# Descripción de las APIS de Asterisk

Las APIS de Asterisk estamos hablando de lenguajes de programación sea java, php, Pyton etc. Ya cuando el programador o desarrollador quiere interactuar con un Asterisk de manera directa. Cuando alguien quería hacer algo que no estuviera dentro del Asterisk, la única opción era crearlo uno mismo su módulo de Asterisk, compilarlo y arrancarlo. Para intentar subsanar esto se crearon dos APIS, el primero el AMI (Asterisk Manager Interface), se trabaja con un lenguaje de programación que permitía ejecutar acciones sobre el Asterisk, ejecutar algunas acciones sobre las aplicaciones y monitorea los eventos que sucedan en Asterisk. También viene el tema del AGI (Asterisk Gateway Interface) lo que permite ejecutar programas desde el dial plan con lo cual con una llamada se podría ejecutar un programa externo. Después tenemos el tema de ARI (Asterisk Rest Interface) que básicamente te permite construir tu propia aplicación, en general cubre las deficiencias del AMI y el ARI.

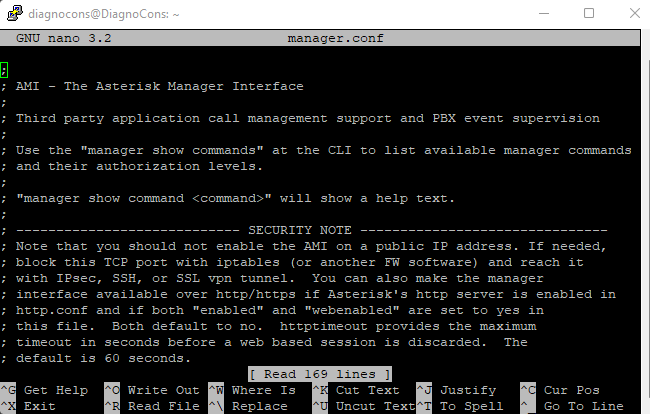


# Configuración del manager.conf

Como habíamos comentado el AMI nos permitirá ejecutar acciones de Asterisk desde un programa externo y a la vez leer eventos del Asterisk desde un programa externo y puede estar en cualquier lenguaje. En este caso vamos a trabajar con Python, pero ese lenguaje ejecuta acciones, mediante las acciones actúa sobre el Asterisk y mediante los eventos escucha que pasa en el central Asterisk.

El archivo involucrado en lo que es Asterisk se llama “manager.conf” ubicado en el directorio “/etc/asterisk/” y para ingresar a el usaremos el sig. comando:

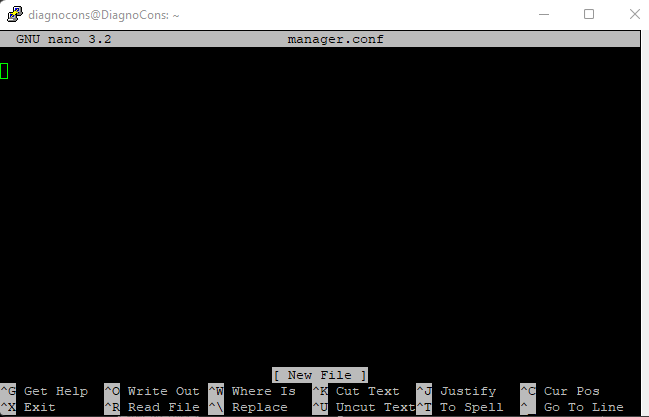
nano manager.conf



Nota: este es el archivo encargado de lo que es la configuración del Asterisk, si entramaos la primera vez veremos la documentación, que podremos leer, que algunos parámetros que están por default.

Como siempre lo movemos al bak con el sig. comando:

mv manager.conf manager.conf.bak



Viene la parte de las secciones, vienen parámetros del “general” y parámetros de los usuarios, podemos tener un usuario para conectarnos y leer eventos o ejecutar acciones, podemos tener dos usuarios para que cada uno haga diferentes eventos o diferentes acciones. Una vez aclarado eso escribiremos lo sig.:

[general]

enabled=yes (Estamos habilitando el manager, con eso ya estamos activando el AMI)

webenabled=no (Estamos indicando que podremos conectarnos vía web, en un inicio le daremos “no”)

port = 5038 (Este es el puerto por default)

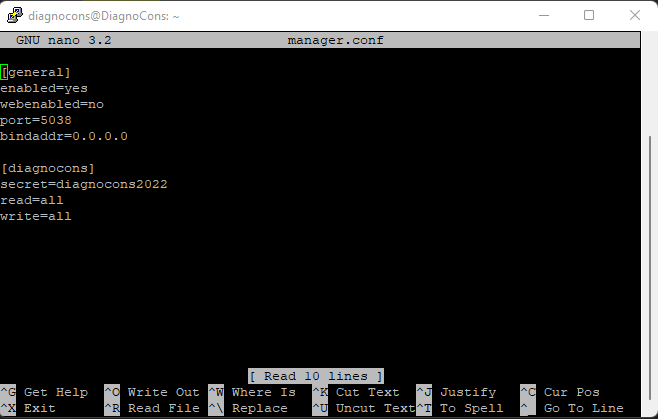
bindaaddr=0.0.0.0 (Aquí colocaremos la una IP determinada, localhost o la IP de la misma máquina para poder acceder, de momento dejaremos cualquier IP, que se pueda conectar de cualquier ubicación)

[diagnocons] (Usuario)

secret = diagnocons2022 (Contraseña)

read=all

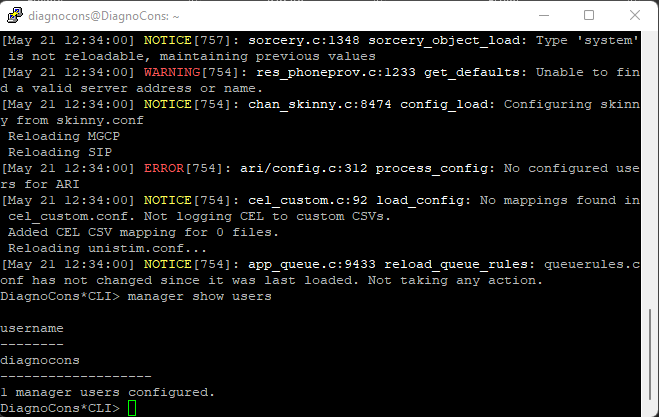
write=all (Estos dos últimos parámetros hacen referencia a los permisos como lo que son lectura y escritura, nosotros escogemos los permisos, se puede observar en la documentación)



Con esto ya estaría funcionando el manager.

Después entramos al Asterisk le escribimos un “relaod” y escribimos lo sig.:

manager show users



Nota: nos mostrara los usuarios correctamente configurados.

Sobre el manager podremos ejecutar acciones o se puede monitorear eventos, usando el comando “manager commands” entre ellos están:

* Login: para ingresar en el manager
  + Para ver su sintaxis podemos usar el comando:

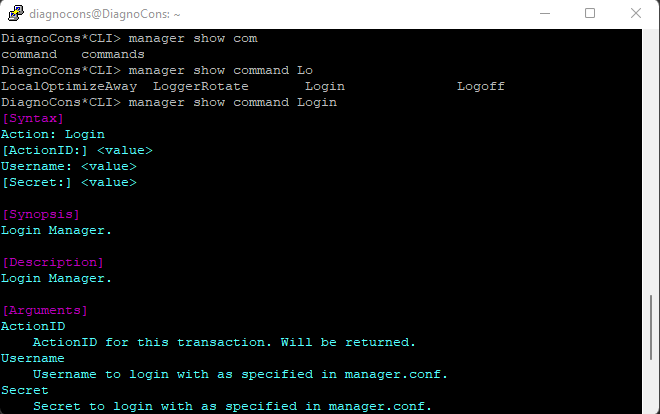
manager show command Login

Action: login

[ActionID:] <value>

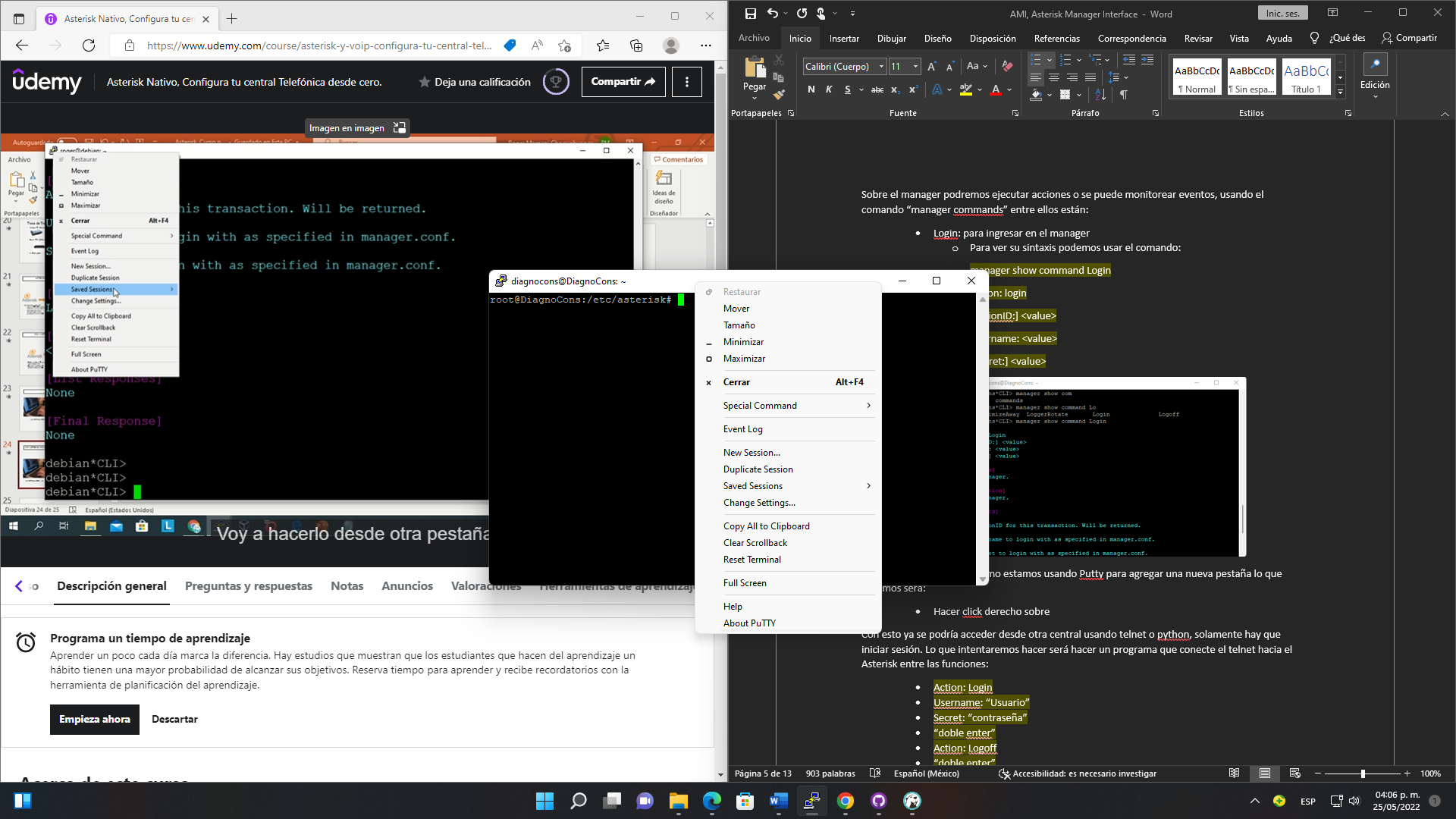
Username: <value>

[Secret:] <value>

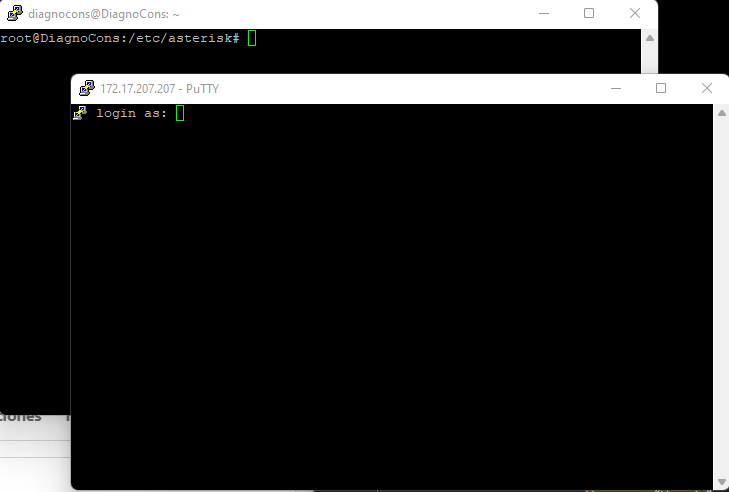


Ahora vamos a probarlo, como estamos usando Putty para agregar una nueva pestaña lo que haremos será:

* Hacer click derecho sobre la barra del programa y nos mostrara un menú.



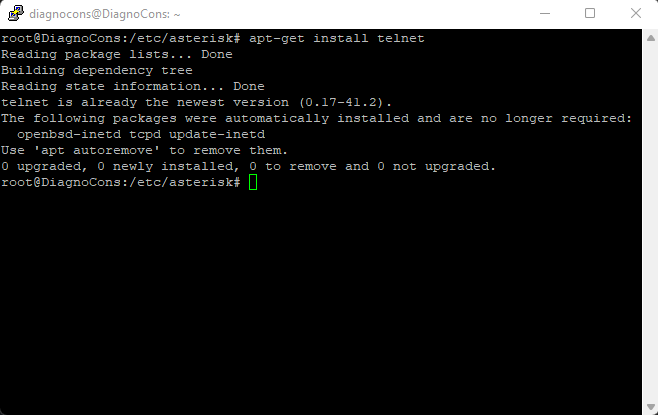
* Después hacemos click donde dice “Duplicate Session” y nos abrirá otra terminal (Es del mismo servidor).



* Después solo le colocamos las credenciales, una sesión debe de estar en la consola Asterisk.

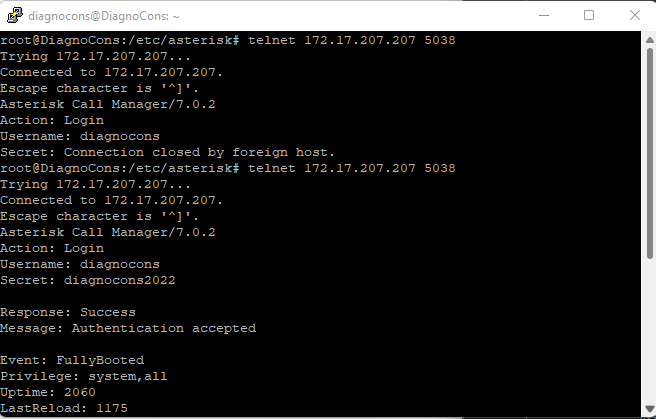
Después para continuar será necesario tener instalado telnet en caso de que no se tenga, lo instalaremos con el sig. comando:

apt-get install telnet



Ahora desde una sesión usando telnet nos loguaremos con el manager (Asterisk) para lo cual ingresaremos el sig. comando:

telnet “IP del servidor” “Puerto default”

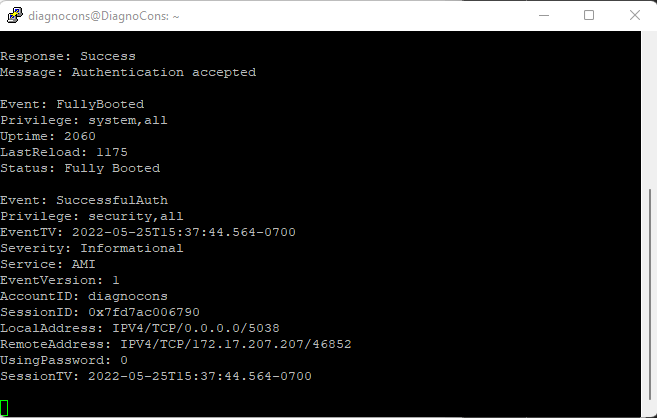


Una vez colocado es importante que lo sig. lo coloquemos de manera rápida o nos cerrara la sesión:

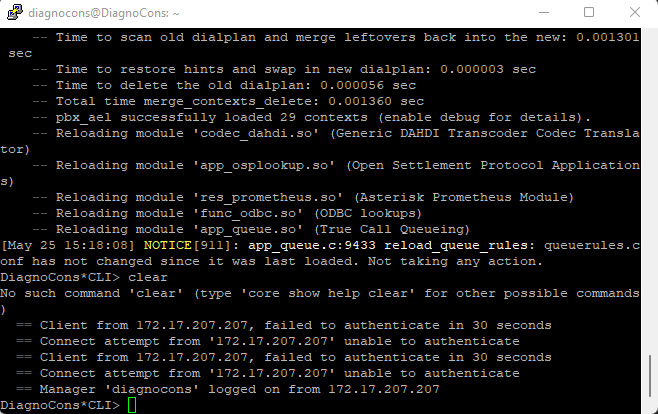
* Action: Login
* Username: “Usuario”
* Secret: “contraseña”
* “doble enter”

Nota: es importante se escriba tal cual es y de forma rápida.

Con esto ya habremos ingresado al manager de forma exitosa.

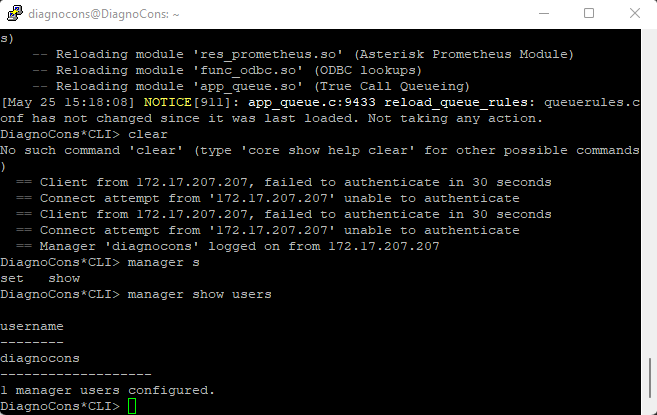


En la otra sesión donde ingresamos a la consola Asterisk nos debe salir un mensaje del usuario que se ha conectado y su IP.

