Projet développement

Rapport

# 

Table des matières

[1](#_Toc27850858)

[Rappel théorique 3](#_Toc27850859)

[Programmation orienté objet : 3](#_Toc27850860)

[Objet : 3](#_Toc27850861)

[Instancier : 3](#_Toc27850862)

[Classe : 3](#_Toc27850863)

[Méthode : 3](#_Toc27850864)

[Module : 3](#_Toc27850865)

[Socket : 3](#_Toc27850866)

[Argparse : 3](#_Toc27850867)

[Request : 3](#_Toc27850868)

[Pynput : 4](#_Toc27850869)

[Keylogger : 4](#_Toc27850870)

[Attaque DDOS : 4](#_Toc27850871)

[Implémentation 5](#_Toc27850872)

[Coté maitre : 5](#_Toc27850873)

[Coté esclave : 9](#_Toc27850874)

[Erreurs connues 12](#_Toc27850875)

[Sources 13](#_Toc27850876)

# Rappel théorique

## Programmation orienté objet :

Il existe deux grands types de programmation : l’orienté objet et procédurale, souvent confronté l’une contre l’autre. La programmation orienté objet (POO) permet des créer des objets, des entités, que l’on peut manipuler sans limite et les faire interagir entre eux.

## Objet :

Un objet est tout simplement une variable spéciale à qui on a attribué de plusieurs caractéristiques. De plus, les objets peuvent avoir des facultés qui sont bien définis dans le code.

## Instancier :

L’instanciation permet de créer un objet à partir d’un modèle connu. Elle est mise en place par deux grandes opérations : l’allocation et l’initialisation. L’allocation accorde un espace mémoire au nouvel objet tandis que l’initialisation lui fixe ses attributs, son état.

## Classe :

Les classes contiennent tous les attributs d’un objet, qui représente son état. Mais aussi toutes les méthodes qui impose le comportement de celui-ci. C’est grâce à une classe que l’on peut créer un objet.

## Méthode :

Une méthode est une fonction qui est associé à une classe et elle est utilisé lors d’une instance. Les méthodes sont définies dans la classe et elles permettent de donner aux objets une capacités, une interaction. Il existe des méthodes spéciales que Python reconnait et utilise facilement, comme par exemple : def \_\_init\_\_ () : cette méthode est souvent appelée « constructeur d’objet »

## Module :

Les modules sont des extensions du langage. Ce sont plusieurs fonctions que le langage n’intègre pas directement. On les retrouve souvent forme fichier. Par exemple, si je souhaite faire une racine carrée d’un nombre. Nous devons importer le module « math » car le langage ne dispose pas de fonction pour faire la racine carrée en une seule ligne de code.

## Socket :

Le module socket consiste à créer un lien entre deux machines. C’est un module de réseau qui permet de créer une relation entre plusieurs machines. Pour ce faire, il faut réserver un port qui sera identique pour chaque machine et avoir une adresse IP.

## Argparse :

Le module argparse a été publié pour remplacer les modules : getopt et optparse. Il permet de créer des paramètres pour l’interface de ligne de commande.

## Requests :

Le module requests permet d’utiliser le protocole http. En effet, il permet d’envoyer une requête http sur une url de manière très simple.

## Pynput :

Le module pynput vous autorise à contrôler les appareils de votre ordinateur, tels que : l’écran, la souris, le clavier, etc… Il utilise la fonction listener () qui un thread (un processus) afin d’écouter l’appareil ciblé. Dans notre cas, celui-ci s’est avéré très utile pour notre keylogger.

## Keylogger :

Un Keylogger est un programme informatique à but malveillant qui enregistre toutes les frappes d’un clavier. Ils ne sont pas répertoriés comme virus, car ils ne font que d’enregistrer les informations. C’est un logiciel dit « d’espionnage ».

