Volume: 9 Issue: VIII Month of publication: August 2021
DOI: https://doi.org/10.22214/ijraset.2021.37668

International Journal for Research in Applied Science & Engineering Technology (IJRASET)
ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.429
Volume 9 Issue VIII Aug 2021- Available at www.ijraset.com

Python the game changer in the field of Machine Learning, Data Science and IoT: A Review

Prithwish Parial[1], Debrupa Pal[2]
[1]*Student, Computer Application, Narula Institute of Technology, Kolkata, West Bengal, India*
[2]*Assistant Professor, Computer Application, Narula Institute of Technology, Kolkata, West Bengal, India*

Abstract: Python is the finest, easily adoptable object-oriented programming language developed by Guido van Rossum, and first released on February 20, 1991 It is a powerful high-level language in the recent software world. In this paper, our discussion will be an introduction to the various Python tools applicable for Machine learning techniques, Data Science and IoT. Then describe the packages that are in demand of Data science and Machine learning communities, for example- Pandas, SciPy, TensorFlow, Theano, Matplotlib, etc. After that, we will move to show the significance of python for building IoT applications. We will share different codes throughout an example. To assistance, the learning experience, execute the following examples contained in this paper interactively using the Jupiter notebooks.
Keywords: Machine learning, Real world programming, Data Science, IOT, Tools, Different packages, Languages- Python.

I. INTRODUCTION

A. *Introduction to Python*

Python is a broadly used general-purpose, high-level programming language which is in the recent software world. Its design legibility emphasizes code easy to logically understand, and its syntax allows programmers to express their concepts in a handful of lines of code than would be possible in languages like C. The most significant feature of the Python language it supports multiple programming paradigms, including object-oriented, imperative, and procedural styles. Nowadays interpreters are supports in almost every operating system. Python provides a large set of the widespread valuable library which is absolutely extensible and embedded. Python is widely used in computational science such as SciPy for Mathematics, Science, and Engineering sector and Pandas for data analysis and modeling, IPython for easy editing and helps visualizations and parallel computing. In this paper, we are going to introduce the brief idea of python in the field of Data Science, Machine learning, and IOT. Python is known to have a sufficiency of libraries that assist with data analysis and computational science. for instance, we can build the various applications of Python which helps data analysts to analyze high amounts of data for computational science. We know the basic understanding of statistics, on top of experience in any C-style language.   This following links will help to aid your learning experience: [1].

B. *Introduction to Machine Learning*

Machine learning is having a dramatic impact on the software development sector. Increasing amounts of data becoming available there is good reason to believe that smart data analysis will become even more prevalent as an essential ingredient for technological progress in recent times. We are surrounded by machine learning technology: such as Messenger (which helps us to connect with friends, relatives in different parts of the world.) Smart Bands (It is a fitness tracker it helps us track record our heart rate, daily burned calories, and step counts. Check Spo2 level, exercise more, and sleep better experience). And also automated Traffic Control System that provides good traffic management in road prevent the lives from the accident. It helps programmers to write complex code with detailed specifications. Suppose that intelligent beings, there are many abilities, skills that are obtained or clarified through learning from our experience (on the other hand following obvious instructions given to us). Machine learning tools are concerned with unceasing programs with the ability to "learn "and adapt quickly with the help of large volumes of data. If you are focusing on this field to build your career in the future, It might be better to start with Python [2].

C. *Introduction to Data Science*

Data science is an incorporative field, it has scientific techniques, procedures, various algorithms, and systems to bring out knowledge and insights from structured and unstructured data, and apply knowledge and actionable insights from data across a wide range of application domains. Data science is associated or linked to data mining, machine learning, and big data. Data Science procedure to deliver analytics solutions and bright applications professionally. Statistics can help us to go deeper sharpness into the data, and the foremost discipline to analyze and quantify uncertainty. Python Can provides various pre-decided modules to work on Data science projects [3].

D. Introduction to IoT

In a simple word "Internet" can be described as the communication network that connects each and every piece of information while The Internet of Things (IoT) is an interconnected system only addressable physical items with different degrees of processing, sensing, and actuation capabilities that share the capability to interaction and communication through the Internet as their joint platform. Thus, the main purpose of the Internet of Things is to make it possible for objects to be connected with other objects, individuals, at any time or anywhere using any network, path, and also the main thing is to automatically transfer the data more to a network without requiring human-to-human or human-To-computer communication. in line with the TIOBE index, Python was the programming language of the year in 2018 With a rating of 10.020%, and Python was also the 3rd most popular language in 2019. but it has gained popularity in the IoT system in Recent times because of it has a large number of libraries that helps coder to write codes in few lines. Python is the Right decision, for data analysis in IoT systems. The language is simple, easy to understand for coders [4] .

This following links will help to aid your learning experience:
1) *Numpy:* https://github.com/mdbloice/MLDS/blob/master/NumPy.ipynb
2) *Pandas:* https://github.com/mdbloice/MLDS/blob/master/Pandas.ipynb
3) *Plotting:* https://github.com/mdbloice/MLDS/blob/master/Plotting.ipynb
4) *Python in Data Science hand book:* https://flaviocopes.nyc3.digitaloceanspaces.com/python-handbook/python-handbook/python-handbook.pdf

II. LITERATURE SURVEY

The main purposes of this study are to understand the various features of Python, Explore the packages for Data Science like SciPy, it carries modules for optimization, integration, linear algebra, interpolation, special functions, signal and image processing, etc. TensorFlow is an absolutely free and open-source software library in machine learning. Matplotlib, it is a plotting library for Python and It offers an object-oriented API for embedding plots within applications utilize in general-purpose GUI toolkits like Tkinter.

A. Working Environment Related

Python is Interpreted programming language; we do not need to compile Our program before executing it. This is close to PERL and PHP language. It is a very Interactive programming language − We can actually sit at a Python quickly interact with the interpreter directly to write our programs. And it is also Python is a Beginner's Language a very awesome language for beginner-level programmers and we can create many interesting games like – balloon shooter, snake game, Flappy Bird game, etc. and published it in a World-Wide range.

