**Dennis Osafo**

dennis.osafo@slu.edu | (314) 657-7723

**Work Samples**

Table of Contents

Computer Science: Developing 2D Backgammon game3

Backgammon Interface 3

Programming Language: Python (snippet of the code) 4

Business Analytics: Data analysis using R and Tableau5

Data analysis with R 5

Data Analysis: Histogram (Dash Board)6

Data Analysis: Column charts7

Data Analysis: Scattered plot8

Data Analysis: Box plot10

Business analytics with Tableau11

Data Visualization (HR Analysis: Employee Profile) 11

Data Visualization (Employee company data) 12

**Echo Application to measure TCP Performance……………………………………………………………………13**

Echo Client-Server Application………………………………………………………………………………….13

Performing RTT and Throughput Measurements……………………………………………………….16

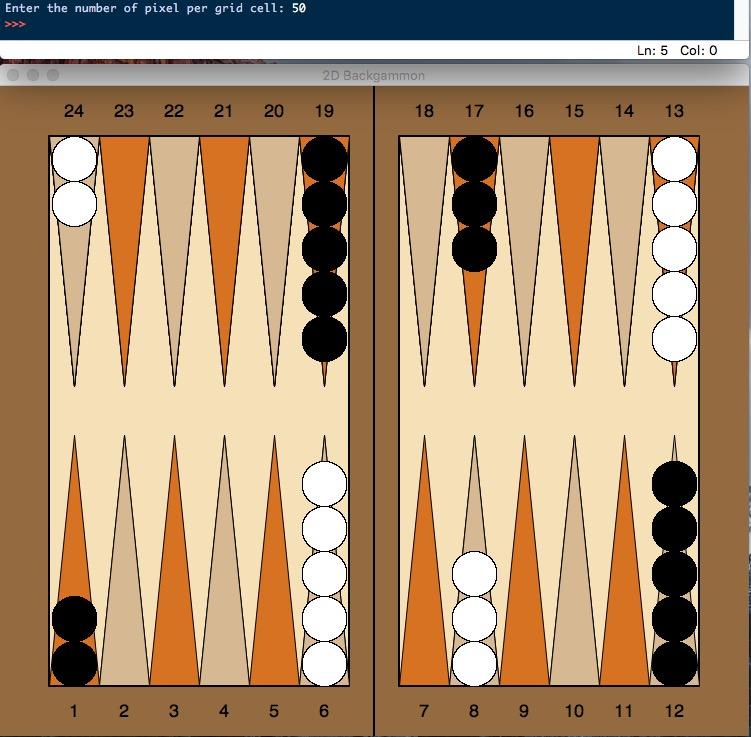
RTT (round trip time) Measurement - client-server results………………………………………...21

**Automation Script: Remove special characters from data fields of raw data……………………….22**

**Computer Science: Developing 2D Backgammon game**

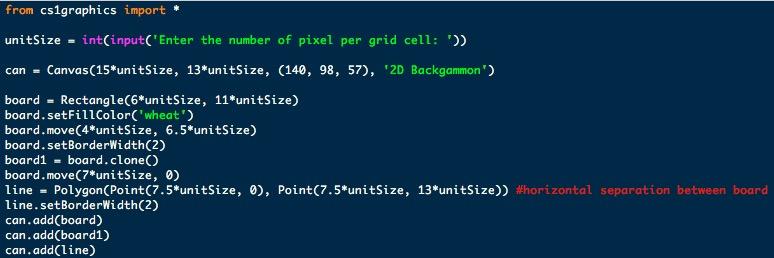
**Background Interface**

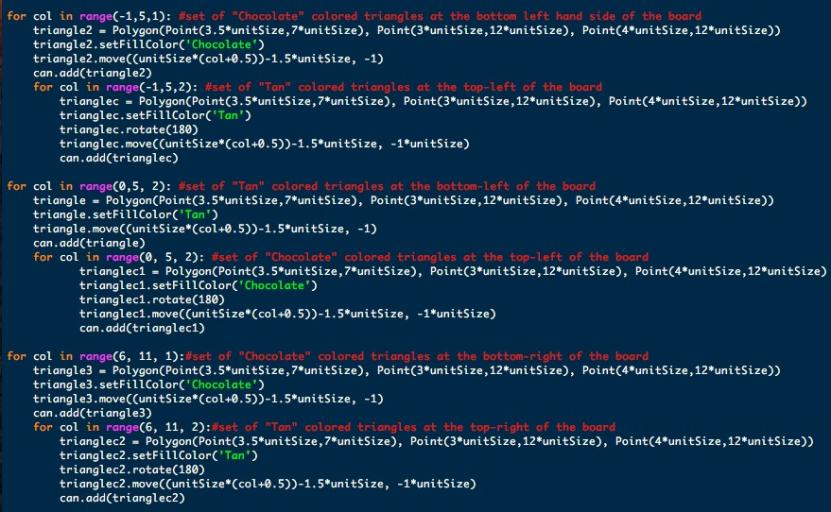
Engineered and developed a backgammon program using Python and cs1graphics for my final project. Below is an extract of the interface of the backgammon game:



**Python (snippet of the code)**

Python was the preferred programming language used to create this program. Below is a dashboard of the extracted codes used in, conjunction with the cs1graphics library developed by **Michael H. Goldwasser and David Letscher** from Saint Louis University, to display the graphics:





Business Analytics: Data analysis using R and Tableau

**Data analysis with R**

A data set in .csv format was used to complete the data analysis based on Laptop sales for the month of January 2008 (i.e. “LaptopSalesJanuary2008.csv”).

Below is the diagram and a snippet of the code used to create a histogram for the retail price of laptops:

**Code:**

>>Laptops <- read.csv("LaptopSalesJanuary2008.csv")

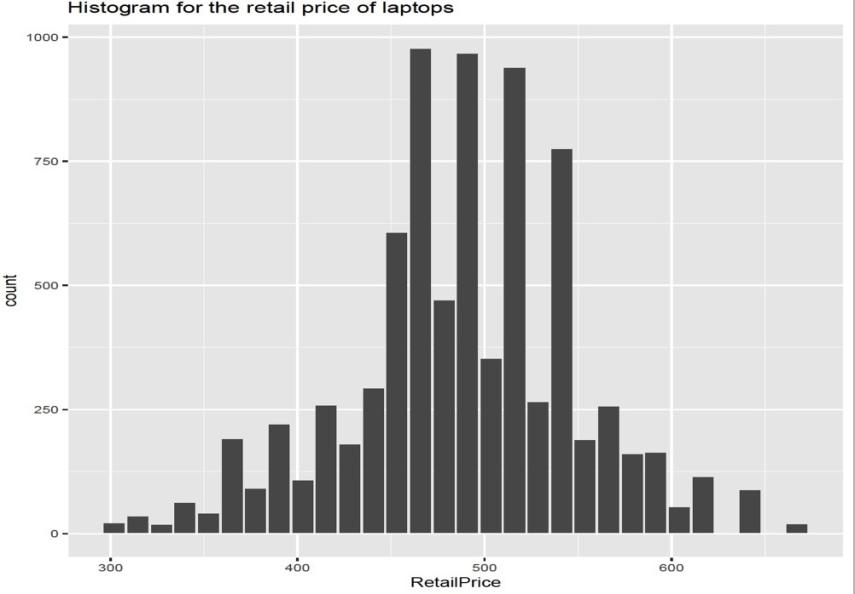
>>head(Laptops) # display the first six rows of data

>>RetailPrice <- Laptops$Retail.Price

>>library(ggplot2)

>>ggplot(Laptops, aes(x=RetailPrice)) + geom\_histogram() + ggtitle("Histogram for the retail price of laptops")

**Results:**

****

**Dashboard of Histograms**

Below is a panel to display four histograms displaying the Retail Price, hard drive size (HD Size), Battery Life, and RAM GB of the Laptop in LaptopSalesJanuary2008.csv dataset:

**Code:**

>>HDSize <- Laptops$HD.Size..GB.

>>BatteryLife<-Laptops$Battery.Life..Hours.

>>RAM\_GB<-Laptops$RAM..GB.

>>display1 <- ggplot(Laptops, aes(x=RetailPrice)) + geom\_histogram() + ggtitle("Histogram for retail price of laptops")

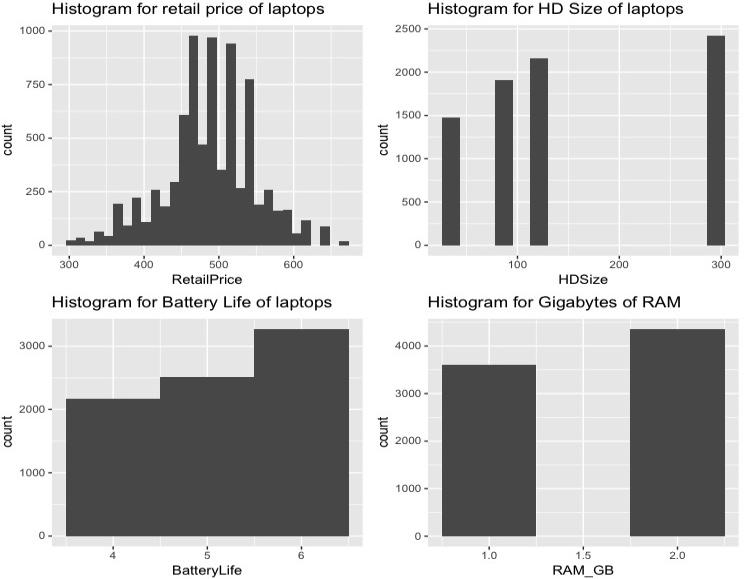
>>display2 <- ggplot(Laptops, aes(x=HDSize)) + geom\_histogram(bins=16) + ggtitle("Histogram for HD Size of laptops")

>>display3 <- ggplot(Laptops, aes(x=BatteryLife)) + geom\_histogram(bins=3) + ggtitle("Histogram for Batter Life of laptops")

>>display4 <- ggplot(Laptops, aes(x=RAM\_G)) + geom\_histogram(bins=3) + ggtitle("Histogram for RAM of laptops")

>>grid.arrange(display1,display2,display3,display4, nrow = 2)

**Results:**

****

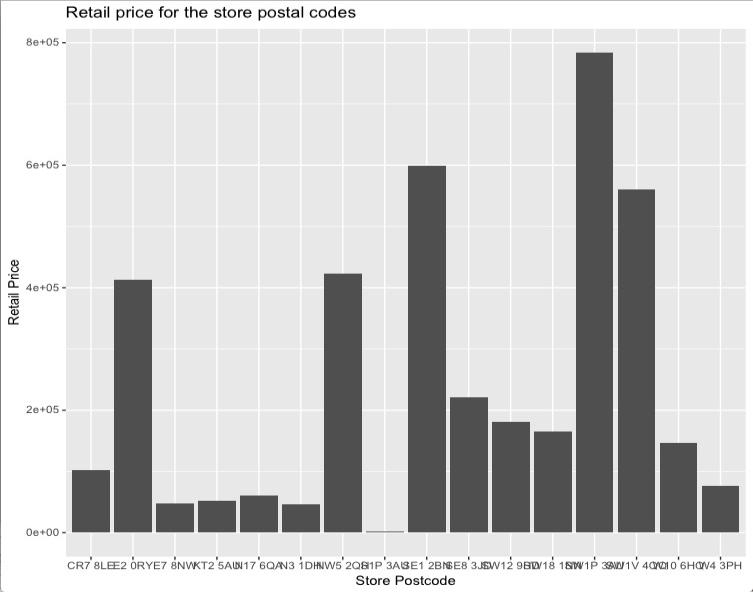
**Data Analysis: Column Charts**

Column chart is plotted to show aggregated retail price for store postal codes using LaptopSalesJanuary2008.csv dataset

**Code:**

ggplot(retail\_Postcode\_agg, aes(x = Store.Postcode, y = RetailPrice)) + geom\_col() + ggtitle("Retail price for the store postal codes") + xlab("Store Postcode") + ylab("Retail Price")

**Results:**

****

**Data Analysis: Scatterplot**

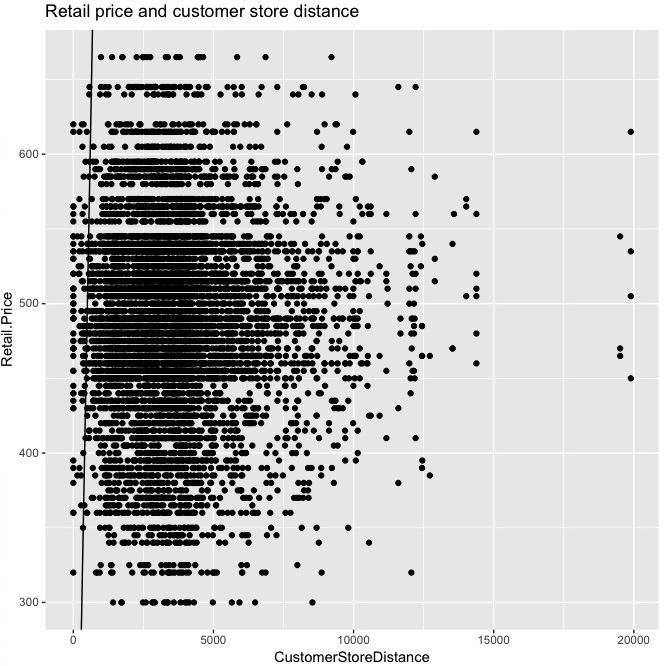
Scatterplot is displayed to show the relationship between retail price and consumer store distance, extracted from LaptopSalesJanuary2008.csv dataset, using a trend line.

**Conclusion to the relationship:** There was no correlation between retail price and customer store distance as a result retail price was not dependent on customer store.

**Code:**

ggplot(Laptops, aes(x = CustomerStoreDistance, y = Retail.Price)) + geom\_point() + ggtitle("Retail price and customer store distance") + geom\_abline()

**Results:**

****

Scatterplot displays the relationship between retail price and customer store distance with the data points shaded representing battery life in hours (i.e. darker data points refer to higher battery life)

**Conclusion:** There is a positive correlation between battery life and retail price of laptops

**Code:**

ggplot(Laptops, aes(x = CustomerStoreDistance, y = Retail.Price)) + geom\_point(aes(color = Battery.Life..Hours.)) + scale\_color\_gradient(low="black", high="red") + ggtitle("Retail price and customer store distance")

**Results:**

****

**Data Analysis: Boxplot**

A boxplot of retail price for all the store post codes showing outliers displayed in red using LaptopSalesJanuary2008.csv dataset