## Attaque DDOS :

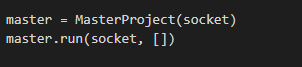
Une attaque DDOS vise à rendre indisponible un service en ligne. C’est grâce à une multitude de requête envoyé depuis plusieurs ordinateurs que le service devient indisponible. Les ordinateurs sont dits « zombie » car ils ont été contaminés et ils sont dirigés par un ordinateur maitre. Il existe aujourd’hui plusieurs moyens de se protéger de cette attaque.

# Implémentation

## Coté maitre :

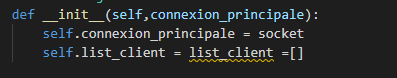
La classe MasterProject () :

Nous disposons d’une seule classe pour notre maitre. On y retrouve toutes nos méthodes à l’intérieur de cette classe. On retrouvera en dehors de cette classe deux lignes codes :



L’appellation de notre classe puis la méthode run () afin de lancer le corps de notre programme.

Dans notre méthode \_\_init\_\_ () on y retrouvera deux objets : notre socket et une liste qui contiendra toutes les machines esclaves connectés à notre maitre.



La méthode run () va contenir la plus grade partie du code. Les paramètres de cette méthode seront notre socket et notre liste qui ont été tous les deux créent dans notre def \_\_init\_\_ ()

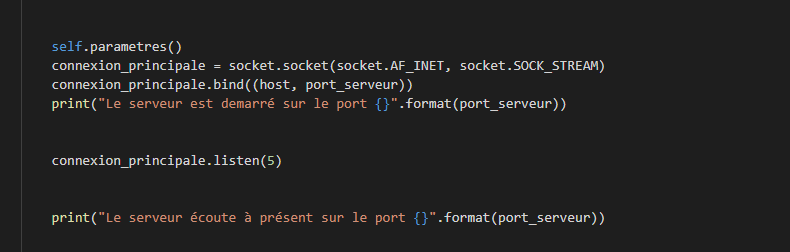


Dans notre méthode run () on trouvera :

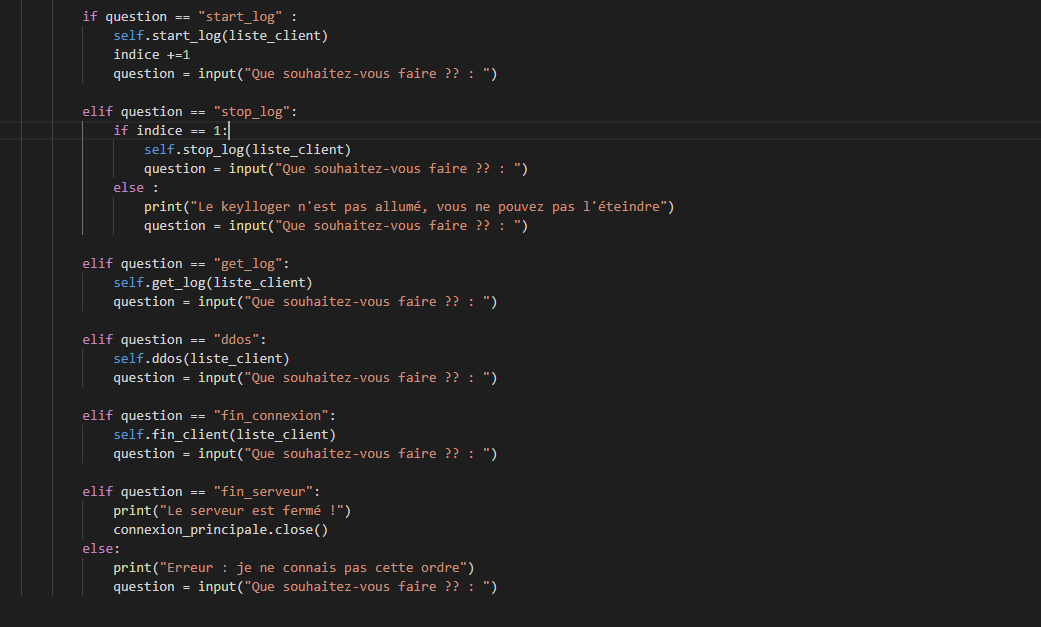
* L’appellation de notre argparse via une méthode parametre () qui instanciera notre argument
* L’instanciation de notre socket plus une information sur le port utiliser qui sera afficher sur le pc maitre