Let us take an example of game name is "Flappy Bird":-

This is the hero of the game: -

Fig.1 Graphical Interface

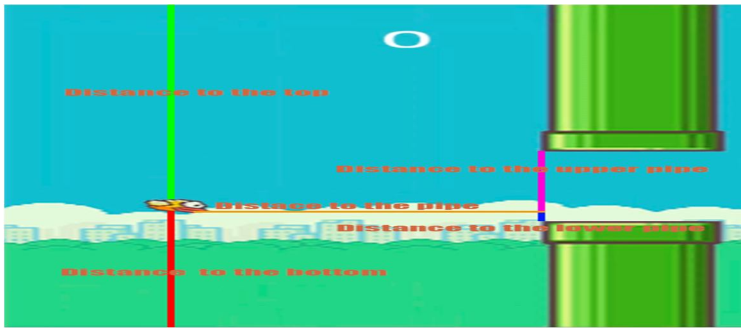


Fig.2 Opening Interface

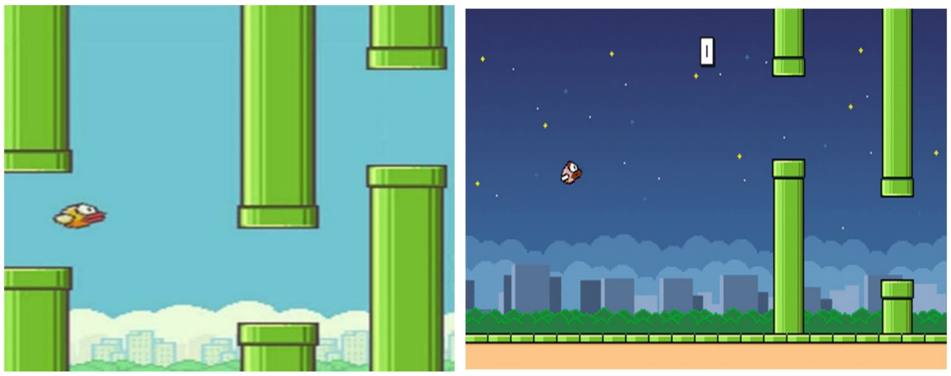


Fig.3 Some Interesting Gameplays

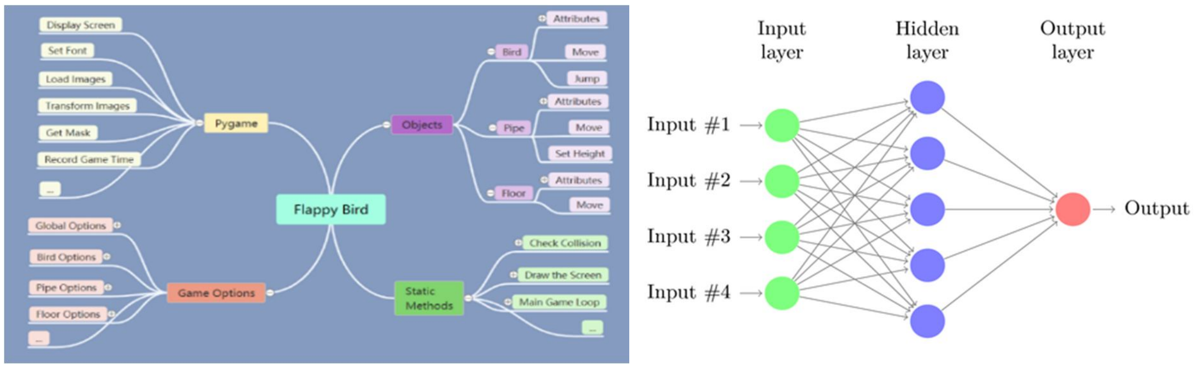


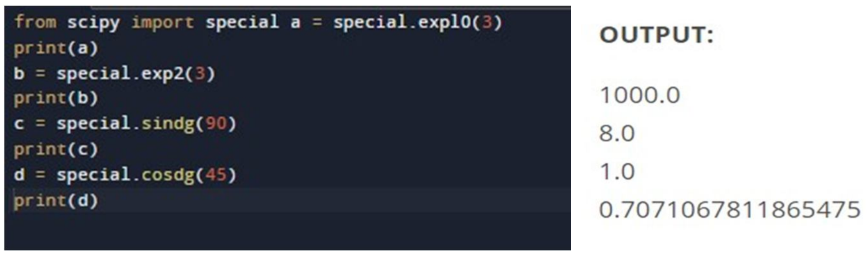
Fig.4 Neural Network of Flappy Bird

We provide all the necessary Flappy bird files billow in the link:-
https://drive.google.com/drive/folders/1U5t6IHvIT1wqSM1qutVNmIX2rj3y3Voj?usp=sharing

B. Python for Data Science
1) SciPy
2) Matplotlib
3) TensorFlow

1) SciPy: SciPy stands for **"Scientific Python".** It is a bunch of open source scientific and numerical tools for Python. It helps us to special functions, integration, ordinary differential equation (ODE) solvers etc [5].

a) Exponential and Trigonometric Functions: SciPy's Special Function package provides a number of functions through which you can find exponents and solve trigonometric problems.



```
from scipy import special a = special.exp10(3)
print(a)
b = special.exp2(3)
print(b)
c = special.sindg(90)
print(c)
d = special.cosdg(45)
print(d)
```

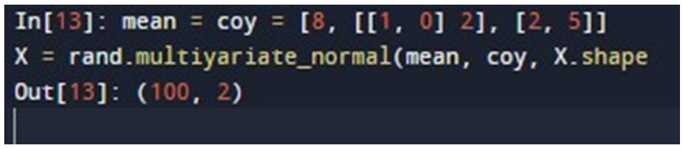
OUTPUT:

```
1000.0
8.0
1.0
0.7071067811865475
```

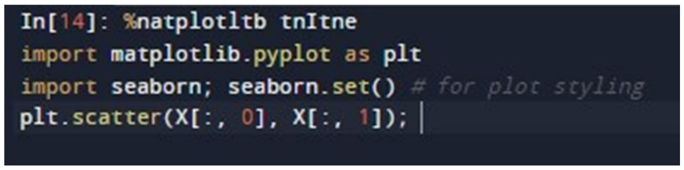
Fig.5 Exponential and Trigonometric Functions Code

There are many other functions present in the special functions package of SciPy that you can try for yourself.

b) *Example: Selecting Random Points:* One common use of fancy categorization is that the choice of subsets of rows from a matrix. for instance, we would have an N by D matrix representing N points in D dimensions, just like the following points drawn from a two-dimensional traditional distribution:



```
In[13]: mean = coy = [8, [[1, 0] 2], [2, 5]]
X = rand.multiyariate_normal(mean, coy, X.shape
Out[13]: (100, 2)
```



```
In[14]: %natplotltb tnItne
import matplotlib.pyplot as plt
import seaborn; seaborn.set() # for plot styling
plt.scatter(X[:, 0], X[:, 1]);
```

