CHAT BOT

INTEGRANTES:

ORTIGOZA MARTÍNEZ HUITZIL

IBARRAZA RIVERA LIZETH

SÁNCHEZ SÁNCHEZ JOSÉ GABRIEL

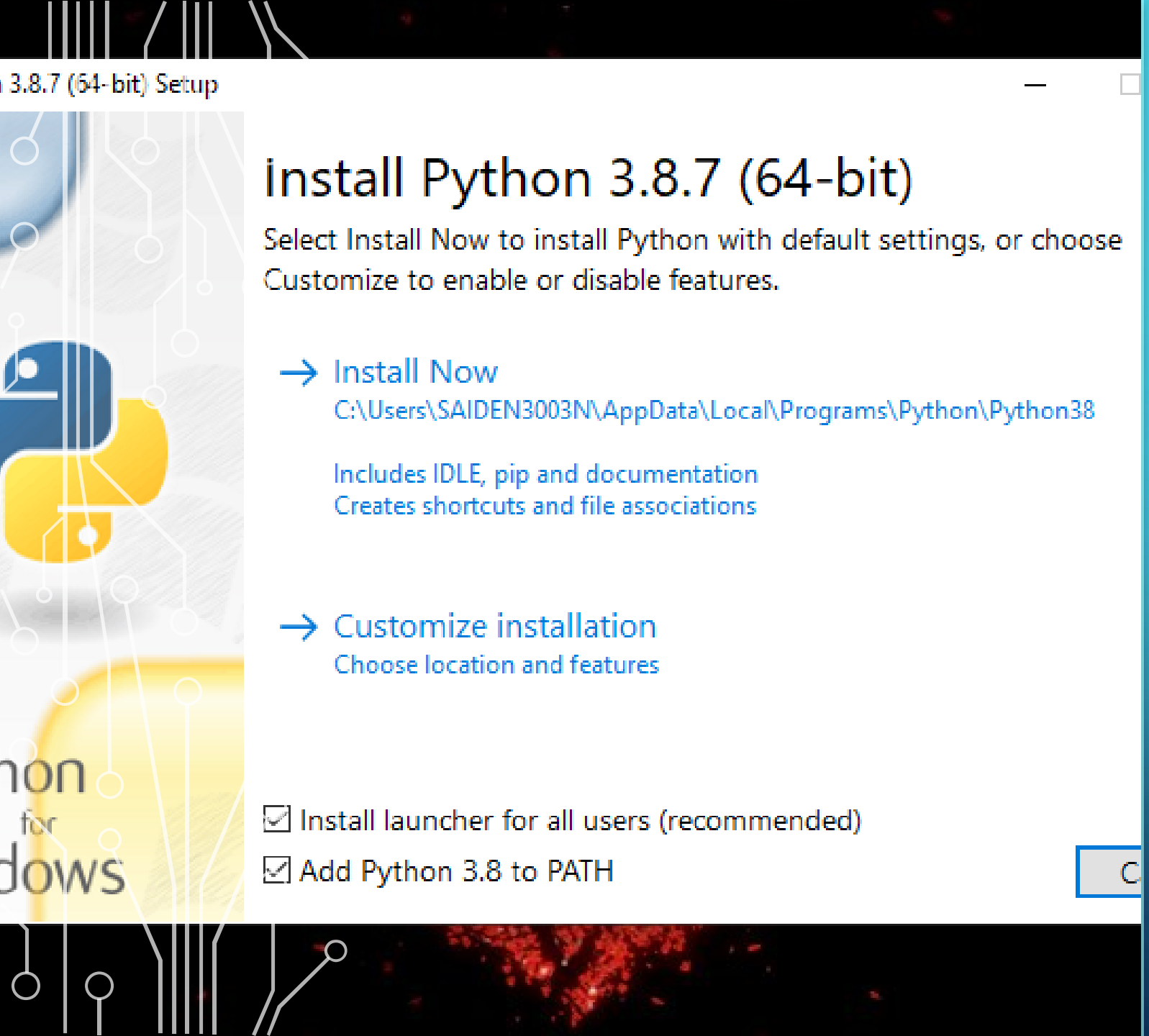
ROSALES MARTÍNEZ AARON

HERRAMIENTAS

Python 3.8

Anaconda 3

INSTALACIÓN DE LAS HERRAMIENTAS

