CHATBOT HACKATHON

OVERVIEW & PURPOSE:

The goal of this hackathon is to build a conversational bot to interact with the user and achieve the desired outcome through the conversation. We shall take a two-pronged approach towards this task.

- Use Amazon Alexa APIs to build a bot easily by defining the desired utterances and intents
- See how a bot may be built from scratch and understand the underlying logic

UNDERSTANDING CHATBOTS

Understand the basic definitions of a chatbot and how Alexa Skills help in building them

- What is a Chatbot?
  - Conversational AI chat-bot — Architecture overview | by Ravindra Kompella | Towards Data Science
- What is Alexa Skills Kit and how does it help with building a chatbot?
  - The Alexa Skills Kit is a software development kit (SDK) that enables a developer to build skills, also called conversational applications, on the Amazon Alexa artificial intelligence assistant

GETTING READY

- Create an **Amazon Developer Account**:
  - To build any type of skill, you need an Amazon developer account. Go to the developer console at https://developer.amazon.com/. You can create a new Amazon account. Follow the prompts to enter your registration information and give consent to the Amazon Developer Services Agreement. Once you have done this, you can build skills for Alexa using this account.

- Learn the **Nuances of dialog management in Alexa**:
  - This is critical in building a good, interactive, and intelligent chatbot. Efforts in understanding this will contribute to building a good chatbot. A clear understanding of the different aspects of Alexa Dialog's management will help to crack the hackathon easily.

TO UNDERSTAND ABOUT AMAZON ALEXA

Go through the below links:

- In this link, you will understand how to build Alexa.
- Click here to understand how utterances, Intents, and slots are used in Alexa.

CHATBOT ARCHITECTURE

The chatbot is made to be provided as a pluggable service, and configurable through configuration files, in addition to modifying code.

CONCEPTS

Intent: This is a particular skill of a chatbot. It could be a get Zodiac Sign or Movie Suggest, etc. Each intent has a few parameters, which are basically information required to be elicited by the user for the chatbot to complete the particular intent. Example: birth date of the user, language preferences, etc.

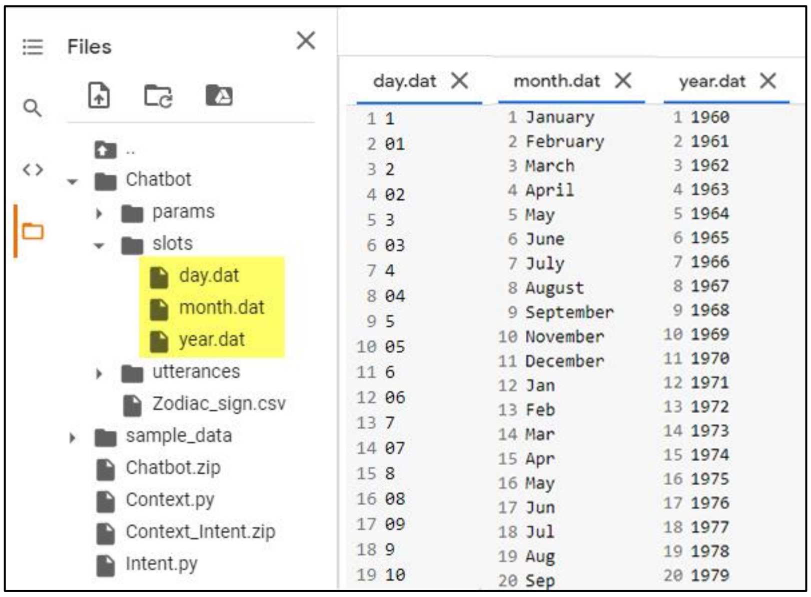
Actions: Each intent also had action that is a function call after all the parameters have been fulfilled.

CONFIGURATION FILES

Slots Folder:

Here each file contains example slots.