Fig.6 Snapshot of Selected Random Point Code

c) *Normally Distributed Points*

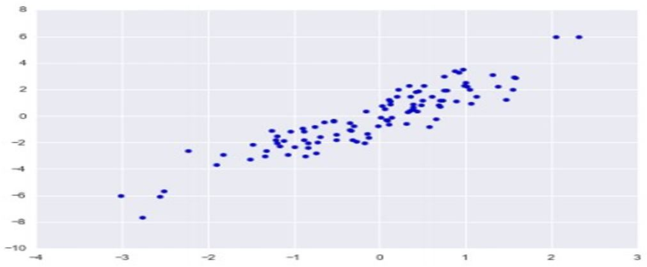
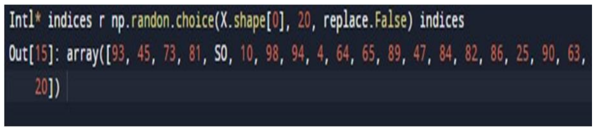
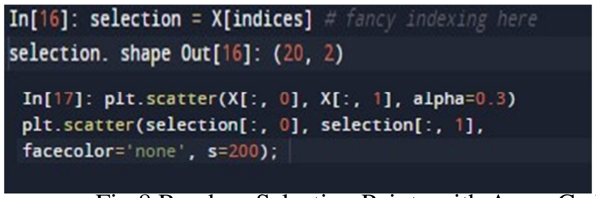


Fig.7 Graphical Representation of Normally Distributed Point

Let's use fancy indexing to select 20 random points. We'll do this by first selecting 20 random indices with no repeats, and use these indices to select a portion of the main array:



```
Intl* indices r np.randon.choice(X.shape[0], 20, replace.False) indices
Out[15]: array([93, 45, 73, 81, SO, 10, 98, 94, 4, 64, 65, 89, 47, 84, 82, 86, 25, 90, 63,
   20])
```



```
In[16]: selection = X[indices] # fancy indexing here
selection. shape Out[16]: (20, 2)

 In[17]: plt.scatter(X[:, 0], X[:, 1], alpha=0.3)
 plt.scatter(selection[:, 0], selection[:, 1],
 facecolor='none', s=200);
```

Fig.8 Random Selection Points with Array Code

d) *Random Selection among Points*

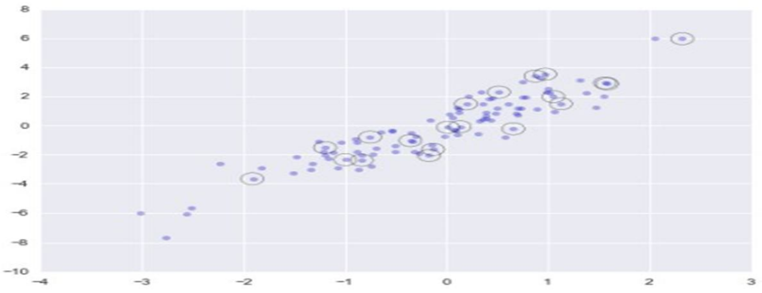


Fig.9 Graphical Representation of Random Selection Among Points

C. Python for Machine Learning

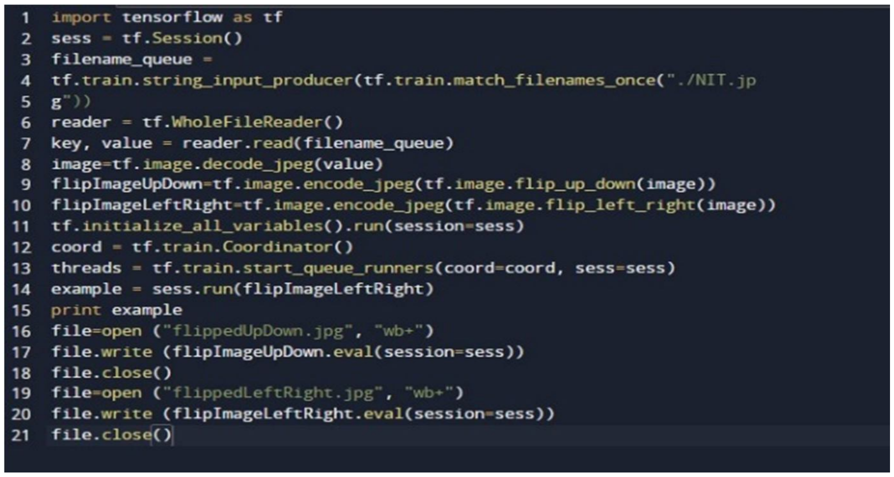
1) *TensorFlow:* TensorFlow provides a collection of workflows to develop models by using Python and to easily deploy in the cloud, in the browser, data API enables you to build complex input pipelines from simple, reusable pieces [6].

a) *Reading Image Data:* TensorFlow can help Reading image data. It assists in importing the data from image formats, it really helps image-oriented models. The internal representation would be 8 unit tensors, one rank two tensors for each image channel: [7]

Fig.10 Reading Image Data

b) *Loading and Processing the Images:* In this example, we upload an image and apply some additional processing to it, saving the resulting images in separate files:

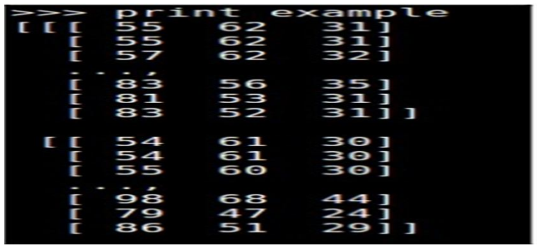
c) *Codes*



```
import tensorflow as tf
sess = tf.Session()
filename_queue =
tf.train.string_input_producer(tf.train.match_filenames_once("./NIT.jp
g"))
reader = tf.WholeFileReader()
key, value = reader.read(filename_queue)
image=tf.image.decode_jpeg(value)
flipImageUpDown=tf.image.encode_jpeg(tf.image.flip_up_down(image))
flipImageLeftRight=tf.image.encode_jpeg(tf.image.flip_left_right(image))
tf.initialize_all_variables().run(session=sess)
coord = tf.train.Coordinator()
threads = tf.train.start_queue_runners(coord=coord, sess=sess)
example = sess.run(flipImageLeftRight)
print example
file=open ("flippedUpDown.jpg", "wb+")
file.write (flipImageUpDown.eval(session=sess))
file.close()
file=open ("flippedLeftRight.jpg", "wb+")
file.write (flipImageLeftRight.eval(session=sess))
file.close()
```

