CS372: Socket Programming Project 1

Regina Sanchez, April 25th, 2024

How to run my application

To run my application, ensure that you are in the correct folder (can run pwd to check). Then in the terminal input:

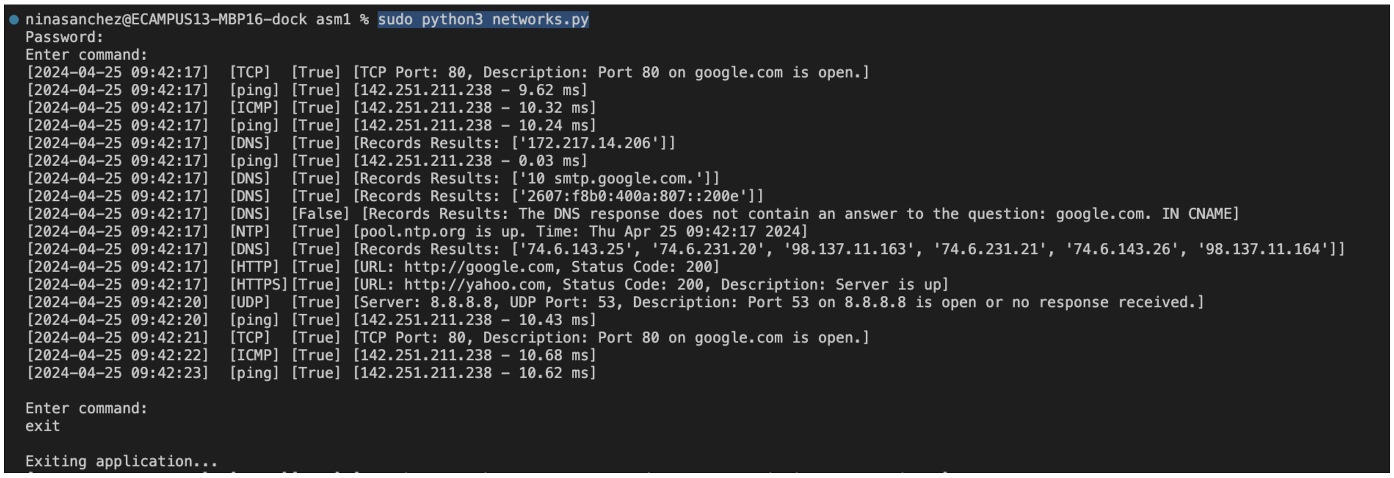
- sudo python3 networks.py

*you will be prompted to enter your device password

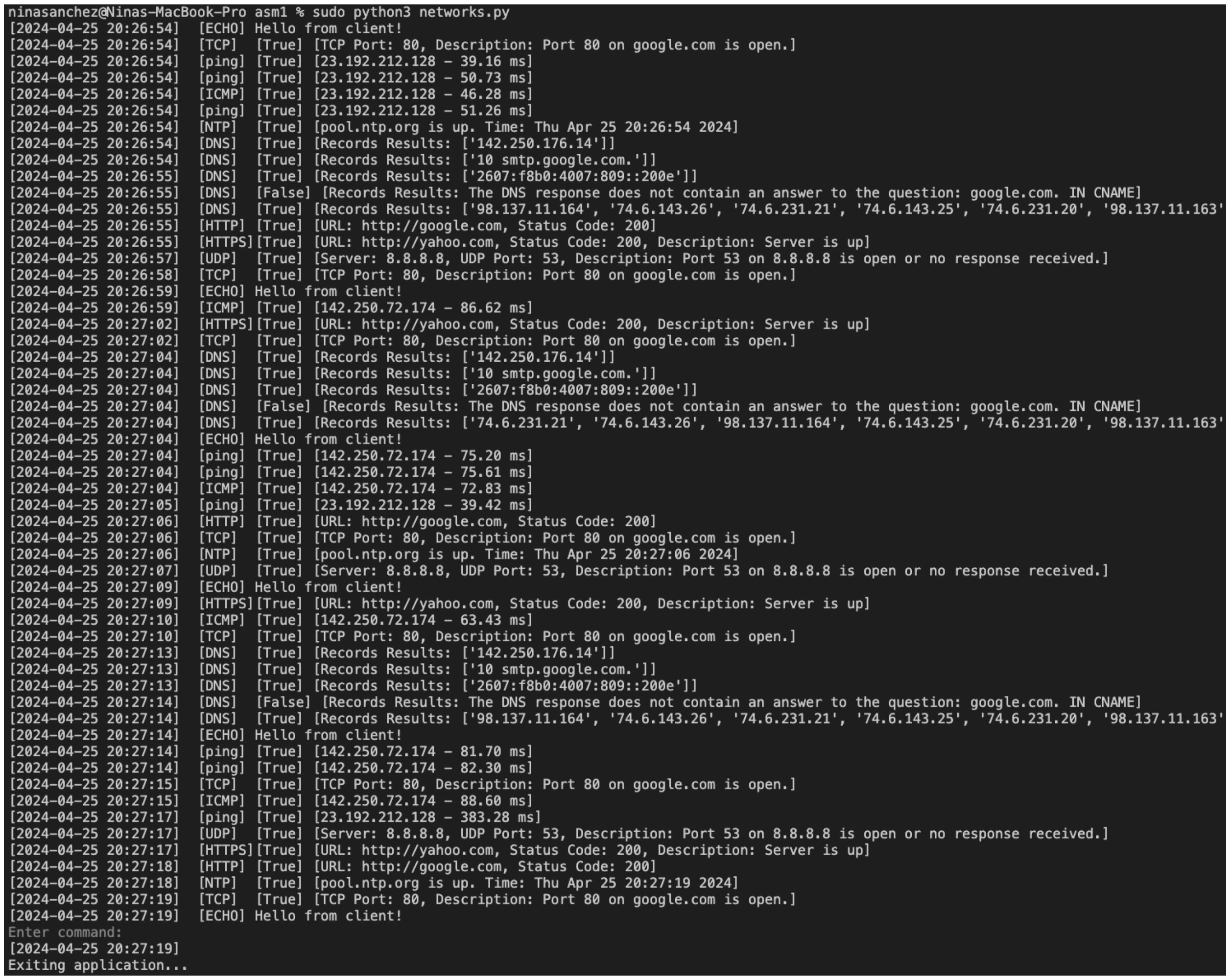
For echo:
Run in two terminals:

1. python3 udp_echo_server.py
2. python3 udp_echo_client.py

Then the application should be running! The two images show when you are asked to enter your sudo password, and the second image shows a longer run in the terminal



```
● ninasanchez@ECAMPUS13-MBP16-dock asm1 % sudo python3 networks.py
  Password:
  Enter command:
  [2024-04-25 09:42:17]  [TCP]  [True] [TCP Port: 80, Description: Port 80 on google.com is open.]
  [2024-04-25 09:42:17]  [ping] [True] [142.251.211.238 - 9.62 ms]
  [2024-04-25 09:42:17]  [ICMP] [True] [142.251.211.238 - 10.32 ms]
  [2024-04-25 09:42:17]  [ping] [True] [142.251.211.238 - 10.24 ms]
  [2024-04-25 09:42:17]  [DNS]  [True] [Records Results: ['172.217.14.206']]
  [2024-04-25 09:42:17]  [ping] [True] [142.251.211.238 - 0.03 ms]
  [2024-04-25 09:42:17]  [DNS]  [True] [Records Results: ['10 smtp.google.com.']]
  [2024-04-25 09:42:17]  [DNS]  [True] [Records Results: ['2607:f8b0:400a:807::200e']]
  [2024-04-25 09:42:17]  [DNS]  [False] [Records Results: The DNS response does not contain an answer to the question: google.com. IN CNAME]
  [2024-04-25 09:42:17]  [NTP]  [True] [pool.ntp.org is up. Time: Thu Apr 25 09:42:17 2024]
  [2024-04-25 09:42:17]  [DNS]  [True] [Records Results: ['74.6.143.25', '74.6.231.20', '98.137.11.163', '74.6.231.21', '74.6.143.26', '98.137.11.164']]
  [2024-04-25 09:42:17]  [HTTP] [True] [URL: http://google.com, Status Code: 200]
  [2024-04-25 09:42:17]  [HTTPS][True] [URL: http://yahoo.com, Status Code: 200, Description: Server is up]
  [2024-04-25 09:42:20]  [UDP]  [True] [Server: 8.8.8.8, UDP Port: 53, Description: Port 53 on 8.8.8.8 is open or no response received.]
  [2024-04-25 09:42:20]  [ping] [True] [142.251.211.238 - 10.43 ms]
  [2024-04-25 09:42:21]  [TCP]  [True] [TCP Port: 80, Description: Port 80 on google.com is open.]
  [2024-04-25 09:42:22]  [ICMP] [True] [142.251.211.238 - 10.68 ms]
  [2024-04-25 09:42:23]  [ping] [True] [142.251.211.238 - 10.62 ms]

  Enter command:
  exit

  Exiting application...
```



```
ninasanchez@Ninas-MacBook-Pro asm1 % sudo python3 networks.py
[2024-04-25 20:26:54]  [ECHO] Hello from client!
[2024-04-25 20:26:54]  [TCP]  [True] [TCP Port: 80, Description: Port 80 on google.com is open.]
[2024-04-25 20:26:54]  [ping] [True] [23.192.212.128 - 39.16 ms]
[2024-04-25 20:26:54]  [ping] [True] [23.192.212.128 - 50.73 ms]
[2024-04-25 20:26:54]  [ICMP] [True] [23.192.212.128 - 46.28 ms]
[2024-04-25 20:26:54]  [ping] [True] [23.192.212.128 - 51.26 ms]
[2024-04-25 20:26:54]  [NTP]  [True] [pool.ntp.org is up. Time: Thu Apr 25 20:26:54 2024]
[2024-04-25 20:26:54]  [DNS]  [True] [Records Results: ['142.250.176.14']]
[2024-04-25 20:26:54]  [DNS]  [True] [Records Results: ['10 smtp.google.com.']]
[2024-04-25 20:26:55]  [DNS]  [True] [Records Results: ['2607:f8b0:4007:809::200e']]
[2024-04-25 20:26:55]  [DNS]  [False] [Records Results: The DNS response does not contain an answer to the question: google.com. IN CNAME]
[2024-04-25 20:26:55]  [DNS]  [True] [Records Results: ['98.137.11.164', '74.6.143.26', '74.6.231.21', '74.6.143.25', '74.6.231.20', '98.137.11.163'
[2024-04-25 20:26:55]  [HTTP] [True] [URL: http://google.com, Status Code: 200]
[2024-04-25 20:26:55]  [HTTPS][True] [URL: http://yahoo.com, Status Code: 200, Description: Server is up]
[2024-04-25 20:26:57]  [UDP]  [True] [Server: 8.8.8.8, UDP Port: 53, Description: Port 53 on 8.8.8.8 is open or no response received.]
[2024-04-25 20:26:58]  [TCP]  [True] [TCP Port: 80, Description: Port 80 on google.com is open.]
[2024-04-25 20:26:59]  [ECHO] Hello from client!
[2024-04-25 20:26:59]  [ICMP] [True] [142.250.72.174 - 86.62 ms]
[2024-04-25 20:27:02]  [HTTPS][True] [URL: http://yahoo.com, Status Code: 200, Description: Server is up]
[2024-04-25 20:27:02]  [TCP]  [True] [TCP Port: 80, Description: Port 80 on google.com is open.]
[2024-04-25 20:27:04]  [DNS]  [True] [Records Results: ['142.250.176.14']]
[2024-04-25 20:27:04]  [DNS]  [True] [Records Results: ['10 smtp.google.com.']]
[2024-04-25 20:27:04]  [DNS]  [True] [Records Results: ['2607:f8b0:4007:809::200e']]
[2024-04-25 20:27:04]  [DNS]  [False] [Records Results: The DNS response does not contain an answer to the question: google.com. IN CNAME]
[2024-04-25 20:27:04]  [DNS]  [True] [Records Results: ['74.6.231.21', '74.6.143.26', '98.137.11.164', '74.6.143.25', '74.6.231.20', '98.137.11.163'
[2024-04-25 20:27:04]  [ECHO] Hello from client!
[2024-04-25 20:27:04]  [ping] [True] [142.250.72.174 - 75.20 ms]
[2024-04-25 20:27:04]  [ping] [True] [142.250.72.174 - 75.61 ms]
[2024-04-25 20:27:04]  [ICMP] [True] [142.250.72.174 - 72.83 ms]
[2024-04-25 20:27:05]  [ping] [True] [23.192.212.128 - 39.42 ms]
[2024-04-25 20:27:06]  [HTTP] [True] [URL: http://google.com, Status Code: 200]
[2024-04-25 20:27:06]  [TCP]  [True] [TCP Port: 80, Description: Port 80 on google.com is open.]
[2024-04-25 20:27:06]  [NTP]  [True] [pool.ntp.org is up. Time: Thu Apr 25 20:27:06 2024]
[2024-04-25 20:27:07]  [UDP]  [True] [Server: 8.8.8.8, UDP Port: 53, Description: Port 53 on 8.8.8.8 is open or no response received.]
[2024-04-25 20:27:09]  [ECHO] Hello from client!
[2024-04-25 20:27:09]  [HTTPS][True] [URL: http://yahoo.com, Status Code: 200, Description: Server is up]
[2024-04-25 20:27:10]  [ICMP] [True] [142.250.72.174 - 63.43 ms]
[2024-04-25 20:27:10]  [TCP]  [True] [TCP Port: 80, Description: Port 80 on google.com is open.]
[2024-04-25 20:27:13]  [DNS]  [True] [Records Results: ['142.250.176.14']]
[2024-04-25 20:27:13]  [DNS]  [True] [Records Results: ['10 smtp.google.com.']]
[2024-04-25 20:27:13]  [DNS]  [True] [Records Results: ['2607:f8b0:4007:809::200e']]
[2024-04-25 20:27:14]  [DNS]  [False] [Records Results: The DNS response does not contain an answer to the question: google.com. IN CNAME]
[2024-04-25 20:27:14]  [DNS]  [True] [Records Results: ['98.137.11.164', '74.6.143.26', '74.6.231.21', '74.6.143.25', '74.6.231.20', '98.137.11.163'
[2024-04-25 20:27:14]  [ECHO] Hello from client!
[2024-04-25 20:27:14]  [ping] [True] [142.250.72.174 - 81.70 ms]
[2024-04-25 20:27:14]  [ping] [True] [142.250.72.174 - 82.30 ms]
[2024-04-25 20:27:15]  [TCP]  [True] [TCP Port: 80, Description: Port 80 on google.com is open.]
[2024-04-25 20:27:15]  [ICMP] [True] [142.250.72.174 - 88.60 ms]
[2024-04-25 20:27:17]  [ping] [True] [23.192.212.128 - 383.28 ms]
[2024-04-25 20:27:17]  [UDP]  [True] [Server: 8.8.8.8, UDP Port: 53, Description: Port 53 on 8.8.8.8 is open or no response received.]
[2024-04-25 20:27:17]  [HTTPS][True] [URL: http://yahoo.com, Status Code: 200, Description: Server is up]
[2024-04-25 20:27:18]  [HTTP] [True] [URL: http://google.com, Status Code: 200]
[2024-04-25 20:27:18]  [NTP]  [True] [pool.ntp.org is up. Time: Thu Apr 25 20:27:19 2024]
[2024-04-25 20:27:19]  [TCP]  [True] [TCP Port: 80, Description: Port 80 on google.com is open.]
[2024-04-25 20:27:19]  [ECHO] Hello from client!
Enter command:
[2024-04-25 20:27:19]
Exiting application...
```

