**UNIVERSIDAD MARIANO GALVÉZ DE GUATEMALA**

**SEDE MAZATENANGO**

**INTELIGENCIA ARTIFICIAL**

**ING. MIGUEL LEMUS**

# PROYECTO: CHATBOT

**PEDRO ALEJANDRO SANDOVAL SANTIAGO**

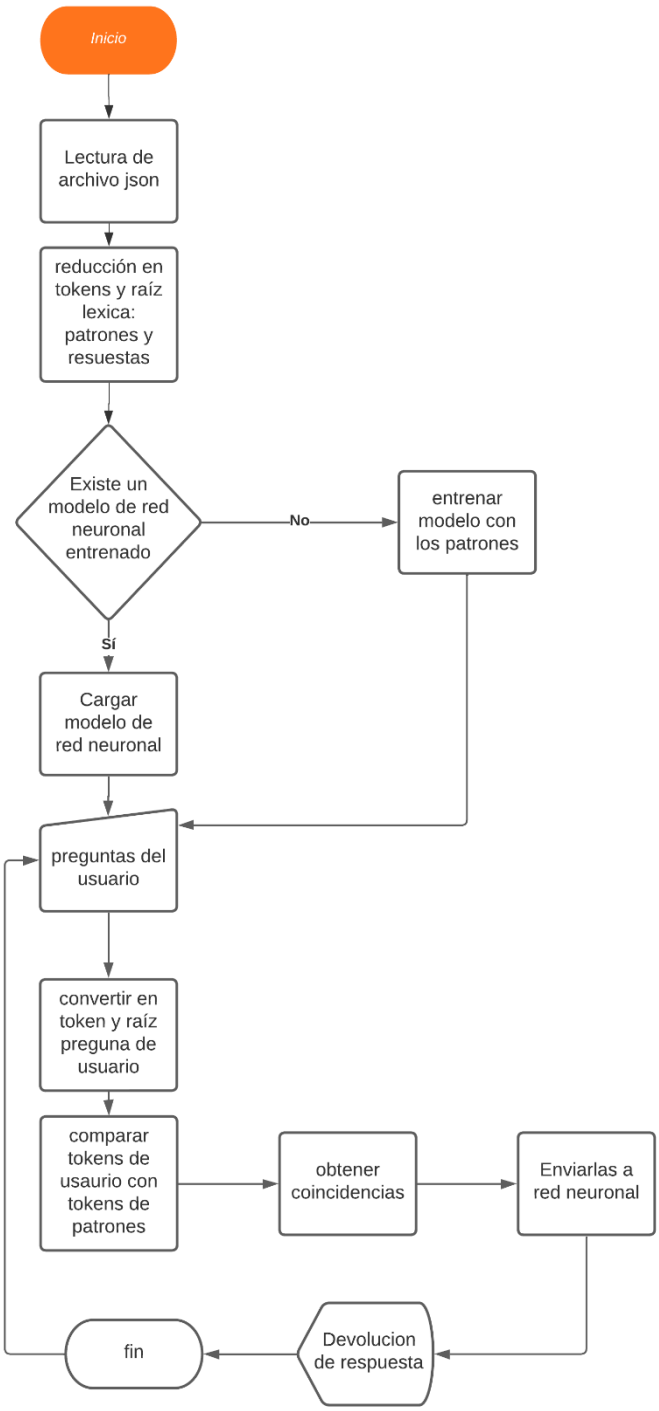
**3090-18-10443**

**4 DE JUNIO DE 2022**

# INTRODUCCIÓN

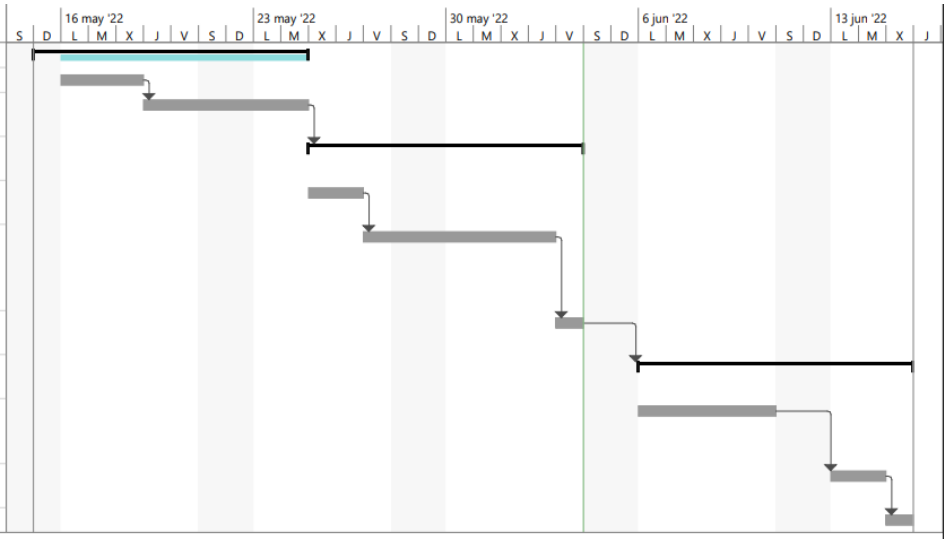
Las formas en cómo se desarrolla todo aspecto en la actualidad dependen directamente de la tecnología en su mayoría dentro de nuestra sociedad, de forma que la investigación y la creación de nueva tecnología siempre está presente en nuestro diario vivir. Así pues, es importante aprender a manejarla y aplicarla, basado en ello es que se ha presentado el proyecto de chatbot el cual sirve como un prototipo de solución a la automatización de las respuestas de información general de las carreras, requisitos y pasos de inscripción de la Universidad Mariano Gálvez de Guatemala en la sede Mazatenango, Suchitepéquez.

# DIAGRAMA DE FLUJO DE LA APLICACIÓN



# CRONOGRAMA DE ACTIVIDADES

| Nombre de tarea | Duración | Comienzo | Fin |
|---|---|---|---|
| **Inicio de investigacion chatbot** | **8 días** | **dom 15/05/22** | **mar 24/05/22** |
| Abstraccion de los procesos | 3 días | lun 16/05/22 | mié 18/05/22 |
| Instalación de librerías y resolución de problemas | 4 días | jue 19/05/22 | mar 24/05/22 |
| **Implementacion principales librerias** | **8 días** | **mié 25/05/22** | **vie 3/06/22** |
