# Remote administration tool (RAT)

Il s’agit d’un programme permettant la prise de contrôle à distance d’un ordinateur depuis une autre machine. A l’origine, ce type de programme est légitime dans le milieu professionnel, Il est utilisé pour assurer la gestion à distance d'un serveur ou le dépannage d'un poste de travail, cependant l’utilisation celui-ci peut très vite être détournée à des fins malveillantes. En effet ce type de programme peut également être identifié comme une backdoor et être exploité par un acteur malveillant qui souhaite s’introduire dans les systèmes d’un ordinateur à l’insu de son propriétaire.

Un RAT se compose d’un client et d’un serveur :

* Le client : est positionné sur l’ordinateur de la machine cible afin d’établir une prise de contrôle du poste.
* Le serveur : est positionné sur l’ordinateur qui prendra le contrôle de la cible.

Il existe toutefois des limites et des conditions à son utilisations :

* Les privilèges : d’un potentiel assaillant sont limités aux privilèges de l’utilisateur actuellement connecté à la machine et à la sécurité de la configuration système et réseaux du poste.
* Les postes : client et serveur doivent disposer d’une connexion internet.
* Le client : devra être exécuté au minimum une fois afin de permettre la connexion avec le serveur et potentiellement garantir l’insertion d’une tâche automatique sur le poste client.

Les options :

Dans le cadre du projet, nous avons privilégiez l’utilisation de méthodes pouvant :

* exfiltré des informations ;
* permettre une prise de contrôle à distance sur un terminal de commande ;
* établir une action qui subsistera sur le poste ;
* infiltré des fichiers ;
* contrôler l’état physique de l’ordinateur cible.

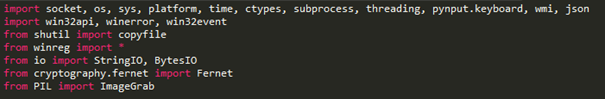
# 

# Le client

Il s’agit d’un programme qui reste à l’écoute d’un serveur afin d'exécuter toutes les commandes qui lui seront communiquées. Le serveur communiquera des mots clés au client, ces mots seront associés aux fonctions afin de faciliter l'exécution des commandes.

## Les modules

La réalisation du client nécessite l’intégration d’un ensemble de modules afin d’exercer les différentes actions qui vous seront présentées. Voici un court descriptif de leurs utilités



Les différents modules qui sont présent sur l’image suivante permettent :

**Socket** : l’interconnexion entre le serveur et le client.

**OS** : l’utilisation des fonctionnalités dépendantes du système d’exploitation.

**Sys** : l’exploitation de certaines variables utilisées et maintenues par l'interpréteur.

**Platform** : l’interrogation d’un système afin de récupérer des informations le concernant (OS, version, matériel, …).

**Time** : l’utilisation des fonctions de temps.

**Ctypes** : l’utilisation de fonctions externes à python (DLL et C).

**Subprocess** : la création de nouveaux processus et l’utilisation de sous-processus.

**Threading** : l’utilisation de thread.

**Pynput** : de surveiller et de récupérer les entrées clavier.

**Wmi** : l’utilisation de commande Windows afin de vérifier le type d’os utilisé.

**Win32api, winerror et win32event** : l’utilisation de commande Windows (il s’agit de la librairie **pywin32**).

**Shutil** : d’établir des opérations sur des fichiers (copie, suppression, …).

**Winreg** : l’utilisation des clés de registre Windows.

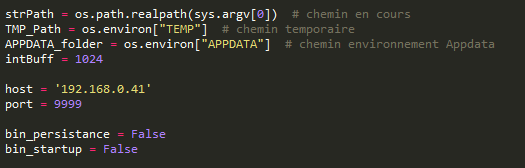
**IO** : l’écriture et la lecture de flux de données en bytes.

**Cryptography** : d’une librairie de chiffrement de type Fernet.

**PIL** : l’utilisation d’une librairie permettant la réalisation de screenshot.

## Variables globals

L’utilisation de variable global, rend une variable accessible à l’ensemble des fonctions.



Ces variables vont permettre :

- Windows :

o strPath : d’identifier la position de l’assaillant dans l’arborescence ;

o TMP\_ PATH : de se positionner dans le répertoire TEMP ;

o APPDATA\_filder : de se positionner dans le répertoire APPDATA.

