Python Block Challenge Lab

Scenario:

You come into work one day to an urgent message from USCYBERCOM. They found a couple of strange looking binary files, but attached was an ICD! Reading the ICD, you realize that the binary files are a weird kind of messaging format. Thankfully, the messages themselves are in cleartext, but any sort of metadata is in binary. As per USCYBERCOM's request, you are to develop a script in python that can read a gzip-compressed tar archive of these files and translate the binary into something human readable. Quick, the world is counting on you!

Part 1

Your first goal should be writing a parser to parse the binmsg files correctly. However, do **NOT** untar the tar archive to disk. You should be reading it into memory and accessing the file members there.

To see if your parser is working correctly, parse the example files and export the data as a CSV file, with each line containing all the decoded data in the data unit header as well as the payload. The timestamp header can just be the default string output of Python's datetime, and the IP fields should be in their human readable form.

You can compare your output to the csv file located in the answer_key folder in the challenge lab. There is a sample parser there as well, but please don't look at this until you've tried to attempt this yourself.

Part 2

Looking at the CSV with all the decoded headers, you should realize that the messages seem to be jumbled up. Each .binmsg file is not a single conversation, but just contains a series of messages. Use the source and destination IP addresses to distinguish each unique conversation, ordering the messages using the “sequence” header.

The customer also wants us to resolve each IP address to a user on the platform. Thankfully, intel has given us a list of usernames that correspond to an IP address that we see in the file.

User	IP Address
Capt Lee	11.17.88.92
Capt Schmitt	1820:26db:91fe:c5a2:1d07:a0e1:0700:2d5a
Capt Tracy	78.52.12.78
Capt Rosales	201.78.52.72
Capt Trusnik	16d9:017c:c1ed:b740:0c46:24b1:3e86:5254

The requested deliverable should be a text file in this format:

```
User 1 and User 2: (alphabetical order)
     [timestamp (default python datetime string)] User: “Message”
     [timestamp (default python datetime string)] User: “Message”

User 3 and User 4: (alphabetical order)
     [timestamp (default python datetime string)] User: “Message”
     [timestamp (default python datetime string)] User: “Message”
```

These conversation blocks should be ordered by the first timestamp of each conversation. Again, you can compare your own output to the example output given in the *answer_key* folder. The code that runs the parser is in there as well, which contains the logic that sorts the messages. Please don’t look at this until you’ve tried to solve it yourself.

ICD:

Each .binmsg file contains a 14-byte header. The *timestamp* is when the file was created, the *total data units* field is the total number of data units present in the file, and the *length of data units* contains the length in bytes of the following data units.

Binmsg Header	Length (bytes)
Timestamp (file creation)	4
Total Data Units	4
Length of Data Units	6

Each data unit in a .binmsg file contains a header. There are two types of data unit, IPv4 and IPv6. IPv4 data units contain a 0x04 in the *IP Version* field of the header, and the IPv6 data units contain a 0x06. These fields appear in the header in descending order (header length is first). The *header length* field just contains the total length of the header in bytes, *sequence* is the relative message number in a conversation; the rest of the fields should be self-explanatory. The payload of each data unit is just a base64 encoded utf-8 string.

IPv4 Header	Length (bytes)
Header Length	1
Timestamp	4
Sequence	2
IP Version (IPv4)	1
Source IP	4
Destination IP	4
Source Port	2
Destination Port	2
Protocol Number	1
Payload Length	4

IPv6 Header	Length (bytes)
Header Length	1
Timestamp	4
Sequence	2
IP Version (IPv6)	1
Source IP	16
Destination IP	16
Source Port	2
Destination Port	2
Protocol Number	1
Payload Length	4