**Code:**

>>Store\_Postcode <- Laptops$Store.Postcode

>>RetailPrice <- Laptops$Retail.Price

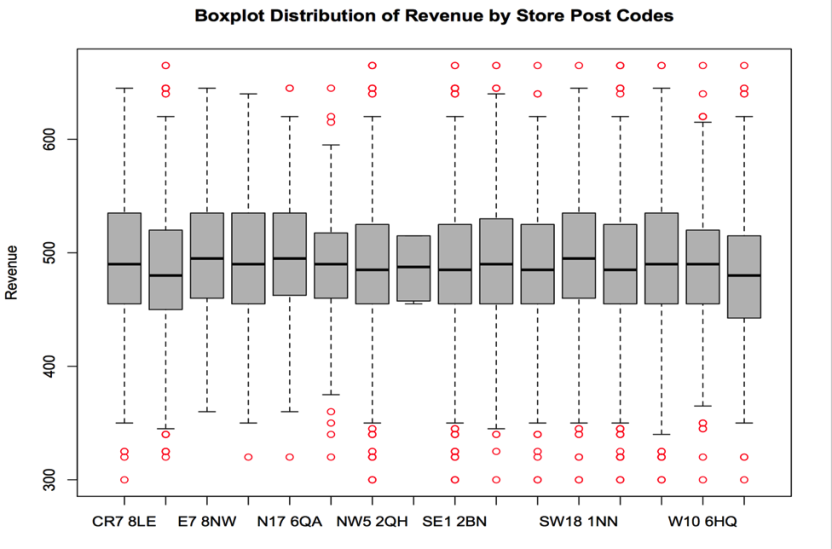
>>boxplot(RetailPrice)

>>boxplot(RetailPrice, ylab = ‘Revenue’, main = 'Boxplot Distribution of Revenue of Store Post Codes')

>>boxplot(RetailPrice~Store\_Postcode, ylab='Revenue', main='Boxplot Distribution of Revenue by Store Post Codes')

>>boxplot(RetailPrice~Store\_Postcode, ylab='Revenue', main='Boxplot Distribution of Revenue by Store Post Codes', col="grey", pars=list(outcol="red"))