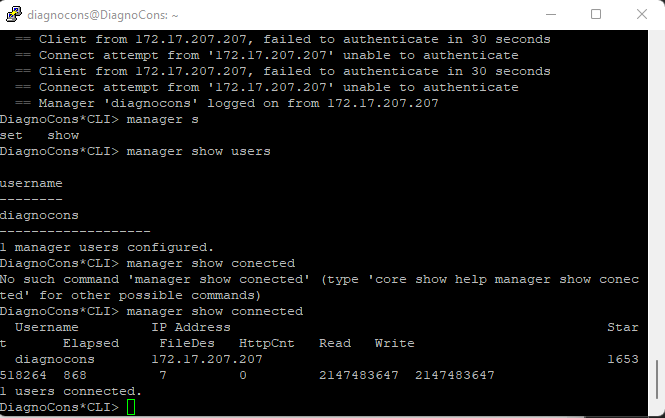


Con respecto al manager contamos con las sigs. funciones:

* manager show users: nos mostrara los usuarios que hay en el manager.

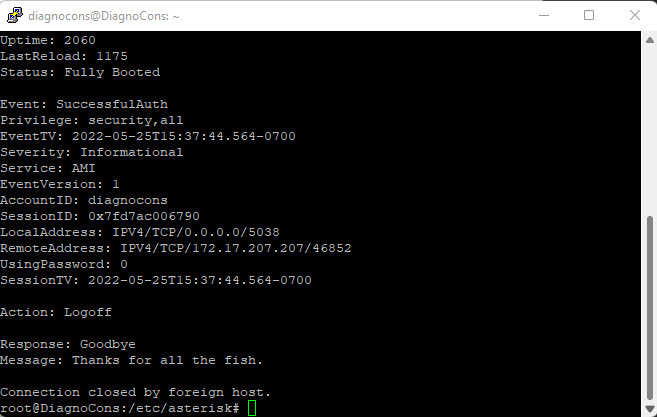


* manager show conected: nos mostrara los usuarios conectados.



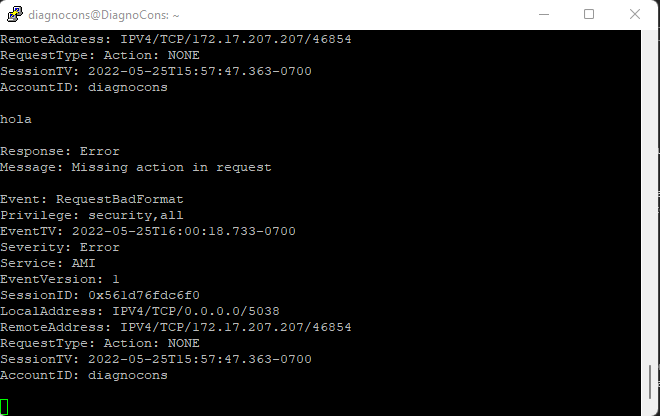
Para cerrar la sesión lo único que hay que hacer es escribir lo sig.:

* Action: Logoff
* “doble enter”

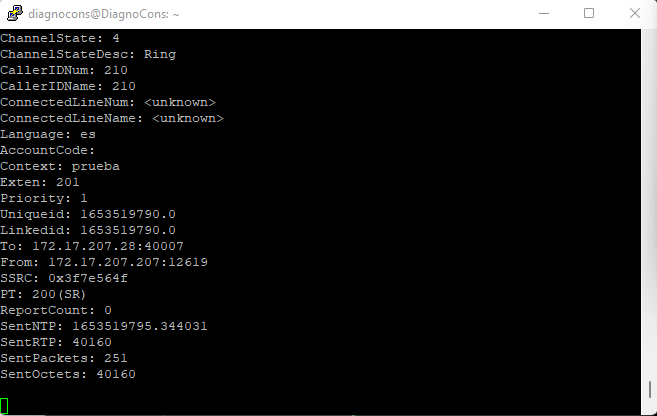


# Ejecutando Acciones y leyendo eventos desde telnet

Lo primer será habernos logueado. Una vez logueado empezara a escuchar automáticamente todo lo que está pasando.

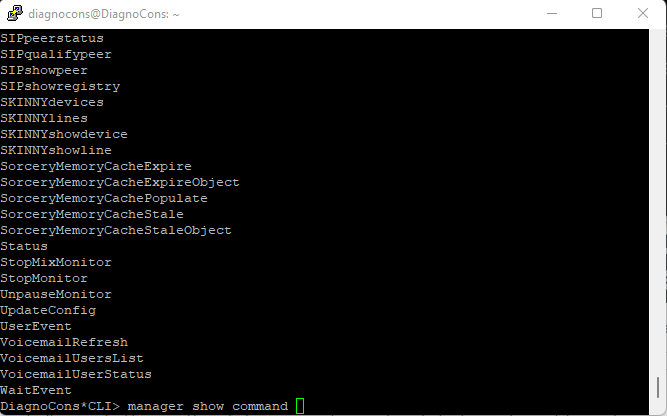


Va actualizarse constantemente, desde que se enciende una terminal e incluso cuando se inicia o termina una llamada.

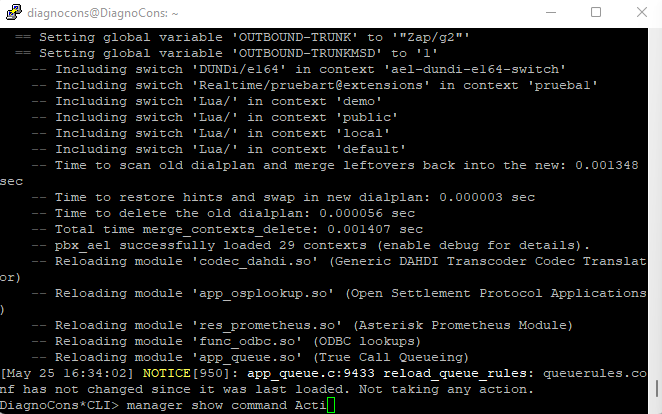


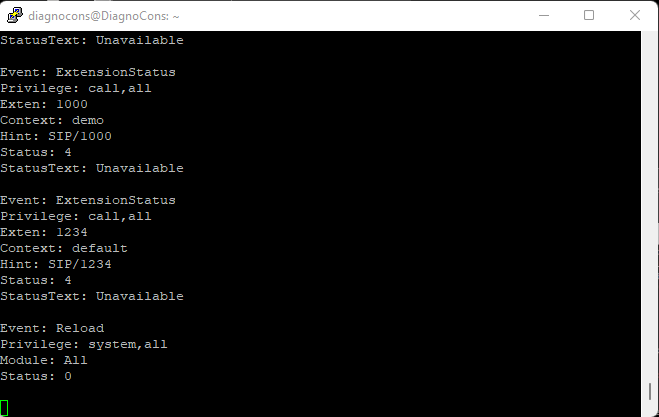
Para ver todas las acciones que se pueden hacer con el manager usaremos el sig. comando (desde la consola asterisk):

manager show command “tab”



Cada vez que en el manager hagamos una acción automáticamente se verá reflejado en la consola Asterisk.





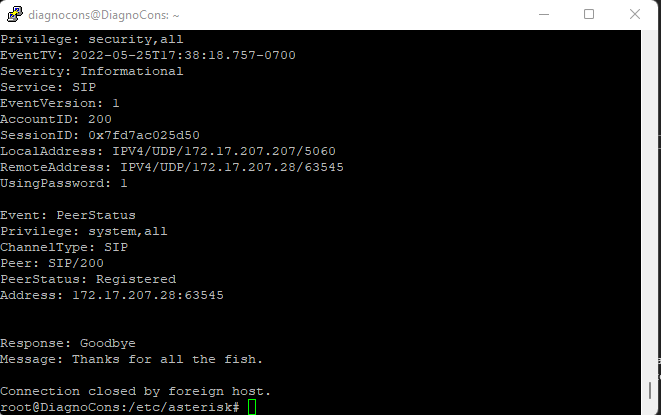
Por ejemplo, para originar una llamada usaremos la accion “Originate” la cual realizara una llamada, si queremos marcar al 200, escribiremos lo sig.:

* Action: Originate
* Channel: SIP/200
* Context: prueba
* Exten: 902
* Priority: 1



Para terminar, simplemente hacemos un “Logoff”.

* Action: Logoff
* “doble enter”



# Login al asterisk desde Python usando AMI

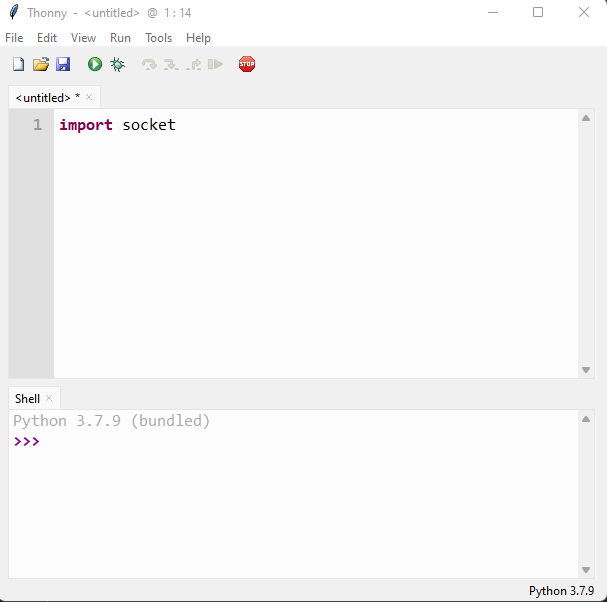
Para empezar, usaremos el programa IDE thommy. Lo que intentaremos hacer será hacer un programa que conecte el telnet hacia el Asterisk recreando las sigs. acciones:

* Action: Login
* Username: “Usuario”
* Secret: “contraseña”
* “doble enter”
* Action: Logoff
* “doble enter”

En una instancia, usaremos “sockets” en lugar de “librerías” para observar cómo sale todo este proceso, y después se usará las “librerías” las librerías para simplificar el trabajo.

