COMP 4971C Independent Project
Web Scraping Websites with Python for Database Construction

Student: HALIM, Kevin
E-mail: khalim@ust.hk

Supervised by: Dr. David Rossiter
Department of Computer Science and Engineering

Contents

1. Introduction

1.1. Purpose:

The purpose of this project is to support the DisPlay project, a new and innovative software that can transform the way people use their gadgets from individual smart device usage to multiple units working together, in particular, regarding the multiple device display in aggregate of one display. In this mode the image from 1 device is displayed on *n* devices, treating the *n* devices as 'virtual windows' into part of the display. The result is that the whole display is visible when all devices are viewed together. A simple illustrative figure is given below. In the proposed DisPlay System the devices can be physically organized in any creative and interesting way, making their usefulness far greater than would be the case if a fixed row of identical devices was used. This applies to businesses as well as individuals using the system. For example, an advertising business can use a creative array of varying size devices in a varying set of angles and distances from each other, but when viewed together give the appearance of small windows into a single display. This would be highly innovative, attractive and attention grabbing, not to mention highly affordable compared to the very high cost of comparable large screen displays.

Figure 1 Illustration of Display Project, in particular, the multiple device display in aggregate of one display

In the example above, the picture is showed realistically with the gadgets acting as “window” to view the picture. In order to achieve this, a database that contains information regarding the approximate size and length of a gadget’s bezel (the area on your gadget that is not the screen) will be needed.

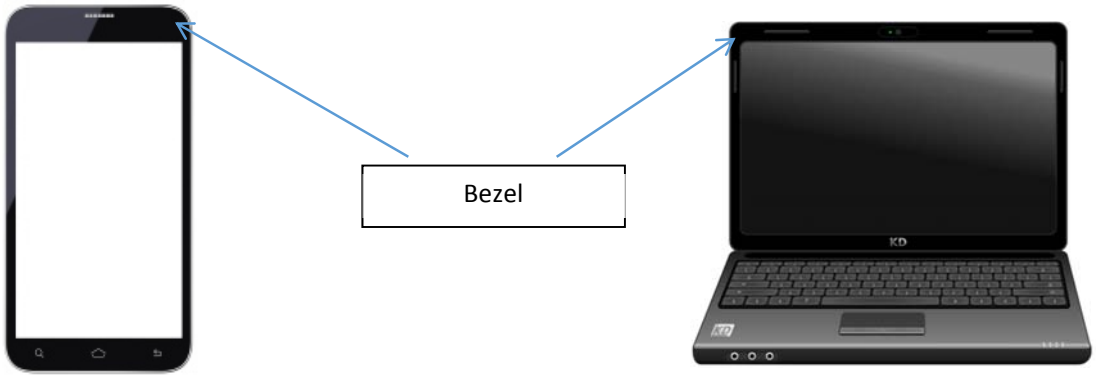

Figure 2 Illustration of Bezel

1.2. Scope:

The project covers 3 types of gadgets: smartphones, tablets and laptops, taken from 2 different websites, http://www.gsmarena.com/ and http://www.lapspecs.com/.

1.3. Method:

This project mainly utilizes Python Programming language with various libraries in order to access a specific page, retrieve relevant information and then save them in a txt file in a JSON (Javascript Object Notation) format automatically (hence, the term Web Scraper) so it can be accessed later on to retrieve the appropriate relevant data.

The libraries used in this program is string, requests, re, math, time, json and bs4

1.4. Limitations:

Since a program can take a lot of data in a short amount of time, this can result the scraper getting blocked by the website that is being scraped. In order to not get blocked, a time delay is used in between every data taken (approximately 2-4 seconds/data) to lower the chance of getting blocked, but this also results in a long time spent for constructing the database.

Also not all data is available in every webpage, some webpages may contain the whole information needed for the bezels, while others may not, hence there might be slight errors

and incomplete information in the database. In addition, some of the webpages are not uniformly presented and so in order to grab the necessary and correct data, standardization and many alternative cases must be considered and handled in order to prevent the program running into an error.

1.5. Assumptions:

We will assume that the information given in the website is correct and when we compute the bezels, we will assume that the screen of a certain gadget will be positioned exactly in the middle of that gadget. This will mean that the bezels will be symmetric in size, where both the left and right bezel will have the same lengths, same goes for the top and bottom bezel.

