

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

## Data Parsing/Beautiful Soup

Crontab that executes every hour

```
addollar@ip-172-31-43-34: ~/lab9_10
# Edit this file to introduce tasks to be run by cron.
#
# Each task to run has to be defined through a single line
# indicating with different fields when the task will be run
# and what command to run for the task
#
# To define the time you can provide concrete values for
# minute (m), hour (h), day of month (dom), month (mon),
# and day of week (dow) or use '*' in these fields (for 'any').#
# Notice that tasks will be started based on the cron's system
# daemon's notion of time and timezones.
#
# Output of the crontab jobs (including errors) is sent through
# email to the user the crontab file belongs to (unless redirected).
#
# For example, you can run a backup of all your user accounts
# at 5 a.m every week with:
# 0 5 * * 1 tar -zcf /var/backups/home.tgz /home/
#
# For more information see the manual pages of crontab(5) and cron(8)
#
# m h  dom mon dow  command
0 * * * * /home/addollar/lab9_10/script.sh
~
~
~
```

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

## System command “touch”

```
systemC.py > ...  
1  import os  
2  
3  os.system('touch hello.sh')  
4
```

This python script calls the OS command “touch hello.sh”, which touches the given file.

```
addollar@ip-172-31-43-34: ~/lab9_10  
/home/addollar/lab9_10$ Yes Master > crontab -e  
crontab: installing new crontab  
/home/addollar/lab9_10$ Yes Master > python systemC.py  
/home/addollar/lab9_10$ Yes Master > ls -l  
total 8  
-rw-rw-r-- 1 addollar addollar  0 Apr  8 17:30 hello.sh  
-rwx----- 1 addollar addollar 19 Apr  8 16:56 script.sh  
-rwx----- 1 addollar addollar 39 Apr  8 17:27 systemC.py  
/home/addollar/lab9_10$ Yes Master > █
```

As shown in the above terminal snippet, the hello.sh file was last accessed when I called the systemC.py command just above it

## Create file with dummy data

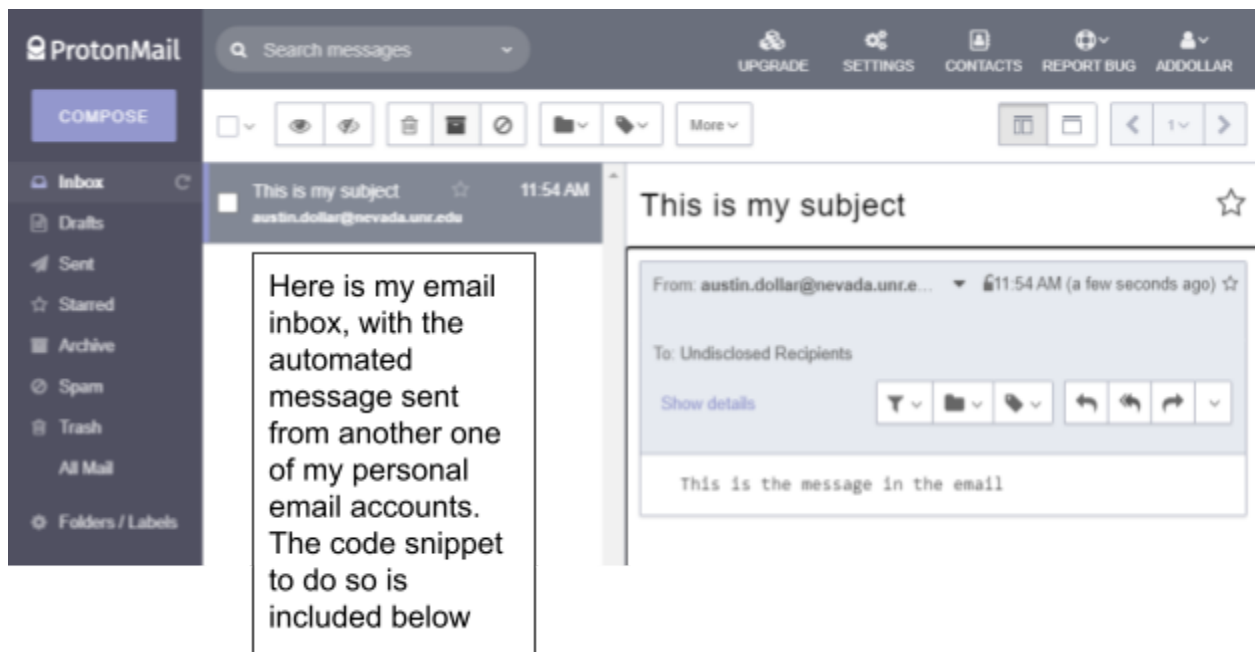
```
dummyData.py > ...  
1  f= open("dummy.txt","w+")  
2  
3  for i in range(10):  
4      f.write("This is line %d\r\n" % (i+1))  
5  
6  f.close() |
```

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

```
dummyData.py dummy.txt net10.sh script on system.  
/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



```
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 "olddata" 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' , 'olddata.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('HHHHHHHHHHHHHHHHHHHH FOUND HHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHH')
                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"))