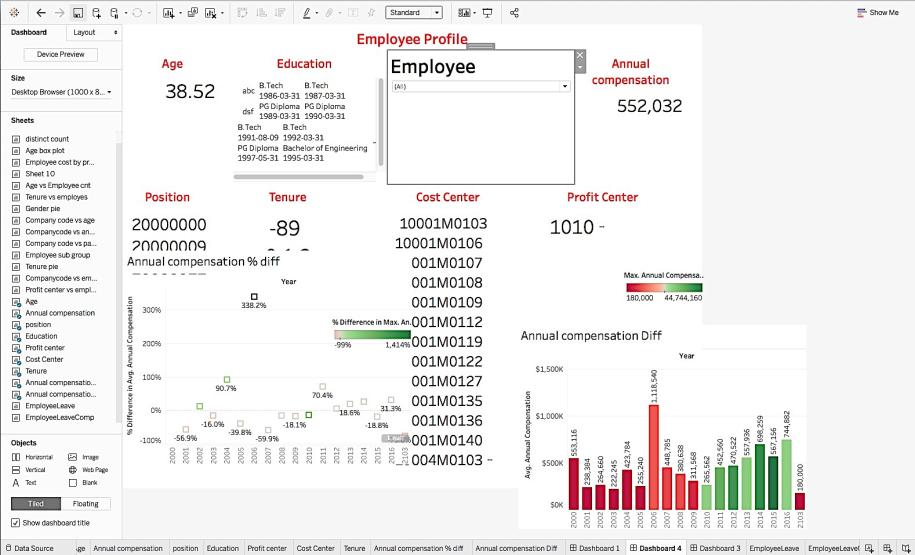
**Results:**

****

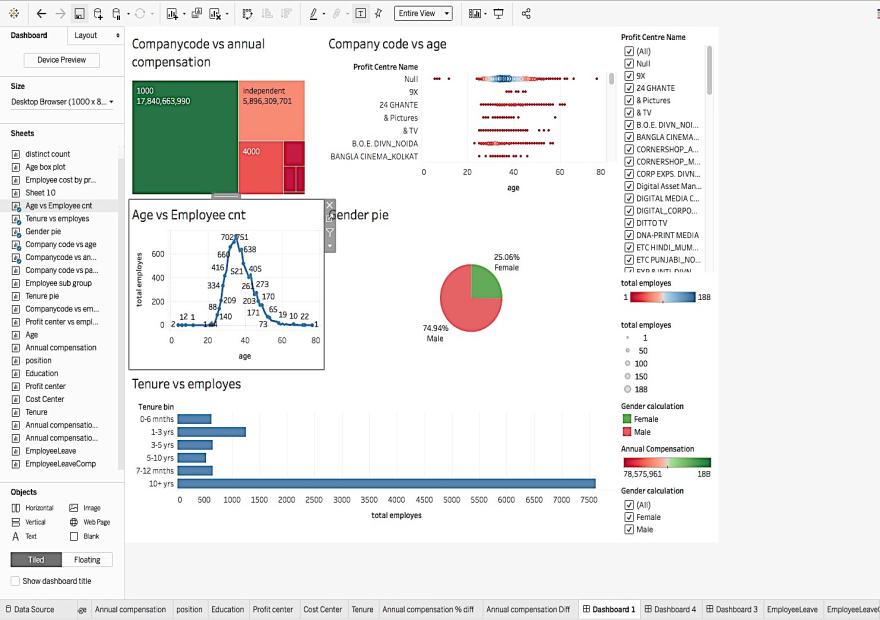
**Business analytics with Tableau**

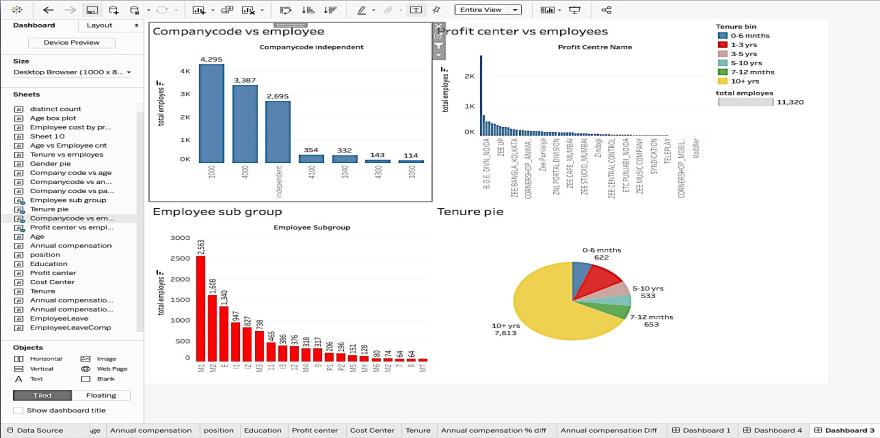
**Dashboard of Employee Profile**

Data analysis based on data collected on employee’s education, age, employee ID annual compensation, profit center, cost center, tenure, and position.



**Dashboard of Employee based on economic, demographic, and company factors**





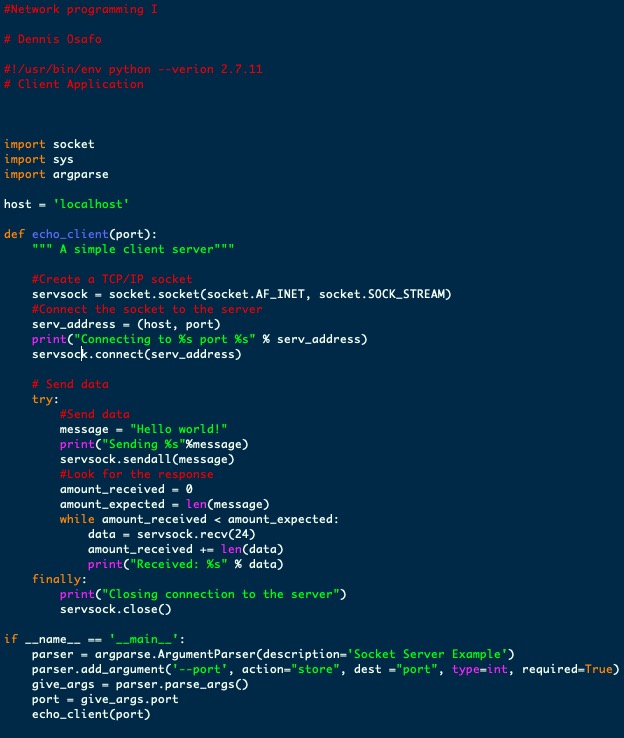
**Echo Application to measure TCP Performance**

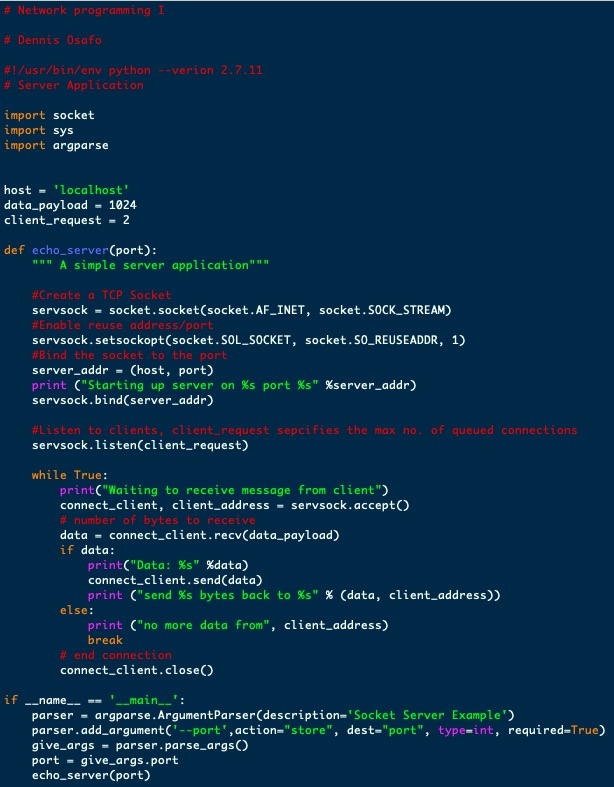
**Echo Client-Server Application**

**Overview:** Implement a client and a server that communicate over a network using TCP. The server is essentially an echo server, which simply echoes the message it receives from the client. Here is what the server and client application must accomplish:

* The server should accept, as a command line argument, a port number that it will run at. After being started, the server should repeatedly accept an input message from a client and send back the same message
* The client should accept, as command line arguments, a host name (or IP address), as well as a port number for the server. Using this information, it creates a connection (using TCP) with the server, which should be running already. The client program then sends a message (text string) to the server using the connection. When it receives back the message, it prints it and exits.

