CS3031 Project 1

A Web Proxy Server

David Scolard 16322277

The objective of the exercise is to implement a Web Proxy Server. A Web proxy is a local server, which fetches items from the Web on behalf of a Web client instead of the client fetching them directly. This allows for caching of pages and access control.

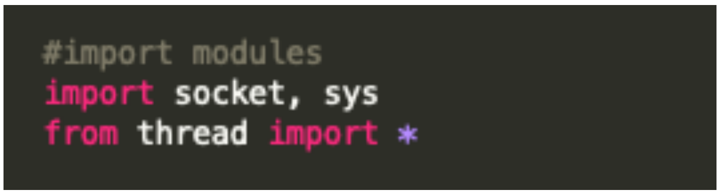
Intro

Proxy servers are extremely useful tools for managing connections between the web server and the client. They can be used for many things including caching frequently accessed pages as well as access control to certain sites.

To implement my proxy server I decided to write it in python. After looking at multiple languages I decided it had the most intuitive socket programming libraries which could help me in this project. Throughout this report I have supplied my python code which is heavily commented explaining what each step accomplishes.

Libraries

First step is to import the necessary modules for my proxy server program:

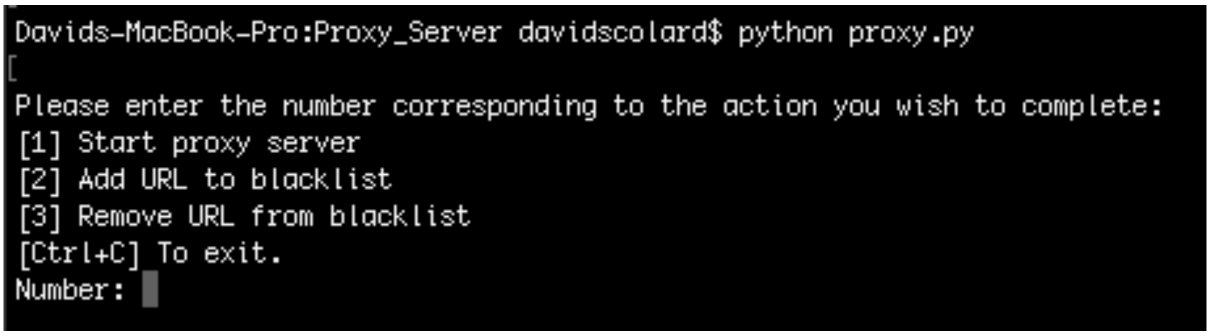


```
#import modules
import socket, sys
from thread import *
```

The socket library makes working with sockets extremely manageable with simple functions such as socket.bind(port_number), to bind your socket to a desired port, or socket.listen(connections) to start your socket listening for incoming data etc. The Thread module is necessary so that we can implement a threaded server which is capable of handling multiple requests simultaneously with ease.

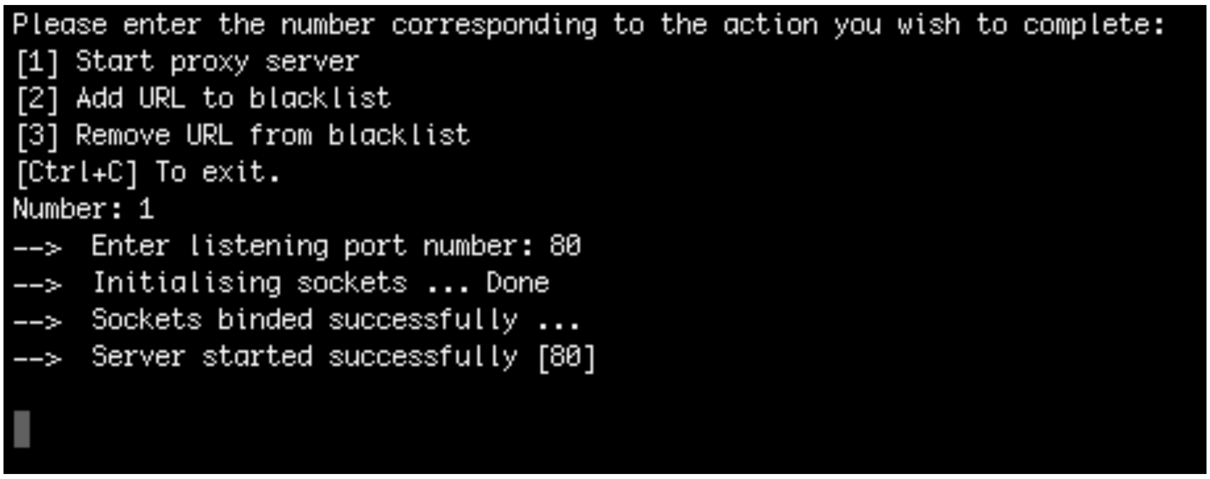
Management Console

Next I set up my management system, this system allows the user to start the server as well as allowing them to add and remove URL’s to the blacklist:



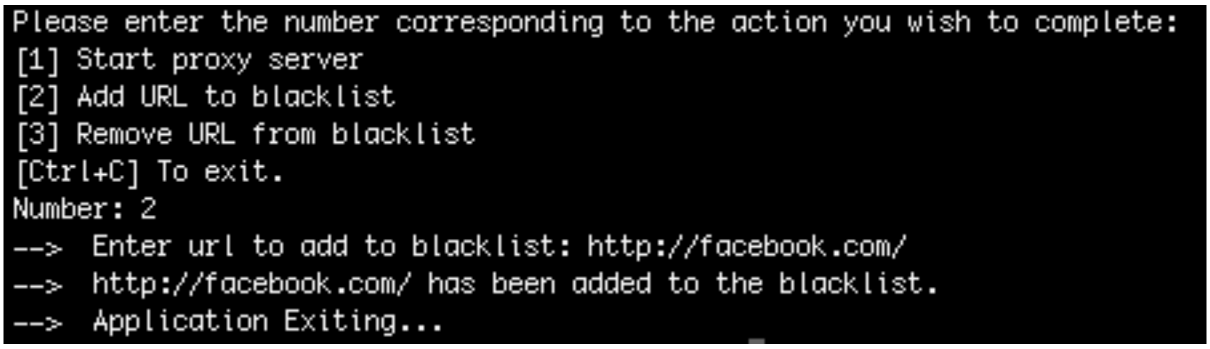
```
Davids-MacBook-Pro:Proxy_Server davidscolard$ python proxy.py
[
Please enter the number corresponding to the action you wish to complete:
[1] Start proxy server
[2] Add URL to blacklist
[3] Remove URL from blacklist
[Ctrl+C] To exit.
Number:
```

The user simply chooses by entering the corresponding number of what they want to do. For example entering 1 will prompt you to first enter the port number on which you wish to start the server before proceeding to start it up and connect the sockets for use:



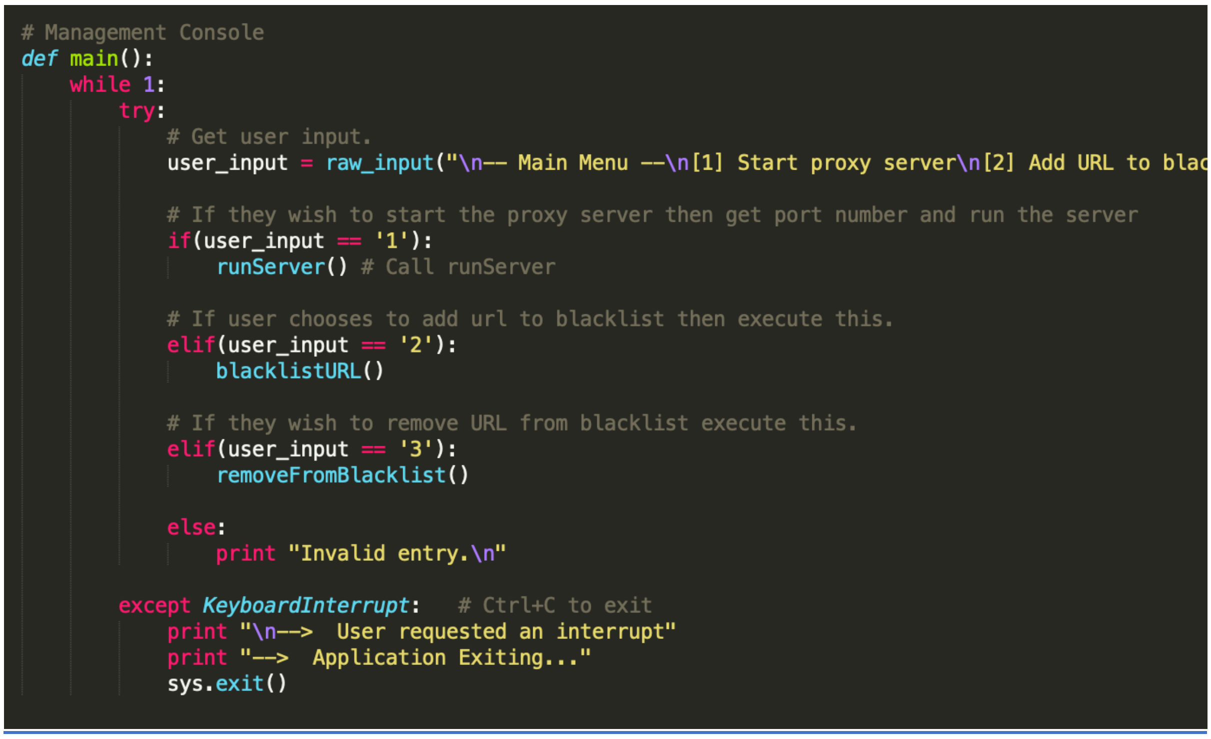
```
Please enter the number corresponding to the action you wish to complete:
[1] Start proxy server
[2] Add URL to blacklist
[3] Remove URL from blacklist
[Ctrl+C] To exit.
Number: 1
-->  Enter listening port number: 80
-->  Initialising sockets ... Done
-->  Sockets binded successfully ...
-->  Server started successfully [80]
```