- IntBuff : la valeur du buffer pour les envois et les réceptions ;

- Host : l’adresse IP du serveur ;

- Port : le port de connexion du serveur ;

- Bin\_persistance : Un booléen qui vérifie de l’état d’une fonction ;

- Bin\_startup : Un booléen qui vérifie de l’état d’une fonction ;

- KeyListener : Cette variable permet d’initialiser la fonction keylogger pour la mettre en mode écoute. Elle utilise deux fonctions (Listener du module pynput et OnkeyboardEvent qui est une fonction du programme) ;

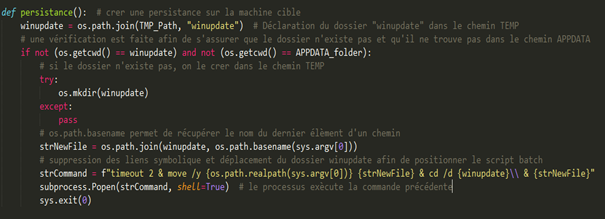
- Key : Il s’agit d’une variable qui va garder en mémoire la fonction **Key** du module **pynput**.

## 

## Les fonctions

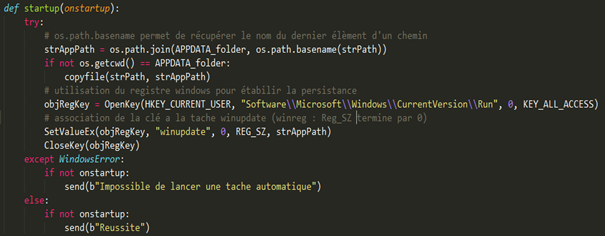
Les fonctions qui vous seront présentées, sont les définitions des actions qui permettent au programme d’agir comme nous le souhaitons.

### Persistance :

Cette fonction va permettre la création d’une tâche du nom de « winupdate », celle-ci servira à positionner notre tâche automatique régulière.

### Startup :

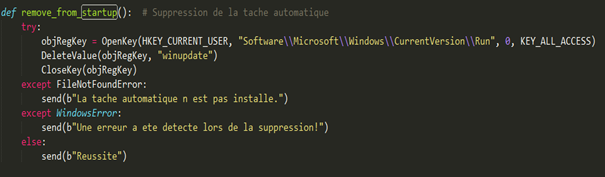
Cette fonction va permettre d’immiscer une tache automatique au démarrage du poste dans les processus Windows.



La variable ObjRegKey est utilisée afin de cibler le registre des clés du profil d’utilisateur en cours.

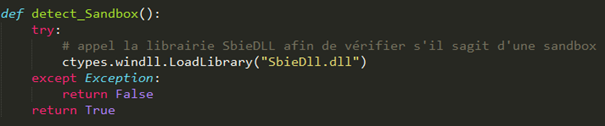
### Remove\_from\_startup :

Cette fonction va permettre la suppression de la tâche automatique réalisée avec la fonction startup(). Nous réalisons une suppression de la tâche « winupdate » dans le registre des clés. Des vérifications complémentaires sont faites pour vérifier que la tâche existe.



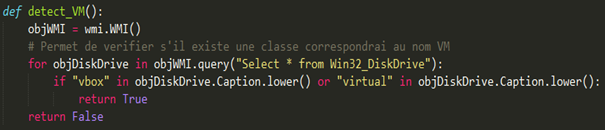
### Detect\_Sandbox :

Cette fonction va permettre de vérifier à l’aide de la librairie système de la machine cible, si l’environnement infecté est une sandbox.



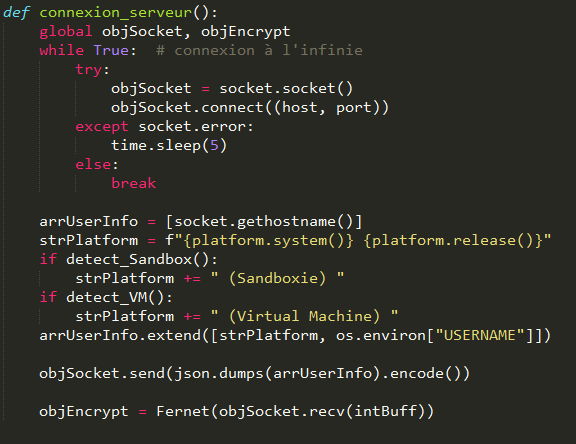
### Detect\_VM :

L’exécution de cette fonction va réaliser une vérification système de la machine cible afin d’identifier si les mots « vbox » ou « virtual » existent dans le registre des classes de la machine afin de déterminer s’il s’agit d’une machine virtuelle ou d’une véritable machine.



### Connexion\_socket :

Cette fonction va initialiser notre socket sur le port et l’adresse IP précédemment définie afin d’émettre une connexion sur le serveur distant. Lors de l’exécution de celle-ci une vérification du type de système sera réalisée à l’aide des fonctions « **Detect\_VM** » et « **Detect\_sandbox** ».



Ces informations seront envoyées sous un format json au serveur à l’aide de la fonction « **send** » sous un format encodé.

Le serveur émettra la clé chiffrement au client par la même occasion.

