Project 1: Email, Web Scraping with Python

Due Oct. 11th, 2019

Hand in: Electronic submission of entire project (source files, report, etc.) using 'Assignment' on Canvas. Please zip up your whole project directory and submit the zip file. In your submission, please write down a comprehensive **report**, which explains your code and demos the functions of the project with screen **snapshots**. Your project must be implemented by **Python**. Do **NOT** write down your email or password in your report or your Python code!!!!!

First, you will play with mail server. We will do this part step-by-step.

- First, we want to play the mail server of TCNJ. To play with the mail servers, you need to find the mail server. Open your terminal, type

  ```
  nslookup –type=mx tcnj.edu
  ```

  In the terminal, there will be two mail servers displayed in terminal. Pick one. Without loss of generality, we assume the picked server is xxxx.tcnj.edu. Type the following command in terminal.

  ```
  telnet xxxx.tcnj.edu 25
  ```

  Then, type the commands you see on page 60 of the chapter 2 slides. Replace the sender and the receiver email address to be your own email address. Check your mailbox, see whether you get the email or not.

  Take a snapshot of your terminal when you type the commands.

  Hint: It will NOT be surprising to see you cannot send out email to the mail server on campus. These email servers reverse your IP address to a registered hostname (PTR) on DNS server. If the reversed hostname does not match your claim hostname, your request will be rejected. The sad fact is our own computers are definitely not registered in PTR record (some TCNJ computers are registered).

- Since TCNJ outsourced the email service to Gmail a few years ago, let's try the mail server of Gmail. First, we need to figure out which gmail server we want to talk with. If we type the following command in terminal

  ```
  nslookup –type=mx gmail.com
  ```

  we will see the following results:

```
Non-authoritative answer:
gmail.com     mail exchanger = 20 alt2.gmail-smtp-in.l.google.com.
gmail.com     mail exchanger = 30 alt3.gmail-smtp-in.l.google.com.
gmail.com     mail exchanger = 5 gmail-smtp-in.l.google.com.
gmail.com     mail exchanger = 40 alt4.gmail-smtp-in.l.google.com.
gmail.com     mail exchanger = 10 alt1.gmail-smtp-in.l.google.com.

Authoritative answers can be found from:
alt3.gmail-smtp-in.l.google.com    internet address = 74.125.140.27
alt3.gmail-smtp-in.l.google.com    has AAAA address
2a00:1450:400c:c08::1a
gmail-smtp-in.l.google.com    internet address = 173.194.66.26
gmail-smtp-in.l.google.com    has AAAA address 2607:f8b0:400d:c01::1a
alt4.gmail-smtp-in.l.google.com    internet address = 74.125.143.26
alt4.gmail-smtp-in.l.google.com    has AAAA address
2a00:1450:4013:c03::1b
alt1.gmail-smtp-in.l.google.com    internet address = 64.233.190.26
alt1.gmail-smtp-in.l.google.com    has AAAA address 2800:3f0:4003:c01::1a
alt2.gmail-smtp-in.l.google.com    internet address = 209.85.203.26
alt2.gmail-smtp-in.l.google.com    has AAAA address
2a00:1450:400b:c03::1a
```

It looks there are a bunch of mail servers in Gmail. It is a bit confusing, if we visit the gmail webpage at https://support.google.com/mail/answer/7126229?hl=en , which says the IMAP server (incoming emails) is imap.gmail.com and the SMTP (outgoing emails) server is smtp.gmail.com. As you can see imap.gmail.com and smtp.gmail.com are alias names and the names returned by nslookup are canonical names. In this project, we just use alias name. More specifically, to simplify the project and discussion, this project use smtp.gmail.com. Once you understand SMTP, you can easily figure out IMAP details.

According to the webpage at https://support.google.com/mail/answer/7126229?hl=en, smtp.gmail.com has two port numbers, 465 and 587, for SMTP service. Remember that we used port number 25 in class? Why is Gmail using 465 and 587, instead of 25? The short answer is that port 25 is for plaintext SMTP, while 465 and 587 are for encrypted SMTP transfer. As we show in the lab, we can easily use a network sniffer like Wireshark to capture the plaintext transfer. So today, it is common to use 465 and 587 for security reasons.

What is the difference between 465 and 587 for Gmail? There are two popular protocols to do network encryption/decryption. One is SSL (including version 1.0, 2.0, 3.0), the other one is TLS (including version 1.0, 1.1, 1.2, and 1.3). SSL is an older protocol, it is gradually phasing out. TLS is a more recent/advanced one. If it is possible, TLS is preferred over SSL. A detailed discussion of TLS will be provided in security course. For this project, you do not have to worry about the details of TLS itself. By default, we should always use 587 (using TLS) to submit email to server. Port 465 is obsolete.

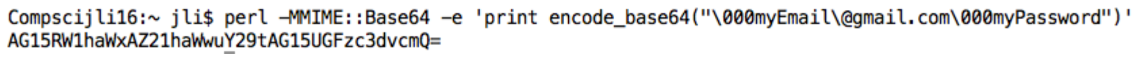
Can we use telnet to connect to the port 587 smtp.gmail.com? No! telnet itself is plaintext protocol, it does not support SSL or TLS. To use SSL or TLS, you should use software like openssl.

Do the following steps in your terminal, let's try to connect to smtp.gmail.com **manually**.

1. Assume your gmail address is myEmail@gmail.com, and your password is myPassword. Type the following command in the terminal. In your own command, myEmail@gmail.com shall be replaced by your gamil account and myPassword should be replaced by your own password.

   ```
   perl -MMIME::Base64 -e 'print
   encode_base64("\000myEmail\@gmail.com\000myPassword")'
   ```

   Please note the above command is supposed to be one line, just like the following snapshot.

   ```
   Compscijli16:~ jli$ perl -MMIME::Base64 -e 'print encode_base64("\000myEmail\@gmail.com\000myPassword")'
   AG15RW1haWxAZ21haWwuY29tAG15UGFzc3dvcmQ=
   ```

   In above, AG15RW1haWxAZ21haWwuY29tAG15UGFzc3dvcmQ= is the output of the command. We will use it as credential to login Gmail (your output is different from the example).

2. Type following command to connect to smtp.gmail.com

   openssl s_client -starttls smtp -connect smtp.gmail.com:587 -crlf -ign_eof

3. then type:

   EHLO localhost

4. then type:

   AUTH PLAIN AG15RW1haWxAZ21haWwuY29tAG15UGFzc3dvcmQ=

5. try to type:

   MAIL FROM: <myEmail@gmail.com>

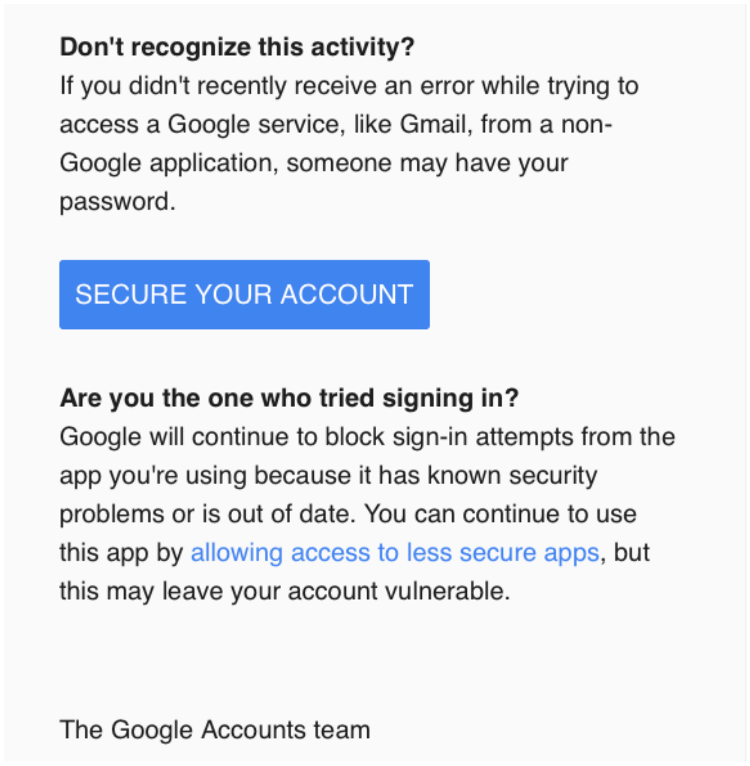
   RCPT TO: <myEmail@gmail.com>

   DATA

   Subject: Hello gmail!

```
It works!

.

quit
```

Does the commands work? If not, open your gmail account in a browser. Do you get a warning email about your login? Does the warning email looks similar to this snapshot?