| Aplicación de metodos con JSON | 2 días | mié 25/05/22 | jue 26/05/22 |
| Aplicación de Redes Neuronales con tensorflow y tflearn(entrenamiento, seguimiento y pruebas) | 5 días | vie 27/05/22 | jue 2/06/22 |
| Aplicación de numpy (metodos) | 1 día | vie 3/06/22 | vie 3/06/22 |
| **Ajuste de codigo, red neuronal y json** | **8 días** | **lun 6/06/22** | **mié 15/06/22** |
| Añadir más categorías de reconocimiento a json y alimentar la red neuronal | 5 días | lun 6/06/22 | vie 10/06/22 |
| Creacion total de la funcion principal(usuario) | 2 días | lun 13/06/22 | mar 14/06/22 |
| presentacion avances | 1 día | mié 15/06/22 | mié 15/06/22 |



## ARQUITECTURA DE LA APLICACIÓN

La arquitectura utilizada para este proyecto es basada en componentes de acuerdo a que cada funcionamiento está definido por sus características que nos ayudan a poder separar de una manera correcta cada uno de los módulos según su funcionamiento.

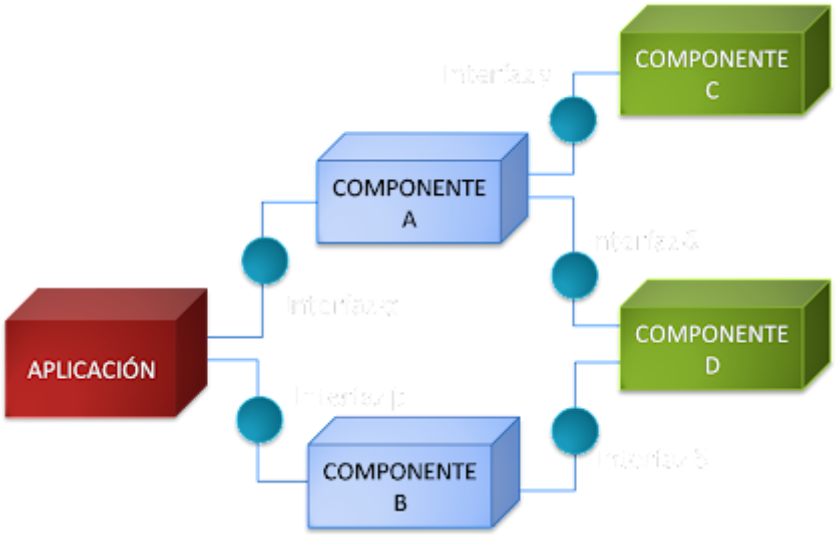
Una arquitectura basada en componentes describe una aproximación de ingeniería de software al diseño y desarrollo de un sistema. Esta arquitectura se enfoca en la descomposición del diseño en componentes funcionales o lógicos que expongan interfaces de comunicación bien definidas. Esto provee un nivel de abstracción mayor que los principios de orientación por objetos y no se enfoca en asuntos específicos de los objetos como los protocolos de comunicación y la forma como se comparte el estado.

