UTM

UNIVERSITI TEKNOLOGI MALAYSIA

SMJE4383

ADVANCED PROGRAMMING (ADVANCED PROGRAMMING)

Assignment 1

Automate the CSV Generation Process using

Robotic Process Automation (RPA) & Python

SECTION: 1

1. **Muhammad Danish Danial bin Idjarmizuan A19MJ3036**
2. **Muhammad Haziq Aiman bin Kamarulzaman A19MJ3041**

Githublink : https://github.com/hhaziqaimann/ASSIGNGMENT-1-SMJE-4383-adv-programming

INTRODUCTION

The process of generating a CSV (Comma Separated Values) file typically involves the following steps:

1. Collecting data: The first step is to gather the data that you want to include in the CSV file. This data can come from a variety of sources, such as a database, spreadsheet, or manual input.
2. Organizing data: Once the data has been collected, it needs to be organized into a format that can be easily exported to a CSV file. This typically involves creating a table or spreadsheet with rows and columns that correspond to the data fields and records.
3. Saving as CSV: After the data has been organized, it can be exported to a CSV file using a spreadsheet program, such as Microsoft Excel or Google Sheets, or a programming language such as Python or R.
4. Importing the CSV: Finally, the CSV file can be imported into another program, such as a database or another spreadsheet, for further analysis or manipulation.

Meanwhile, we need to use RPA (Robotic Process Automation) and Python to execute the program. RPA is a technology that enables organizations to automate repetitive, normal processes that are usually carried out by humans. To simulate the actions of a human worker, such as accessing a website, filling out forms, or copying and pasting data, it utilize software robots, or "bots." It makes use of software robots, sometimes known as "bots," to mimic human operations such as visiting websites, completing forms, and copying and pasting data.

OBJECTIVES

The objectives for this assignment are to analyze, debug, solve and automate the CSV Generation Process using RPA & Python. We had manage to execute the task given by using the appropriate codes while try and error according to what we had learned during the subject.

METHODOLOGY

First, we scrap data from internet and print in terminal. After that we logged the data into CSV file that can be open using CSV Buddy. Then after program complete, we send the email to users by calling another python file that are made to send mail which are sendmail.py into the main python file which are assignment1.py

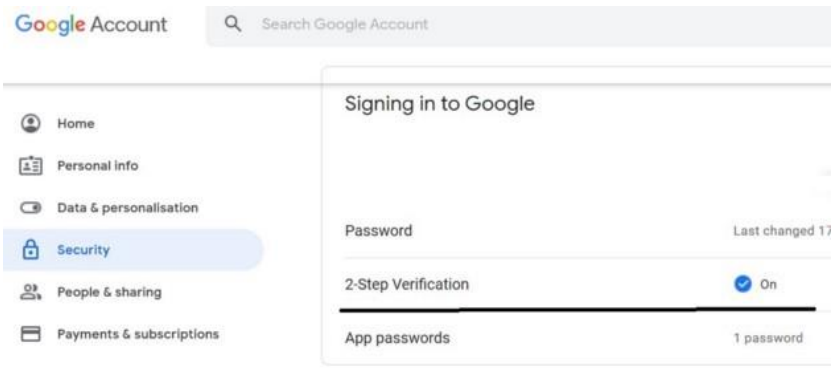
To scrape data from a website and display it in the terminal using Python, we use a library called Beautiful Soup. To make a request to a website, the Beautiful Soup library to parse the HTML content, and the print function to display the data in the terminal and the specific elements you want to scrape with the corresponding website that we want to scrape and the element we want to scrape.

Then we scrape data from a website and save it to a CSV file using Python, you also use a library called Beautiful Soup. Once the program is done the program will automatically create a CSV file and save it in the folder in our computer, and the file can be open using CSV buddy.