Lo primero que necesitaremos es la librería para conectarnos a través de red llamada “Socket”, por lo cual escribiremos:

import socket



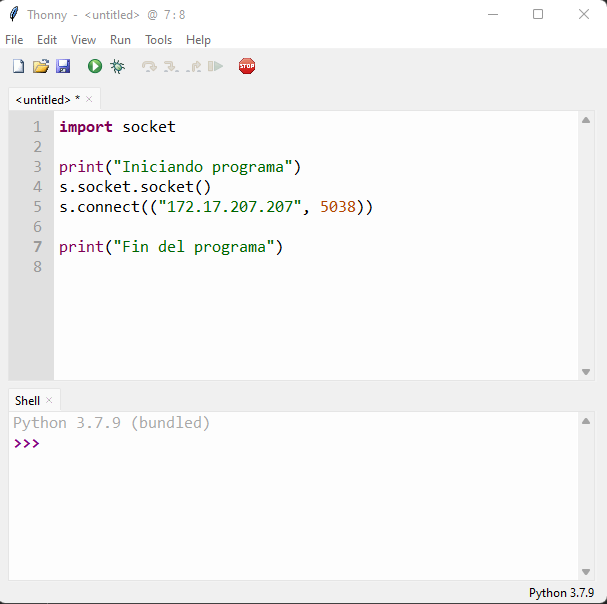
Después definimos el objeto “socket.socket” y después nos conectamos al manager de la sig. manera:

print(“Iniciando programa”)

s=socket.socket()

s.connect((“IP”, Puerto)) (El puerto 5038 es el puerto por default)

print(“Fin del programa”)



Después guardamos el archivo y le ejecutamos en el botón verde.

Hasta ahora lo que hemos hecho ha sido crear el socket y nos mandamos conectar, pero como el límite son 30 segundos para loguearse nos finalizó la sesión porque no nos hemos autenticado.

Después sigue el “Login” para lo cual le agregaremos lo sig.:

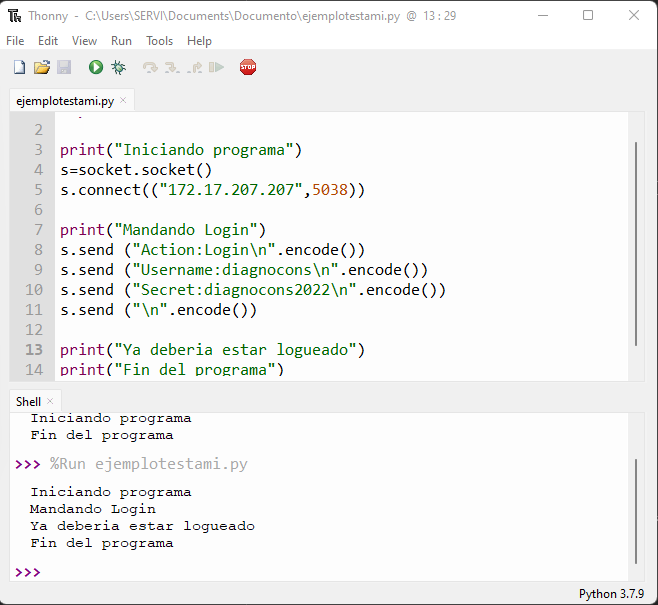
print("Mandando Login")

s.send ("Action:Login\n".encode())

s.send ("Username:diagnocons\n".encode())

s.send ("Secret:diagnocons2022\n".encode())

s.send ("\n".encode())

print("Ya deberia estar logueado") 

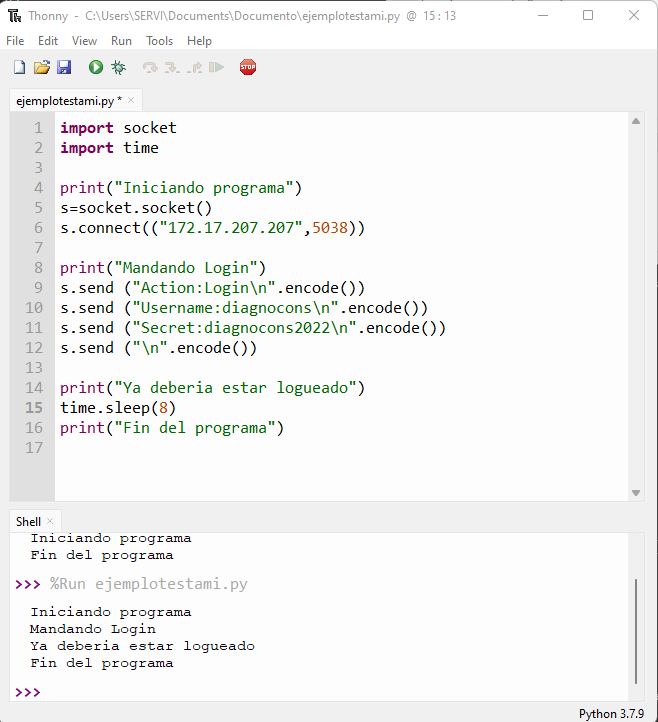
Nota: el "\n" sirve para enviar lo que es un “enter”, “s.send sirve para mandar las acciones”.

Después importaremos una librería llamada “time” de la sig. manera:

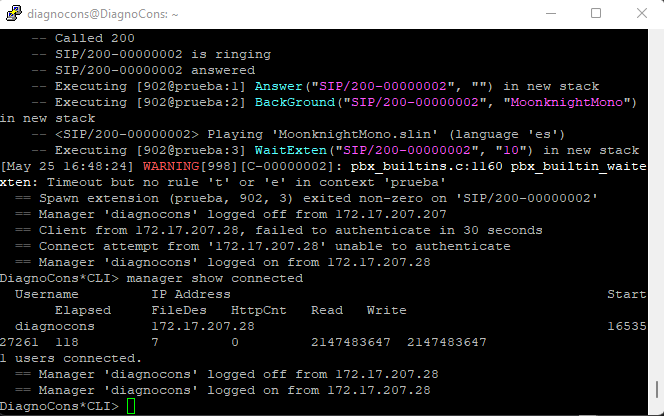
import time

Después de la parte que dice “Ya debería estar logueado” le agregaremos por asi decirlo un contador, que en este caso demorara ocho segundos como muestra de ello.

time.sleep(8)



Esto último seria que después del “Login” pasarían ocho segundos y después vendría la parte de “Fin de programa”, luego simplemente lo probamos.



Como vemos funciona, pero para este punto nuestro programa solo ha logueado, pero aun no finaliza la sesión por ello modificaremos lo sig.:

Eliminamos el print “Ya deberia estar logueado”

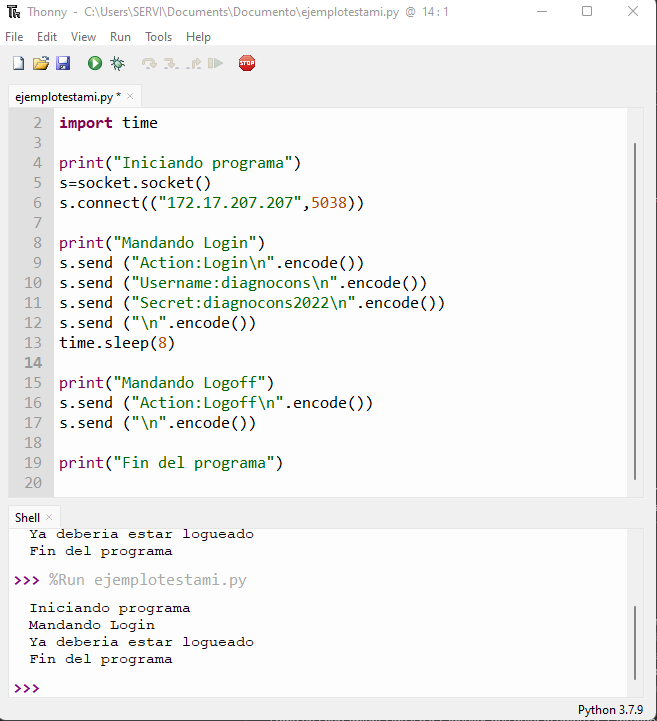
Subimos el “time.sleep”

Despues colocamos lo sig.

