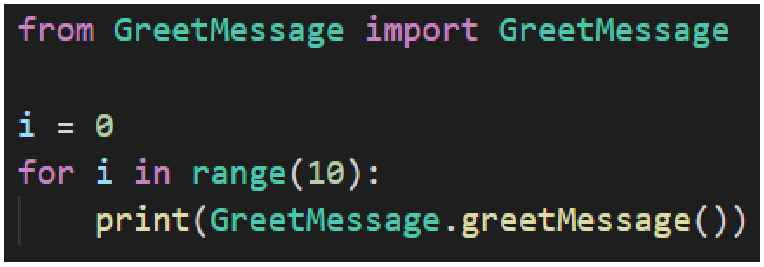
Names of methods or variables may not match.

Testing was done in the following order:

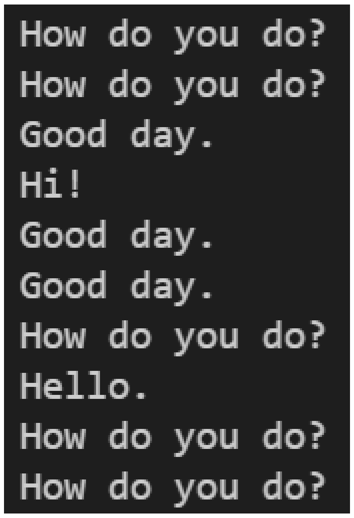
- GreetMessage:
  - Test: Create a new class. Import the class GreetMessage, and call the function, greet_message(), 10 times. The function is working correctly if (1) the greeting messages show up correctly, and (2) randomly. Rinse and repeat 3 times. If all three runs work, then the function has passed the test..
    - Code used to test:



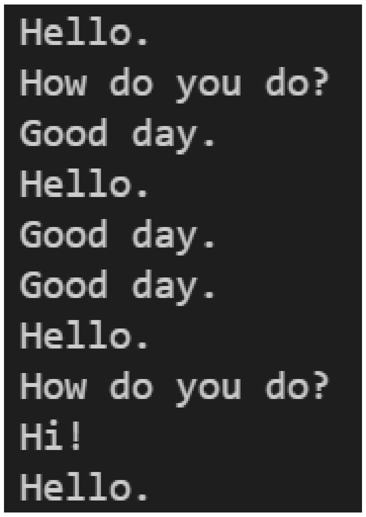
```
from GreetMessage import GreetMessage

i = 0
for i in range(10):
    print(GreetMessage.greetMessage())
```

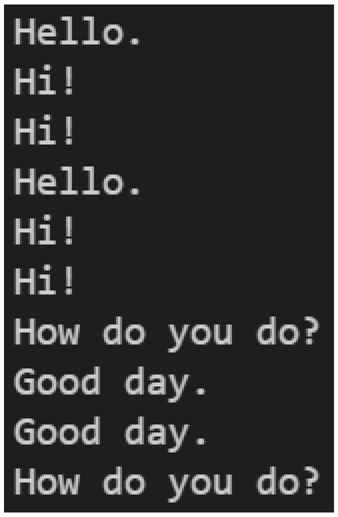
    - Outcome of the test:



```
How do you do?
How do you do?
Good day.
Hi!
Good day.
Good day.
How do you do?
Hello.
How do you do?
How do you do?
```



```
Hello.
How do you do?
Good day.
Hello.
Good day.
Good day.
Hello.
How do you do?
Hi!
Hello.
```

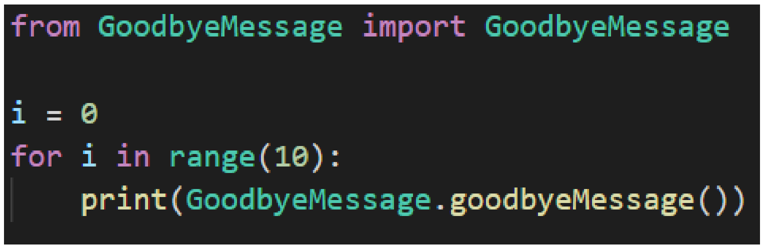


```
Hello.
Hi!
Hi!
Hello.
Hi!
Hi!
How do you do?
Good day.
Good day.
How do you do?
```