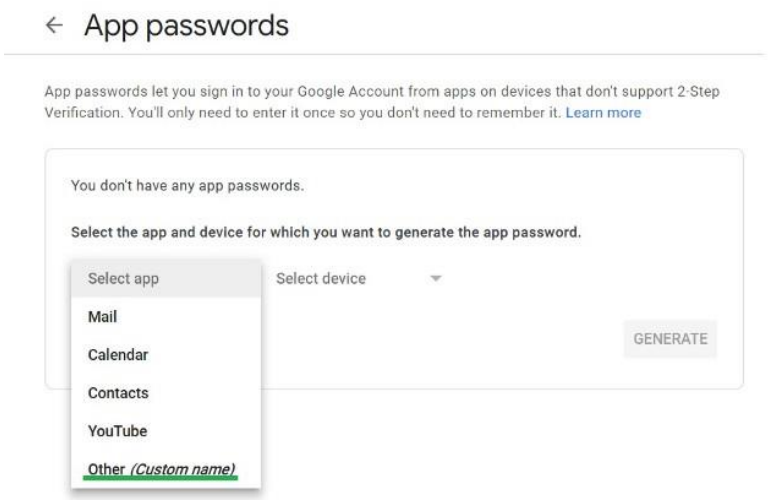
CSV Buddy is a tool that can be used to view, edit, and manipulate CSV (Comma Separated Values) files. It is a standalone application that runs on Windows operating system. CSV Buddy is a useful tool for anyone who needs to work with CSV files and it can save a lot of time and effort compared to manually editing and manipulating large CSV files in a text editor.

After the progress is done, we use python to send email to notify us That the program is done scrapping the data and the CSV file is now can be opened using CSV buddy.

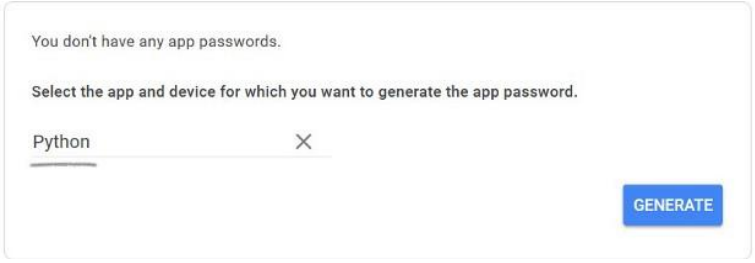
In this scenario, we'll use a Python script to send emails using a Gmail account, therefore we'll need to create an app password. The standard Gmail password can only be used for web login for security reasons. first activate 2-Step Verification. Simply select 2-Step Verification under your Google Account's Security > Signing in to Google, then follow the on-screen instructions.



Make an app password next. Simply choose "App passwords" under "2-Step Verification," and the window that appears will look like this. In the dropdown for "Select app," choose "Other."



Input a name, like Python, and then click "GENERATE." Be aware that this name is completely unrelated to the Python script and might be anything.



You will then receive a new app password. To use in your Python script, copy and store the 16-character password without a space.

Then using python, we use the built-in smtplib library in Python to send emails. to set the apps to send email,We need to code another file of python and combine or call those files in my main file which are the assigngment1.py.

Below are the full code for assignent1.py and sendmail.py: you can also click on https://github.com/hhaziqaimann/ASSIGNGMENT-1-SMJE-4383-adv-programming

assignment1.py:

```
import base64
import smtplib
import ssl
from email.mime.text import MIMEText
from email.utils import formatdate
from urllib.request import urlopen as uReq
from bs4 import BeautifulSoup as soup
from sendmail import sendmail

def ambildata():

    my_url = 'https://www.newegg.com/Desktop-Graphics-Cards/SubCategory/ID-
48?Tid=7709'
    uclient = uReq(my_url)
    page_html = uclient.read()
    uclient.close()
    page_soup = soup(page_html,"html.parser")

    containers = page_soup.findAll("div", {"class":"item-container"})

    file_name = "assignment1.csv"
    f = open(file_name, "w")

    headers = "brand , shipping , price \n"
    f.write(headers)



    for container in containers:
        brand = container.a.img["title"]
```

```
        # title_container = container.findAll("a", {"class":"item-titles"})
        # product_name = title_container[0].text

        shipping_container = container.findAll("li", {"class":"price-ship"})
        shipping = shipping_container[0].text.strip()

        price_list = container.findAll("li", {"class":"price-current"})
        price = price_list[0].text.strip().replace("|","").replace('\r',
'').replace('\n', '')


        print("brand : " + brand)
        # print("product_name : " + product_name)
        print("shipping :  " + shipping)
        print("price : " + price)
        print("_____________________________________________________________
_____________________________________")


        f.write(brand.replace(",","|") + "," + shipping + "," +
price.replace(",",".") + "\n")

    f.close()

def complete(self):
        sendmail()
if __name__ == "__main__":
    ambildata()
    sendmail()
```