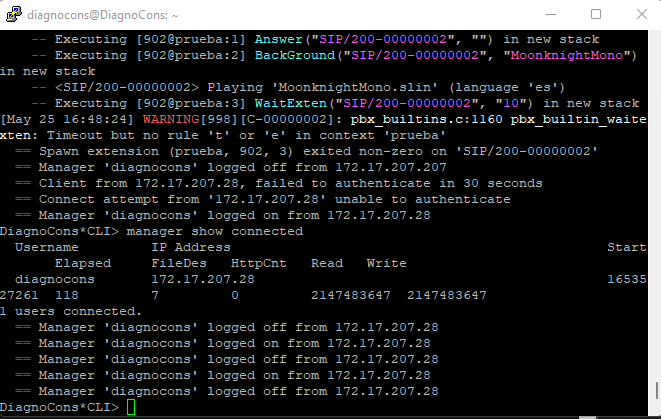
print("Mandando Logoff")

s.send ("Action:Logoff\n".encode())

s.send ("\n".encode())



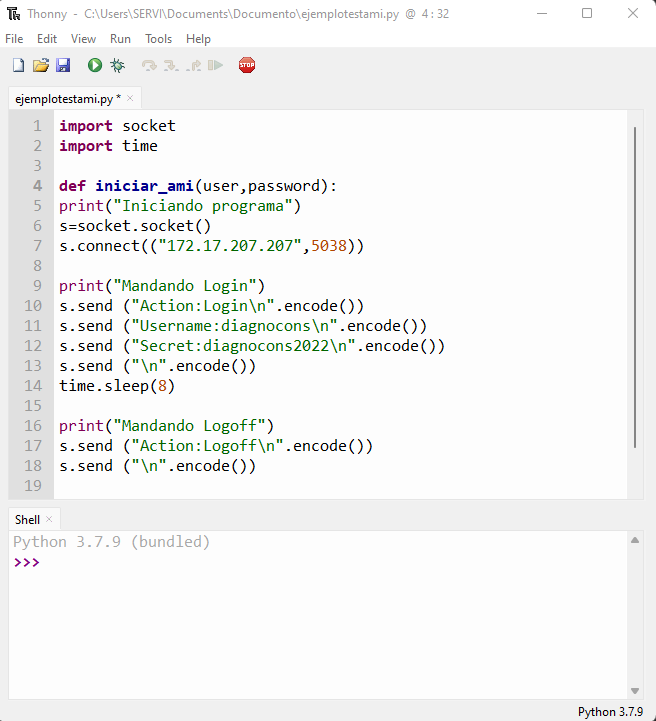
Esto significa que una vez logueados, pasaran ocho segundos y finalizara la sesión.



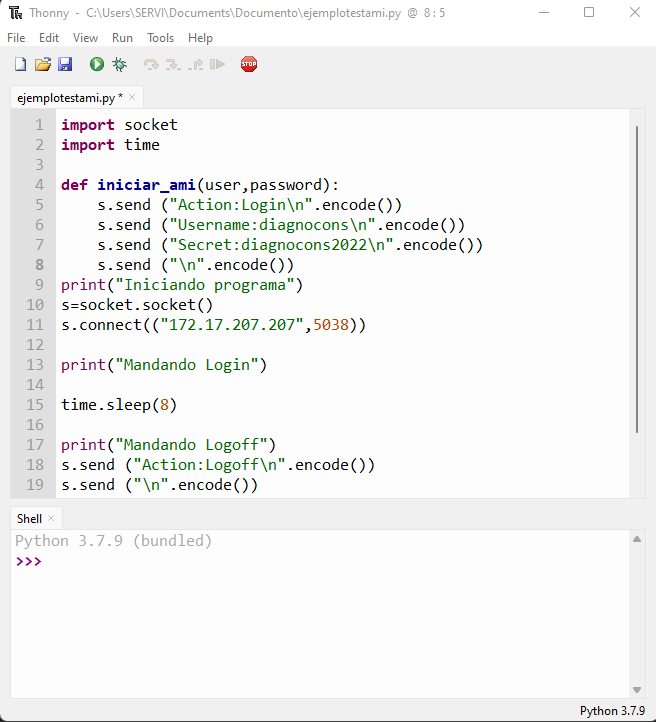
Una vez visto que funciona, ahora vamos a hacerlo de una manera mas ordenada y lo haremos de la sig. manera:

* Creamos una función llamada “iniciar\_ami” y lo que necesita para iniciar AMI que es el “user” y el “password”

def iniciar\_ami(user,password) :



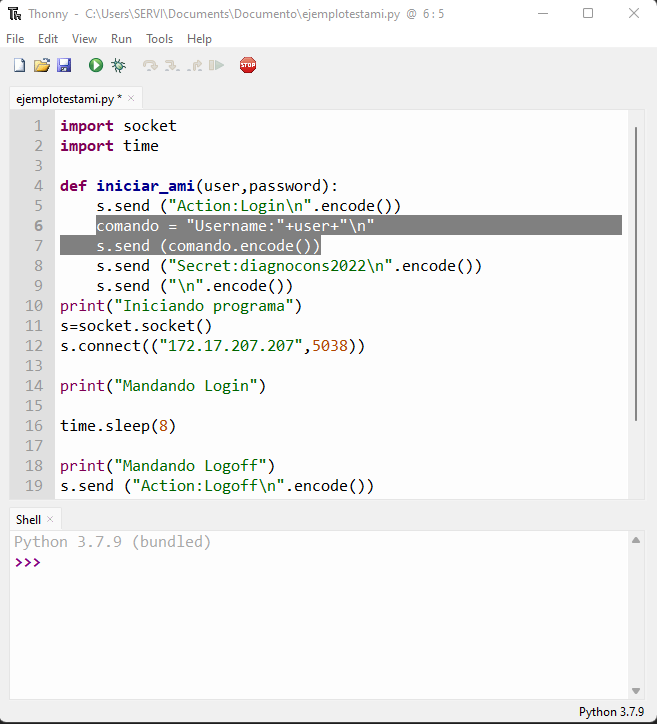
* Después enseguida de la función movemos todos los “s.send”.



* Ahora modificaremos el “s.send” de “Username”, lo que haremos será antes del “Username” le añadiremos lo sig.:

comando = "Username:"+user+"\n"

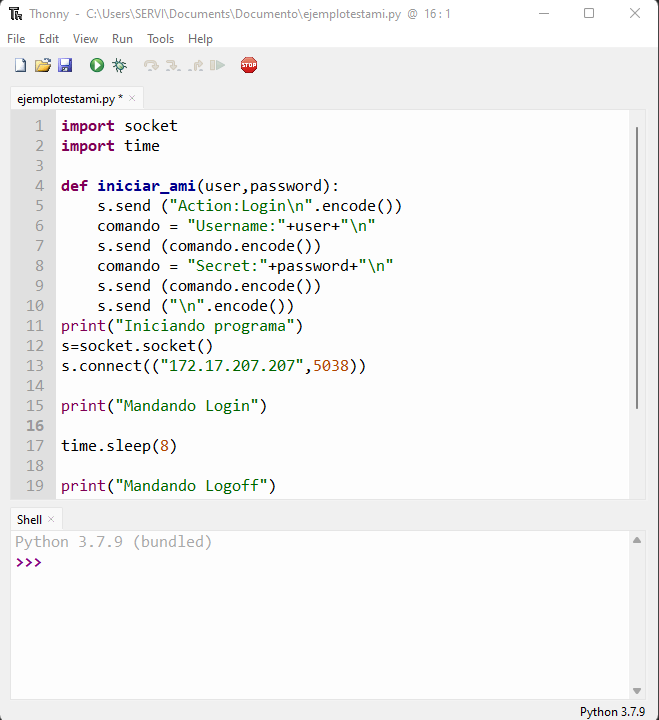
s.send (comando.encode())



* Después volveríamos a hacer lo mismo, pero ahora para la contraseña, es decir el “Secret”.

comando = "Secret:"+password+"\n"

s.send (comando.encode())

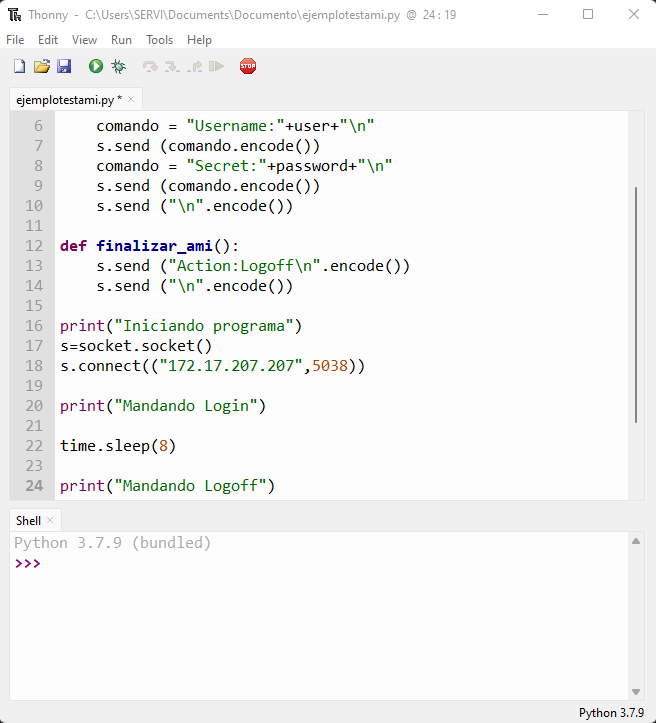


* Después definimos una nueva función llamada “finalizar\_ami” y después pasamos lo referente a “Logoff”.

def finalizar\_ami():

s.send ("Action:Logoff\n".encode())

s.send ("\n".encode())



* Después escribimos las funciones de la sig. manera:

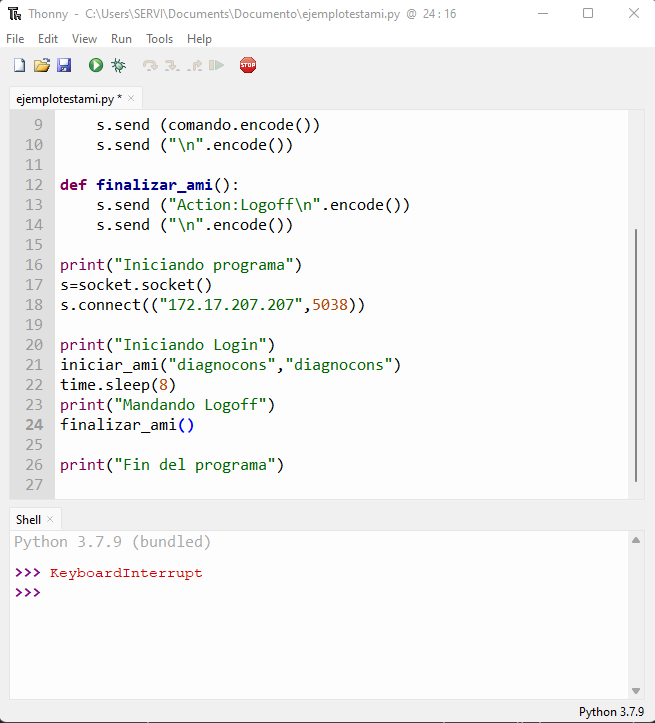
print("Iniciando Login")

iniciar\_ami("diagnocons","diagnocons")

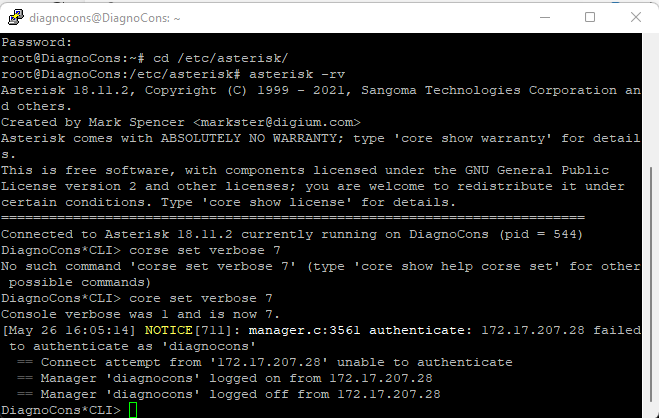
time.sleep(8)

print("Mandando Logoff")

finalizar\_ami()



* Después simplemente corremos el programa.



# Click to Dial usando Python con AMI.

Esta vez vamos a hacer un click to dial, es decir que presione un botón y haga una llamada. Para ello vamos a completar las funciones “iniciar\_ami” y ”finalizar\_ami” donde vamos a adicionarle que puedan hacer llamada con los sig. parámetros:

Action: Originate

Channel: SIP/200 (Es el unico parametron que va a variar)

Context: prueba

Exten: 902

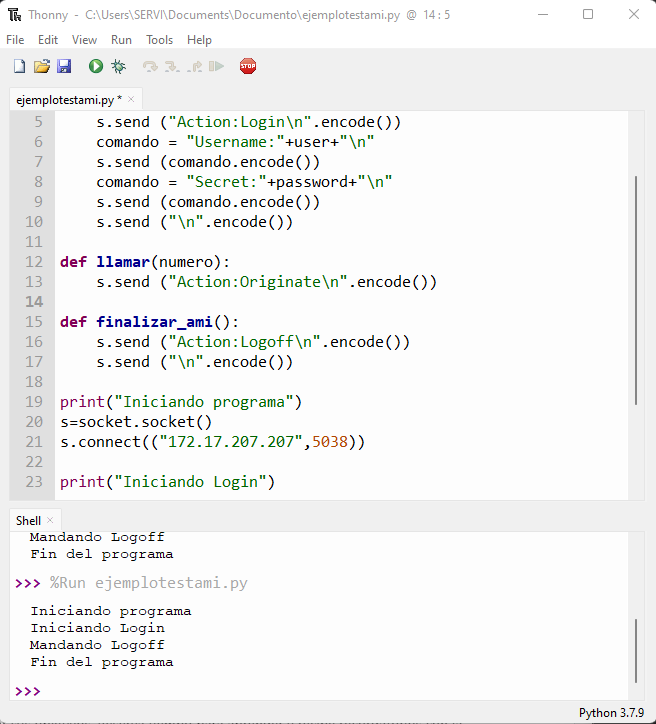
Priority: 1

Lo primero que haremos será definir una función llamada “llamar” que va a recibir el “numero” y dentro de esta función vamos a mandar los comandos, escribiendo lo sig.:

* Empezamos con la parte de “Action: Originate” escribiendo lo siguiente:

def llamar(numero):

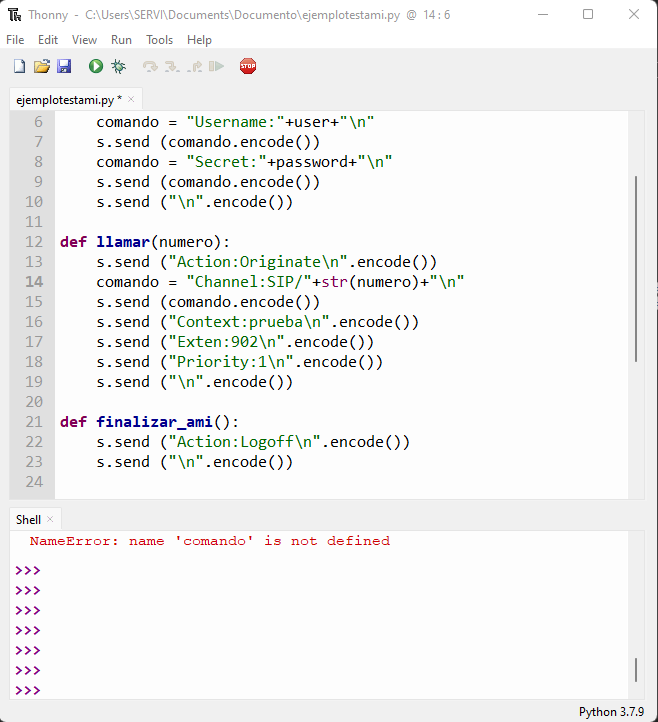
s.send ("Action:Originate\n".encode())



* Después para el “Channel:SIP/200” lo agregaremos como comando, donde la variable “numero” es el numero de la terminal de la sig. manera:

comando = "Channel:SIP/"+str(numero)+"\n"

s.send (comando.encode())



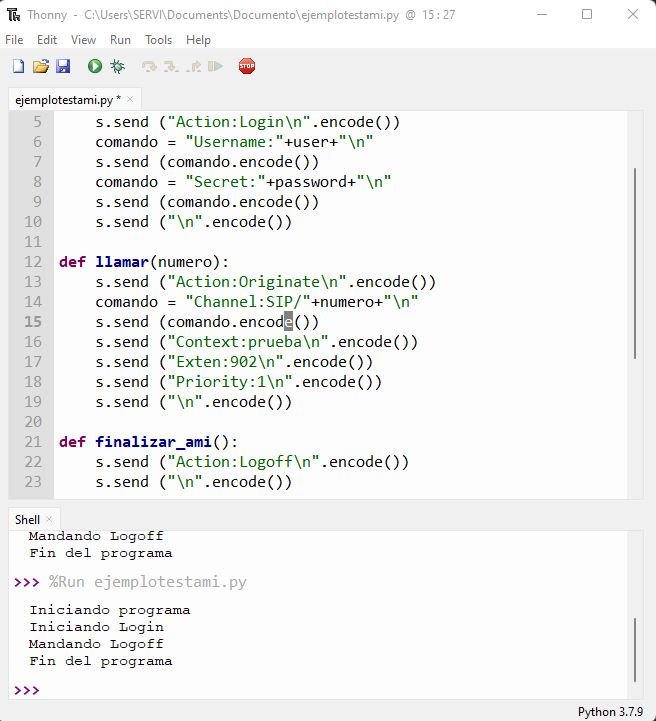
* Luego enviamos los demás valores:

s.send ("Context:prueba\n".encode())

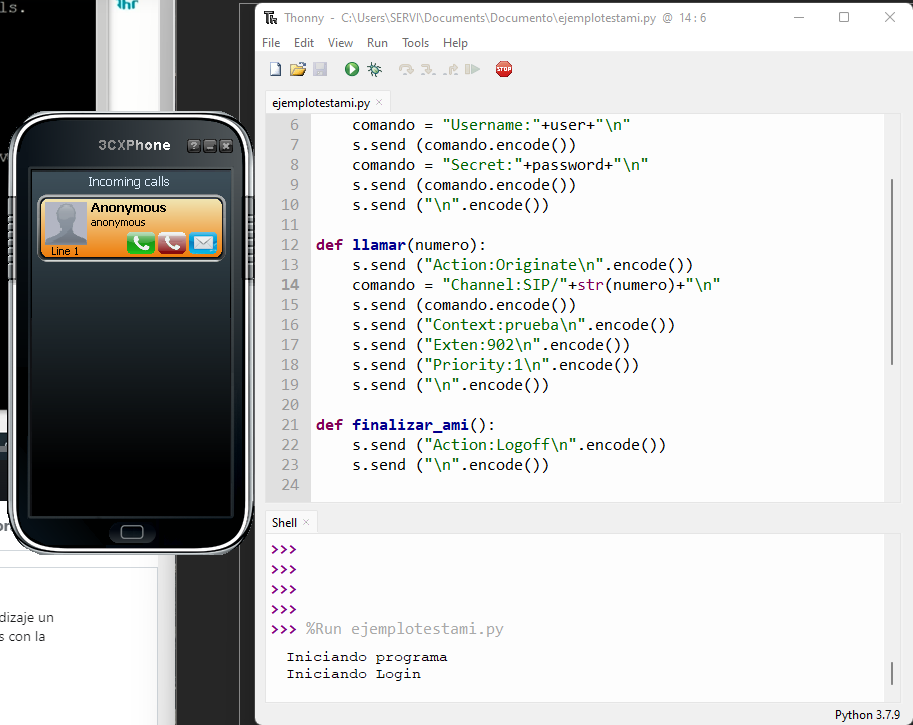
s.send ("Exten:902\n".encode())

s.send ("Priority:1\n".encode())

s.send ("\n".encode())

