Assignment 4

2/2/17

In this assignment, you will implement an Eliza program like that described in previous assignments, but you will use a function to make the code more comprehensible. Implementation notes are at the end of the various Build requirements.

# Build 1

First, we will build a function that takes the user’s “complaint” input as its parameter, and returns whether or not it suggests depression. This is based on whether the complaint contains one of a list of key words. (For example, “I am saddened” contains the word “sad”.) The function returns a set which is either {“Depression”} or {} (empty). Here are the specifications of the function *get\_complaint\_type*():

*# A list of key words for depression*key\_words = [**'depress'**, **'sad'**, **'down'**]

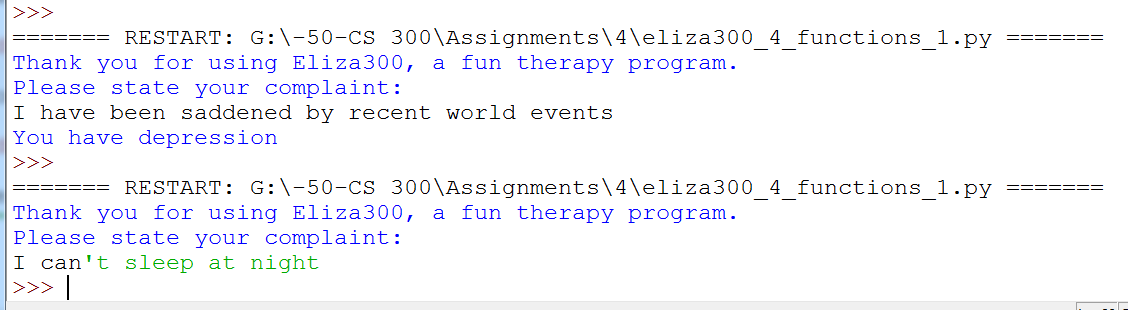
**def** get\_complaint\_type(a\_user\_complaint):  
 *'''  
 Precondition: a\_user\_complaint is a string*

*Postcondition:  
 EITHER a\_user\_complaint contains one of key\_words  
 AND observed\_complaint\_type is the set consisting of "Depression"  
 OR observed\_complaint\_type is the empty set  
  
 Returns: observed\_complaint\_type   
  
 Example: user entered “I’ve been saddened by world conflicts”   
 and {"Depression"} was returned.  
 '''*

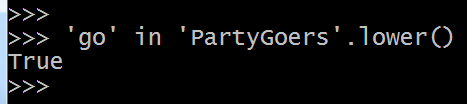
Use the following code following the definition of *get\_complaint\_type*() to complete the exercise:

*'''  
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\_type): observed\_complaint\_type = get\_complaint\_type(user\_complaint)  
2 (Advice displayed): EITHER "You have depression" OR nothing is displayed according to observed\_complaint\_type  
  
'''  
  
# (Welcome): Postcondition 1*print(**"Thank you for using Eliza300, a fun therapy program."**)  
  
*# (user\_complaint): Postcondition 2*print(**"Please state your complaint:"**)  
user\_complaint = input()  
  
*# (observed\_complaint\_type): Postcondition 3*observed\_complaint\_type = get\_complaint\_type(user\_complaint)  
  
*# (Advice displayed): Postcondition 4***if** len(observed\_complaint\_type) > 0:  
 print(**"You have depression"**)

Here is example I/O:



You may find the following useful:



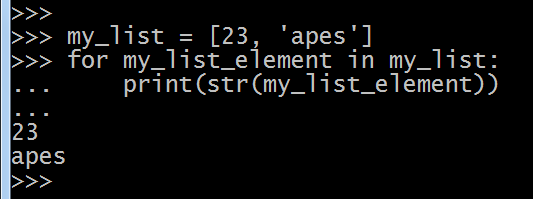
* To make an empty set, use *set*(), as in

*my\_set = set()*

* Add elements to sets with *add*() as in

*my\_set.add(my\_element)*

* *for* is a versatile, forgiving operator in Python. For example, you can use this kind of looping:



# Build 2

Now we’ll make *get\_complaint\_type*() more capable. It will return the corresponding complaint types (that it knows about). For example, if the user enters “I’ve been saddened by world conflicts,” the function returns the set consisting of “Depression” and “Human Relations.” It’s possible that the function returns the empty set. Here are the specifications:

*# A list of types of emotional complaints and corresponding key words for each*complaint\_type = [**'Depression'**, **'Human Relations'**, **'Substance Abuse'**]  
key\_words = [[**'depress'**, **'sad'**, **'down'**],  
 [**'conflict'**, **'argument'**, **'mistreat'**, **'quarrel'**],  
 [**'drug'**, **'alcohol'**, **'drink'**, **'cocaine'**, **'opioid'**, **'heroin'**]]  
  
  
**def** get\_complaint\_type(a\_user\_complaint):  
 *'''  
 Precondition:*

*1. a\_user\_complaint is a string*

*2. complaint\_type is a list of strings*

*3. key\_words is a list of lists of strings*

*3. complaint\_type and key\_words are the same length*

*Returns: observed\_complaint\_type, which consis*ts of the indices in  
 complaint\_type that correspond to key\_words elements partly in a\_user\_complaint.

Example: if the user enters “I’ve been saddened by world conflicts”,

the function returns the set consisting of 0 and 1 because “I’ve been saddened …”

contains “sad” and “conflict”.

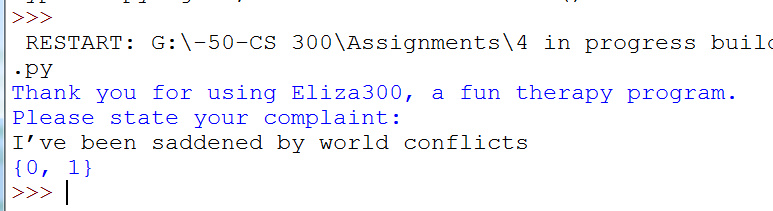
'''

The code following the function has the following specifications:

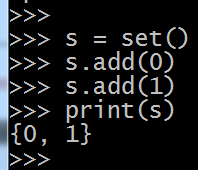
*'''  
Postconditions  
1 (Welcome): A welcome message is on the console   
2 (user\_complaint): user\_complaint is the user's response in reply to "Please state your complaint:"  
3 (observed\_complaint\_types): observed\_complaint\_types = get\_complaint\_type(user\_complaint)  
4 (Indices displayed): observed\_complaint\_types is on the monitor*

*'''*

Here is a typical session with Build 2:



Notes on Build 2: You may benefit from the following:



* The exercise now uses a list of lists, such as [[‘aaa’, ‘bb’],[‘aaa’, ‘b’],[‘cc’, ‘ddddd’]]
* The function returns: *observed\_complaint\_type*, which consists of those indices in *complaint\_type* that correspond to *key\_words* partly in *a\_user\_complaint*.
* Sets do not recognize order. The sets {33, 44, 55} and {44, 55, 33} are considered the same.

# Build 3

In the this last build, the specifications for the function *get\_complaint\_type*() are unchanged but the rest of the code (that follows the definition) has the following specifications:

advice\_per\_type = [  
 [**'Get out more.'**, **'Take up a hobby that you love.'**],  
 [**"Don't expect too much of people."**, **"Don't take offence easily."**],  
 [**'Get counseling.'**, **"Don't delay action on counseling."**]]  
  
**'''  
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.  
'''**

Here is typical I/O:

