Name: _____

Project 1 - Sockets and HTTP

Introduction

Welcome to your first coding project! In each of these we will ask you to write short programs that will enhance your understanding of networking fundamentals and possibly assist you in your future work.

For these assignments, we will be asking for both your source code and a short writeup that includes directions on how to run your code and screenshots of your code running. We may try to run your code on our own machines, and if it doesn't work there will be a point reduction, or we will ask you to fix it, or both.

The "What to turn in" section at the bottom of this document includes a list of items to submit. Make sure you have something for each of those. Do not include any part of this document unless specifically asked to. Submissions will be in the form of a doc, docx, or pdf format, and also your code listings.

Every project has a comments section where you can talk about the challenges you may have faced and how you overcame them. Below you will see how I researched and found info on the choice for IP address in a loopback mode. You should include links to sources like these your comments section.

Be aware that code out on the web can be full of errors. Just because it runs does not mean that it's good code. The best advice I can give to you future software engineers, is to make sure you fully understand what every line of your code is doing and why.

A quick word on academic integrity: We expect as coders that you get inspiration from the web. After all, that's how most of us do our jobs. We do not expect that you will copy and paste code verbatim from the web or from your fellow classmates and turn it in with your name on it. You wouldn't do that in a job, and you shouldn't do that here. Cheaters never prosper, and often get caught. Again, you want to make sure you fully understand what every line of your code is doing and why. And the best way to do that, is to write every line of it yourself.

Using a socket to GET a file

How do internet browsers work? They all manage networking "sockets" to contact a web server and download HTML files and resources. In this part you will make a bare-bones socket program to do some of what your internet browser does.

Create a simple python program that uses a socket to interact with a server. Note that your program MUST USE THE PYTHON SOCKET API. Yes, it is possible to do this in

one line of code with Python Requests or some other library. But that wouldn't be any fun, would it?

Your program shall make a socket connection to the host: "gaia.cs.umass.edu" and send the GET request for the URI: "/wireshark-labs/INTRO-wireshark-file1.html". To do this, you will send the following HTTP-compliant GET request to the server *exactly* as shown:

"GET /wireshark-labs/INTRO-wireshark-file1.html HTTP/1.1\r\nHost:gaia.cs.umass.edu\r\n\r\n"

Note: If you are interested in the specification of an HTTP request, see:
https://www.w3.org/Protocols/rfc2616/rfc2616
https://www.w3.org/Protocols/rfc2616/rfc2616-sec5.html#sec5.1.2

Run your program and receive a response from the server. It should look like this:

```
Williams-MacBook-Pro:Project0 williampfeil$ python connect_socket.py
Request: GET /wireshark-labs/INTRO-wireshark-file1.html HTTP/1.1
Host:gaia.cs.umass.edu


[RECV] - length: 382
HTTP/1.1 200 OK
Date: Thu, 13 Feb 2020 21:56:16 GMT
Server: Apache/2.4.6 (CentOS) OpenSSL/1.0.2k-fips PHP/5.4.16 mod_perl/2.0.11 Perl/v5.16.3
Last-Modified: Thu, 13 Feb 2020 06:59:01 GMT
ETag: "51-59e6f9fb89784"
Accept-Ranges: bytes
Content-Length: 81
Content-Type: text/html; charset=UTF-8

<html>
Congratulations!  You've downloaded the first Wireshark lab file!
</html>
```

GET the data for a large file

Your first program was probably written with a single read or recv command. This works fine for very small files, but would not be able to download anything larger. For this next part, you will write a socket program to receive arbitrarily large files.

Your program will make a socket connection to the host: "gaia.cs.umass.edu" and send the GET request for the URI: "/wireshark-labs/HTTP-wireshark-file3.html". To do this, you will send the following HTTP-compliant GET request to the server *exactly* as below:

"GET /wireshark-labs/HTTP-wireshark-file3.html HTTP/1.1\r\nHost:gaia.cs.umass.edu\r\n\r\n"

There is no end-of-transmission (EOT) with sockets, so knowing when you've received all the data can be difficult. Fortunately for this project, the gaia.cs.umass.edu server will close the connection after sending its data, so the easy thing to do is detect when recv or read return <= 0 bytes in a loop.

Run your program and take screenshots. Take one screenshot of the first few lines, and another of the last few lines of the result like the images below.

```
Williams-MacBook-Pro:Projects williampfeil$ python lab2.py
Request: GET /wireshark-labs/HTTP-wireshark-file3.html HTTP/1.1
Host:gaia.cs.umass.edu

[RECV] - length: 4805
HTTP/1.1 200 OK
Date: Thu, 13 Feb 2020 23:44:10 GMT
Server: Apache/2.4.6 (CentOS) OpenSSL/1.0.2k-fips PHP/5.4.16 mod_perl/2.0.11 Perl/v5.16.3
Last-Modified: Thu, 13 Feb 2020 06:59:01 GMT
ETag: "1194-59e6f9fb85903"
Accept-Ranges: bytes
Content-Length: 4500
Content-Type: text/html; charset=UTF-8

<html><head>
<title>Historical Documents:THE BILL OF RIGHTS</title></head>

<body bgcolor="#ffffff" link="#330000" vlink="#666633">
<p><br>
</p>
<p></p><center><b>THE BILL OF RIGHTS</b><br>
  <em>Amendments 1-10 of the Constitution</em>
</center>

<p>The Conventions of a number of the States having, at the time of adopting
the Constitution, expressed a desire, in order to prevent misconstruction
```

```
<p></p><p>Excessive bail shall not be required, nor excessive fines
imposed, nor cruel and unusual punishments inflicted.

</p><p><a name="9"><strong><h3>Amendment IX</h3></strong></a>

<p></p><p>The enumeration in the Constitution, of certain rights, shall
not be construed to deny or disparage others retained by the people.

</p><p><a name="10"><strong><h3>Amendment X</h3></strong></a>

<p></p>
<p>The powers not delegated to the United States by the Constitution, nor prohibited
  by it to the states, are reserved to the states respectively, or to the people.</p>
</body></html>
Williams-MacBook-Pro:Projects williampfeil$ █
```

The world's simplest HTTP server

Now you're going to create an HTTP server using the python socket api. Your program will create a listening socket bound to '127.0.0.1' or 'localhost', and a random port number > 1023. You will then use your web browser to connect to your server and receive data. Note that your server could be running on any host within your LAN (as long as there's no firewall that can block access to it).

If you are interested in the choice of IP address, here is a deep-dive explanation. The short explanation, is that this is the local machine loopback address, so that you can have both client and server running locally and communicating.

When your socket accepts a request, a new socket is created. Read the socket request on the new socket and print it. Then send the following html data on the new socket and close it.

```
data =  "HTTP/1.1 200 OK\r\n"\
    "Content-Type: text/html; charset=UTF-8\r\n\r\n"\
    "<html>Congratulations!  You've downloaded the first Wireshark lab file!</html>\r\n"
```

Run your program (http_server.py). Start up your web browser and navigate to 127.0.0.1:xxxx (where xxxx is the port you specified in your server). Take screenshots of your server and your browser:


```
(venv_python3) Williams-MacBook-Pro:solution williampfeil$ python3 http_server.py
Connected by ('127.0.0.1', 64269)

Received: b'GET / HTTP/1.1\r\nHost: 127.0.0.1:8001\r\nConnection: keep-alive\r\nUpgrade-Insecure-Reque
sts: 1\r\nUser-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_14_6) AppleWebKit/537.36 (KHTML, like
Gecko) Chrome/79.0.3945.130 Safari/537.36\r\nSec-Fetch-User: ?1\r\nAccept: text/html,application/xhtml
+xml,application/xml;q=0.9,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.9\r\nS
ec-Fetch-Site: none\r\nSec-Fetch-Mode: navigate\r\nAccept-Encoding: gzip, deflate, br\r\nAccept-Langua
ge: en-US,en;q=0.9\r\n\r\n'

Sending>>>>>>>>
HTTP/1.1 200 OK
Content-Type: text/html; charset=UTF-8

<html>Congratulations!  You've downloaded the first Wireshark lab file!</html>

<<<<<<<<
(venv_python3) Williams-MacBook-Pro:solution williampfeil$
```

What to turn in

1. In the Word doc:
   a. **Include instructions on how to run your programs. Are they python3?**
   b. Include screenshots of your running code.
   c. Include comments / questions (optional)
2. In your code listings:
   a. Include sources you used (web pages, tutorials, books, etc)
   b. Comment your code