### 

### Recv :

Cette fonction va permettre de recevoir et de déchiffrer tous les éléments qui sont transmis par le serveur. A l’aide de la variable « objEncrypt » précédemment associé à la clé de chiffrement Fernet, il sera capable de déchiffrer tous les flux.



Ici la fonction prend en argument la variable « buffer », qui retournera les éléments reçus sous un format string.

### Send :

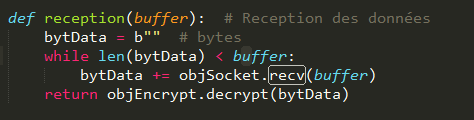
Cette fonction va permettre l’envoi et chiffrement des données vers le serveur. A l’aide de la variable « objEncrypt » précédemment associé à la clé de chiffrement Fernet, il sera capable de chiffrer toutes les données.



Ici la fonction prend en argument la variable « data », qui enverra à travers la socket les données.

### Reception :

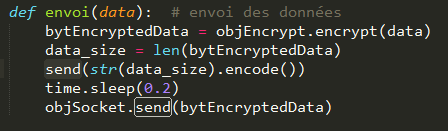
Cette fonction va permettre de réceptionner de grand flux de données à l’aide de la boucle while. Elle servira au téléchargement de fichiers.



### Envoi :

Cette fonction nécessite que les données soient converties en bytes.

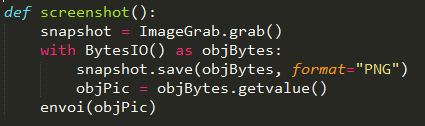
Afin que l’intégrité des données ne soit pas corrompue, nous informons le serveur de la taille des données qui seront transférées, puis nous lui envoyons les données.



### Screenshot :

A l’aide du module « **Pillow** » nous pouvons utiliser la classe « **ImageGrabe** » qui nous permet de réaliser des screenshots dans un format « png » à la suite de quoi nous convertissons le screenshot en bytes.

Enfin nous faisons appel à la fonction « **envoi** » afin d’envoyer le fichier au serveur.

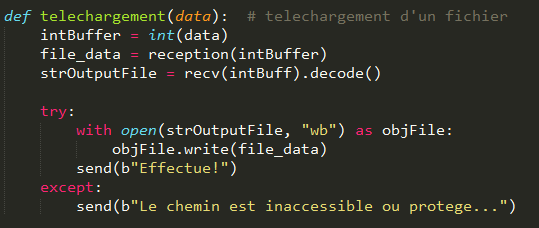


### 

### Telechargement :

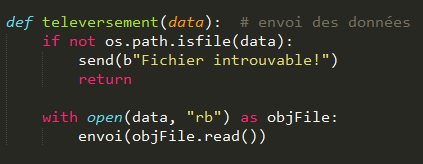
Cette fonction se charge de télécharger un fichier qui sera envoyé par le serveur, l’argument data contiendra la taille du fichier qui sera envoyé.

Il sera fait appel à la fonction « **reception** » pour récupérer le contenue du fichier et à la fonction « **recv** », afin de récupérer le nom que devra portée celui-ci. Une fois ces éléments réceptionnés, les données seront écrites en bytes dans un fichier.



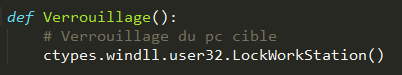
### Televersement :

La fonction va récupérer en argument le nom (avec l’arborescence qui lui est associé) du fichier à envoyer au serveur. Ce fichier est retranscrit en bytes afin d’être envoyé à l’aide de la fonction « **envoi** »



### Verrouillage :

Cette fonction applique un verrouillage de la session en cours. Cette action est réalisée à l’aide du module « **Ctype** » qui nous permet d’exécuter une commande système Windows.

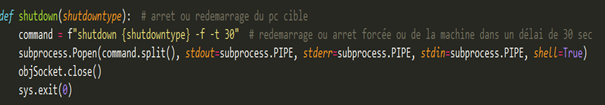


### Shutdown :

Cette fonction dispose de deux particularités, elle peut éteindre ou redémarrer le poste cible en fonction de l’argument qui lui sera communiqué. Cette action a été effectué à l’aide du module « **subprocess**»

Argument possible :