For the echo servers:
In a separate terminal run the UDP echo server

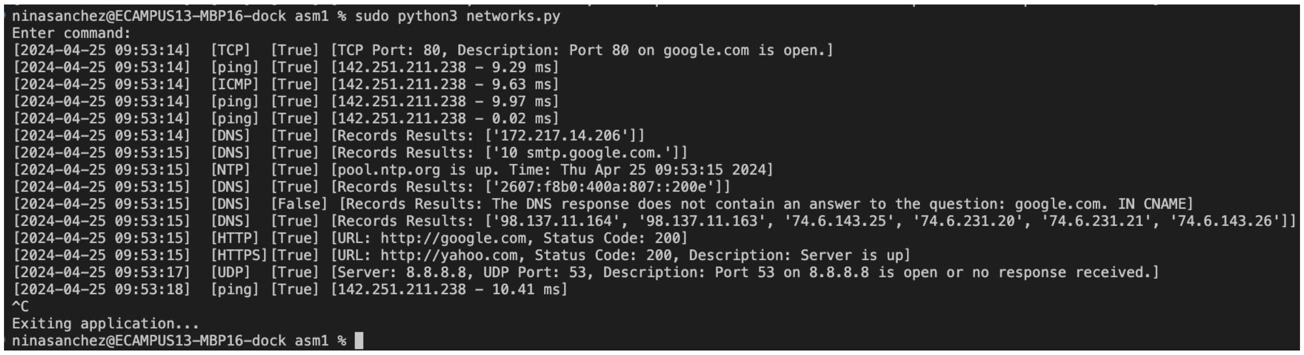
- python3 udp_echo_server.py

Again, in a separate terminal run the

How to stop my application

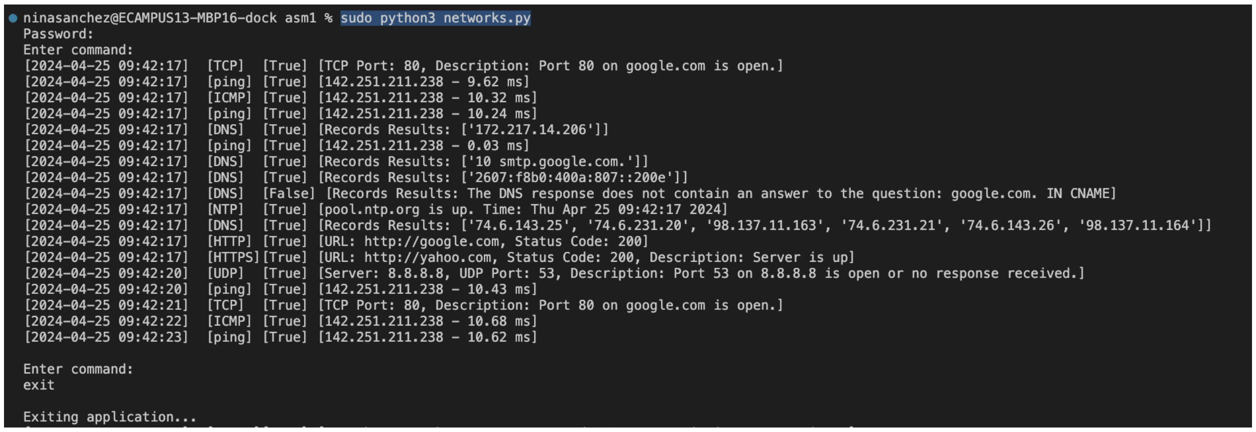
To stop the application while it is running, there are two ways it can be done. You can either press enter / return on your device, the prompt, "Enter Command: ", will appear and you can type in exit → press enter and it will stop running. Or, you can simply press control c (^c) and it will stop running.

This image shows exiting the application via control c



```
ninasanchez@ECAMPUS13-MBP16-dock asm1 % sudo python3 networks.py
Enter command:
[2024-04-25 09:53:14]  [TCP]  [True] [TCP Port: 80, Description: Port 80 on google.com is open.]
[2024-04-25 09:53:14]  [ping] [True] [142.251.211.238 - 9.29 ms]
[2024-04-25 09:53:14]  [ICMP] [True] [142.251.211.238 - 9.63 ms]
[2024-04-25 09:53:14]  [ping] [True] [142.251.211.238 - 9.97 ms]
[2024-04-25 09:53:14]  [ping] [True] [142.251.211.238 - 0.02 ms]
[2024-04-25 09:53:14]  [DNS]  [True] [Records Results: ['172.217.14.206']]
[2024-04-25 09:53:15]  [DNS]  [True] [Records Results: ['10 smtp.google.com.']]
[2024-04-25 09:53:15]  [NTP]  [True] [pool.ntp.org is up. Time: Thu Apr 25 09:53:15 2024]
[2024-04-25 09:53:15]  [DNS]  [True] [Records Results: ['2607:f8b0:400a:807::200e']]
[2024-04-25 09:53:15]  [DNS]  [False] [Records Results: The DNS response does not contain an answer to the question: google.com. IN CNAME]
[2024-04-25 09:53:15]  [DNS]  [True] [Records Results: ['98.137.11.164', '98.137.11.163', '74.6.143.25', '74.6.231.20', '74.6.231.21', '74.6.143.26']]
[2024-04-25 09:53:15]  [HTTP] [True] [URL: http://google.com, Status Code: 200]
[2024-04-25 09:53:15]  [HTTPS][True] [URL: http://yahoo.com, Status Code: 200, Description: Server is up]
[2024-04-25 09:53:17]  [UDP]  [True] [Server: 8.8.8.8, UDP Port: 53, Description: Port 53 on 8.8.8.8 is open or no response received.]
[2024-04-25 09:53:18]  [ping] [True] [142.251.211.238 - 10.41 ms]
^C
Exiting application...
ninasanchez@ECAMPUS13-MBP16-dock asm1 %
```

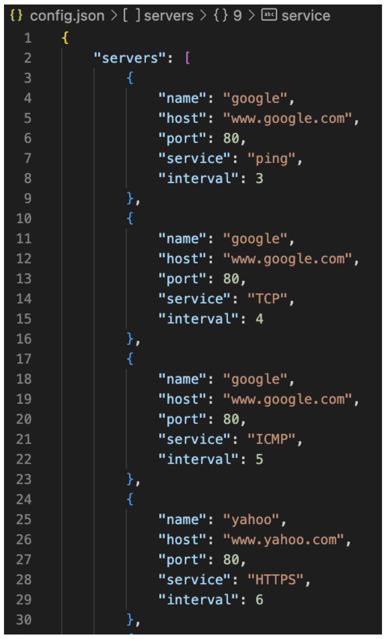
This image shows exiting the application via pressing enter and entering "exit"



```
ninasanchez@ECAMPUS13-MBP16-dock asm1 % sudo python3 networks.py
Password:
Enter command:
[2024-04-25 09:42:17]  [TCP]  [True] [TCP Port: 80, Description: Port 80 on google.com is open.]
[2024-04-25 09:42:17]  [ping] [True] [142.251.211.238 - 9.62 ms]
[2024-04-25 09:42:17]  [ICMP] [True] [142.251.211.238 - 10.32 ms]
[2024-04-25 09:42:17]  [ping] [True] [142.251.211.238 - 10.24 ms]
[2024-04-25 09:42:17]  [DNS]  [True] [Records Results: ['172.217.14.206']]
[2024-04-25 09:42:17]  [ping] [True] [142.251.211.238 - 0.03 ms]
[2024-04-25 09:42:17]  [DNS]  [True] [Records Results: ['10 smtp.google.com.']]
[2024-04-25 09:42:17]  [DNS]  [True] [Records Results: ['2607:f8b0:400a:807::200e']]
[2024-04-25 09:42:17]  [DNS]  [False] [Records Results: The DNS response does not contain an answer to the question: google.com. IN CNAME]
[2024-04-25 09:42:17]  [NTP]  [True] [pool.ntp.org is up. Time: Thu Apr 25 09:42:17 2024]
[2024-04-25 09:42:17]  [DNS]  [True] [Records Results: ['74.6.143.25', '74.6.231.20', '98.137.11.163', '74.6.231.21', '74.6.143.26', '98.137.11.164']]
[2024-04-25 09:42:17]  [HTTP] [True] [URL: http://google.com, Status Code: 200]
[2024-04-25 09:42:17]  [HTTPS][True] [URL: http://yahoo.com, Status Code: 200, Description: Server is up]
[2024-04-25 09:42:20]  [UDP]  [True] [Server: 8.8.8.8, UDP Port: 53, Description: Port 53 on 8.8.8.8 is open or no response received.]
[2024-04-25 09:42:20]  [ping] [True] [142.251.211.238 - 10.43 ms]
[2024-04-25 09:42:21]  [TCP]  [True] [TCP Port: 80, Description: Port 80 on google.com is open.]
[2024-04-25 09:42:22]  [ICMP] [True] [142.251.211.238 - 10.68 ms]
[2024-04-25 09:42:23]  [ping] [True] [142.251.211.238 - 10.62 ms]

Enter command:
exit

Exiting application...
```

How to add / remove addresses

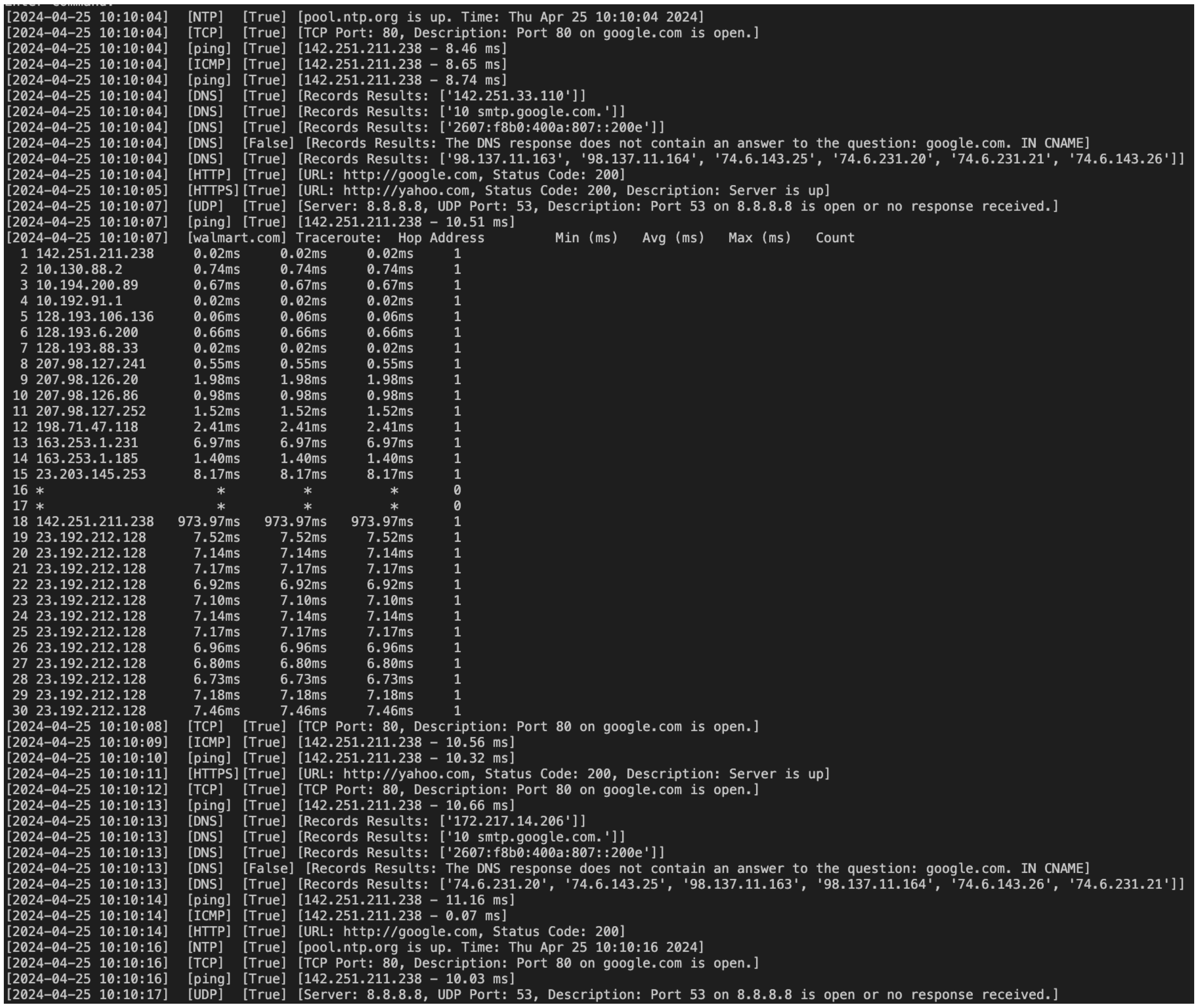
The different addresses that I am pinging, trace routing, etc, are in my json file (config.json). Currently, the file has service for ping, HTTP, HTTPS, ICMP, DNS, NTP, TCP, UDP, and traceroute already configured. If you wish to add a new one simply add in 5 different pieces of information. Firstly, add in the host name or IP address, which will be called "name". Secondly, add in the host, which is the URL. Thirdly, add in the port number. Fourthly, add in the service wanted (HTTP, DNS, etc). Lastly, add in the time wanted to be checking on that which is called "interval". The image below shows the current addresses and service being done. When adding and removing, ensure that the files arent currently running, implement the changes wanted, save the changes, then re-run the program and the changes should appear!



```json
{
    "servers": [
        {
            "name": "google",
            "host": "www.google.com",
            "port": 80,
            "service": "ping",
            "interval": 3
        },
        {
            "name": "google",
            "host": "www.google.com",
            "port": 80,
            "service": "TCP",
            "interval": 4
        },
        {
            "name": "google",
            "host": "www.google.com",
            "port": 80,
            "service": "ICMP",
            "interval": 5
        },
        {
            "name": "yahoo",
            "host": "www.yahoo.com",
            "port": 80,
            "service": "HTTPS",
            "interval": 6
        },
```

Showing all SRS entries + related photos output + code

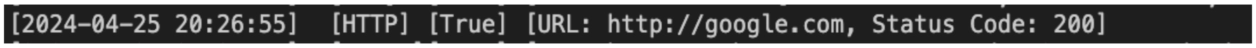
The image below shows the different SRS entries → HTTP, HTTPS, ICMP, DNS, NTP, TCP, UDP, ping, and traceroute.



