**Name: Nikhil H. Sahu**

Market Tract

Aim: **To create an application which scrapes E-mail, Contact number and Address in CSV file given name and location in CSV input file.**

## ****What is the purpose of the project?****

* There is a need of data to the organization for marketing or other purposes. The extraction of data is needed from websites.
* This is called Web Scraping. It is the process of extracting useful information from internet.
* For extracting data, there will be some mechanism which needs to be designed.
* The designing is done by Developer or Data Scientist by using some programming language.
* The programming language Python is preferred for Web Scraping because of its user-friendliness and ease of writing.
* Here, a python program is developed by using Some Libraries like BeautifulSoup4, googlesearch, requests, re.

## ****What are the advantages of using python instead of doing manually?****

* **Python is adaptive, versatile and user-friendly and it focuses on readability.**
* Python allows you to take the best of different paradigms of programming.
* Python is open source and writing code in a matter of minutes.
* It has all the libraries you can imagine that means more support.
* It also performs automatic memory management.
* Interactive, Interpreted, Modular, Dynamic, Object-oriented, Portable, High Level, Extensible in C and C++.

# TOOLS USED:

## Python 3.7.4:

* Python 3.7. 4, documentation released on 08 July 2019.
* Python is a dynamic object-oriented programming language that can be used for many kinds of software development.
* Many Python programmers report substantial productivity gains and feel the language encourages the development of higher quality, more maintainable code.
* It offers strong support for integration with other languages and tools, comes with extensive standard libraries, and can be learned in a few days.
* Python is distributed under an OSI-approved open source license that makes it free to use, even for commercial products.
* Python runs on Windows, Linux/Unix, Mac OS X, OS/2, Amiga, Palm Handhelds, and Nokia mobile phones. Python has also been ported to the Java and .NET virtual machines.
* Python programming is widely used in Artificial Intelligence, Natural Language Generation, Neural Networks and other advanced fields of Computer Science.

## IDLE (Python 3.7 64-bit)

* IDLE is Integrated Development and Learning Environment or Integrated Development Environment.
* IDLE is intended to be a simple IDE and suitable for beginners, especially in an educational environment. To that end, it is cross-platform, and avoids feature clutter.
* It is packaged as an optional part of the Python packaging with many Linux distributions.
* Multi-window text editor with syntax highlighting, autocompletion, smart indent and other.
* Integrated debugger with stepping, persistent breakpoints, and call stack visibility.
* It is completely written in Python and the Tkinter GUI toolkit (wrapper functions for T It is completely written in Python and the [Tkinter](https://en.wikipedia.org/wiki/Tkinter) GUI toolkit (wrapper functions for [Tcl](https://en.wikipedia.org/wiki/Tcl" \o "Tcl)/[Tk](https://en.wikipedia.org/wiki/Tk_(framework)).cl/Tk).

# LIBRARIES USED:

## BeautifulSoup

* Beautiful Soup is a Python library for getting data out of HTML, XML, and other markup languages.
* It is a tool for web scraping that helps you clean up and parse the documents you have pulled down from the web.
* Beautiful Soup helps you pull particular content from a webpage, remove the HTML markup, and save the information.
* The first step is getting a copy of the HTML page(s) want to scrape.
* You can combine BeautifulSoup with urllib3 to work directly with pages on the web.
* To install: pip install beautifulsoup4
* An bs4 object: soup = BeautifulSoup(page.content, 'html.parser')

## Requests

* Requests allows you to send HTTP/1.1 requests extremely easily.
* Requests is an elegant and simple HTTP library for Python, built for human beings. You are currently looking at the documentation of the development release.
* A large number of open source projects you find today are GPL Licensed.
* Licensed under the Apache License, Version 2.0 (the “License”); you may not use this file except in compliance with the License.
* Making a request with Requests is very simple.
* To install: pip install requests
* A request : page = requests.get(url)

## Googlesearch

* Library for scraping google search results.
* Run a Google search and fetch the individual results (full HTML and text contents).
* By default the result URLs are fetched eagerly when the search request is made with 10 parallel requests.
* To install: pip install google-search
* How to search:

search(query,

tld = 'com',

lang = 'en',

num = 10,

start = 0,

stop = 1,

pause = 2.0)