Alternatively if you enter 2 or 3 you will be asked for a URL to add to or delete from the blacklist:



```
Please enter the number corresponding to the action you wish to complete:
[1] Start proxy server
[2] Add URL to blacklist
[3] Remove URL from blacklist
[Ctrl+C] To exit.
Number: 2
-->  Enter url to add to blacklist: http://facebook.com/
-->  http://facebook.com/ has been added to the blacklist.
-->  Application Exiting...
```

Code for the management console:

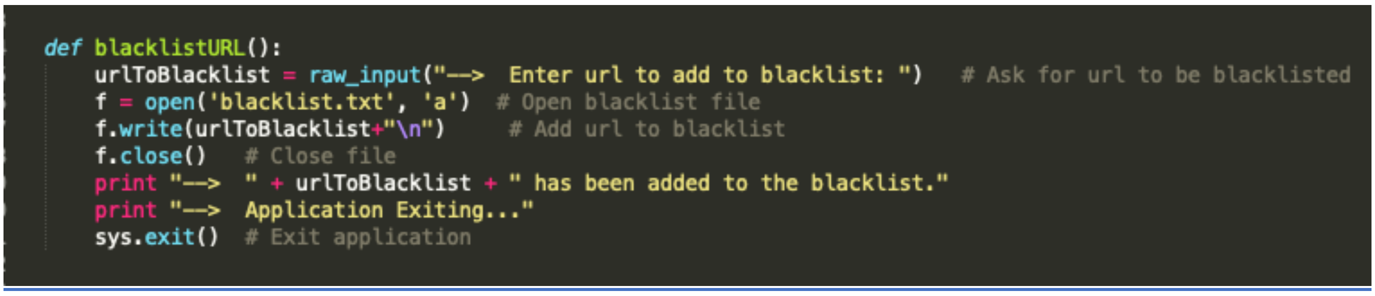


```
# Management Console
def main():
    while 1:
        try:
            # Get user input.
            user_input = raw_input("\n-- Main Menu --\n[1] Start proxy server\n[2] Add URL to blac

            # If they wish to start the proxy server then get port number and run the server
            if(user_input == '1'):
                runServer() # Call runServer

            # If user chooses to add url to blacklist then execute this.
            elif(user_input == '2'):
                blacklistURL()

            # If they wish to remove URL from blacklist execute this.
            elif(user_input == '3'):
                removeFromBlacklist()

            else:
                print "Invalid entry.\n"

        except KeyboardInterrupt:   # Ctrl+C to exit
            print "\n-->  User requested an interrupt"
            print "-->  Application Exiting..."
            sys.exit()
```

Here you can see this was implanted using simple if statements. And calling the appropriate function then when the user selects an option.