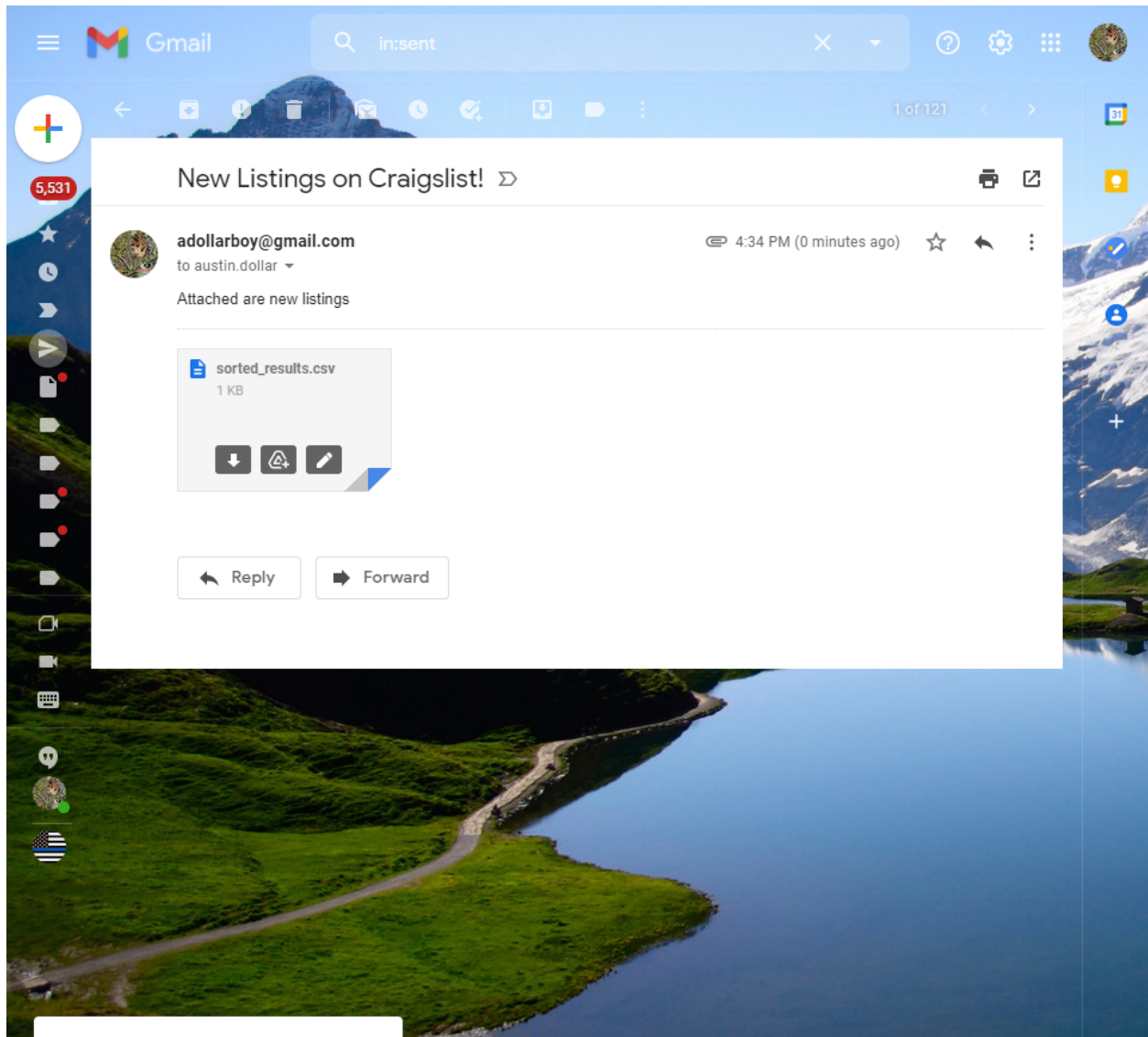
    file_to_attach = "sorted_results.csv"

    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

```
addollar@ip-172-31-43-34: ~/lab9_10
# Edit this file to introduce tasks to be run by cron.
#
# Each task to run has to be defined through a single line
# indicating with different fields when the task will be run
# and what command to run for the task
#
# To define the time you can provide concrete values for
# minute (m), hour (h), day of month (dom), month (mon),
# and day of week (dow) or use '*' in these fields (for 'any').#
# Notice that tasks will be started based on the cron's system
# daemon's notion of time and timezones.
#
# Output of the crontab jobs (including errors) is sent through
# email to the user the crontab file belongs to (unless redirected).
#
# For example, you can run a backup of all your user accounts
# at 5 a.m every week with:
# 0 5 * * 1 tar -zcf /var/backups/home.tgz /home/
#
# For more information see the manual pages of crontab(5) and cron(8)
#
# m h dom mon dow   command
0 * * * * /home/addollar/lab9_10/scrapper.py
~
~
~
~
~
~
~
~
~
~
~
```

Here is the updated crontab which executes the webscraper rather than the arbitrary script from earlier.

Here is the updated crontab which executes the webscraper rather than the arbitrary script from earlier.

**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.