ПРОГРАММА ДЛЯ ЭВМ

##### Утилита для скачивания и анализа сообщений с форума velomania.ru

Фрагменты исходного текста программы

Листов \_4\_

Правообладатель-автор:Трофимов Владислав Александрович

© Трофимов Владислав Александрович, 2016

г. Санкт-Петербург

2016 г.

import re

import csv

import requests

from bs4 import BeautifulSoup

\_\_author\_\_ = 'Stranger'

thread\_url = 'http://forum.velomania.ru/showthread.php?t=74626'

image\_label = '\*\*IMAGE\*\*'

video\_label = '\*\*VIDEO\*\*'

output\_file = 'thread1.tsv'

def extract\_data\_from\_url(url):

html = requests.get(url).content

soup = BeautifulSoup(html, 'html.parser')

posts = soup.findAll('div', {'class': 'postdetails'})

extracted = []

for post in posts:

author = get\_author(post)

content = get\_content(post)

contains\_video = video\_label in content

contains\_images = image\_label in content

extracted.append((url, author, content, contains\_images, contains\_video))

return extracted

def get\_content(post):

raw\_content = post.find('div', {'class': 'content'}).prettify()

pre\_content = remove\_tags(raw\_content)

pre\_content = linearize\_and\_remove\_trash(pre\_content)

pre\_content = replace\_images\_and\_video\_with\_label(pre\_content)

pre\_content = linearize(pre\_content)

# pre\_content = highlight\_links(pre\_content)

content = remove\_nonprintable\_chars(pre\_content)

return content

def get\_author(post):

author\_block = post.find('a', {'class': 'username'})

author = author\_block.find('strong').contents[0]

if not isinstance(author, str):

author = author.contents[0]

return author

def remove\_nonprintable\_chars(pre\_content):

content = re.sub('[\u200e\ufffd\u0301\u0394\u221e]', ' ', pre\_content)

return content

def highlight\_links(pre\_content):

content = re.sub('http://', '\nhttp://', pre\_content)

return content

def linearize(pre\_content):

pre\_content = re.sub('\s+', ' ', pre\_content)

return pre\_content

def replace\_images\_and\_video\_with\_label(pre\_content):

pre\_content = re.sub('<img alt=', ' ' + image\_label + ' ', pre\_content)

pre\_content = re.sub('http://video.yandex.ru.\* ', ' ' + video\_label + ' ', pre\_content)

pre\_content = re.sub('http://.\*you.\* ', ' ' + video\_label + ' ', pre\_content)

pre\_content = re.sub('http://.\*\.jpg.\* ', ' ' + image\_label + ' ', pre\_content)

pre\_content = re.sub('http://.\*\.png.\* ', ' ' + image\_label + ' ', pre\_content)

pre\_content = re.sub('http://.\*\.JPG.\* ', ' ' + image\_label + ' ', pre\_content)

pre\_content = re.sub('http://`.\*\.PNG.\* ', ' ' + image\_label + ' ', pre\_content)

pre\_content = re.sub('http://photofile.\* ', ' ', pre\_content)

pre\_content = re.sub('\w+\.png', ' ' + image\_label + ' ', pre\_content)

return pre\_content

def linearize\_and\_remove\_trash(pre\_content):

pre\_content = re.sub('\n', ' ', pre\_content)

pre\_content = re.sub(('\n'

'|<!-->'

'|<embed width='

'|&gt;|&lt;'), ' ', pre\_content)

return pre\_content

def remove\_tags(raw\_content):

pre\_content = re.sub(('<br/>|<br>'

'|<div.\*>|</div>'

'|<i>|</i>'

'|<b>|</b>'

'|<ul>|</ul>'

'|<li.\*>|</li>'

'|<ol.\*>|</ol>'

'|<object.\*>|</object>'

'|<legend>|</legend>'

'|<strike>|</strike>'

'|<u>|</u>'

'|<span.\*>|</span>'

'|<strong>|</strong>'

'|<param.\*>|</param>'

'|<font.\*>|</font>'

'|<fieldset.\*>|</fieldset>'

'|</blockquote>|<blockquote.\*>'

'|<img alt=\"Цитата\".\*>'

'|<a href=\"showthread.php.\*>'

'|<a href=\"'

'|".\*>'

'|http://.{29}\.\.\..{14}'

'|<img .\*src=\"images.\*>'

'|&lt;.\*&gt;'

'|<!--.\*-->'

'|</a>'

'|\[.\*\]'), '', raw\_content)

return pre\_content

def get\_all\_pages\_url():

html = requests.get(thread\_url).content

soup = BeautifulSoup(html, 'html.parser')

last\_page\_url = soup.find('span', {'class': 'first\_last'}).find('a')['href']

matches = re.match('showthread\.php.\*&page=(\d+).\*', last\_page\_url)

if matches:

last\_page\_number = int(matches.group(1))

else:

raise Exception('can\'t parse last page number')

pages = list()

for current\_page\_number in range(1, last\_page\_number + 1):

current\_page\_url = thread\_url + '&page=' + str(current\_page\_number)

pages.append(current\_page\_url)

return pages

def print\_to\_csv(data):

with open(output\_file, 'w', newline='') as csvfile:

csvwriter = csv.writer(csvfile, delimiter='\t')

for row in data:

csvwriter.writerow(list(row))

def main():

data = []

pages = get\_all\_pages\_url()

for url in pages:

print('Parsing page ' + url + ' of ' + str(len(pages)), end='\r')

data += extract\_data\_from\_url(url)

print\_to\_csv(data)

if \_\_name\_\_ == '\_\_main\_\_':

main()

|  |
| --- |
| **Всего пронумеровано и прошнуровано**  \_4\_ **листов фрагментов исходного текста программы**  **Правообладатель**  **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ /\_\_\_\_**Трофимов В.А.\_  подпись |