sendmail.py:

```
import base64
import smtplib
import ssl
from email.mime.text import MIMEText
```

```
from email.utils import formatdate

def sendmail():
    main_text = "complete scrap data from website and export to CSVbuddy "
    charset = "utf-8"
    if charset == "utf-8":
        msg = MIMEText(main_text, "plain", charset)
    elif charset == "iso-2022-jp":
        msg = MIMEText(base64.b64encode(main_text.encode(charset, "ignore")),
"plain", charset)


    msg.replace_header("Content-Transfer-Encoding", "base64")
    msg["Subject"] = "!!COMPLETE EXPORT CSV!!"
    msg["From"] = "aimanhaziq0978@gmail.com"
    msg["To"] = "danishdnial1998@gmail.com"
    msg["Date"] = formatdate(None,True)


    host = "smtp.gmail.com"
    nego_combo = ("ssl", 465)

    if nego_combo[0] == "no-encrypt":
        smtpclient = smtplib.SMTP(host, nego_combo[1], timeout=10)
    elif nego_combo[0] == "starttls":
        smtpclient = smtplib.SMTP(host, nego_combo[1], timeout=10)
        smtpclient.ehlo()
        smtpclient.starttls()
        smtpclient.ehlo()
    elif nego_combo[0] == "ssl":
        context = ssl.create_default_context()
        smtpclient = smtplib.SMTP_SSL(host, nego_combo[1], timeout=10,
context=context)
    smtpclient.set_debuglevel(2)


    username = "aimanhaziq0978@gmail.com"
    password = "uiegevqstlnqwadm"
    smtpclient.login(username, password)
```

```
    smtpclient.send_message(msg)
    smtpclient.quit()

if __name__ == "__main__":
    sendmail()
```

The function `def ambildata()` are for scrapping data from the website https://www.newegg.com/Desktop-Graphics-Cards/SubCategory/ID-48?Tid=7709

And the data that we scrapping are GPU type, the spiing cost and the price of each GPU. Then all the data are written in CSV file named assignment1.py

The function `def complete()` are for calling the sendmail.py file which are used for send mail when `def ambildata()` is completed.

RESULT

By using beautifulsoup libraries we successfully scrap data from https://www.newegg.com/Desktop-Graphics-Cards/SubCategory/ID-48?Tid=7709

and extract the data that we want whick are the list of all the GPU that are on those website and the cost of shipping and the price of each GPU. Figure 1 below shown the data that we scrap using beautifulaoup and display it in the terminal