Fig.11 Reading Image Code

d) *The Print Example Line Will Show A Line-By-Line Summary Of The Rgb Values In The Image*



```
>>> print example
[[[ 55  62  31]
  [ 55  62  31]
  [ 57  62  32]
  ...,
  [ 83  56  35]
  [ 81  53  31]
  [ 83  52  31]]

 [[ 54  61  30]
  [ 54  61  30]
  [ 55  60  30]
  ...,
  [ 98  68  44]
  [ 79  47  24]
  [ 86  51  29]]
```

Fig.12 Print Example of RGB Image

e) The Final Images will look Like

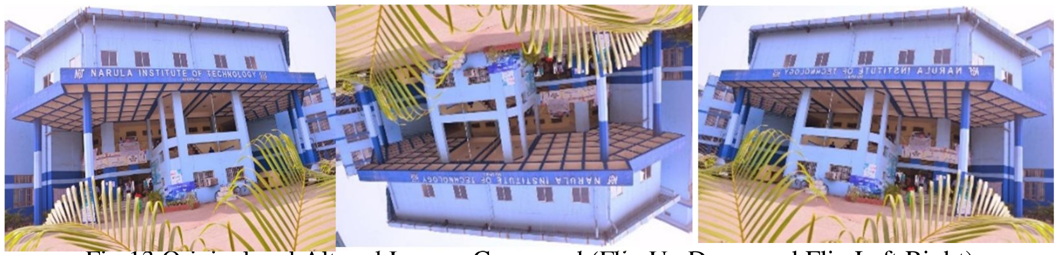


Fig.13 Original and Altered Images Compared (Flip Up Down and Flip Left Right)

D. Python for IoT

In The IoT section, Python is a good option for the backend side of development also the software development of any devices. Python helps to interact with devices with users, it makes life simple day by day [8].

1) *Python Chatbot Project:* A chatbot is an intelligent and quick responsive software that is capable of interacting and answering like a human. Chatbots are used for customer interaction, online marketing area, social network sites and instantly messaging the client [9].

Fig.14 Welcome Screen

Now we are going to create the chatbot using Python but first, let us see the structure of a file and the type of files we will be creating:[10].

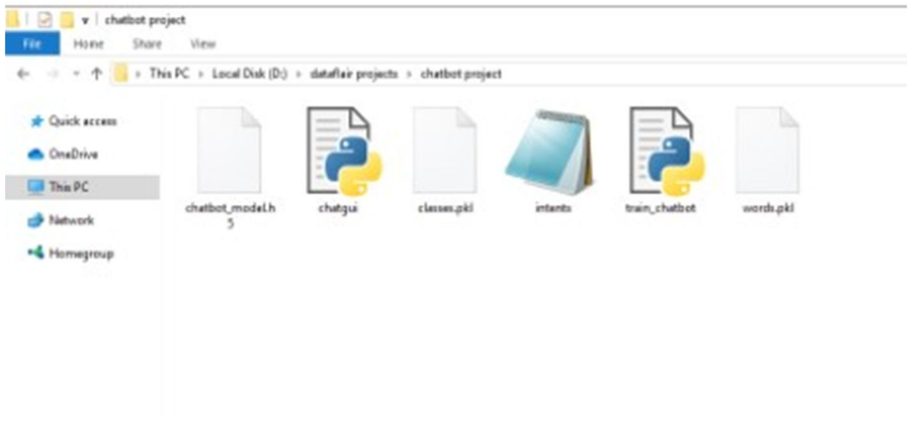


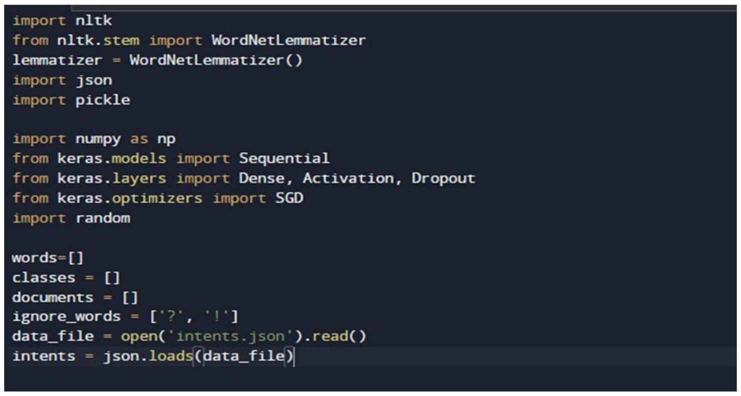
Fig.15 Created Files

a) *Intents.json:* The data file has predefined patterns and responses.
b) *Train_chatbot.py:* In this Python file, we write a script to create the model and train our Chabot.
c) *Words.pkl:* This file store the words Python object that contains a list of our vocabulary.
d) *Classes.pkl:* This file contains the list of categories.
e) *Chatbot_model.h5:* This is the trained model that contains information about the model and has weights of the neurons.
f) *Chatgui.py:* In this Python script in we implemented GUI for our chatbot. Users can easily interact with the bot.

2) *Steps to create a chatbot in Python from scratch*

a) *Import and Load the data File:* Firstly, type a file name as train_chatbot.py, import the necessary packages for our chatbot and initialize the variables we will use in our Python project.

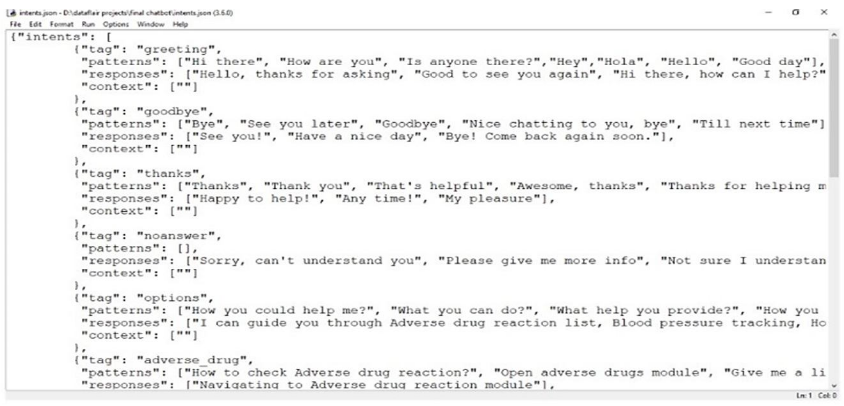
The data file is in JSON format



```
import nltk
from nltk.stem import WordNetLemmatizer
lemmatizer = WordNetLemmatizer()
import json
import pickle

import numpy as np
from keras.models import Sequential
from keras.layers import Dense, Activation, Dropout
from keras.optimizers import SGD
import random

words=[]
classes = []
documents = []
ignore_words = ['?', '!']
data_file = open('intents.json').read()
intents = json.loads(data_file)
```