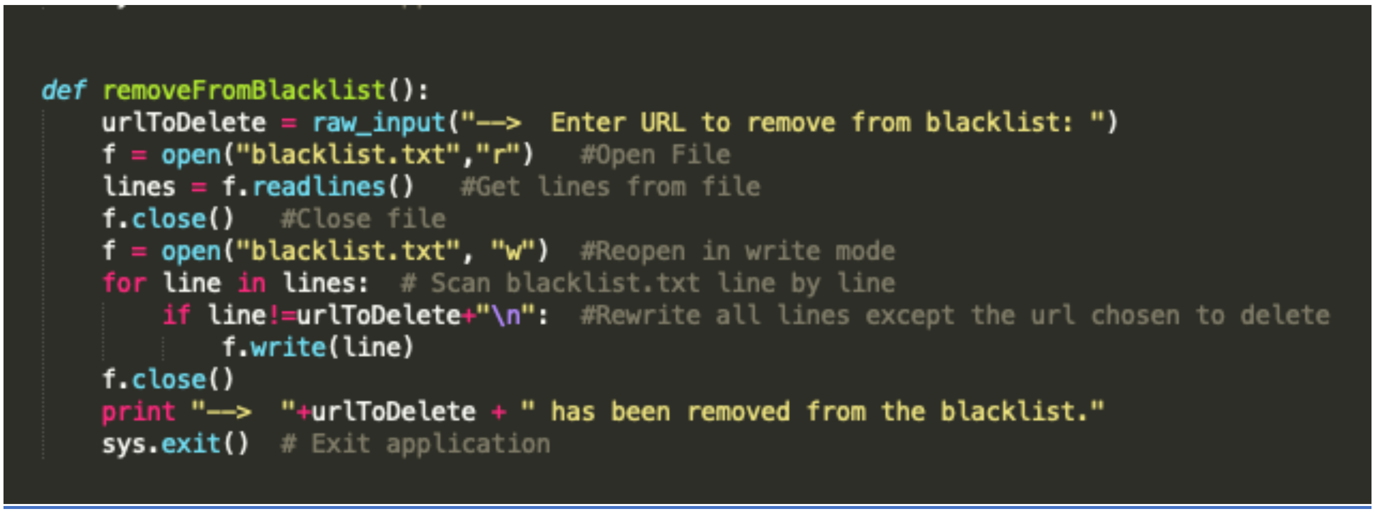
Blacklist

The blacklist is a text file with a line separating each URL. To add to the blacklist we simply ask the user to enter URL they wish to block and we append it to the blacklist.



```
def blacklistURL():
    urlToBlacklist = raw_input("--> Enter url to add to blacklist: ")   # Ask for url to be blacklisted
    f = open('blacklist.txt', 'a')  # Open blacklist file
    f.write(urlToBlacklist+"\n")     # Add url to blacklist
    f.close()   # Close file
    print "--> " + urlToBlacklist + " has been added to the blacklist."
    print "--> Application Exiting..."
    sys.exit()  # Exit application
```

For removing URL’s we read the blacklist line by line. We rewrite each line in the file back into the blacklist as long as it doesn’t match the URL which we want to remove.