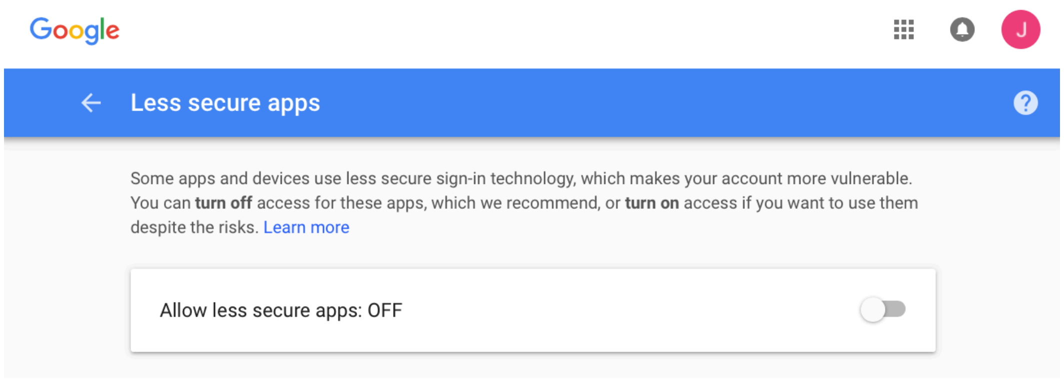


Don't recognize this activity?
If you didn't recently receive an error while trying to access a Google service, like Gmail, from a non-Google application, someone may have your password.

Are you the one who tried signing in?
Google will continue to block sign-in attempts from the app you're using because it has known security problems or is out of date. You can continue to use this app by allowing access to less secure apps, but this may leave your account vulnerable.

The Google Accounts team

If you do receive this email from Gmail, click "allowing access to less secure apps" in the email, **turn on** the switch on the following snapshot.



Some apps and devices use less secure sign-in technology, which makes your account more vulnerable. You can **turn off** access for these apps, which we recommend, or **turn on** access if you want to use them despite the risks. Learn more

Allow less secure apps: OFF

After turning on the switch, redo the steps from 1 to 5. Check Gmail in browser again. Do you receive the email from yourself? Take a snapshot of your terminal, **sanitize/blur** the credential part. Take a snapshot of the email in browser.

After all this is done, **turn OFF** the “Allow less secure app” switch.

PART ONE

- OK, you have played with SMTP and got a feeling how it works. Before writing the Python code, let’s imagine that we are interested in stock market. If there is a significant market movement, you want to be notified by the email and cell phone message (SMS text). Suppose that we are particularly interested in the volatility of SP500, Let’s write a Python code doing the following things:

    1. Check the latest VIX value online, you can check it from Yahoo finance at https://finance.yahoo.com/quote/^VIX?p=^VIX , or from Google Finance at https://finance.google.com/finance?q=INDEXCBOE:VIX .

    2. Send an email to your own email address about the current VIX value.

    3. Send a SMS text to your own cell phone about the current VIX value.

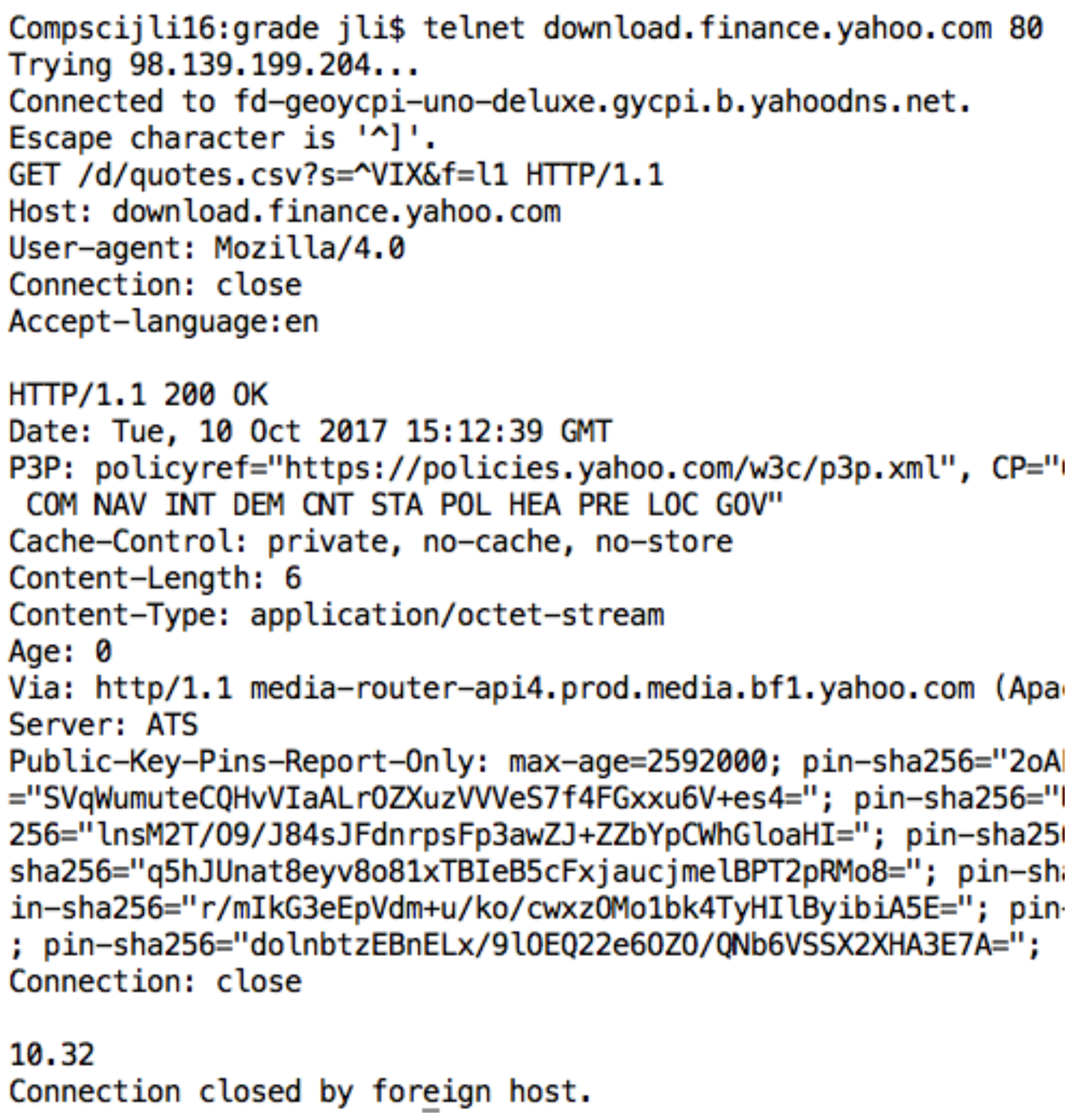
  Since this course is networking course, we want to learn the stuff inside the network. You are PROHITBITED to use Yahoo Finance API or Google Finance API. Instead, we want to process the raw data exchanged between our Python code and Yahoo Finance (as example).

  To get ^VIX (here VIX is an index value, not real stock value. So it is preceded with ^. The real stock symbol, such as AAPL or GOOGL, are not preceded with ^) data from Yahoo. We can type the following command in terminal:

  ```
  curl -s 'http://download.finance.yahoo.com/d/quotes.csv?s=^vix&f=l1'
  ```

  It will display a recent ^VIX value, such as 10.33, which is the value we want. If you replace ^VIX in the above command with any other stock symbol, you will get the price quote of that stock symbol.

  However, our Python code shall NOT call curl. The code will send GET request to the yahoo server. The following snapshot shows how command line works in terminal. Your Python code should work in a similar way.



```
Compscijli16:grade jli$ telnet download.finance.yahoo.com 80
Trying 98.139.199.204...
Connected to fd-geoycpi-uno-deluxe.gycpi.b.yahoodns.net.
Escape character is '^]'.
GET /d/quotes.csv?s=^VIX&f=l1 HTTP/1.1
Host: download.finance.yahoo.com
User-agent: Mozilla/4.0
Connection: close
Accept-language:en

HTTP/1.1 200 OK
Date: Tue, 10 Oct 2017 15:12:39 GMT
P3P: policyref="https://policies.yahoo.com/w3c/p3p.xml", CP="
 COM NAV INT DEM CNT STA POL HEA PRE LOC GOV"
Cache-Control: private, no-cache, no-store
Content-Length: 6
Content-Type: application/octet-stream
Age: 0
Via: http/1.1 media-router-api4.prod.media.bf1.yahoo.com (Apa
Server: ATS
Public-Key-Pins-Report-Only: max-age=2592000; pin-sha256="2oA
="SVqWumuteCQHvVIaALrOZXuzVVVeS7f4FGxxu6V+es4="; pin-sha256="
256="lnsM2T/O9/J84sJFdnrpsFp3awZJ+ZZbYpCWhGloaHI="; pin-sha25
sha256="q5hJUnat8eyv8o81xTBIeB5cFxjaucjmelBPT2pRMo8="; pin-sh
in-sha256="r/mIkG3eEpVdm+u/ko/cwxzOMo1bk4TyHIlByibiA5E="; pin
; pin-sha256="dolnbtzEBnELx/9lOEQ22e6OZO/QNb6VSSX2XHA3E7A=";
Connection: close

10.32
Connection closed by foreign host.
```

As you can see from above, 10.32 is the returned ^VIX value. This is the value that we need.

In the above snapshot, the GET header of our request is:

```
GET /d/quotes.csv?s=^VIX&f=l1 HTTP/1.1
Host: download.finance.yahoo.com
User-agent: Mozilla/4.0
Connection: close
Accept-language:en
```

In your Python code, you should send the same request header to the Yahoo download server. To get a feeling how a Python client and a Python server work together, I list a sample client and a sample server as follows. Please copy & paste them into two different Python files. Run them, test them, then twist them.

Client code:

```
import socket
import sys

# Create a TCP/IP socket
sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)

# Connect the socket to the port on the server given by the
caller
server_address = (sys.argv[1], 10000)
print >>sys.stderr, 'connecting to %s port %s' % server_address
sock.connect(server_address)

try:

    message = 'This is the message.  It will be repeated.'
    print >>sys.stderr, 'sending "%s"' % message
    sock.sendall(message)

    amount_received = 0
    amount_expected = len(message)
    while amount_received < amount_expected:
        data = sock.recv(16)
        amount_received += len(data)
        print >>sys.stderr, 'received "%s"' % data

finally:
    sock.close()
```

Server code:

```
import socket
import sys

# Create a TCP/IP socket
sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)

# Bind the socket to the address given on the command line
server_name = sys.argv[1]
server_address = (server_name, 10000)
print >>sys.stderr, 'starting up on %s port %s' % server_address
sock.bind(server_address)
sock.listen(1)

while True:
    print >>sys.stderr, 'waiting for a connection'
    connection, client_address = sock.accept()
    try:
        print >>sys.stderr, 'client connected:', client_address
        while True:
            data = connection.recv(16)
            print >>sys.stderr, 'received "%s"' % data
            if data:
                connection.sendall(data)
            else:
                break
    finally:

        connection.close()
```

Please play with these client and server codes, modify the client code, make the client to communicate with Yahoo download site. If everything works smoothly, your code should be able to download HTML page containing the ^VIX value.

Once your code receives the response from Yahoo, the code need to extract the ^VIX value from the response. Of course, you need to know the basic of HTML at https://www.w3schools.com/html/ .

To send an email with the ^VIX value inside, you will send to an email address from your own Gmail account (not TCNJ account even it is supported by Gmail). To comply with Gmail security requirements, we will use Gmail API to send out the email. Go to the Gmail website https://developers.google.com/gmail/api/quickstart/python , follow the instruction, setup configurations for Gmail API. Modify the sample code on the webpage, integrate your existing codes together. Add ^VIX value to the body of the email and send it out to your email address. Take a snapshot of the email in your Gmail account.

Sending email is not sufficient. Sometimes we need additional reminders. In this project, you are required to send a SMS text message to your own cell phone. In fact, it is quite easy to do it with your Python code. Please visit the website at https://www.fullstackpython.com/blog/send-sms-text-messages-python.html, follow the instruction, add the sample code to your existing code. Take a snapshot of your SMS text received on the cell phone.

PART TWO

WARNING:

1. Do not abuse the web server, your script should pace the requests generously. One+ second(s) between requests is mandatory. Because a large number of requests can be generated by your Python scripts by mistake, your Python script is **prohibited to put any click() or get() function inside ANY loop**. A violated submission will get grade 0.

2. Do NOT share your username and password with anyone, including your partner. Whenever you need to share the code with your partner, remove

your username and password from the script!!!! If you and your team-mate have to sit side-by-side and debug the code, you can change your password to a temporary one, change it back later.

The second part of this project is to search courses on PAWS. We want to simplify the searching process in PAWS. The goal is to run a Python script. This script opens the browser and automatically clicks links and fills forms, collects the results from PAWS, and presents the consolidated data. Ideally, the whole process takes seconds in a normal networking condition.

Selenium (https://selenium-python.readthedocs.io) is a powerful and user-friendly tool to automate web browsing. In other words, you can write a Python script that automatically browse certain website. Automated web browsing is widely used by industry to test website, collect data from website. Many companies developed their own web browsing software. There are quite a lot of tools supporting web browsing (https://github.com/dhamaniasad/HeadlessBrowsers), Selenium is one of them. In the second part of this project, we use Selenium to search course on PAWS.

To finish this part of the project:

1. Install selenium on your computer.

2. Read the sample code "paws.py", which use multithread. This script can create different threads for different accounts. In this project, we use only one account username/password and create one thread.

3. While working with part of the project, you will use xpath to search certain contents in the HTML files. To get an idea of xpath, please look at the introduction at https://www.guru99.com/xpath-selenium.html .

4. To process the results returned from web server, beautifulsoup (https://pypi.org/project/beautifulsoup4/) can be extremely helpful. In this project, when you consolidate the data and prepare the result of the project, beautifulsoup can significantly reduce the workload. To get a quick start with beautifulsoup, take a look at the following tutorial. (https://www.dataquest.io/blog/web-scraping-tutorial-python/). To practice beautifulsoup, you can run Python in a terminal, and copy & paste the following lines into Python.

```
>>> from bs4 import BeautifulSoup

>>> html = """

... <html>

... <body>
...     <table>
...       <table id="table1">
...         <th><td>column 1</td><td>column 2</td></th>
...         <tr><td>value 1</td><td>value 2</td></tr>
...       </table>
...       <table id="table2">
...        <th><td>column 1</td><td>column 3</td></th>
...        <th><td>column 1</td><td>column 4</td></th>
...       </table>
...     </table>
... </body>
... </html>
... """
>>> soup = BeautifulSoup(html)
>>> tables = soup.findChildren('table', id="table2")
>>> my_table = tables[0]
>>> rows = my_table.findChildren(['th', 'tr'])
>>> for row in rows:
...     cells = row.findChildren('td')
...     for cell in cells:
...         value = cell.string
...         print("The value in this cell is %s" % value)
```

5. Modify the script so that it one particular course such as CSC230 in the current semester. Your script should print out the Open Sections, then Closed Sections. For example, to search CSC220. You type the following command in terminal:

   ```
   python3 paws.py csc 220
   ```

   The returned result is:

   ```
   Opening...

   send username

   jli login in process...

   jli login succeed

   ===============Open Sections===================

   83618



   Mo 12:30PM - 1:50PM

   Th 2:00PM - 3:20PM

   Th 12:30PM - 1:50PM



   Sharif Mohammad Shahnewaz Ferdous

   Sharif Mohammad Shahnewaz Ferdous

   Sharif Mohammad Shahnewaz Ferdous



   -------------------------

   83624



   TuTh 5:30PM - 7:30PM
   ```

Mark Russo

83637

Fr 11:00AM - 12:20PM

Fr 2:00PM - 3:20PM

Tu 2:00PM - 3:20PM

Glenn Rhoads

Glenn Rhoads

Glenn Rhoads

================Closed Sections===================

83613

Tu 11:00AM - 12:20PM

Tu 2:00PM - 3:20PM

Fr 11:00AM - 12:20PM

Michael Bloodgood

Michael Bloodgood

Michael Bloodgood

83611

TuFr 3:30PM - 4:50PM

Fr 2:00PM - 3:20PM

Michael Bloodgood

Michael Bloodgood

83623

Mo 2:00PM - 3:20PM

Mo 3:30PM - 4:50PM

Th 3:30PM - 4:50PM

Sharif Mohammad Shahnewaz Ferdous

Sharif Mohammad Shahnewaz Ferdous

Sharif Mohammad Shahnewaz Ferdous

```
--------------------------
```

If you search CSC230. You can type:

```
python3 paws.py CSC 230
```

The result is:

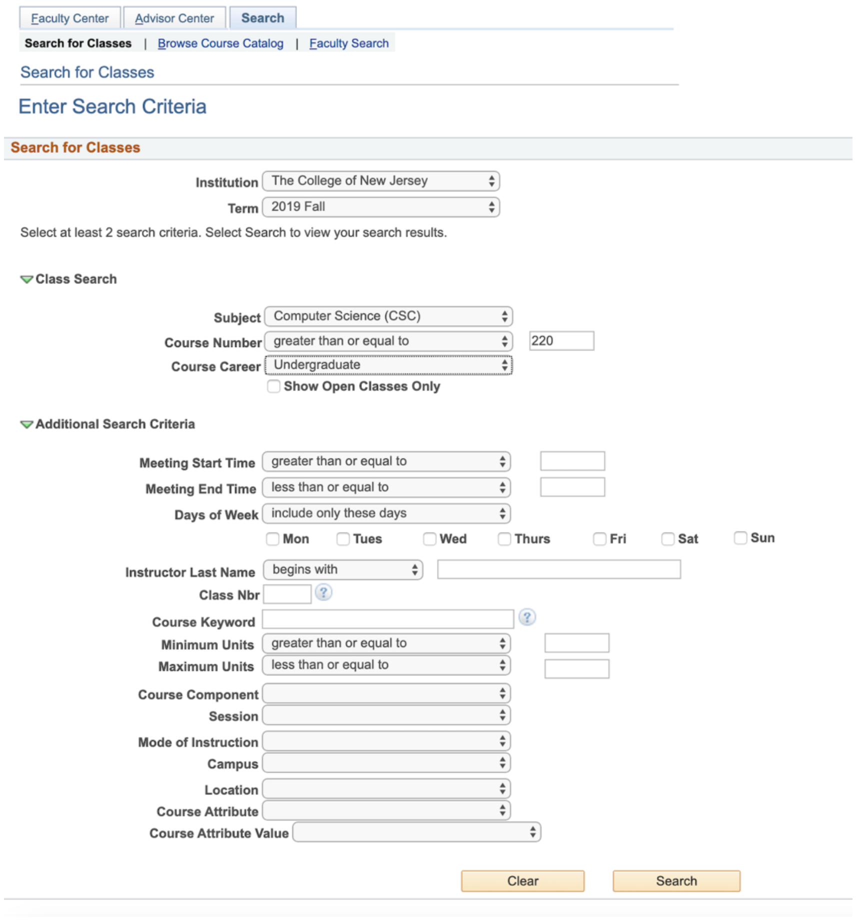
```
Opening...

find username field

send username

jli login in process...

jli login succeed

1

================Open Sections===================

83640


TuFr 9:30AM - 10:50AM

Tu 11:00AM - 12:20PM


Glenn Rhoads

Glenn Rhoads


--------------------------

================Closed Sections===================

83606
```

```
TuFr 11:00AM - 12:20PM

Tu 2:00PM - 3:20PM



John DeGood

John DeGood



--------------------------
```

If you search BIO330, the command and result looks like the following printout.

```
python3 paws.py BIO 330

Opening...

send username

jli login in process...

jli login succeed

======= No Match =========
```

By default, the script searches the Fall semester of 2019. When you write the searching part of the script, use the following picture as reference.



Search for Classes

Enter Search Criteria

Search for Classes

Institution	The College of New Jersey
Term	2019 Fall

Select at least 2 search criteria. Select Search to view your search results.

Class Search

Subject	Computer Science (CSC)	
Course Number	greater than or equal to	220
Course Career	Undergraduate	

☐ Show Open Classes Only

Additional Search Criteria

Meeting Start Time	greater than or equal to	
Meeting End Time	less than or equal to	
Days of Week	include only these days	

☐ Mon ☐ Tues ☐ Wed ☐ Thurs ☐ Fri ☐ Sat ☐ Sun

Instructor Last Name	begins with	
Class Nbr		
Course Keyword		
Minimum Units	greater than or equal to	
Maximum Units	less than or equal to	
Course Component		
Session		
Mode of Instruction		
Campus		
Location		
Course Attribute		
Course Attribute Value		