* Une boucle qui mettra d’enregistrer tous les nouveaux clients dans une liste afin de les compter et de savoir envoyer un message à tous

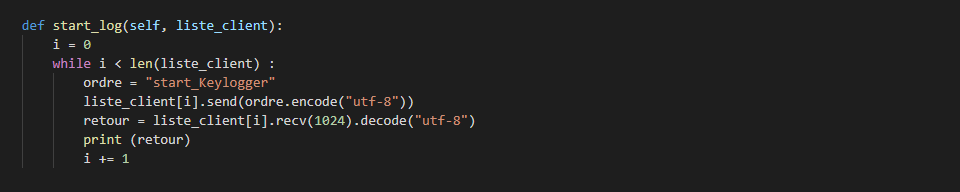


* Un menu qui, en fonction d’un message tapé par l’utilisateur, va lancer/démarrer une méthode bien précise. Une de ces méthodes arrête le serveur.

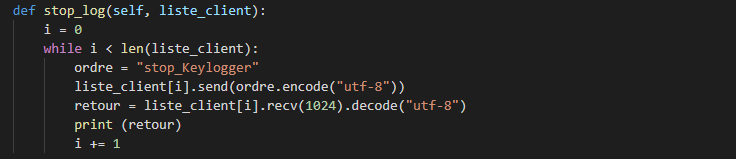


Les méthodes recommandées :

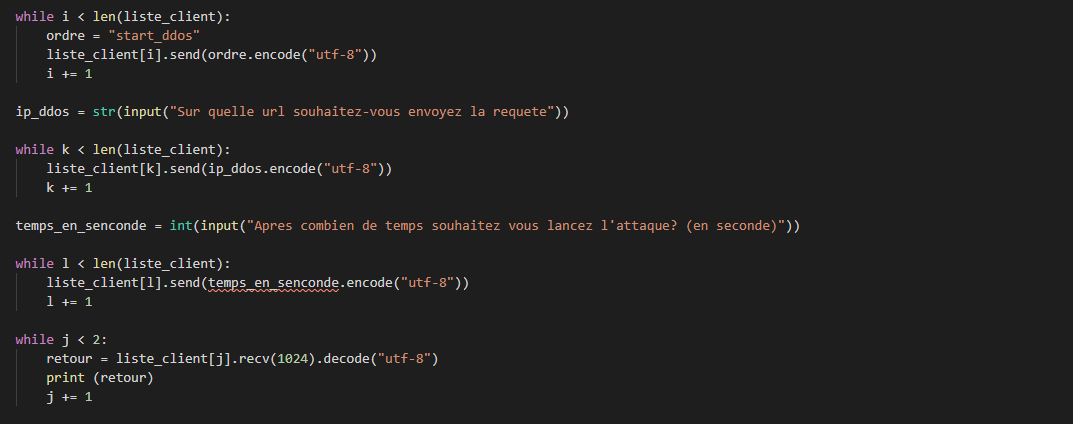
* Start\_log () : cette méthode permet d’envoyer un mot aux esclaves et de lancer le keylogger, ensuite, le maitre reçoit un message comme quoi le keylogger est bien lancé.