Also we will compute the width and depth/height of a screen with the following equation.

$$\text{Screen Width} = \sqrt{\frac{(\text{Screen in mm})^2 \times (\text{Width ratio})^2}{(\text{Width ratio})^2 + (\text{Height ratio})^2}}$$

&

$$\text{Screen Height or Depth} = \sqrt{(\text{Screen in mm})^2 - (\text{Screen Width})^2}$$

Figure 3 Equation for Computing Screen Width and Screen Height / Depth

With this, we will be able to compute the sides and top / bottom bezel with the following equation.

$$\text{Side bezel} = \frac{\text{Phone Width} - \text{Screen Width}}{2}$$

&

$$\text{Top/Bottom Bezel} = \frac{\text{Phone Height} - \text{Screen Height}}{2}$$

Figure 4 Side Bezel and Top / Bottom Bezel for Phones and Tablets

$$\text{Side Bezel} = \frac{\text{Laptop Width} - \text{Screen Width}}{2}$$

$$\text{Top/Bottom Bezel} = \frac{\text{Laptop Depth} - \text{Screen Depth}}{2}$$

Figure 5 Side Bezel and Top / Bottom Bezel for Laptops

2. Implementations

2.1. Phone + Tablet Bezels Database

The website used in this section is “http://www.gsmarena.com”

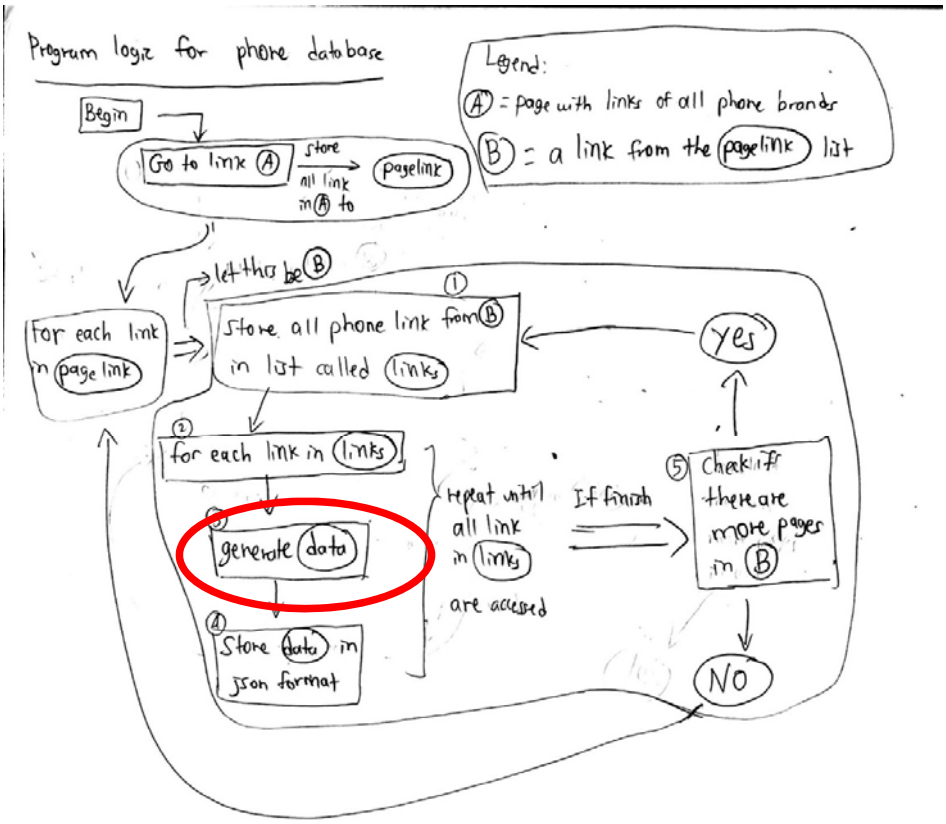

Figure 6 General Flow of the Scraper for Phone and Tablet Bezel Information Database

2.1.1. Preliminary preparations (data access and process)

The primary function in the program is the function that can access a certain webpage and get the data (noted by the red circle above), and in general the function works in the following way:

1. Access webpage with the “requests” library and turn the webpage into a set of html (HyperText Markup Language) text using the “beautiful soup” library.
2. Access and save certain sections in the html text where the data is accessed.
3. Process some of the sections taken from Step 2. For example, taking only numbers from a group of text, considering all cases of information available (whether or not a certain information is available or not. If some are not available then need to make certain adjustments to the values of certain variables, etc).
4. Print the data that has been taken and processed in step 2 and 3, to make sure that the program is working fine. Also, group all of the obtained values into 1 group so it can be saved into a JSON format.

