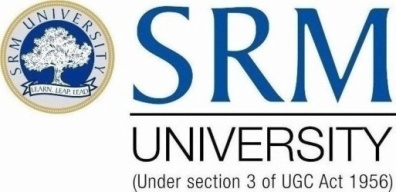
**WEB SCRAPPER**

**15IT322E- PYTHON PROGRAMMING PROJECT REPORT**

***Submitted by***

**Sukriti Tiwari [RA1511008010407]**

***for the assessment of Semester V Minor Project***



**SRMUNIVERSITY**

KATTANKULATHUR

**Nov 2017**

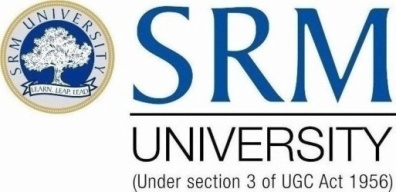
**WEB SCRAPPER**

**15IT322E- PYTHON PROGRAMMING PROJECT REPORT**

***Submitted by***

**Kshama Sharma [RA1511008010253]**

***for the assessment of Semester V Minor Project***



**SRMUNIVERSITY**

KATTANKULATHUR

**Nov 2017**

# ABSTRACT

**Web scraping** (**web harvesting** or **web data extraction**) is data scraping used for extracting data from websites. Web scraping software may access the World Wide Web directly using the Hypertext Transfer Protocol, or through a web browser. While web scraping can be done manually by a software user, the term typically refers to automated processes implemented using a bot or web crawler. It is a form of copying, in which specific data is gathered and copied from the web, typically into a central local database or spread sheet, for later retrieval or analysis.

Web scraping a web page involves fetching it and extracting from it. Fetching is the downloading of a page (which a browser does when you view the page). Therefore, web crawling is a main component of web scraping, to fetch pages for later processing. Once fetched, then extraction can take place. The content of a page may be parsed, searched, reformatted, its data copied into a spreads sheet, and so on. Web scrapers typically take something out of a page, to make use of it for another purpose somewhere else. An example would be to find and copy names and phone numbers, or companies and their URLs, to a list (contact scraping).

Web scraping is used for contact scraping, and as a component of applications used for web indexing, web mining and data mining, online price change monitoring and price comparison, product review scraping (to watch the competition), gathering real estate listings, weather data monitoring, website change detection, research, tracking online presence and reputation, web mash up and, web data integration.

# DECLARATION

I Sukriti Tiwari [RA1511008010407]studying in III year B.Tech Information Technology program at, SRM University, Kattankulathur, Chennai, hereby declare that this project is an original work of mine and I have not verbatim copied / duplicated any material from sources like internet or from print media, excepting some vital company information / statistics and data that is provided by the company itself.

Signature of the Student

Date:

Place:

# DECLARATION

I Kshama Sharma [RA1511008010253]studying in III year B.Tech Information Technology program at, SRM University, Kattankulathur, Chennai, hereby declare that this project is an original work of mine and I have not verbatim copied / duplicated any material from sources like internet or from print media, excepting some vital company information / statistics and data that is provided by the company itself.

Signature of the Student

Date:

Place:

## CONTENTS

1. Abstract
2. Aim
3. Requirements
4. Module Description
5. Screenshots
6. Conclusion
7. References

AIM

Web scraping tools are software (i.e., bots) programmed to sift through databases and extract information. A variety of bots types are used, many being fully customizable to:

* Recognize unique HTML site structures
* Extract and transform content
* Store scraped data
* Extract data from APIs

Since all scraping bots have the same purpose—to access site data—it can be difficult to distinguish between legitimate and malicious bots.

Aim of this project is to get data easily through web scraping. We can get the email IDs of all teachers in the department we choose, in engineering through web scraping. The crawler will go and fetch all the code of that side and then finds the tag with a particular class in order to display the required output.

# REQUIREMENTS

**Software Requirements:**

* Pycharm Professional/Community Version

**Hardware Requirements:**

**Windows:**

* Microsoft Windows 10/8/7/Vista/2003/XP (incl.64-bit)
* 1 GB RAM minimum
* 2 GB RAM recommended
* 1024x768 minimum screen resolution
* [**Python**](http://www.qbssoftware.com/products/PyCharm/system-requirements/_prodjbpycharm) 2.4 or higher, Jython, PyPy or IronPython

**Mac:**

* Mac OS X 10.8 or higher
* 1 GB RAM minimum
* 2 GB RAM recommended
* [**Python**](http://www.qbssoftware.com/products/PyCharm/system-requirements/_prodjbpycharm) 2.4 or higher, Jython, PyPy or IronPython

**Linux:**

* 512 MB RAM minimum, 1 GB RAM recommended
* 1024x768 minimum screen resolution
* [**Python**](http://www.qbssoftware.com/products/PyCharm/system-requirements/_prodjbpycharm) 2.4 or higher, Jython, PyPy or IronPython

# MODULE DESCRIPTION

import requests,re

from bs4 import BeautifulSoup

print("Fetch mail id of all faculty of:")

print("1.MATHEMATICS")

print("2.MECHANICAL")

print("3.SOFTWARE")

print("Enter your choice:")

k=input()

emails=[]

def GetEmail(fac\_info):

email=[]

for i in fac\_info.keys():

html = requests.get(fac\_info[i])

soup = BeautifulSoup(html.content, "html.parser")

mail = soup.find('tbody')

if (mail != None and mail.find('a') != None):

em = re.findall('.+@', mail.find('a').text)

if (len(em) > 0):

email.append(em[0] + 'ktr.srmuniv.ac.in')

return email

if k=='1':

#mathematics

print("hello")

html=requests.get("http://www.srmuniv.ac.in/engineering/school-of-basicsciences/department-of-mathematics/faculty")

soup=BeautifulSoup(html.content,"html.parser")

linkname=soup.findAll('a',{"target":"\_self"})

fac\_info={}

for i in linkname:

fac\_info[i.text]=i['href']

print(i['href'])

emails=GetEmail(fac\_info)

print(emails)

#mechanical

elif k=='2':

html=requests.get("http://www.srmuniv.ac.in/mech-engg/faculty-list")

soup=BeautifulSoup(html.content,"html.parser")

linkname=soup.findAll('a',{"target":"\_self"})

fac\_info={}

for i in linkname:

fac\_info[i.text]=i['href']

print(i['href'])

emails=GetEmail(fac\_info)

print(emails)

#software

if k=='3':

html=requests.get("http://www.srmuniv.ac.in/engineering/department-of-software-engineering/faculty")

soup=BeautifulSoup(html.content,"html.parser")

linkname=soup.findAll('a',{"class":"contentlink"})

fac\_info={}

for i in linkname:

fac\_info[i.text]='http://www.srmuniv.ac.in'+i['href']

print('http://www.srmuniv.ac.in'+i['href'])

emails=GetEmail(fac\_info)

print(emails)

print("Do you want to send mail to all these faculty:")

print("Y/N")

j=input()

if(j=='Y'):

print("ENTER YOUR MESSAGE TO BE SENT ON MAIL ID:")

j=input()

import smtplib

server = smtplib.SMTP('smtp.gmail.com:587')

server.starttls()

server.login("sukritit24@gmail.com", "bleh@456bleh")

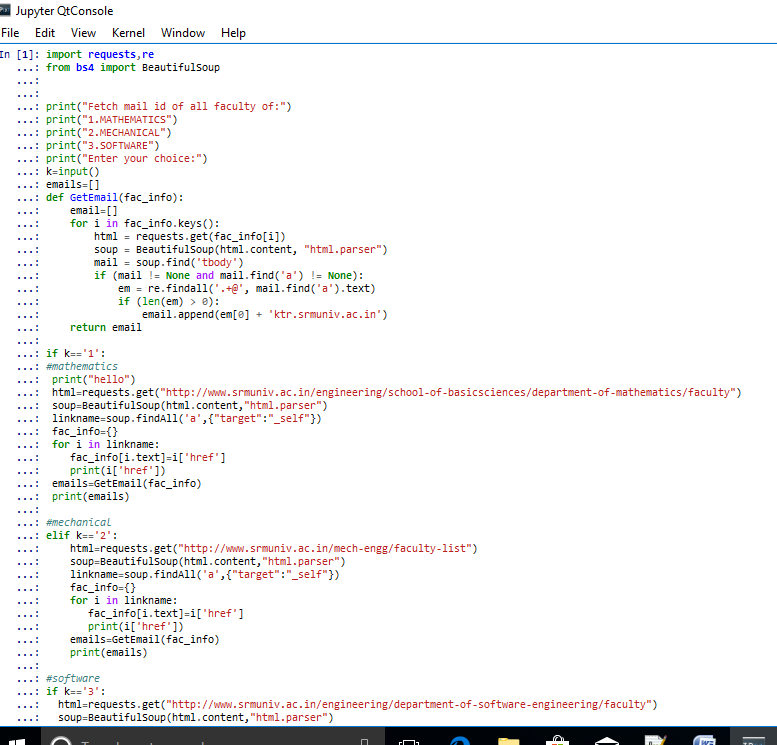
msg = j

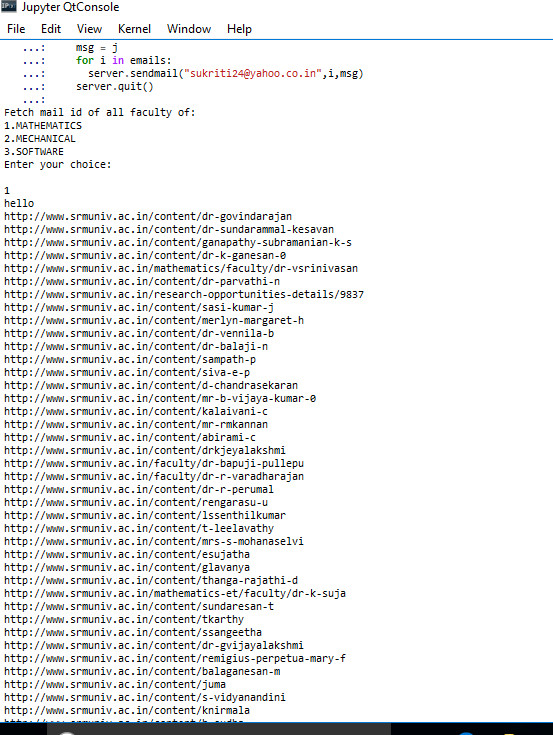
for i in emails:

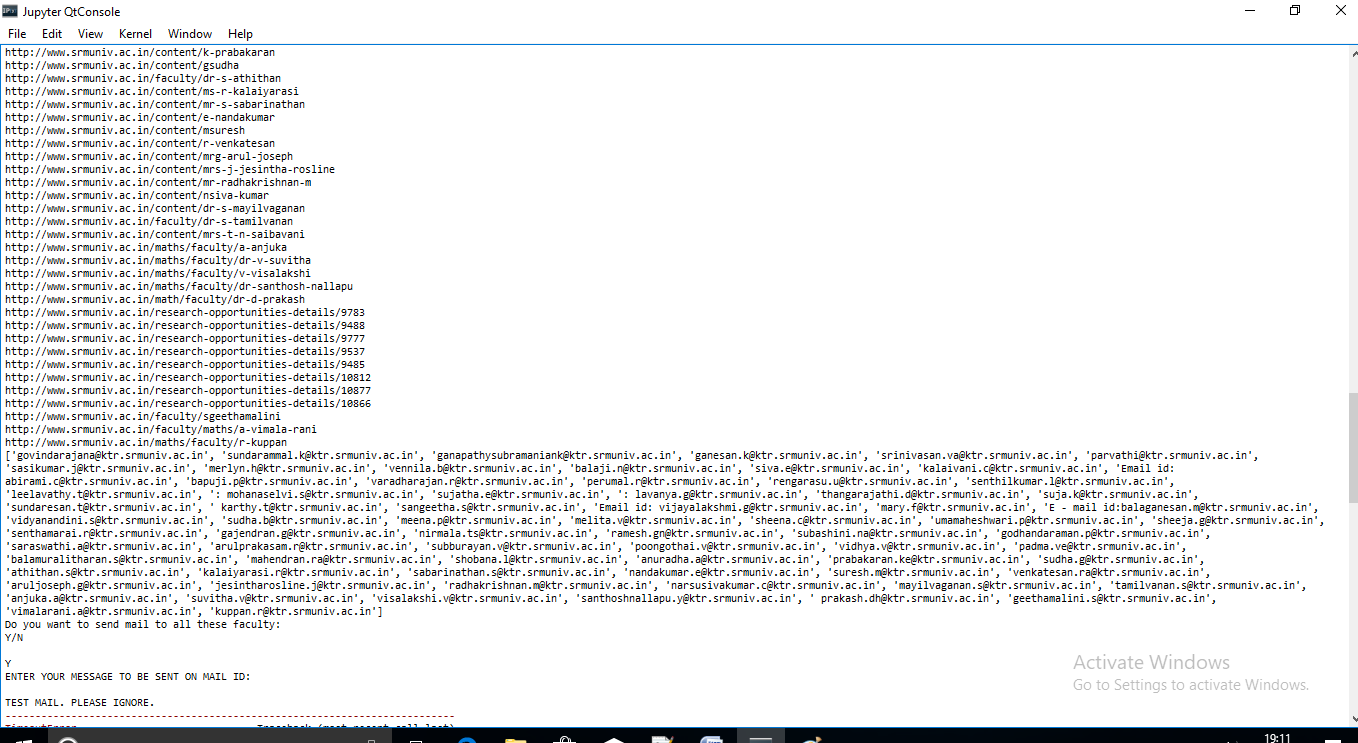
server.sendmail("sukritit24@gmail.com",i,msg)

