numero de bytes sendo captado a cada iteracao, colocando as strings em
#cada posicao do vetor
dwFile.close() #fecha o arquivo
```

Figura 2: Programa que copia o texto de um site e cria um arquivo txt no diretório com o texto

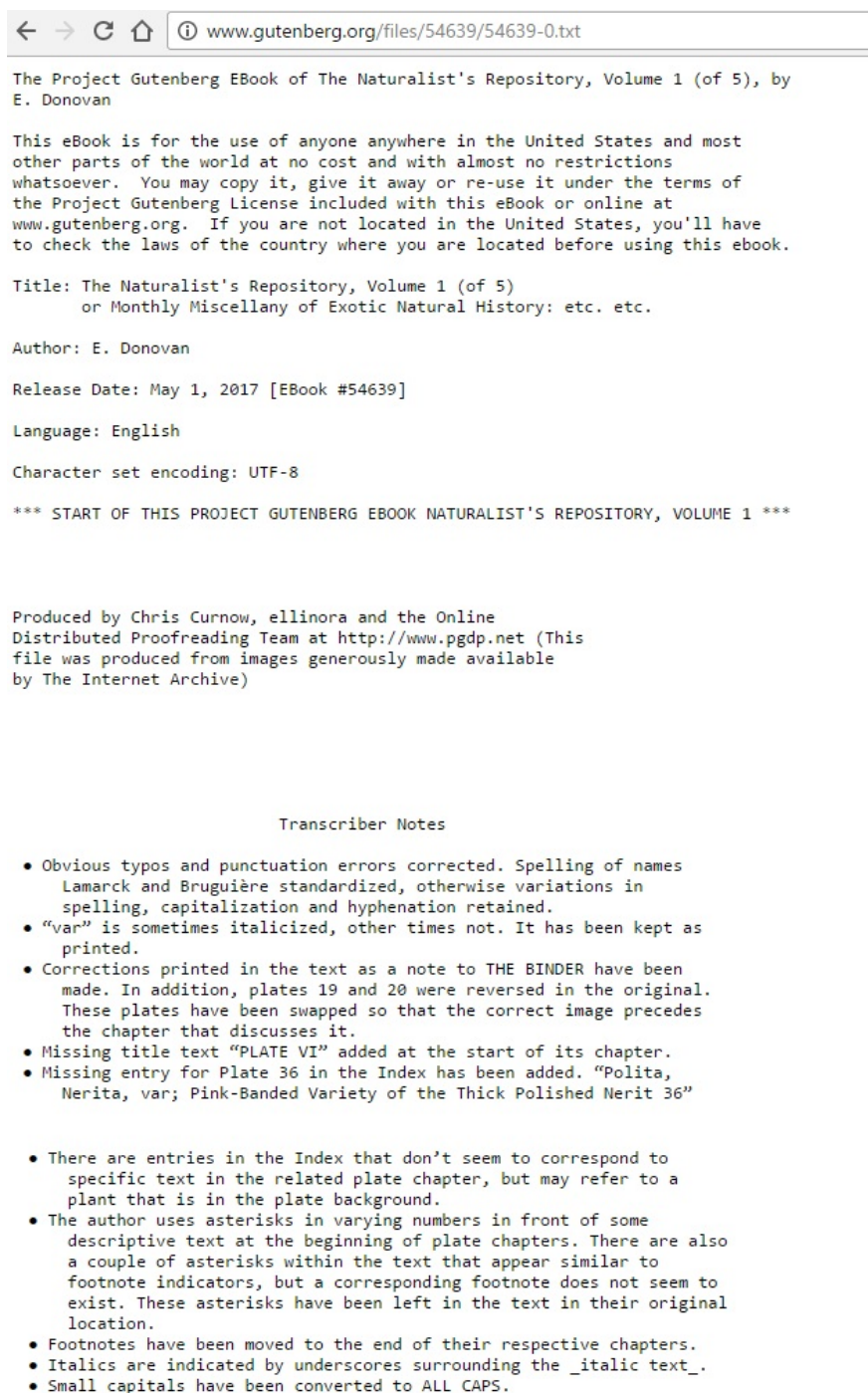


Figura 3: Nesta figura vemos o site do qual foi retirado as informações

The Project Gutenberg EBook of The Naturalist's Repository, Volume 1 (of 5), by E. Donovan

This eBook is for the use of anyone anywhere in the United States and most other parts of the world at no cost and with almost no restrictions whatsoever. You may copy it, give it away or re-use it under the terms of the Project Gutenberg License included with this eBook or online at www.gutenberg.org. If you are not located in the United States, you'll have to check the laws of the country where you are located before using this ebook.

Title: The Naturalist's Repository, Volume 1 (of 5)
or Monthly Miscellany of Exotic Natural History: etc. etc.

Author: E. Donovan

Release Date: May 1, 2017 [EBook #54639]

Language: English

Character set encoding: UTF-8

*** START OF THIS PROJECT GUTENBERG EBOOK NATURALIST'S REPOSITORY, VOLUME 1 ***

Produced by Chris Curnow, ellinora and the Online Distributed Proofreading Team at <http://www.pgdp.net> (This file was produced from images generously made available by The Internet Archive)

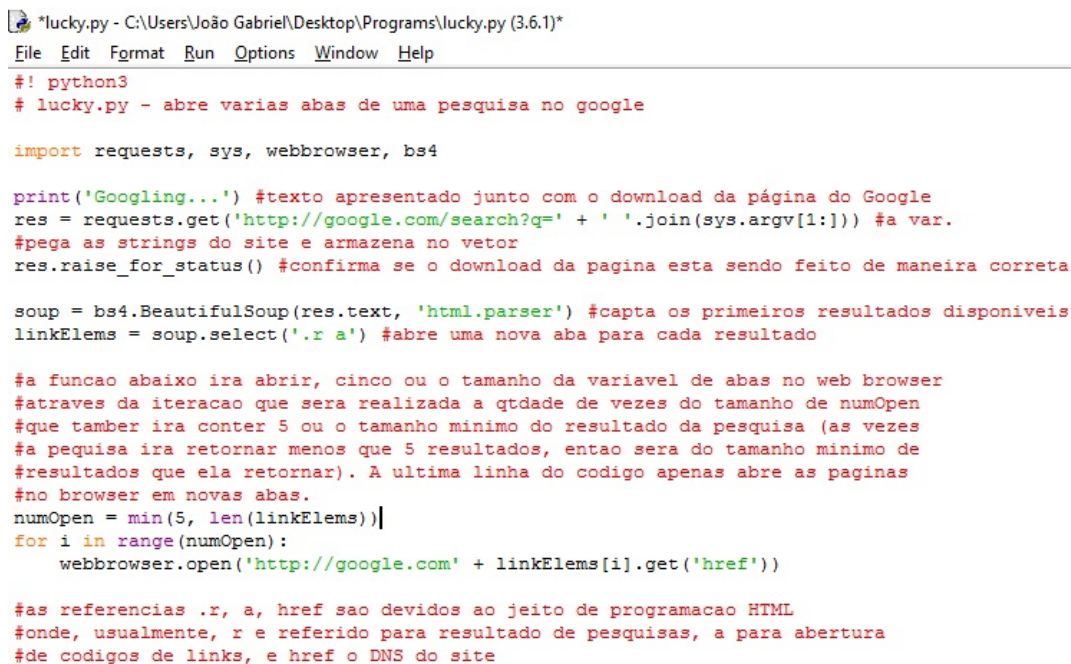
Transcriber Notes

- Obvious typos and punctuation errors corrected. Spelling of names Lamarck and Bruguière standardized, otherwise variations in spelling, capitalization and hyphenation retained.
- "var" is sometimes italicized, other times not. It has been kept as printed.

Figura 4: Aqui como ficou o arquivo retirado em formato txt

3 BeautifulSoup

Procurar informações precisas no meio de códigos HTMLs é uma função bastante complicada, uma vez que existem elementos implícitos dentro do código fonte. Por isso, fora criada a biblioteca *bs4* para poder facilitar e direcionar uma busca baseada na classe. É uma função muito bem utilizada com as outras que vimos anteriormente.