The values that are saved into each entry are declared with the following variables:

Phone Name: name

Phone Height: height (obtained from the dimension of the gadget)

Phone Width: width (obtained from the dimension of the gadget)

Phone Diagonal in inch: inch (Taken from the display size of the gadget)

Phone Diagonal in mm: mm (Derived from the Phone Diagonal in inch)

Pixel Width: pixWidth (Obtained from the Resolution of the phone)

Pixel Height: pixHeight (Obtained from the Resolution of the phone)

Ratio of Width: ratioW (Obtained from the ratio of Pixel Width and Pixel Height)

Ratio of Height: ratioH (Obtained from the ratio of Pixel Width and Pixel Height)

Screen Width: screenW (Derived from the above values and equation from section 1.5.)

Screen Height: screenH (Derived from the above values and equation from section 1.5.)

Side Bezels: sidebezel (derived from the above values and equation from section 1.5.)

Top and Bottom Bezel: topbotbezel (derived from the above values and equation from section 1.5.)

Status: status (determines whether or not the data is complete / incomplete based on how complete the information above is available in a certain webpage)

Date Taken: currentdate

Taken From: mainlink (the link that the data above are taken from)

2.1.2. Scraping the website

Scraping the website GSMArena can be seen from diagram 1 above and the code works in the following way:

1. Access the page "http://www.gsmarena.com/makers.php3" and store all the links that contains the brand names of all available gadget information provided by GSMArena.
2. Each link obtained in step 1 will lead to a page that contains links of gadgets that each brand currently have. Hence, we will take all of these links and store them into another list. [Represented by the "get_links" function]
3. For each links in the list obtained at the end of step 2, access that particular link and run the function from the section above (2.1.1 Preliminary preparations (data access and process) to obtain and store the data of that particular gadget. [Represented by the "phoneinfo" function]
4. Repeat step 2 if the page that contains links of gadgets that each brand currently have is more than one (since 1 page only contain up to 40 links) [Represented by the "navpage" function]

Extra Note: Step 2 and 3 are represented as the "start" function and step 1 to 4 is represented by the "main" function. Also for all of the functions mentioned in this function, add a delay of 2 seconds to make sure that we don't take the data too quickly.

All of the code for all of the explanation above can be seen in appendix 1, with several comments (denoted by the # symbol at the beginning of a line).

2.2. Laptop Bezels Database

The website used in this section is “http://www.lapspecs.com”

