Project #1: Warmup: Socket programming 101

CS 352 Internet Technology

Released: September 11, 2024; Due: September 24, 2024

The goal of this project is to help you explore Python programming and specifically Python's socket programming interface. Further, this exercise will serve as a foundation for other upcoming programming projects that use sockets.

Starting with the sample working code in `proj.py`, you will construct two programs `client.py` and `server.py` which communicate with each other over the socket API. Currently, `proj.py` consists of server code and client code written as two separate threads.

How to work through this project

Step 1: Understand the functionality as is

Understand the functionality implemented in the `proj.py` program. First, download, save and execute the program as is in your `ilab` environment. Make sure it executes successfully and according to how you would expect.

For example, you might see something like this if you run the command `python3 proj.py` on your `ilab` machine terminal.

If you run the program on your MAC or windows machine with a hostname that is not resolvable through DNS, you will receive an error (see the notes later in this document). We recommend that you run this program on ilab.

```
[S]: Server socket created
[S]: Server host name is porthos.cs.rutgers.edu
[S]: Server IP address is 128.6.25.90
[C]: Client socket created
[S]: Got a connection request from a client at ('128.6.25.90', 59814)
[C]: Data received from server: Welcome to CS 352!
Done.
```

Step 2: Remove sleeps and re-run

Remove the two statements in the program that say `time.sleep(5)`. Now run the program once, and run it immediately again after it finishes successfully.

What do you see? Can you explain why?

Step 3: Separate `proj` into two programs

Separate the server code and client code into two different programs, `server.py` and `client.py`. Execute the server program first and then execute the client program. You should still get the same set of print messages as in the combined threaded code in `proj.py`.

Step 4: Reversing the message

In the program provided, the server just sends a message string to the client after it connects. Modify the server so that when the client sends a string to the server, the server reverses the string before sending it back to the client. Further, we would like you to swap the case (uppercase $\rightarrow$ lowercase and vice versa) of the string sent by the client. For example, if the client sends `HELLO` to the server, the client should receive `olleh`. Your client program should print the string sent by the client and the corresponding string received by the client from the server.

You may look at the methods of the Python string class `https://docs.python.org/3/library/stdtypes.html#string-methods` to determine which ones may be usable to complete this project.

Step 5: Reading strings from a file

Now make your client program read each line from an input file `in-proj.txt` and send it to the server. Your **server program** should print each reversed output to a file `out-proj.txt`. Your output filename must match exactly with the one shown. A sample input file is provided. You are welcome to modify this file and try with your own sample inputs. If it helps, you may assume each line will be at most 200 characters long. The input file may contain multiple lines. Each input line may contain any character that can be typed on a standard US qwerty keyboard, including space.

Some notes

(1) A reference output file is provided in `sample-out-proj.txt`.

(2) If you run `proj.py` on certain machines (not ilab), for example, on your laptop directly, you might see an error if name resolution fails. Here is an example when running `proj.py` on my Apple Mac OS Mojave machine:

```
   [S]: Server socket created
   [S]: Server host name is APPLEs-MacBook-Pro.local
   [C]: Client socket created
   Exception in thread server:
   Traceback (most recent call last):
     File "/System/Library/Frameworks/Python.framework/Versions/2.7/lib/python2.7/threading
line 810, in __bootstrap_inner
       self.run()
     File "/System/Library/Frameworks/Python.framework/Versions/2.7/lib/python2.7/threading
line 763, in run
```

```
      self.__target(*self.__args, **self.__kwargs)
    File "proj.py", line 20, in server
      localhost_ip = (socket.gethostbyname(host))
  gaierror:  [Errno 8] nodename nor servname provided, or not known
  Done.
  Exception in thread client:
  Traceback (most recent call last):
    File "/System/Library/Frameworks/Python.framework/Versions/2.7/lib/python2.7/threading
line 810, in __bootstrap_inner
      self.run()
    File "/System/Library/Frameworks/Python.framework/Versions/2.7/lib/python2.7/threading
line 763, in run
      self.__target(*self.__args, **self.__kwargs)
    File "proj.py", line 43, in client
      localhost_addr = socket.gethostbyname(socket.gethostname())
  gaierror:  [Errno 8] nodename nor servname provided, or not known
```

The program failed with the above error because the local host name could not be resolved through the `gethostbyname` call.

(3) If you're connected on a VPN when you're running `proj.py` on your local machine (not ilab), the program may not work as expected because the client may be unable to connect to the server using the machine's VPN IP address. Here is an example of this error on my laptop:

```
  [S]: Server socket created
  [S]: Server host name is ipo39121132.rad.rutgers.edu
  [S]: Server IP address is 10.66.6.157
  [C]: Client socket created
  Done.
  Exception in thread client:
  Traceback (most recent call last):
    File "/System/Library/Frameworks/Python.framework/Versions/2.7/lib/python2.7/threading
line 810, in __bootstrap_inner
      self.run()
    File "/System/Library/Frameworks/Python.framework/Versions/2.7/lib/python2.7/threading
line 763, in run
      self.__target(*self.__args, **self.__kwargs)
    File "proj.py", line 47, in client
      cs.connect(server_binding)
    File "/System/Library/Frameworks/Python.framework/Versions/2.7/lib/python2.7/socket.py
line 228, in meth
      return getattr(self._sock,name)(*args)
  error:  [Errno 60] Operation timed out
```

(4) How we will run your code: We will run your client and server programs with the commands `python3 server.py` and `python3 client.py`, using Python version 3 on the ilab machines.

What to submit

You must submit three files: your `client.py`, `server.py`, and `report.pdf`. The report must be in PDF file format. Please compress the files into a single Zip archive before uploading to Canvas. Only one team member must submit.

The client and server must implement all functionality up to step 5.

Your report must answer the following questions.

1. Team details: Clearly state the names and netids of your team members (there are 2 of you).
2. Collaboration: Who did you collaborate with on this project? What resources and references did you consult? Please also specify on what aspect of the project you collaborated or consulted.
3. What did you observe after running step 2 above? Can you explain why you see what you see?
4. Is there any portion of your code that does not work as required in the description above? Please explain.
5. Did you encounter any difficulties? If so, explain.
6. What did you learn from working on this project? Add any interesting observations not otherwise covered in the questions above. Be specific and technical in your response.

Contact the course staff on Piazza if you have any questions.