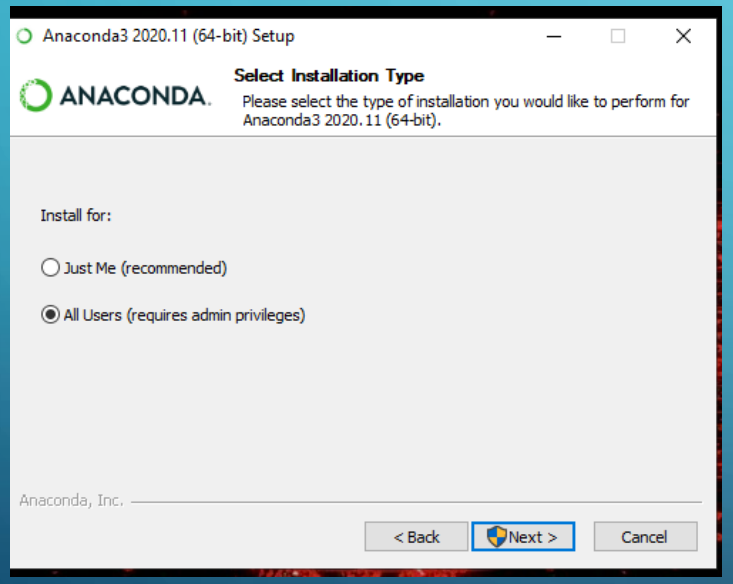


PYTHON

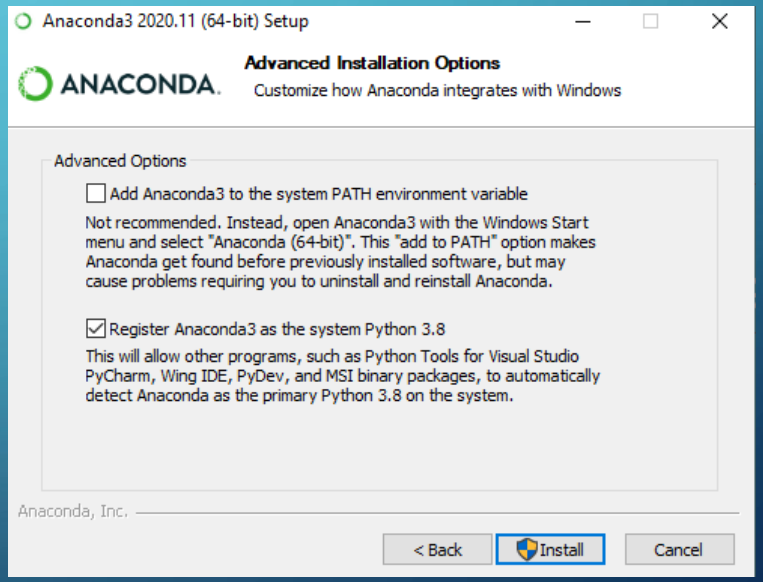
- Para instalar Python por el archivo ejecutable que se descarga de su pagina oficial.
- se seleccionaría las dos opciones agregando al path

