* Lundi 22 Janvier 2024 :
* Socket:  communication endpoint that serves as sending and receiving data across a network.
* TCP: a protocol assures that every that are in order and received by the client. (Text/image missing, webpages not in order…)
* UDP: same as TCP except that it only sends data and doesn’t guarantee data delivery but it’s faster since it doesn’t check if the data is delivered.
* Proxy server: an intermediary server that retrieve data from the destination server for the client using an IP address other than the client’s address
* Threading library in python: it allows the program to not wait other blocks of code to start.
* Mardi 23 Janvier 2024:
* 0.0.0.0: when the server and the client use the same machine
* Use local IP address if they aren’t
* Creation of server and client program
* Socket AF\_INET corresponds to IPv4
* Socket SOCK\_STREAM corresponds to TCP protocol
* Mercredi 24 Janvier 2024:
* socket.socket(): create an object that can be used for the communication between the server and the client device. (server’s socket)
* Test d’envoi de data vers serveur
* Test si client toujours connecté au serveur (not yet done)
* Jeudi 25 Janvier 2024 :
* Test d’envoi de plusieurs inputs vers le serveur (done)
* Test lecture de plusieurs inputs venant du client(done)
* How to connect several clients on the server at the same time?
* Vendredi 26 Janvier 2024:
* Threading: allows the program to run more task at the same time
* Handling disconnection for a single client(done)
* Change the code to handle multiple clients at the same time using Threading(done)
* How to handle client request?
* Handling request by extracting the host name from the request
* How to accept a request and give the client access to the website?
* How to connect my proxy server to the destination server?
* Lundi 29 Janvier 2024:
* Redirecting the blocked website to another website (still dk)
* Learning HTTP configuration to code the response to allow access to certain website (not needed)
* Install pip install requests(useless)
* Install pip install request beautifulsoup4(useless)
* Hiding user’s IP address by using the server’s IP address, i.e. the computer ‘IP address that is hosting the server
* ssl in python for handling client’s request (tomorrow)
* Mardi 30 Janvier 2024:
* Learning ssl (not started)
* Gestion des exceptions (done)
* Manao request any am le web server
* pip install flask
* Mercredi 31 Janvier 2024:
* Problème avec l’IP adresse qui peut être bloqué par le serveur web
* Connection is le fako
* Ajout de « client\_socket.setblocking(False) » 🡪 Engendre erreur nefa hilaina
* Send request to the web server for real while preventing my IP address from being blocked by the web server; therefore, it’s better trying to connect to a poor web server such as those at Madagascar.
* Debugging HTTP and HTTPS issues (ministère Malgache)(done)
* Jeudi 01 Février 2024: N/A
* Vendredi 02 Février 2024:
* Search for new proxy server project
* Mastering HTTP proxy server firstly
* Asking website’s IP address using python socket(mandeha)
* Lundi 05 Février 2024:
* Asking the IP address of a HTTP website causes issues(vita)
* Mety probleme na script rhf atao en parameter
* [IMPORTANT] Rhf misy blem lay input ny user de lasa misy ‘\r’ ny farany le izy
* Mardi 06 Février 2024 :
* Connect proxy server to website server
* DNS look-up – TCP – SSL/TLS connection(security) 3auxInfo
* Destination\_server connection
* Mercredi 07 Fevrier 2024:
* Error 400 when asking data from the server. There is a problem with the request that the program sent hence triggering the “bad request error”.
* Learning buffer
* The real request must be extracted from the request
* Issue: How to remove the \r\n at the end of the request and the port for HTTPS website
* Jeudi 08 Fevrier 2024:
* Using split solves the problem with the issue above and using ‘r’ (r’\r\n’) solves the issue with including '\r\n” inside an input without affecting it
* Converting the string request into byte is a must since the transmission is done over a network and it uses byte format; therefore, the use of .encode(‘utf-8’) was needed in order to achieve that
* Wednesday 14 February 2024:
* Creation of a valid ssl certificate is needed in order to manipulate HTTPS website.
* Installing “pyOpenssl” to achieve that
* The valid IP address list is unfinished. (better do it last)
* Friday 16 February 2024:
* Learning how to create a valid ssl certificate. A lot steps have to be made to create one (idk yet) (I know now)
* Creating a self-signed ssl certificate was successful but using it in a professional environment is not recommended since self-signed certificate are recognized as a true ssl certificate by the websites. Fortunately, there a solution for that, using a third party website to give valid ssl certificate freely. However, in order to have obtain those, my python program have to be able to navigate in the website which is quite complicated to say the least.
* Monday 19 February 2024:
* The verification of the ssl certificate can be bypassed at the risk of getting a man-in-the-middle attacks. To achieve that, using the ssl import is important since it has an attribute that can nullify the verification. (verify\_mode = CERT\_NONE). That attack can only be done by the server, fortunately, the server is under my control so any mischievous attack is very unlikely to happen.
* The request syntax for GET and CONNECT are different; therefore, it is highly possible that the other methods might differ from each other. Thus, for each method, different lines of code must be used to ensure that everything is working fine.
* LOZAAAAAAAAAA!!!! Extracting the request from the message sent by the client is useless because the request import already has an attribute for each type of methods were the host is only needed as an argument.
* Tuesday 20 February 2024:
* Knowing the difference between CONNECT, POST, GET, DELETE, PUT is important since it can cause connection and their syntaxes are different from one another.