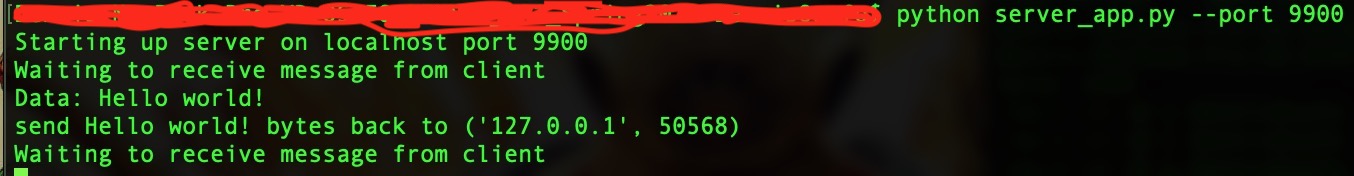
**Client application source code(client\_app.py)**

****

**Server application source code (server\_app.py)**

**Client requests**

****

**Server response**

**Performing RTT and Throughput Measurements**

**Overview:** We are extending the echo client-server application to measure the round trip time (RTT) and throughput of the path connecting the client to the server. To measure RTT, you will use TCP to send and receive messages of size 1, 100, 200,400, 800 and 1000 bytes. To measure throughput, you will use TCP to send and receive messages of size 1K, 2K, 4K, 8K, 16K and 32K bytes. For each measurement and for each message size, the client will send at least ten probe messages to the server, which will echo back the messages.

***For the purpose of this work sample would only use 1,000 bytes to measure RTT and 2K to measure throughput.***

**Protocol phases:** The echo application will be extended, however, by specifying the exact protocol interactions between the client and the server. This entails specifying the exact message formats, as well as the different communication phases, as outlined next.

1. **Connection Setup Phase (CSP):** This is the first phase in the protocol where the client informs the server that it wants to conduct active network measurements in order to compute the RTT and throughput of its path to the server.
   1. **CSP: Client -** After setting up a TCP connection to the server, the client must send a single message to the server having the following format:

**<PROTOCOL PHASE><WS><M−TYPE><WS><MSG SIZE><WS><PROBES><WS><SERVER DELAY>\n**

* **PROTOCOL PHASE:** The protocol phase during the initial setup will be denoted by the lower case character ‘s’. This allows the server to differentiate between the different protocol phases that the client can be operating at, as we will see.
* **M-TYPE (Measurement Type):** Allows the client to specify whether it wants to compute the RTT, denoted by “RTT”, or the throughput, denoted by “TTPUT”.
* **MSG SIZE (Message Size):** Specifies the number of bytes in the probe’s payload
* **PROBES:** Allows the client to specify the number of measurement probes that the server should expect to receive. Once all the probe messages have been echoed back and a sample measurement is taken for each one, the client should compute an estimate of the mean (average) RTT or mean throughput, depending on the type of measurement being performed.
* **SERVER DELAY:** Specifies the amount of time that the server should wait be-fore echoing the message back to the client. The default value should be 0. You will vary this value later to emulate paths with longer propagation delays. Even though increasing the server delay merely increases the processing time at the server, it nevertheless causes the feedback delay, observed by the sender, to in-crease, which has an effect somewhat similar to increasing the path’s propagation delay.
* **WS:** A single white space to separate the different fields in the message. The white space could serve as a delimiter for the server when parsing or tokenizing the received message.
* The last is a new line character that indicates the end of the message.
  1. **CSP: Server** - The server should parse the connection setup message to log the values of all the variables therein since they will be needed for error checking purposes. Upon the reception of a valid connection setup message, the server should respond with a text message containing the string “200 OK: Ready” informing the client that it can proceed to the next phase. On the other hand, if the connection setup message is incomplete or invalid, the server should respond with a text message containing the string “404 ERROR: Invalid Connection Setup Message” and then terminate the connection.
  2. **CSP: Summary** - During correct operation, after setting up a TCP connection to the server, the client sends a single connection setup message to the server. The server parses and logs the information in the message and responds with a “200 OK: Ready ”text message informing the client to proceed to the next phase.