El estilo de arquitectura basado en componentes tiene las siguientes características:

• Es un estilo de diseño para aplicaciones compuestas de componentes individuales.

• Pone énfasis en la descomposición del sistema en componentes lógicos o funcionales que tienen interfaces bien definidas.

• Define una aproximación de diseño que usa componentes discretos, los que se comunican a través de interfaces que contienen métodos, eventos y propiedades.



# LIBRERÍAS

## Librería NLTK

```
import nltk as nt
```

El Kit de herramientas de lenguaje natural (NLTK) es una biblioteca de Python de código abierto para el procesamiento del lenguaje natural.

NLTK es una plataforma líder para construir programas Python para trabajar con datos de lenguaje humano. Proporciona interfaces fáciles de usar para más de 50 recursos corporativos y léxicos como WordNet, junto con un conjunto de bibliotecas de procesamiento de texto para clasificación, tokenización, stemming, etiquetado, análisis y razonamiento semántico, envoltorios para bibliotecas de PNL de fuerza industrial y un foro de discusión activo.

## Librería Tensorflow

```
import tensorflow as tf
```

TensorFlow es una plataforma de código abierto de extremo a extremo para el aprendizaje automático. Cuenta con un ecosistema integral y flexible de herramientas, bibliotecas y recursos de la comunidad que permite que los investigadores innoven con el aprendizaje automático y los desarrolladores creen e implementen aplicaciones con tecnología de AA fácilmente

La principal biblioteca de código abierto para enseñarte a desarrollar y entrenar modelos de AA. Comienza enseguida y ejecuta notebooks de Colab directamente en tu navegador.

## Librería Tflearn

```
import tflearn as tl
```

TFlearn es una biblioteca de aprendizaje profundo modular y transparente construida sobre Tensorflow. Fue diseñado para proporcionar una API de nivel superior a TensorFlow con el fin de facilitar y acelerar las experimentaciones, sin dejar de ser totalmente transparente y compatible con él.

Las características de TFLearn incluyen:

- API de alto nivel fácil de usar y comprender para implementar redes neuronales profundas, con tutoriales y ejemplos.
- Prototipado rápido a través de capas de redes neuronales integradas altamente modulares, regularizadores, optimizadores, métricas...

### Librería Numpy

```
import numpy as np
```

NumPy es una librería de Python especializada en el cálculo numérico y el análisis de datos, especialmente para un gran volumen de datos. Incorpora una nueva clase de objetos llamados arrays que permite representar colecciones de datos de un mismo tipo en varias dimensiones, y funciones muy eficientes para su manipulación.

La ventaja de Numpy frente a las listas predefinidas en Python es que el procesamiento de los arrays se realiza mucho más rápido (hasta 50 veces más) que las listas, lo cual la hace ideal para el procesamiento de vectores y matrices de grandes dimensiones.