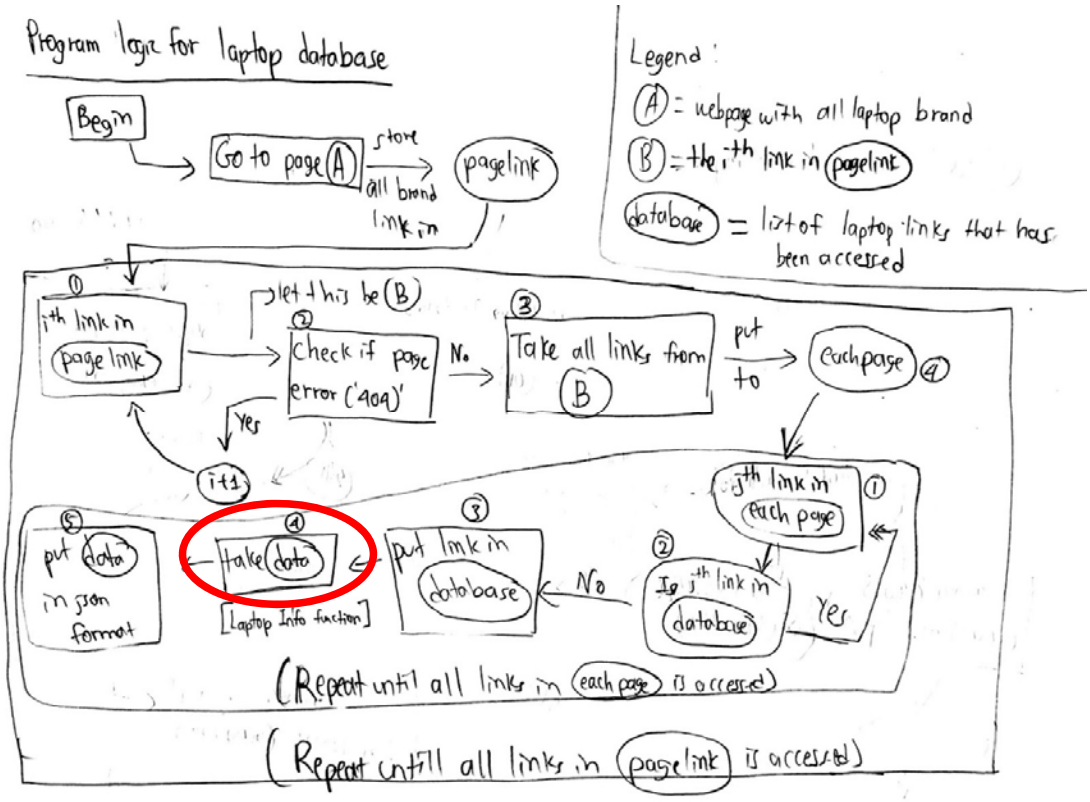

Figure 7 General Flow of the Scraper for Laptop Bezels Information Database

2.2.1. Preliminary preparations (data access and process)

The primary function in the program is the function that can access a certain webpage and get the data (noted by the red circle above), and in general the function works in the following way:

1. Access webpage with the “requests” library and check whether or not the webpage is available or gives out a page error, if it returns a page error, then exit the function, if not then proceed to step 2.
2. Turn the webpage into a set of html (HyperText Markup Language) text using the “beautiful soup” library.
3. Access and save certain sections in the html text where the data is accessed.
4. Process some of the sections taken from Step 2. For example, taking only numbers from a group of text, considering all cases of information available (whether or not a

certain information is available or not. If some are not available then need to make certain adjustments to the values of certain variables, etc).

5. Print the data that has been taken and processed in step 2 and 3, to make sure that the program is working fine. Also, group all of the obtained values into 1 group so it can be saved into a JSON format.

The values that are saved into each entry are declared with the following variables:

Laptop name = name

Screen Size in inch = screeninch (Either derived from screen size in mm or obtained from the screen section)

Screen Size in mm = screenmm (Either derived from screen size in inch or obtained from the screen section)

Screen Width = screenW (Obtained from the dimensions section)

Screen Depth = screenD (Obtained from the dimensions section)

Width Pixel = Wpix (Obtained from the screen section)

Depth Pixel = Dpix (Obtained from the screen section)

Ratio of Width = ratioW (Derived from ratio of Width Pixel and Depth Pixel)

Ratio of Depth = ratioD (Derived from ratio of Width Pixel and Depth Pixel)

Width of laptop = width (Derived from the above values and equation from section 1.5.)

Depth of laptop = depth (Derived from the above values and equation from section 1.5.)

Side Bezel = sidebezel (Derived from the above values and equation from section 1.5.)

Top / Bottom Bezel = topbotbezel (Derived from the above values and equation from section 1.5.)

Status = status (Either “Data Complete” or “Data Incomplete” depending on how complete the values above are available or not)

Date Taken = currentdate

Taken From = link (the link that the data above are taken from)

Extra = extra (Either “recheck”, if any of the sidebezels are too large / is negative, or “no abnormalities” if all of the data makes sense)

2.2.2. Scraping the website

Scraping the website LapSpecs.com can be seen from diagram 2 above and the code works in the following way:

1. Access the page "http://www.lapspecs.com/" and store all the links that contains the brand names of all available laptop information provided by LapSpecs.
2. For each links obtained from step 1, check if it returns a page error, if it does, skip the link and try the next link. If it doesn't return a page error, proceed to step 3.
3. Each link obtained in step 1 will lead to a page that contains all links of laptops that each brand currently have. Hence, we will take all of these links and store them into another list. [Represented by the "laptoplinks" function]
4. For each links in the list obtained at the end of step 3, check if it is already accessed by cross referencing that link with a list of links in the database list. If the link is not yet in the database, add that link in the database list and proceed to step 5. If the link is already in the database list, repeat step 4.
5. Access that particular link and run the function from the section above (2.2.1 Preliminary preparations (data access and process)) to obtain and store the data of that particular gadget. [Represented by the "laptopinfo" function], repeat step 4 until all links has been accessed. Then proceed to step 6.
6. Repeat step 2 to 5 for the next link obtained from step 1.

Extra Note: Step 1 to 6 is represented by the "main" function. Also for all of the functions mentioned in this function, add a delay of 2 seconds to make sure that we don't take the data too quickly.

All of the code for all of the explanation above can be seen in appendix 2, with several comments (denoted by the # symbol at the beginning of a line).

3. Results

At the end of the project, 2 databases (one for phones + tablets bezels and one for laptops bezels) have been made. For the phones + tablets bezels database, over 7949 data of gadgets have been taken and processed, while the laptop bezels database has over 2191 data.

The measurements of the side bezels and top/bottom bezels have also been cross checked with real measurements with only minor differences (around 0.1 to 0.4 mm difference). Hence we think that this database is quite accurate in terms of accuracy.

