Python - Assignment - 3.

1. Explain basic steps of project : maplt.py with webbrowser module.

→ The project : maplt.py with the webbrowser module. The webbrowser module's open() function can launch a new browser to a specified URL. Enter the following into the interactive shell:

```
import webbrowser
webbrowser.open('http://inventwithpython.com/')
```

Step 1: Figure out the URL.

Based on the instructions in Appendix B, set up maplt.py so that when you run it from the command line, like so...

```
C:\> mapit 870 valencia st, san fransisco, CA 94110
```

The script will use the command line arguments instead of the clip-board. If there are no command line arguments then the program will know to use the contents of the clipboard.

Step 2 :- Handle the command line Arguments make your code look like this:

```
#! python 3
# report.py - launches a map in the browser using an address
# command line or clipboard.
import webbrowser, sys
if len(sys.argv) > 1:
    # get address from command line.
    address = ' '.join(sys.argv[1:])
# TODO : get address from clipboard
```

Step 3: Handle the clipboard content & launch the Browser Make your code looks like the following.

```
#! Python 3
# mapIt.py - launches a map in the browser using an
```

```
# command line (or) clipboard.
import webbrowser, sys, pyperclip.
if len(sys.argv) > 1:
    # get address from command line. addr
    address = pyperclip.paste()
    webbr else:
    # get address from clipboard.
    address = pyperclip.past()
    webbrowser.open('https://www.google.com/maps/place/'+
                    address)
```

(5) How HTML can be posed with Beautiful Soup Module.

Beautiful Soup is a module for extracting information from an HTML page. The Beautiful Soup name is bs4

Creating a beautiful soup object from HTML:

The bs4.BeautifulSoup() function needs to be called with a string containing the HTML it will parse. The bs4.BeautifulSoup() function returns is a BeautifulSoup object.

Enter the following into the interactive shell while your computer is connected to the Internet:

```
>>> import requests, bs4.
>>> res = requests.get('http://nostpech.com')
>>> nostarchSoup = bs4.BeautifulSoup(res.text)
<class 'bs4.BeautifulSoup'>
```

You can also load an HTML file from your hard drive by passing a file object to bs4.BeautifulSoup()

Enter the following into the interactive shell:

```
>>> example file = open('example.html')
>>> example Soup = bs4.BeautifulSoup(example file)
>>> type(example Soup)
```

(3) Explain project: 'I'm feeling lucky' google search

→ Step 1: Get the command line Arguments & Request the search page.

```
#! python 3
# lucky.py - opens several Google search results import
requests, sys, webbrowser, bs4.
print (Googling...!)
res = requests.get ('http://google.com/search?q='+'
      join (sys.argv [1:])
res.raise_for_status()
# TODO: Retrieve top search result links
# TODO: open a browser tab for each result
```

STEP 2: find all the results

```
#! python 3.
# lucky.py - Opens several google search results.
import requests, sys, webbrowser, bs4.
--snip--
# Retrieve top. search result links. soup = bs4.BeautifulSoup
                                          (res.text)
# open a browser tab for each result link Elems = soup.select
('.r a')
```

Step 3: Open web Browser for Each Result

```
#! python 3
# lucky.py - opens several google search results.
import requests, sys, webbrowser, bs4.
--snip--
# open a browser tab for each result.
link Elems = soup.select ('.r a')
numOpen = min (5, len (link Elems))
```

```
for i in range (numopen):
webbrowser.open ('http://google.com' + linkElems[i].
get ('href'))
```

4. How excel document can be read. Explain with Example.

You can either create the spreadsheet yourself or download it from http://nostarch.com/automatestuff/ figure shows the tabs for the three default sheets named sheet1, sheet2, & sheet3 that Excel automatically provides for new workbooks.

(5) How to get sheets from workbook & call from sheets

→ Getting sheets from the workbook:

You can get a list of all sheet names in the workbook by calling the get_sheet_names() method. Enter the following into the interactive shell:

```
import openpyxl
wb = openpyxl.load_workbook('example.xlsx')
wb.get_sheet_names()
['Sheet1', 'Sheet2', 'Sheet3']
sheet = wb.get_sheet_by_name('Sheet3')
~~sheet = wb.get_she~~
sheet
<Worksheet "Sheet3">
type(sheet)
sheet.title
'Sheet3'
anotherSheet = wb.get_active_sheet()
anotherSheet.
```

Getting Cells from the sheets:

Once you have a worksheet object, you can access a cell object by its name. Enter the following into the interactive shell:

```
import openpyxl
wob = openpyxl.load_workbook('example.xlsx')
sheet = wob.get_sheet_by_name('sheet1')
>>> sheet['A1']
<cell sheet['A1'].
datetime.datetime[2021,4,5,13,34,2)
c=sheet['B1']
c.value.
'Apples'
'row' + str(c.row) + 'column' + c.column + 'is' + c.value.
'cell' + c.coordinate + 'c.coordinate + 'is' + c.value.
'cell B1 is Apples'
sheet['c1'] value.
13.
```