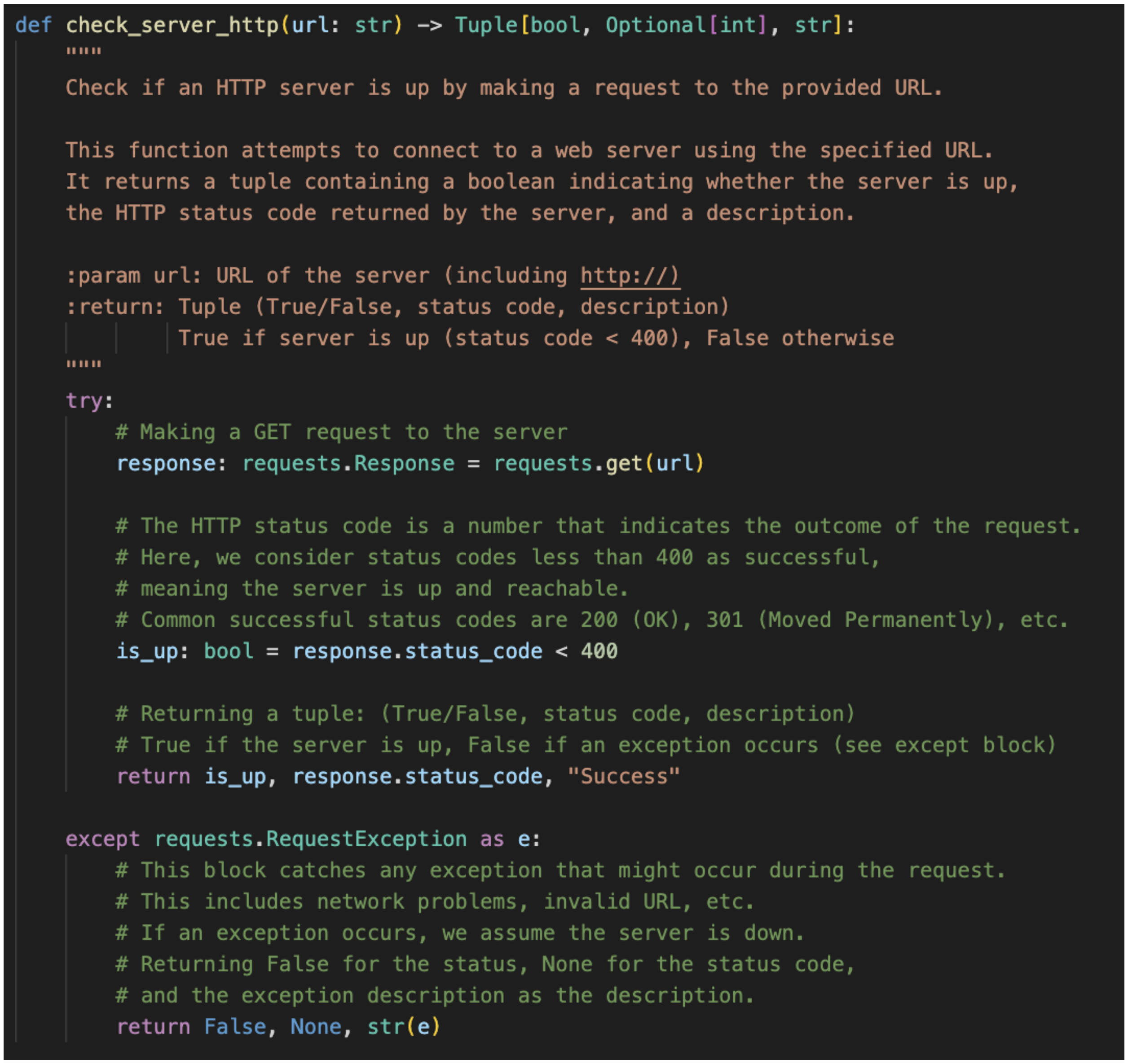
```
[2024-04-25 10:10:04]  [NTP]  [True] [pool.ntp.org is up. Time: Thu Apr 25 10:10:04 2024]
[2024-04-25 10:10:04]  [TCP]  [True] [TCP Port: 80, Description: Port 80 on google.com is open.]
[2024-04-25 10:10:04]  [ping] [True] [142.251.211.238 - 8.46 ms]
[2024-04-25 10:10:04]  [ICMP] [True] [142.251.211.238 - 8.65 ms]
[2024-04-25 10:10:04]  [ping] [True] [142.251.211.238 - 8.74 ms]
[2024-04-25 10:10:04]  [DNS]  [True] [Records Results: ['142.251.33.110']]
[2024-04-25 10:10:04]  [DNS]  [True] [Records Results: ['10 smtp.google.com.']]
[2024-04-25 10:10:04]  [DNS]  [True] [Records Results: ['2607:f8b0:400a:807::200e']]
[2024-04-25 10:10:04]  [DNS]  [False] [Records Results: The DNS response does not contain an answer to the question: google.com. IN CNAME]
[2024-04-25 10:10:04]  [DNS]  [True] [Records Results: ['98.137.11.163', '98.137.11.164', '74.6.143.25', '74.6.231.20', '74.6.231.21', '74.6.143.26']]
[2024-04-25 10:10:04]  [HTTP] [True] [URL: http://google.com, Status Code: 200]
[2024-04-25 10:10:05]  [HTTPS][True] [URL: http://yahoo.com, Status Code: 200, Description: Server is up]
[2024-04-25 10:10:07]  [UDP]  [True] [Server: 8.8.8.8, UDP Port: 53, Description: Port 53 on 8.8.8.8 is open or no response received.]
[2024-04-25 10:10:07]  [ping] [True] [142.251.211.238 - 10.51 ms]
[2024-04-25 10:10:07]  [walmart.com] Traceroute:  Hop Address         Min (ms)   Avg (ms)   Max (ms)   Count
  1 142.251.211.238     0.02ms     0.02ms     0.02ms     1
  2 10.130.88.2         0.74ms     0.74ms     0.74ms     1
  3 10.194.200.89       0.67ms     0.67ms     0.67ms     1
  4 10.192.91.1         0.02ms     0.02ms     0.02ms     1
  5 128.193.106.136     0.06ms     0.06ms     0.06ms     1
  6 128.193.6.200       0.66ms     0.66ms     0.66ms     1
  7 128.193.88.33       0.02ms     0.02ms     0.02ms     1
  8 207.98.127.241      0.55ms     0.55ms     0.55ms     1
  9 207.98.126.20       1.98ms     1.98ms     1.98ms     1
 10 207.98.126.86       0.98ms     0.98ms     0.98ms     1
 11 207.98.127.252      1.52ms     1.52ms     1.52ms     1
 12 198.71.47.118       2.41ms     2.41ms     2.41ms     1
 13 163.253.1.231       6.97ms     6.97ms     6.97ms     1
 14 163.253.1.185       1.40ms     1.40ms     1.40ms     1
 15 23.203.145.253      8.17ms     8.17ms     8.17ms     1
 16 *                      *          *          *       0
 17 *                      *          *          *       0
 18 142.251.211.238   973.97ms   973.97ms   973.97ms     1
 19 23.192.212.128      7.52ms     7.52ms     7.52ms     1
 20 23.192.212.128      7.14ms     7.14ms     7.14ms     1
 21 23.192.212.128      7.17ms     7.17ms     7.17ms     1
 22 23.192.212.128      6.92ms     6.92ms     6.92ms     1
 23 23.192.212.128      7.10ms     7.10ms     7.10ms     1
 24 23.192.212.128      7.14ms     7.14ms     7.14ms     1
 25 23.192.212.128      7.17ms     7.17ms     7.17ms     1
 26 23.192.212.128      6.96ms     6.96ms     6.96ms     1
 27 23.192.212.128      6.80ms     6.80ms     6.80ms     1
 28 23.192.212.128      6.73ms     6.73ms     6.73ms     1
 29 23.192.212.128      7.18ms     7.18ms     7.18ms     1
 30 23.192.212.128      7.46ms     7.46ms     7.46ms     1
[2024-04-25 10:10:08]  [TCP]  [True] [TCP Port: 80, Description: Port 80 on google.com is open.]
[2024-04-25 10:10:09]  [ICMP] [True] [142.251.211.238 - 10.56 ms]
[2024-04-25 10:10:10]  [ping] [True] [142.251.211.238 - 10.32 ms]
[2024-04-25 10:10:11]  [HTTPS][True] [URL: http://yahoo.com, Status Code: 200, Description: Server is up]
[2024-04-25 10:10:12]  [TCP]  [True] [TCP Port: 80, Description: Port 80 on google.com is open.]
[2024-04-25 10:10:13]  [ping] [True] [142.251.211.238 - 10.66 ms]
[2024-04-25 10:10:13]  [DNS]  [True] [Records Results: ['172.217.14.206']]
[2024-04-25 10:10:13]  [DNS]  [True] [Records Results: ['10 smtp.google.com.']]
[2024-04-25 10:10:13]  [DNS]  [True] [Records Results: ['2607:f8b0:400a:807::200e']]
[2024-04-25 10:10:13]  [DNS]  [False] [Records Results: The DNS response does not contain an answer to the question: google.com. IN CNAME]
[2024-04-25 10:10:13]  [DNS]  [True] [Records Results: ['74.6.231.20', '74.6.143.25', '98.137.11.163', '98.137.11.164', '74.6.143.26', '74.6.231.21']]
[2024-04-25 10:10:14]  [ping] [True] [142.251.211.238 - 11.16 ms]
[2024-04-25 10:10:14]  [ICMP] [True] [142.251.211.238 - 0.07 ms]
[2024-04-25 10:10:14]  [HTTP] [True] [URL: http://google.com, Status Code: 200]
[2024-04-25 10:10:16]  [NTP]  [True] [pool.ntp.org is up. Time: Thu Apr 25 10:10:16 2024]
[2024-04-25 10:10:16]  [TCP]  [True] [TCP Port: 80, Description: Port 80 on google.com is open.]
[2024-04-25 10:10:16]  [ping] [True] [142.251.211.238 - 10.03 ms]
[2024-04-25 10:10:17]  [UDP]  [True] [Server: 8.8.8.8, UDP Port: 53, Description: Port 53 on 8.8.8.8 is open or no response received.]
```

HTTP

For my HTTP, on the image above, you can see that the output shows the date + time, the HTTP tag, and the connection. The code I used was from the skeleton code provided.

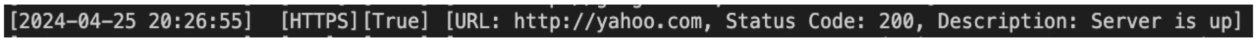


```
[2024-04-25 20:26:55]  [HTTP] [True] [URL: http://google.com, Status Code: 200]
```

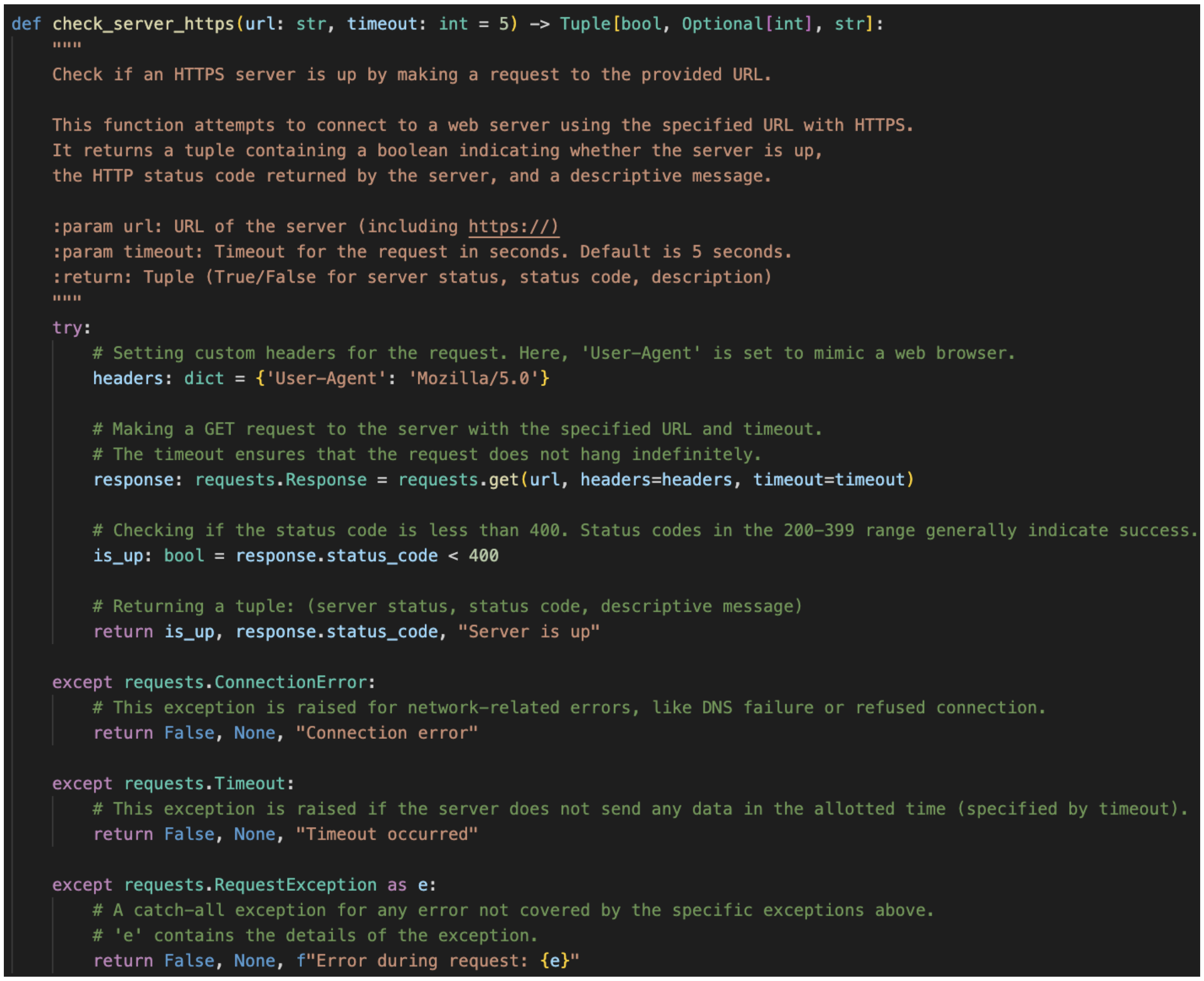


```python
def check_server_http(url: str) -> Tuple[bool, Optional[int], str]:
    """
    Check if an HTTP server is up by making a request to the provided URL.

    This function attempts to connect to a web server using the specified URL.
    It returns a tuple containing a boolean indicating whether the server is up,
    the HTTP status code returned by the server, and a description.

    :param url: URL of the server (including http://)
    :return: Tuple (True/False, status code, description)
                 True if server is up (status code < 400), False otherwise
    """
    try:
        # Making a GET request to the server
        response: requests.Response = requests.get(url)

        # The HTTP status code is a number that indicates the outcome of the request.
        # Here, we consider status codes less than 400 as successful,
        # meaning the server is up and reachable.
        # Common successful status codes are 200 (OK), 301 (Moved Permanently), etc.
        is_up: bool = response.status_code < 400

        # Returning a tuple: (True/False, status code, description)
        # True if the server is up, False if an exception occurs (see except block)
        return is_up, response.status_code, "Success"

    except requests.RequestException as e:
        # This block catches any exception that might occur during the request.
        # This includes network problems, invalid URL, etc.
        # If an exception occurs, we assume the server is down.
        # Returning False for the status, None for the status code,
        # and the exception description as the description.
        return False, None, str(e)
```