4. Future Recommendation

In relation to the database, it would be recommended to make another program that can let you obtain all of the data of a certain phone by only inputting the phone's name, most probably using php. This is done in order to be able to access the database much easier and more convenient.

In relation to the database creation, it would be recommended to use the technique illustrated in the laptop bezels database, in particular the part where you check whether or not you've accessed a certain link or not. This helps in cutting time of creating the database, especially when you're in the process of making the program, since if you encounter any problem in the middle of a database creation, being able to skip all the links that you've accessed will be much more efficient than going through all of them once more.

5. Appendix

Appendix 1: Phone Scraper Code

```
import requests
import re
import math
import time
import json
from bs4 import BeautifulSoup

url = "http://www.gsmarena.com/makers.php3"
root_url= "http://www.gsmarena.com/"
information = []

mainpage = requests.get(url)
soup = BeautifulSoup(mainpage.text,'html.parser')
table = soup.find('div', class_="st-text")
links = table.find_all('a')
#take all links that leads to a list of every gadget of a particular brand
pagelink = [a.attrs.get('href') for a in links]

#get links from a particular page from the pagelink list
def get_links(href):
  page = requests.get(root_url + str(href))
  soup = BeautifulSoup(page.text,'html.parser')
  table = soup.find('div',class_="section-body")
  links = table.find_all('a')
  return [a.attrs.get('href') for a in links]

#get phone info from 1 page
def phoneinfo(directlink):
  mainlink = root_url + str(directlink)
  response = requests.get(mainlink)
  soup = BeautifulSoup(response.content,'html.parser')
  currentdate = time.strftime("%c")
  name = soup.find("h1", class_="specs-phone-name-title").get_text()
  specs = soup.find("div", id="specs-list")
  table = specs.find_all("table")
  dimension = table[2].get_text()
  print("Phone Name:" + " " + name)

  d=[]
#  to collect only numbers from dimension text
  for t in dimension.split():
      try:
```

```
            d.append(float(t))
        except ValueError:
            pass

    screen = table[3].get_text()
    S=[]
#  to collect only numbers from the screen text
    for b in screen.split():
        try:
            S.append(float(b))
        except ValueError:
            pass

    if len(d) < 4 and len(S) > 2 :
        height = "-"
        width = "-"
        inch = S[0]
        mm = round (float(S[0])*25.4,2)
        pixWidth = S[1]
        pixHeight = S[2]
        ratio = round(int(S[1])/int(S[2]),3)
        if 0.6 > ratio >=0.562:
            ratioW = 9
            ratioH = 16
        elif 0.625> ratio >= 0.6:
            ratioW = 3
            ratioH = 5
        elif 0.667> ratio >= 0.625:
            ratioW = 10
            ratioH = 16
        elif 0.75> ratio >= 0.667:
            ratioW = 2
            ratioH = 3
        elif 0.8 > ratio >= 0.75:
            ratioW = 3
            ratioH = 4
        else:
            ratioW = 4
            ratioH = 5
        screenW = round(math.sqrt((math.pow(mm,2) *
math.pow(ratioW,2))/(math.pow(ratioW,2)+math.pow(ratioH,2))),2)
        screenH = round(math.sqrt((math.pow(mm,2) - math.pow(screenW,2))),2)
        sidebezel = "-"
