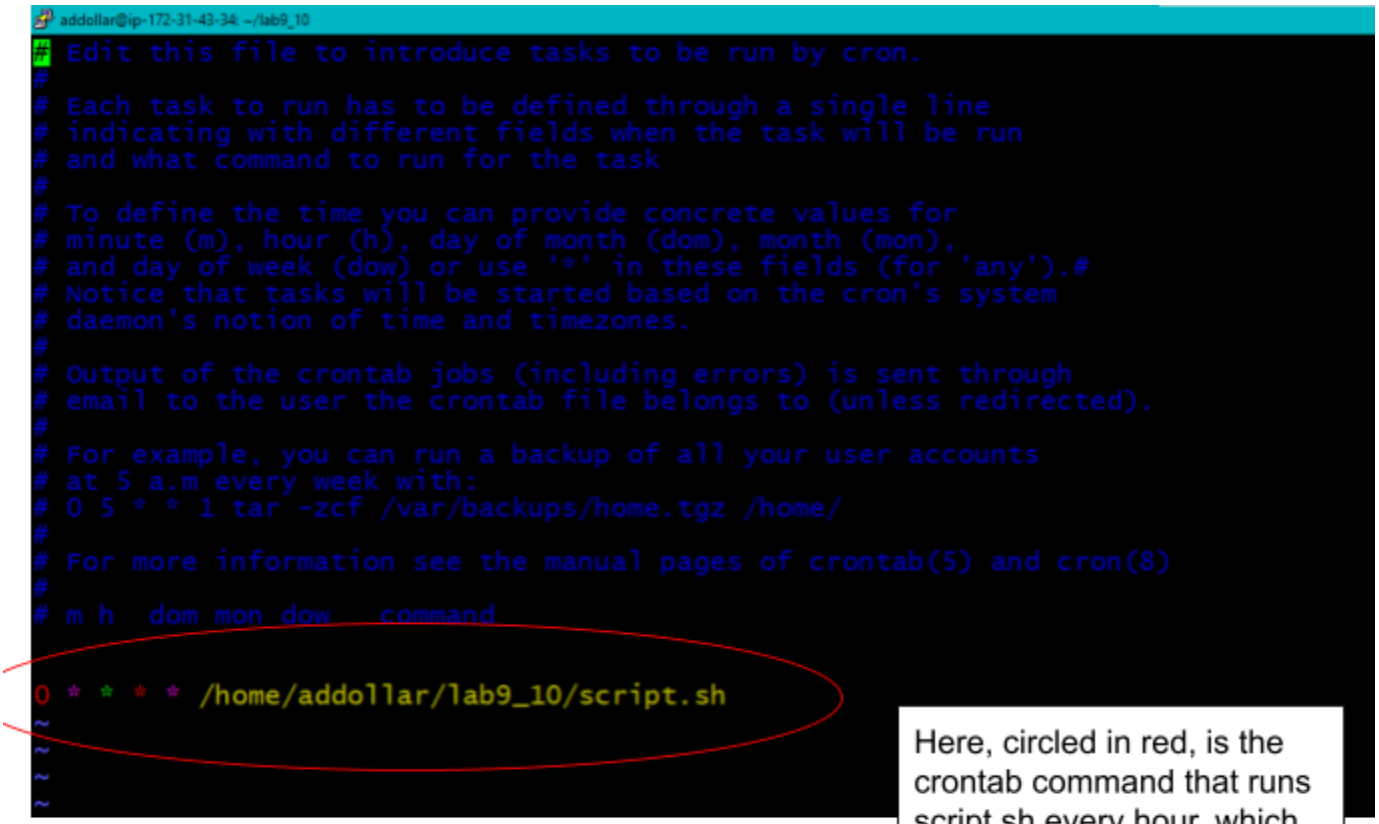
Austin Dollar
4/8/21
CSCI 344
Lab 9/10

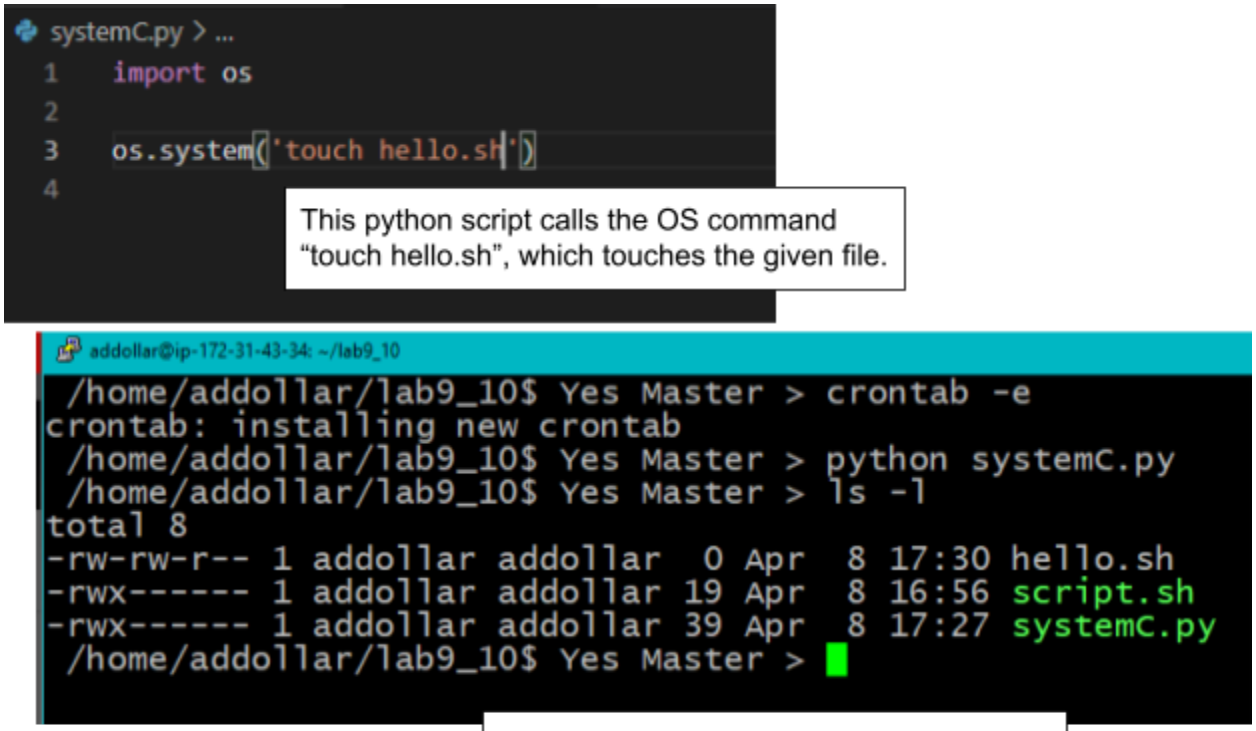
Data Parsing/Beautiful Soup

Crontab that executes every hour

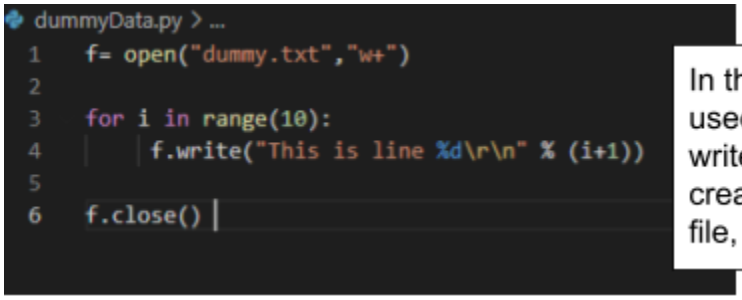


Here, circled in red, is the crontab command that runs script.sh every hour, which just outputs “hello world”

System command “touch”