## Pandas

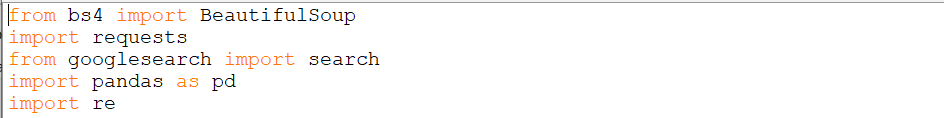
* The name is derived from the term "panel data", an econometrics term for data sets that include observations over multiple time periods for the same individuals.
* *pandas* aims to be the fundamental high-level building block for doing practical, real world data analysis in Python.
* pandas is a Python package providing fast, flexible, and expressive data structures designed to make working with “relational” or “labeled” data both easy and intuitive.
* To install: pip install pandas
* It aims to be the fundamental high-level building block for doing practical, realworld data analysis in Python.
* the most powerful and flexible open source data analysis / manipulation tool available in any language.
* Reading a file: df = pd.read\_csv('nameloc.csv')

## Re

* Re stands for Regular Expression.
* This module provides regular expression matching operations similar to those found in Perl.
* Regular expressions use the backslash character ('\') to indicate special forms or to allow special characters to be used without invoking their special meaning.
* Both patterns and strings to be searched can be Unicode strings (str) as well as 8-bit strings (bytes).
* To install: pip install re
* Re for mail: mail\_list = re.search("\w+@\w+\.{1}\w+", html\_text)

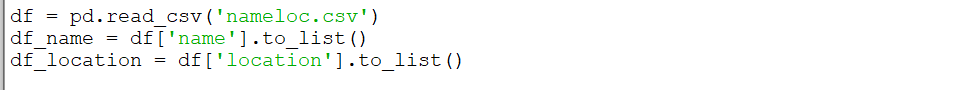
# Code Explanation:

## Importing Libraries:



Importing all the libraries like BeautifulSoup, requests, googlesearch, pandas and re in order to use their functionality.

## Reading CSV file



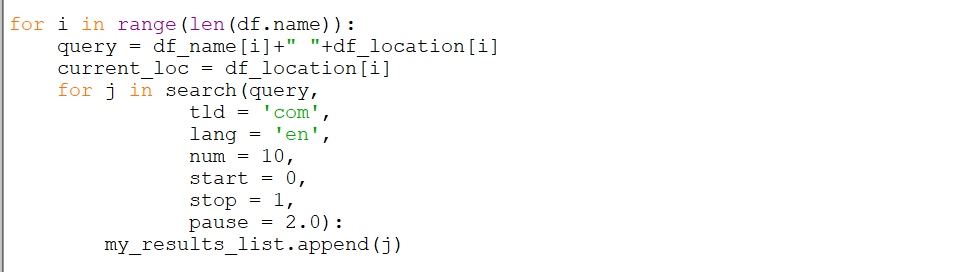
**Reading the CSV file using pandas library. Variable df is reading nameloc.csv file. Then creating lists named df\_name and df\_location using to\_list function.**

## ****Predefining Lists and variables****



Predefining global lists and variables so that they can be called and accessed globally.

## Google search



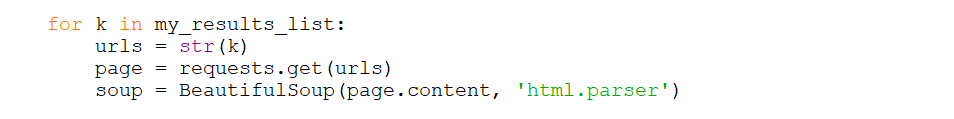
* **pause :**Lapse to wait between HTTP requests. Lapse too short may cause Google to block your IP. Keeping significant lapse will make your program slow but its safe and better option.
* **Return :**Generator (iterator) that yields found URLs. If the stop parameter is None the iterator will loop forever.

