**Ahmed Abdiqadir Dahir**

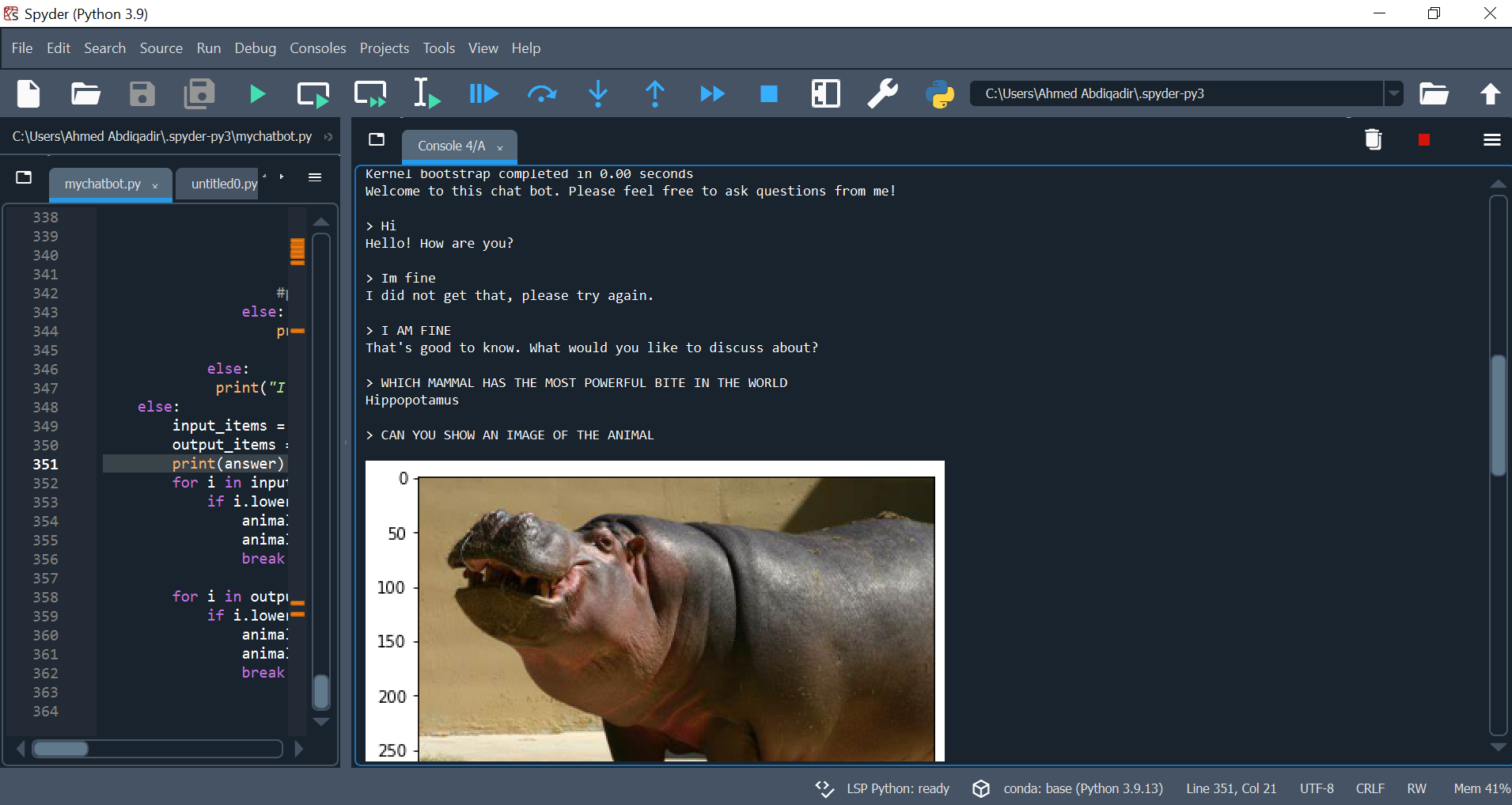
**T0269931**

**Artificial Intelligence**

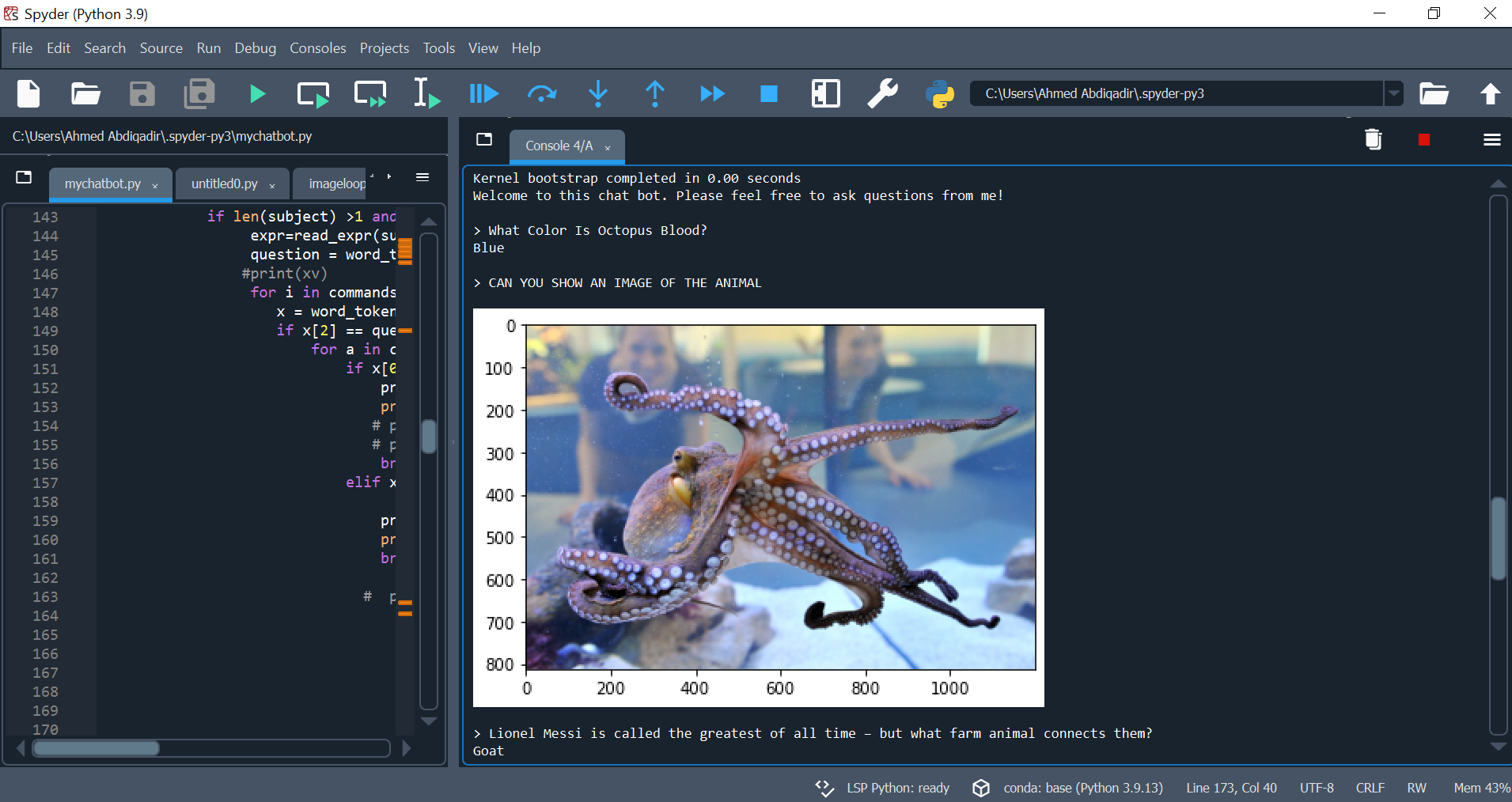
This documentation will illustrate all the features and functionalities that I have created for my chatbot.