Note: All the slot values are case-sensitive



```
..
Chatbot
  params
  slots
    day.dat
    month.dat
    year.dat
  utterances
  Zodiac_sign.csv
sample_data
Chatbot.zip
Context.py
Context_Intent.zip
Intent.py
```

day.dat

```
1
01
2
02
3
03
4
04
5
05
6
06
7
07
8
08
09
9
10
```

month.dat

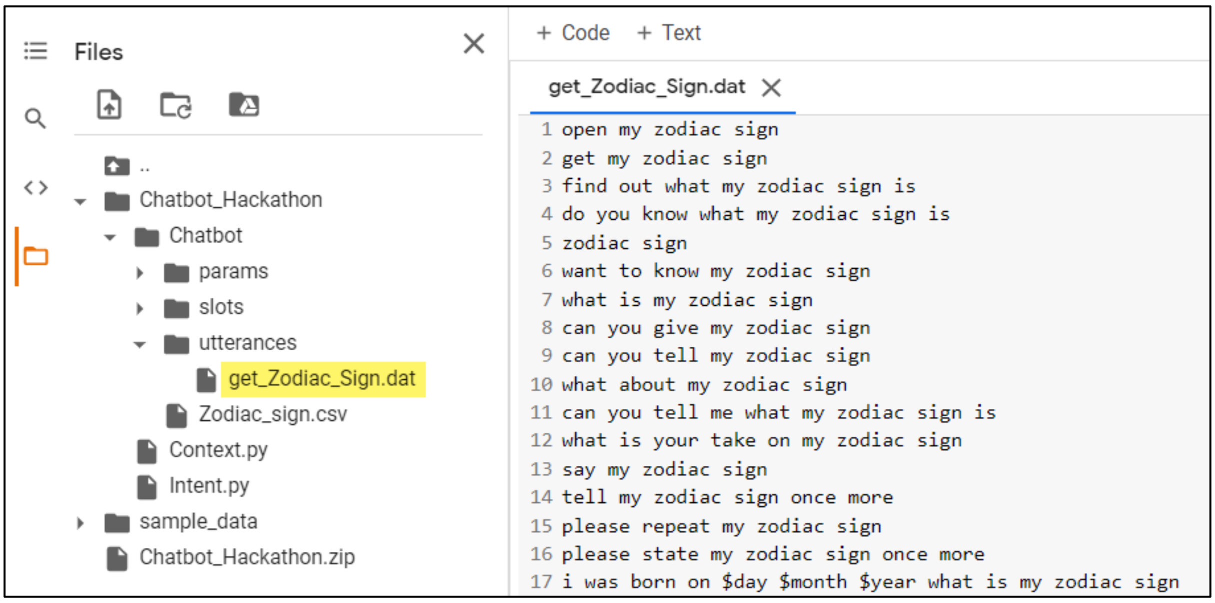
```
January
February
March
April
May
June
July
August
September
November
December
Jan
Feb
Mar
Apr
May
Jun
Jul
Aug
Sep
```

year.dat

```
1960
1961
1962
1963
1964
1965
1966
1967
1968
1969
1970
1971
1972
1973
1974
1975
1976
1977
1978
1979
```

Utterances Folder:

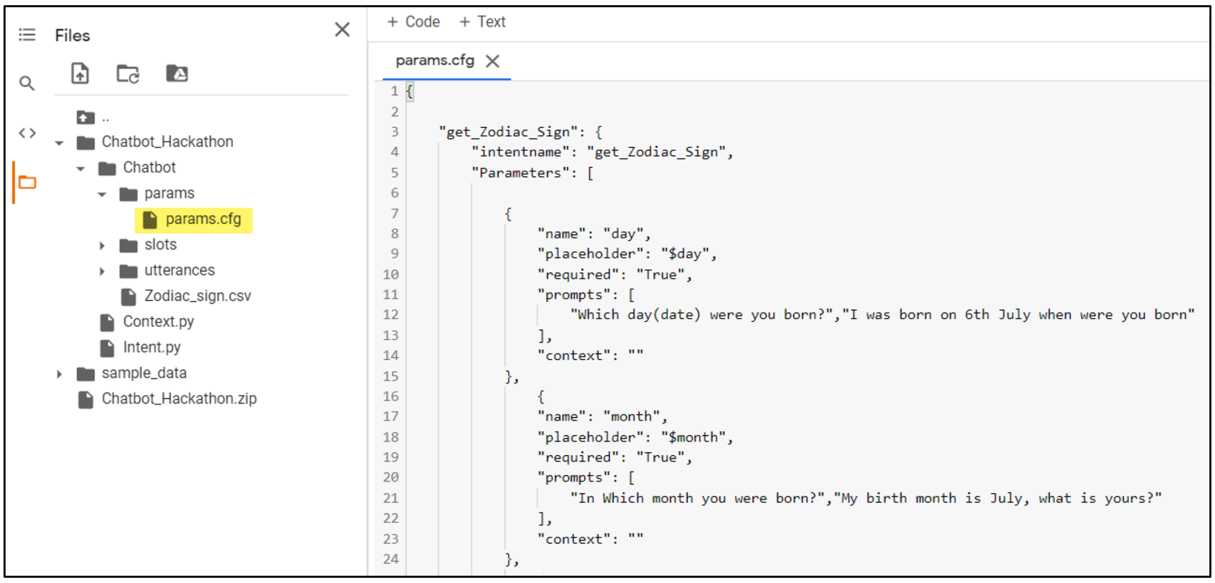
Here each file contains example utterances for each intent. Note the placeholders starting with $for entity masking (as done by get_attributes())



```
open my zodiac sign
get my zodiac sign
find out what my zodiac sign is
do you know what my zodiac sign is
zodiac sign
want to know my zodiac sign
what is my zodiac sign
can you give my zodiac sign
can you tell my zodiac sign
what about my zodiac sign
can you tell me what my zodiac sign is
what is your take on my zodiac sign
say my zodiac sign
tell my zodiac sign once more
please repeat my zodiac sign
please state my zodiac sign once more
i was born on $day $month $year what is my zodiac sign
```

params/params.cfg :