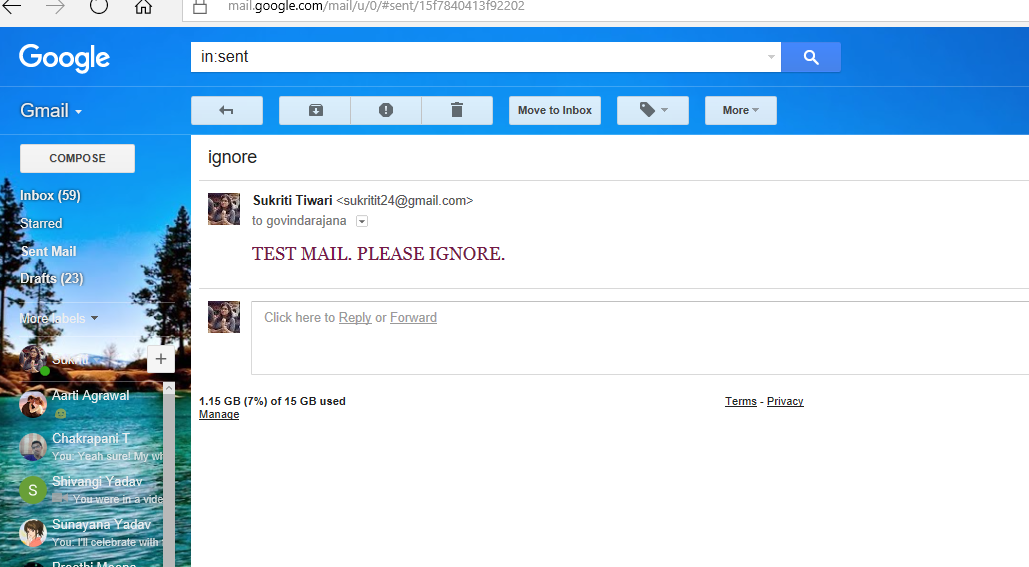
server.quit()

# SCREENSHOTS









# CONCLUSION

It is important to recognize that in certain circumstances web scraping can be illegal. If the terms and conditions of the web site we are scraping specifically prohibit downloading and copying its content, then we could be in trouble for scraping it.

In practice, however, web scraping is a tolerated practice, provided reasonable care is taken not to disrupt the “regular” use of a web site, as we have seen above.

In a sense, web scraping is no different than using a web browser to visit a web page, in that it amounts to using computer software to access data that is publicly available on the web.

In general, if data is publicly available (the content that is being scraped is not behind a password-protected authentication system), then it is OK to scrape it, provided we don’t break the web site doing so. What is potentially problematic is if the scraped data will be shared further. For example, downloading content off one website and posting it on another website (as our own), unless explicitly permitted, would constitute copyright violation and be illegal.

However, most copyright legislations recognize cases in which reusing some, possibly copyrighted, information in an aggregate or derivative format is considered “fair use”. In general, unless the intent is to pass off data as our own, copy it word for word or trying to make money out of it, reusing publicly available content scraped off the internet is OK.

## REFERENCES

* 1. Deily, Ned (3 October 2017). "Python 3.6.3 is now available". Python Insider. The Python Core Developers. Retrieved 4 October 2017.
  2. Peterson, Benjamin (16 September 2017). "Python 2.7.14 released". Python Insider. The Python Core Developers. Retrieved 17 September 2017.