Assignment #9 - Chatbot

The Assignment contains the following subjects:

- Python syntax
- Bottle web framework

Submitting instructions:

- You should submit this assignment **alone**
- Push the full folder hierarchy of the project to your own repository on GitHub.
  Please add the following in hive:
  - A link to the repository
  - Free text - a description of the quiz. Stuff that you found hard to implement, known bugs and your review of this assignment

Understanding the task

A chatter robot is a type of conversational agent, a computer program designed to simulate an intelligent conversation with one or more human users via auditory or textual methods.
In our version of the concept, we will use a client side (already implemented for you) that will interact with the user and a server side that will process the user's sentences and try to come up with a reasonable answer.
Implementing a convincing chatter bot is a complex task that requires skills from the fields of Natural Language Processing, Artificial intelligence and Machine learning. We do not expect your solutions to pass the Turing test…
Moreover; our server side is stateless which means that each request does not hold information about what happened before. Boto does not "remember" anything (only the current sentence)

The interaction with the user is straightforward

1. The user will Type a sentence in the input field *(implemented for you)*
2. The app will send the sentence to the server using the POST method *(implemented for you)*
3. The server will process the sentence (**your part**, more on that later)
4. The result will be displayed to the user textually and with the relevant animation *(implemented for you)*

Approaching the assignment

1. Fork the project on GitHub.
2. Go through the provided code, see that you understand how the client side of the app works (basically you can run the app without a server although Boto will alert you that he encountered an error)
3. Create a simple Bottle server that handles the /chat URL for POST, return the same result no matter what the user's message is.
4. Dive into the sentence processing code.

Basic Requirements

1. Use all of the animations provided
2. Boto should reasonably respond to different types of sentences
3. Boto should respond to swear words
    a. use the 'any' function to check for swear words (hint: use user_message.split(" ") to separate the words in the user's message)
4. Use string formatting in one of Boto's responses

Implementation constraints

1. You are not allowed to change anything in the provided code. You should submit the server side only
2. Write at least 3 server side functions

Tips

1. Use simple classification algorithm to find out the sentence type like "ends with ?" , "starts with I" , "user cursing exists" etc.
2. Process each sentence type in a separated function

Geek out

Add the ability to request a joke from Boto. That joke should be randomly selected from a predefined jokes archive.

Unleash the ninja within

1. Try to overcome the statelessness of the server by using cookies and making your Boto smarter
2. Allow Boto to get the weather, using an API call to an external service