HTTPS
In these photos, you can see the code used to get the HTTPS output, which I got from the skeleton code, and what the output looks like

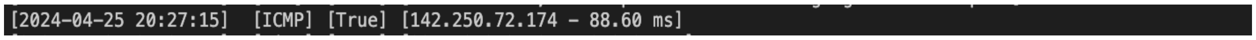


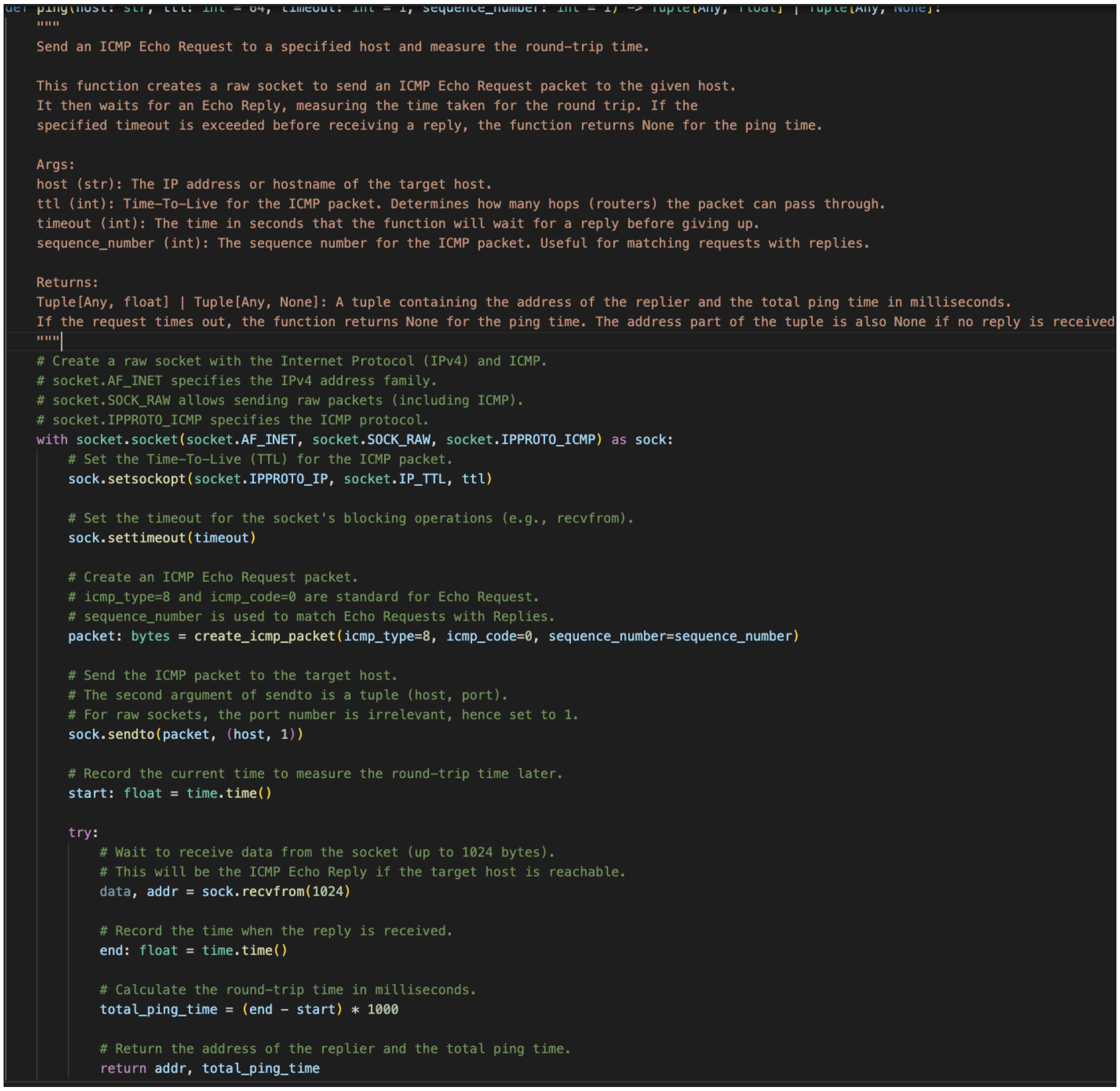
```
[2024-04-25 20:26:55]  [HTTPS][True] [URL: http://yahoo.com, Status Code: 200, Description: Server is up]
```


```
def check_server_https(url: str, timeout: int = 5) -> Tuple[bool, Optional[int], str]:
    """
    Check if an HTTPS server is up by making a request to the provided URL.

    This function attempts to connect to a web server using the specified URL with HTTPS.
    It returns a tuple containing a boolean indicating whether the server is up,
    the HTTP status code returned by the server, and a descriptive message.

    :param url: URL of the server (including https://)
    :param timeout: Timeout for the request in seconds. Default is 5 seconds.
    :return: Tuple (True/False for server status, status code, description)
    """
    try:
        # Setting custom headers for the request. Here, 'User-Agent' is set to mimic a web browser.
        headers: dict = {'User-Agent': 'Mozilla/5.0'}

        # Making a GET request to the server with the specified URL and timeout.
        # The timeout ensures that the request does not hang indefinitely.
        response: requests.Response = requests.get(url, headers=headers, timeout=timeout)

        # Checking if the status code is less than 400. Status codes in the 200-399 range generally indicate success.
        is_up: bool = response.status_code < 400

        # Returning a tuple: (server status, status code, descriptive message)
        return is_up, response.status_code, "Server is up"

    except requests.ConnectionError:
        # This exception is raised for network-related errors, like DNS failure or refused connection.
        return False, None, "Connection error"

    except requests.Timeout:
        # This exception is raised if the server does not send any data in the allotted time (specified by timeout).
        return False, None, "Timeout occurred"

    except requests.RequestException as e:
        # A catch-all exception for any error not covered by the specific exceptions above.
        # 'e' contains the details of the exception.
        return False, None, f"Error during request: {e}"
```

ICMP

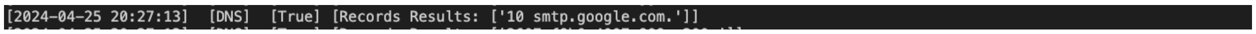
This shows the output for an ICMP call, as well as the code used to get that output. The code used was from the skeleton code provided.


```
[2024-04-25 20:27:15]  [ICMP] [True] [142.250.72.174 - 88.60 ms]
```

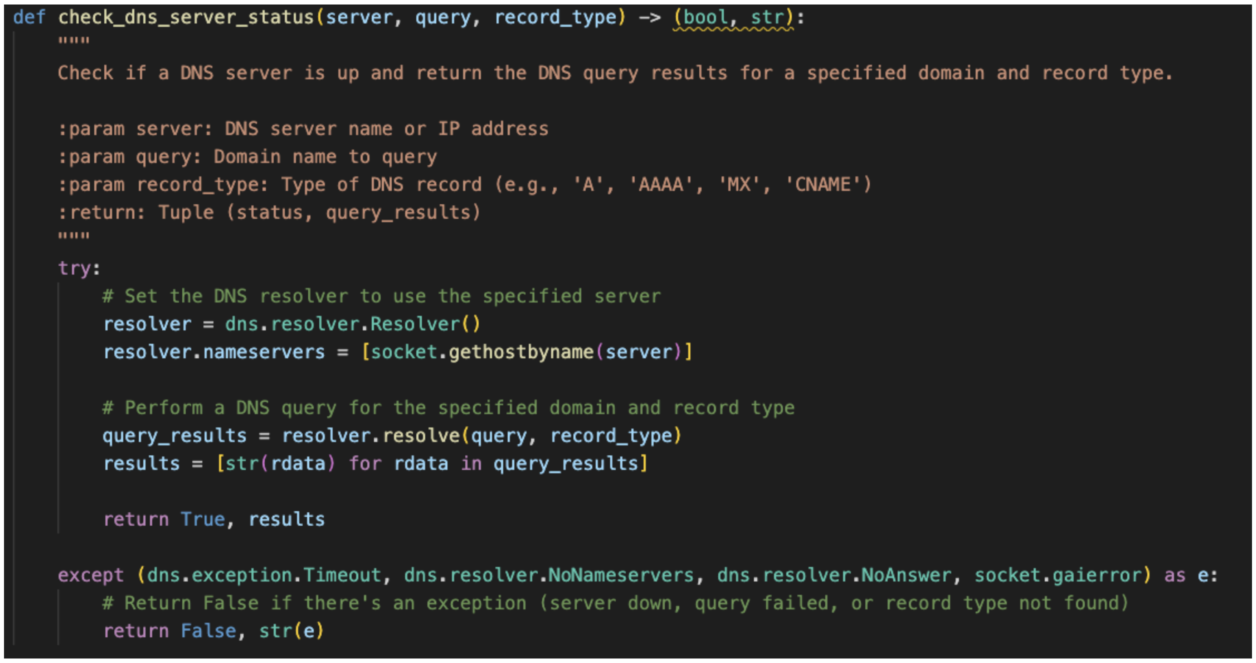


```python
def ping(host: str, ttl: int = 64, timeout: int = 1, sequence_number: int = 1) -> Tuple[Any, float] | Tuple[Any, None]:
    """
    Send an ICMP Echo Request to a specified host and measure the round-trip time.

    This function creates a raw socket to send an ICMP Echo Request packet to the given host.
    It then waits for an Echo Reply, measuring the time taken for the round trip. If the
    specified timeout is exceeded before receiving a reply, the function returns None for the ping time.

    Args:
    host (str): The IP address or hostname of the target host.
    ttl (int): Time-To-Live for the ICMP packet. Determines how many hops (routers) the packet can pass through.
    timeout (int): The time in seconds that the function will wait for a reply before giving up.
    sequence_number (int): The sequence number for the ICMP packet. Useful for matching requests with replies.

    Returns:
    Tuple[Any, float] | Tuple[Any, None]: A tuple containing the address of the replier and the total ping time in milliseconds.
    If the request times out, the function returns None for the ping time. The address part of the tuple is also None if no reply is received
    """
    # Create a raw socket with the Internet Protocol (IPv4) and ICMP.
    # socket.AF_INET specifies the IPv4 address family.
    # socket.SOCK_RAW allows sending raw packets (including ICMP).
    # socket.IPPROTO_ICMP specifies the ICMP protocol.
    with socket.socket(socket.AF_INET, socket.SOCK_RAW, socket.IPPROTO_ICMP) as sock:
        # Set the Time-To-Live (TTL) for the ICMP packet.
        sock.setsockopt(socket.IPPROTO_IP, socket.IP_TTL, ttl)

        # Set the timeout for the socket's blocking operations (e.g., recvfrom).
        sock.settimeout(timeout)

        # Create an ICMP Echo Request packet.
        # icmp_type=8 and icmp_code=0 are standard for Echo Request.
        # sequence_number is used to match Echo Requests with Replies.
        packet: bytes = create_icmp_packet(icmp_type=8, icmp_code=0, sequence_number=sequence_number)

        # Send the ICMP packet to the target host.
        # The second argument of sendto is a tuple (host, port).
        # For raw sockets, the port number is irrelevant, hence set to 1.
        sock.sendto(packet, (host, 1))

        # Record the current time to measure the round-trip time later.
        start: float = time.time()

        try:
            # Wait to receive data from the socket (up to 1024 bytes).
            # This will be the ICMP Echo Reply if the target host is reachable.
            data, addr = sock.recvfrom(1024)

            # Record the time when the reply is received.
            end: float = time.time()

            # Calculate the round-trip time in milliseconds.
            total_ping_time = (end - start) * 1000

            # Return the address of the replier and the total ping time.
            return addr, total_ping_time
```

DNS

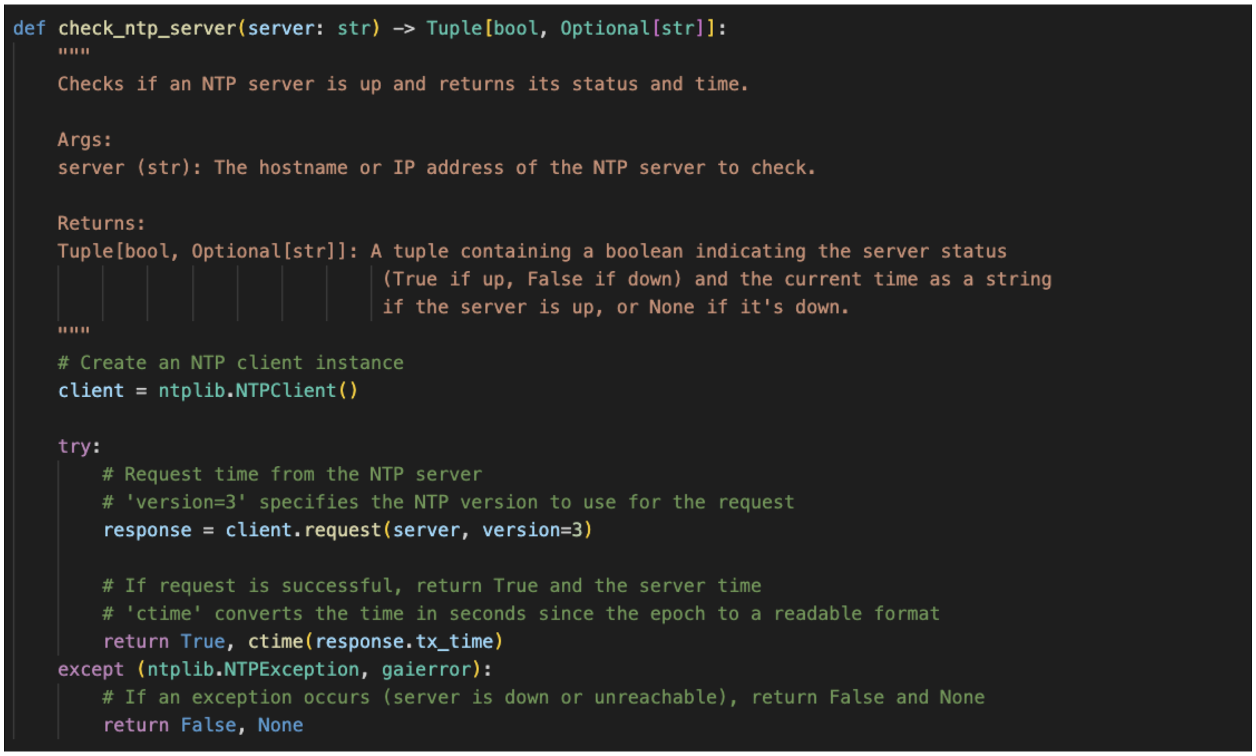


```
[2024-04-25 20:27:13]  [DNS]  [True] [Records Results: ['10 smtp.google.com.']]
```