* The request can’t handle connect request so the request has to be done manually
* Wednesday 21 February 2024:
* OKAYYY, I found a solution concerning the request issue. There is no need to learn all the method syntaxes since the message, i.e. the client’s request, can be forward to the web server in order to make the request instead of writing the request manually. NICE!!!!!!
* Need to control the buffer size in order to complete “HTTP” request
* Some issue appeared with the domain name being wronged the browser
* I’m done for today. For tomorrow, I will have to resolve the error with “Connection Aborted” by the client\_socket.recv(1024). Why did this error occur?
* Thursday 22th February 2024:
* I still don’t know why “Connection Aborted” occurred
* The reason why the forwarding isn’t working anymore is due to the message sent to the destination server - the request -. Normally it should work, like yesterday, but it’s not today. The issue is due to the domain being named wrong by something (have to find out why)
* Zoma 23th February 2024:
* Found out why the request message is wrong. I have to remove the second content of the message.
* Making request with the client message is a huge success.
* Working on the ssl now.
* Monday 26th February 2024:
* Python offers valid ssl certificates that are recognized by major web browsers. It is found in the “certifi” module with certifi.where().
* When trying to connect to webserver, we must use the hostname rather than its IP address, else it will cause an error.
* Nmap give the information on which ssl version is supported by the webserver. =>“nmap --script ssl-enum-ciphers -p 443 <hostname>”
* For today, I stopped on the part where the code treats http and https website differently.
* Tuesday 27th February 2024:
* When the verify mode needs to be set to false, the check hostname has to be set to CERT\_NONE beforehand.
* Have to know if the certificates provided by python are already configured to make the handshake with the destinated server.
* Wednesday 28 February 2024:
* Dropping the “sendall” method, instead I used the request method to do it since the latter works perfectly with the certify module of python.
* Thursday 29th February 2024:
* For now, the destination server doesn’t need to be wrapped inside the context with the ssl module. Meaning that the issue isn’t caused by the request made by the proxy to the destination server but rather caused the proxy sending the server response to the client.
* I think after send the server response to the client, the server must send another response to the client in order to complete everything
* Friday 01 – Monday 04th March 2024:
* Creation of the trusted ssl certificate in order to let the server makes the handshake with the browser.
* Creation in process. (Done)
* Tuesday 05th March 2024:
* Creation and implementation the ssl certificate in the code is complete, but it seems that an error occurred like the certificate is out of date. Have to check if the creation of the certificate didn’t go wrong in process.
* Wednesday 06th, Thursday 07th March 2024:
* Debugging the ssl error with the handshake and maybe the one about the certificate.
* It seems that it’s the client that should be wrapped inside the ssl context but not the server itself.
* Friday 08th March 2024:
  + Creation of the function which stops the server manually and the program that checks if the ssl certificate is a valid one.
* Mondy 11th March 2024:
* I just noticed that the server isn’t connected to the server itself but only the clients. In other words, the server can execute the request method using its module without creating a ssl certificate.
* TCP tunnel
* Tuesday 12th March2024:
* Creation of the webserver to display the content onto the client browser. (Not needed)
* Wednesday 13th March 2024:
* Browsers treat proxy servers in a different way than a simple client since browsers use a specific request method to make an http request. Therefore, if a client tries to connect to the proxy server directly to the browser and thinks that everything will work fine, they’re wrong.
* Thursday 14th March 2024:
* Creation of a new ssl certificate for the LOCALHOST server.
* The browser checks the ssl certificate authenticity of the server in order to create the ssl/tsl tunnel for the secured connection. The client socket doesn’t need a ssl certificate but just the CA of the ssl certificate of the server to check if it’s communicating with the correct server.
* Creation of ssl certificate is complete and for now, everything is looking good, I hope it’ll stay like that for a while.
* Debugging the ssl error about the mismatch with the key file and the certificate file. (Solved)
* The cert-key file is the private key for the ssl certificate but not the ca-key file which is the private key for the CA certificate.
* Monday 18th March 2024:
* A single server using socket can’t treat both http and https; therefore, two servers should exist at the same time to treat both http and https connection
* For now, I’m going to create the server handling the http connection. (Done)
* Tuesday 19th March 2024:
* The server managed to output the result of a https request on the client browser without a ssl certificate but I was just the header though. (Ay kay http connection lety e!)
* Found out that the true domain name of the server wasn’t ‘localhost’ but actually TOAVINA.home. (lol)(Wrong mbola localhost ihany)
* Wednesday 20th March 2024:
* First, the server should send to the client socket that the connection has been established and only then does the server makes the http request to the web server and sends the response to the client socket.
* I need to understand if the client socket still sends a message after it had made its request.
* The ssl handshake is still difficult to grasp, I have to make a breakdown of each step of this handshake thing
* Wednesday 27th March 2024:
* When creating the ssl certificate, the common name isn’t localhost but the address IP, in other words, there isn’t a DNS.
* There is what they called a ssl handshake that allows the creation of the ssl/tsl connection between the client and the server which allows the transfer of the encrypted data. After the handshake has been done, the client sends a new encrypted request to the web server and the server responds with a header first confirming that the confirming that the connection has been established and afterwards, another encrypted message is sent by the web server containing the body of the page.
* Thursday 28th March 2024:
* **DO NOT FORGET TO INSTALL THE CA FILE IN THE ROOT FOLDER OF THE CLIENT COMPUTER!!!**
* Modification of the ssl certificate if a website isn’t supported is done. Next, modification of the extfile.cnf file to remove duplicate domain names .