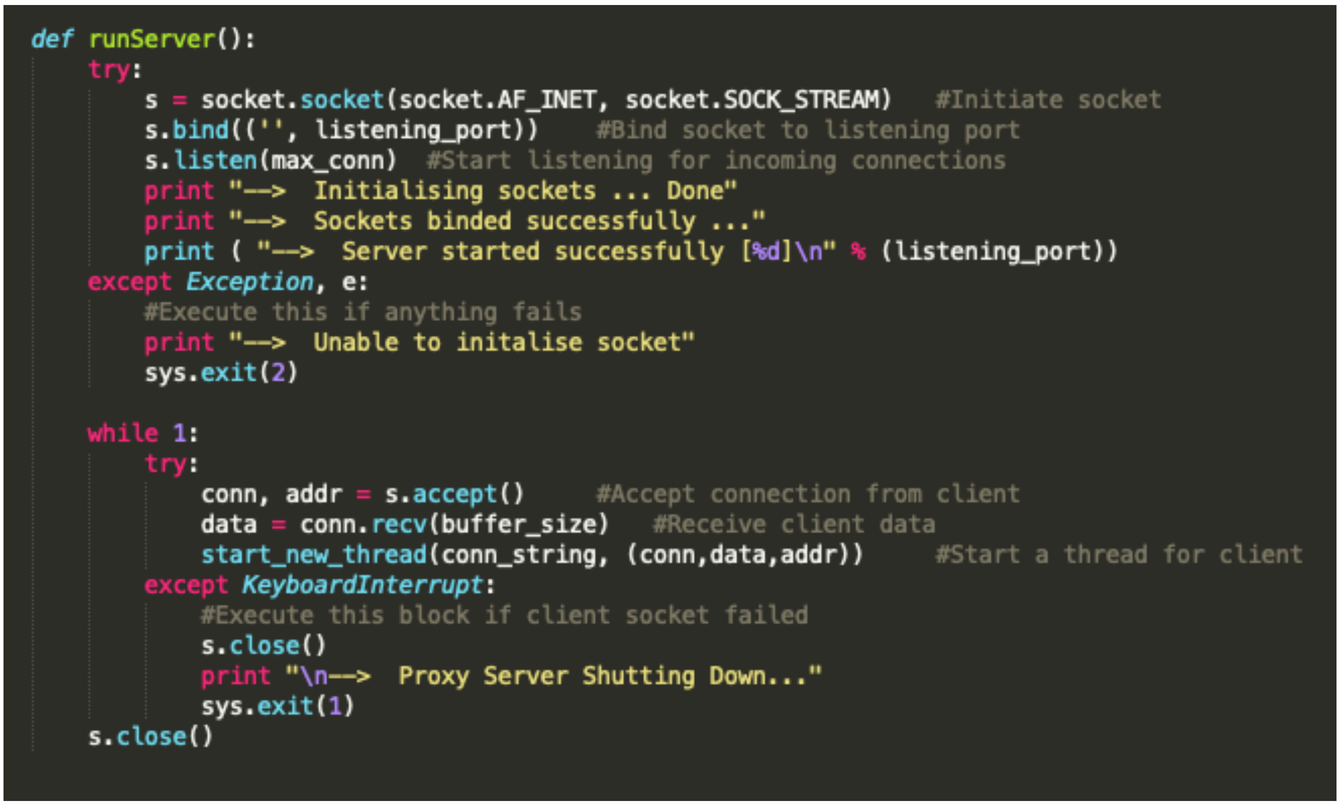
```
def removeFromBlacklist():
    urlToDelete = raw_input("--> Enter URL to remove from blacklist: ")
    f = open("blacklist.txt","r")   #Open File
    lines = f.readlines()   #Get lines from file
    f.close()   #Close file
    f = open("blacklist.txt", "w")  #Reopen in write mode
    for line in lines:  # Scan blacklist.txt line by line
        if line!=urlToDelete+"\n":  #Rewrite all lines except the url chosen to delete
            f.write(line)
    f.close()
    print "--> "+urlToDelete + " has been removed from the blacklist."
    sys.exit()  # Exit application
```

Setting up sockets

Next I had to set up my socket connections. This is done in the runServer() function.
This function:

- Sets up the socket
- Bind the sockets
- Starts listening for incoming connections

If an error occurs then the server will shut itself down.