Fig.16 Initialization of The Variables

It look like:-

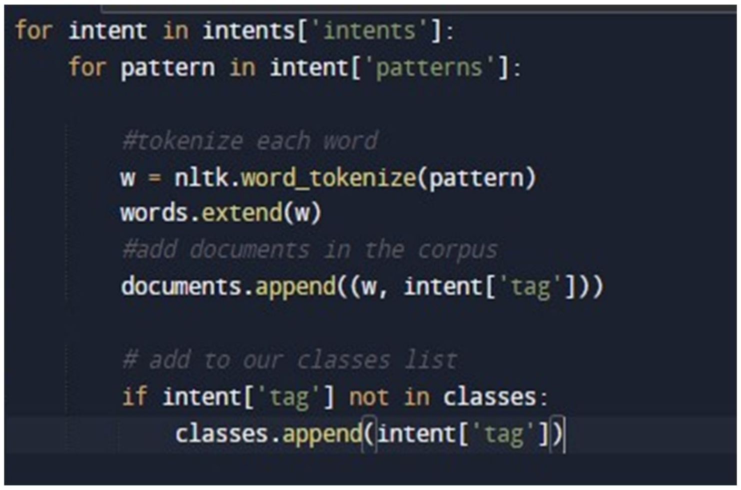


intents.json - D:\dataflair projects\final chatbot\intents.json (3.6.0)
File Edit Format Run Options Window Help
Ln: 1 Col: 0

```
{"intents": [
        {"tag": "greeting",
         "patterns": ["Hi there", "How are you", "Is anyone there?","Hey","Hola", "Hello", "Good day"],
         "responses": ["Hello, thanks for asking", "Good to see you again", "Hi there, how can I help?"
         "context": [""]
        },
        {"tag": "goodbye",
         "patterns": ["Bye", "See you later", "Goodbye", "Nice chatting to you, bye", "Till next time"]
         "responses": ["See you!", "Have a nice day", "Bye! Come back again soon."],
         "context": [""]
        },
        {"tag": "thanks",
         "patterns": ["Thanks", "Thank you", "That's helpful", "Awesome, thanks", "Thanks for helping m
         "responses": ["Happy to help!", "Any time!", "My pleasure"],
         "context": [""]
        },
        {"tag": "noanswer",
         "patterns": [],
         "responses": ["Sorry, can't understand you", "Please give me more info", "Not sure I understan
         "context": [""]
        },
        {"tag": "options",
         "patterns": ["How you could help me?", "What you can do?", "What help you provide?", "How you
         "responses": ["I can guide you through Adverse drug reaction list, Blood pressure tracking, Ho
         "context": [""]
        },
        {"tag": "adverse_drug",
         "patterns": ["How to check Adverse drug reaction?", "Open adverse drugs module", "Give me a li
         "responses": ["Navigating to Adverse drug reaction module"],
```

Fig.17 JSON Format

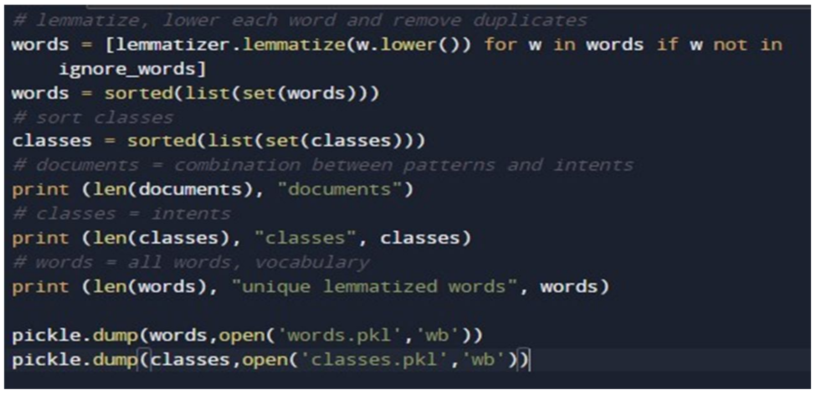
b) *Preprocess Data:* Here we repeat through the patterns and tokenize the sentence using nltk. word tokenize () function and append each word in the words list. We also create a list of classes for our tags.



```
for intent in intents['intents']:
    for pattern in intent['patterns']:

        #tokenize each word
        w = nltk.word_tokenize(pattern)
        words.extend(w)
        #add documents in the corpus
        documents.append((w, intent['tag']))

        # add to our classes list
        if intent['tag'] not in classes:
            classes.append(intent['tag'])
```

Fig.18 Snapshot of Preprocess Data Code

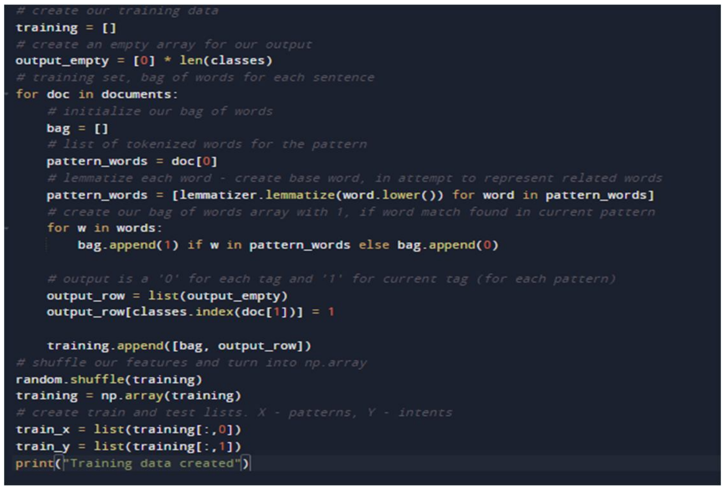
Now we will make word Lemmatization and delete duplicate words from the list. and then creating a pickle file to store the Python objects in which we will use while predicting.



```
# lemmatize, lower each word and remove duplicates
words = [lemmatizer.lemmatize(w.lower()) for w in words if w not in
    ignore_words]
words = sorted(list(set(words)))
# sort classes
classes = sorted(list(set(classes)))
# documents = combination between patterns and intents
print (len(documents), "documents")
# classes = intents
print (len(classes), "classes", classes)
# words = all words, vocabulary
print (len(words), "unique lemmatized words", words)

pickle.dump(words,open('words.pkl','wb'))
pickle.dump(classes,open('classes.pkl','wb'))
```