ANACONDA

SE LE OTORGARA PERMISOS DE USUARIOS



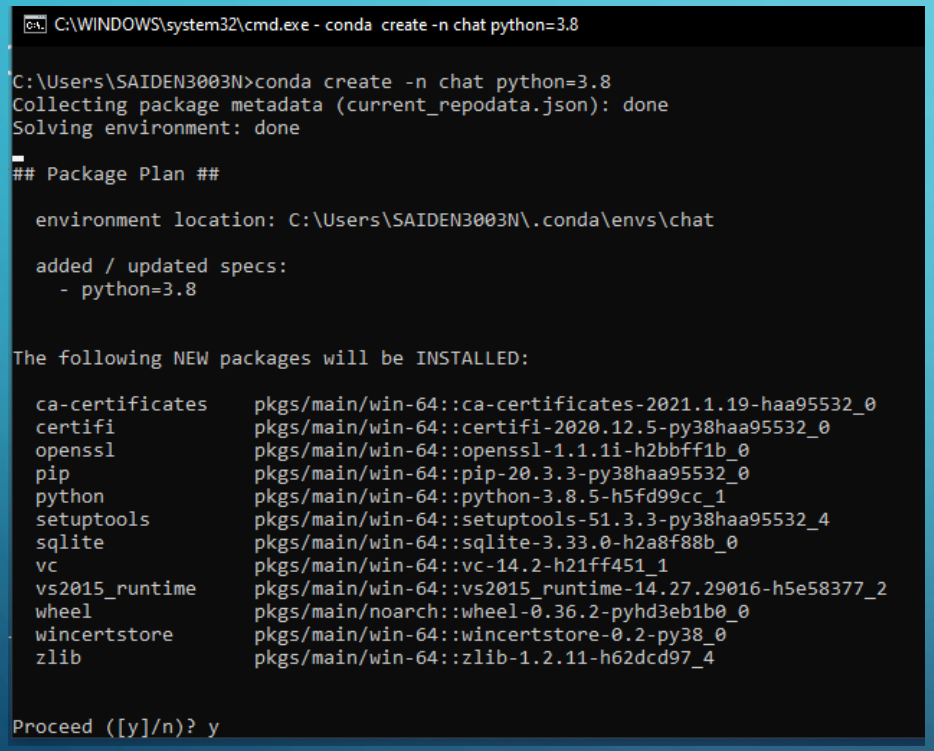
AL IGUAL QUE PYTHON SE AGREGARA AL PATH



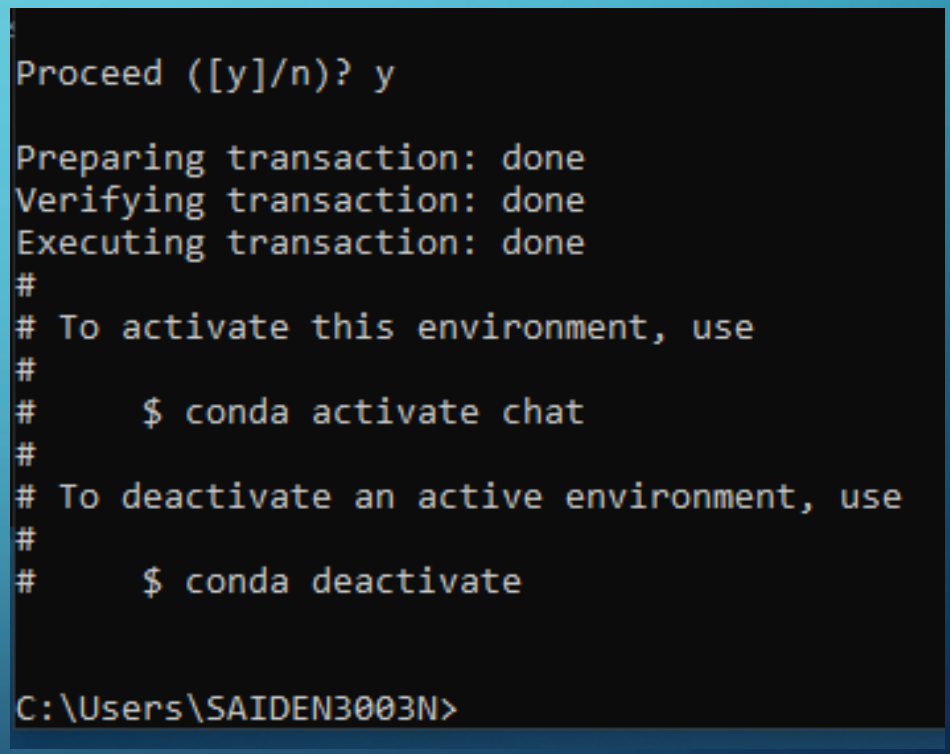
CREACIÓN DEL ENTORNO Y DESCARGA DE LIBRERÍAS

