Final Project Information

106 Final Project

Due Midnight April 18, 2017 (70 Points)

For your final project, you may select one of the following options:

1. **DarkSky Net:** Build a chat bot that can also give weather information for any city
2. **Wheel of Python:** Build a "Wheel of Fortune" game with a computer player
3. **Propose your own project** of comparable difficulty. You must e-mail the instructors list with a project description **by April 2nd** if you plan to do this.

You must specify which project you plan to do **before midnight Sunday April 9th (in the last question of Problem Set 11)**.

Option 1: DarkSky Net

In this project, you will build a chat bot that can give weather information for any city:

```
> hello!
What can I call you?

> Steve
Nice to meet you Steve.

> Are you a robot?
How did you know I am a machine?

> What's the weather like in Detroit?
In Detroit, it is 41.55 and Mostly Cloudy

> How hot will it get in Ann Arbor this week?
In Ann Arbor it will reach 52.51

> Is it going to rain in Paris today?
It probably will not rain in Paris

> What's the weather like in FAKE CITY?
Is FAKE CITY a city?

> exit
```

In this interaction, our bot already (through pre-loaded AIML; see below) knows how to respond to:

- "hello!"
- "Steve"
- "Are you a robot?"

You must in the responses for:

- "What's the weather like in Detroit?"
- "How hot will it get in Ann Arbor this week?"
- "Is it going to rain in Paris today?"
- "What's the weather like in FAKE CITY?"
- "exit"

Requirements

README:

Your submission must include a file called "README.txt" with your name and instructions for the graders on how to run your file (including which file to run). Your ready should also include a few example interactions.

Query Types:

Your program must support the following query examples (wildcards in bold; you should be able to accept any city name):

- What's the weather like in **Ann Arbor**?
- Is it going to rain in **Ypsilanti** this week?
- How hot will it get in **Detroit** this week?
- How cold will it get in **Flint** this week?
- Is it going to rain in **East Lansing** this week?
- How hot will it get in **Grand Rapids** this week?
- How cold will it get in **Kalamazoo** this week?

Caching:

Your program **must support caching** requests from every API it uses. When you submit your project, we should be able to run it with cached cities without needing to connect to the internet.

Loading AIML Files:

You must load all of the AIML files in the `aiml_data` directory (on Canvas).

Accepting User Input:

Your program should use a `while` loop to indefinitely accept and respond to user input **until** the user types `"exit"`, after which your program should stop running.

Computing Rain Probability:

When the user asks "is it going to rain in XXX this week", your code should compute a rain probability by taking:

1-((probability it will NOT rain on day 1) * (probability it will NOT rain on day 2) * ... * (probability it will NOT rain on day 7))

If the resulting probability is **below 0.1**, your chatbot should respond with:

```
It almost definitely will not rain in (city)
```

If the resulting probability is **between 0.1 and 0.5**, your chatbot should respond with:

```
It probably will not rain in (city)
```

If the resulting probability is **between 0.5 and 0.9**, your chatbot should respond with:

```
It probably will rain in (city)
```

If the resulting probability is **above 0.9**, your chatbot should respond with:

```
It will almost definitely rain in (city)
```

Error Handling:

When the Google Geocoding API fails (typically, because the user did not enter a valid city name) your chat bot should respond:

```
"Is (city) a city?"
```

When the Dark Sky API fails, your chat bot should respond:

```
"Sorry, I don't know"
```

Getting Started

This project uses a version of the AIML (AI Markup Language: https://github.com/creatorrr/pyAIML (https://github.com/creatorrr/pyAIML)) that we modified. **Do not use the "standard" version of AIML for this project. Use our modified implementation (on Canvas)**. AIML encodes "stimulus/response" pairs for chat bots ("when the user says X, respond with Y"). AIML files contain collections of these stimulus/response pairs. You will not need to modify or know anything about AIML besides this.

The starter code contains:

- `aiml/`: code for parsing AIML files (*do not modify*)
- `aiml_data/`: built-in responses for user queries (*do not modify*)
- `chatbot.py`: starter code for you to modify
- `README.txt`: Instructions to allow us to run and grade your submission

"Learning" AIML Files

To load our modified version of the AIML parser, download the entire project starter code from Canvas. Then, in `chatbot.py`, type:

```
import aiml
```

to load the AIML parser.

Within the `aiml` package, you will use the `Kernel` class to learn how to respond to user queries. You can create a new instance with:

```
kernel = aiml.Kernel()
```

To load an AIML file into your kernel, call: `kernel.learn('aiml_data/std-hello.aiml')` (the starter code uses `os.path.join`)

You should use `os.listdir` to load every AIML file in the `aiml_data` directory.

Adding Custom Commands

Our modified version of AIML adds a new method to the `Kernel` class: `.addPattern()`

`.addPattern()` accepts two arguments: - A **pattern string** to match against - A **function** to call that returns a string that the chat bot will respond with.

The pattern string is a regular string with "named wildcards". For example:

```
"i live in {city}, {state}"
```

will match:

- `"I live in Ann Arbor, Michigan"`
  - city: `"Ann Arbor"`
  - state: `"Michigan"`
- `I live in FAKE CITY, SUPER FAKE STATE`
  - city: `"FAKE CITY"`
  - state: `"SUPER FAKE STATE"`
  - (our pattern matching algorithm isn't "smart", but the Geocoding API will realize that this isn't a real place)

The function you provide **must accept arguments that match the wildcards in the pattern string** (in the above example, `city` and `state`). This function should return a string to respond with:

```
def myExampleResponse(city, state):
    return '{}, eh? Do you like it in {}?'.format(state, city)


kernel.addPattern('i live in {city}, {state}', myExampleResponse)
```

...now the chat bot should respond:

```
> I live in Ann Arbor, Michigan
Michigan, eh? Do you like it in Ann Arbor?
```

Responding to User Queries

To get the chatbot's response to a given user query `q`, use `.respond()`:

```
# Print's the kernel's response to user query q
print(kernel.respond(q))
```

Getting Weather Data:

In order to get weather data, you will use two REST APIs:

- **The Google Geocoding API** to get a latitude and longitude for a given city
  - Documentation: https://developers.google.com/maps/documentation/geocoding/intro (https://developers.google.com/maps/documentation/geocoding/intro)
  - Example:
    - City name: "Ann Arbor"
    - Result:
      - latitude: 42.2808256,
      - longitude: -83.7430378
  - Note: requires an API key
- **The Dark Sky API** to get the weather for a given latitude and longitude
  - Documentation: https://darksky.net/dev/ (https://darksky.net/dev/)
  - Note: requires an API key