Project 1: Eliza

Due: 20 September 2019

The modern-day chatbot has a pretty long and venerable history; the earliest chatbot was written almost fifty years ago, and was called ELIZA. for instance, a user that typed "I like cheese" getting a response such as "What do you like about cheese?"—but only because a user that typed *any* sentence that starts with "I like" would get a response of that form. It was a pretty simple program, doing all its "thinking" using basic pattern matching.

The kinds of patterns that might be matched include:

- Words at the beginning of the line, such as "I like", where the response uses the *rest* of the line (as in the example above).
- Words anywhere in the line, that yield a stock response about that word (e.g. a line including the word "mother" might give a response "Tell me about your mother.")
- Patterns that match in multiple places in the line, and adapt the response to be grammatical (e.g. "I xxx my yyy") might give ("Why do you xxx your yyy?")[1]
- When nothing else matches, pulls a response randomly from a set of standard default responses.

In this project, you'll implement a version of Eliza.

Checkpoint

For the checkpoint (due next Monday at 4pm), you should have written a program that reads input until the input ends, and for each line of input prints a response (any response).

This should be pretty easy, but it's my way of checking that you can deal with the programming environment, running the compiler, using the handin script, and so on.

[1]Ok if the response is not grammatical in *all* cases; solving the general version of this is a hard research problem!

Final version

A full-credit final version will include logic implementing at least one pattern-and-response for each of the bullet points at the top of this handout, and will convince me through its documentation and test cases that it has done so. Since I'm not going to be able to spend a ton of time with your program (and in fact may not read it at all except to verify the high-point rubric items, and definitely won't do your debugging for you), your documentation will need to tell me what your program does and convince me that it does what you claim!

Credit is on the following scale:

0: Doesn't run, or immediately crashes when it's run.

10: Runs, does no more than specified for the checkpoint.

20: Makes at least two different responses (of any form and for any reason).

30: Responds according to pattern rules of at least two of the kinds listed on the front of this handout.

40: Responds according to pattern rules of at least three of the kinds listed on the front of this handout.

50: Responds according to pattern rules of all four of the kinds listed on the front of this handout.

+5: At least at the 30 level (two kinds of pattern rules) plus uses well-named functions to effectively break down the problem and perform most/all of the pattern matching and replacing.

+5: At least at the 30 level (two kinds of pattern rules) plus has at least a couple test cases, runnable with `pytest-3`, to verify that the correct pattern replacement is being made

+5: Previous item plus complete test cases for a full implementation *even if* the test cases don't pass due to the implementation not being complete.

Again, to get the points you have to not just tell me but *show* me that you've done those things. Document your work and tell me what to look for!

Handing in

Hand it in as `proj1` using the handin script.