PARA LA CREACIÓN DEL ENTORNO DE DESARROLLO SE USA LA LÍNEA

CONDA CRÉATE –N CHAT PYTHON 3.8



SE ACEPTARA EL PROCESO DE INSTALACIÓN



Se instalaran las librerías tenserflow, nampy, tflearn y nltk.
para hacerlo se necesita entrar al entorno con la línea:
Activate chat
Y se decargaran las librerías con: pip install tensorflow, o el nombre de la librería que se desea descargar

```
(chat) C:\Users\SAIDEN3003N>pip install nltk
Collecting nltk
  Using cached nltk-3.5-py3-none-any.whl
Collecting click
  Using cached click-7.1.2-py2.py3-none-any.whl (82 kB)
Collecting joblib
  Using cached joblib-1.0.0-py3-none-any.whl (302 kB)
Collecting regex
  Using cached regex-2020.11.13-cp38-cp38-win_amd64.whl (270 kB)
Collecting tqdm
  Using cached tqdm-4.56.0-py2.py3-none-any.whl (72 kB)
Installing collected packages: tqdm, regex, joblib, click, nltk
Successfully installed click-7.1.2 joblib-1.0.0 nltk-3.5 regex-2020.11.13 tqdm-4.56.0

(chat) C:\Users\SAIDEN3003N>pip install tensorflow
Collecting tensorflow
  Using cached tensorflow-2.4.1-cp38-cp38-win_amd64.whl (370.7 MB)
Requirement already satisfied: wheel~=0.35 in c:\users\saiden3003n\.conda\envs\chat\lib\site-packages (from tensorflow)
(0.36.2)
Collecting gast==0.3.3
  Using cached gast-0.3.3-py2.py3-none-any.whl (9.7 kB)
Collecting absl-py~=0.10
  Using cached absl_py-0.11.0-py3-none-any.whl (127 kB)
Collecting astunparse~=1.6.3
```

```
(chat) C:\Users\SAIDEN3003N>pip install numpy
Requirement already satisfied: numpy in c:\users\saiden3003n\.conda\

(chat) C:\Users\SAIDEN3003N>pip install tflearn
Collecting tflearn
  Using cached tflearn-0.5.0-py3-none-any.whl
Requirement already satisfied: six in c:\users\saiden3003n\.conda\en
Requirement already satisfied: numpy in c:\users\saiden3003n\.conda\
Collecting Pillow
  Using cached Pillow-8.1.0-cp38-cp38-win_amd64.whl (2.2 MB)
Installing collected packages: Pillow, tflearn
Successfully installed Pillow-8.1.0 tflearn-0.5.0

(chat) C:\Users\SAIDEN3003N>
```

ARCHIVO .JSON

```
    "patrones":["hola","un saludo","hello","buenos dias"],
    "respuestas":["hola que tal, en que puedo ayudarte","Comó te va","un gusto de verte","buenos dias, en que puedo ayudarte"]
},
{"tag":"postre",
    "patrones":["postre","que postre tienes","que hay de postre","que pontres hay"],
    "respuestas":["pastel de chocolate, panseillos, flan, helado,arroz con leche"]
},

{"tag":"despedida",
    "patrones":["adios","hasta luego","nos vemos","hasta la proxima"],
    "respuestas":["cuidate","adios", "nos vemos pronto", "te estare esperando"]

},
{"tag":"abierto",
    "patrones":["abierto","abren","a que hora abren","desde a que hora estan abiertos","a que hora aceptan clientes"],
    "respuestas":["El restaurante abre a las 10 am","aceptamos clientes desde las 10am"]

},
{"tag":"sin",
    "patrones":["sin reservacion","aceptan clientes sin reservacion"],
    "respuestas":["solo se acepta clientes con reservacion","no se aceptan clientes sin reservacion, deberias hacer una reservacion"]

},
{"tag":"cerrado",
    "patrones":["cerrado","cierran","a que hora cierran"],
    "respuestas":["El restaurante cierra a las 12 pm, pero dejamos de recivir clientes con reservacion a las 10 pm"]

},
{"tag":"personas",
    "patrones":["cuantas personas son en la reservacion","cuantas personas por reservacion","cuantos comensales pueden ir en una reservacion"],
    "respuestas":["son 5 personas maximo","5 personas por reservacion","en una reservacio pueden ir 5 comensales"]
},
{"tag":"reservacion",
    "patrones":["cuantas veces puedo reservar","puedo hacer mas de una reservacion","puedo hacer mas reservaciones"],
    "respuestas":["solo puedes hacer una reservacion por persona","si ya hiciste una reservacion ya no puedes hacer otra"]
},
```

en este archivo se alojara el contenido que consta de:

Tags: la acción

Patrones: que son las palabras que están formada una oración

Respuestas: aquí se encontraran todas las respuestas que se pudieran dar a los patrones

IMPORTAR LAS LIBRERIAS

- La librería nltk permitirá el procesamiento de lenguaje natural.
- Nltk.stem.Lancaster esto permitirá transformar las palabras, para que sean mas entendibles para el chatbot
- Json que ahí se tendrá la información
- Randon para las respuestas aleatorias
- Pickle para guardar el proyecto

SE MANDA A LLAMAR AL ARCHIVO .JSON

mainbot.py > ...

```python
import nltk
from nltk.stem.lancaster import LancasterStemmer
stemmer = LancasterStemmer()
import numpy
import tflearn
import tensorflow
import json
import random
import pickle

#nltk.ddowload('punkt')
with open("contenido.json") as archivo:
    datos = json.load(archivo)
print(datos)
```

Y EL RESULTADO ES TODA LA INFORMACIÓN DE EL


```
{'contenido': [{'tag': 'saludo', 'patrones': ['hola', 'un saludo', 'hello', 'buenos dias'], 'respuestas': ['hola que tal
, en que puedo ayudarte', 'ComÃ³ te va', 'un gusto de verte', 'buenos dias, en que puedo ayudarte']}, {'tag': 'postre',
'patrones': ['postre', 'que postre tienes', 'que hay de postre', 'que pontres hay'], 'respuestas': ['pastel de chocolate
, panseillos, flan, helado,arroz con leche']}, {'tag': 'despedida', 'patrones': ['adios', 'hasta luego', 'nos vemos', 'h
asta la proxima'], 'respuestas': ['cuidate', 'adios', 'nos vemos pronto', 'te estare esperando']}, {'tag': 'abierto', 'p
atrones': ['abierto', 'abren', 'a que hora abren', 'desde a que hora estan abiertos', 'a que hora aceptan clientes'], 'r
espuestas': ['El restaurante abre a las 10 am', 'aceptamos clientes desde las 10am']}, {'tag': 'sin', 'patrones': ['sin
reservacion', 'aceptan clientes sin reservacion'], 'respuestas': ['solo se acepta clientes con reservacion', 'no se acep
tan clientes sin reservacion, deberias hacer una reservacion']}, {'tag': 'cerrado', 'patrones': ['cerrado', 'cierran', '
a que hora cierran'], 'respuestas': ['El restaurante cierra a las 12 pm, pero dejamos de recivir clientes con reservacio
n a las 11pm', '']}, {'tag': 'personas', 'patrones': ['cuantas personas son en la reservacion', 'cuantas personas por re
servacion', 'cuantos comensales pueden ir en una reservacion'], 'respuestas': ['son 5 personas maximo', '5 personas por
reservacion', 'en una reservacio pueden ir 5 comensales']}, {'tag': 'reservacion', 'patrones': ['cuantas veces puedo res
ervar', 'puedo hacer mas de una reservacion', 'puedo hacer mas reservaciones'], 'respuestas': ['solo puedes hacer una re
servacion por persona', 'si ya hiciste una reservacion ya no puedes hacer otra']}, {'tag': 'niÃ±os', 'patrones': ['puede
n entrar niÃ±os', 'pueden entrar bebes', 'cuantos niÃ±os pueden ir por reservacion'], 'respuestas': ['se admiten niÃ±os,
 PAGAN DOBLE, solo toma encuenta el menu del restaurante', 'acuerdate que solo son 5 personas maximo, contanto a los niÃ
±os']}, {'tag': 'estas', 'patrones': ['como estas', 'hola como estas', 'que tal estas'], 'respuestas': ['no lo se, tu di
me']}]}

(chat) C:\Users\SAIDEN3003N\chat>
```