Task1

One of the features included in the chatbot will allow users to ask the chatbot questions and will then respond with the correct answer to question. Furthermore, the chatbot loops through both the question and answer to find the animal in question and saves its information. This will enable the user to ask for a picture which will allow the chatbot to display it.



The question and answers in the csv will also follow the same principle. The role of the csv file is to check for the closest question that the user asks which is not part of aiml file. It also follows the same format whereby the user is able to ask for picture of the animal being discussed. An example of this is shown below:



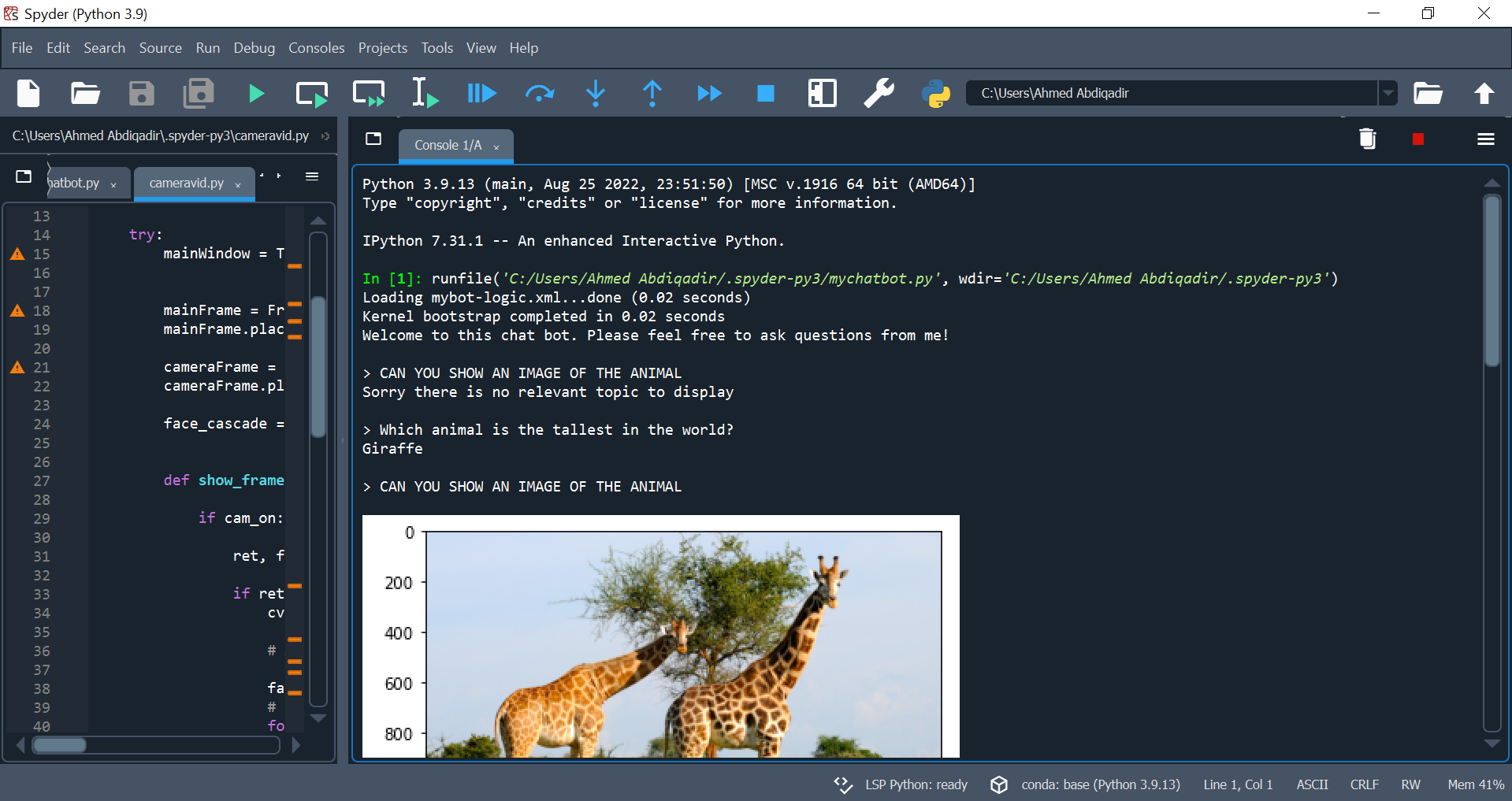
A screenshot of a computer

Description automatically generated with medium confidence

As you can see from the displays above, the chatbot is able to check for the animal being discussed from both question and answer and save it in its knowledge base. In our first example, the topic being discussed is octopus but as you can see it cannot be found in the answer. But the chatbot is able to do an operation that will enable it to extract the topic. After each log of question, the information on the knowledge is cleared to allow room for a new topic.