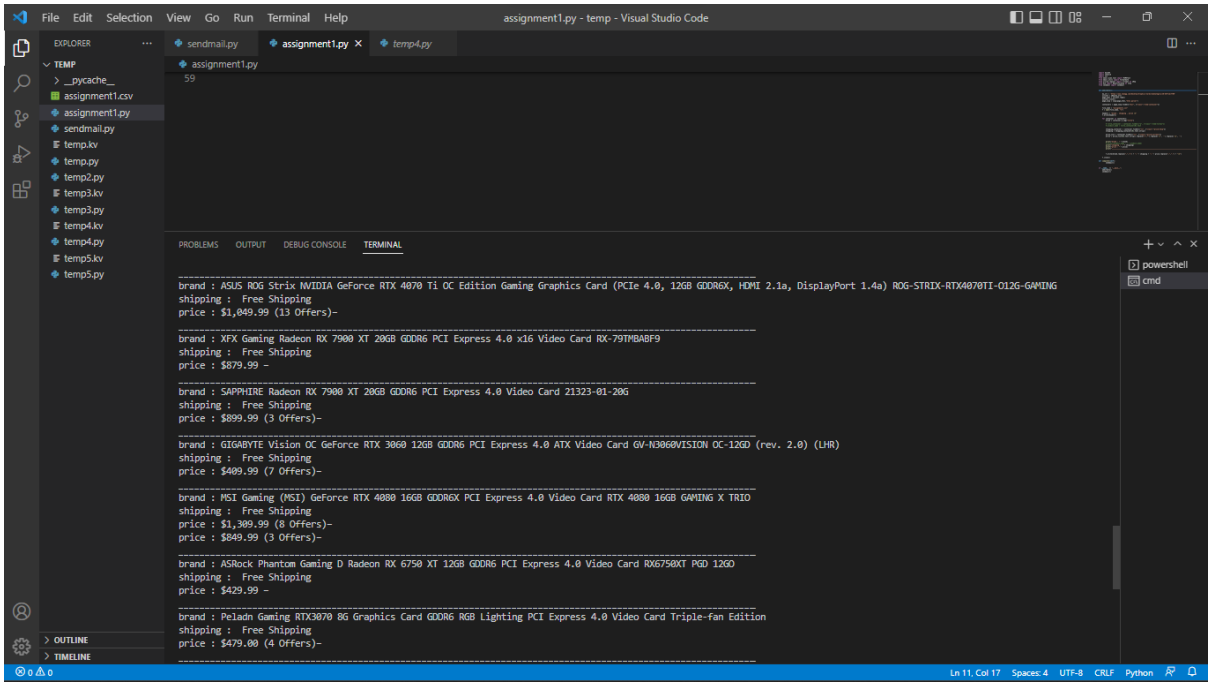


Figure 1: the data scaped are shown in terminal

Then we write all those data into csv file by writing this line in our program

```
file_name = "assignment1.csv"

f = open(file_name, "w")

headers = "brand , shipping , price \n"

f.write(headers)
```

and the output are a CSV file that can be opened using CSV Buddy name assignment1.py are automatically created as shown as figure 2 below and the figure 3 shown the CSV file in the CSV buddy.

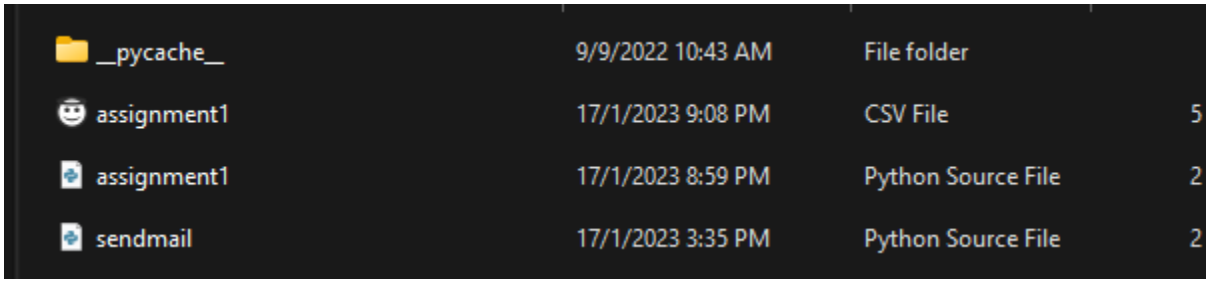


Figure 2: a CSV file named assignment1 is automatically create in the folder of the program