```
*lucky.py - C:\Users\João Gabriel\Desktop\Programs\lucky.py (3.6.1)*
File Edit Format Run Options Window Help

#! python3
# lucky.py - abre varias abas de uma pesquisa no google

import requests, sys, webbrowser, bs4

print('Googling...') #texto apresentado junto com o download da página do Google
res = requests.get('http://google.com/search?q=' + ' '.join(sys.argv[1:])) #a var.
#pega as strings do site e armazena no vetor
res.raise_for_status() #confirma se o download da pagina esta sendo feito de maneira correta

soup = bs4.BeautifulSoup(res.text, 'html.parser') #capta os primeiros resultados disponiveis
linkElems = soup.select('.r a') #abre uma nova aba para cada resultado

#a funcao abaixo ira abrir, cinco ou o tamanho da variavel de abas no web browser
#atraves da iteracao que sera realizada a qtdade de vezes do tamanho de numOpen
#que tambem ira conter 5 ou o tamanho minimo do resultado da pesquisa (as vezes
#a pesquisa ira retornar menos que 5 resultados, entao sera do tamanho minimo de
#resultados que ela retornar). A ultima linha do codigo apenas abre as paginas
#no browser em novas abas.
numOpen = min(5, len(linkElems))
for i in range(numOpen):
    webbrowser.open('http://google.com' + linkElems[i].get('href'))

