Assignment 5

2/9/17

In this assignment, you will implement an Eliza program like that described in assignment 4, but you will use separate .py files to modularize the code.

# Build 1

Modularize the code in Assignment 4 so that:

* The controlling code is in eliza300\_5\_run\_1. The application starts from this file.
* The function *get\_complaint\_type()*, together with the data that it needs, are in *eliza300\_5\_functions\_1*. The “data that it needs” is in the form of lists as before.

# *read\_complaint\_data*()

This part will prepare for Build 2.

Thus far, the key words and complaint types are part of the code—in the form of lists *complaint\_type* and *key\_words*. This is not particularly desirable because whenever new kinds of complaints and key words arise, a programmer must modify the source code. It would be far better to enter these in a text file and have the program read them at runtime. Here are the specifications for the function *read\_complaint\_data()* that you are required to implement (precede the function definition with the two lines shown):

complaint\_types = []  
key\_words = []  
  
  
**def** read\_complaint\_data():  
 *'''  
 Intent: Get complaint\_types and key\_words from local ElizaData.txt  
  
 Precondition =========  
  
 ElizaData.txt is a local file consisting of paragraphs of the form  
  
 On first line: 'Key Words for '<phrase describing a complaint category>  
 On second line: <words, separated by blanks, that may occur within a  
 description of the corresponding category>  
  
 Example of ElizaData.txt:  
  
 Key Words for Depression  
 depress sad  
  
 Key Words for Human Relations  
 conflict argument mistreat  
  
 Postconditions =========  
  
 (1) complaint\_types = list of the phrases in ElizaData.txt describing all  
 complaint categories  
 {2) key\_words = list of lists of words in ElizaData.txt that may occur  
 within phrases that describe the corresponding complaint category  
  
 '''*

Place the following code immediately after the definition of *read\_complaint\_data()*:

read\_complaint\_data() *# need to execute this here*print(**"Priting complaint\_types and key\_words from ...runtime\_data..."**)  
print(complaint\_types)  
print(key\_words)

Below is an example, showing an *ElizaData.txt* file, followed by the output.

Key Words for Depression

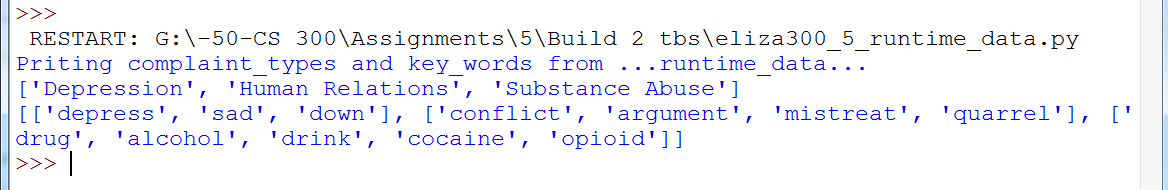
depress sad down

Key Words for Human Relations

conflict argument mistreat quarrel

Key Words for Substance Abuse

drug alcohol drink cocaine opioid

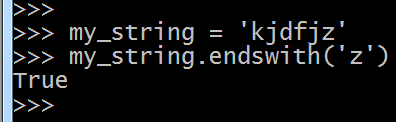


Hints:

* Since read\_complaint\_data() changes the value of variables defined outside it, you have to tell that to Python within the function definition as follows:

global complaint\_types, key\_words

* The function *readline()* returns the next line. The returned string ends with ‘\n’ unless it’s the last line. Use this knowledge to read all the lines with a *while* loop.
* Note the function *endswith*():



* Here are the first three of the four objectives that I used to organize the program:

# --(data\_source): data\_source represents 'ElizaData.txt' (local file) within the program

# AND data\_source (which is like a cursor) is at the beginning of the first unread line

# --(Postconditions for all read): Post(1) and Post(2) are valid for the data read so far

# --(line\_read): line\_read = contents of most recently read line from data\_source

You can fulfill all three of these objectives together (rather than one at a time) with a few lines of code. To fulfill the “line\_read” (last of the three) objective, I assigned '\n' to line\_read—in other words, without actually reading from the file yet.

* Your *while* loop starts with a *readline*() and then has to restore the above three objectives each time around since you want them to remain true. I used the following:

*while line\_read.endswith('\n'):*

# Build 2

Now modularize the entire program further by actually using *read\_complaint\_data*(), as follows:

* The controlling code is in *eliza300\_5\_run*. The application starts from this file as before.
* The function *get\_complaint\_type()* is in eliza300\_5\_functions as before
* The variables *complaint\_types* and *key\_words* are defined in *eliza300\_5\_runtime\_data* and collected there when read as before, and
* The function *read\_complaint\_data()*, which obtains *complaint\_types* and *key\_words*, is in the file *eliza300\_5\_runtime\_data* as before.
* Be sure that the following code follows directly after the definition of *read\_complaint\_data()*:

read\_complaint\_data() *# need to execute this here*(i.e., remove the print statement that followed this previously—as required in Build 1—you no longer need it). The line above will actually do the reading of data.

Instead of giving values to complaint\_types and key\_words in *eliza300\_5\_run* as you did in previous assignments, place the following line at the beginning of eliza300\_5\_functions:

**from** eliza300\_5\_runtime\_data\_3 **import** complaint\_types, key\_words

This will automatically execute *read\_complaint\_data()* in *eliza300\_5\_runtime\_*data, and *complaint\_types* and *key\_words* will be given data from the file as you programmed in Part 2.

***'''***

***Postconditions***

***1 (Welcome): A welcome message is on the console***

***2 (user\_complaint): user\_complaint is the user's response in reply to "Please state your complaint:"***

***3 (observed\_complaint\_types): observed\_complaint\_types = get\_complaint\_type(user\_complaint)***

***4 (Remedies displayed): the remedies in advice\_per\_type corresponding to***

***observed\_complaint\_types are on the monitor, one line for each.***

***Example: the user entered “I’ve been saddened by world conflicts,” and the***

***following is on the console after execution:***

***Get out more.***

***Take up a hobby that you love.***

***Don't expect too much of people.***

***Don't take offence easily.***

***'''***