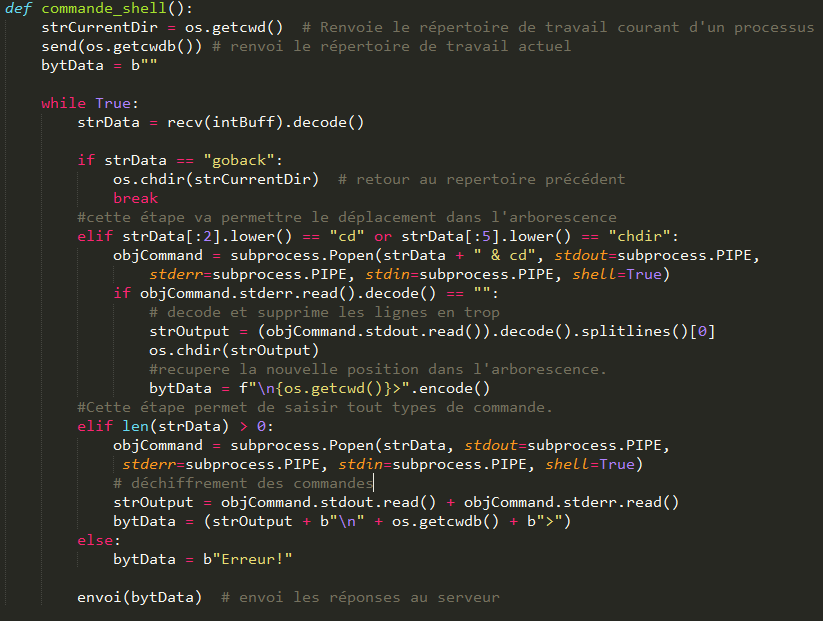
* S : Arrêt du poste ;
* R : Redémarrage du poste.



### 

### Commande\_shell :

Cette étape nécessite l’utilisation de plusieurs commandes du module « **OS** » afin de naviguer dans l’arborescence de la cible. De nombreuses conversions seront réalisées à l’aide du module « **subprocess** » ,afin d'adapter la saisie faite par le serveur pour que celle-ci soit interprétable par le terminal de commande Windows de la cible.



### 

### MessageBox :

La fonction fait appelle au module « **OS** afin de créer un pointeur sur le fichier « m.vbs »

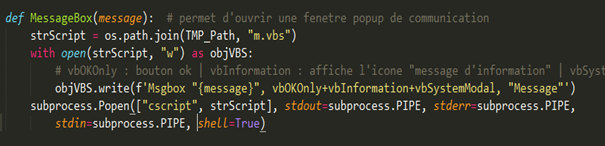
A l’aide de ce fichier, il est possible d’afficher un message popup sur l’écran de l’utilisateur.

La concaténation des options de la fonction « write » permettent de définir des options sur la fenêtre popup :

- vbOKOnly : bouton ok

- vbInformation : affiche l'icône "message d'information

- vbSystemModal : interrompt les applications



### OnKeyboardEvent :

Cette fonction a pour objectif d’identifier les commandes spéciales afin de les interpréter dans la réponse qui est envoyée au serveur. Cette fonction est par la suite appelée par la variable global « **KeyListener** » qui enregistrera tous les évènements.

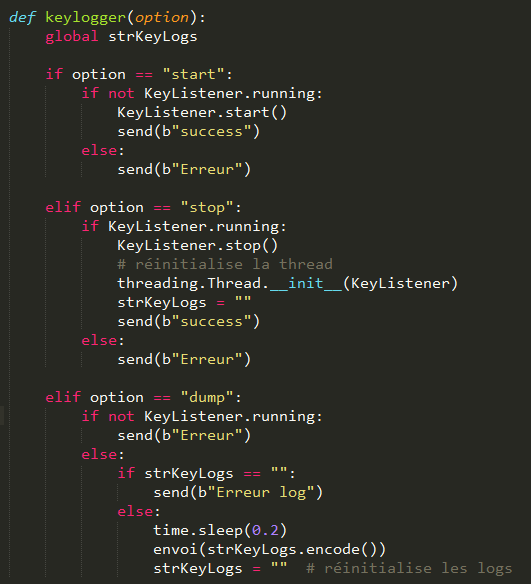


### Keylogger :

Cette fonction nécessite une déclaration de paramètre de la part du serveur afin d’être exécutée. Les paramètres sont start, stop et dump.

Ils permettent de définir l’état de fonctionnement du keylogger. Ici il sera principalement fait appelle aux variables globals Keylistener et strKeyLogs.

Keylistener est responsable du statut du keylogger alors que strKeyLogs se chargera de garder en mémoire l’ensemble des commandes.



### Main :

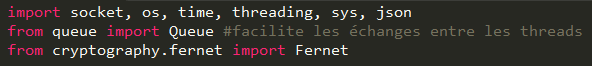
Il s’agit de la fonction principale qui est responsable de l’exécution des commandes transmises par le serveur. Il s’agit d’une cascade de conditions dépendantes des messages envoyés par le serveur.



# Le serveur

Le serveur est chargé de la communication entre les différents clients, il détient le rôle de maître. Il transmet ses directives aux différents clients et réceptionne certaines informations. Il lui est également possible de prendre le contrôle à distance de l’un de ses clients.