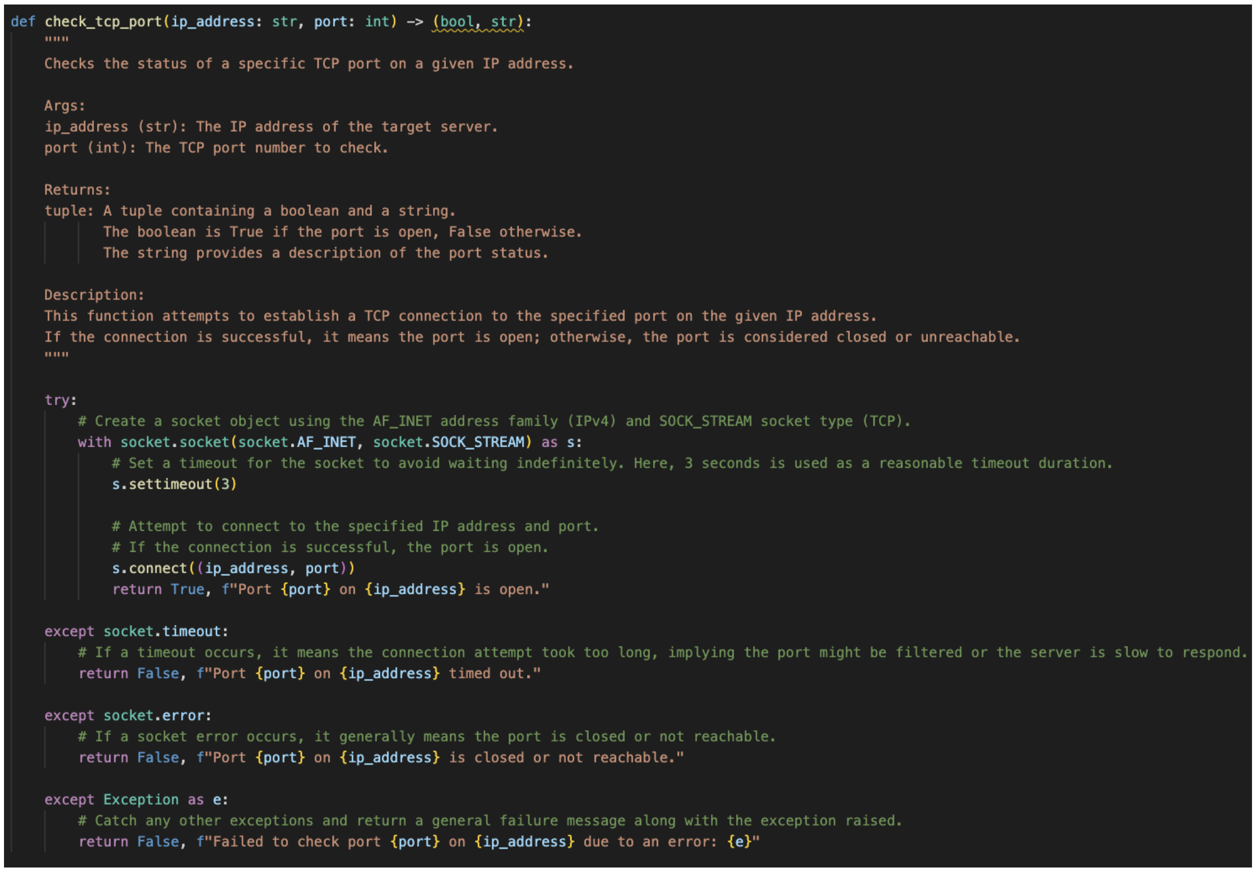
```python
def check_dns_server_status(server, query, record_type) -> (bool, str):
    """
    Check if a DNS server is up and return the DNS query results for a specified domain and record type.

    :param server: DNS server name or IP address
    :param query: Domain name to query
    :param record_type: Type of DNS record (e.g., 'A', 'AAAA', 'MX', 'CNAME')
    :return: Tuple (status, query_results)
    """
    try:
        # Set the DNS resolver to use the specified server
        resolver = dns.resolver.Resolver()
        resolver.nameservers = [socket.gethostbyname(server)]

        # Perform a DNS query for the specified domain and record type
        query_results = resolver.resolve(query, record_type)
        results = [str(rdata) for rdata in query_results]

        return True, results

    except (dns.exception.Timeout, dns.resolver.NoNameservers, dns.resolver.NoAnswer, socket.gaierror) as e:
        # Return False if there's an exception (server down, query failed, or record type not found)
        return False, str(e)
```

NTP



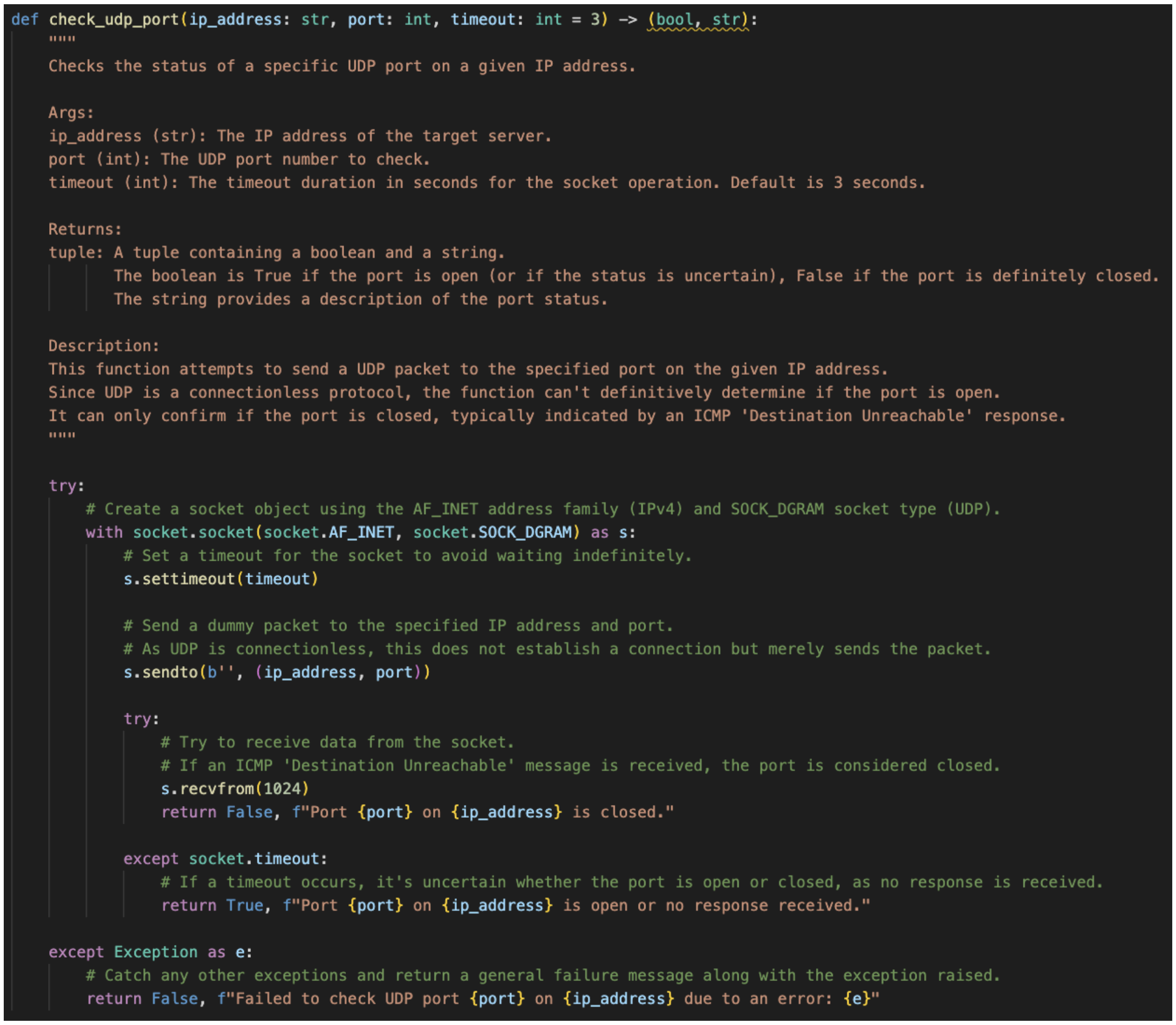
```python
def check_ntp_server(server: str) -> Tuple[bool, Optional[str]]:
    """
    Checks if an NTP server is up and returns its status and time.

    Args:
    server (str): The hostname or IP address of the NTP server to check.

    Returns:
    Tuple[bool, Optional[str]]: A tuple containing a boolean indicating the server status
                                 (True if up, False if down) and the current time as a string
                                 if the server is up, or None if it's down.
    """
    # Create an NTP client instance
    client = ntplib.NTPClient()

    try:
        # Request time from the NTP server
        # 'version=3' specifies the NTP version to use for the request
        response = client.request(server, version=3)

        # If request is successful, return True and the server time
        # 'ctime' converts the time in seconds since the epoch to a readable format
        return True, ctime(response.tx_time)
    except (ntplib.NTPException, gaierror):
        # If an exception occurs (server is down or unreachable), return False and None
        return False, None
```

TCP



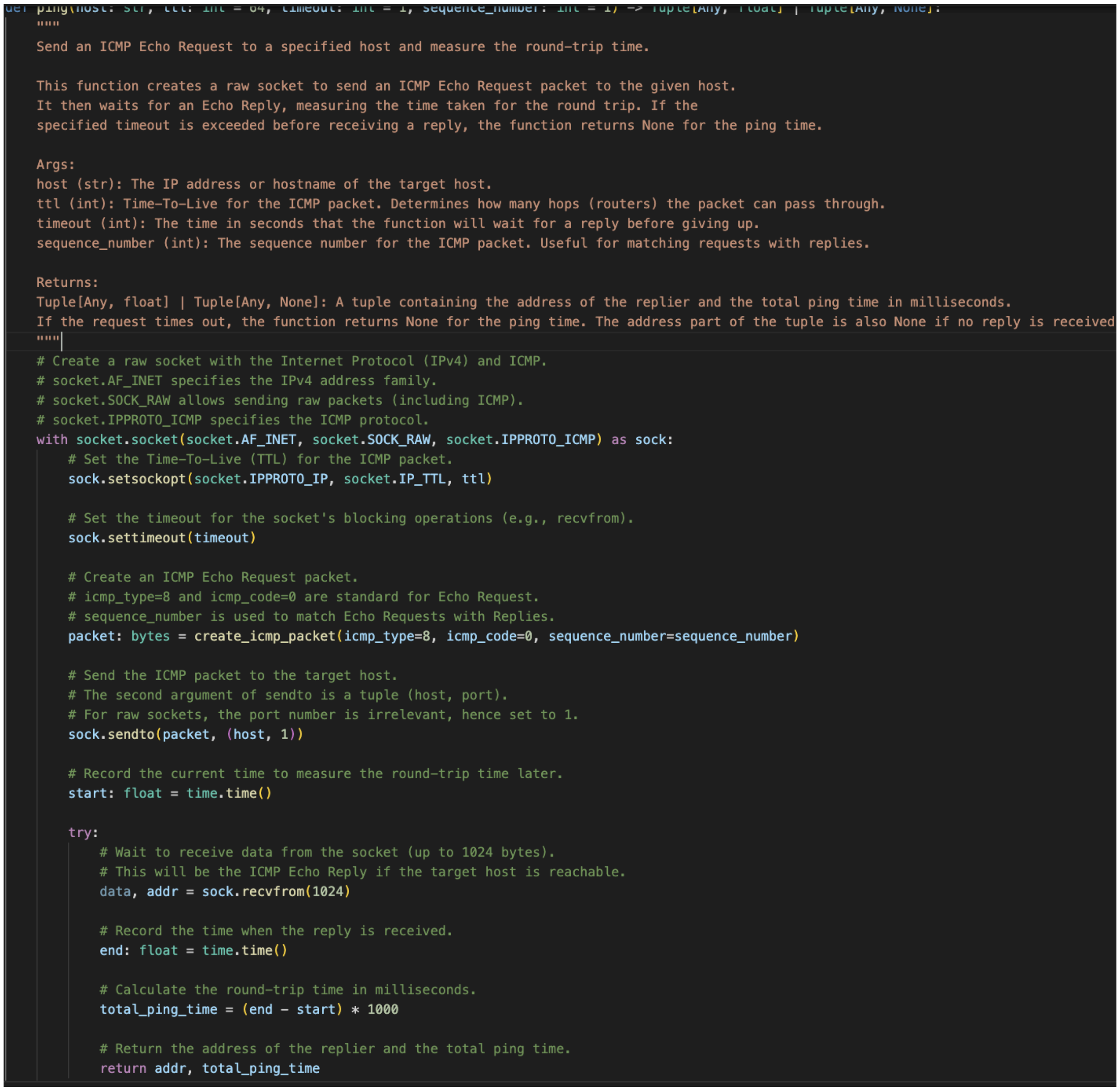
```python
def check_tcp_port(ip_address: str, port: int) -> (bool, str):
    """
    Checks the status of a specific TCP port on a given IP address.

    Args:
    ip_address (str): The IP address of the target server.
    port (int): The TCP port number to check.

    Returns:
    tuple: A tuple containing a boolean and a string.
           The boolean is True if the port is open, False otherwise.
           The string provides a description of the port status.

    Description:
    This function attempts to establish a TCP connection to the specified port on the given IP address.
    If the connection is successful, it means the port is open; otherwise, the port is considered closed or unreachable.
    """

    try:
        # Create a socket object using the AF_INET address family (IPv4) and SOCK_STREAM socket type (TCP).
        with socket.socket(socket.AF_INET, socket.SOCK_STREAM) as s:
            # Set a timeout for the socket to avoid waiting indefinitely. Here, 3 seconds is used as a reasonable timeout duration.
            s.settimeout(3)

            # Attempt to connect to the specified IP address and port.
            # If the connection is successful, the port is open.
            s.connect((ip_address, port))
            return True, f"Port {port} on {ip_address} is open."

    except socket.timeout:
        # If a timeout occurs, it means the connection attempt took too long, implying the port might be filtered or the server is slow to respond.
        return False, f"Port {port} on {ip_address} timed out."

    except socket.error:
        # If a socket error occurs, it generally means the port is closed or not reachable.
        return False, f"Port {port} on {ip_address} is closed or not reachable."

    except Exception as e:
        # Catch any other exceptions and return a general failure message along with the exception raised.
        return False, f"Failed to check port {port} on {ip_address} due to an error: {e}"
```