se creara un for para entrar al contenido en los tags para los patrones y acceder a ellos, el contenido de ello se almacenara en auxpalabra, junto con nltk.word_tokenize para reconocer los signos especiales.

En auxY se guardaran los tags repetido para pasarlos a individual se usa un if.

```
palabras=[]
tags=[]
auxX=[]
auxY=[]

for contenido in datos["contenido"]:
    for patrones in contenido["patrones"]:
        auxPalabra = nltk.word_tokenize(patrones)
        palabras.extend(auxPalabra)
        auxX.append(auxPalabra)
        auxY.append(contenido["tag"])

        if contenido["tag"] not in tags:
            tags.append(contenido["tag"])
print(palabras)
print(auxX)
print(auxY)
print(tags)
```

```
']
[['hola'], ['un', 'saludo'], ['hello'], ['buenos', 'dias'], ['postre'], ['que', 'postre', 'tienes'], ['que', 'hay', 'de'
, 'postre'], ['que', 'pontres', 'hay'], ['adios'], ['hasta', 'luego'], ['nos', 'vemos'], ['hasta', 'la', 'proxima'], ['a
bierto'], ['abren'], ['a', 'que', 'hora', 'abren'], ['desde', 'a', 'que', 'hora', 'estan', 'abiertos'], ['a', 'que', 'ho
ra', 'aceptan', 'clientes'], ['sin', 'reservacion'], ['aceptan', 'clientes', 'sin', 'reservacion'], ['cerrado'], ['cierr
an'], ['a', 'que', 'hora', 'cierran'], ['cuantas', 'personas', 'son', 'en', 'la', 'reservacion'], ['cuantas', 'personas'
, 'por', 'reservacion'], ['cuantos', 'comensales', 'pueden', 'ir', 'en', 'una', 'reservacion'], ['cuantas', 'veces', 'pu
edo', 'reservar'], ['puedo', 'hacer', 'mas', 'de', 'una', 'reservacion'], ['puedo', 'hacer', 'mas', 'reservaciones'], ['
pueden', 'entrar', 'niños'], ['pueden', 'entrar', 'bebes'], ['cuantos', 'niños', 'pueden', 'ir', 'por', 'reservacion'],
['como', 'estas'], ['hola', 'como', 'estas'], ['que', 'tal', 'estas']]
['saludo', 'saludo', 'saludo', 'saludo', 'postre', 'postre', 'postre', 'postre', 'despedida', 'despedida', 'despedida',
'despedida', 'abierto', 'abierto', 'abierto', 'abierto', 'abierto', 'sin', 'sin', 'cerrado', 'cerrado', 'cerrado', 'pers
onas', 'personas', 'personas', 'reservacion', 'reservacion', 'reservacion', 'niños', 'niños', 'niños', 'estas', 'estas',
 'estas']
['saludo', 'postre', 'despedida', 'abierto', 'sin', 'cerrado', 'personas', 'reservacion', 'niños', 'estas']

(chat) C:\Users\SAIDEN3003N\chat>
```

Se usara la línea:

palabras = [stemmer.stem(w.lower())
for w in palabras if w!="?"]

Para pasar la palabra en minúscula siempre y cuando la palabra sea diferente a ?