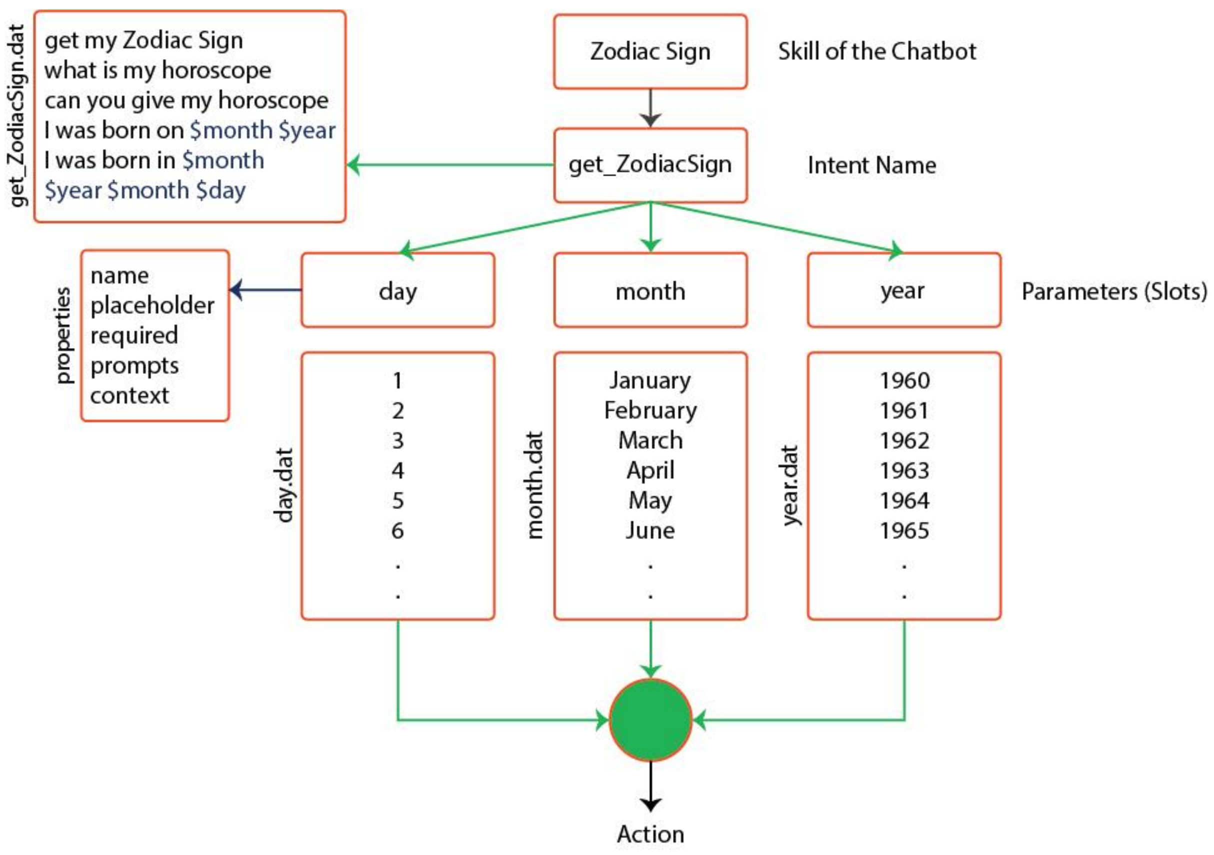
This file contains the JSON which holds the parameters for each intent, and its corresponding values, such as placeholder, required, etc.



```
{

    "get_Zodiac_Sign": {
        "intentname": "get_Zodiac_Sign",
        "Parameters": [

            {
                "name": "day",
                "placeholder": "$day",
                "required": "True",
                "prompts": [
                    "Which day(date) were you born?","I was born on 6th July when were you born"
                ],
                "context": ""
            },
                {
                "name": "month",
                "placeholder": "$month",
                "required": "True",
                "prompts": [
                    "In Which month you were born?","My birth month is July, what is yours?"
                ],
                "context": ""
            },
```



Picture depicting the various configuration files pipeline

Class Objects

- **Contexts** (context.py) - Here contexts are some flags that we switched on with each of the parameters or these can be used in conditional if-else statements. Below are the functions:
    - activate_context( )
    - deactivate_context( )

- Classes which are inheriting the properties of Context
    - FirstGreeting( )
    - IntentComplete( )

- **Intents** (intent.py) - Every Intent has its name and particular action and parameters. Parameter class is used to hold the attributes of intent from the JSON file.

Conversation Flow:

The image below shows how a given user input affects the chatbot's conversational flow using different methods and returns the appropriate response.

Take the user input from the recursive while loop, **Eg: I was born on 10 January 1996 what is my Zodiac Sign.**

The Reply() function from the Session class takes the user input argument and returns the respective prompt.

The following are the four main variables that flow through the process:

1. The user given input: **user_input = I was born on 10 January 1996 what is my Zodiac Sign**
2. Context from (context.py): context.name registers either "FirstGreeting" or "IntentComplete" as input, **context = FirstGreeting**
3. Attributes: A dictionary which stores the respective filled slots information intially Empty. i.e., **attributes = {}**
4. Current_intent: The intent predicted by the model based on the user input, intially None. i.e., **current_intent = None**

input_processor() helps in identifying all the slots in the user utterance. Identify and map them to the parameters.

attributes = {'day': '10', 'month': 'January', 'year': '1996'}

clean_input = "i was born on $day $month $year what is my zodiac sign"

The model generates a score for intent identification based on stored sample intent utterances upon cleaned user data. Following that, a logical decision making based on contexts at intent identifier to switch to a new intent or remain at the same can be done. loadIntent() takes the path (Chatbot/params/params.cfg) and intent_name (get_ZodiacSign) as inputs and Return's object of Intent class (intent.py) with updated name; parameters; actions of current Intent's.

