Implementation

pinger.py is a modified version of sample_pinger.py converted to work with Python3 rather than Python2. Below are changes done to each of the methods with #Fill in start and #Fill in end.

doOnePing:
For this method all that I need to do is to create a socket with ICMP protocol and set it to the mySocket variable. This is required for other parts of the code.

recieveOnePing:
For this method I first unpacked the recPacket variable similar to how it's packed in the sendOnePing method. I choose to store recPacket data in 2 parts(unpacked and unpackRemain). unpacked contains the time send, checksum, and sequence number, in which I store it in the respective variables timeSend, checksum, and seq. unpackedRemain contains the time to live, in which I store it in the variable ttl.

Now that I have parsed all the variables I need I can then calculate the rtt by taking the difference of the timeReceived(a given variable) and timeSent.

This rtt is used to update the global variables. That includes updating the current global variables rtt_min and rtt_max if the variable rtt is respectively less or more. rtt is added to the global variable rtt_sum. rtt_cnt is incremented by 1. Finally I return the string in the specified format. The string is returned for each successful ping until the user closes the program with control-c.

ping > KeyboardInterrupt:.
For this method I need to implement the KeyboardInterrupt part which upon user pressing control-c will terminate the program and display the statistics. I included an if statement in the case that packets are lost. If the rtt_cnt is greater than 0 I will display the statistics which uses the global variables(rtt_min, rtt_max) and variable rtt_avg computed with (rtt_avg = rtt_sum / rtt_cnt.). If rtt_cnt is not greater than 0, the program prompts the user that there is no pong received.

Scenario A

Pinging localhost(127.0.0.1)

```
C:\Users\Mizushino\Desktop\HW>python pinger.py 127.0.0.1
Pinging 127.0.0.1 using Python:
36 bytes from 127.0.0.1; time =0.0 ms
36 bytes from 127.0.0.1; time =0.0 ms
36 bytes from 127.0.0.1; time =0.0 ms
36 bytes from 127.0.0.1; time =0.0 ms
36 bytes from 127.0.0.1; time =0.0 ms
36 bytes from 127.0.0.1; time =0.0 ms
--- 127.0.0.1 ping statistics ---
round-trip min/avg/max 0.000/0.000/0.000 ms
```

Scenario B

Pinging cs.stonybrook.edu

```
C:\Users\Mizushino\Desktop\HW>python pinger.py cs.stonybrook.edu
Pinging 23.185.0.2 using Python:
36 bytes from 23.185.0.2; time =20.8 ms
36 bytes from 23.185.0.2; time =16.8 ms
36 bytes from 23.185.0.2; time =12.4 ms
36 bytes from 23.185.0.2; time =16.5 ms
36 bytes from 23.185.0.2; time =12.2 ms
36 bytes from 23.185.0.2; time =35.6 ms
36 bytes from 23.185.0.2; time =17.6 ms
36 bytes from 23.185.0.2; time =22.6 ms
--- cs.stonybrook.edu ping statistics ---
round-trip min/avg/max 12.232/19.315/35.569 ms
```

Scenario C

Below are the results from pinging servers from 4 different continents. The server IPs that are used for this scenario are obtained through the website (https://public-dns.info/).

1. France/Orshwihr : 82.127.168.41

```
C:\Users\Mizushino\Desktop\HW>python pinger.py 82.127.168.41
Pinging 82.127.168.41 using Python:
36 bytes from 82.127.168.41; time =112.9 ms
36 bytes from 82.127.168.41; time =116.4 ms
36 bytes from 82.127.168.41; time =112.9 ms
36 bytes from 82.127.168.41; time =112.2 ms
36 bytes from 82.127.168.41; time =108.4 ms
36 bytes from 82.127.168.41; time =112.4 ms
36 bytes from 82.127.168.41; time =112.7 ms
36 bytes from 82.127.168.41; time =122.5 ms
--- 82.127.168.41 ping statistics ---
round-trip min/avg/max 108.379/113.800/122.526 ms
```

2. Brazil/Cicero Dantas : 45.225.123.88

```
C:\Users\Mizushino\Desktop\HW>python pinger.py 45.225.123.88
Pinging 45.225.123.88 using Python:
36 bytes from 45.225.123.88; time =135.2 ms
36 bytes from 45.225.123.88; time =132.2 ms
36 bytes from 45.225.123.88; time =131.4 ms
36 bytes from 45.225.123.88; time =131.8 ms
36 bytes from 45.225.123.88; time =136.2 ms
36 bytes from 45.225.123.88; time =130.9 ms
36 bytes from 45.225.123.88; time =133.1 ms
36 bytes from 45.225.123.88; time =127.7 ms
36 bytes from 45.225.123.88; time =130.6 ms
--- 45.225.123.88 ping statistics ---
round-trip min/avg/max 127.673/132.120/136.192 ms
```

3. India/Mumbai : 182.56.167.143

```
C:\Users\Mizushino\Desktop\HW>python pinger.py 61.69.106.78
Pinging 61.69.106.78 using Python:
36 bytes from 61.69.106.78; time =251.3 ms
36 bytes from 61.69.106.78; time =250.6 ms
36 bytes from 61.69.106.78; time =249.3 ms
36 bytes from 61.69.106.78; time =248.7 ms
36 bytes from 61.69.106.78; time =254.1 ms
36 bytes from 61.69.106.78; time =249.9 ms
36 bytes from 61.69.106.78; time =249.0 ms
--- 61.69.106.78 ping statistics ---
round-trip min/avg/max 248.703/250.415/254.078 ms
```

4. Australia/Brisbane : 61.69.106.78

```
C:\Users\Mizushino\Desktop\HW>python pinger.py 182.56.167.143
Pinging 182.56.167.143 using Python:
36 bytes from 182.56.167.143; time =207.6 ms
36 bytes from 182.56.167.143; time =207.6 ms
36 bytes from 182.56.167.143; time =208.5 ms
36 bytes from 182.56.167.143; time =209.0 ms
36 bytes from 182.56.167.143; time =208.5 ms
36 bytes from 182.56.167.143; time =208.9 ms
--- 182.56.167.143 ping statistics ---
round-trip min/avg/max 207.551/208.358/209.045 ms
```

Scenario D

The table below organizes the minimum round trip times of the respective servers and their estimated distances from my location. There is a clear trend that the further away the server is the higher the minimum round trip times. These scenarios (A, B and C) show that a server's and a client's geological location greatly contributes to the round trip time.

	localhost	cs.stonybrook.edu	France	Brazil	India	Australia

Min RTT(ms)	0	12.232	108.379	127.673	207.551	248.703
Distance from me (miles)	0	~60	~3,900	~4,300	~7,800	~9,700

Min RTT is obtain through the scenarios above
Distance from me is obtained with Google Maps' Measure Distance function.