## Les modules



Les modules qui sont présent le serveur permettent :

**Socket** : l’interconnexion entre le serveur et le client.

**OS** : l’utilisation des fonctionnalités dépendantes du système d’exploitation.

**Sys** : l’exploitation de certaines variables utilisées et maintenues par l'interpréteur.

**Time** : l’utilisation des fonctions de temps.

**Threading** : l’utilisation de thread.

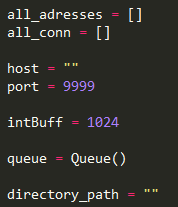
**Cryptography** : l’utilisation d’une librairie de chiffrement de type Fernet.

**Queue**: de faciliter les échanges d’informations entre plusieurs threads.

## 

## Les variables globals

L’utilisation de variable global, rend une variable accessible à l’ensemble des fonctions.



Les variables servent à :

* all\_adresse : stocker dans une liste l’ensemble des adresses IP qui se connectent au serveur ;
* all\_con : stocker dans une liste l’ensemble des objets de connexions de la classe socket ;
* host : définir l’adresse ip du serveur (localhost) ;
* port : définir le port de communication ;
* intBuff : la valeur en octet des transferts de données ;
* directory\_path : définir le nom du dossier qui contiendra les éléments récupérer (screenshot, logs).

## Les fonctions

Les fonctions qui vous seront présentées, sont les définitions des actions qui permettent au programme d’agir comme nous le souhaitons.

### Suppression\_Quotes :

Elle permet de supprimer de toutes les guillemets contenus dans une string.



### Centrer :

Elle permet de centrer les listes d’informations récupérées lors d’une connexion, afin d’établir un affichage propre et intuitif des éléments de connexion.



### Recv :

Cette fonction va permettre de recevoir et de déchiffrer tous les éléments qui sont transmis par le serveur. A l’aide de la variable « objEncrypt » précédemment associé à la clé de chiffrement Fernet, il sera capable de déchiffrer tous les flux.



Ici la fonction prend en argument la variable « buffer », qui retournera les éléments reçus sous un format string.

### Send :

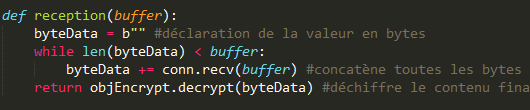
Cette fonction va permettre l’envoi et chiffrement des données vers le serveur. A l’aide de la variable « objEncrypt » précédemment associé à la clé de chiffrement Fernet, il sera capable de chiffrer toutes les données.



Ici la fonction prend en argument la variable « data », qui enverra à travers la socket les données.

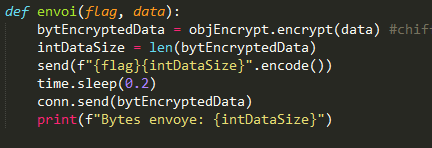
### Reception :

Cette fonction va permettre de réceptionner de grand flux de données à l’aide de la boucle while. Elle servira au téléchargement de fichiers.



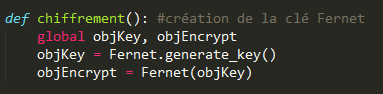
### Envoi :

Cette fonction est exclusivement réservée à l’envoi de fichier vers un client.  
Le paramètre flag va permettre l'initialisation de la réception côté client. Lors d’un transfert, la taille du fichier à émettre est communiqué afin de préserver l'intégrité du fichier lors de son transfert.



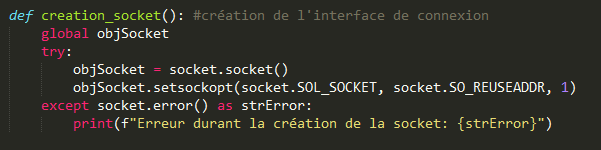
### Chiffrement :

À l’aide du module « chiffrement », nous avons généré une clé Fernet. Celle-ci sera communiquée à l’un des clients lors de sa tentative de connexion. Cette clé sera utilisée afin de chiffrer l’ensemble des échanges.



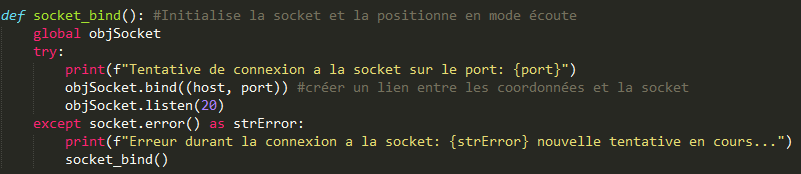
### Creation\_socket :

À l’aide du module « socket » nous avons créé une interface de connexion.