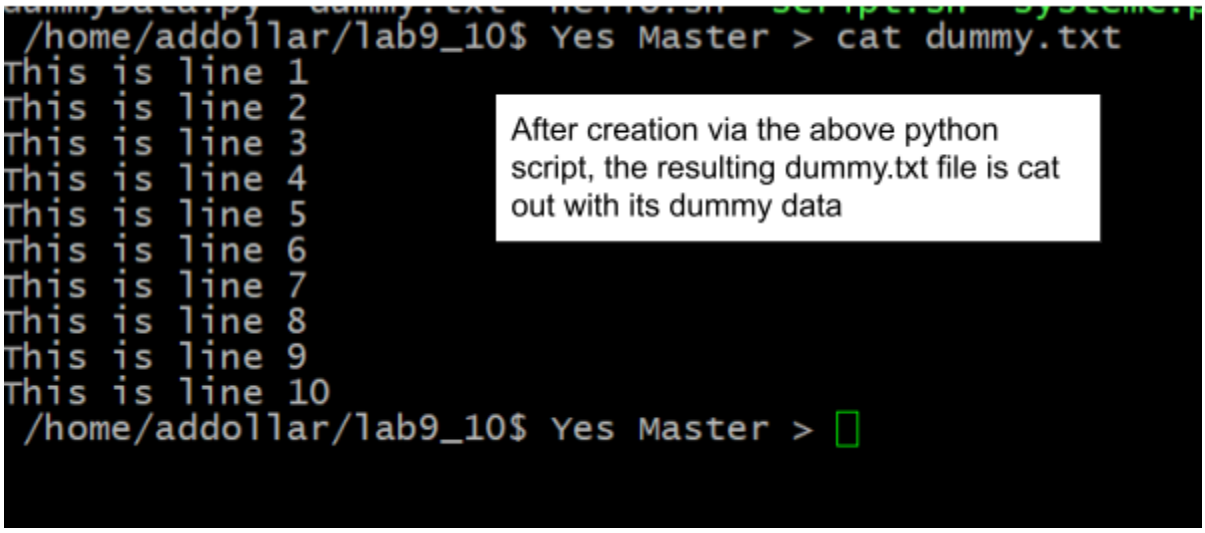
Create file with dummy data



dummyData.py > ...

```
1   f= open("dummy.txt","w+")
2
3   for i in range(10):
4       f.write("This is line %d\r\n" % (i+1))
5
6   f.close()
```

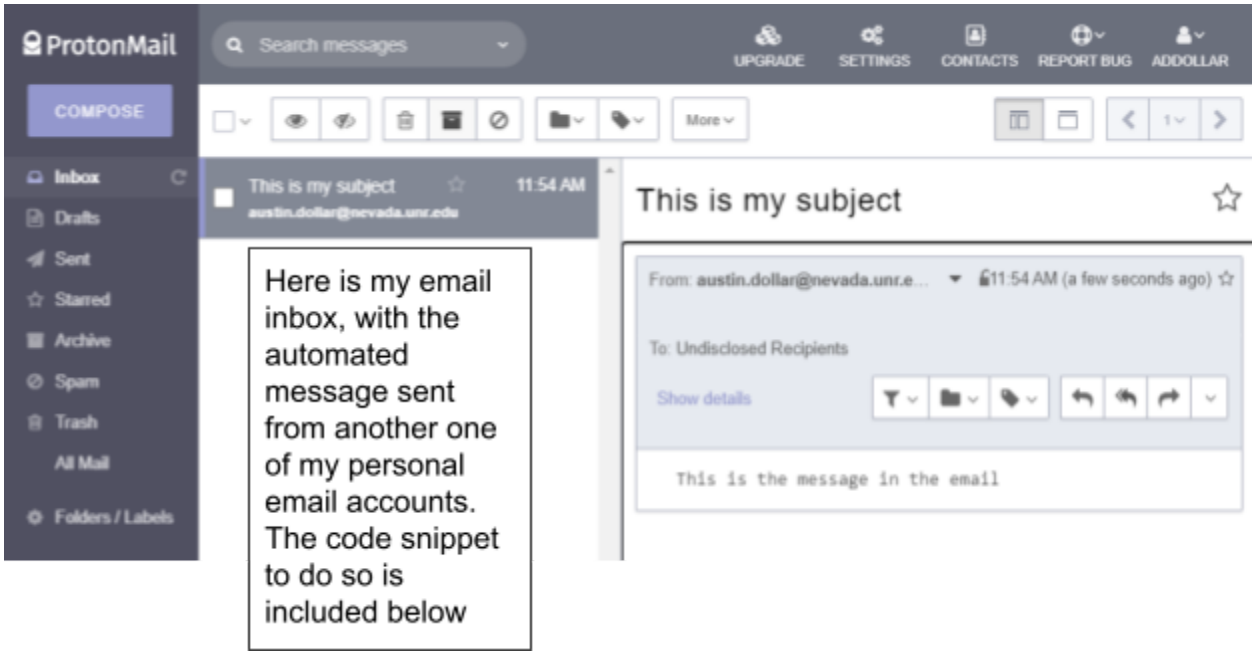
In this code snippet I used the open and write function to create and write to a file, “dummy.txt”



```
/home/addollar/lab9_10$ Yes Master > cat dummy.txt
This is line 1
This is line 2
This is line 3
This is line 4
This is line 5
This is line 6
This is line 7
This is line 8
This is line 9
This is line 10
/home/addollar/lab9_10$ Yes Master >
```