UDP



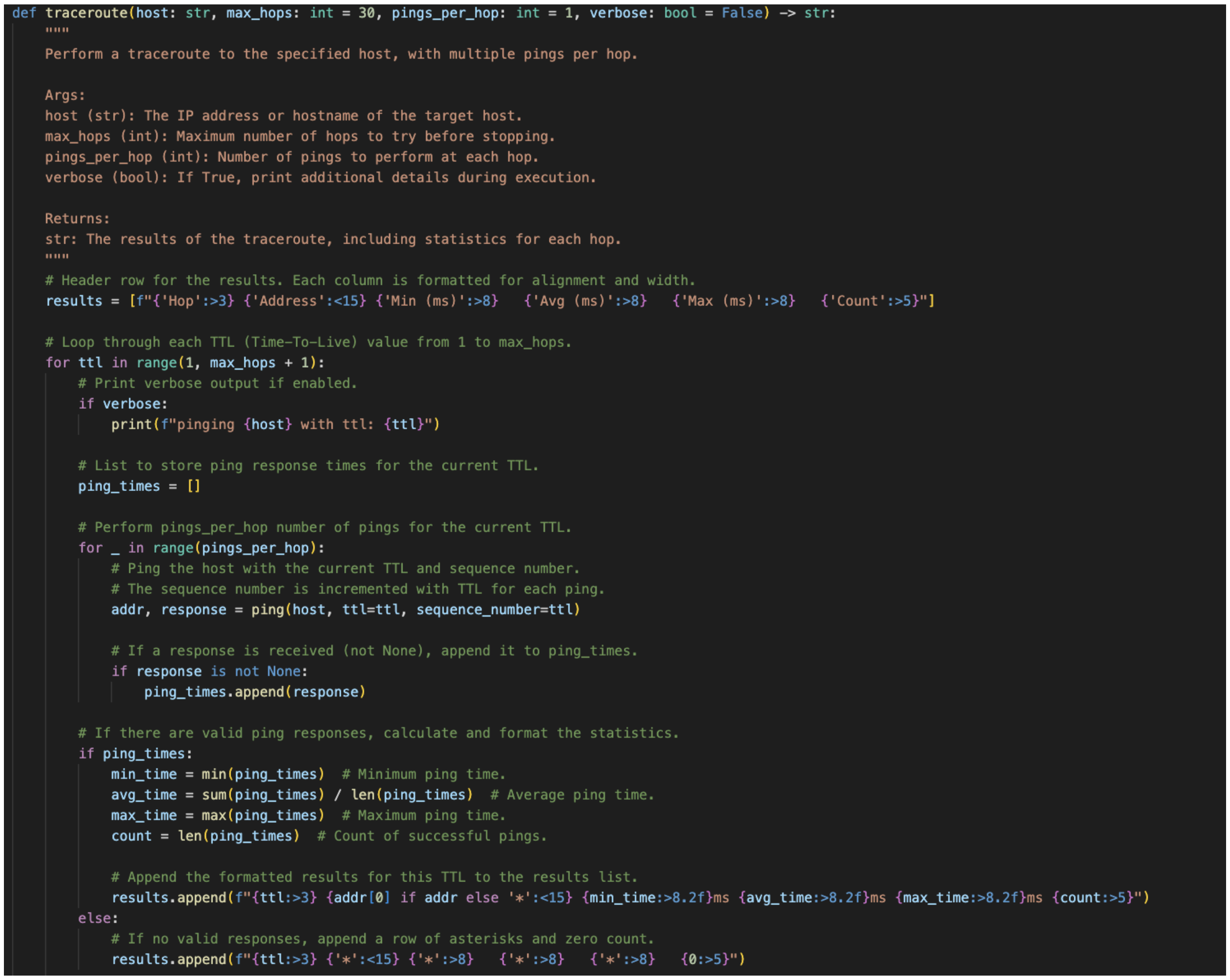
```python
def check_udp_port(ip_address: str, port: int, timeout: int = 3) -> (bool, str):
    """
    Checks the status of a specific UDP port on a given IP address.

    Args:
    ip_address (str): The IP address of the target server.
    port (int): The UDP port number to check.
    timeout (int): The timeout duration in seconds for the socket operation. Default is 3 seconds.

    Returns:
    tuple: A tuple containing a boolean and a string.
           The boolean is True if the port is open (or if the status is uncertain), False if the port is definitely closed.
           The string provides a description of the port status.

    Description:
    This function attempts to send a UDP packet to the specified port on the given IP address.
    Since UDP is a connectionless protocol, the function can't definitively determine if the port is open.
    It can only confirm if the port is closed, typically indicated by an ICMP 'Destination Unreachable' response.
    """

    try:
        # Create a socket object using the AF_INET address family (IPv4) and SOCK_DGRAM socket type (UDP).
        with socket.socket(socket.AF_INET, socket.SOCK_DGRAM) as s:
            # Set a timeout for the socket to avoid waiting indefinitely.
            s.settimeout(timeout)

            # Send a dummy packet to the specified IP address and port.
            # As UDP is connectionless, this does not establish a connection but merely sends the packet.
            s.sendto(b'', (ip_address, port))

            try:
                # Try to receive data from the socket.
                # If an ICMP 'Destination Unreachable' message is received, the port is considered closed.
                s.recvfrom(1024)
                return False, f"Port {port} on {ip_address} is closed."

            except socket.timeout:
                # If a timeout occurs, it's uncertain whether the port is open or closed, as no response is received.
                return True, f"Port {port} on {ip_address} is open or no response received."

    except Exception as e:
        # Catch any other exceptions and return a general failure message along with the exception raised.
        return False, f"Failed to check UDP port {port} on {ip_address} due to an error: {e}"
```

Ping

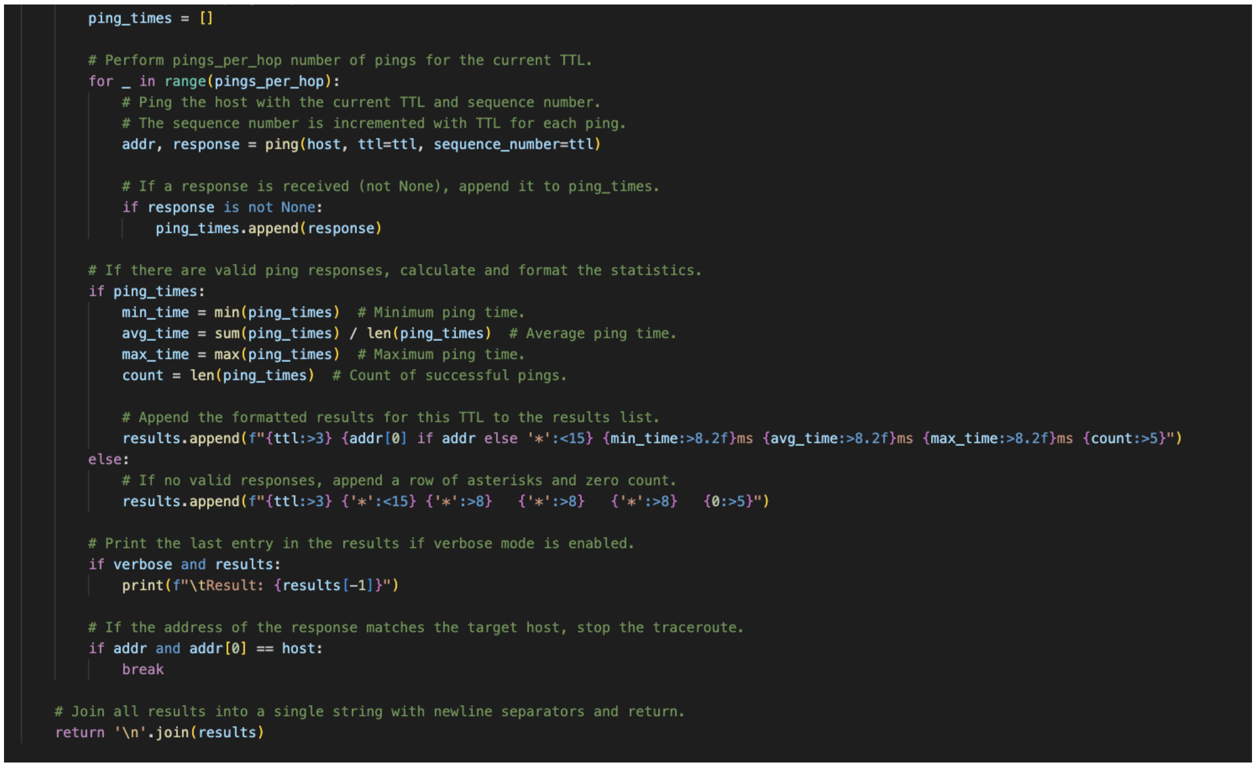


```python
def ping(host: str, ttl: int = 64, timeout: int = 1, sequence_number: int = 1) -> Tuple[Any, float] | Tuple[Any, None]:
    """
    Send an ICMP Echo Request to a specified host and measure the round-trip time.

    This function creates a raw socket to send an ICMP Echo Request packet to the given host.
    It then waits for an Echo Reply, measuring the time taken for the round trip. If the
    specified timeout is exceeded before receiving a reply, the function returns None for the ping time.

    Args:
    host (str): The IP address or hostname of the target host.
    ttl (int): Time-To-Live for the ICMP packet. Determines how many hops (routers) the packet can pass through.
    timeout (int): The time in seconds that the function will wait for a reply before giving up.
    sequence_number (int): The sequence number for the ICMP packet. Useful for matching requests with replies.

    Returns:
    Tuple[Any, float] | Tuple[Any, None]: A tuple containing the address of the replier and the total ping time in milliseconds.
    If the request times out, the function returns None for the ping time. The address part of the tuple is also None if no reply is received
    """
    # Create a raw socket with the Internet Protocol (IPv4) and ICMP.
    # socket.AF_INET specifies the IPv4 address family.
    # socket.SOCK_RAW allows sending raw packets (including ICMP).
    # socket.IPPROTO_ICMP specifies the ICMP protocol.
    with socket.socket(socket.AF_INET, socket.SOCK_RAW, socket.IPPROTO_ICMP) as sock:
        # Set the Time-To-Live (TTL) for the ICMP packet.
        sock.setsockopt(socket.IPPROTO_IP, socket.IP_TTL, ttl)

        # Set the timeout for the socket's blocking operations (e.g., recvfrom).
        sock.settimeout(timeout)

        # Create an ICMP Echo Request packet.
        # icmp_type=8 and icmp_code=0 are standard for Echo Request.
        # sequence_number is used to match Echo Requests with Replies.
        packet: bytes = create_icmp_packet(icmp_type=8, icmp_code=0, sequence_number=sequence_number)

        # Send the ICMP packet to the target host.
        # The second argument of sendto is a tuple (host, port).
        # For raw sockets, the port number is irrelevant, hence set to 1.
        sock.sendto(packet, (host, 1))

        # Record the current time to measure the round-trip time later.
        start: float = time.time()

        try:
            # Wait to receive data from the socket (up to 1024 bytes).
            # This will be the ICMP Echo Reply if the target host is reachable.
            data, addr = sock.recvfrom(1024)

            # Record the time when the reply is received.
            end: float = time.time()

            # Calculate the round-trip time in milliseconds.
            total_ping_time = (end - start) * 1000

            # Return the address of the replier and the total ping time.
            return addr, total_ping_time
```

Traceroute



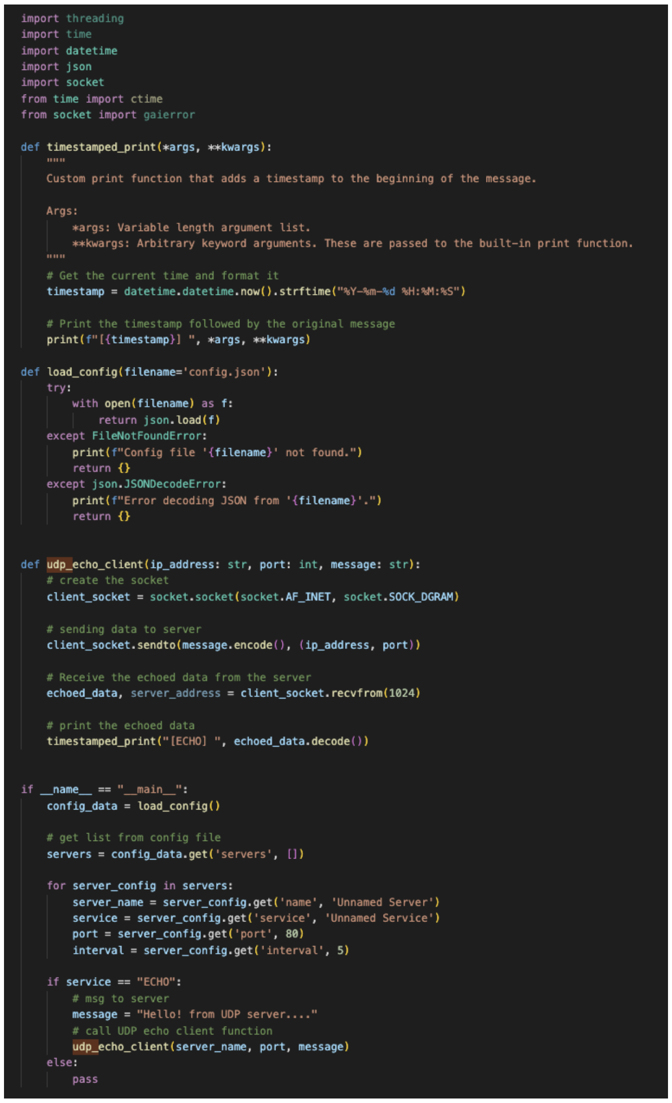
~~~python
def traceroute(host: str, max_hops: int = 30, pings_per_hop: int = 1, verbose: bool = False) -> str:
    """
    Perform a traceroute to the specified host, with multiple pings per hop.

    Args:
    host (str): The IP address or hostname of the target host.
    max_hops (int): Maximum number of hops to try before stopping.
    pings_per_hop (int): Number of pings to perform at each hop.
    verbose (bool): If True, print additional details during execution.

    Returns:
    str: The results of the traceroute, including statistics for each hop.
    """
    # Header row for the results. Each column is formatted for alignment and width.
    results = [f"{'Hop':>3} {'Address':<15} {'Min (ms)':>8}   {'Avg (ms)':>8}   {'Max (ms)':>8}   {'Count':>5}"]

    # Loop through each TTL (Time-To-Live) value from 1 to max_hops.
    for ttl in range(1, max_hops + 1):
        # Print verbose output if enabled.
        if verbose:
            print(f"pinging {host} with ttl: {ttl}")

        # List to store ping response times for the current TTL.
        ping_times = []

        # Perform pings_per_hop number of pings for the current TTL.
        for _ in range(pings_per_hop):
            # Ping the host with the current TTL and sequence number.
            # The sequence number is incremented with TTL for each ping.
            addr, response = ping(host, ttl=ttl, sequence_number=ttl)

            # If a response is received (not None), append it to ping_times.
            if response is not None:
                ping_times.append(response)

        # If there are valid ping responses, calculate and format the statistics.
        if ping_times:
            min_time = min(ping_times)  # Minimum ping time.
            avg_time = sum(ping_times) / len(ping_times)  # Average ping time.
            max_time = max(ping_times)  # Maximum ping time.
            count = len(ping_times)  # Count of successful pings.

            # Append the formatted results for this TTL to the results list.
            results.append(f"{ttl:>3} {addr[0] if addr else '*':<15} {min_time:>8.2f}ms {avg_time:>8.2f}ms {max_time:>8.2f}ms {count:>5}")
        else:
            # If no valid responses, append a row of asterisks and zero count.
            results.append(f"{ttl:>3} {'*':<15} {'*':>8}   {'*':>8}   {'*':>8}   {0:>5}")
```]
~~~



```python
        ping_times = []

        # Perform pings_per_hop number of pings for the current TTL.
        for _ in range(pings_per_hop):
            # Ping the host with the current TTL and sequence number.
            # The sequence number is incremented with TTL for each ping.
            addr, response = ping(host, ttl=ttl, sequence_number=ttl)

            # If a response is received (not None), append it to ping_times.
            if response is not None:
                ping_times.append(response)

        # If there are valid ping responses, calculate and format the statistics.
        if ping_times:
            min_time = min(ping_times)  # Minimum ping time.
            avg_time = sum(ping_times) / len(ping_times)  # Average ping time.
            max_time = max(ping_times)  # Maximum ping time.
            count = len(ping_times)  # Count of successful pings.