    - Test success: Passed

---

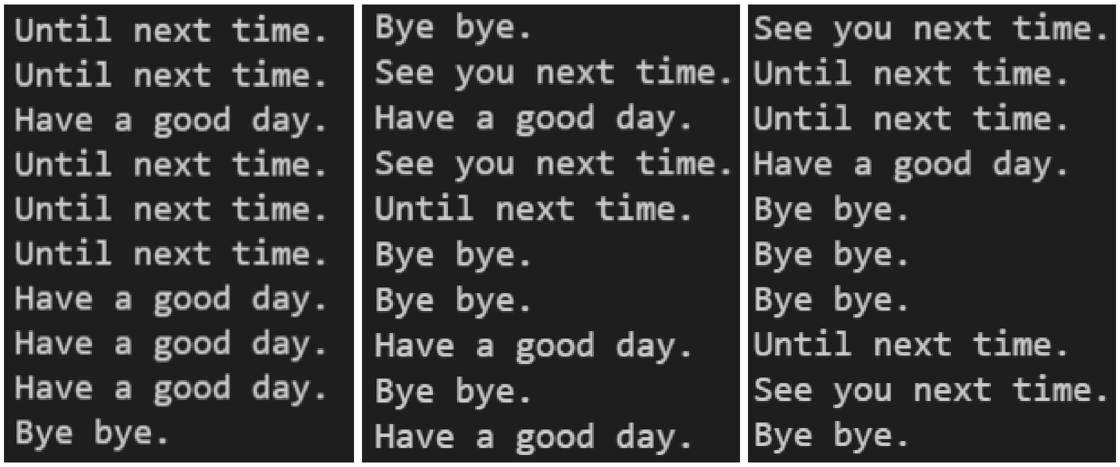
- GoodbyeMessage:
  - Test: Create a new class. Import the class GoodbyeMessage, and call the function, goodbye_message(), 10 times. The function is working correctly if (1) the goodbye messages show up correctly, and (2) randomly. Rinse and repeat 3 times. If all three runs work, then the function has passed the test.
    - Code used to test:



```
from GoodbyeMessage import GoodbyeMessage

i = 0
for i in range(10):
    print(GoodbyeMessage.goodbyeMessage())
```

- Outcome of the test:



```
Until next time.
Until next time.
Have a good day.
Until next time.
Until next time.
Until next time.
Have a good day.
Have a good day.
Have a good day.
Bye bye.
```

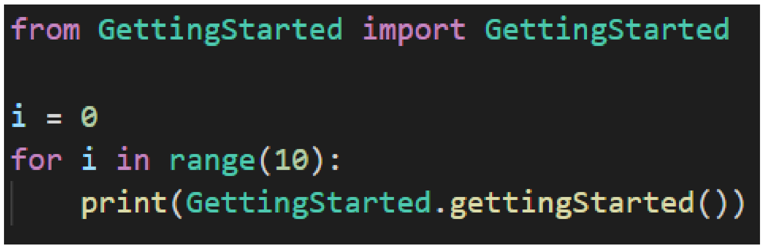
```
Bye bye.
See you next time.
Have a good day.
See you next time.
Until next time.
Bye bye.
Bye bye.
Have a good day.
Bye bye.
Have a good day.
```

```
See you next time.
Until next time.
Until next time.
Have a good day.
Bye bye.
Bye bye.
Bye bye.
Until next time.
See you next time.
Bye bye.
```

- Test success: Passed

---

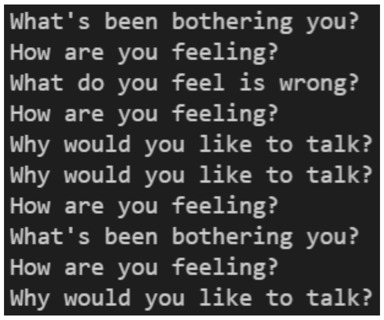
- GettingStarted:
  - Test: Create a new class. Import the class GettingStarted, and call the function, getting_started(), 10 times. The function is working correctly if (1) the getting started messages show up correctly, and (2) randomly. Rinse and repeat 3 times. If all three runs work, then the function has passed the test.
    - Code used to test:



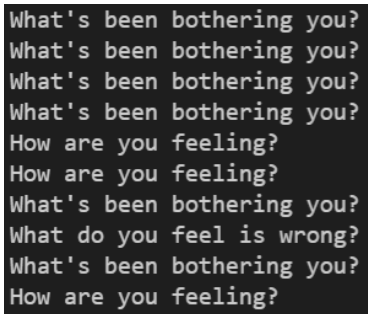
```
from GettingStarted import GettingStarted

i = 0
for i in range(10):
    print(GettingStarted.gettingStarted())
```

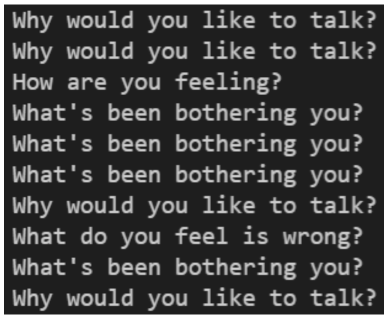
- Outcome of the test:



```
What's been bothering you?
How are you feeling?
What do you feel is wrong?
How are you feeling?
Why would you like to talk?
Why would you like to talk?
How are you feeling?
What's been bothering you?
How are you feeling?
Why would you like to talk?
```



```
What's been bothering you?
What's been bothering you?
What's been bothering you?
What's been bothering you?
How are you feeling?
How are you feeling?
What's been bothering you?
What do you feel is wrong?
What's been bothering you?
How are you feeling?
```