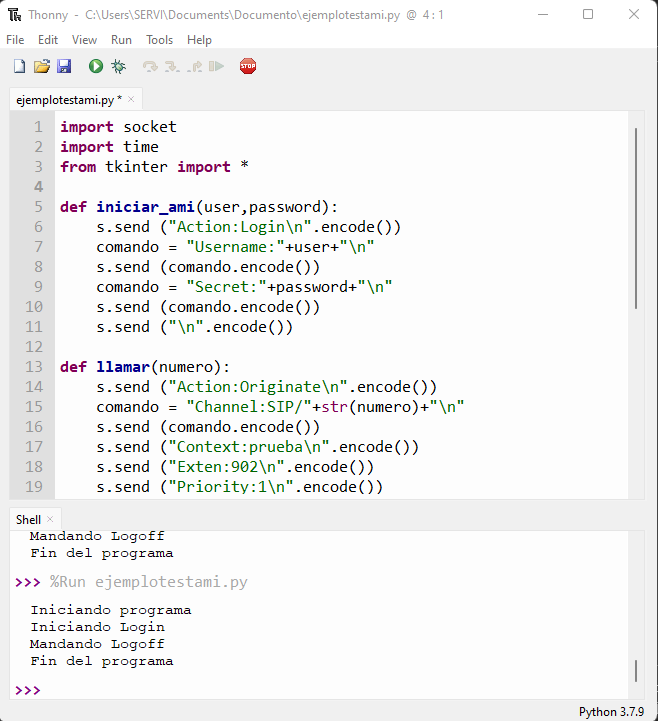


* Después vamos a probarlo, para ello colocamos la función llamar, después del “Login” donde le daremos el valor de 200, para que llame a dicha extensión.



Una vez terminado eso, pasaremos a hacer la aparte gráfica, lo primero se importa la librería “tkinter” donde importaremos todo de la sig. manera:

from tkinter import \*



Luego en la parte de abajo creamos la ventana escribiendo lo sig.:

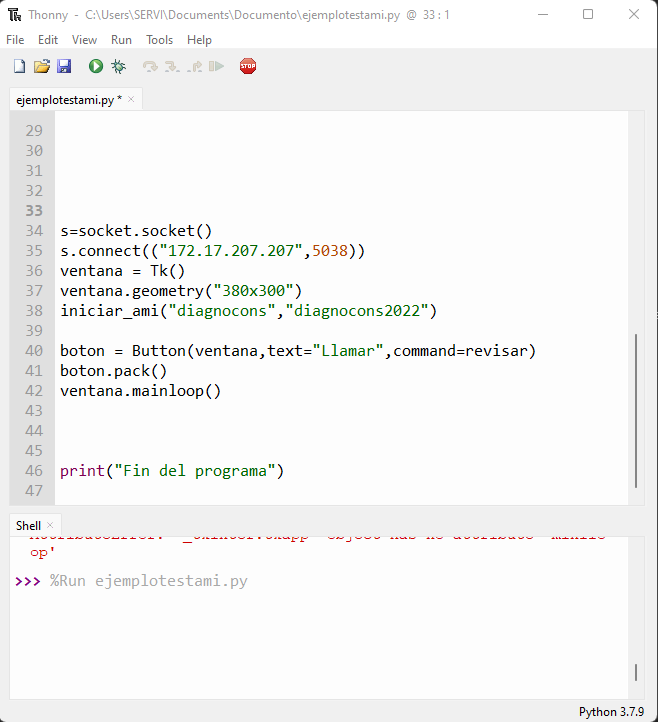
ventana = Tk() (Generamos ventana)

ventana.geometry(“380x300”) (Definimos el tamaño de la ventana)

boton = Button(ventana,text=”Llamar”, command=revisar) (Creamos un boton, el cual se tendrá un texto “Llamar” dentro y el “comand” hará funcionar la función “Revisar” que posteriormente será creada)

boton.pack()

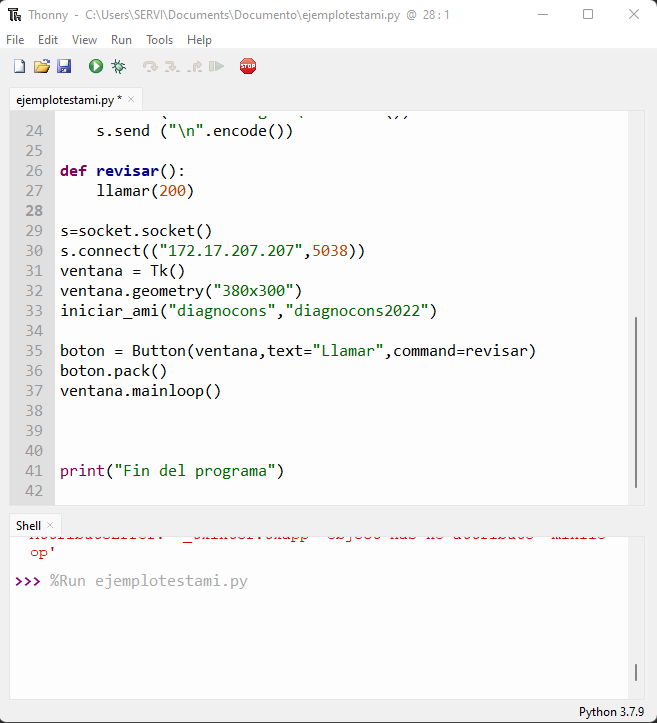
ventana.mainloop()



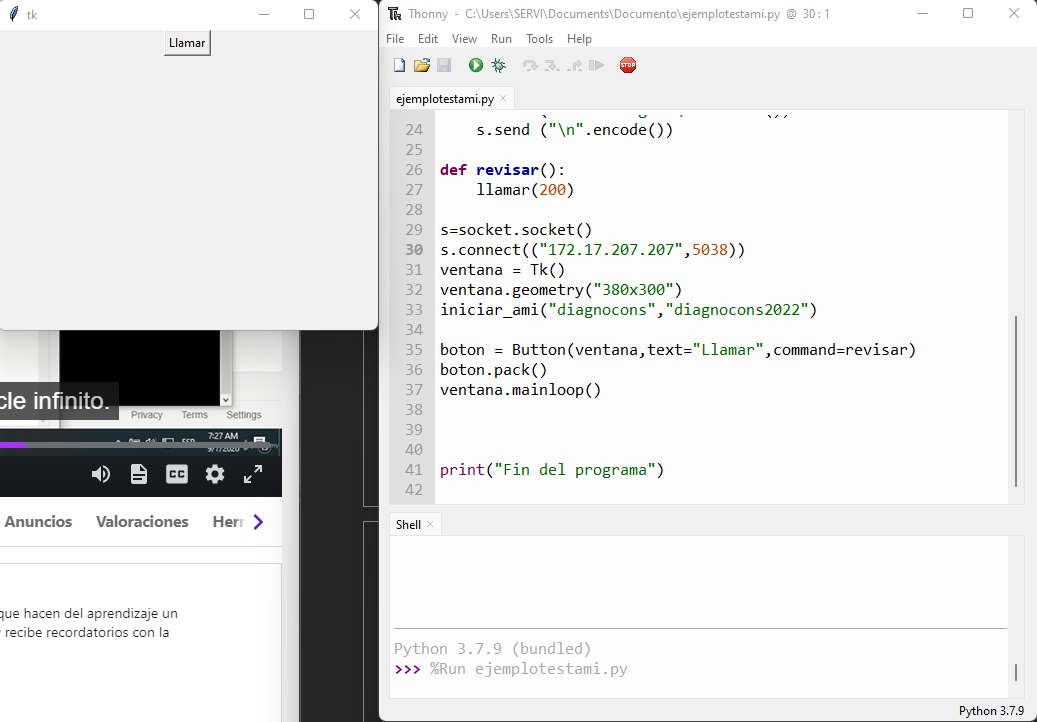
Después definimos una función llamada “Revisar” donde colocaremos la función “llamar” y a la vez subiríamos la función “iniciar\_ami” y solo por un momento ocultamos el “time.sleep” y el “finalizar\_ami” de la sig. manera:

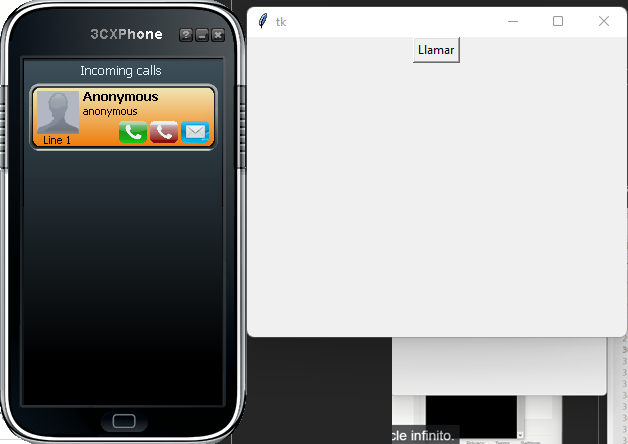
def revisar():

llamar(200)



Después solamente probamos y ya debería ya salir la ventana.