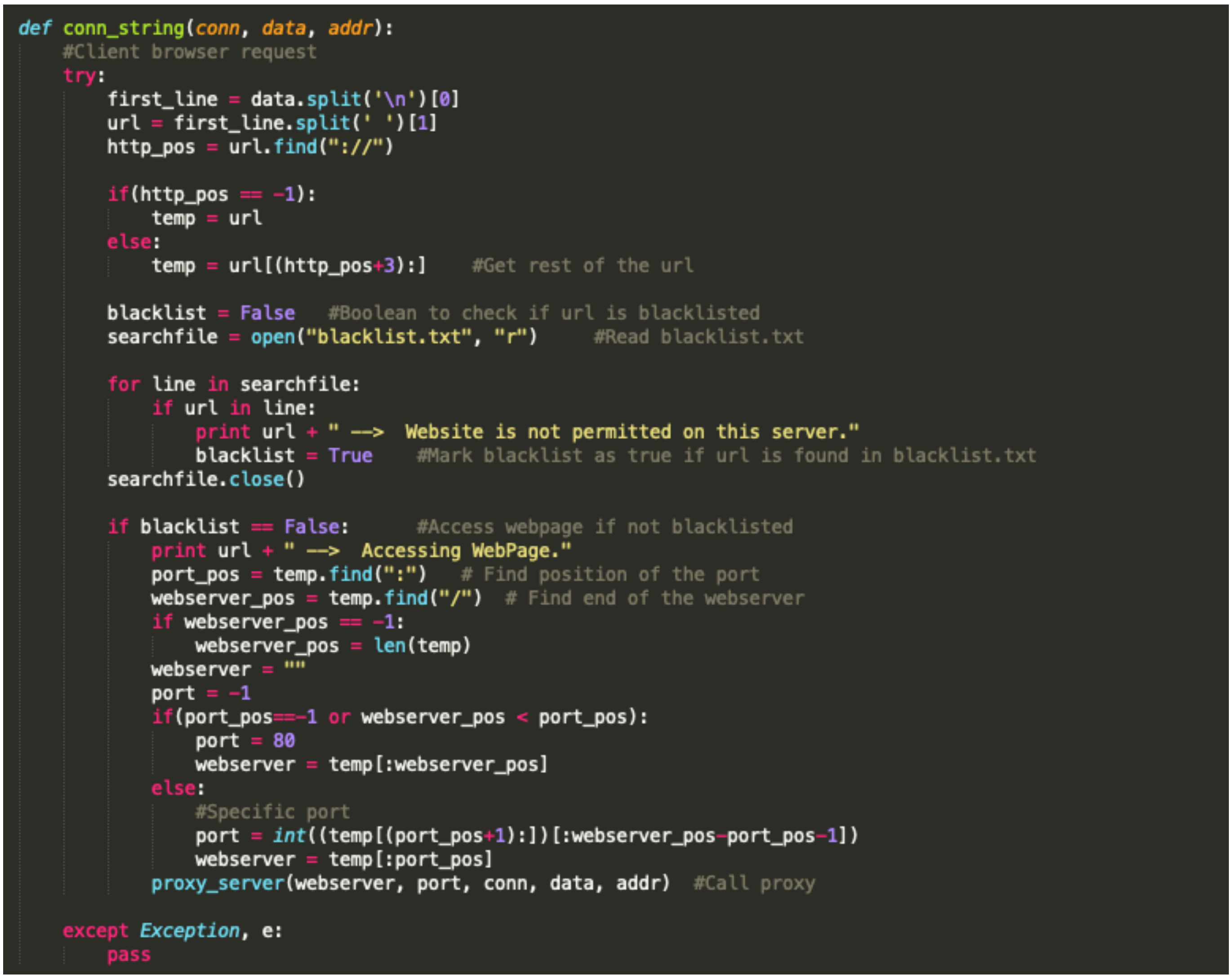


```
def runServer():
    try:
        s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)   #Initiate socket
        s.bind(('', listening_port))    #Bind socket to listening port
        s.listen(max_conn)  #Start listening for incoming connections
        print "--> Initialising sockets ... Done"
        print "--> Sockets binded successfully ..."
        print ( "--> Server started successfully [%d]\n" % (listening_port))
    except Exception, e:
        #Execute this if anything fails
        print "--> Unable to initalise socket"
        sys.exit(2)

    while 1:
        try:
            conn, addr = s.accept()     #Accept connection from client
            data = conn.recv(buffer_size)   #Receive client data
            start_new_thread(conn_string, (conn,data,addr))     #Start a thread for client
        except KeyboardInterrupt:
            #Execute this block if client socket failed
            s.close()
            print "\n--> Proxy Server Shutting Down..."
            sys.exit(1)
    s.close()
```

We see in the above code that when a connection is made we accept it and receive the clients data. We then start a new connection thread with this client.
start_new_thread(conn_string, (conn,data,addr))

