Programming Project 1

**Task 1:**

I will be using **Python** for this project and for all subsequent projects throughout this course.

**Task 2:**

In Python, the threading module provides a very simple and intuitive API for spawning multiple threads in a program. Let us consider a simple example using a threading module:

**import** threading

**def** print\_cube(num):

    print("Cube: {}" .format(num **\*** num **\*** num))

**def** print\_square(num):

**print**("Square: {}" .format(num **\*** num))

**if** \_\_name\_\_ **==**"\_\_main\_\_":

    t1 **=** threading.Thread(target**=**print\_square, args**=**(10,))

    t2 **=** threading.Thread(target**=**print\_cube, args**=**(10,))

    t1.start()

    t2.start()

t1.join()

    t2.join()

    print("Done!")

In the above code,

import threading: To import the threading module.

target: the function to be executed by thread.

args: the arguments to be passed to the target function.

Start(): To start the thread.

Join(): for synchronization.

As a result, the current program will hold off till t1 and subsequently t2 have finished. The remaining statements of the running program are then executed once they have ended.

**Task 3:**

**Inter process Communication:**

A server is a software that monitors client requests and responds appropriately. A client, on the other hand, requests this service. A client software asks the server for certain resources, and the server fulfills the request. The socket is where a server and client's two-way communication link ends. Sockets may communicate between processes on the same machine, between processes on separate machines, or within a single process. We must establish a connection through a socket port to communicate with any distant software.

 The steps can be considered like this.

1. Python socket server program will execute at first and wait for the request
2. Python socket client program will initiate the conversation at first.
3. Then server program will respond accordingly to client requests.
4. Client program will terminate if user enters “bye” message. Server program will also terminate when client program terminates.

**Python socket server:**

We will save python socket server program as socket\_server.py. To use python socket connection, we need to import socket module. Then, we need to perform some task to establish connection between server and client. We can obtain host address by using socket.gethostname() function. It is recommended to user port address above 1024 because port number lesser than 1024 are reserved for standard internet protocol.

**Python Socket Client:**

We will save python socket client program as socket\_client.py. This program is like the server program, except for binding. The difference between server and client program is, in server program, it needs to bind host address and port address together.

In this project, we use,

.socket(): To instantiate.

.bind(): To bind host address and port together.

.accept(): To accept the connection.

.send(): To send data

.recv(): To receive the data

.connect(): To connect to the server.

.close(): To close the connection.

Task 4. Implementing the Client

**Python socket Client:**

**Text

Description automatically generated**

Task 5. Implementing the Server

**Python socket server:**

**Text

Description automatically generated**

Task 6. Documentation

Steps to check the output,

1.Run the socket server program.

2.Then run the socket client program.

3.Write something from client program in client console.

4.Then again write reply from server program in server console.

5.At last, write **bye** from client program to terminate both server and client program.

Server program output:Text

Description automatically generated

Client Program Output:

Text

Description automatically generated