Fig.19 Snapshot of Pickle File Code

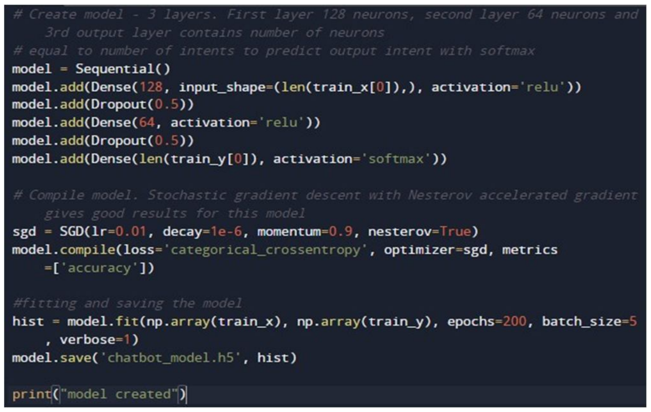
c) *Create Training and Testing Data:* Now, we will create the training data, we provide the input and the output. Our input will be the pattern and output will be the class, But the computer doesn't understand the text so that we will convert text into numbers.



```
# create our training data
training = []
# create an empty array for our output
output_empty = [0] * len(classes)
# training set, bag of words for each sentence
for doc in documents:
    # initialize our bag of words
    bag = []
    # list of tokenized words for the pattern
    pattern_words = doc[0]
    # lemmatize each word - create base word, in attempt to represent related words
    pattern_words = [lemmatizer.lemmatize(word.lower()) for word in pattern_words]
    # create our bag of words array with 1, if word match found in current pattern
    for w in words:
        bag.append(1) if w in pattern_words else bag.append(0)

    # output is a '0' for each tag and '1' for current tag (for each pattern)
    output_row = list(output_empty)
    output_row[classes.index(doc[1])] = 1

    training.append([bag, output_row])
# shuffle our features and turn into np.array
random.shuffle(training)
training = np.array(training)
# create train and test lists. X - patterns, Y - intents
train_x = list(training[:,0])
train_y = list(training[:,1])
print("Training data created")
```

Fig.20 Snapshot of Training Data Code

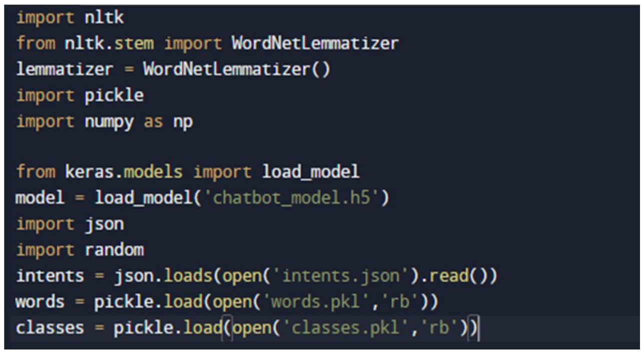
d) *Build the Model:* We have our training data ready, now we create a deep neural network that has 3 layers. We take the Keras sequential API for this. After training the model for 200 epochs, we reach 100% accuracy on our model. Let us save the model as 'chatbot_model.h5'.



```
# Create model - 3 layers. First layer 128 neurons, second layer 64 neurons and
    3rd output layer contains number of neurons
# equal to number of intents to predict output intent with softmax
model = Sequential()
model.add(Dense(128, input_shape=(len(train_x[0]),), activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(64, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(len(train_y[0]), activation='softmax'))

# Compile model. Stochastic gradient descent with Nesterov accelerated gradient
    gives good results for this model
sgd = SGD(lr=0.01, decay=1e-6, momentum=0.9, nesterov=True)
model.compile(loss='categorical_crossentropy', optimizer=sgd, metrics
    =['accuracy'])

#fitting and saving the model
hist = model.fit(np.array(train_x), np.array(train_y), epochs=200, batch_size=5
    , verbose=1)
model.save('chatbot_model.h5', hist)

print("model created")
```

Fig.21 Snapshot of Build the Model Code

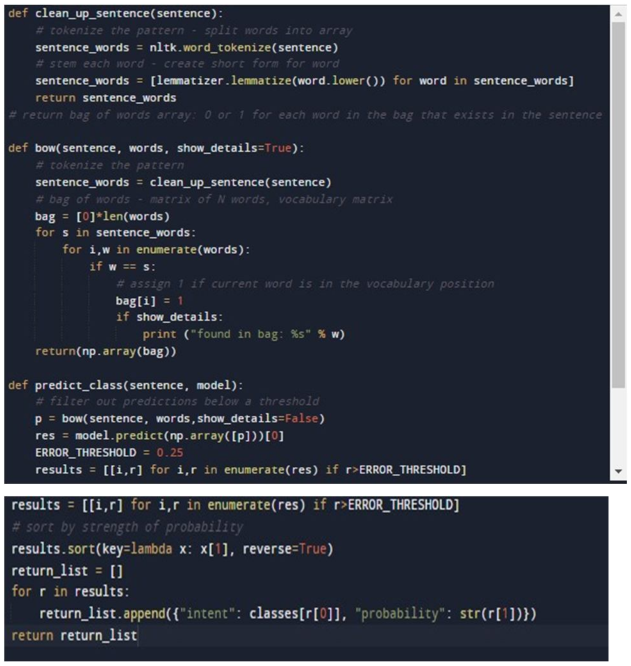
e) Predict the response (Graphical User Interface): Let us create a new file 'chatapp.py' for the predicted response of the user and also trained the bot to respond back. Once again, we import the important packages and load the 'words.pkl' and 'classes.pkl' pickle files which we have created when we trained our model:



```python
import nltk
from nltk.stem import WordNetLemmatizer
lemmatizer = WordNetLemmatizer()
import pickle
import numpy as np

from keras.models import load_model
model = load_model('chatbot_model.h5')
import json
import random
intents = json.loads(open('intents.json').read())
words = pickle.load(open('words.pkl','rb'))
classes = pickle.load(open('classes.pkl','rb'))
```

Fig.22 Snapshot of chatapp.py Code

predicting the class, we will need to provide input in the same as while training. So we will create some functions that will perform text preprocessing and then predict the class.