EXTRA TASK -`In this section, I also added a grammar model that checks for any misspelt word or the structure of the sentence. It will the recommend a new sentence which will then be used by the chatbot for its operations.

If we were to run the command when there is no topic in place, for example, When the chatbot first executes, their will be no display of image. The process is shown below:



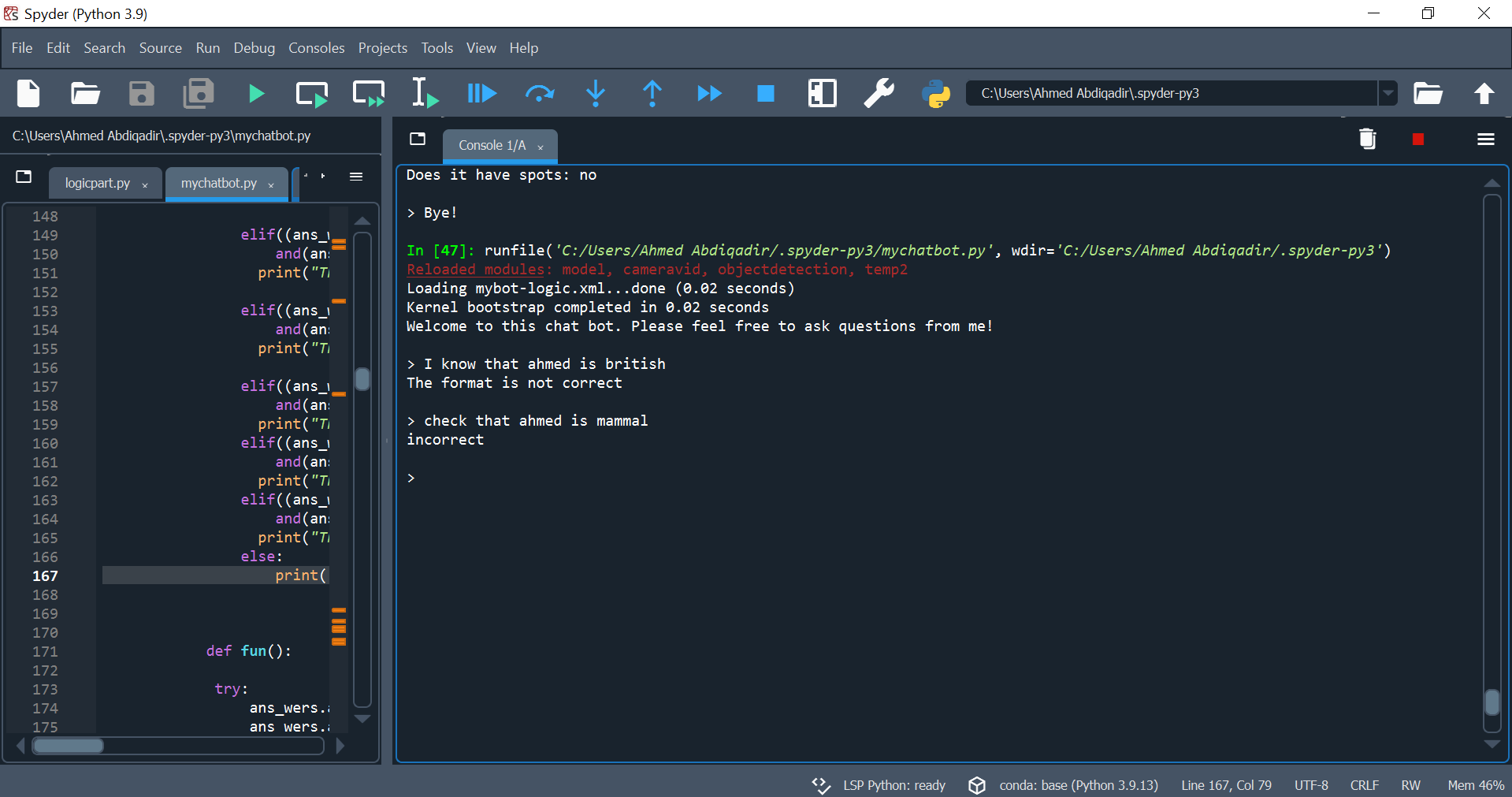
Task 2

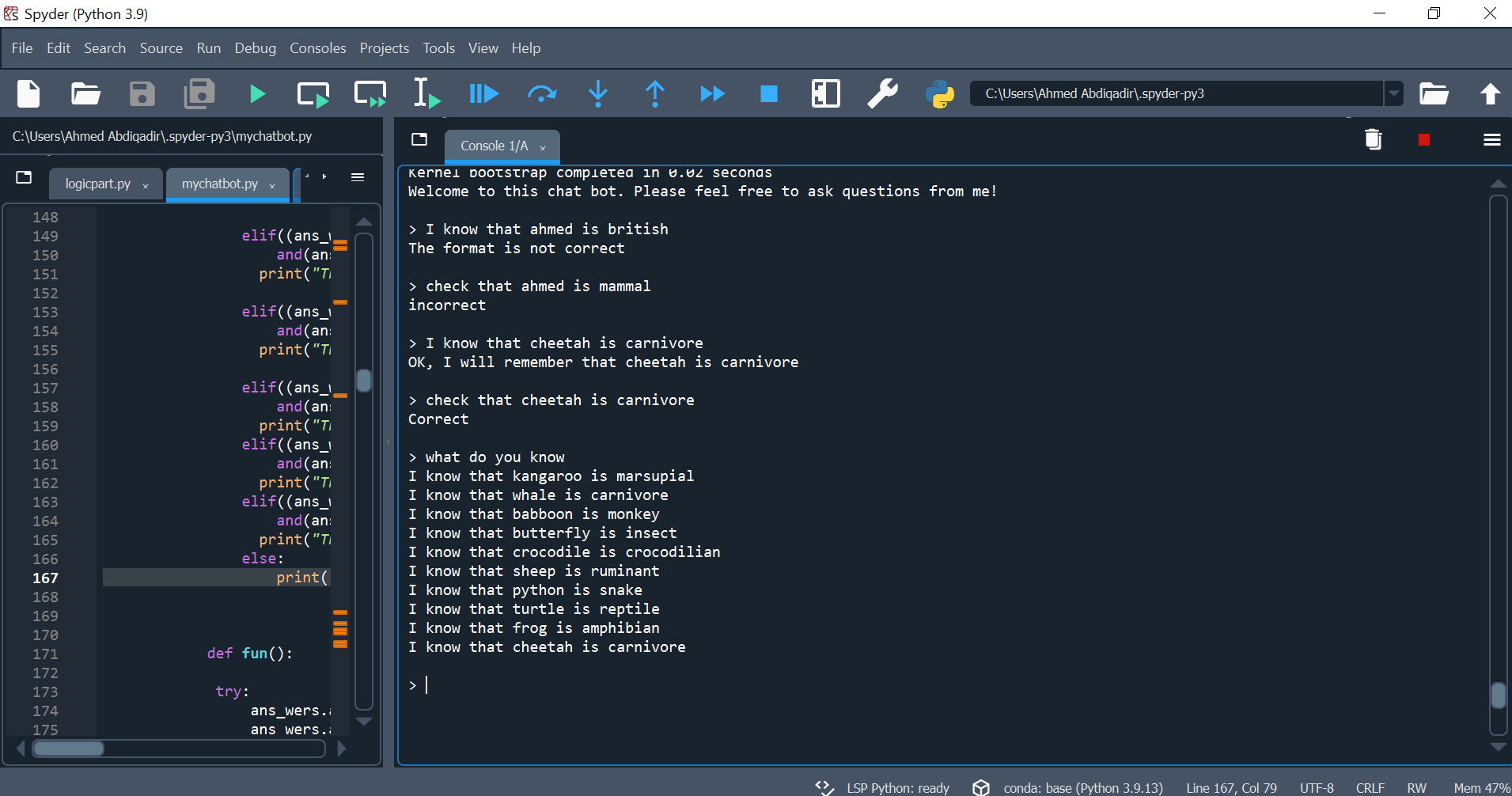
One of the features of this section is a first order logic implementation. The chatbot interacts with the knowledge base whenever a user inputs the commands “I know that \* is \*”,“I know that \* is \*” and what do you know.

The structure of the knowledge base is shown below:

|  |
| --- |
| Marsupial (x) -> Mammal (x) |
| Monkey (x) -> Primate (x) |
| Primate (x) -> Mammal (x) |
| Mammal (x) -> Animal (x) |
| Cat (x) -> Carnivore (x) |
| Ruminant (x) -> Herbivore (x) |
| Herbivore (x) -> Mammal (x) |
| Carnivore (x) -> Mammal (x) |
| Snake (x) -> Reptile (x) |
| Crocodilian (x) -> Reptile (x) |
| Reptile (x) -> Animal (x) |
| Amphibian (x) -> Animal (x) |
| Insect (x) -> Animal (x) |

The user is only able to interact with kb if the expression is correct. An example when the users input contradicts the knowledge base is shown below:

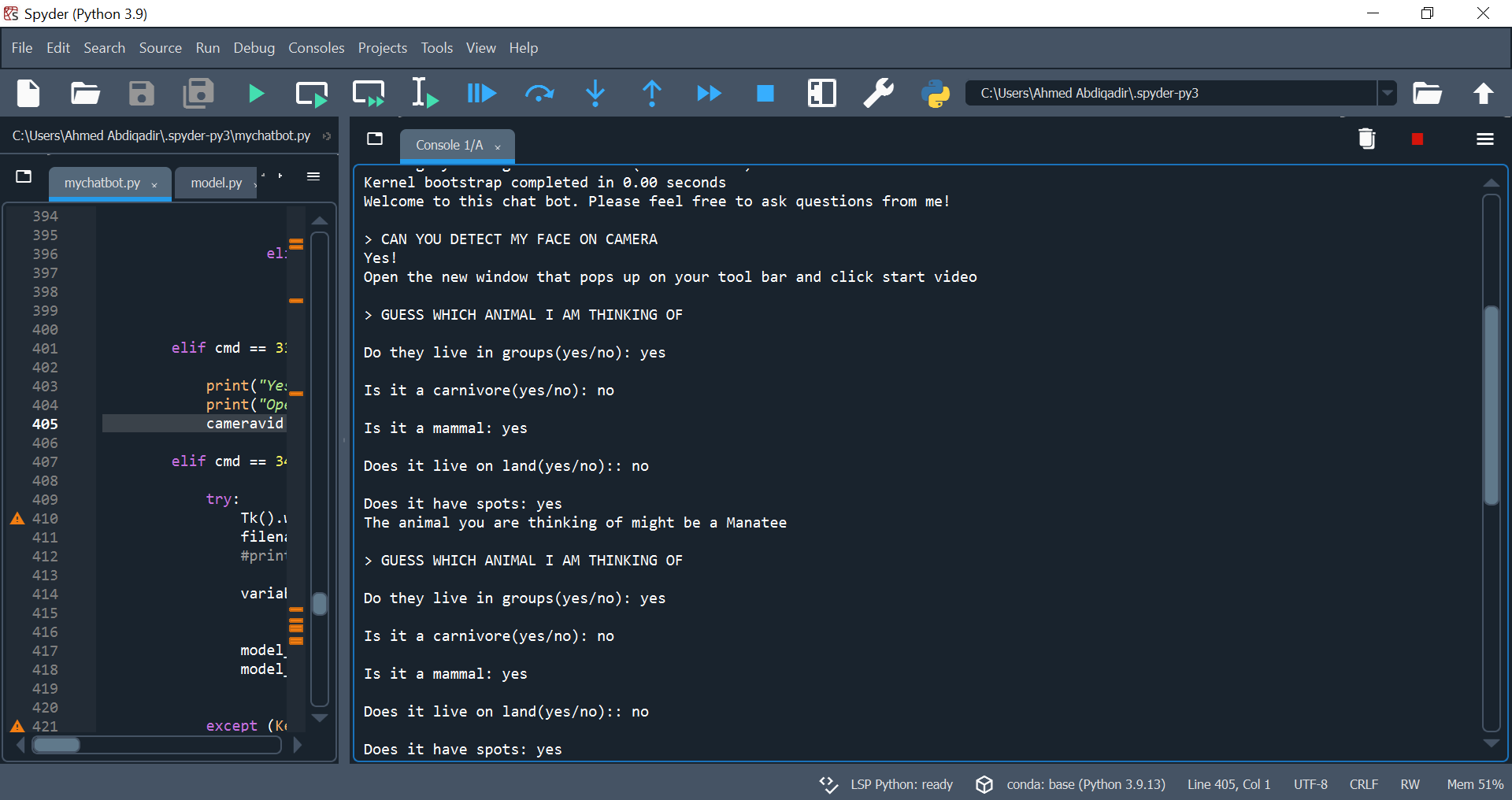


An illustration of the FOL statement operations is shown below

Extra task For task 2:

In this section there is a fuzzy logic operation. This operation is able to guess what the user is thinking about. It asks the user different types of questions and then checks the if statement for the relevant answer.

It requires the command “GUESS WHICH ANIMAL I AM THINKING OF”



TASK 3

One of the features and functionalities that has been included in the chatbot is a trained CNN model with three classes. The classes are Elephant, Lion and Zebra.

The first step before the CNN model was trained was to collect images on each class. The images for each class was then divided into two sections i.e. the training set and validation set before feeding it to the model.

image\_size = (360, 360)

batch\_size = 128

train\_ds, val\_ds = tf.keras.utils.image\_dataset\_from\_directory(

data\_dir,

validation\_split=0.2,

subset="both",

seed=1337,

image\_size=image\_size,

batch\_size=batch\_size,

)

Afterwards the model was trained and saved to a h5 file in the computer.

Using : model.save("mycnnmodel.h5", include\_optimizer=True)

The model is then loaded to perform the operations that are needed.

mymodel = keras.models.load\_model(".spyder-py3/mycnnmodel.h5")

img = keras.preprocessing.image.load\_img(

".spyder-py3/data/ellee.jpg", target\_size=(360,360)

)

img\_array = keras.preprocessing.image.img\_to\_array(img)

img\_array = tf.expand\_dims(img\_array, 0) # Create batch axis

predictions = modelmine.predict(img\_array)

print(predictions)

score = (predictions[0])

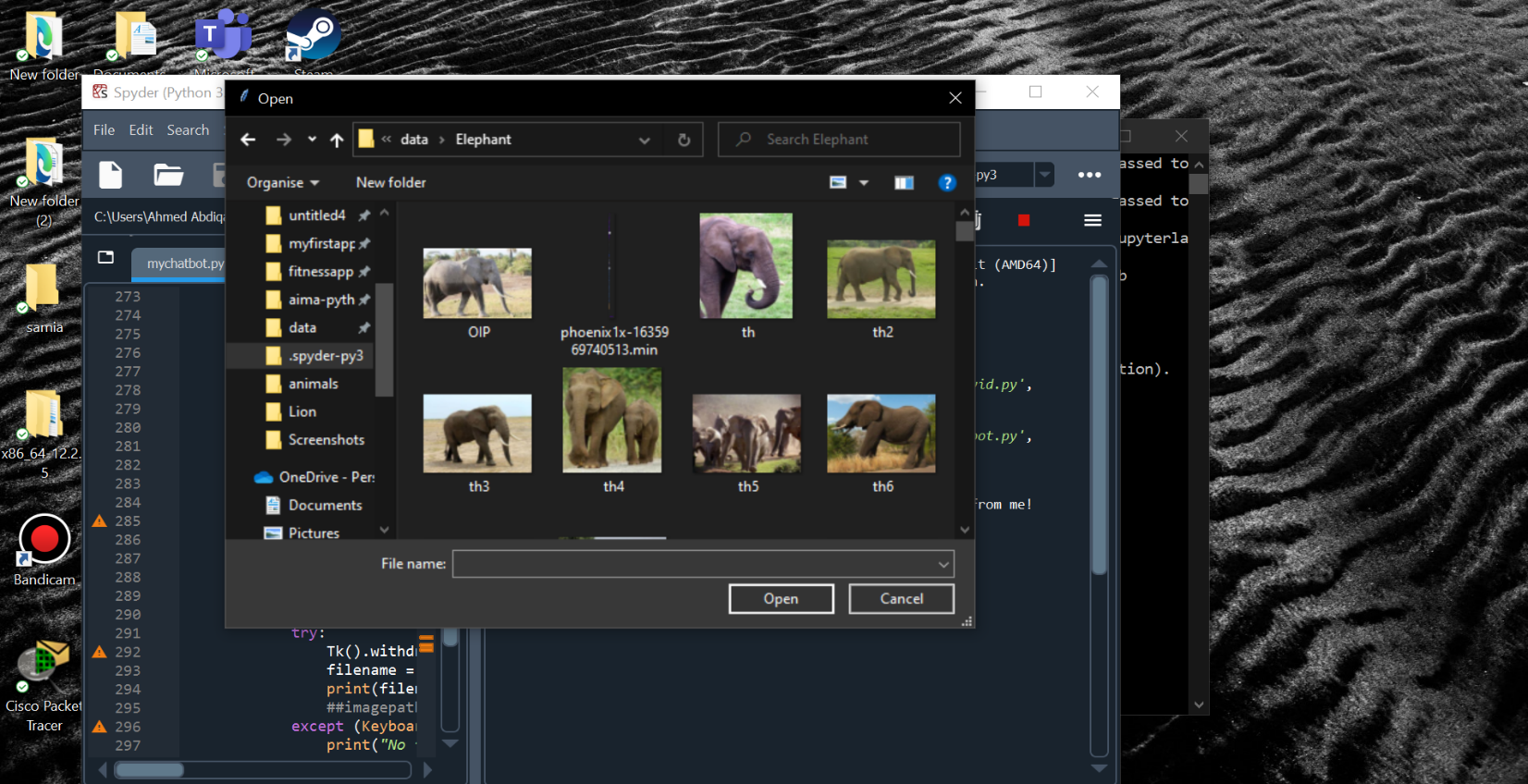
The score that has the largest number will therefore predict which class the test image will belong to.