After creation via the above python script, the resulting dummy.txt file is cat out with its dummy data

Automate email sending




```python
import smtplib


server = smtplib.SMTP(host='smtp.gmail.com', port=587)

server.ehlo()

server.starttls()

server.login('austin.dollar@nevada.unr.edu', '1Kaladog')

subject = 'This is my subject'
body = 'This is the message in the email'

message = f'Subject: {subject}\n\n{body}'

server.sendmail('austin.dollar@nevada.unr.edu', 'addollar@protonmail.com'
server.quit()
```

Python Script for scraping, emailing if .csv changed

The Python Code

This is my python web scraping script. It searches the given craigslist sporting goods section across all pages. It then looks for listings that are above $20, and below $250. It also looks for listings with the keyword “fish” to find all fishing products. “fish “ is spelled with a variety of capitalizations to grab as many listings as possible. Then, the found matches are then stored in a .csv file titled “results.csv”. This file is then sorted by price via pandas.
If the script has been ran previously, a sorted_results file will already exist. In this case, we rename the file “oldata” before passing in new data. After the script populates sorted_results, it compares itself to the old data, if it exists. If it is different, or old data does not exist, meaning this is the first run of the script, it sends an email notification.(Code Copied and pasted for readability)

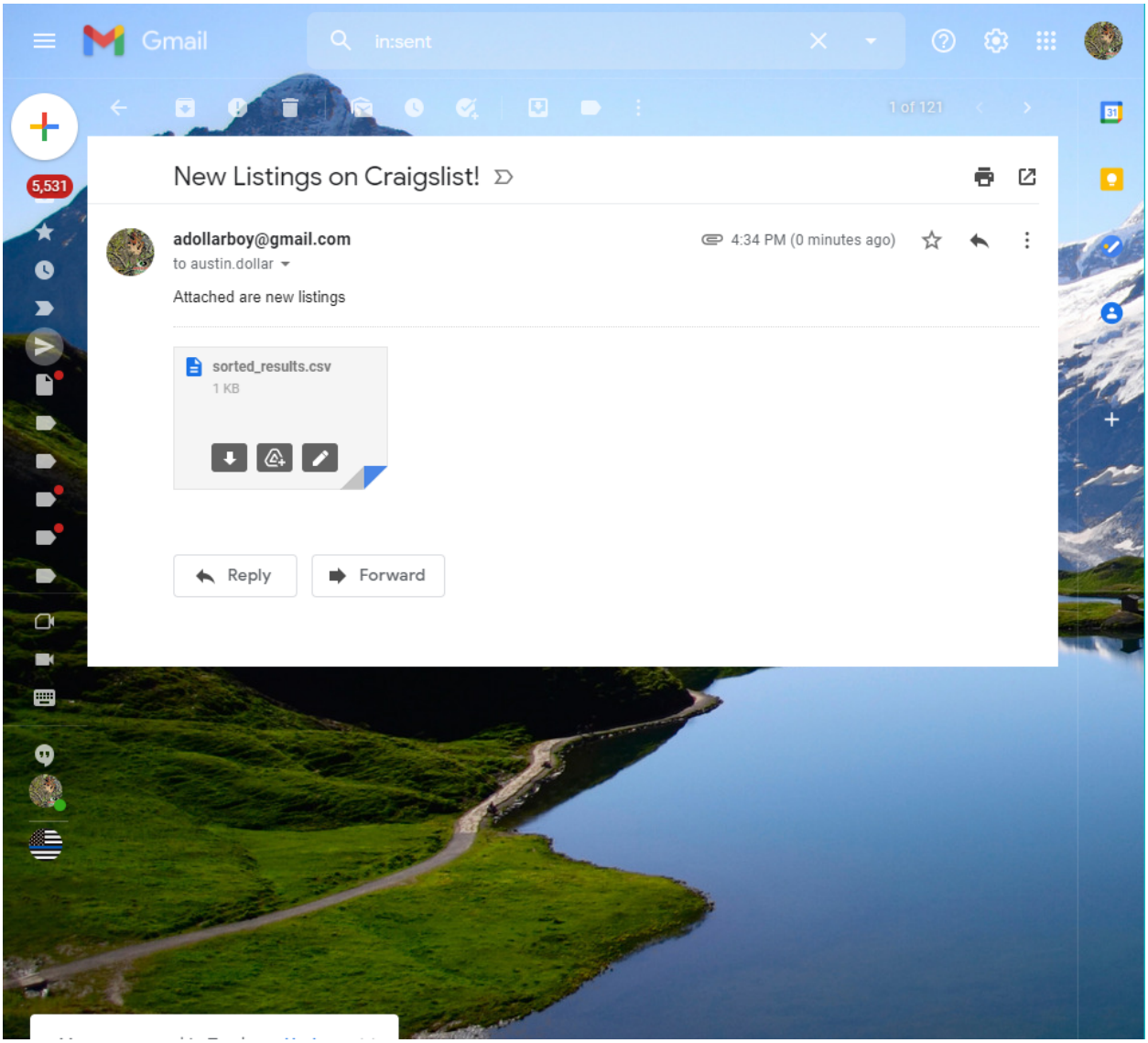
```
import pandas as pd
import smtplib
import os
import email
import filecmp
import ssl
from email import encoders
from email.mime.base import MIMEBase