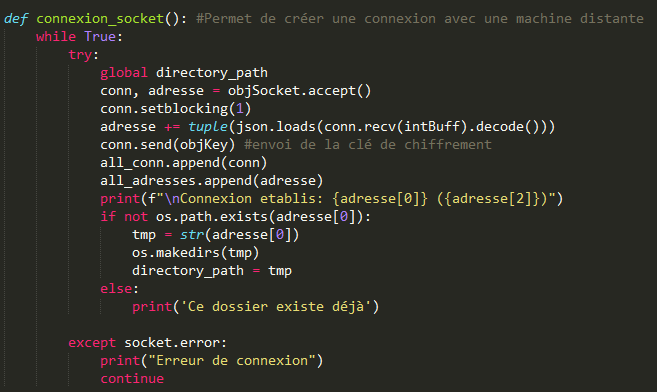
### Socket\_Bind :

Une initialisation de la socket est effectuée sur l'adresse IP et le port qui ont été déclarés dans les variables globales. Suite à l’initialisation, elle se met en écoute afin de permettre à un client de se connecter.



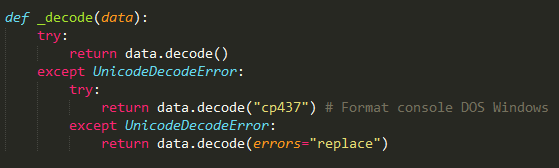
### Connexion\_socket :

Les variables conn et adresse vont récupérer les informations de connexion du client qui tente de se connecter. Ces informations sont stockées dans les listes all\_adresses et all\_con. Suite à cette connexion, la clé qui avait été générée par la fonction “**chiffrement**” est communiquée au client. Dans une logique de classement des informations qui seront récupérées ultérieurement, un dossier du nom de l’adresse IP est créé, son chemin d’accès sera stocker dans la variable global “directory\_path”.



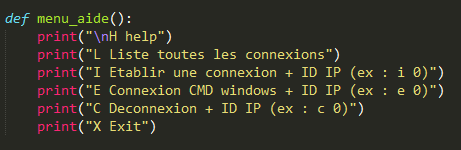
### decode\_cmd :

Cette fonction à pour rôle de décoder les données reçu, plus particulièrement lorsqu’il s'agit de commande cmd Windows afin de les rendrent interprétable par le terminal serveur.



### Menu\_Aide :

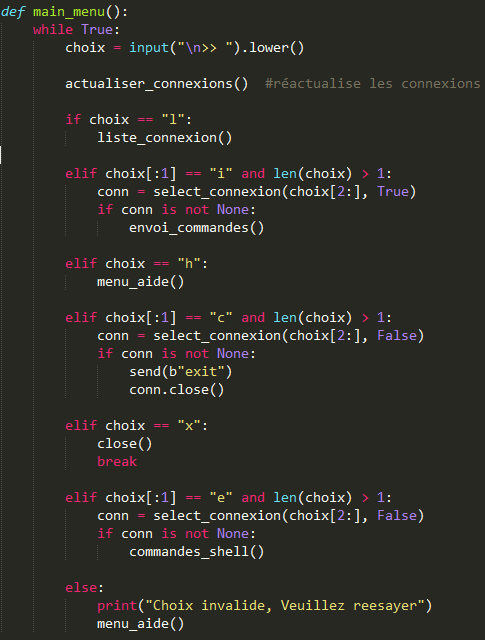
Cette fonction sert de rappel à l’utilisateur, elle indique les options qui peuvent être saisies dans le menu principal.



### Main\_Menu :

Il s’agit du menu principal de notre programme, ces différents choix vont permettre de réaliser une action sur l’une des machines qui s’est connectée au serveur. Trois de ces actions nécessitent la saisie de l’ID qui est attribué au client connecté.

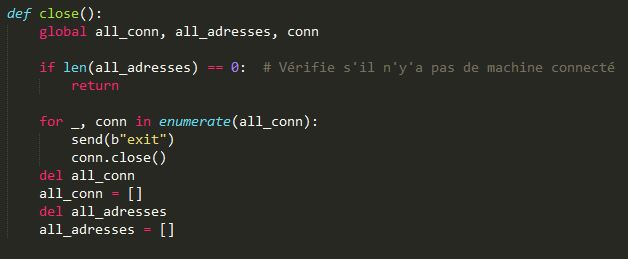
On retrouve les fonctions “**menu\_aide**” vu dans la partie supérieur, “Select\_connexion” qui permet de se connecter à un client ainsi que la fonction “**actualiser\_connexion**” qui, comme son nom l’indique, actualise la liste des connexions clients.



### 

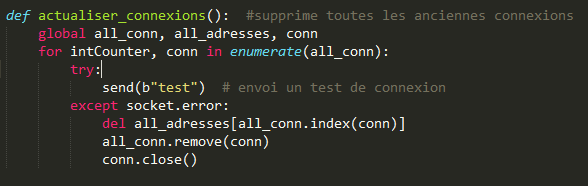
### Close :

Cette fonction permet de réinitialiser les listes globales qui contenait les informations de connexion afin de fermer toutes les connexions client.