current_intent = object of get_ZodiacSign intent

check_required_params() goes through required parameters of an intent object and generates a reply for user to provide that missing parameter. The prompt will be None if all the parameters are satisfied

prompt = None; context = FirstGreeting

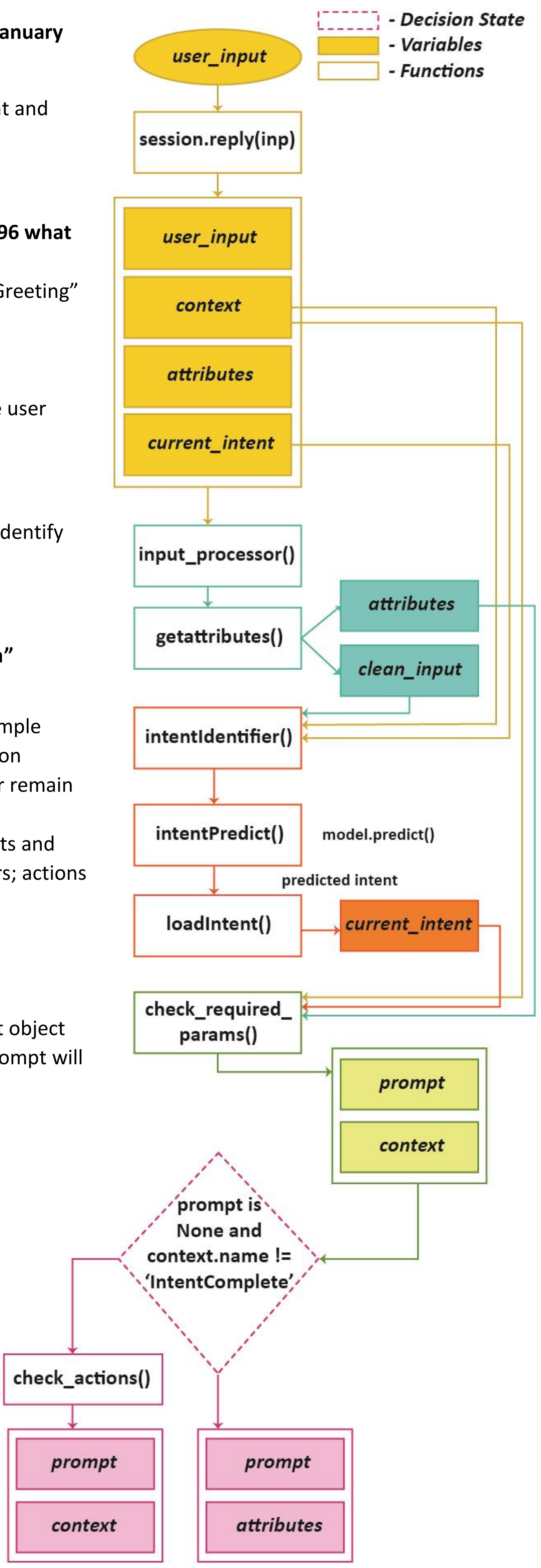
If prompt is None and the context is still FirstGreeting the check_actions() function assigns the context.name value to "IntentComplete" and performs the action for the intent as mentioned in the intent config file.

The check_action() function performs actions pertaining to current intent value and returns the appropriate prompt and context values

prompt = Your Zodiac sign is Capricorn

context = IntentComplete

attributes = {'day': '10', 'month': 'January', 'year': '1996'}


- Decision State
- Variables
- Functions
user_input
session.reply(inp)
user_input
context
attributes
current_intent
input_processor()
getattributes()
attributes
clean_input
intentIdentifier()
intentPredict()
model.predict()
predicted intent
loadIntent()
current_intent
check_required_params()
prompt
context
prompt is None and context.name != 'IntentComplete'
check_actions()
prompt
context
prompt
attributes