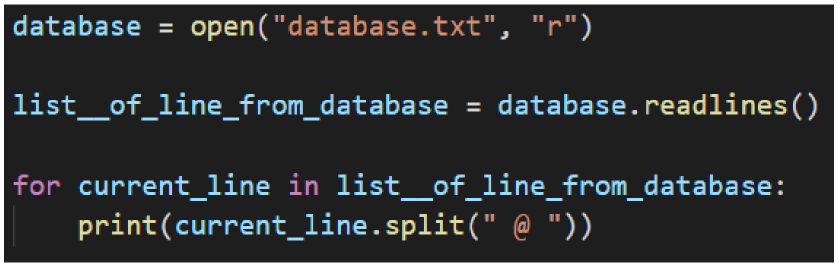


```
Why would you like to talk?
Why would you like to talk?
How are you feeling?
What's been bothering you?
What's been bothering you?
What's been bothering you?
Why would you like to talk?
What do you feel is wrong?
What's been bothering you?
Why would you like to talk?
```

- Test success: Passed

---

- Database:
  - Test: Create a new class. Import the class and in a loop that goes through each line, and storing, temporarily, said line into a variable, current_line. After storing the line in said current_line, use the method .split(" @ ") on current_line. Print said application, i.e., print(current_line.split(" @ ")). After running the loop, scan over the printed lines, and make sure (i) that each "prompt" and answer are in a size 2 list, (ii) the "prompt" comes before the "answer", (iii) there is no "@", (iv) the "prompt", in the list, does not have a space at the end, (v) the "answer", in the list, does not have a space at the beginning and end (vi) the "prompt" has at least 3 words. If all the sentences have passed the six metrics, then the Database has passed the test.
    - Code used to test:

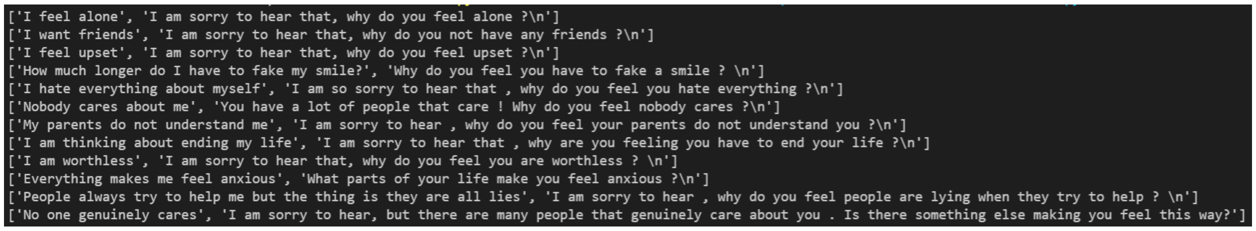


```
database = open("database.txt", "r")

list__of_line_from_database = database.readlines()

for current_line in list__of_line_from_database:
    print(current_line.split(" @ "))
```

    - Outcome of the test:



```
['I feel alone', 'I am sorry to hear that, why do you feel alone ?\n']
['I want friends', 'I am sorry to hear that, why do you not have any friends ?\n']
['I feel upset', 'I am sorry to hear that, why do you feel upset ?\n']
['How much longer do I have to fake my smile?', 'Why do you feel you have to fake a smile ? \n']
['I hate everything about myself', 'I am so sorry to hear that , why do you feel you hate everything ?\n']
['Nobody cares about me', 'You have a lot of people that care ! Why do you feel nobody cares ?\n']
['My parents do not understand me', 'I am sorry to hear , why do you feel your parents do not understand you ?\n']
['I am thinking about ending my life', 'I am sorry to hear that , why are you feeling you have to end your life ?\n']
['I am worthless', 'I am sorry to hear that, why do you feel you are worthless ? \n']
['Everything makes me feel anxious', 'What parts of your life make you feel anxious ?\n']
['People always try to help me but the thing is they are all lies', 'I am sorry to hear , why do you feel people are lying when they try to help ? \n']
['No one genuinely cares', 'I am sorry to hear, but there are many people that genuinely care about you . Is there something else making you feel this way?']
```

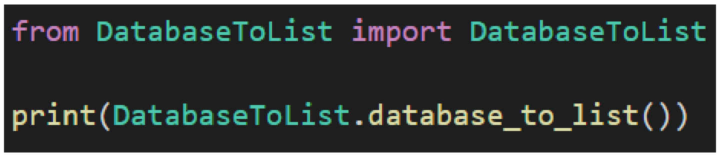
    - Test success: Passed

---

- DatabaseToList:
  - Test: Create a new class. Import the DatabaseToList, and call the function, database_to_list(). Make sure you have implemented class Database first. Because it will return a list, print said list. Scan

over the printed list, and make sure (i) that each "prompt" and "answer" pair are in a size 2 list, (ii) the "prompt" comes before the "answer", (iii) there is no "@", (iv) the "prompt", in the list, does not have a space at the end, (v) the "answer", in the list, does not have a space at the beginning, (vi) the "prompt" has at least 3 words, (vii) the printed list contains list(s) of size 2. If printed list has passed the seven metrics, then the DatabaseToList has passed the test.