from email.mime.multipart import MIMEMultipart
from email.mime.text import MIMEText
from bs4 import BeautifulSoup
from urllib.request import urlopen
from urllib.request import urljoin

exist = os.path.isfile("sorted_results.csv")

if exist == True:
    os.rename('sorted_results.csv' , 'oldata.csv')

site = 'http://chico.craigslist.org/search/sga/'
html = urlopen(site)
file = open("results.csv", "w") # "a" to append
file.write("Title,Price,City\n")
counter1 = 0
counter2 = 0
soup = BeautifulSoup(html, 'html.parser')
```

```
for listing in soup.find_all('li', class_='result-row'):
    counter1 += 1
    i = listing.find('span', class_='result-price')
    #print(counter1,"value of i = ",i)
    if i != None:
        lines = listing.text.splitlines()
        price = listing.text[3:6]
        name = lines[10]
        city = lines[14]
        city = city.replace("(", "")
        city = city.replace(")", "")
        city = city.replace(" ", "")

        if "," in name:
            name = name.replace(",", "")


        if "," not in price and "\nfa" not in price and "fish" in name or "FISH"
in name or "Fish" in name:
            price = int(price)
            if price > 20 and price <=250:
                print('HHHHHHHHHHHHHHHHH FOUND HHHHHHHHHHHHHHHHHHHHHHHHHHHHH')
                print(price)
                print(name)
                print(city)

                file.write(name)
                file.write(",")
                file.write(str(price))
                file.write(",")
                file.write(city)
                file.write(" \n")

                print("--------------------------")


file.close()

df = pd.read_csv("results.csv")
```

