my_traceroute.py

```
import argparse
import time
import subprocess
import platform
import scapy.all
import socket


def main():

    parser = argparse.ArgumentParser()
    parser.add_argument("d")
    parser.add_argument("-n", action="store_true") #count
    parser.add_argument("-S", action="store_true" ) #packetSize
    parser.add_argument("-q", type=int, default=3) #packetSize

    args = parser.parse_args()
    dest_IP = socket.gethostbyname(args.d)
    ttlCount = 1
    while(True):
        port = 53
        nonans = 0
```

Main function that reads in arguments applies filters and constructs packets before sending probes across network to implement traceroute. Arguments: None Returns:None

```
        for query in range(args.q):
            ip_packet = scapy.all.IP(dst=dest_IP, ttl=ttlCount)
            udp_packet = scapy.all.UDP(dport=port)
```

Sending the packet and receive a reply

```
            packet = ip_packet / scapy.all.ICMP() # Had to use ICMP packets instead to get a response
            reply = scapy.all.sr1(packet, timeout=5, verbose=0)
            port += 1
            if reply is None:
```

Here I tried to create a UDP packet header with the correct port however UDP packets were getting blocked for some reason.

```
                nonans += 1
                print(f"{ttlCount} *")
            else:
                try:
                    symbolic = socket.gethostbyaddr(reply.src)[0]
                except(socket.herror):
                    symbolic = "No Host Name Found"
```

No reply, print * for timeout

```
                if reply.haslayer(scapy.all.ICMP) and reply[scapy.all.ICMP].type == 0:
```

Destination reached, print the details

```
                    if args.n:
                     print(f"{ttlCount}\t{reply.src}")
                    else:
                     print(f"{ttlCount}\t{reply.src}\t{symbolic}")

                    return
                else:
```

Printing the IP address of the intermediate hop

```
                    if args.n:
                        print(f"{ttlCount}\t{reply.src}")
                    else:
                        print(f"{ttlCount}\t{reply.src}\t{symbolic}")

        if args.S:
            print("Number of unanswered packets:" , nonans)
        ttlCount += 1
        if ttlCount > 30:
            break

    return

if __name__ == "__main__":
    main()
```

my_ping.py

```
import argparse
import time
import subprocess
import platform
import math
import socket
import scapy.all

def sendPacket(host, packetSize, packetCount):
```

Function that takes in a users params to create and send ICMP packets to a host. Waits for ECHO_RESPONSE and displays to screen.

Arguments: Host -> Name of the end host the ping will travel too. packetSize -> Amount of data added into the payload of the ICMP packet packetCount -> Amount of packets to send in the Ping Returns: False -> If there is an invalid host Name

```
    try:
        dest_IP = socket.gethostbyname(host)
        for count in range(packetCount):
            packet = scapy.all.IP(dst=dest_IP) / scapy.all.ICMP() / ("Aayan" * packetSize)
            reply = scapy.all.sr1(packet, timeout=2, verbose=False)
            symbolic = socket.gethostbyaddr(reply.src)[0]
            print("Original Ping Address:", host, "\nHost IP:", reply.src, "\nInternal Host Name:", symbolic)
            if (count != packetCount - 1):
                print("\n\n")
    except:
       print("Could not find host name.")
       return False

def main():
    # """
```

Main function that reads in arguments before applying filters and Sending a ping

Arguments: None Returns:None

```
    # """
    parser = argparse.ArgumentParser()
    parser.add_argument("d")
    parser.add_argument("-c", type=int) #count
    parser.add_argument("-i", type=int) #wait
    parser.add_argument("-s", type=int) #packetSize
    parser.add_argument("-t", type=int) #time before terminate
    args = parser.parse_args()

    packetSize = 56
    packetCount = 1
    sleepVal = 1
    startTime = time.time()
    while(True):
        destination = args.d
        if (args.s):
            packetSize = args.s
        if (args.i):
            sleepVal = args.i
        if (args.c):
            for x in range(args.c):
                sendPacket(destination, packetSize, packetCount)
                print("\n\n")
                time.sleep(sleepVal)
            break
        if (args.t):
            currTime = time.time()
            if (currTime - startTime) > args.t:
                break

        print("\n\n")
        keepRun = sendPacket(destination, packetSize, packetCount)
        time.sleep(sleepVal)

        if keepRun == False:
            print("Please rerun the ping with a valid host name.")
            break




main ()
```