This will generate Here, the variable query contains two keywords df\_name and df\_keyword in string format. Google search is done by search function.

* **query :**query string that we want to search for.
* **tld :**tld stands for top level domain which means we want to search our result on google.com or google.in or some other domain.
* **lang :**lang stands for language.
* **num :**Number of results we want.
* **start :**First result to retrieve.
* **stop :**Last result to retrieve. Use None to keep searching forever.

This generates one url per query.

## Parsing



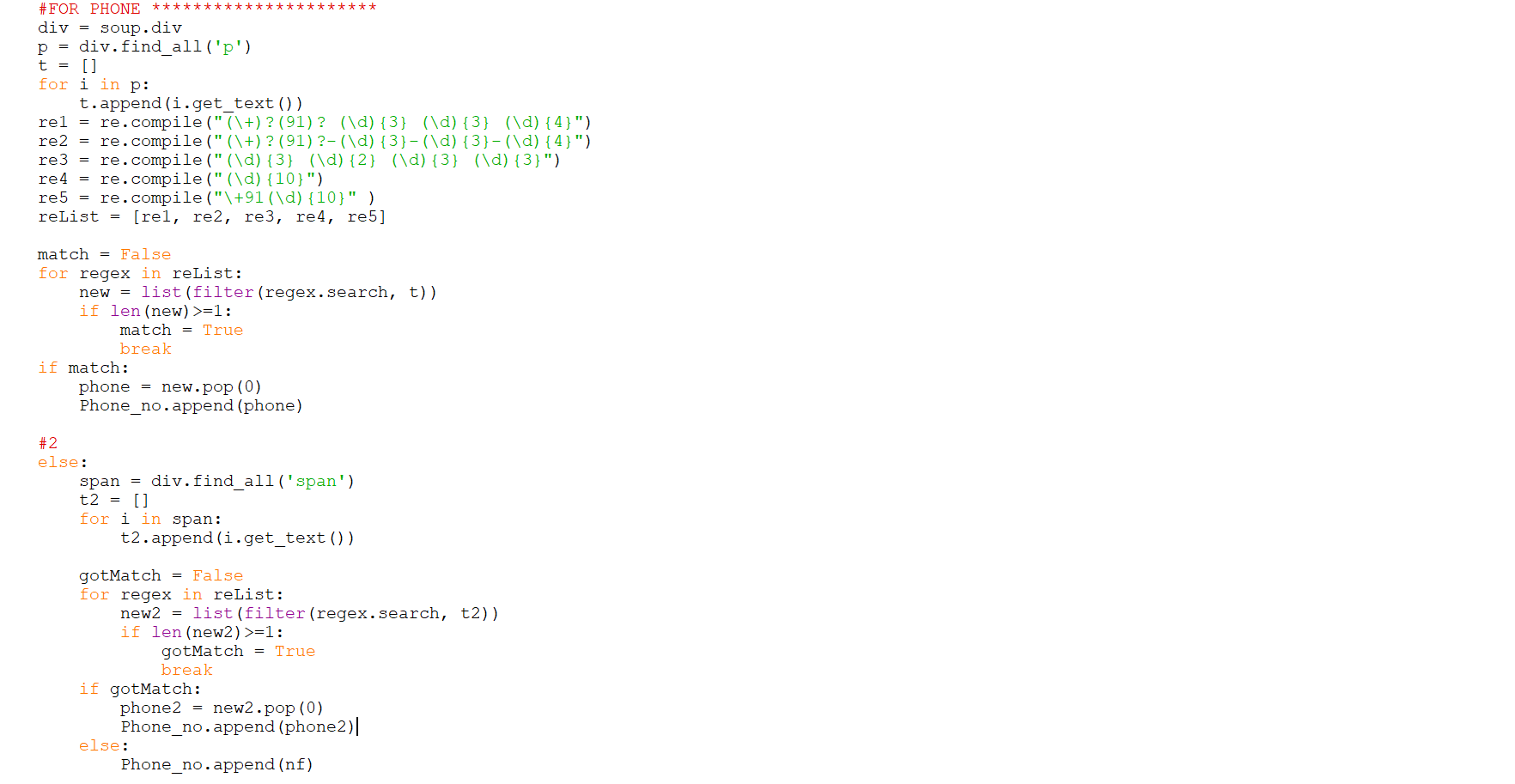
Variable my\_results\_list contains url fetched by google search. Each url is parse into HTML document. Request is sent by requests.get(urls). And response is saved in variable page. Then parsed using BeautifulSoup and html parser.

## Email Scraping



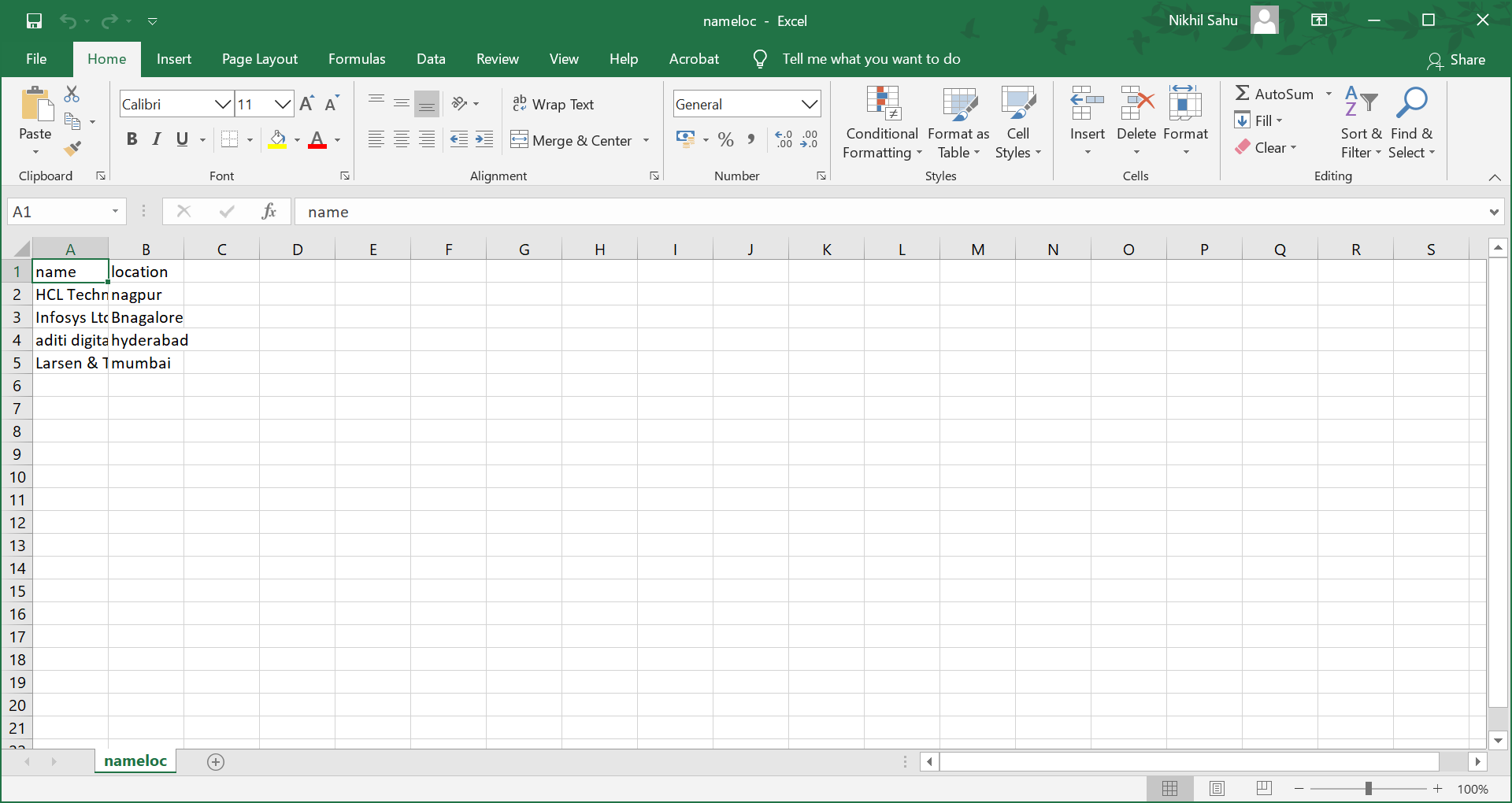
* + All <a> tag is iterated by for loop.
  + All href attribute in <a> is stored in my\_list list.
  + Variable emailregex stores regular expression for matching email pattern.
  + My\_list’s all iterables are converted to string using map function.
  + List newlist matches emailregex pattern with my\_list elements.
  + If pattern is matched then pop newlist and append to Email\_list.

## Phone number scraping

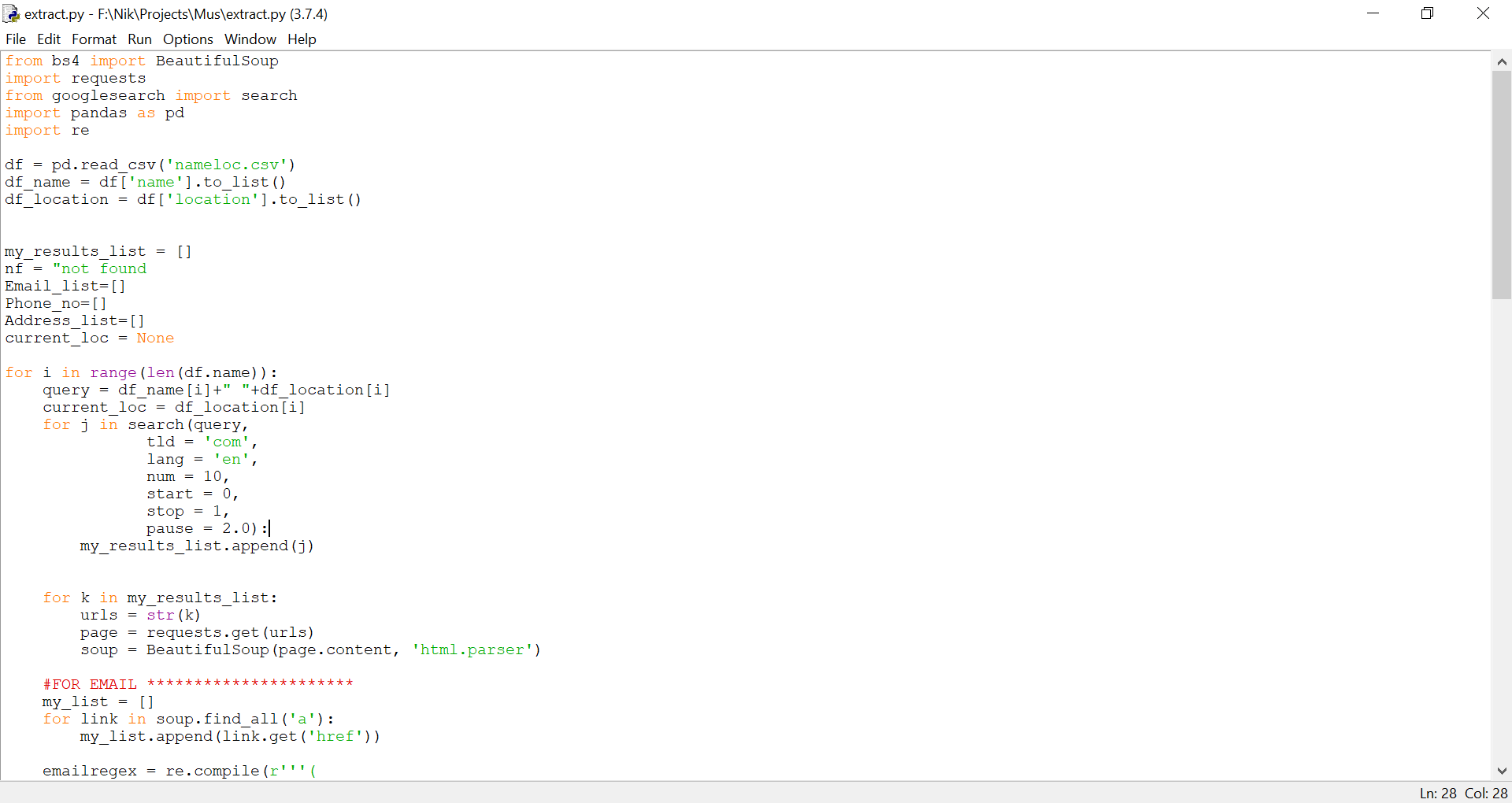


* All <p> tags are found out and stored in var p.
* All texts under <p> tag are stored in list t().
* Five regular expressions are created and stored in list relist().
* All regular expressions are matched with list t().
* If found then append in Phone\_no().
* Else find all <span> tag under <div> tag.
* Again match regex.
* If found then append in Phone\_no().

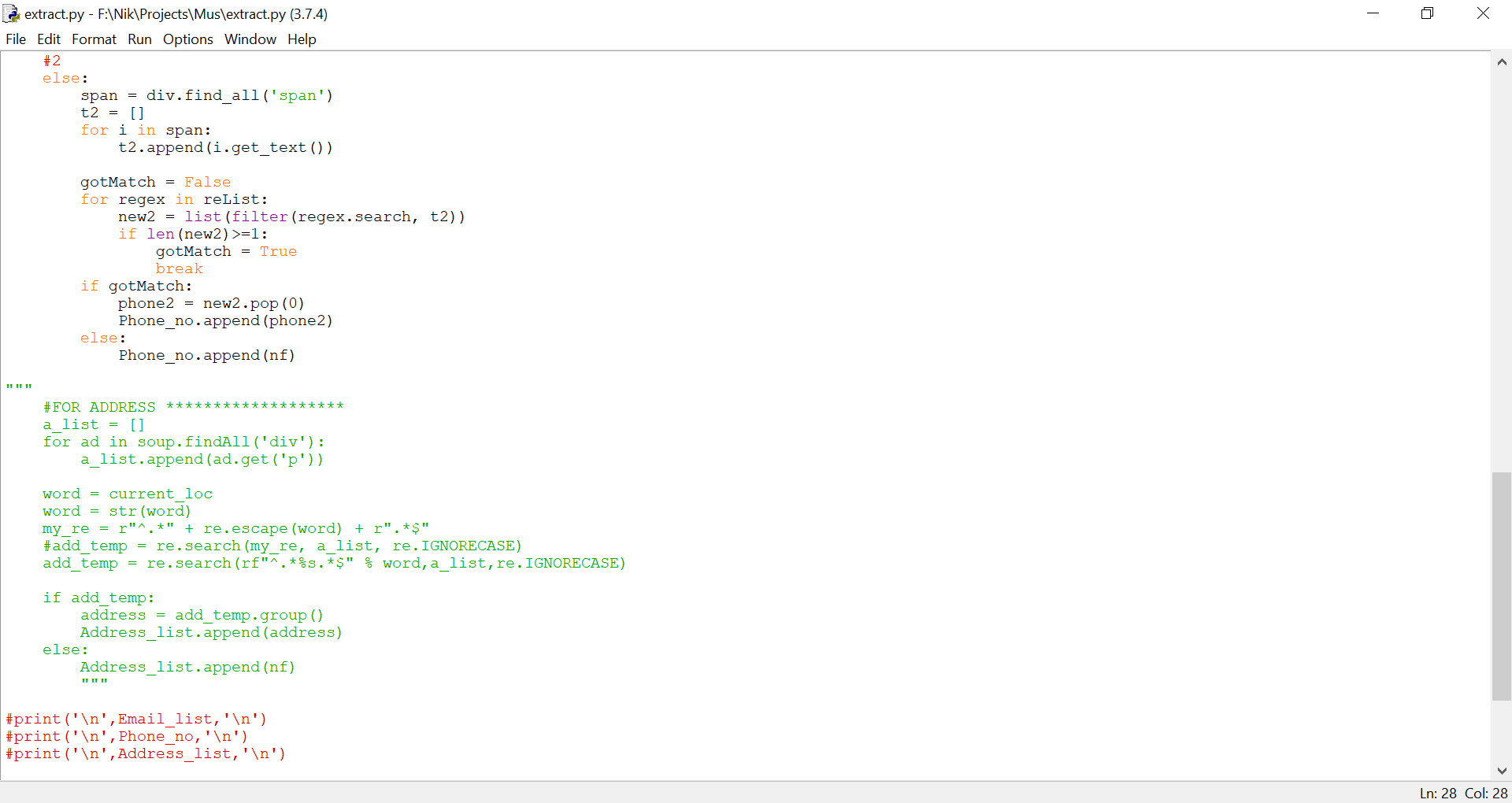
# Input:



# CODE:







# CODE (in text):

from bs4 import BeautifulSoup

import requests

from googlesearch import search

import pandas as pd

import re

df = pd.read\_csv('nameloc.csv')

df\_name = df['name'].to\_list()

df\_location = df['location'].to\_list()

my\_results\_list = []

nf = "not found

Email\_list=[]

Phone\_no=[]

Address\_list=[]

current\_loc = None

for i in range(len(df.name)):

query = df\_name[i]+" "+df\_location[i]

current\_loc = df\_location[i]

for j in search(query,

tld = 'com',

lang = 'en',

num = 10,

start = 0,

stop = 1,

pause = 2.0):

my\_results\_list.append(j)

for k in my\_results\_list:

urls = str(k)

page = requests.get(urls)

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

#FOR EMAIL \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

my\_list = []

for link in soup.find\_all('a'):

my\_list.append(link.get('href'))

emailregex = re.compile(r'''(

[a-zA-Z0-9.\_%+-:]+

@

[a-zA-Z0-9.-]+

\.[a-zA-Z]{2,4}

)''', re.VERBOSE)

my\_list = list(map(str, my\_list))

newlist = list(filter(emailregex.search, my\_list))

if newlist:

email = newlist.pop(0)

email = email[7:]

Email\_list.append(email)

else:

Email\_list.append(nf)

#FOR PHONE \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

div = soup.div

p = div.find\_all('p')

t = []

for i in p:

t.append(i.get\_text())

re1 = re.compile("(\+)?(91)? (\d){3} (\d){3} (\d){4}")

re2 = re.compile("(\+)?(91)?-(\d){3}-(\d){3}-(\d){4}")

re3 = re.compile("(\d){3} (\d){2} (\d){3} (\d){3}")

re4 = re.compile("(\d){10}")

re5 = re.compile("\+91(\d){10}" )

reList = [re1, re2, re3, re4, re5]

match = False

for regex in reList:

new = list(filter(regex.search, t))

if len(new)>=1:

match = True

break

if match:

phone = new.pop(0)

Phone\_no.append(phone)

#2

else:

span = div.find\_all('span')

t2 = []

for i in span:

t2.append(i.get\_text())

gotMatch = False

for regex in reList:

new2 = list(filter(regex.search, t2))

if len(new2)>=1:

gotMatch = True

break

if gotMatch:

phone2 = new2.pop(0)

Phone\_no.append(phone2)

else:

Phone\_no.append(nf)

"""

#FOR ADDRESS \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

a\_list = []

for ad in soup.findAll('div'):

a\_list.append(ad.get('p'))

word = current\_loc

word = str(word)

my\_re = r"^.\*" + re.escape(word) + r".\*$"

#add\_temp = re.search(my\_re, a\_list, re.IGNORECASE)

add\_temp = re.search(rf"^.\*%s.\*$" % word,a\_list,re.IGNORECASE)

if add\_temp:

address = add\_temp.group()

Address\_list.append(address)

else:

Address\_list.append(nf)

"""

#print('\n',Email\_list,'\n')

#print('\n',Phone\_no,'\n')

#print('\n',Address\_list,'\n')

dict = {'email': Email\_list, 'phone': Phone\_no}

df = pd.DataFrame(dict)

df.to\_csv('file1.csv')

# OUTPUT:

