Guide To Python Sockets

Why Sockets Matter

Similarly to the nc command, sockets in python will allow us to establish an end-point communication between 2 devices. We can use this to either send or receive information from those endpoints. By establishing this connection using python we have a wide array of tools and libraries that we can use to customize the information we send to a server and to format the information we receive. Moreover, we can automate what would otherwise be a very tedious process. We will see some examples of this in the later problems of the workshop. Keep in mind some of what we do today will be similar to the CyberLab problems we worked on yesterday, just in python form.

Code that will exist in all the sockets

1. First you must import the 'sys' library. This will allow us to process command line arguments and will be used to get our HOST and PORT

Code: `import sys`

2. Then you must import the 'socket' library. This will allow us to use the functions from this library

Code: `import socket`

3. Now that you have imported the socket library, you can grab the HOST from the command line using sys.argv[1]. **Keep in mind all 'argv' elements are strings. This is the python equivalent of when we specify the host in nc**

Code: `HOST = sys.argv[1]`

4. Now grab the PORT from the command line using sys.argv[2]. **Since 'argv' elements are strings and we need our port number to be an int, we must cast it as an int. Also, port will be the second argument on the command line, this is why we are doing sys.argv[2]. This is the python equivalent of when we specify the port in nc**

Code: `PORT = int(sys.argv[2])`

5. Since we have the HOST and the PORT setup, we can now create a socket that can communicate over TCP/IP. **Do no try to memorize this, it is okay to copy and re-use this line for all of the problems. All this line is saying is that we are trying to connect to a service running on an IP4 address on TCP/IP**

Note: socket.AF_INET = IP4 address

Note: socket.SOCK_STREAM = TCP protocol

***(Moderately)* Fun Fact:** AF stands for Address Family and PF stands for Protocol Family

Code: with socket.socket(socket.AF_INET, socket.SOCK_STREAM) as s:

6. We can now connect using our HOST and PORT. Please keep in mind, this must be indented under the previous line and we use 's' because that is the alias we chose in the previous line (see the "as s" in the code above). **We use double parentheses here on purpose, when we have two parentheses we are sending a tuple.******For indentation help, please see the full code at Figure 1****

Code: s.connect((HOST, PORT))

7. Now that we have connected to the host & port, we can perform multiple actions. The first action we will look at is receiving data from the host. This is very similar to what you all experienced with the 'Independence' lab from the previous day. The code below will receive the first 1024 bytes of data and will set it to our 'data' variable. **Please note: this is not currently a string and will need to be decoded/translated into a string in the next step**

Code: data = s.recv(1024)

8. As stated previously, 'data' needs to be converted to string format. We will do this by using the '.decode()' function. This functions comes from our imported socket library.

Code: decoded_data = data.decode()

Figure 1: Steps 1- 8 Combined (without main)

```
#!/usr/bin/env python3
```

```
# Import 'sys' module to process command line arguments
import sys
# Import 'socket' module to make network connections
import socket

# Variables for connection info
HOST = sys.argv[1]
# Convert the second argument to a number. All 'argv' elements #
are strings
PORT = int(sys.argv[2])

# Create a socket
with socket.socket(socket.AF_INET, socket.SOCK_STREAM) as s:
    # Connect to the host & port provided on the command line
    s.connect((HOST, PORT))
    # Receive the first 1024 bytes of data
    data = s.recv(1024)
    # Decode the byte string to get the data as a string
    decoded_data = data.decode()
```

Figure 2: Steps 1- 8 Combined (with the main method, use this for all the questions after ‘White Socks, with Boots')

```
#!/usr/bin/env python3

# Import 'sys' module to process command line arguments
import sys
# Import 'socket' module to make network connections
import socket


def main():
    # Variables for connection info
    HOST = sys.argv[1]
    # Convert the second argumment to a number. All 'argv'
elements are strings
```

```
    PORT = int(sys.argv[2])

    # Create a socket
    with socket.socket(socket.AF_INET, socket.SOCK_STREAM) as s:
        # Connect to the host and port provided on the command
line
        s.connect((HOST, PORT))
        # Variable to store the flag
        flag_not_found = True
        # Receive the initial prompt
        prompt = s.recv(1024)
```

Code that is problem specific

These are not listed in any particular order, it is up to you to figure out how to piece them together to get the answers.