            # Append the formatted results for this TTL to the results list.
            results.append(f"{ttl:>3} {addr[0] if addr else '*':<15} {min_time:>8.2f}ms {avg_time:>8.2f}ms {max_time:>8.2f}ms {count:>5}")
        else:
            # If no valid responses, append a row of asterisks and zero count.
            results.append(f"{ttl:>3} {'*':<15} {'*':>8}   {'*':>8}   {'*':>8}   {0:>5}")

        # Print the last entry in the results if verbose mode is enabled.
        if verbose and results:
            print(f"\tResult: {results[-1]}")

        # If the address of the response matches the target host, stop the traceroute.
        if addr and addr[0] == host:
            break

    # Join all results into a single string with newline separators and return.
    return '\n'.join(results)
```

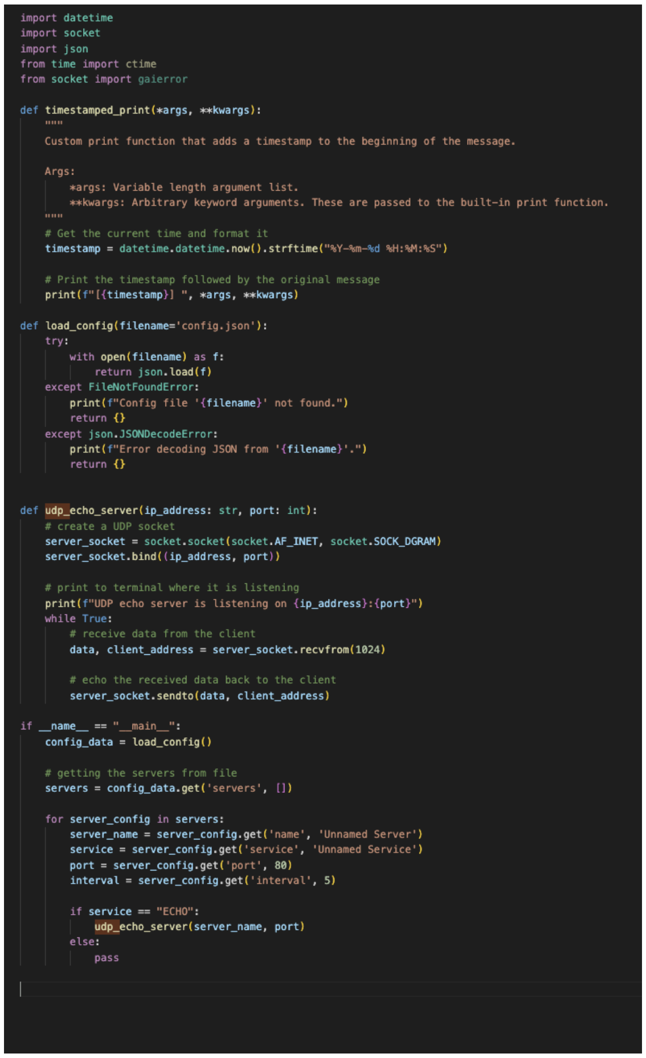
Echo
For my echo, I followed the course code given in the exploration to perform a UDP connection. I ended up creating two files, a UDP client and a UDP server. The client file is imported to my networks file, I just left it this way as I thought it was cleaner to see the two different aspects.

Client:



```python
import threading
import time
import datetime
import json
import socket
from time import ctime
from socket import gaierror

def timestamped_print(*args, **kwargs):
    """
    Custom print function that adds a timestamp to the beginning of the message.

    Args:
        *args: Variable length argument list.
        **kwargs: Arbitrary keyword arguments. These are passed to the built-in print function.
    """
    # Get the current time and format it
    timestamp = datetime.datetime.now().strftime("%Y-%m-%d %H:%M:%S")

    # Print the timestamp followed by the original message
    print(f"[{timestamp}] ", *args, **kwargs)

def load_config(filename='config.json'):
    try:
        with open(filename) as f:
            return json.load(f)
    except FileNotFoundError:
        print(f"Config file '{filename}' not found.")
        return {}
    except json.JSONDecodeError:
        print(f"Error decoding JSON from '{filename}'.")
        return {}


def udp_echo_client(ip_address: str, port: int, message: str):
    # create the socket
    client_socket = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)

    # sending data to server
    client_socket.sendto(message.encode(), (ip_address, port))

    # Receive the echoed data from the server
    echoed_data, server_address = client_socket.recvfrom(1024)

    # print the echoed data
    timestamped_print("[ECHO] ", echoed_data.decode())


if __name__ == "__main__":
    config_data = load_config()

    # get list from config file
    servers = config_data.get('servers', [])

    for server_config in servers:
        server_name = server_config.get('name', 'Unnamed Server')
        service = server_config.get('service', 'Unnamed Service')
        port = server_config.get('port', 80)
        interval = server_config.get('interval', 5)

    if service == "ECHO":
        # msg to server
        message = "Hello! from UDP server...."
        # call UDP echo client function
        udp_echo_client(server_name, port, message)
    else:
        pass
```

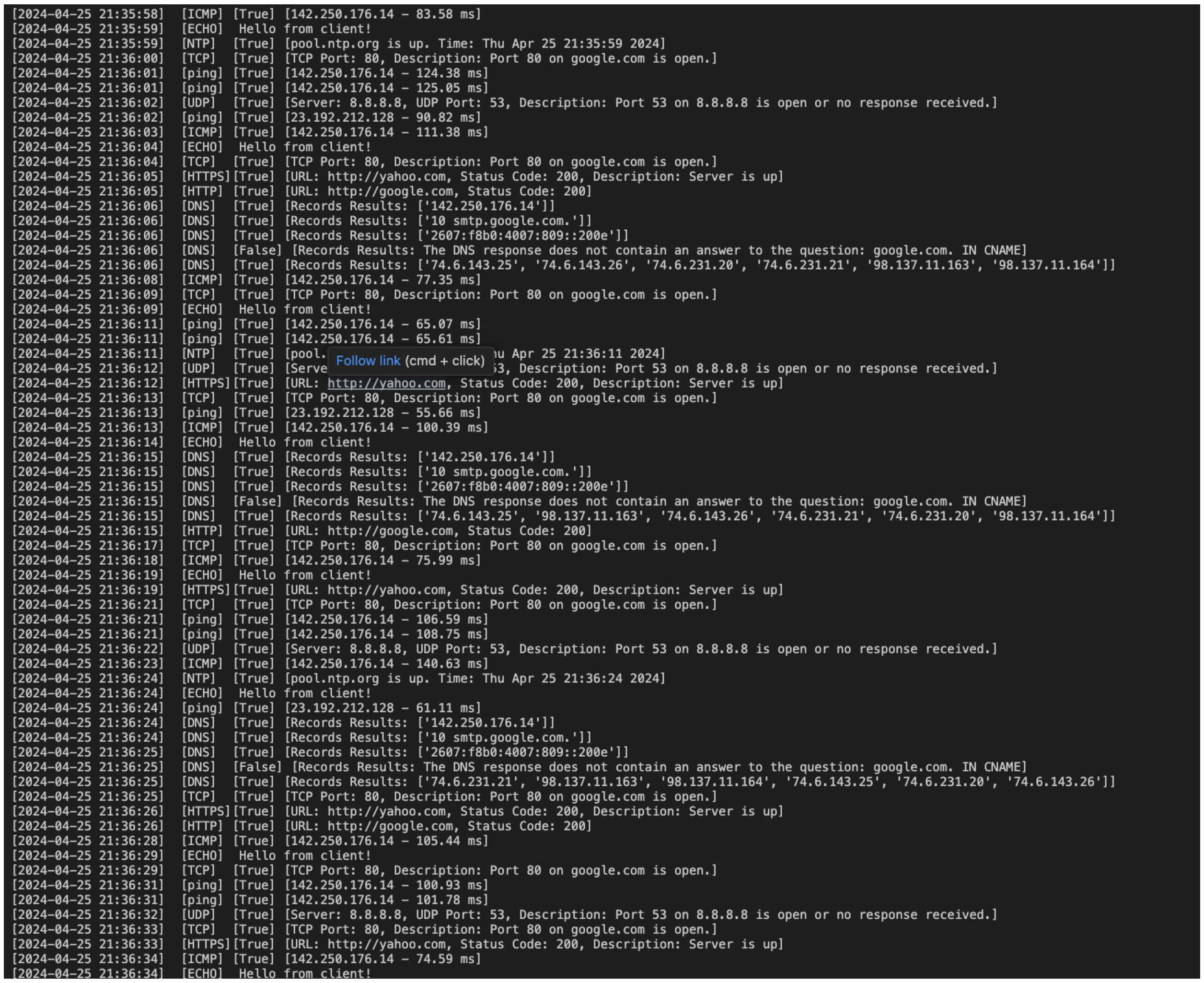
Server:



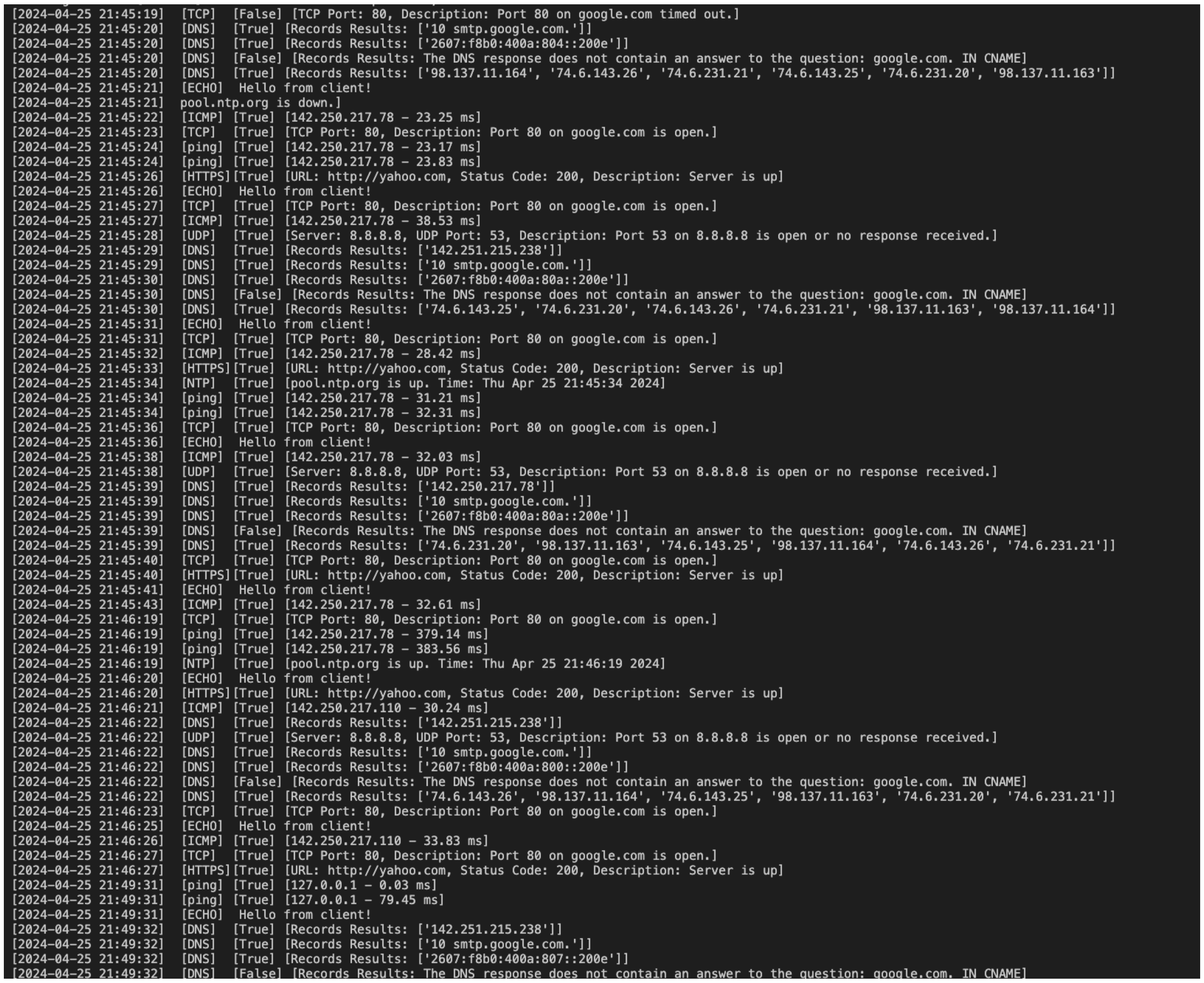
```python
import datetime
import socket
import json
from time import ctime
from socket import gaierror

def timestamped_print(*args, **kwargs):
    """
    Custom print function that adds a timestamp to the beginning of the message.

    Args:
        *args: Variable length argument list.
        **kwargs: Arbitrary keyword arguments. These are passed to the built-in print function.
    """
    # Get the current time and format it
    timestamp = datetime.datetime.now().strftime("%Y-%m-%d %H:%M:%S")

    # Print the timestamp followed by the original message
    print(f"[{timestamp}] ", *args, **kwargs)

def load_config(filename='config.json'):
    try:
        with open(filename) as f:
            return json.load(f)
    except FileNotFoundError:
        print(f"Config file '{filename}' not found.")
        return {}
    except json.JSONDecodeError:
        print(f"Error decoding JSON from '{filename}'.")
        return {}


def udp_echo_server(ip_address: str, port: int):
    # create a UDP socket
    server_socket = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
    server_socket.bind((ip_address, port))

    # print to terminal where it is listening
    print(f"UDP echo server is listening on {ip_address}:{port}")
    while True:
        # receive data from the client
        data, client_address = server_socket.recvfrom(1024)

        # echo the received data back to the client
        server_socket.sendto(data, client_address)

if __name__ == "__main__":
    config_data = load_config()

    # getting the servers from file
    servers = config_data.get('servers', [])

    for server_config in servers:
        server_name = server_config.get('name', 'Unnamed Server')
        service = server_config.get('service', 'Unnamed Service')
        port = server_config.get('port', 80)
        interval = server_config.get('interval', 5)

        if service == "ECHO":
            udp_echo_server(server_name, port)
        else:
            pass
```

Show it running for several minutes + images

Here is the program running for about 15 minutes, as you can see all of the intervals are correctly done.