### Librería Json

```
import json
```

JSON, acrónimo de JavaScript Object Notation, es un formato de texto ligero para el intercambio de datos. JSON es un subconjunto de la notación literal de objetos de JavaScript, aunque hoy, debido a su amplia adopción como alternativa a XML, se considera un formato de lenguaje independiente. Es utilizado para proyectos de lenguajes de programación diferentes como C, C++, Java, JavaScript, Perl, Python y muchos más. Estas propiedades hacen que JSON sea el lenguaje ideal para el intercambio de datos

Python hace que sea sencillo trabajar con archivos JSON. El módulo utilizado para este propósito es el módulo json. Este módulo debe ser incluido (built-in) dentro de tu instalación Python y, por lo tanto, no será necesario instalar módulos externos como ya hicimos cuando trabajamos con archivos PDF y Excel, por ejemplo. Lo único que necesitas hacer con el fin de utilizar este módulo es importarlo:

### Librería Random

```
import random
```

El módulo aleatorio de Python es un módulo incorporado de Python que se utiliza para generar números aleatorios. Estos son números pseudoaleatorios, lo que significa que estos no son verdaderamente aleatorios. Este módulo se puede utilizar para realizar acciones aleatorias como generar números aleatorios, imprimir al azar un valor para una lista o cadena, etc.

### Librería Pickle

```
import pickle
```

El módulo pickle implementa protocolos binarios para serializar y desserializar una estructura de objetos Python. "Pickling" es el proceso por el cual una jerarquía de objetos Python se convierte en un flujo de bytes, y "unpickling" es la operación inversa, mediante la cual un flujo de bytes (de un archivo binario u objeto similar a bytes) se convierte de nuevo en una jerarquía de objetos. El decapado (y desencofrado) se conoce alternativamente como "serialización", "marshalling", 1 o "aplanamiento"; sin embargo, para evitar confusiones, los términos utilizados aquí son "encurtido" y "desencojado".

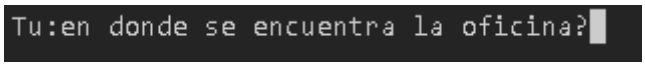
**Manual de Usuario**

Para poder interactuar en el chatbot únicamente la aplicación debe estar ejecutada y aparecerá una consola con la leyenda “tu:” eso significa que puedes empezar a escribir tus preguntas.


Luego de terminar de escribir presionas “enter” y así se procesa la respuesta a tu pregunta.

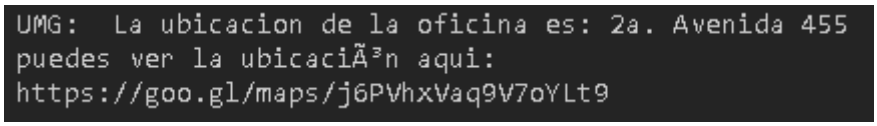
Ejemplo:

1-Digita pregunta




2-presiona enter

3- Recibe respuesta




4- si desea salir puede presionar ctrl+c

## DISTRIBUCIÓN

### Archivos

***Principal.py:***


Este archivo esta destinado a todas las funciones con relación a:

Interacción con archivo JSON.

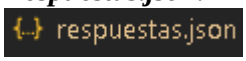
Creación y Manipulación de variables.

Procesamiento de patrones de Archivo JSON.

Creación, configuración y entrenamiento de red neuronal.

Interacción con Usuario final.

***Respuestas.json:***




Este archivo se encarga de contener los siguientes datos:

Contenido general de las respuestas.

Categorías separadas en etiquetas(tags).

Patrones de reconocimiento de cada categoría(tags).

Respuestas según patrón de reconocimiento de cada categoría

# MANUAL TÉCNICO

## Archivo Principal.py

***Lectura de Archivo Json y Declaración de listas***

```
#/*/*/*/*/*/*/*/LECTURA DE ARCHIVO JSON/*/*/*/*/*/*/*/
#OBTENEMOS LOS DATOS DE NUESTRO ARCHIVO JSON
with open("respuestas.json") as f:
    #VOLCAMOS LOS DATOS DEL ARCHIVO JSON EN UNA VARIABLE
    archivoJson = json.load(f)
#/*/*/*/*/*/*/*/ DECLARACIÓN LISTAS/*/*/*/*/*/*
#LISTA QUE CONTIENE LAS PALABRAS RECONOCIDAS
palabras = []
#LISTA QUE CONTIENE LAS ETIQUETAS DE LAS PALABRAS
tags = []
#LISTAS AUXILIARES
auxX=[]
auxY=[]
```