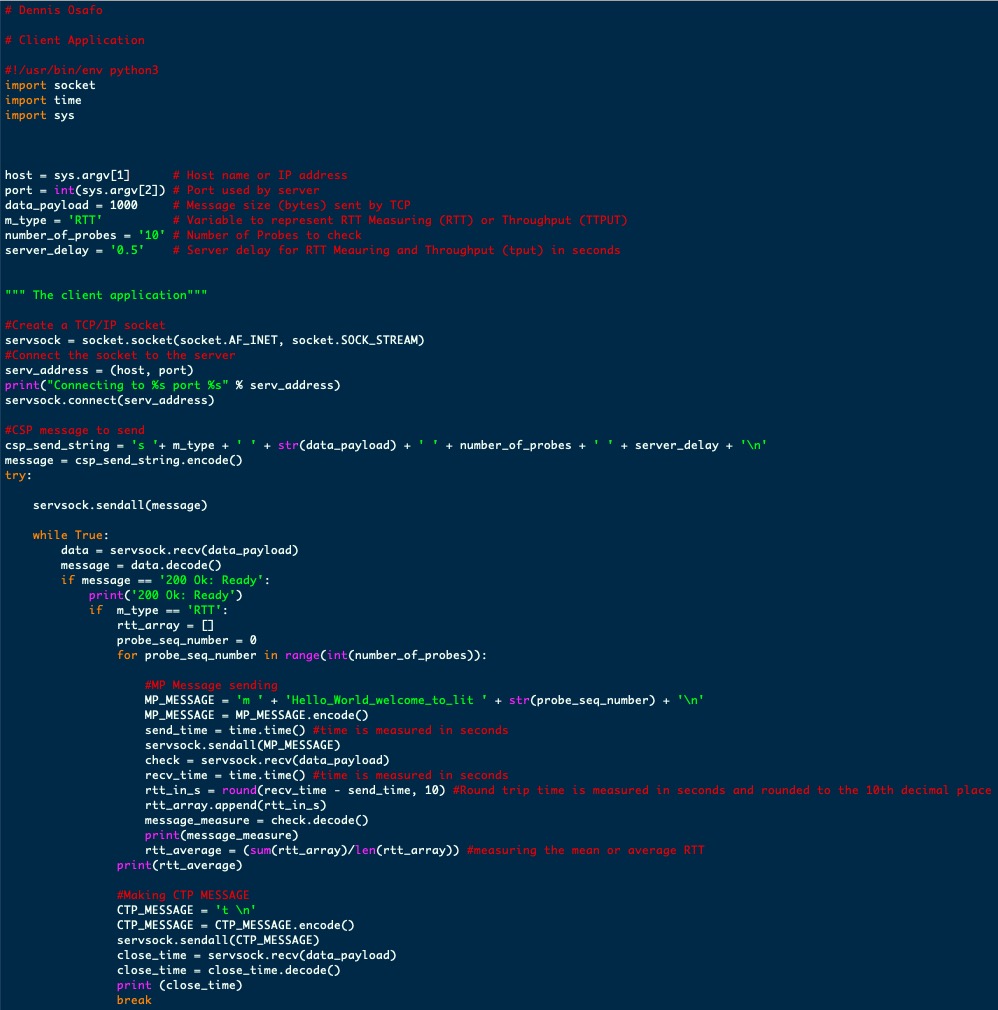
1. **Measurement Phase (MP)** - In this phase, the client starts sending probe messages to the server in order to make the appropriate measurements required for computing the mean RTT or the mean throughput of the path connecting it to the server.
   1. **MP: Client -**The client should send the specified number of probe messages to the server with an increasing sequence number starting from 1. More specifically, the message format is as follows:<PROTOCOL PHASE><WS><PAYLOAD><WS><PROBE SEQUENCE NUMBER>\n

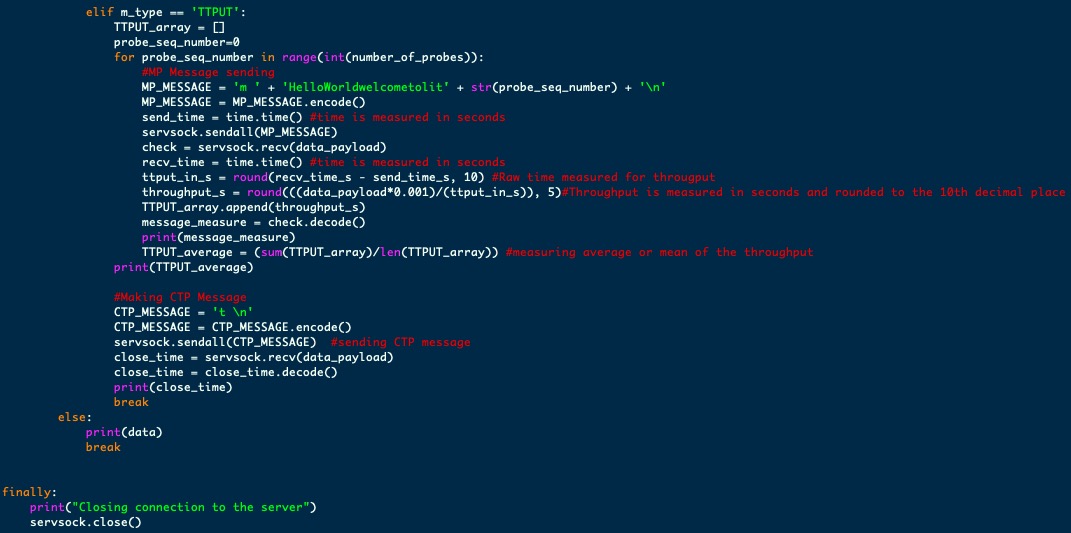
* **PROTOCOL PHASE**: The protocol phase when conducting the measurements will be denoted by the lower case character ‘m’.
* **PAYLOAD:** This is the probe’s payload and can be any arbitrary text whose size was specified in the connection setup message using the MESSAGE SIZE (MESS\_SIZE) variable.
* **PROBE SEQUENCE NUMBER:** The probe messages should have increasing sequence numbers starting from 1 up to the number of probes specified in the connection setup message using the NUMBER OF PROBES variable.
  1. **MP: Server-** the server should echo back every probe message received. It should also keep track of the probe sequence numbers to make sure they are indeed being incremented by 1 each time and do not exceed the number of probes specified in the connection setup phase. If the probe message is incomplete or invalid (contains an incorrect sequence number for example) the server should not echo the message back. Instead, the server should respond with a text message containing the string “404ERROR: Invalid Measurement Message” and then terminate the connection
  2. **MP: Summary-** The client repeatedly sends measurement messages to the server in an attempt to compute the mean RTT and/or mean throughput. A sample measurement is taken for each probe sent out. The server repeatedly echoes messages back to the client unless it detects erroneous behavior in which case it sends an error message and terminates the connection.