Se ordenara las palabras y los tags

palabras = sorted(list(set(palabras)))

tags = sorted(tags)

```
palabras = [stemmer.stem(w.lower()) for w in palabras if w!="?"]
palabras = sorted(list(set(palabras)))
tags = sorted(tags)
```

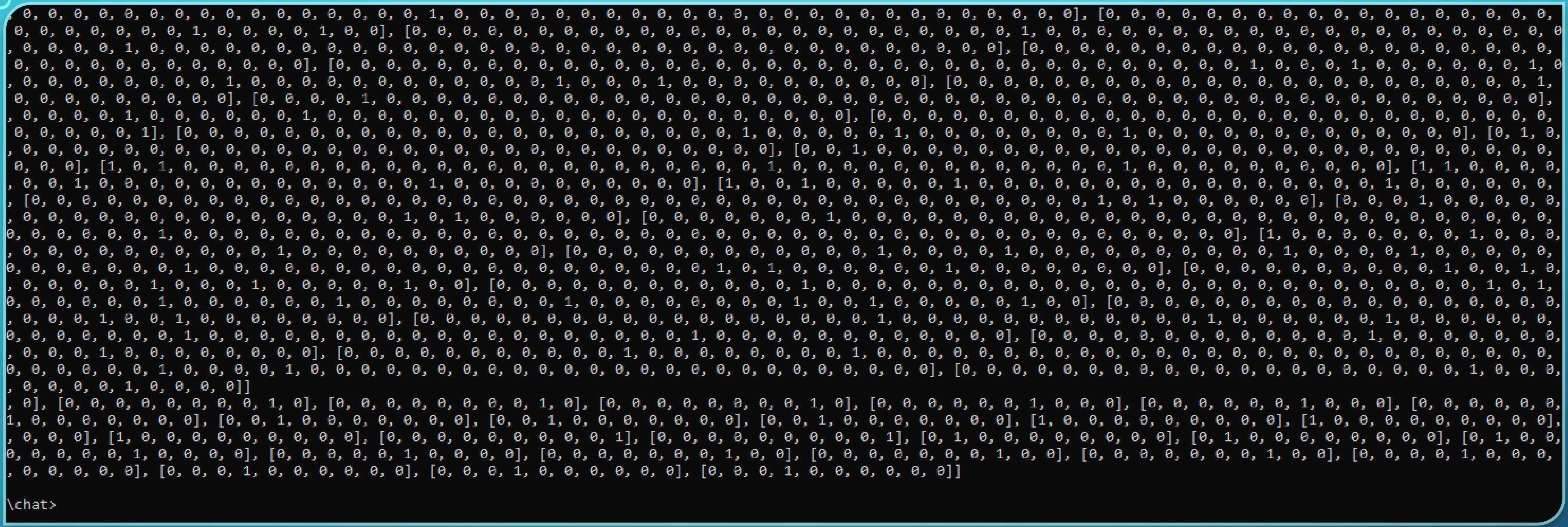
Para entrenar al bot, se hace uso de una cubeta, par esto se usara un for, el cual enumerara las palabras por un índice guardado en x

En el for w, si esta en palabras se añadirá en cubeta, 1 si esta 0 si no esta.

La fila salida ayudara para obtener el índice de y para asignarle el valor de 1

En entrenamiento se agregara la cubeta

```
for x, documento in enumerate(auxX):
    cubeta=[]
    auxPalabra= [stemmer.stem(w.lower()) for w in documento]
    for w in palabras:
        if w in auxPalabra:
            cubeta.append(1)
        else:
            cubeta.append(0)
    filaSalida = salidaVacia[:]
    filaSalida[tags.index(auxY[x])]=1
    entrenamiento.append(cubeta)
    salida.append(filaSalida)
print(entrenamiento)
print(salida)
```



El resultado es una lista de listas, se muestra los 1 donde se encontró la palabra encontrada