        topbotbezel = "-"
        status = "Data Incomplete"
```

```
elif len(d) >= 4 and len(S)<=2:
    height = d[0]
    width = d[1]
    inch = "-"
    mm = "-"
    pixWidth = "-"
    pixHeight = "-"
    ratioW = "-"
    ratioH = "-"
    screenH = "-"
    screenW = "-"
    sidebezel ="-"
    topbotbezel = "-"
    status = "Data Incomplete"
elif len(d) < 4 and len(S)<=2:
    height = "-"
    width = "-"
    inch = "-"
    mm = "-"
    pixWidth = "-"
    pixHeight = "-"
    ratioW = "-"
    ratioH = "-"
    screenW ="-"
    screenH = "-"
    sidebezel = "-"
    topbotbezel = "-"
    status = "Data Incomplete"
else:
    height = d[0]
    width = d[1]
    inch = S[0]
    mm = round (float(S[0])*25.4,2)
    pixWidth = S[1]
    pixHeight = S[2]
    ratio = round(int(S[1])/int(S[2]),3)
    if 0.6 > ratio >=0.562:
        ratioW = 9
        ratioH = 16
    elif 0.625> ratio >= 0.6:
        ratioW = 3
        ratioH = 5
    elif 0.667> ratio >= 0.625:
        ratioW = 10
        ratioH = 16
```

```
      elif 0.75> ratio >= 0.667:
          ratioW = 2
          ratioH = 3
      elif 0.8 > ratio >= 0.75:
          ratioW = 3
          ratioH = 4
      else:
          ratioW = 4
          ratioH = 5
      screenW = round(math.sqrt((math.pow(mm,2) *
math.pow(ratioW,2))/(math.pow(ratioW,2)+math.pow(ratioH,2))),2)
      screenH = round(math.sqrt((math.pow(mm,2) - math.pow(screenW,2))),2)
      sidebezel = round((d[1]- screenW) / 2, 2)
      topbotbezel = round((d[0] - screenH) / 2, 2)
      status = "Data Complete"
  # print all information to double check
  print("Phone Height:"+ " " + str(height))
  print("Phone Width:" + " " + str(width))
  print("Phone Diagonal in inch:" + " " + str(inch))
  print("Phone Diagonal in mm:" + " " + str(mm))
  print("Aspect Ratio=" +str(ratioW)+":"+str(ratioH))
  print("Pixel Width:" +" " + str(pixWidth))
  print("Pixel Height:" + " " + str(pixHeight))
  print("Screen Width:" + " " + str(screenW))
  print("Screen Height:" + " " + str(screenH))
  print("Side Bezels:" + " "+ str(sidebezel))
  print("Top and Bottom Bezel:" + " "+ str(topbotbezel))
  print("")
  entry ={"name": name, "phoneH": height,"phoneW": width,"sizeinch": inch,"sizemm":
mm,"pixelW": pixWidth,"pixelH": pixHeight,"ratioW": ratioW,"ratioH": ratioH,"screenW":
screenW,"screenH": screenH,"sideB": sidebezel,"tbbezel": topbotbezel, "status":status,
"Date Taken":currentdate, "Taken From":mainlink}
  information.append(entry)
  time.sleep(2)

#  collect nav page info
def navpage(href):
  response = requests.get(root_url+str(href))
  soup = BeautifulSoup(response.text,'html.parser')
  table = soup.find('div', class_="nav-pages")
  links = table.find_all('a')
  return [a.attrs.get('href') for a in links]

#  program to start getting data from the 1st phone brand
def start(href):
```

```
  links = get_links(href)
  for directlink in links:
      phoneinfo(directlink)
      try:
          with open('Phone Database.txt','w') as p:
              json.dump(information, p, indent=1)
      except:
          continue
      time.sleep(2)
time.sleep(2)

# main program
def main():
  for href in pagelink:
       start(href)
          try:
              extrapages = navpage(href)
              for directlink in extrapages:
                  start(directlink)
          except:
              Continue
        time.sleep(2)

print(main())
```

Appendix 2: Laptop Scraper Code

```
import string
import requests
import re
import math
import time
import json
from bs4 import BeautifulSoup

url="http://www.lapspecs.com/"

page = requests.get(url)
soup = BeautifulSoup(page.text, 'html.parser')
maintable = soup.find("div", class_="levelnew")
links = maintable.find_all('a')
pagelink = [a.attrs.get('href') for a in links]
information = []
database =[]

print(pagelink)
try:
  # load laptop database
  with open('newlaptopdatabase.txt') as data_file:
      data = json.load(data_file)
  for eachdata in data:
      database.append(eachdata)

  # load laptop data
  with open('newlaptopdata.txt') as main_file:
      file = json.load(main_file)
  for eachfile in file:
      information.append(eachfile)
except:
  print("No Database yet")

def laptoplinks(href):
  page = requests.get(href)
  soup = BeautifulSoup(page.text,'html.parser')
  maintable = soup.find("div", class_="levelnew")
  links = maintable.find_all('a' )
  return [a.attrs.get('href') for a in links]

def laptopinfo(href):
  link = href
  page = requests.get(href)
```

```
status = page.status_code
if status ==  404:
    print("No Laptop Information Found")
    print("")
else:
    currentdate = time.strftime("%c")
    page = requests.get(href)
    soup = BeautifulSoup(page.text, 'html.parser')
    nametable = soup.find("div",class_="level2")
    name = nametable.find("td",class_="col1").get_text()
    specstable = soup.findAll("div", class_="level3")
    headings=[]
    content=[]

    for eachtable in specstable:
        col0 = eachtable.find_all("td", class_="col0")
        col1 = eachtable.find_all("td", class_="col1")
        for a in col0:
            headings.append(a.get_text())
        for b in col1:
            content.append(b.get_text())
    print(name)

    #check if table exist by checking the existence of the word "GB"
    count = 0
    for content in content:
        if ("GB" or 'gb') in content:
            count+=1

    # dimension table
    if count > 0:
        physical = specstable[7].find("td", class_="col1").get_text()
        m = re.findall("[-+]?\d+[\.]?\d*", physical)
        check = len(m)
        sortedM=[]
        # check to see whether or not a webpage contains the dimensions for a certain
        laptop
        if check < 1:
            width = "Unknown"
            depth = "Unknown"
        Else:
            # sort the numbers in dimension because even with different dimension, the
            sequence will always be dimensions in cm and then the dimensions in inch
            for dimension in m:
                size = float(dimension)
```

```
            sortedM.append(size)

        sortedM.sort(reverse=True)
        if 'cm' in physical:
            width = round(float(sortedM[0])*10,2)
            depth = round(float(sortedM[1])*10,2)
        elif ('inches' in physical) or ('inch' in physical) or ('"' in physical):
            width = round(float(sortedM[0])*25.4,2)
            depth = round(float(sortedM[1])*25.4,2)
        elif 'mm' in physical:
            width = float(sortedM[0])
            depth = float(sortedM[1])
        else:
            width = round(float(sortedM[0])*25.4,2)
            depth = round(float(sortedM[1])*25.4,2)
        print(sortedM)

    #screen details column
    screencol = specstable[3].find("td", class_="col1").get_text()
    screendetails = re.findall("[-+]?\d+[\.,]?\d*", screencol)
    i = len(screendetails)

    # delete comma from screendetails
    updatedscreendetails=[]
    for screen in screendetails:
        detail = screen.replace(',','')
        updatedscreendetails.append(detail)

    if i != 0:
        if float(updatedscreendetails[0]) > 20 and 'cm' in screencol:
            screeninch = round(float(updatedscreendetails[0]) * 0.0393701,2)
            screenmm = float(updatedscreendetails[0])
        elif float(updatedscreendetails[0])>20:
            screeninch = round(float(updatedscreendetails[0]) * 0.393701,2)
            screenmm = round(float(updatedscreendetails[0])*10,2)
        else:
            screeninch = updatedscreendetails[0]
            screenmm = round(float(updatedscreendetails[0]) * 25.4,2)

        # sort the numbers in screendetails that has no comma
        sorteddetails=[]
        for details in updatedscreendetails:
            det = float(details)
            sorteddetails.append(det)
```

```
            sorteddetails.sort(reverse=True)

            if (len(screendetails) < 3) and (('Widescreen' in screencol) or
('widescreen' in screencol) or ('WS' in screencol)):
                Wpix = "Not available"
                Dpix = "Not available"
                ratioW = 16
                ratioD = 9
                screenW = round(math.sqrt((math.pow(screenmm,2) *
math.pow(ratioW,2))/(math.pow(ratioW,2)+math.pow(ratioD,2))),2)
                screenD = round(math.sqrt((math.pow(screenmm,2) -
math.pow(screenW,2))),2)
                status = "Data Incomplete"
            elif (len(screendetails)>=3) and (('Widescreen' in screencol) or
('widescreen' in screencol) or ('WS' in screencol)):
                Wpix = sorteddetails[0]
                Dpix = sorteddetails[1]
                ratioW=16
                ratioD=9
                screenW = round(math.sqrt((math.pow(screenmm,2) *
math.pow(ratioW,2))/(math.pow(ratioW,2)+math.pow(ratioD,2))),2)
                screenD = round(math.sqrt((math.pow(screenmm,2) -
math.pow(screenW,2))),2)