Some problems will require you to send information in order to activate a specific condition

Code: s.send({YOUR INPUT})

Many problems will require you to strip the output you receive from the host. If the strip function is left with no arguments, it will drop the leading and trailing whitespace on the string

Code: {YOUR STRING}.strip()

E.g. (assuming decoded_data is a string)**:** decoded_data.strip()

The Python string method **startswith()** checks whether string starts with *str*.

Code: {YOUR STRING}.startswith()

Examples:

```
text = "Python is easy to learn."

result = text.startswith('is easy')
# returns False
print(result)

result = text.startswith('Python is ')
# returns True
print(result)

result = text.startswith('Python is easy to learn.')
# returns True
print(result)
```

Output

```
False
True
True
```

Hint for ‘White Socks, With Boots’

This problem will be very similar to the code in Figure 1 just make sure we format the output to match what the problem is asking for.

Pseudo Code for ‘Pure Luck’

General Overview: This problem is asking you to setup a python socket that will establish a connection to the host and will pass an incrementing count to the host until the correct number is hit. As always, steps 1 - 8 are necessary to setup our connection. Moreover, with this information in mind, we know we will need a way to **send information** to the host as well. Overall, for this problem we will need to establish a connection, receive data, send a number, check if the number is correct for the flag, increment the number if it is not correct, and this will continue until the correct number is sent and a flag is returned. Since we will keep checking until a specific condition is met, we are going to use a while loop for this problem

1. Setup the socket (lines 1 - 8 above) in the main method (this code should be re-used from Figure 2 and the imports should be above the main function)
2. Strip and decode the initial information received and set that value in a variable
3. Create a boolean that will have a specific starting value. This will be what we use to break the loop, i.e. once we guess the correct number this boolean should change
4. Create an empty counter variable that we will increment in our while loop
5. Create a while loop that uses the boolean we created earlier
6. **Send** the guess (number) to the server
7. Prompt the server for a response
8. If the response is the same prompt as before then you know you need to try another number, therefore, increment by one
9. Else the server has responded with the flag and you should set the boolean to change, hence breaking the loop

Pseudo Code for ‘Electric Maths’

General Overview: This problem is asking you to setup a python socket that will establish a connection to the host and will add one to the number that it sends back to you. As always, steps 1 - 8 are necessary to setup our connection. The first thing we will need to do is to receive the information from our connection to see what it would like us to do. Then we will need to access the last word in the response add one to it.

1. Setup the socket (lines 1 - 8 above) in the main method (this code should be re-used from Figure 2 and the imports should be above the main function)
2. Strip and decode the initial information received and set that value in a variable
3. Create a boolean that will have a starting value of False. This will be what we use to break the loop, i.e. once we get to the correct number
4. Create a while loop that uses the boolean we created earlier
5. Receive the prompt from the server
6. Access the last word in the prompt
7. Add "1" to that (may need to cast it specifically to do this)
8. Send this response back to the server
9. The rest is for you to figure out!

FAQ / Frequent Errors

Error: TypeError: socket.connect() takes exactly one argument (2 given)

Answer: Check your connect() line and make sure it has double parentheses around it. If you do not have those it will read what you are sending as two arguments instead of one tuple

Error: AttributeError: partially initialized module 'socket' has no attribute 'AF_INET' (most likely due to a circular import)

Answer: You cannot name your file "socket.py" otherwise it will interact and mess with the 'import socket' code.It will see that import as an attempt to import itself and will return a circular error

Error: IndexError: list index out of range

Answer: You are probably not giving enough arguments for both the HOST and PORT. Make sure you provide both for your program. If this is not the case then make sure to check any for loops you have and to make sure your loop does not extend past the list's max index

LIST OF FUNCTIONS WE USE FROM THE SOCKET LIBRARY

- decode()
- recv()
- send()
- connect()

LIST OF FUNCTIONS WE MAY ALSO USE

- strip()
- startswith()
- split()
- round()
- join()