***Recorrido Archivo Json y Llenado de Variables (patrones y etiquetas)***

```
#/*/*/*/*/*/*/*/RECORRIDO ARCHIVO JSON Y LLENADO DE VARIABLES(patrones y tags)/*/*/*/*/*/*
#RECORREMOS EL ARCHIVO JSON EN EL ARREGLO DE CATEGORÍAS
for i in archivoJson["respuestas"]:
    #RECORREMOS EL ARCHIVO JSON EN EL ARREGLO DE PATRONES
    for patrones in i["patrones"]:
        #OBTENEMOS LAS FRASES DEL ARREGLO DE PATRONES Y RECONOCE LAS PALABRAS
        #RECONOCIMIENTO DE PALABRAS CON word_tokenize
        auxPalabra = nt.word_tokenize(patrones)
        #INSERTAMOS LAS PALABRAS RECONOCIDAS EN TOKENS EN palabras
        palabras.extend(auxPalabra)
        #INSERTAMOS LAS PALABRAS RECONOCIDAS EN TOKENS EN auxX
        auxX.append(auxPalabra)
        #INSERTAMOS LOS TAG DE NUESTRO JSON EN auxY
        auxY.append(i["tag"])

        #CONDICIONAL SI EL TAG NO ESTA EN NUESTRO ARREGLO tags
        if i["tag"] not in tags:
            #INSERTAMOS ESE TAG EN EL ARREGLO tags
            tags.append(i["tag"])
```

***Aplicación de Método Stem (reducción a raíz) a Patrones de reconocimiento***

```
#/*/*/*/*/*/*/ PROCESO STEM EN PATRONES TOKENIZADOS/*/*/*/*/*/*
#EL ARREGLO PALABRAS CONTIENE LOS TOKENS Y BUSCAMOS LAS RAÍCES DE LAS PALABRAS
#Y EVITAMOS LOS SIGNOS DE INTERROGACIÓN
palabras = [stemmer.stem(p.lower()) for p in palabras if p != "?"]
#DEVOLVEMOS UNA LISTA ORDENADA DE palabras
palabras = sorted(list(set(palabras)))
#DEVOLVEMOS UNA LISTA ORDENADA DE tags
tags = sorted(tags)

#/*/*/*/*/*/*/DECLARACIÓN LISTAS/*/*/*/*/*/*
#CREAMOS LISTA PARA ENETRENAMIENTO DE LA RED NEURONAL
entrenamiento = []
#CREAMOS LISTA PARA LOS DATOS DE SALIDA
salida = []
#CREAMOS UNA LISTA LLENA DE LA CANTIDAD DE CEROS COMO LA LONGITUD DE LA LISTA tags
salidaVacia = [0 for _ in range(len(tags))]
```

***Contraste de Raíces de palabras con todos los patrones de categorías ordenadas y declaración de Variables de entrenamiento***

```
#/*/*/*/*/*/*/*/APLICACIÓN DEL ALGORITMO DE LA CUBETA/*/*/*/*/*/*//*/
for x,documento in enumerate(auxX):
    #CREAMOS UNA LISTA VACIA LLAMADA cubeta
    cubeta= []
    #OBTENEMOS LA RAÍZ DE LAS PALABRAS DE LA VARIABLE documento
    auxPalabra = [stemmer.stem(p.lower()) for p in documento]

    #RECORREMOS LA LISTA palabras
    for w in palabras :
        #SI EL CONTENIDO COINCIDE CON EL COTENIDO DE auxpalabra
        #EN LA LISTA CUBETA INSERTAMOS UN 1
        if w in auxPalabra:
            cubeta.append(1)
        #DE LO CONTRARO INSERTAMOS EN LA LISTA CUBETA UN 0
        else:
            cubeta.append(0)
    #ASIGMANOS A filaSalida EL CONTENIDO DE salidaVacia
    filaSalida = salidaVacia[:]
    #OBTENEMOS EL ELEMENTO DE auxY EN LA POSICION x,
    #LUEGO OBTENEMOS EL INDICE DE ESE ELEMENTO EN LA LISTA tags
    #PARA LUEGO ASIGNARLE UN 1
    filaSalida[tags.index(auxY[x])] = 1
    #INSERTAMOS EL RESULTADO DE LA LISTA cubeta en LA LISTA entrenamiento
    entrenamiento.append(cubeta)
    #INSERTAMOS EL RESULTADO DE LA LISTA filaSalida en salida
    salida.append(filaSalida)

print(entrenamiento)
#/*/*/*/**/*/CONVERTIMOS LAS 2 LISTAS EN ARRAYS DE NUMPY/*/*/*/*/*/*/*/*/*/*
entrenamiento = np.array(entrenamiento)
salida = np.array(salida)
```