                status = "Data Complete"
            elif len(screendetails)>=3 and float(updatedscreendetails [i-2])>100:
                Wpix = float(sorteddetails[0])
                Dpix = float(sorteddetails[1])
                ratio = round(float(sorteddetails[1])/float(sorteddetails[0]),3)
                if 0.6 > ratio >=0.562:
                    ratioW = 16
                    ratioD = 9
                elif 0.625> ratio >= 0.6:
                    ratioW = 5
                    ratioD = 3
                elif 0.667> ratio >= 0.625:
                    ratioW = 16
                    ratioD = 10
                elif 0.75> ratio >= 0.667:
                    ratioW = 3
                    ratioD = 2
                elif 0.8 > ratio >= 0.75:
                    ratioW = 4
                    ratioD = 3
                else:
                    ratioW = 5
```

```
                    ratioD = 4
                screenW = round(math.sqrt((math.pow(screenmm,2) *
math.pow(ratioW,2))/(math.pow(ratioW,2)+math.pow(ratioD,2))),2)
                screenD = round(math.sqrt((math.pow(screenmm,2) -
math.pow(screenW,2))),2)
                status = "Data Complete"
            elif len(screendetails)>=3 and float(updatedscreendetails [i-1])<100:
                Wpix = "Unavailable"
                Dpix = "Unavailable"
                ratio = round(float(screendetails[i-1])/float(screendetails [i-2]),3)
                if 0.6 > ratio >=0.562:
                    ratioW = 16
                    ratioD = 9
                elif 0.625> ratio >= 0.6:
                    ratioW = 5
                    ratioD = 3
                elif 0.667> ratio >= 0.625:
                    ratioW = 16
                    ratioD = 10
                elif 0.75> ratio >= 0.667:
                    ratioW = 3
                    ratioD = 2
                elif 0.8 > ratio >= 0.75:
                    ratioW = 4
                    ratioD = 3
                else:
                    ratioW = 5
                    ratioD = 4
                screenW = round(math.sqrt((math.pow(screenmm,2) *
math.pow(ratioW,2))/(math.pow(ratioW,2)+math.pow(ratioD,2))),2)
                screenD = round(math.sqrt((math.pow(screenmm,2) -
math.pow(screenW,2))),2)
                status = "Data Incomplete"
            else:
                ratioW = "Unavailable"
                ratioD = "Unavailable"
                Wpix = "Unavailable"
                Dpix = "Unavailable"
                screenW = "Unavailable"
                screenD = "Unavailable"
                status = "Data Incomplete"
            try:
                sidebezel = round((width- screenW) / 2, 2)
                topbotbezel = round((depth - screenD) / 2, 2)
                if sidebezel > 30 or sidebezel < 0 or topbotbezel<0:
```

```
                extra = "Recheck"
            else:
                extra = "No Abnormalities"
        except:
            sidebezel = "Unavailable"
            topbotbezel = "Unavailable"
            extra = "No Abnormalities"
    else:
        screeninch = "Unavailable"
        screenmm = "Unavailable"
        Wpix = "Unavailable"
        Dpix = "Unavailable"
        ratioW = "Unavailable"
        ratioD = "Unavailable"
        screenW = "Unavailable"
        screenD = "Unavailable"
        status = "Data Incomplete"
        sidebezel = "Unavailable"
        topbotbezel = "Unavailable"
        extra =" No Abnormalitites"
else:
    screeninch = "Unavailable"
    screenmm = "Unavailable"
    width = "Unavailable"
    depth = "Unavailable"
    Wpix = "Unavailable"
    Dpix = "Unavailable"
    ratioW = "Unavailable"
    ratioD = "Unavailable"
    screenW = "Unavailable"
    screenD = "Unavailable"
    sidebezel = "Unavailable"
    topbotbezel = "Unavailable"
    status = "No Data"
    extra =" No Abnormalitites"

print("Screen Size (mm)= " + str(screenmm))
print("Screen Width =" + str(screenW))
print("Screen Depth =" + str(screenD))
print("Width of laptop =" + str(width))
print("Depth of laptop =" + str(depth))
print("Width Pixel = " + str(Wpix))
print("Depth Pixel = " + str(Dpix))
print("Ratio of Width ="  + str(ratioW))
print("Ratio of Depth =" + str(ratioD))
```

```
      print("Side Bezel =" + str(sidebezel))
      print("Top Bot Bezel =" + str(topbotbezel))
      print("")

      entry ={"name": name, "laptopW": width,"laptopD": depth,"screeninch":
screeninch,"screenmm": screenmm,"pixelW": Wpix,"pixelD": Dpix,"ratioW": ratioW,"ratioD":
ratioD,"screenW": screenW,"screenD": screenD,"sideB": sidebezel,"tbbezel": topbotbezel,
"status":status, "Date Taken":currentdate, "Taken From":link, "Extra":extra}
      information.append(entry)
  time.sleep(2)

def main():
  for link in pagelink:
      page = requests.get(link)
      status = page.status_code
      if status == 404:
          print("Page Error")
          print("")
      else:
          eachpage = laptoplinks(link)
          for href in eachpage:
              if href in database:
                  print("Link Taken")
              else:
                  database.append(href)
                  laptopinfo(href)
                  with open('newlaptopdata.txt','w') as p:
                      json.dump(information, p, indent=1)
                  with open('newlaptopdatabase.txt','w') as w:
                      json.dump(database, w, indent=1)
                  time.sleep(2)
      time.sleep(2)

print(main())
```