# Python

<https://openwebinars.net/blog/que-es-python/>

Es un lenguaje de programación versátil multiplataforma y multiparadigma que se destaca por su código legible y limpio. Una de las razones de su éxito es que cuenta con una licencia de código abierto que permite su utilización en cualquier escenario. Esto hace que sea uno de los lenguajes de iniciación de muchos programadores siendo impartido en escuelas y universidades de todo el mundo.

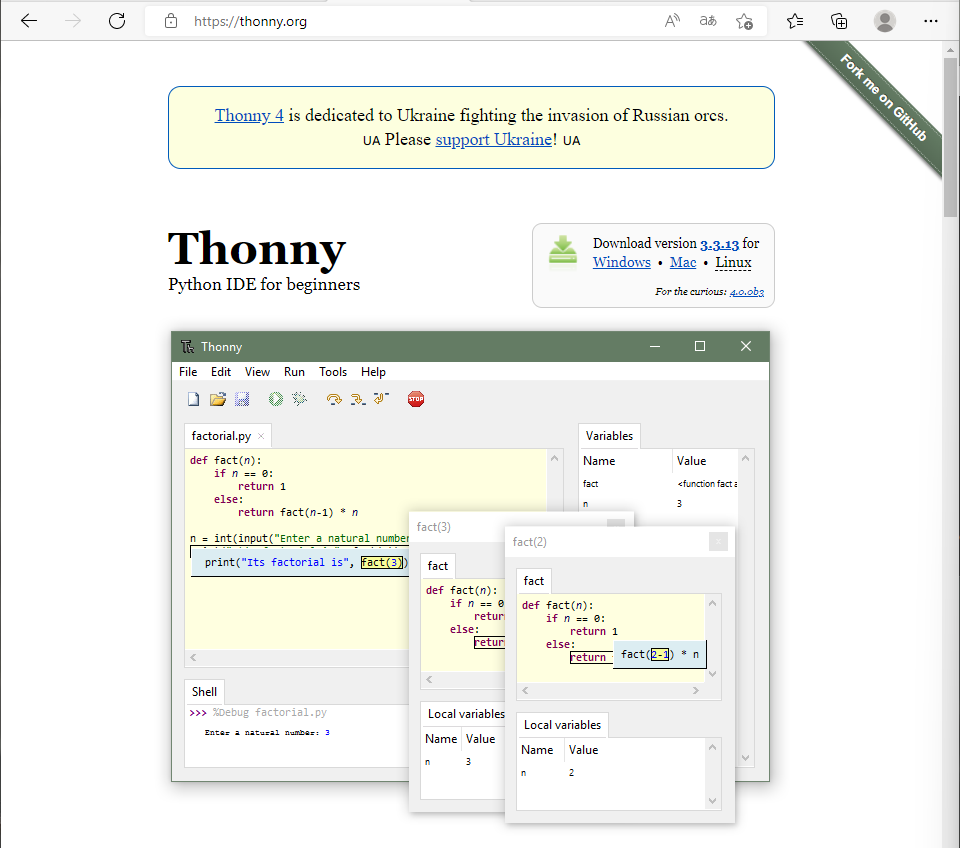
# Thonny

https://www.muylinux.com/2018/02/19/thonny-ide-python/

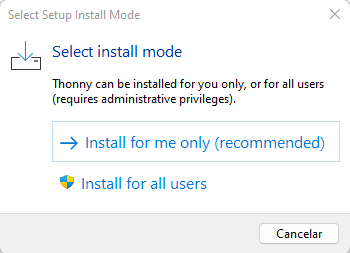
Python es uno de los lenguajes más recomendados para adentrarse en el mundo de la programación, gracias a que su sintaxis es muy sencilla y está tipado dinámicamente (aunque fuertemente). Estos factores lo vuelven mucho más accesible que otros lenguajes también muy extendidos, como Java, C# y C++, siendo posiblemente el último demasiado difícil para empezar. De ahí surge Thonny, se trata de un sencillo IDE de Python amigable con el novato, y como ya hemos comentado que este lenguaje es ideal para iniciarse en el sector, se lo podría considerar como una herramienta ideal para aquellos que quieran empezar a aprender a programar, ofreciendo además versiones para GNU/Linux, Windows y Mac.

## Como descargar Thonny.

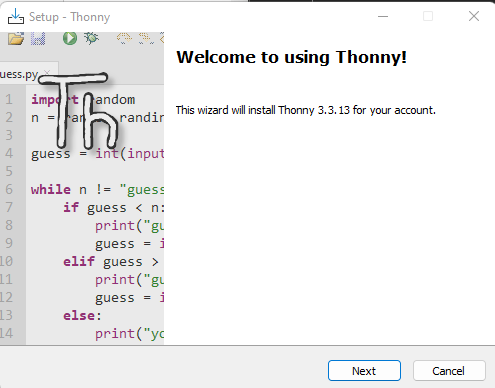
Lo primero su ir a su enlace oficial “<https://thonny.org/> “ y descargarlo.



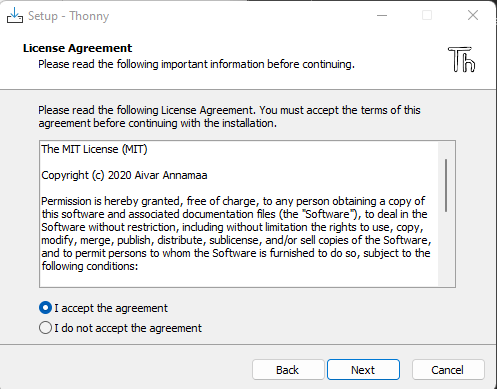
Una vez descargado abrimos el instalador, saldrá un cuadro y le damos en “Install for me only”.



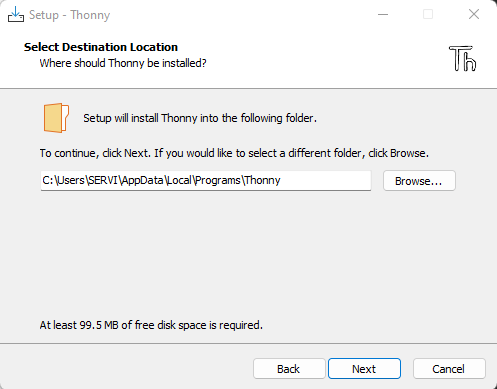
Saldrá el cuadro de instalación y le damos en “Next”.



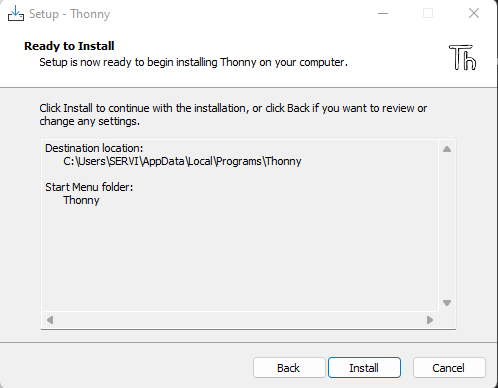
Aceptamos los términos de licencia y le damos en “Next”.



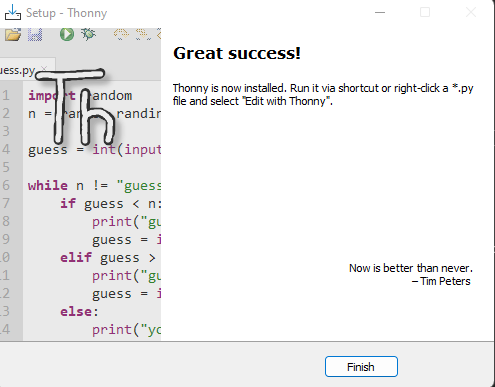
Elegimos la ruta de instalación (se recomienda dejar la de defecto) y le damos “Next”.



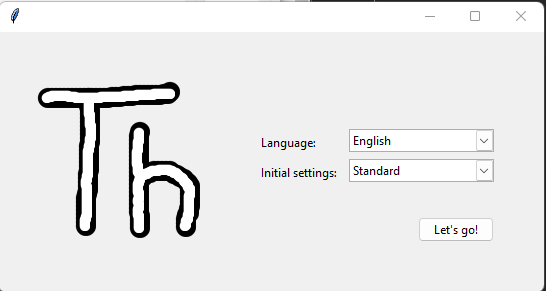
Después a lo que resta le damos “Next” hasta que la parte que diga “Install” y le hacemos click.



Listo con eso habremos instalado Thommy.



Después tocará configurarlo, una vez al abrirlo, nos pedirá el lenguaje (Lo recomendado es dejarlo en inglés).



Listo con eso tendremos Thommy listo.