Connection and Access Control

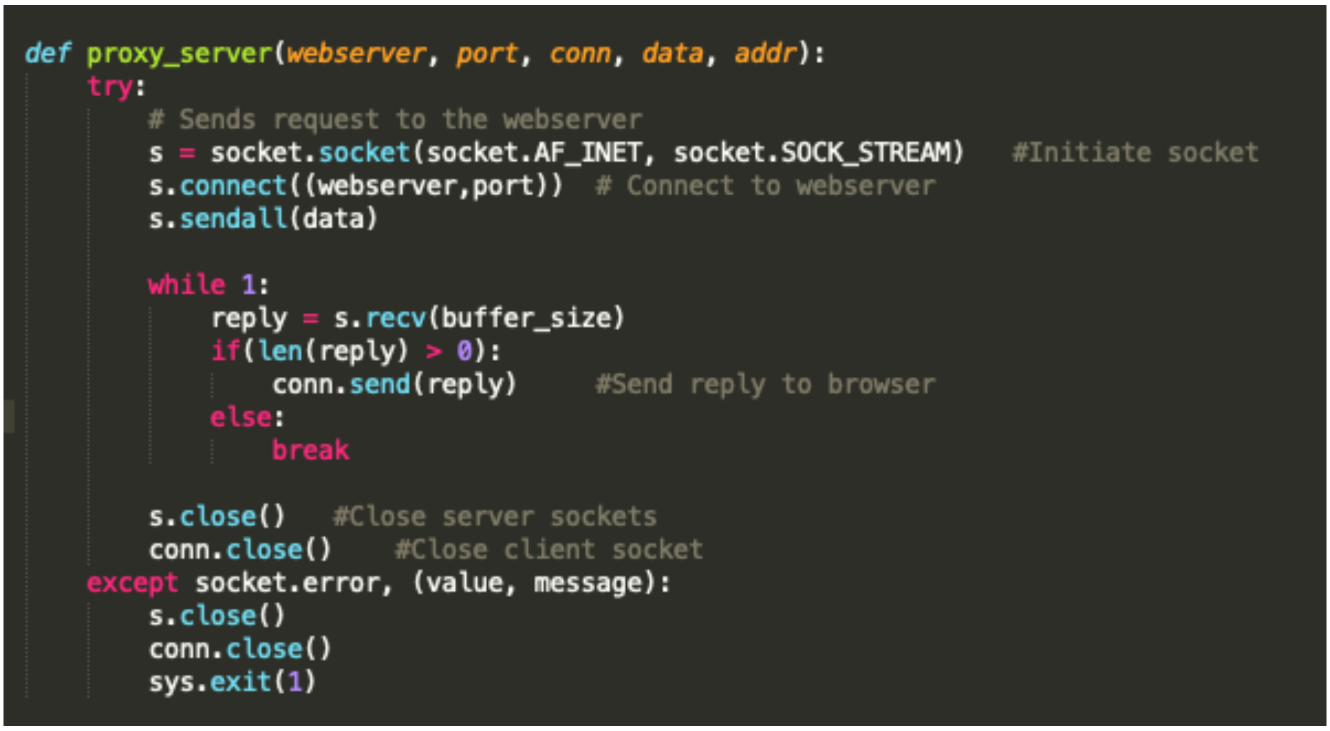
The conn_string() function processes the connection with the browser. It retrieves the host address in the client data and checks it against the blacklist to ensure it is a permitted site on our server. This is done using a single Boolean value which is marked *True* if the URL is found in the blacklist.txt file. If it is not blacklisted then we will pass it to our proxy server:
proxy_server(webserver, port, conn, data, addr)
webserver - host address
port - webserver port
conn - connection stream
addr - address from connection stream.
data - client browser data



```python
def conn_string(conn, data, addr):
    #Client browser request
    try:
        first_line = data.split('\n')[0]
        url = first_line.split(' ')[1]
        http_pos = url.find("://")

        if(http_pos == -1):
            temp = url
        else:
            temp = url[(http_pos+3):]     #Get rest of the url

        blacklist = False    #Boolean to check if url is blacklisted
        searchfile = open("blacklist.txt", "r")      #Read blacklist.txt

        for line in searchfile:
            if url in line:
                print url + " --> Website is not permitted on this server."
                blacklist = True    #Mark blacklist as true if url is found in blacklist.txt
        searchfile.close()

        if blacklist == False:      #Access webpage if not blacklisted
            print url + " --> Accessing WebPage."
            port_pos = temp.find(":")   # Find position of the port
            webserver_pos = temp.find("/")  # Find end of the webserver
            if webserver_pos == -1:
                webserver_pos = len(temp)
            webserver = ""
            port = -1
            if(port_pos==-1 or webserver_pos < port_pos):
                port = 80
                webserver = temp[:webserver_pos]
            else:
                #Specific port
                port = int((temp[(port_pos+1):])[:webserver_pos-port_pos-1])
                webserver = temp[:port_pos]
            proxy_server(webserver, port, conn, data, addr)  #Call proxy

    except Exception, e:
        pass
```

Proxy Server

This function takes in the data regarding the webserver and the host. It then sets up a connection and sends a request to the webserver. It then receives a reply from the webserver and forwards it on to the host without modifying it. Then it closes the streams.



```python
def proxy_server(webserver, port, conn, data, addr):
    try:
        # Sends request to the webserver
        s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)   #Initiate socket
        s.connect((webserver,port))  # Connect to webserver
        s.sendall(data)

        while 1:
            reply = s.recv(buffer_size)
            if(len(reply) > 0):
                conn.send(reply)     #Send reply to browser
            else:
                break

        s.close()   #Close server sockets
        conn.close()    #Close client socket
    except socket.error, (value, message):
        s.close()
        conn.close()
        sys.exit(1)
```