1. **Connection Termination Phase (CTP).** In this phase, the client and the server attempt to gracefully terminate the connection. We will outline the expected behavior from both the client and the server next.
   1. **CTP: Client** - The client should send a termination request to the server and then wait for a response (unless of course the server already terminated the connection due to an error in the Measurement Phase). Once a response is received, the client should terminate the connection. The message format is as follows: <PROTOCOL PHASE>\n

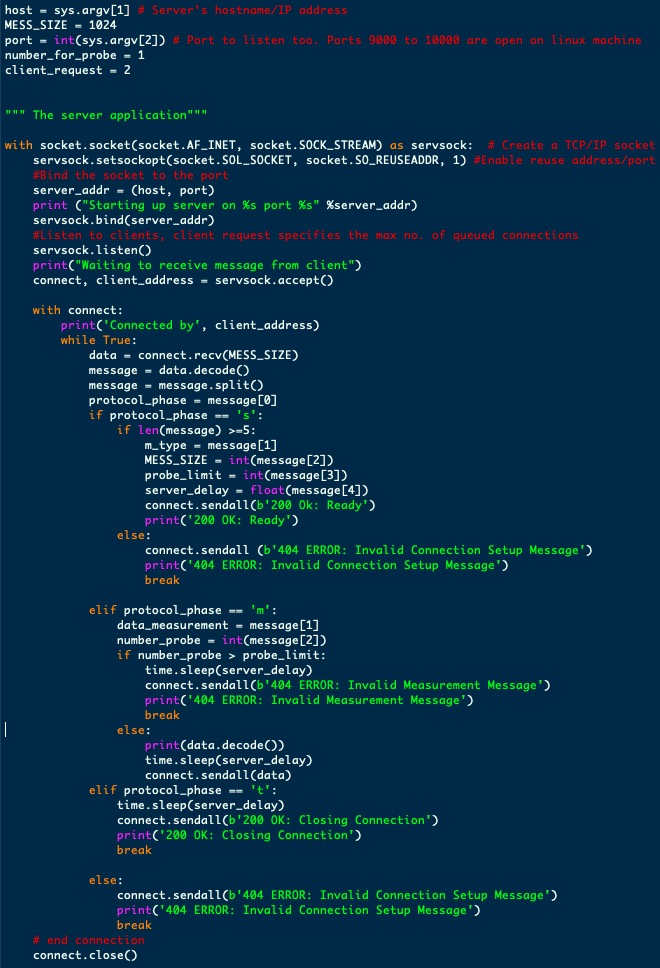
* PROTOCOL PHASE: The protocol phase when terminating the connection will be denoted by the lower case character ‘t’
  1. **CTP: Server -** If the message format is correct, the server should respond with a text message containing the string “200 OK: Closing Connection”. Otherwise, the server should respond with a text message containing the string “404 ERROR: In-valid Connection Termination Message”. Either way the server should terminate the connection.
  2. **CTP: Summary** - During correct operation, the client sends a termination message, the server responds with “200 OK: Closing Connection” text message and then both terminate the connection.

**Client application (source code)**

****

****

**Server application (source code)**

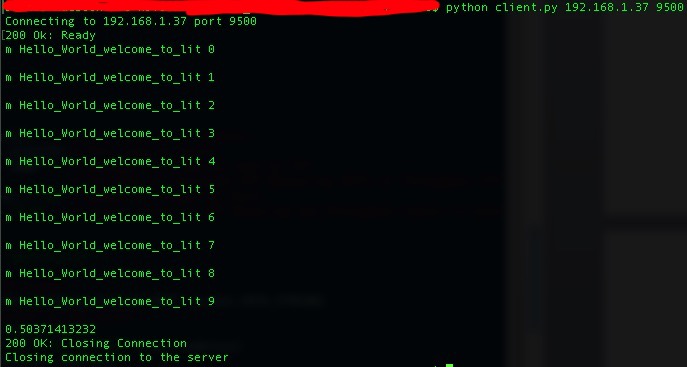


RTT (round trip time) Measurement - client-server application

data\_payload (message size) = 1000 bytes

server\_delay = ‘0.5’ for RTT

**Client results**



**Server results**

