Chat-bot: silly-bot

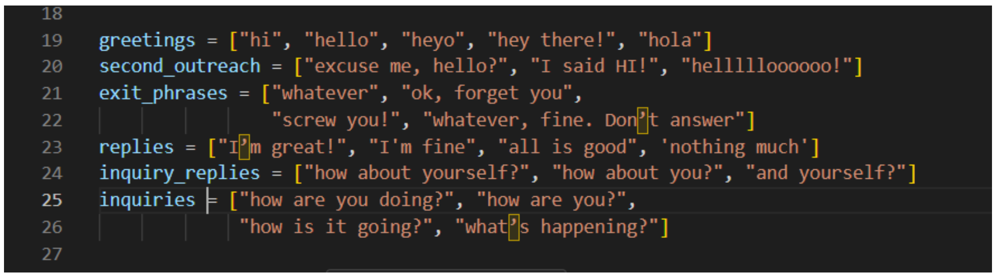
Report by: Mihika Nigam & Kedarnath Chaturvedula

Description

The chatbot is designed to operate on the IRC (Internet Relay Chat) platform with the purpose of engaging in greeting conversations with users in a specified channel.Using sockets, we connected the bot to the IRC server. And nicknamed our bot silly-bot'. We then ran the script which enabled the bot to join the channel and sent commands and requests which gave appropriate responses.

Implementation

To allow our script to connect to the irc chat application we used sockets. To handle the timeout for our application
The bot uses spacy (nlp) to break the command into tokens and determine if the command entered is a 'Greeting', a 'Reply' or an 'Enquiry'.
Below is a set of greetings, inquiries and exit commands that the bot takes care of.



```
greetings = ["hi", "hello", "heyo", "hey there!", "hola"]
second_outreach = ["excuse me, hello?", "I said HI!", "helllllooooo!"]
exit_phrases = ["whatever", "ok, forget you",
                "screw you!", "whatever, fine. Don't answer"]
replies = ["I'm great!", "I'm fine", "all is good", 'nothing much']
inquiry_replies = ["how about yourself?", "how about you?", "and yourself?"]
inquiries = ["how are you doing?", "how are you?",
            "how is it going?", "what's happening?"]
```

The main idea behind the bot is to manage states after each query or text in the chat. We have the following main states for bot:

The whole idea was to create an FSM model to handle the states. Just as shown in the prompt.

We built an extra State class for clarity.

Use of Data Structures

The code to receive commands from the server constantly keeps pinging with the latest command which made the bot keep running the for eg 'forget' command again and again which led to duplicate responses. To handle this we used stack to keep track of the last command entered. If the last command is the exact same as the new ping, the bot is taught to not repeat that again.
Simple python lists and dicts are used to handle different types of queries, and responses.

Extra Features

1. Invention (added by Mihika): This is an extra flavor to bot where the bot reads from the given command a command like so `Where was <ITEM> invented?`. To evaluate this I completed the following steps:
> Search google for the query
> From the search results, use beautiful soup to extract the info I need to get output.
> To extract I made use of a particular class of the 1st query that comes in google. If this query fails or doesn't comeup, the bot returns with a valid "Not Found" Response.
> After Extracting we use Spacy's ent.label property to extract the accurate NER then use that to get the PLACE and DATE entity to get the ans. The output is the g

2.