```
[2024-04-25 21:35:58]  [ICMP] [True] [142.250.176.14 - 83.58 ms]
[2024-04-25 21:35:59]  [ECHO]  Hello from client!
[2024-04-25 21:35:59]  [NTP]  [True] [pool.ntp.org is up. Time: Thu Apr 25 21:35:59 2024]
[2024-04-25 21:36:00]  [TCP]  [True] [TCP Port: 80, Description: Port 80 on google.com is open.]
[2024-04-25 21:36:01]  [ping] [True] [142.250.176.14 - 124.38 ms]
[2024-04-25 21:36:01]  [ping] [True] [142.250.176.14 - 125.05 ms]
[2024-04-25 21:36:02]  [UDP]  [True] [Server: 8.8.8.8, UDP Port: 53, Description: Port 53 on 8.8.8.8 is open or no response received.]
[2024-04-25 21:36:02]  [ping] [True] [23.192.212.128 - 90.82 ms]
[2024-04-25 21:36:03]  [ICMP] [True] [142.250.176.14 - 111.38 ms]
[2024-04-25 21:36:04]  [ECHO]  Hello from client!
[2024-04-25 21:36:04]  [TCP]  [True] [TCP Port: 80, Description: Port 80 on google.com is open.]
[2024-04-25 21:36:05]  [HTTPS][True] [URL: http://yahoo.com, Status Code: 200, Description: Server is up]
[2024-04-25 21:36:05]  [HTTP] [True] [URL: http://google.com, Status Code: 200]
[2024-04-25 21:36:06]  [DNS]  [True] [Records Results: ['142.250.176.14']]
[2024-04-25 21:36:06]  [DNS]  [True] [Records Results: ['10 smtp.google.com.']]
[2024-04-25 21:36:06]  [DNS]  [True] [Records Results: ['2607:f8b0:4007:809::200e']]
[2024-04-25 21:36:06]  [DNS]  [False] [Records Results: The DNS response does not contain an answer to the question: google.com. IN CNAME]
[2024-04-25 21:36:06]  [DNS]  [True] [Records Results: ['74.6.143.25', '74.6.143.26', '74.6.231.20', '74.6.231.21', '98.137.11.163', '98.137.11.164']]
[2024-04-25 21:36:08]  [ICMP] [True] [142.250.176.14 - 77.35 ms]
[2024-04-25 21:36:09]  [TCP]  [True] [TCP Port: 80, Description: Port 80 on google.com is open.]
[2024-04-25 21:36:09]  [ECHO]  Hello from client!
[2024-04-25 21:36:11]  [ping] [True] [142.250.176.14 - 65.07 ms]
[2024-04-25 21:36:11]  [ping] [True] [142.250.176.14 - 65.61 ms]
[2024-04-25 21:36:11]  [NTP]  [True] [pool.                         u Apr 25 21:36:11 2024]
[2024-04-25 21:36:12]  [UDP]  [True] [Serve                         53, Description: Port 53 on 8.8.8.8 is open or no response received.]
[2024-04-25 21:36:12]  [HTTPS][True] [URL: http://yahoo.com, Status Code: 200, Description: Server is up]
[2024-04-25 21:36:13]  [TCP]  [True] [TCP Port: 80, Description: Port 80 on google.com is open.]
[2024-04-25 21:36:13]  [ping] [True] [23.192.212.128 - 55.66 ms]
[2024-04-25 21:36:13]  [ICMP] [True] [142.250.176.14 - 100.39 ms]
[2024-04-25 21:36:14]  [ECHO]  Hello from client!
[2024-04-25 21:36:15]  [DNS]  [True] [Records Results: ['142.250.176.14']]
[2024-04-25 21:36:15]  [DNS]  [True] [Records Results: ['10 smtp.google.com.']]
[2024-04-25 21:36:15]  [DNS]  [True] [Records Results: ['2607:f8b0:4007:809::200e']]
[2024-04-25 21:36:15]  [DNS]  [False] [Records Results: The DNS response does not contain an answer to the question: google.com. IN CNAME]
[2024-04-25 21:36:15]  [DNS]  [True] [Records Results: ['74.6.143.25', '98.137.11.163', '74.6.143.26', '74.6.231.21', '74.6.231.20', '98.137.11.164']]
[2024-04-25 21:36:15]  [HTTP] [True] [URL: http://google.com, Status Code: 200]
[2024-04-25 21:36:17]  [TCP]  [True] [TCP Port: 80, Description: Port 80 on google.com is open.]
[2024-04-25 21:36:18]  [ICMP] [True] [142.250.176.14 - 75.99 ms]
[2024-04-25 21:36:19]  [ECHO]  Hello from client!
[2024-04-25 21:36:19]  [HTTPS][True] [URL: http://yahoo.com, Status Code: 200, Description: Server is up]
[2024-04-25 21:36:21]  [TCP]  [True] [TCP Port: 80, Description: Port 80 on google.com is open.]
[2024-04-25 21:36:21]  [ping] [True] [142.250.176.14 - 106.59 ms]
[2024-04-25 21:36:21]  [ping] [True] [142.250.176.14 - 108.75 ms]
[2024-04-25 21:36:22]  [UDP]  [True] [Server: 8.8.8.8, UDP Port: 53, Description: Port 53 on 8.8.8.8 is open or no response received.]
[2024-04-25 21:36:23]  [ICMP] [True] [142.250.176.14 - 140.63 ms]
[2024-04-25 21:36:24]  [NTP]  [True] [pool.ntp.org is up. Time: Thu Apr 25 21:36:24 2024]
[2024-04-25 21:36:24]  [ECHO]  Hello from client!
[2024-04-25 21:36:24]  [ping] [True] [23.192.212.128 - 61.11 ms]
[2024-04-25 21:36:24]  [DNS]  [True] [Records Results: ['142.250.176.14']]
[2024-04-25 21:36:24]  [DNS]  [True] [Records Results: ['10 smtp.google.com.']]
[2024-04-25 21:36:25]  [DNS]  [True] [Records Results: ['2607:f8b0:4007:809::200e']]
[2024-04-25 21:36:25]  [DNS]  [False] [Records Results: The DNS response does not contain an answer to the question: google.com. IN CNAME]
[2024-04-25 21:36:25]  [DNS]  [True] [Records Results: ['74.6.231.21', '98.137.11.163', '98.137.11.164', '74.6.143.25', '74.6.231.20', '74.6.143.26']]
[2024-04-25 21:36:25]  [TCP]  [True] [TCP Port: 80, Description: Port 80 on google.com is open.]
[2024-04-25 21:36:26]  [HTTPS][True] [URL: http://yahoo.com, Status Code: 200, Description: Server is up]
[2024-04-25 21:36:26]  [HTTP] [True] [URL: http://google.com, Status Code: 200]
[2024-04-25 21:36:28]  [ICMP] [True] [142.250.176.14 - 105.44 ms]
[2024-04-25 21:36:29]  [ECHO]  Hello from client!
[2024-04-25 21:36:29]  [TCP]  [True] [TCP Port: 80, Description: Port 80 on google.com is open.]
[2024-04-25 21:36:31]  [ping] [True] [142.250.176.14 - 100.93 ms]
[2024-04-25 21:36:31]  [ping] [True] [142.250.176.14 - 101.78 ms]
[2024-04-25 21:36:32]  [UDP]  [True] [Server: 8.8.8.8, UDP Port: 53, Description: Port 53 on 8.8.8.8 is open or no response received.]
[2024-04-25 21:36:33]  [TCP]  [True] [TCP Port: 80, Description: Port 80 on google.com is open.]
[2024-04-25 21:36:33]  [HTTPS][True] [URL: http://yahoo.com, Status Code: 200, Description: Server is up]
[2024-04-25 21:36:34]  [ICMP] [True] [142.250.176.14 - 74.59 ms]
[2024-04-25 21:36:34]  [ECHO]  Hello from client!
```



```
[2024-04-25 21:45:19]  [TCP]  [False] [TCP Port: 80, Description: Port 80 on google.com timed out.]
[2024-04-25 21:45:20]  [DNS]  [True] [Records Results: ['10 smtp.google.com.']]
[2024-04-25 21:45:20]  [DNS]  [True] [Records Results: ['2607:f8b0:400a:804::200e']]
[2024-04-25 21:45:20]  [DNS]  [False] [Records Results: The DNS response does not contain an answer to the question: google.com. IN CNAME]
[2024-04-25 21:45:20]  [DNS]  [True] [Records Results: ['98.137.11.164', '74.6.143.26', '74.6.231.21', '74.6.143.25', '74.6.231.20', '98.137.11.163']]
[2024-04-25 21:45:21]  [ECHO]  Hello from client!
[2024-04-25 21:45:21]  pool.ntp.org is down.]
[2024-04-25 21:45:22]  [ICMP] [True] [142.250.217.78 - 23.25 ms]
[2024-04-25 21:45:23]  [TCP]  [True] [TCP Port: 80, Description: Port 80 on google.com is open.]
[2024-04-25 21:45:24]  [ping] [True] [142.250.217.78 - 23.17 ms]
[2024-04-25 21:45:24]  [ping] [True] [142.250.217.78 - 23.83 ms]
[2024-04-25 21:45:26]  [HTTPS][True] [URL: http://yahoo.com, Status Code: 200, Description: Server is up]
[2024-04-25 21:45:26]  [ECHO]  Hello from client!
[2024-04-25 21:45:27]  [TCP]  [True] [TCP Port: 80, Description: Port 80 on google.com is open.]
[2024-04-25 21:45:27]  [ICMP] [True] [142.250.217.78 - 38.53 ms]
[2024-04-25 21:45:28]  [UDP]  [True] [Server: 8.8.8.8, UDP Port: 53, Description: Port 53 on 8.8.8.8 is open or no response received.]
[2024-04-25 21:45:29]  [DNS]  [True] [Records Results: ['142.251.215.238']]
[2024-04-25 21:45:29]  [DNS]  [True] [Records Results: ['10 smtp.google.com.']]
[2024-04-25 21:45:30]  [DNS]  [True] [Records Results: ['2607:f8b0:400a:80a::200e']]
[2024-04-25 21:45:30]  [DNS]  [False] [Records Results: The DNS response does not contain an answer to the question: google.com. IN CNAME]
[2024-04-25 21:45:30]  [DNS]  [True] [Records Results: ['74.6.143.25', '74.6.231.20', '74.6.143.26', '74.6.231.21', '98.137.11.163', '98.137.11.164']]
[2024-04-25 21:45:31]  [ECHO]  Hello from client!
[2024-04-25 21:45:31]  [TCP]  [True] [TCP Port: 80, Description: Port 80 on google.com is open.]
[2024-04-25 21:45:32]  [ICMP] [True] [142.250.217.78 - 28.42 ms]
[2024-04-25 21:45:33]  [HTTPS][True] [URL: http://yahoo.com, Status Code: 200, Description: Server is up]
[2024-04-25 21:45:34]  [NTP]  [True] [pool.ntp.org is up. Time: Thu Apr 25 21:45:34 2024]
[2024-04-25 21:45:34]  [ping] [True] [142.250.217.78 - 31.21 ms]
[2024-04-25 21:45:34]  [ping] [True] [142.250.217.78 - 32.31 ms]
[2024-04-25 21:45:36]  [TCP]  [True] [TCP Port: 80, Description: Port 80 on google.com is open.]
[2024-04-25 21:45:36]  [ECHO]  Hello from client!
[2024-04-25 21:45:38]  [ICMP] [True] [142.250.217.78 - 32.03 ms]
[2024-04-25 21:45:38]  [UDP]  [True] [Server: 8.8.8.8, UDP Port: 53, Description: Port 53 on 8.8.8.8 is open or no response received.]
[2024-04-25 21:45:39]  [DNS]  [True] [Records Results: ['142.250.217.78']]
[2024-04-25 21:45:39]  [DNS]  [True] [Records Results: ['10 smtp.google.com.']]
[2024-04-25 21:45:39]  [DNS]  [True] [Records Results: ['2607:f8b0:400a:80a::200e']]
[2024-04-25 21:45:39]  [DNS]  [False] [Records Results: The DNS response does not contain an answer to the question: google.com. IN CNAME]
[2024-04-25 21:45:39]  [DNS]  [True] [Records Results: ['74.6.231.20', '98.137.11.163', '74.6.143.25', '98.137.11.164', '74.6.143.26', '74.6.231.21']]
[2024-04-25 21:45:40]  [TCP]  [True] [TCP Port: 80, Description: Port 80 on google.com is open.]
[2024-04-25 21:45:40]  [HTTPS][True] [URL: http://yahoo.com, Status Code: 200, Description: Server is up]
[2024-04-25 21:45:41]  [ECHO]  Hello from client!
[2024-04-25 21:45:43]  [ICMP] [True] [142.250.217.78 - 32.61 ms]
[2024-04-25 21:46:19]  [TCP]  [True] [TCP Port: 80, Description: Port 80 on google.com is open.]
[2024-04-25 21:46:19]  [ping] [True] [142.250.217.78 - 379.14 ms]
[2024-04-25 21:46:19]  [ping] [True] [142.250.217.78 - 383.56 ms]
[2024-04-25 21:46:19]  [NTP]  [True] [pool.ntp.org is up. Time: Thu Apr 25 21:46:19 2024]
[2024-04-25 21:46:20]  [ECHO]  Hello from client!
[2024-04-25 21:46:20]  [HTTPS][True] [URL: http://yahoo.com, Status Code: 200, Description: Server is up]
[2024-04-25 21:46:21]  [ICMP] [True] [142.250.217.110 - 30.24 ms]
[2024-04-25 21:46:22]  [DNS]  [True] [Records Results: ['142.251.215.238']]
[2024-04-25 21:46:22]  [UDP]  [True] [Server: 8.8.8.8, UDP Port: 53, Description: Port 53 on 8.8.8.8 is open or no response received.]
[2024-04-25 21:46:22]  [DNS]  [True] [Records Results: ['10 smtp.google.com.']]
[2024-04-25 21:46:22]  [DNS]  [True] [Records Results: ['2607:f8b0:400a:800::200e']]
[2024-04-25 21:46:22]  [DNS]  [False] [Records Results: The DNS response does not contain an answer to the question: google.com. IN CNAME]
[2024-04-25 21:46:22]  [DNS]  [True] [Records Results: ['74.6.143.26', '98.137.11.164', '74.6.143.25', '98.137.11.163', '74.6.231.20', '74.6.231.21']]
[2024-04-25 21:46:23]  [TCP]  [True] [TCP Port: 80, Description: Port 80 on google.com is open.]
[2024-04-25 21:46:25]  [ECHO]  Hello from client!
[2024-04-25 21:46:26]  [ICMP] [True] [142.250.217.110 - 33.83 ms]
[2024-04-25 21:46:27]  [TCP]  [True] [TCP Port: 80, Description: Port 80 on google.com is open.]
[2024-04-25 21:46:27]  [HTTPS][True] [URL: http://yahoo.com, Status Code: 200, Description: Server is up]
[2024-04-25 21:49:31]  [ping] [True] [127.0.0.1 - 0.03 ms]
[2024-04-25 21:49:31]  [ping] [True] [127.0.0.1 - 79.45 ms]
[2024-04-25 21:49:31]  [ECHO]  Hello from client!
[2024-04-25 21:49:32]  [DNS]  [True] [Records Results: ['142.251.215.238']]
[2024-04-25 21:49:32]  [DNS]  [True] [Records Results: ['10 smtp.google.com.']]
[2024-04-25 21:49:32]  [DNS]  [True] [Records Results: ['2607:f8b0:400a:807::200e']]
[2024-04-25 21:49:32]  [DNS]  [False] [Records Results: The DNS response does not contain an answer to the question: google.com. IN CNAME]
```