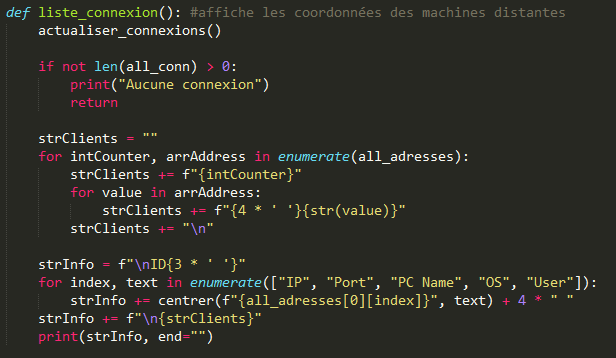
### Actualiser\_Connexion :

Cette fonction va envoyer un message clé a toutes les clients connecté afin de vérifier qu’ils sont bien actifs. Le mot clé “test” n’effectuera aucune action sur le client. Toute adresse entraînant une erreur de socket sera supprimé.



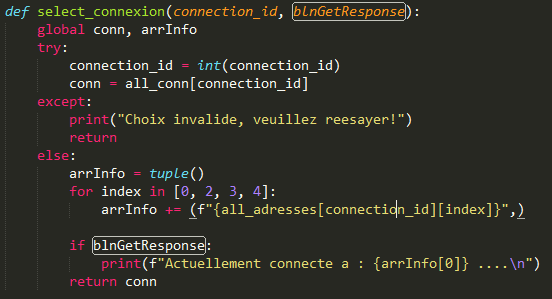
### Liste\_Connexion :

Cette fonction affiche la liste de l’ensemble des connexions sur le serveur.   
Il est fait appelle a deux fonctions “**actualiser\_connexion**” pour vérifier l’état des machines et la fonction “**centrer**” qui comme son nom l’indique va permettre d’ajuster l’affichage des informations collectées.



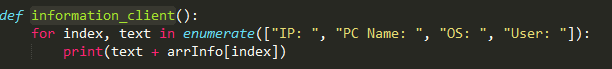
### Select\_Connexion :

Cette fonction permet d’établir une connexion avec un client.  
Pour ce faire, il est nécessaire de saisir l’ID du client ainsi qu’un booléen qui indique s’il est nécessaire de renvoyer une réponse.



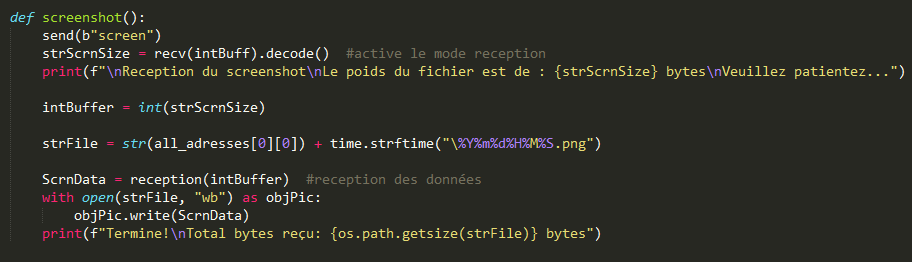
### Information\_Client :

Cette commande permet d’afficher les informations qui concernent le client avec lequel le serveur est connecté.



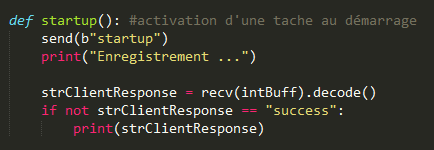
### Screenshot :

Cette fonction se charge d’envoyer une demande de screenshot au client et de réceptionner la capture émise par le client. Le fichier sera transmis et écrit en bytes.



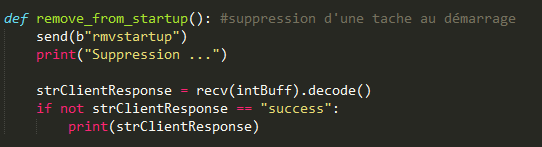
### Startup :

Cette fonction va envoi une commande au client, afin qu’il créait une tâche automatique du nom de winupdate pour d’établir une persistance sur la machine cible.