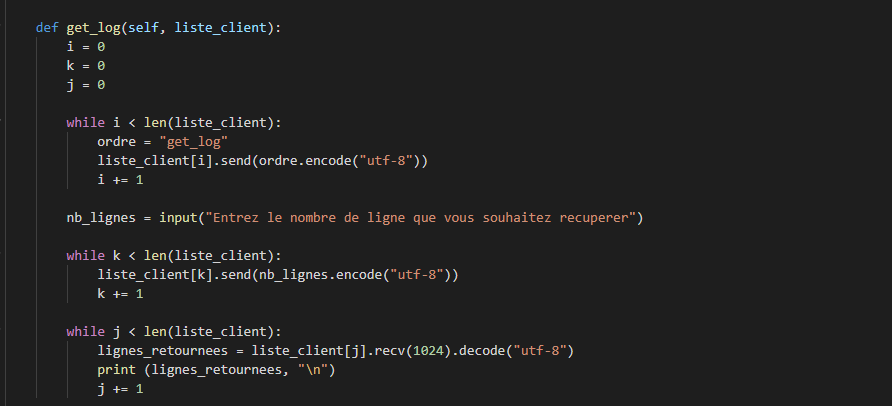
* Stop\_log () : celle-ci envoie un message aux esclaves, stop le keylogger et reçoit un message pour dire que le keylogger est stoppé correctement.



* Get\_log () : Cela envoi message aux esclaves pour récupérer un certain nombre de ligne provenant du keylogger, demande à l’utilisateur combien de ligne il veut et les affiches.

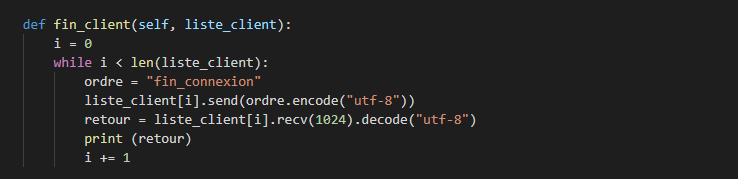


* DDOS () : Permet de dire aux esclaves de faire une requête vers un site web, les esclaves envoient une réponse si la requête a fonctionné.

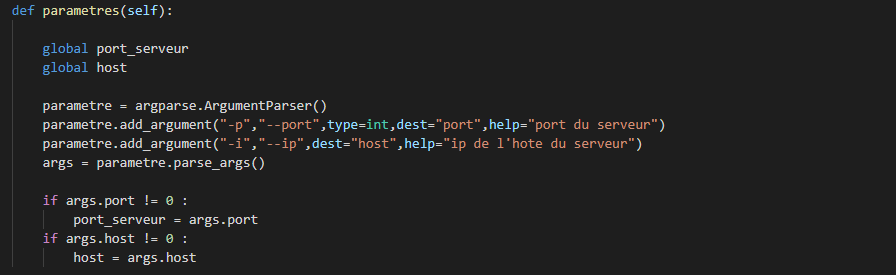


Les autres méthodes utilisées :

* Fin\_client () : Permet d’envoyer un message aux esclaves pour les fermer et le maitre reçoit un message comme quoi les clients sont bien fermés.



* Parametre () : Cette méthode va créer notre argparse et sera utiliser pour déclarer l’IP et le port d’écoute du serveur.



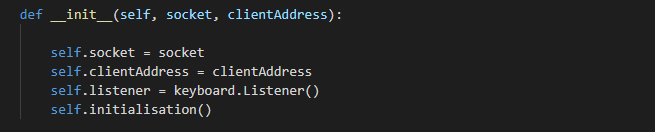
## Coté esclave :

La classe Client () :

Nous avons qu’une seule classe pour notre esclave. Toutes les méthodes sont à l’intérieur de cette classe dont deux méthodes importantes :

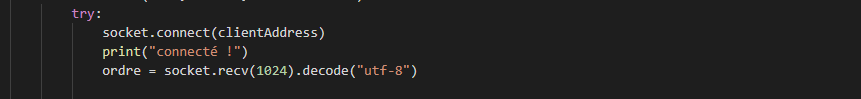
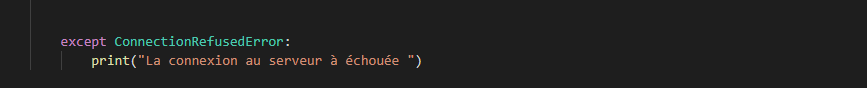
La méthode def \_\_init\_\_ () :

On trouve la déclaration de notre socket, de l’adresse client, de notre listener pour le keylogger et une méthode qui servira à écouter tout le temps.