Se pasa las listas en arreglos con numpy

Con tensorflow aremos que se rinicie todo

En la red se creara una red, con forma shape y la longitud,

Se crea los hidden layers (5), los cuales tendrán las neuronas que tendrán, al igual se creara una red para la longitud de la salida que será 0, esta cera la salida

Con tflearn.regression se obtendrán probabilidades para saber que tag nos referimos

```
entrenamiento = numpy.array(entrenamiento)
salida = numpy.array(salida)
tensorflow.compat.v1.reset_default_graph()

red = tflearn.input_data(shape=[None,len(entrenamiento[0])])
red = tflearn.fully_connected(red,80)
red = tflearn.fully_connected(red,80)
red = tflearn.fully_connected(red,80)
red = tflearn.fully_connected(red,80)
red = tflearn.fully_connected(red,80)
red = tflearn.fully_connected(red,len(salida[0]),activation="softmax")
red = tflearn.regression(red)
```

Se creara un modelo.fity que ayudara a entrenar a la red, con n_epoch será las veces que vea la información, batch_size nos ayudara para saber cuantas entradas tenemos que va a las palabras de los patrones.

Se guardara el modelo en un archivo.tflearn

```
modelo = tflearn.DNN(red)
modelo .fit(entrenamiento,salida,n_epoch=10000,batch_size=84,show_metric=True)
modelo.save("modelo.tflearn")
```

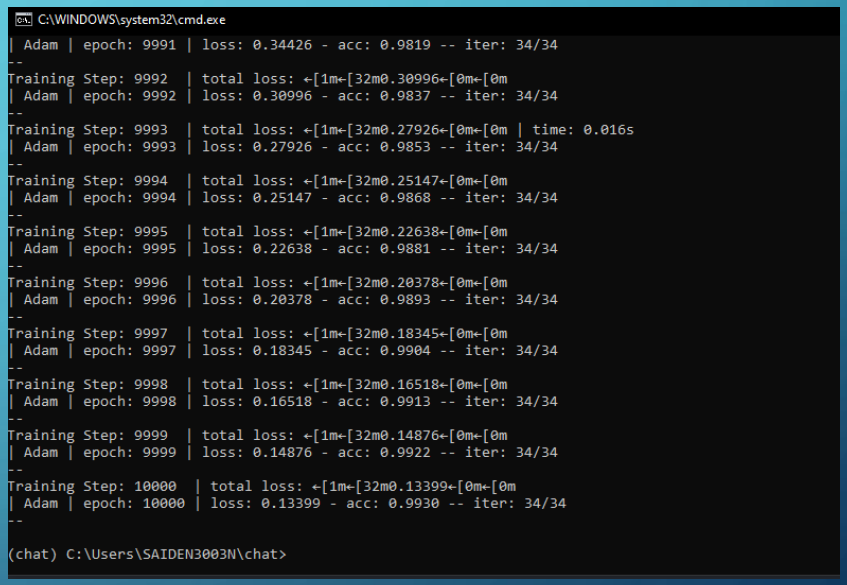
CÓDIGO

```
entrenamiento = numpy.array(entrenamiento)
salida = numpy.array(salida)
tensorflow.compat.v1.reset_default_graph()

red = tflearn.input_data(shape=[None,len(entrenamiento[0])])
red = tflearn.fully_connected(red,80)
red = tflearn.fully_connected(red,80)
red = tflearn.fully_connected(red,80)
red = tflearn.fully_connected(red,80)
red = tflearn.fully_connected(red,80)
red = tflearn.fully_connected(red,len(salida[0]),activation="softmax")
red = tflearn.regression(red)

modelo = tflearn.DNN(red)
modelo .fit(entrenamiento,salida,n_epoch=10000,batch_size=84,show_metric=True)
modelo.save("modelo.tflearn")
```

SE MUESTRAN COMO HACE EL ENTRENAMIENTO



Se crea la función principal, la cual se encargara de tomar al entrada del usuario.

Para esto se usa while, junto con una variable llamada entrada.

Se crea una cubeta para saber que variables esta usando el usuario.