#as referencias .r, a, href sao devidos ao jeito de programacao HTML
#onde, usualmente, r e referido para resultado de pesquisas, a para abertura
#de codigos de links, e href o DNS do site
```

Figura 5: A partir de uma pesquisa qualquer, este script irá abrir os cinco primeiros resultados da busca, ou caso obtenha um resultado menor que 5, abrirá todos



```
downloadXkcd.py - C:\Users\João Gabriel\Desktop\Programs\downloadXkcd.py (3.6.1)
File Edit Format Run Options Window Help

#!/python3
# downloadXkcd.py - downloading all the comics from an website

import requests, os, bs4

url = 'http://xkcd.com' #url inicial
os.makedirs('xkcd', exist_ok=True) #armazena quadrinhos no ./xkcd e deixa claro
#que exista uma pasta pronta
while not url.endswith('#'):#enquanto o url nao termina com a str '#'
    print('Downloading page %s...' %url)#realiza o download
    res = requests.get(url)
    res.raise_for_status()#garante que nao há erro no url
    soup = bs4.BeautifulSoup(res.text, "html.parser")#a funcao fica analisando e
#buscando partes do codigo html que correspondam a variavel

    comicElem = soup.select('#comic img')#procura o url da imagem
    if comicElem == []:
        print('Could not find comic image.')#caso nao ache, printa isso
    else:
        comicUrl = comicElem[0].get('src')
        print('Downloading image %s...' % (comicUrl))
        res = requests.get(comicUrl)
        res.raise_for_status()

#caso ele ache, o programa vai armazenando dentro do vetor as imagens dentro
#do diretorio selecionado, sempre ao final da iteracao checando se o url
#esta correto para evitar erros com a func raise_for_status()

        imageFile = open(os.path.join('xkcd', os.path.basename(comicUrl)), 'wb')
        for chunk in res.iter_content(100000):
            imageFile.write(chunk)
        imageFile.close()

#faz o processo de conversao de bits (existe 100000 bytes possiveis para serem
#preenchidos a cada iteracao e convertidos e escritos em pedacos

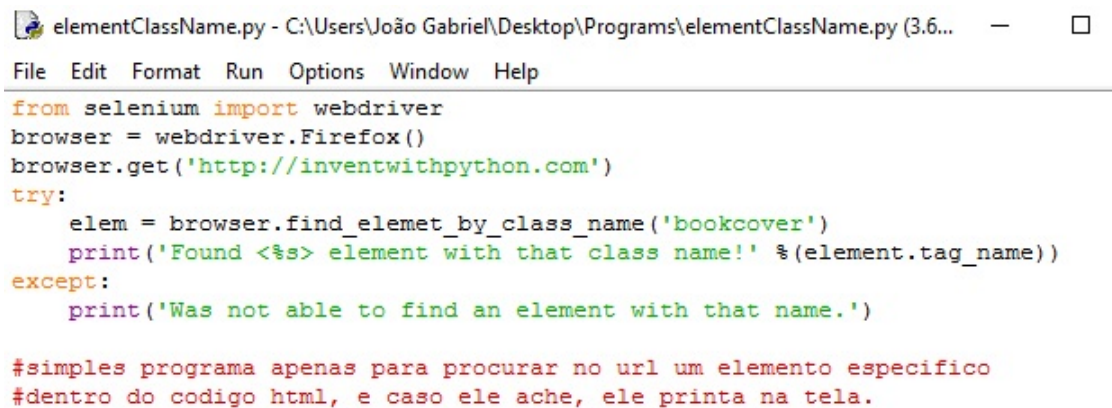
        prevLink = soup.select('a[rel="prev"]')[0]
        url = 'http://xkcd.com' + prevLink.get('href')
#funcao para pegar o link anterior do quadrinho atual, para poder comecar
#toda a iteracao novamente.