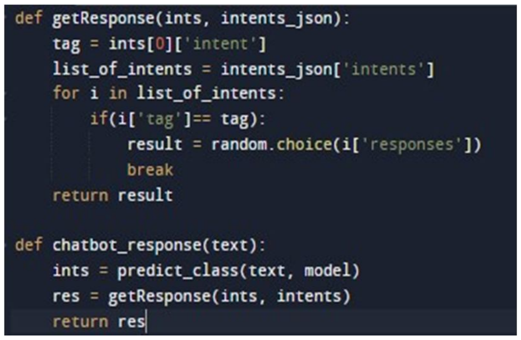


```python
def clean_up_sentence(sentence):
    # tokenize the pattern - split words into array
    sentence_words = nltk.word_tokenize(sentence)
    # stem each word - create short form for word
    sentence_words = [lemmatizer.lemmatize(word.lower()) for word in sentence_words]
    return sentence_words
# return bag of words array: 0 or 1 for each word in the bag that exists in the sentence

def bow(sentence, words, show_details=True):
    # tokenize the pattern
    sentence_words = clean_up_sentence(sentence)
    # bag of words - matrix of N words, vocabulary matrix
    bag = [0]*len(words)
    for s in sentence_words:
        for i,w in enumerate(words):
            if w == s:
                # assign 1 if current word is in the vocabulary position
                bag[i] = 1
                if show_details:
                    print ("found in bag: %s" % w)
    return(np.array(bag))

def predict_class(sentence, model):
    # filter out predictions below a threshold
    p = bow(sentence, words,show_details=False)
    res = model.predict(np.array([p]))[0]
    ERROR_THRESHOLD = 0.25
    results = [[i,r] for i,r in enumerate(res) if r>ERROR_THRESHOLD]
```

```python
results = [[i,r] for i,r in enumerate(res) if r>ERROR_THRESHOLD]
# sort by strength of probability
results.sort(key=lambda x: x[1], reverse=True)
return_list = []
for r in results:
    return_list.append({"intent": classes[r[0]], "probability": str(r[1])})
return return_list
```

Fig.23 Snapshot of Preprocessing Code

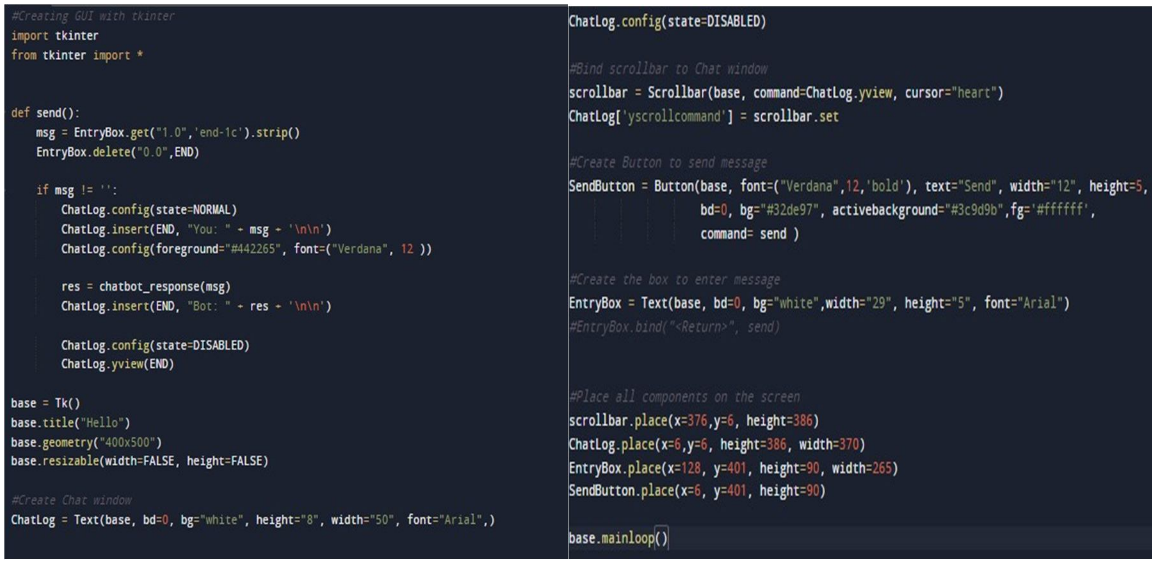
After predicting the class, we will get a random response from the list of intents.



```python
def getResponse(ints, intents_json):
    tag = ints[0]['intent']
    list_of_intents = intents_json['intents']
    for i in list_of_intents:
        if(i['tag']== tag):
            result = random.choice(i['responses'])
            break
    return result

def chatbot_response(text):
    ints = predict_class(text, model)
    res = getResponse(ints, intents)
    return res
```

Fig.24 Snapshot of Random Response Code

Now we will make a graphical user interface, and take a helper function to display the response of a bot.



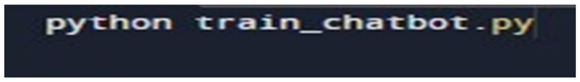
```
#Creating GUI with tkinter
import tkinter
from tkinter import *


def send():
    msg = EntryBox.get("1.0",'end-1c').strip()
    EntryBox.delete("0.0",END)

    if msg != '':
        ChatLog.config(state=NORMAL)
        ChatLog.insert(END, "You: " + msg + '\n\n')
        ChatLog.config(foreground="#442265", font=("Verdana", 12 ))

        res = chatbot_response(msg)
        ChatLog.insert(END, "Bot: " + res + '\n\n')

        ChatLog.config(state=DISABLED)
        ChatLog.yview(END)

base = Tk()
base.title("Hello")
base.geometry("400x500")
base.resizable(width=FALSE, height=FALSE)

#Create Chat window
ChatLog = Text(base, bd=0, bg="white", height="8", width="50", font="Arial",)
```

```
ChatLog.config(state=DISABLED)

#Bind scrollbar to Chat window
scrollbar = Scrollbar(base, command=ChatLog.yview, cursor="heart")
ChatLog['yscrollcommand'] = scrollbar.set

#Create Button to send message
SendButton = Button(base, font=("Verdana",12,'bold'), text="Send", width="12", height=5,
                    bd=0, bg="#32de97", activebackground="#3c9d9b",fg='#ffffff',
                    command= send )

#Create the box to enter message
EntryBox = Text(base, bd=0, bg="white",width="29", height="5", font="Arial")
#EntryBox.bind("<Return>", send)


#Place all components on the screen
scrollbar.place(x=376,y=6, height=386)
ChatLog.place(x=6,y=6, height=386, width=370)
EntryBox.place(x=128, y=401, height=90, width=265)
SendButton.place(x=6, y=401, height=90)

base.mainloop()
```

Fig.25 Snapshot of GUI and Helper Function Code

- *Run the chatbot*

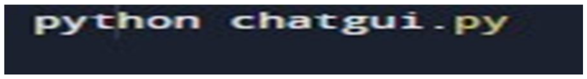
At first, we train the model using the command in the terminal:



```
python train_chatbot.py
```

Fig.26 First File

If we don't see any error in training, we have successfully created the model and run the second file.