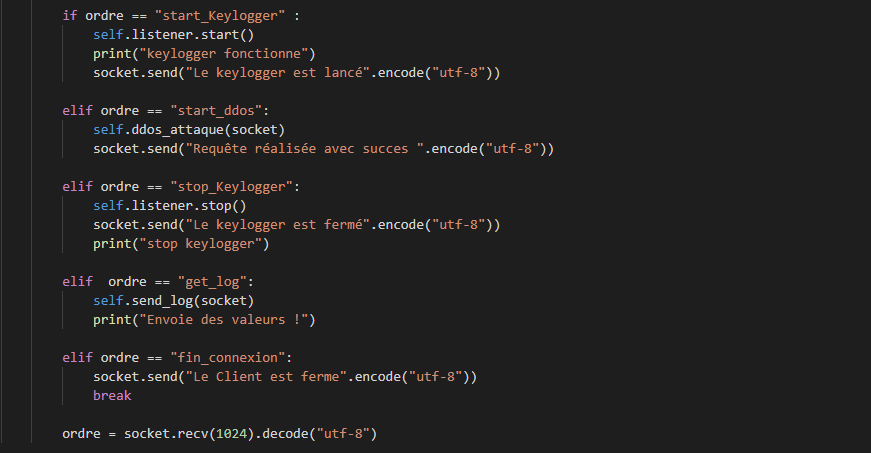


Dans notre méthode connexion () :

* Contient une gestion d’erreur au cas où le client veut se connecter alors que le server n’est pas lancé.
* On se connecte à notre serveur.
* On reçoit un mot, un ordre.



* En fonction de cet ordre on exécute une méthode bien précise et de d’envoyer un message en cas de succès.

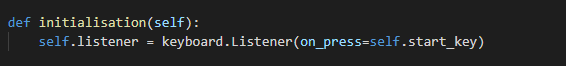


Les autres méthodes :

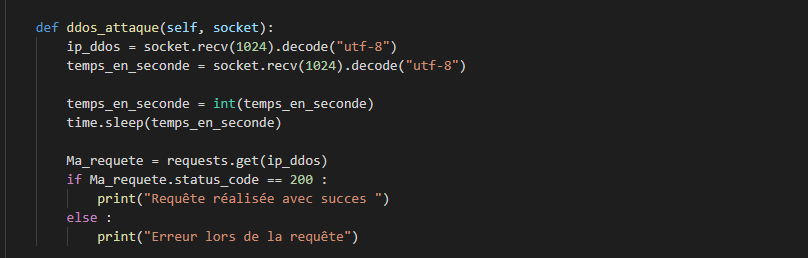
* Start\_key () : permet d’enregistrer tout ce que l’on tape.



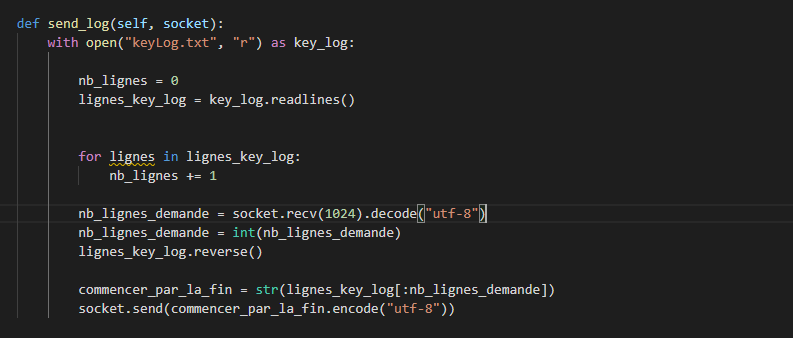
* Initialisation () : cette méthode permet d’écouter tout le temps le clavier de la cible.