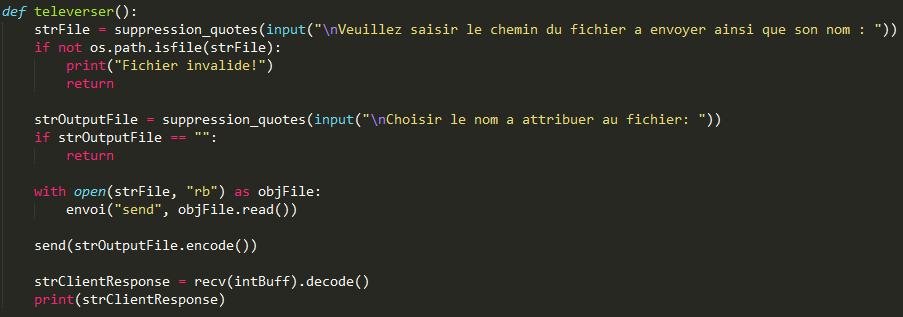
### Remove\_From\_Startup :

Cette fonction envoi une commande au client afin qu’il supprime la tâche automatique créer avec la fonction “startup”.



### Televerser :

Cette fonction permet d’envoyer un fichier au client. Lors d’un envoi de fichier, il est impératif de garder en mémoire que le fichier sera envoyé à l'emplacement de l’arborescence ou est actuellement positionner le client.



### 

### Telecharger :

Cette fonction permet au serveur de télécharger un fichier depuis la machine du client. Il est recommandé de se positionner dans l'arborescence avec les commandes CMD avant de réaliser cette action.



### Commandes\_shell :

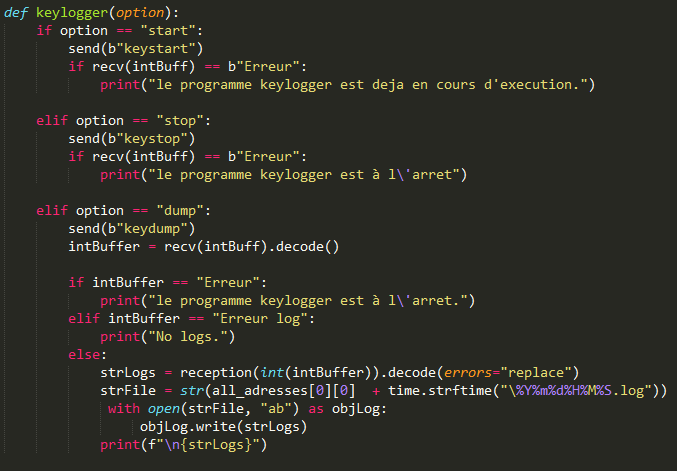
Cette fonction permet au serveur d'émettre une demande au client, afin qu’il ouvre un terminal de commande windows à distance. À l’aide de la fonction “**decode\_cmd**”, le terminal du serveur est capable d'interpréter les réponses reçues.



### 

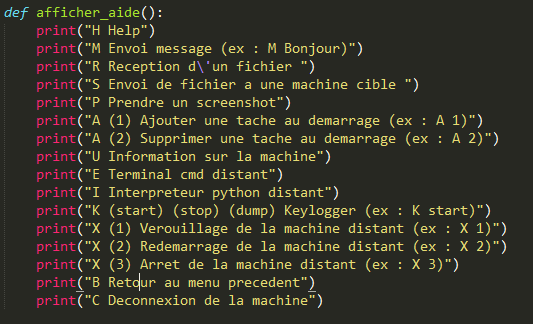
### Keylogger :

Cette fonction transmet les directives sur l'état du keylogger du poste client. Les informations collectées seront stockés dans un fichier log.



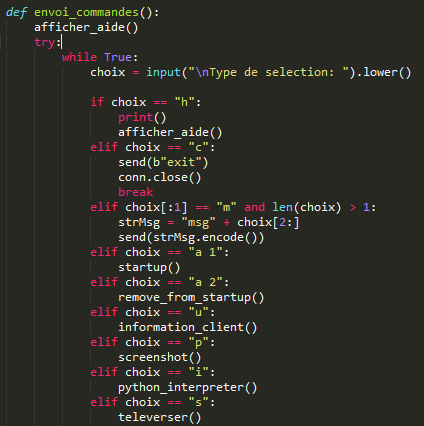
### Afficher\_aide :

Cette fonction des plus simpliste, affiche les options du menu secondaire qui permet l'exécution de certaines fonctions.



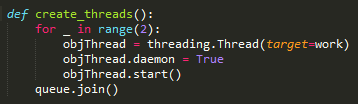
### Envoi\_Commandes :

Cette fonction est une cascade de fonctions qui exécute des fonctions selon le choix de l’utilisateur. Voici une capture partiel de celle-ci



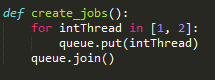
### Create\_Threads :

Cette fonction va permettre de créer deux threads, elles serviront à garantir le bon fonctionnement des interfaces de commandes. Cette fonction permet également d’exécuter la fonction “**Work**”.



### Create\_Jobs :

Cette fonction permet de créer des objets intitulé queue qui vont permettre de faciliter les échanges d’informations entre les threads.



### Work :

Il s’agit de la fonction principale, celle-ci va permettre d’établir de bon déroulement des fonctions primaires.