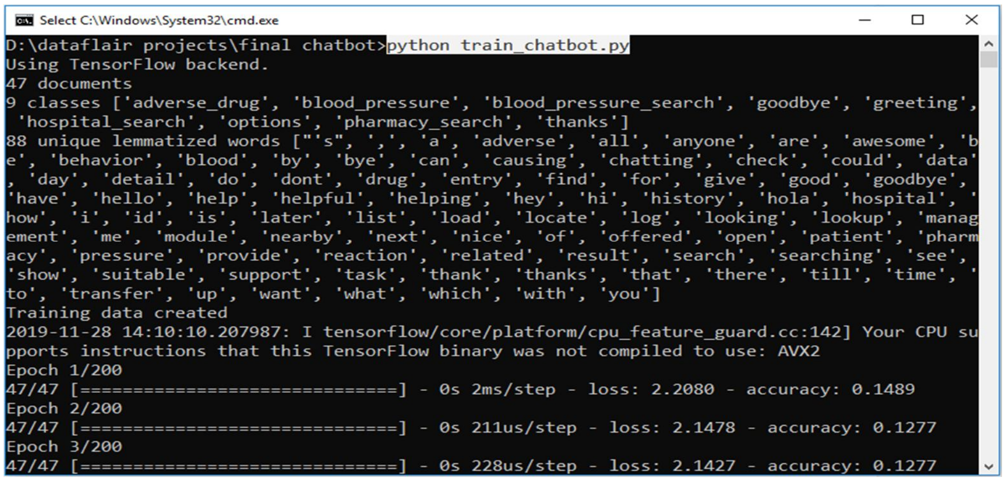


```
python chatgui.py
```

Fig.27 Second File

- *Screenshots*

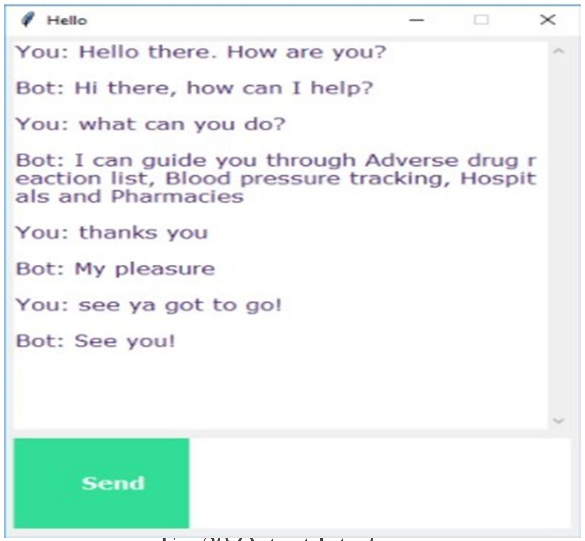
This programme will open in a moment



```
D:\dataflair projects\final chatbot>python train_chatbot.py
Using TensorFlow backend.
47 documents
9 classes ['adverse_drug', 'blood_pressure', 'blood_pressure_search', 'goodbye', 'greeting',
 'hospital_search', 'options', 'pharmacy_search', 'thanks']
88 unique lemmatized words ["'s", ',', 'a', 'adverse', 'all', 'anyone', 'are', 'awesome', 'b
e', 'behavior', 'blood', 'by', 'bye', 'can', 'causing', 'chatting', 'check', 'could', 'data'
, 'day', 'detail', 'do', 'dont', 'drug', 'entry', 'find', 'for', 'give', 'good', 'goodbye',
'have', 'hello', 'help', 'helpful', 'helping', 'hey', 'hi', 'history', 'hola', 'hospital', '
how', 'i', 'id', 'is', 'later', 'list', 'load', 'locate', 'log', 'looking', 'lookup', 'manag
ement', 'me', 'module', 'nearby', 'next', 'nice', 'of', 'offered', 'open', 'patient', 'pharm
acy', 'pressure', 'provide', 'reaction', 'related', 'result', 'search', 'searching', 'see',
'show', 'suitable', 'support', 'task', 'thank', 'thanks', 'that', 'there', 'till', 'time', '
to', 'transfer', 'up', 'want', 'what', 'which', 'with', 'you']
Training data created
2019-11-28 14:10:10.207987: I tensorflow/core/platform/cpu_feature_guard.cc:142] Your CPU su
pports instructions that this TensorFlow binary was not compiled to use: AVX2
Epoch 1/200
47/47 [==============================] - 0s 2ms/step - loss: 2.2080 - accuracy: 0.1489
Epoch 2/200
47/47 [==============================] - 0s 211us/step - loss: 2.1478 - accuracy: 0.1277
Epoch 3/200
47/47 [==============================] - 0s 228us/step - loss: 2.1427 - accuracy: 0.1277
```

Fig.28 Screenshot 1

Our chatbot is ready to run.



```
You: Hello there. How are you?

Bot: Hi there, how can I help?

You: what can you do?

Bot: I can guide you through Adverse drug r
eaction list, Blood pressure tracking, Hospit
als and Pharmacies

You: thanks you

Bot: My pleasure

You: see ya got to go!

Bot: See you!
```

Fig.29 Output Interface

REFERENCES

[1] Introduction to Programming Using Python by Author Liang Y. Daniel
[2] Introduction to Machine Learning Yahoo! Labs Santa Clara and Departments of Statistics and College of Engineering and Computer Science, Computer Science Purdue University Australian National University taken from https://alex.smola.org/drafts/thebook.pdfIntroduction to Data Science Michel by Author D. S. Mesquita
[3] Python Data Science Handbook by Jake VanderPlas, released- November 2016, Publisher(s): O'Reilly Media, Inc. ISBN: 9781491912058
[4] Introduction to IOT by Author Pradyumna Gokhale, Omkar Bhat from University of Virginia, Sagar Bhat from Smt. Kashibai Navale College of Engineering
[5] https://jakevdp.github.io/PythonDataScienceHandbook/02.07-fancy-indexing.html
[6] https://www.guru99.com/what-is-tensorflow.html
[7] http://alvarestech.com/temp/smar/Smar/Book2021/Industry4.0/Rodolfo%20BonninBuilding%20Machine%20Learning%20Projects%20with%20TensorFlow-Packt%20(2016).pdf
[8] https://www.perlego.com/book/4637/internet-of-things-with-python- by Gaston C. Hillar pdf
[9] https://searchcustomerexperience.techtarget.com/definition/chatbot
[10] https://data-flair.training/blogs/python-chatbot-project/