* Ddos\_attaque () : celle-ci va recevoir deux mots qui vont lui servir de paramètre afin d’exécuter une requête http vers un site web. Les deux paramètres sont le temps et la cible.

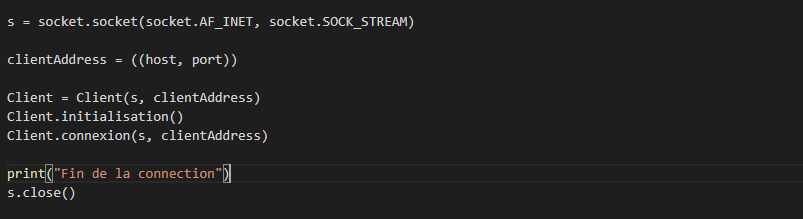


* Send\_log () : La méthode va ouvrir le fichier du keylogger et compter toutes les lignes du fichier. En fonction du message reçu, elle va envoyer, en commençant par la fin, les lignes que le maitre souhaite grâce au slice (découpage).

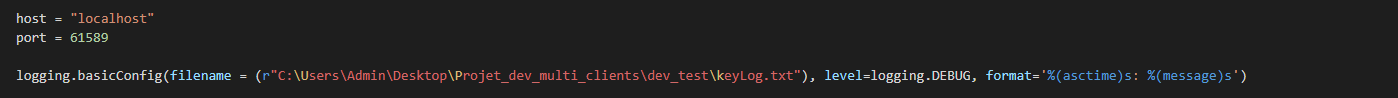


En-dehors de la classe on trouve :

* La déclaration de notre socket plus la fermeture et de l’IP, le port
* Les appels de des méthodes initialisation et connexion via la classe



* Le chemin d’écriture pour le keylogger, savoir où il doit écrire ce qu’il enregistre.



# Erreurs connues

* Une fois qu’on lance le keylogger et qu’on le stop, on ne peut plus le relancer pour la deuxième fois.
* Nos requête http sont écrites dans le fichier du keylogger, c’est peut-être à cause du fait que notre listener écoute tout le temps. Pour nous, c’est un plus mais on ne demande pas d’écrire cela dedans. Même une fois stopper, les requêtes sont quand-même écrites.

# Sources

Keylogger :

* <https://www.tutorialspoint.com/design-a-keylogger-in-python>
* <https://openclassrooms.com/fr/courses/235344-apprenez-a-programmer-en-python/2235545-faites-de-la-programmation-parallele-avec-threading>

Socket :

* <https://www.youtube.com/watch?v=5FqzL9LJkXA>
* <https://portail.henallux.be/moodle/pluginfile.php/87984/mod_resource/content/13/Support%20de%20cours%20v1.12.pdf>
* <https://realpython.com/python-sockets/#handling-multiple-connections>
* https://openclassrooms.com/fr/courses/235344-apprenez-a-programmer-en-python/234698-gerez-les-reseaux

Argparse :

* <https://www.youtube.com/watch?v=q94B9n_2nf0>
* <https://portail.henallux.be/moodle/pluginfile.php/87984/mod_resource/content/13/Support%20de%20cours%20v1.12.pdf>
* https://realpython.com/command-line-interfaces-python-argparse/

Get\_log :

* <https://openclassrooms.com/fr/courses/1206331-utilisation-avancee-des-listes-en-python>
* <https://openclassrooms.com/fr/courses/235344-apprenez-a-programmer-en-python/232431-utilisez-des-fichiers>
* <https://python.doctor/page-lire-ecrire-creer-fichier-python>
* https://waytolearnx.com/2019/04/lire-un-fichier-ligne-par-ligne-en-python.html

DDOSS :

* <https://pypi.org/project/requests/>
* https://requests.readthedocs.io/en/master/