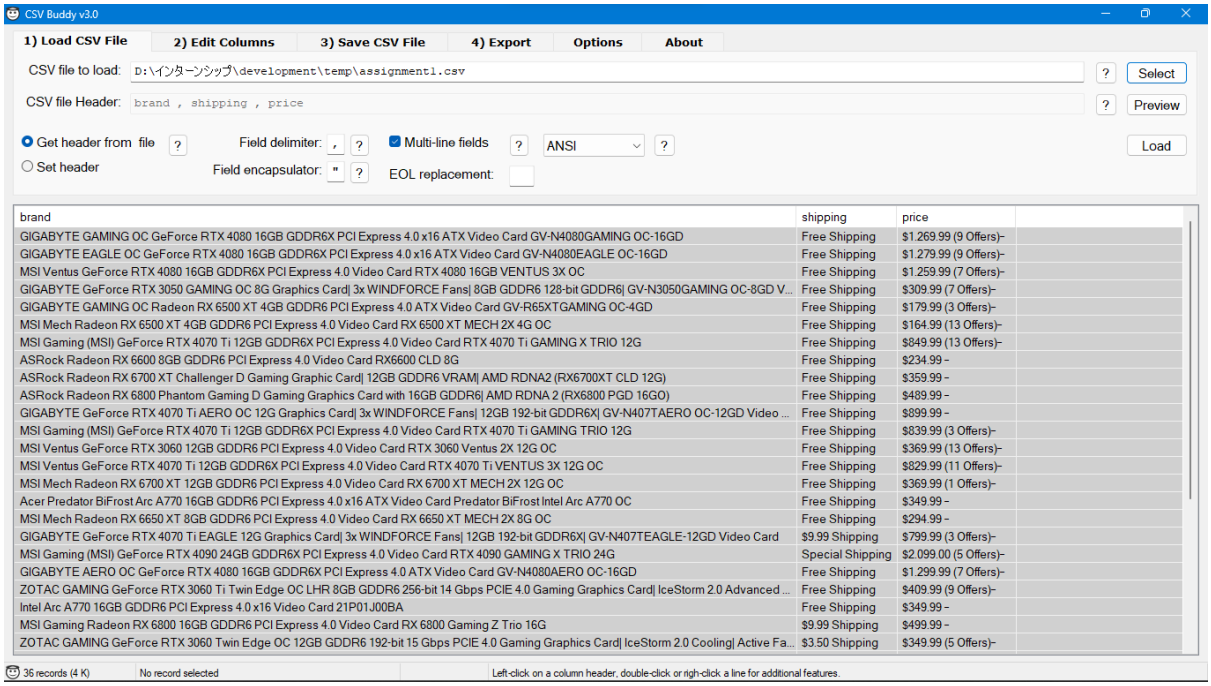


Figure 3: The assigment1.csv file opened in CSV Buddy

And lastly when the program is done it will send email by calling the sendmail.py file to the assignmet1.py file. Figure 4 below show the email that are send to my mail when type of GPU, shipping price and GPU price are done scrapped from the website and all the data are write into CSV is completed.