***Función Principal, Interacción con Usuario Final y Tratamiento de Preguntas***

```
#/*/*/*/**/*/*/*/*/*/INTERACCION USUSARIO/*/*/*/*/*/*/*/*/*/
#DEGINIMOS NUESTRA FUNCION PRINCIPAL DE INTERACCION DE CHAT CON EL USUARIO
def mainChatBot():
    #CREAMOS UN CICLO INFINITO
    while True:
        #/*/*/*/*/*/TRATAMIENTO DE ENTRADA DEL USUARIO/*/*/*/*/
        #ASIGNAMOS LA ENTRADA DE DATOS A "entrada" Y MOSTRAMOS EN PANTALLA "Tu" PARA EL USUARIO
        entrada = input("Tu:")
        #LLENAMOS DE 0  LA CUBETA CON LA LONGITUD DE LA LISTA "palabras"
        cubeta = [0 for _ in range(len(palabras))]
        #SEPARAMAOS EN TOKENS EL STRING QUE RECIBE "entrada"
        entradaProc = nt.word_tokenize(entrada)
        #OBTENEMOS LA RAIZ DE LAS PALABRAS SEPARADAS EN TOKENS CON EL METODO "stem"
        entradaProc = [stemmer.stem(p.lower()) for p in entradaProc]
        #RECORREMOS EL ARREGLO "entradaProc"
        for palabraIndividual in entradaProc:
            #REALIZAMOS EL ALGROITMO DE LA CUBETA
            #EN LA VARIABLE i GUARDAMOS EL NUMERO DE ORDEN
            #Y EN LA VARIABLE "palabra" GUARDAMOS EL VALOR(STRING)
            for i, palabra in enumerate(palabras):
                #SI EL ELEMENTO "palabraIndividual" COINCIDE CON
                #"palabra" SE AGREGA UN 1 EN ESA POSICIÓN A "cubeta"
                if palabra == palabraIndividual:
                    cubeta[i] = 1
```

***Obtención de Resultados de Procesamiento de Red Neuronal***

```
#/*/*/*/*/RESPUESTA PREDICCIÓN RED NEURONAL/*/*/*/
 resultados = model.predict([np.array(cubeta)])
 #CREAMOS VARIABLES DE INDICES VACÍA DE TIPO NUMPY
 resultadosIndices = np.empty(shape=0)
 #OBTENEMOS EL VALOR MÁS ALTO DE LOS RESULTADOS DE PROBABILIDAD
 valor = np.max(resultados)
```

***Validación de Descarte***

```
#/*/*/*/*/*VALIDACION DE DESCARTE/*//*/*/*//*
#VALIDAMOS SI HAY UNA PROBABILIDAD DE COINCIDENCIA QUE SEA SUFICIENTE
#SI ES VERDADERO
if valor > 0.85 :
    #PASAMOS EL INDICE CON MAS PROBABILIDAD DE "resultados" a "resultadosIndices"
    resultadosIndices = np.argmax(resultados)
#DE LO CONTRARIO
else:
    #ASIGNAMOS DIRECTAMENTE EL INDICE DE TAG DE RESPUESTAS DE DESCARTE
    resultadosIndices = 0
#ASIGNAMOS A TAGS EL VALOR DE INDICE DE "resultadosIndices"
tag = tags[resultadosIndices]
```

***Emparejamiento de Índice de Categoría (Usuario-Json)***

```
#/*/*/*/*/MATCH DE CAETEGORÍA USUARIO-JSON/*/*/*/*/*
for tagAyu in archivoJson["respuestas"]:
    #VERIFICAMOS SI LA ETIQUETA DETECTADA SE ENCUENTRA EN EL JSON
    if tagAyu["tag"] == tag:
        #SI ES VERDADERO GUARDAMOS EL ARREGLO DE RESPUESTAS
        #DE ESA CATEGORÍA EN LA VARIABLE "respuestas"
        respuesta = tagAyu["respuesta"]
#SE ELIGE LA RESPUESTA DE FORMA ALEATORIA DEL ARREGLO OBTENIDO
print("\nUMG: ", random.choice(respuesta),"\n")
```

***Llamada a Función Principal (mainChatBot)***

```
#LLAMADA A FUNCION PRINCIPAL
mainChatBot()
```