Code:

```
#		CS3031 A Web Proxy Server
#		Name: David scolard
#		Student Number: 16322277


#import modules
import socket, sys, zlib

from thread import *

max_conn = 20		# Max number of connections to proxy server
buffer_size = 8192	# Max socket buffer size
cache = {}


# Management Console
def main():
		while 1:
			try:
				# Get user input.
				user_input = raw_input("\n-- Main Menu --\n[1] Start proxy server\n[2] Add URL to blacklist\n[3]
Remove URL from blacklist\n[Ctrl+C] To exit.\n\nEnter Number: ")

				# If they wish to start the proxy server then get port number and run the server
				if(user_input == '1'):
					runServer() # Call runServer

				# If user chooses to add url to blacklist then execute this.
				elif(user_input == '2'):
					blacklistURL()

				# If they wish to remove URL from blacklist execute this.
				elif(user_input == '3'):
					removeFromBlacklist()

				else:
					print "Invalid entry.\n"

			except KeyboardInterrupt:		# Ctrl+C to exit
				print "\n--> User requested an interrupt"
				print "--> Application Exiting..."
				sys.exit()


def runServer():
		try:
			listening_port = int(raw_input("--> Enter listening port number: ")) #Get port number to run server on
			s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)	#Initiate socket
			s.bind(('', listening_port))		#Bind socket to listening port
			s.listen(max_conn)	#Start listening for incoming connections
			print "--> Initialising sockets ... Done"
			print "--> Sockets binded successfully ..."
			print ( "--> Server started successfully [%d]\n" % (listening_port))
		except Exception, e:
			#Execute this if anything fails
			print "--> Unable to initalise socket"
			sys.exit(2)

		while 1:
			try:
				conn, addr = s.accept()		# Accept connection
				data = conn.recv(buffer_size)	# Receive data
				start_new_thread(conn_string, (conn,data,addr))		#Start a thread for client
			except KeyboardInterrupt:
				#Execute this block if client socket failed
				s.close()
				print "\n--> Proxy Server Shutting Down..."
				sys.exit(1)
		s.close()


def conn_string(conn, data, addr):
		#Client browser request
		try:
```

```
                first_line = data.split('\n')[0]
                url = first_line.split(' ')[1]
                http_pos = url.find("://")

                if(http_pos == -1):
                        temp = url
                else:
                        temp = url[(http_pos+3):]          #Get rest of the url

                blacklist = False       #Boolean to check if url is blacklisted
                searchfile = open("blacklist.txt", "r")         #Read blacklist.txt

                for line in searchfile:
                   if url in line:
                        print url + " -->  Website is not permitted on this server."
                        blacklist = True        #Mark blacklist as true if url is found in blacklist.txt
                searchfile.close()

                if blacklist == False:                  #Access webpage if not blacklisted
                        print url + " -->  Accessing WebPage."
                        port_pos = temp.find(":")           # Find position of the port
                        webserver_pos = temp.find("/")   # Find end of the webserver
                        if webserver_pos == -1:
                                webserver_pos = len(temp)
                        webserver = ""
                        port = -1
                        if(port_pos==-1 or webserver_pos < port_pos):
                                port = 80  # Default port
                                webserver = temp[:webserver_pos]
                        else:
                                #Specific port
                                port = int((temp[(port_pos+1):])[:webserver_pos-port_pos-1])
                                webserver = temp[:port_pos]

                        proxy_server(webserver, port, conn, data)      #Call proxy
        except Exception, e:
                pass


def proxy_server(webserver, port, conn, data):
        try:
                # Sends request to the webserver
                s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)        #Initiate socket
                s.connect((webserver,port))         # Connect to webserver
                s.sendall(data)

                while 1:
                        reply = s.recv(buffer_size)
                        if(len(reply) > 0):
                                conn.send(reply)         #Send reply to browser
                        else:
                                break

                s.close()     #Close server sockets
                conn.close()            #Close client socket
        except socket.error, (value, message):
                s.close()
                conn.close()
                sys.exit(1)


def blacklistURL():
        urlToBlacklist = raw_input("-->  Enter url to add to blacklist: ")       # Ask for url to be blacklisted
        f = open('blacklist.txt', 'a')           # Open blacklist file
        f.write(urlToBlacklist+"\n")         # Add url to blacklist
        f.close()     # Close file
        print "-->  " + urlToBlacklist + " has been added to the blacklist.\n"
        main()


def removeFromBlacklist():
        urlToDelete = raw_input("-->  Enter URL to remove from blacklist: ")
        f = open("blacklist.txt","r")         #Open File
        lines = f.readlines()     #Get lines from file
        f.close()     #Close file
        f = open("blacklist.txt", "w")        #Reopen in write mode
        for line in lines:          # Scan blacklist.txt line by line
                if line!=urlToDelete+"\n":          #Rewrite all lines except the url chosen to delete
```

```
                        f.write(line)
        f.close()
        print "-->  "+urlToDelete + " has been removed from the blacklist.\n"
        main()


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