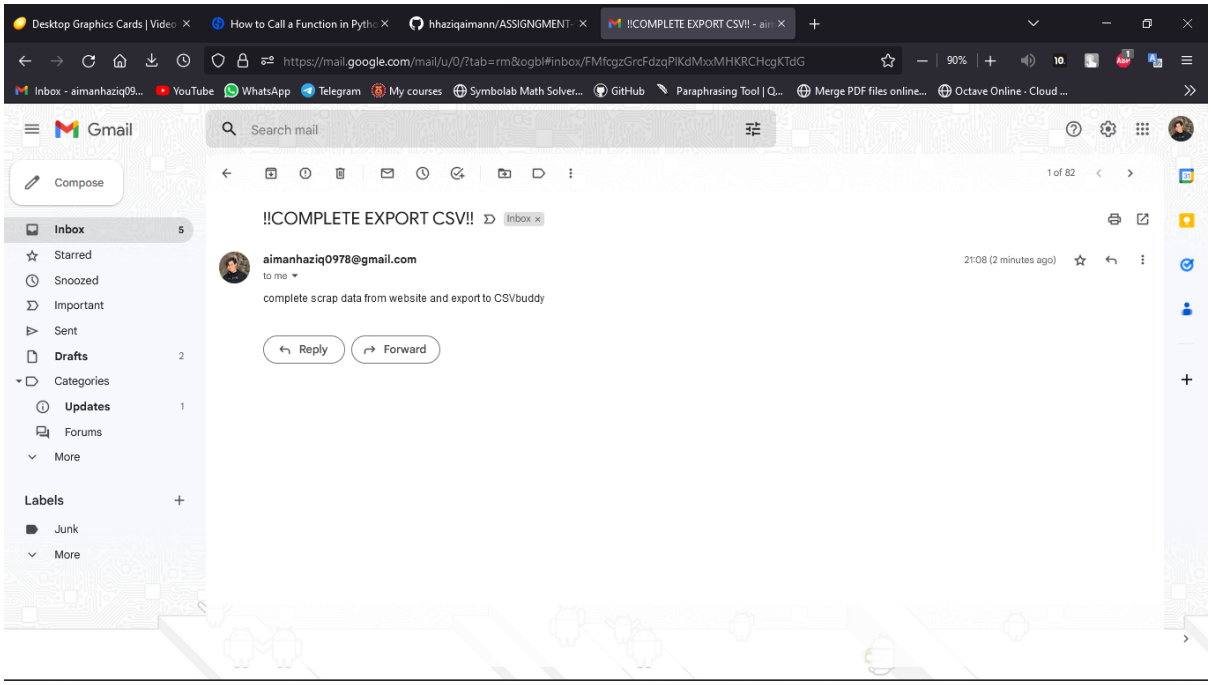


Figure 4: email are send when the CSV file is created

CONCLUSION

The conclusion of robotic process automation (RPA) is that it is a powerful technology that can significantly improve the efficiency and effectiveness of businesses by automating repetitive, routine tasks. RPA can reduce the amount of time and effort required to complete these tasks, increase accuracy and consistency, and reduce the risk of errors. Additionally, RPA can be integrated with other technologies, such as artificial intelligence (AI) and machine learning (ML), to create more advanced automation solutions.

RPA can be applied across a wide range of industries and business functions, such as finance, human resources, customer service, and supply chain management. It can help organizations to improve their operations and reduce costs.

However, it is important to note that RPA is not a one-size-fits-all solution, and it may not be suitable for all tasks or organizations. Before implementing RPA, it is important to carefully evaluate the potential benefits and costs, as well as any potential drawbacks or limitations. Additionally, it is important to have clear processes and governance in place to ensure that RPA is implemented and maintained in a way that aligns with the organization's goals and values.

Python can be used to automate various tasks by interacting with the user interface of the system being automated. For example, Python can be used to automate tasks such as filling out forms, clicking buttons, and extracting data from websites. This can be done using Python libraries such as Selenium and Pyautogui, which provide a way to control the mouse and keyboard, and interact with web pages. Python can also be used to automate tasks that involve working with data. For example, it can be used to extract data from spreadsheets, CSV files, or other sources, and then manipulate or analyze the data using libraries such as Pandas and Numpy.

RPA frameworks like Selenium and Pyautogui are widely used to automate repetitive tasks in a variety of industries, including finance, healthcare, and customer service. These frameworks can also be integrated with other technologies such as machine learning and artificial intelligence to create more advanced automation solutions.

REFERENCES

Eric Allman (1983), Sendmail – An Internetwork Mail Router (PDF), BSD UNIX documentation set, Berkeley: University of California, archived (PDF) from the original on May 20, 2013, retrieved June 29, 2012

"MAIL Parameters". IANA. February 14, 2020. Archived from the original on May 28, 2019. Retrieved May 28, 2019.

Thapelo, Tsaone Swaabow; Namoshe, Molaletsa; Matsebe, Oduetse; Motshegwa, Tshiamo; Bopape, Mary-Jane Morongwa (2021-07-28). "SASSCAL WebSAPI: A Web Scraping Application Programming Interface to Support Access to SASSCAL's Weather Data". Data Science Journal. ***20****: 24. doi:10.5334/dsj-2021-024. ISSN 1683-1470. S2CID 237719804.*

Song, Ruihua; Microsoft Research (Sep 14, 2007). "Joint Optimization of Wrapper Generation and Template Detection" (PDF). The 13th International Conference on Knowledge Discovery and Data Mining: 894. doi:10.1145/1281192.1281287. ISBN 9781595936097. S2CID 833565. Archived from the original (PDF) on October 11, 2016.