1/1 [==============================] - 0s 72ms/step

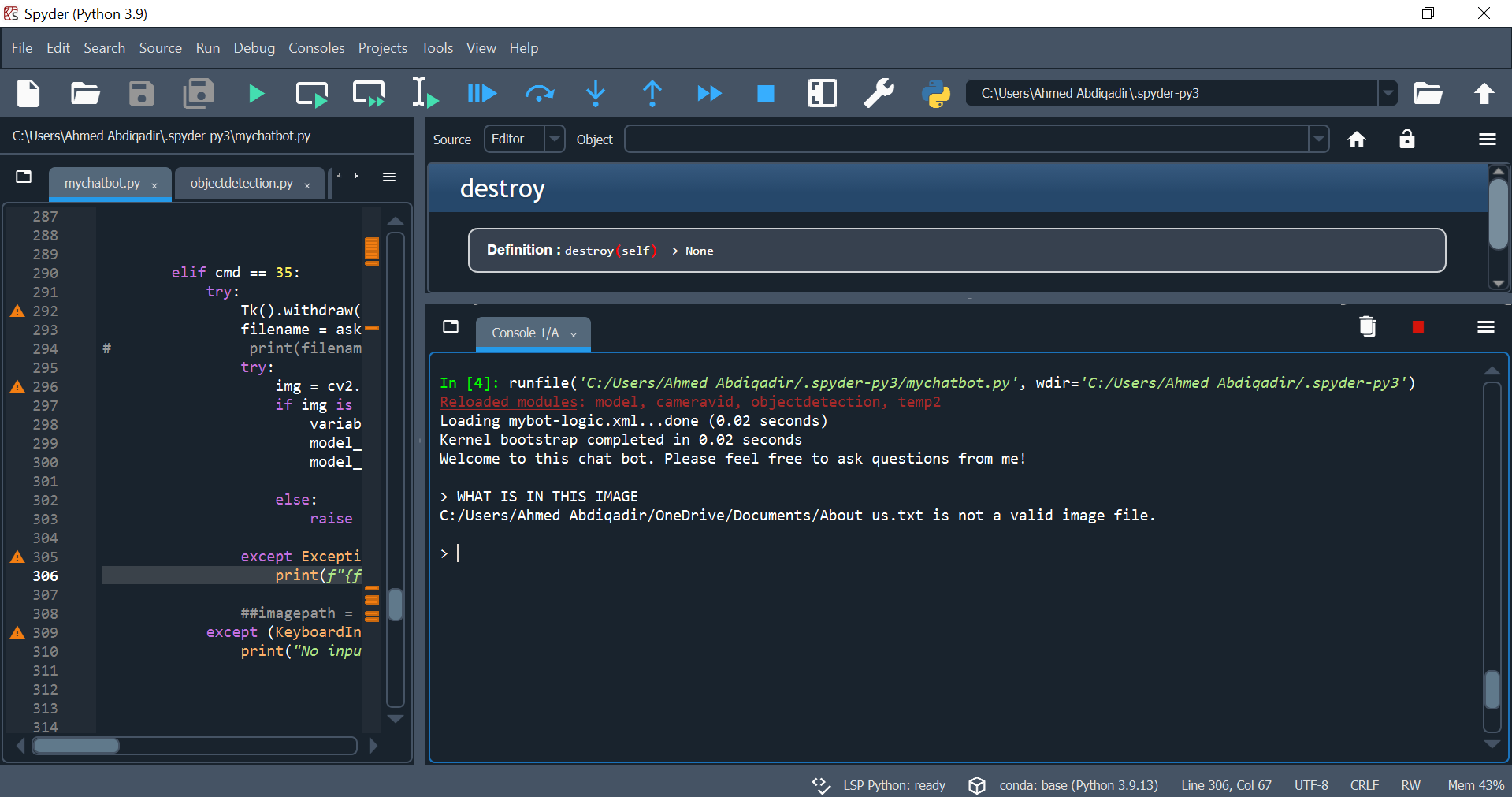
[[ 3.190668 -4.1815176 0.5335471]]

The demonstration of all the above is shown below:

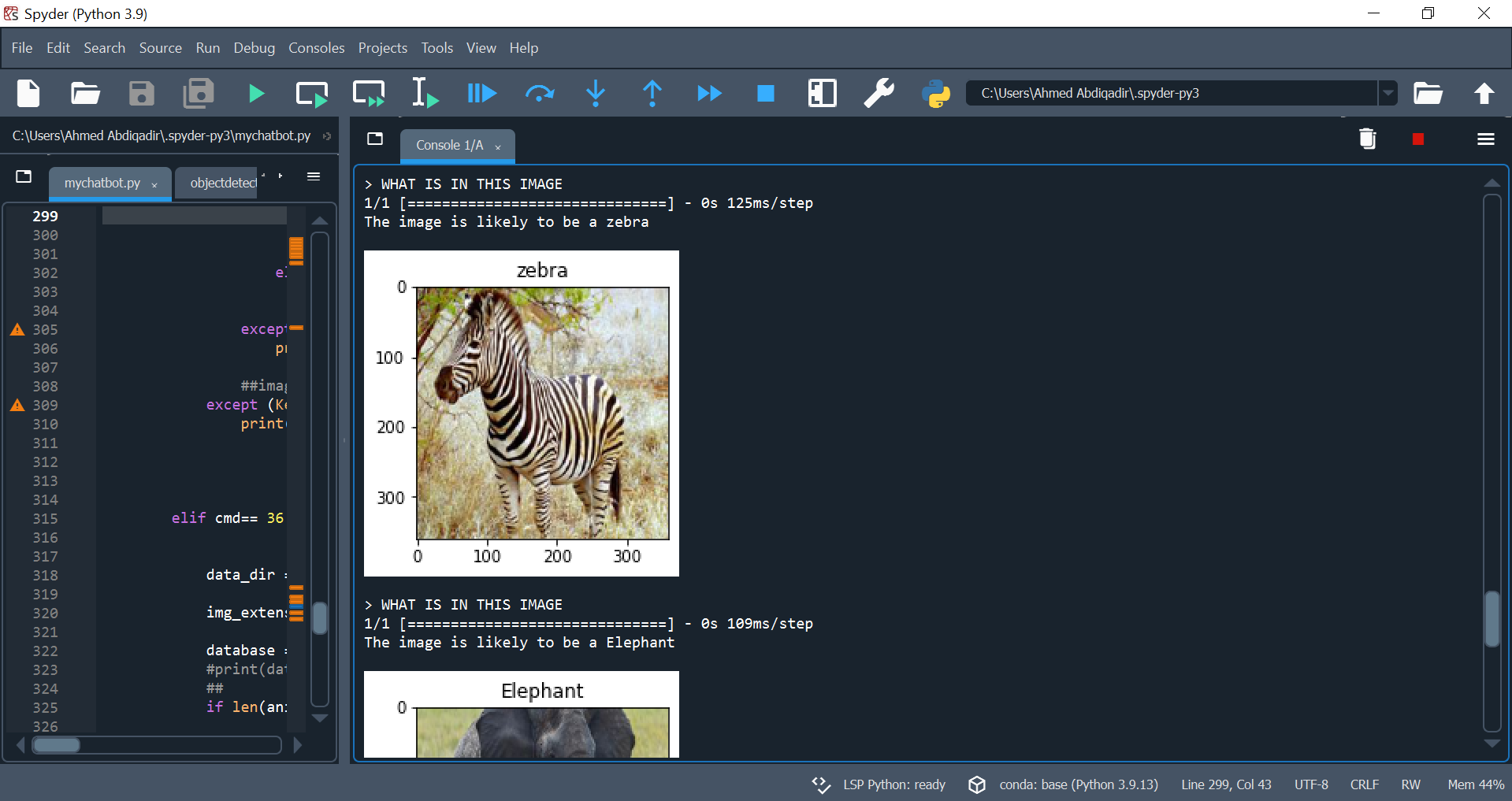
It will need the command “WHAT IS IN THIS IMAGE”

Afterwards, the chatbot opens up a file explorer of its own and allow the user to pick an image from his files. 

The selected file must be an image file otherwise this error will pop up.



And finally, below is the demonstration of the operation performed in Task three CNN model. The chatbot prints the name of the animal and also plots its image in the console



Extra task for task 3:

->There is also a object dejection pre trained model feature in our chatbot. The model is known as EfficientDet-Lite2. It was developed by tensorflow hub with almost 93 pre trained classes comprising of different animals and objects. The model is first loaded and an image of the users choosing will be detected by model as long as it is part of the trained classes.

The command “WHAT ARE THE OBJECTS IN THIS PICTURE” will be used to complete the operation. It also contains the same feature as the CNN model when it comes to fetching the image from the directory.

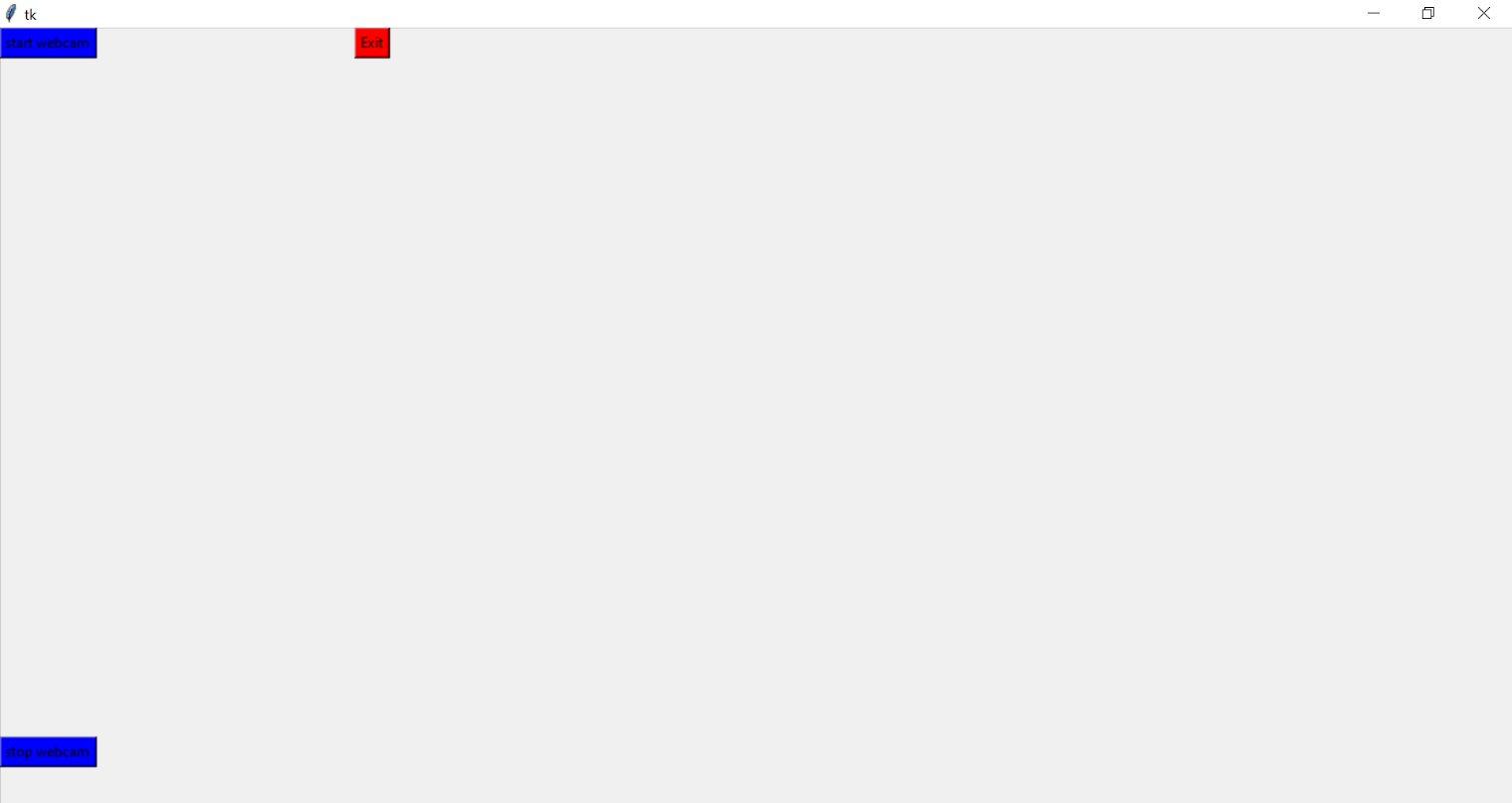
An illustration of object detector is shown belowGraphical user interface

Description automatically generated

As you can see, there are boxes with labels displayed on the image identifying the contents of the image.

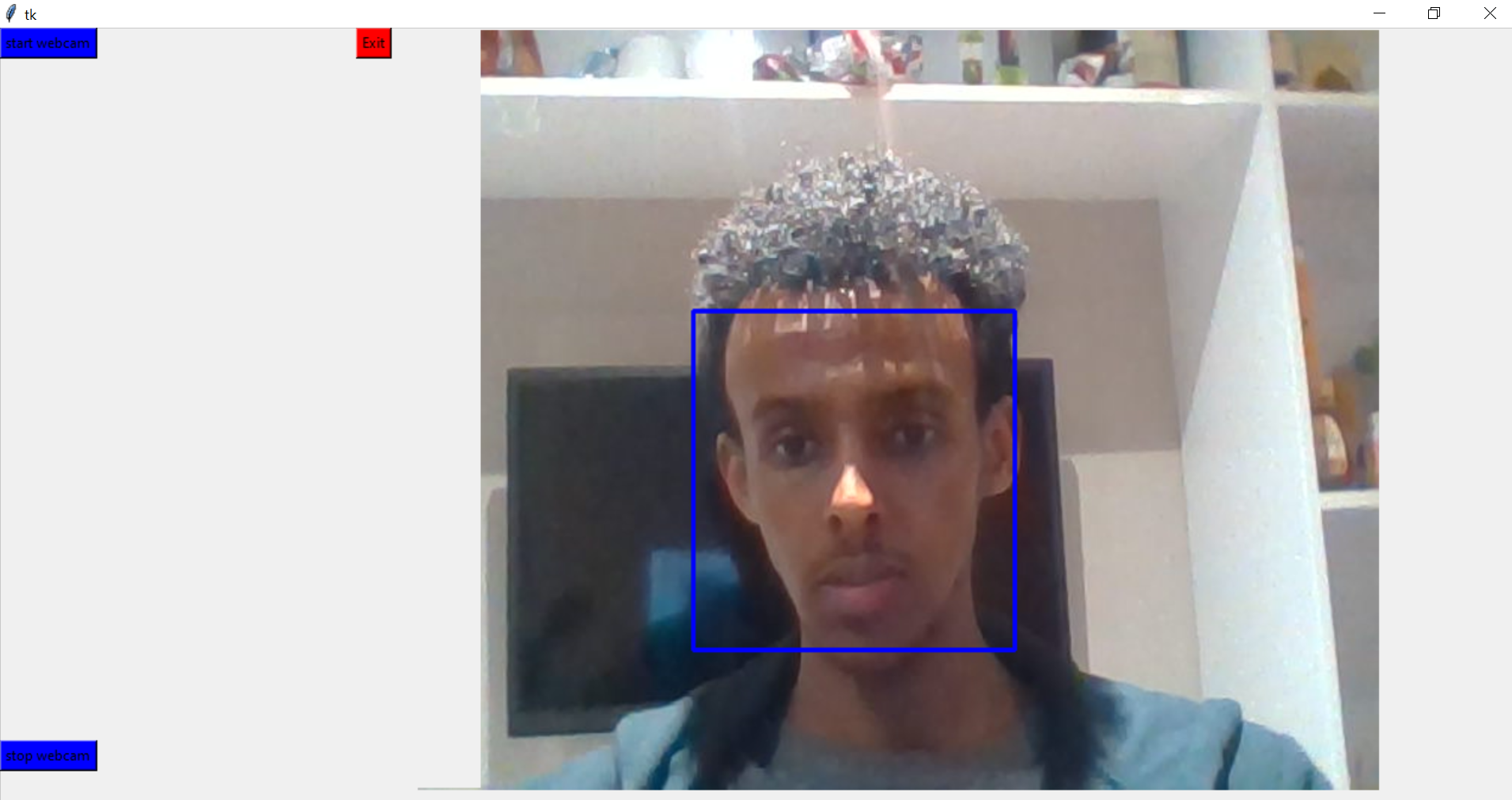
->In this section, we also have a face detection feature. It requires the command “CAN YOU DETECT MY FACE ON CAMERA”. The feature was implemented using tkinter and a pre trained OpenCV model known as haarcascade\_frontalface\_default.XML.

The purpose of tkinter was to enable a face detection graphical user interface that contains buttons which will be used to perform actions such us turning on the webcam, pausing the webcam and allowing the user to exit the GUI application. The picture below demonstrates a GUI application when the webcam is off.



“haarcascade\_frontalface\_default.xml“ is a face detection model that is able to detect faces in videos ,pictures and also webcam.

We can see the operations of the face detector in the figure below:



Reference list

opencv (2019). *opencv/opencv*. [online] GitHub. Available at: https://github.com/opencv/opencv.

tfhub.dev. (n.d.). *TensorFlow Hub*. [online] Available at: https://tfhub.dev/tensorflow/efficientdet/lite2/detection/1 [Accessed 15 Mar. 2023].

Images - web