print('Done.')
```

Figura 6: Programa para baixar todos os quadrinhos de um site usando a função para "voltar ao quadrinho anterior" disponibilizada no site

4 Selenium

Além de abrir uma página na web, procurar informações específicas, obtê-las, desejamos também trabalhar com browsers específicos, mandar e receber informações, manipular dados. Para isso, a biblioteca *selenium* aparece para terminar de unificar um arsenal para trabalhar com páginas na web através do python.

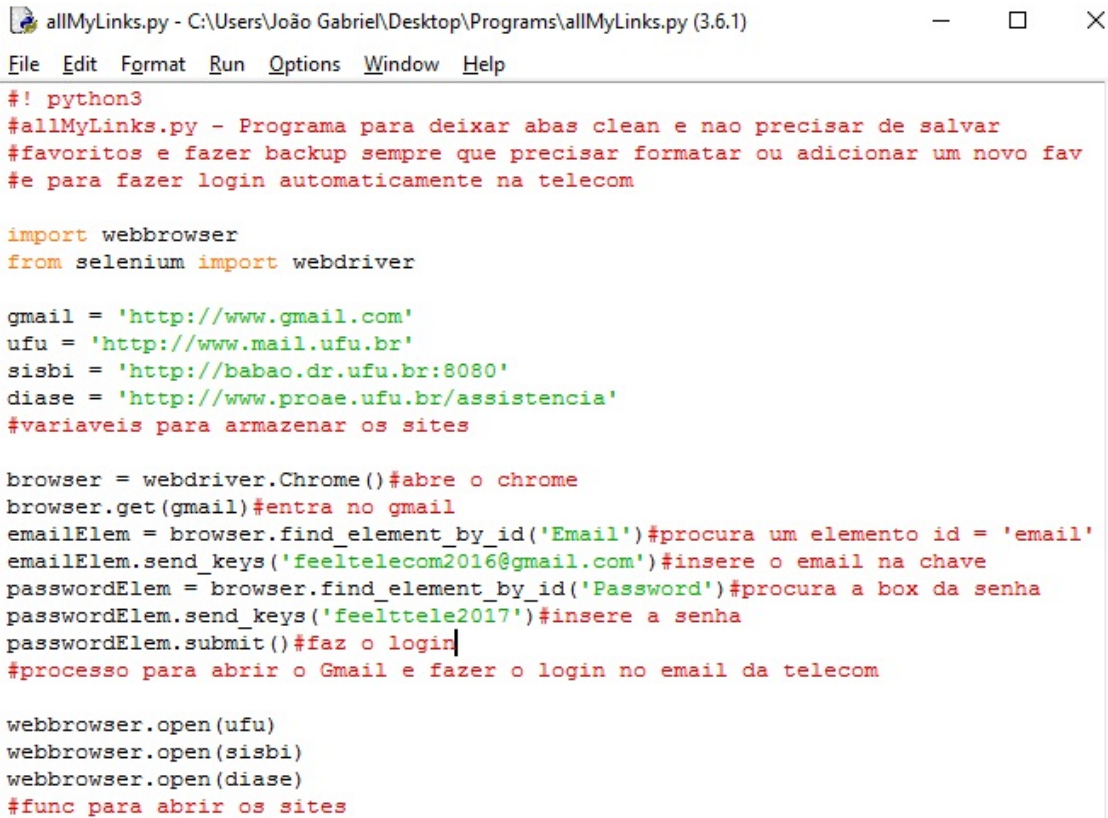


The image shows a screenshot of a Python script named `elementClassName.py` running in a text editor. The script uses Selenium to open a browser, navigate to `http://inventwithpython.com`, and find an element with the class name `bookcover`. If the element is found, it prints the tag name; otherwise, it prints a message indicating the element was not found. The script is commented in Portuguese, stating it is a simple program to search for a specific element in an HTML code and print it if found.

```
elementClassName.py - C:\Users\João Gabriel\Desktop\Programs\elementClassName.py (3.6...
File Edit Format Run Options Window Help
from selenium import webdriver
browser = webdriver.Firefox()
browser.get('http://inventwithpython.com')
try:
    elem = browser.find_element_by_class_name('bookcover')
    print('Found <{s}> element with that class name!' %(element.tag_name))
except:
    print('Was not able to find an element with that name.')

#simples programa apenas para procurar no url um elemento especifico
#dentro do codigo html, e caso ele ache, ele printa na tela.
```

Figura 7: Função básica apenas para abrir um diretório, buscar informações, obtê-las e explicitá-las



```
#!/ python3
#allMyLinks.py - Programa para deixar abas clean e nao precisar de salvar
#favoritos e fazer backup sempre que precisar formatar ou adicionar um novo fav
#e para fazer login automaticamente na telecom

import webbrowser
from selenium import webdriver

gmail = 'http://www.gmail.com'
ufu = 'http://www.mail.ufu.br'
sisbi = 'http://babao.dr.ufu.br:8080'
diase = 'http://www.proae.ufu.br/assistencia'
#variaveis para armazenar os sites

browser = webdriver.Chrome()#abre o chrome
browser.get(gmail)#entra no gmail
emailElem = browser.find_element_by_id('Email')#procura um elemento id = 'email'
emailElem.send_keys('feelttelecom2016@gmail.com')#insere o email na chave
passwordElem = browser.find_element_by_id('Password')#procura a box da senha
passwordElem.send_keys('feelttele2017')#insere a senha
passwordElem.submit()#faz o login
#processo para abrir o Gmail e fazer o login no email da telecom

webbrowser.open(ufu)
webbrowser.open(sisbi)
webbrowser.open(diase)
#func para abrir os sites
```

Figura 8: Neste programa, criei um código que irá abrir 4 sites em novas abas no meu navegador, sendo que no Gmail, ele irá automaticamente fazer o login no e-mail da Telecom