Computer Network 1

Lab 4

Socket Programming in Python: Chat Application

Names:………………………………………………..

Student No: ………………………………………..

1 Objectives

1. Practice with Socket programming in Python.
2. Build a simple chat application using client-server model.
3. Multithreaded application in Python.

2 Content

2.1 Socket programming in Python

Consider the following code:

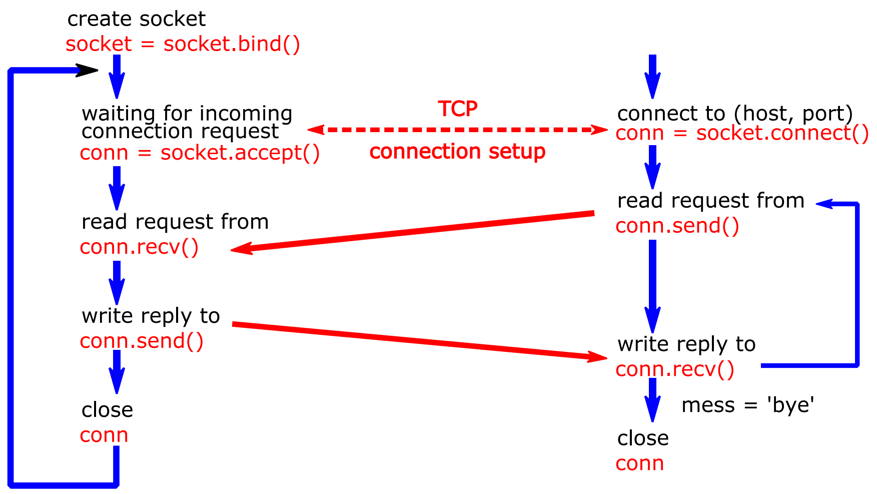
```
import socket

// Get hostname by textual representation of IP address
addr = socket.gethostbyname("127.0.0.1")
// Get hostname by a byte array containing the IP address
addr = '127.0.0.1'
// Get the host name
addr = socket.gethostname()
```

Exercise 1:

Create a program that connects to a web server and downloads the homepage of this website to local computer

Ex: You can test your program with: www.google.com

2.2 *Develop a simple chat application using client-server model*



Server application	Client application
Create a server socket	
Listen and wait for incoming connection request	Identify server IP and port
	Create a TCP socket to the server
	Establish a connection to the server
Accept the connection request	
Send/Receive data	Send/Receive data
	Close connection with message 'bye'
Close connection	

Exercise 2:

Design the user interface for the chat application

2.3 *Multithread in Python*

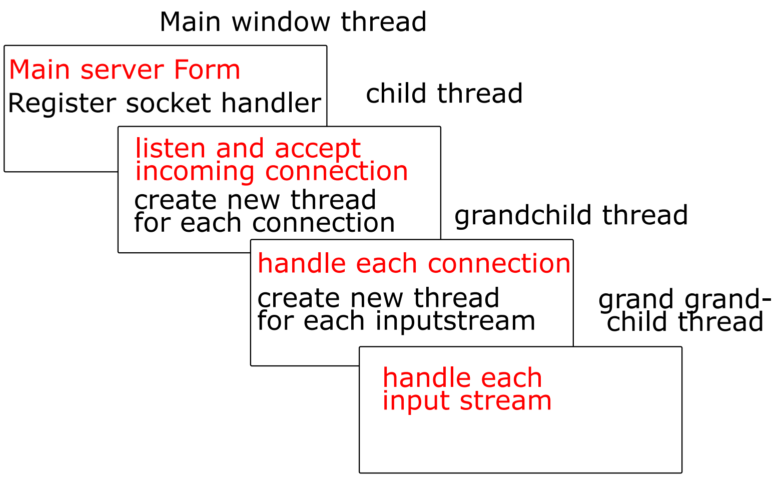
The Thread class with support the return value

```
/* Try to use the Thread with supporting return value */
from threading import Thread


class ThreadWithReturn(Thread):

    def __init__(self, group=None, target=None, name=None,
                 args=(), kwargs={}, Verbose=None):
        Thread.__init__(self, group, target, name, args, kwargs)
        self._return = None   #  parameter for later return value

    #  Override the invoker run() to pass callable object
    def run(self):
        if self._target is not None:
            self._return = self._target(*self._args,
                                        **self._kwargs)

    #  Override value passing
    def join(self, *args):
        Thread.join(self, *args)
        return self._return
```

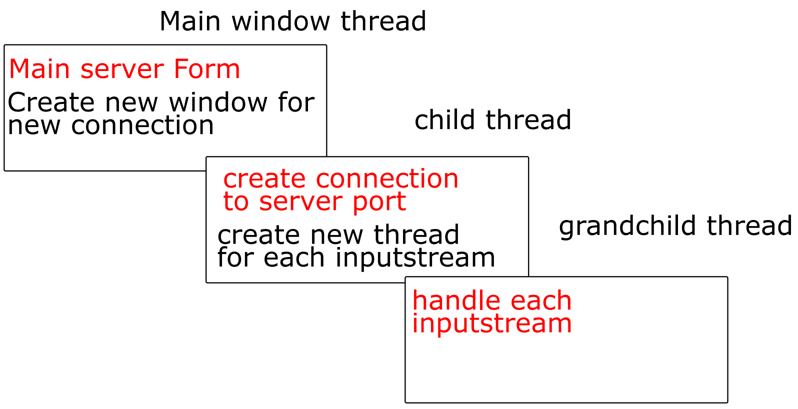
Create an application that has multiple threads running concurrently:

```
from threading import Thread

def register_port(hostname, port_num=5000):
    #  Child thread
    #  Register socket handler


...
binder = Thread(target=register_port, args=(host, port))
binder.start()
```

Multithread in Server Application:



Multithread in Client Application


Main window thread
Main server Form
Create new window for new connection
child thread
create connection to server port
create new thread for each inputstream
grandchild thread
handle each inputstream

Exercise 3:

Using multithread programming model to make the chat application can talk to many different users concurrently.

/* Hint: Passing the connection object and handling the event terminate/return may help in managing the connection session. This usage is an illustration of ThreadWithReturn */

```
from threading import Thread
from threadwithreturn import ThreadWithReturn as RetThread
```

```
def recv_input_stream(conn, addr):
    #  Grand grandchild thread
    #  Handle the input stream



def accept_connection(connection, address):
    #  Grandchild thread
    #  Create new thread for each input stream
    while True:
        input_stream = RetThread(target=recv_input_stream, args=(connection, 
address,))
        input_stream.start()
        val = input_stream.join()
        if val == -1:
            return
```