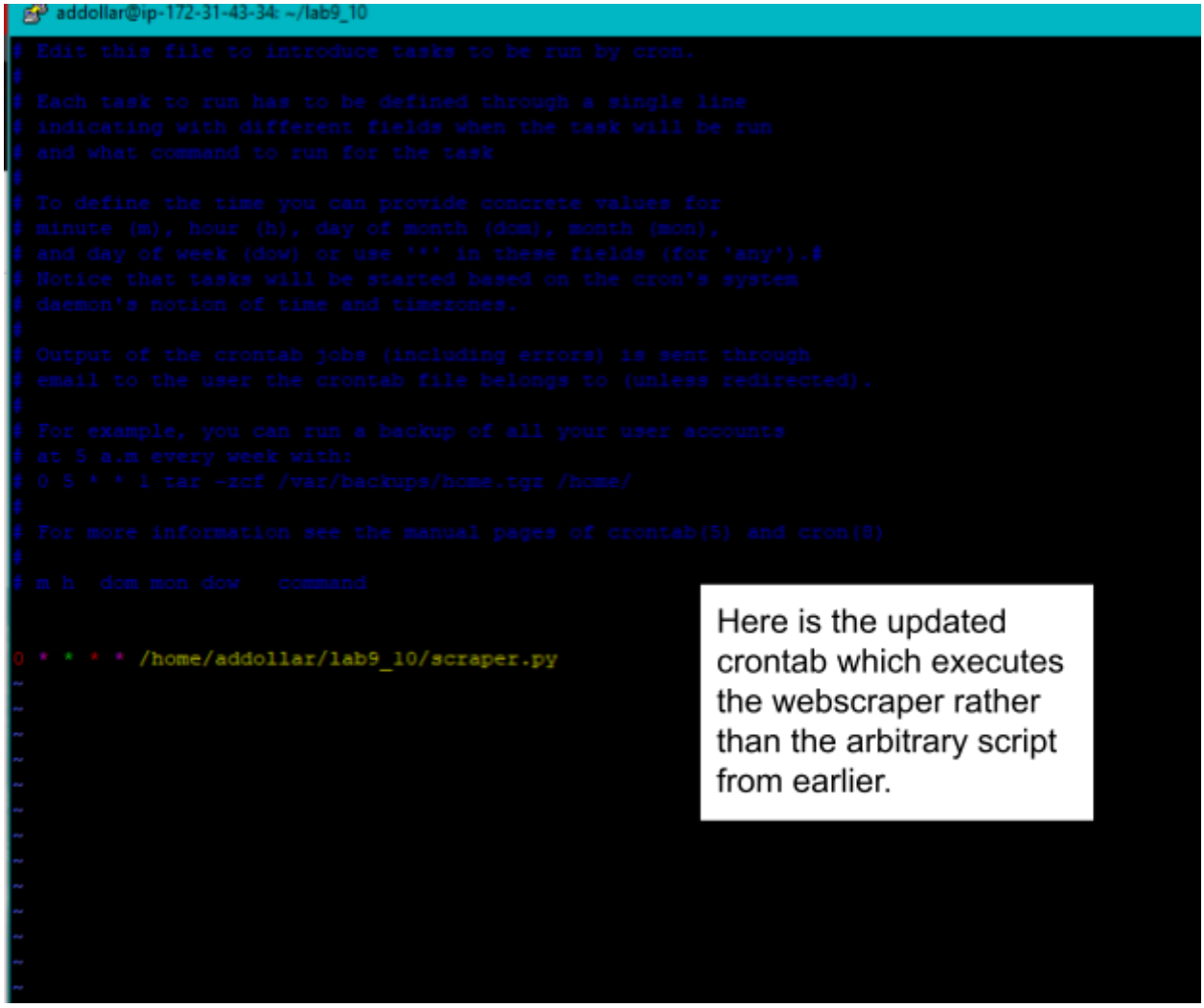
```
sorted_df = df.sort_values(by=["Price"], ascending=True)

sorted_df.to_csv('sorted_results.csv', index=False)


exist2 = os.path.isfile("oldata.csv")

if exist2 == False:
    os.system('touch oldata.csv')

old = "oldata.csv"
new = "sorted_results.csv"

result = filecmp.cmp(old, new)

if result == False:
    subject = "New Listings on Craigslist!"
    body = "Attached are new listings"
    sender_email = "adollarboy@gmail.com"
    receiver_email = "austin.dollar@nevada.unr.edu"
    password = "1Austind"

    message = MIMEMultipart()
    message["From"] = sender_email
    message["To"] = receiver_email
    message["Subject"] = subject

    message.attach(MIMEText(body, "plain"))

    file_to_attach = "sorted_results.csv"

    attach_file = open(file_to_attach, 'rb')
    payload = MIMEBase('application', 'octate-stream')
    payload.set_payload((attach_file).read())
    encoders.encode_base64(payload) #encode the attachment
    payload.add_header('Content-Disposition', 'attachment',
filename=file_to_attach)
    message.attach(payload)
```

```
session = smtplib.SMTP('smtp.gmail.com', 587)
session.starttls()
session.login(sender_email, password)
text = message.as_string()
session.sendmail(sender_email, receiver_email, text)
session.quit()
print('Mail Sent')
```

The email with updated .csv file



Adding script to cron scheduler



State if complete or if lacks certain things

As far as I can determine, my code does not lack anything, as it meets all requirements outlined in the lab documentation.