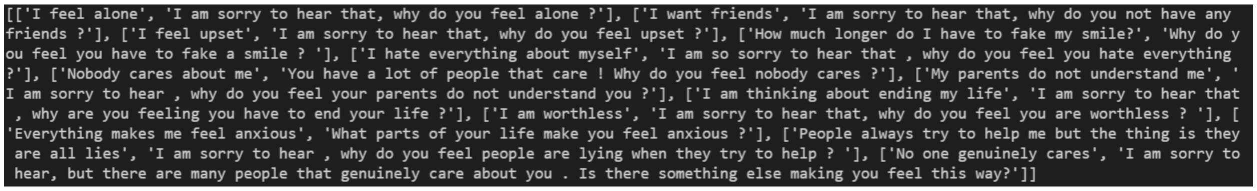
- Code used to test:



```
from DatabaseToList import DatabaseToList

print(DatabaseToList.database_to_list())
```

- Outcome of the test:

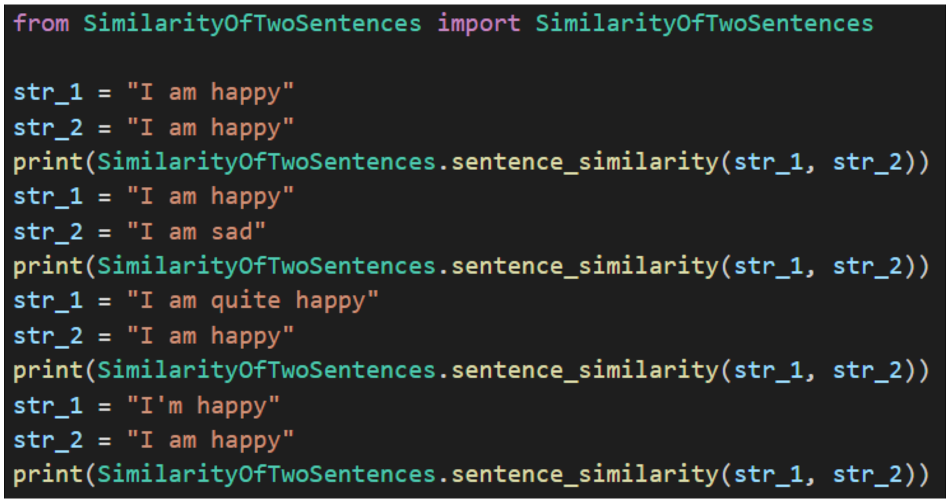


```
[['I feel alone', 'I am sorry to hear that, why do you feel alone ?'], ['I want friends', 'I am sorry to hear that, why do you not have any
friends ?'], ['I feel upset', 'I am sorry to hear that, why do you feel upset ?'], ['How much longer do I have to fake my smile?', 'Why do y
ou feel you have to fake a smile ? '], ['I hate everything about myself', 'I am so sorry to hear that , why do you feel you hate everything
?'], ['Nobody cares about me', 'You have a lot of people that care ! Why do you feel nobody cares ?'], ['My parents do not understand me', '
I am sorry to hear , why do you feel your parents do not understand you ?'], ['I am thinking about ending my life', 'I am sorry to hear that
 , why are you feeling you have to end your life ?'], ['I am worthless', 'I am sorry to hear that, why do you feel you are worthless ? '], [
'Everything makes me feel anxious', 'What parts of your life make you feel anxious ?'], ['People always try to help me but the thing is they
 are all lies', 'I am sorry to hear , why do you feel people are lying when they try to help ? '], ['No one genuinely cares', 'I am sorry to
 hear, but there are many people that genuinely care about you . Is there something else making you feel this way?']]
```

- Test success: Passed

---

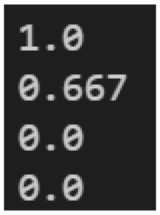
- SimilarityOfTwoSentences:
  - Test: Create a new class. Import the class SimilarityOfTwoSentences, and call the function, sentence_similarity(str_1, str_2); where the four cases are tested: (i) str_1 = "I am happy" and str_2 = "I am happy". Output expected: 1 (ii) str_1 = "I am happy" and str_2 = "I am sad" Output expected:  0.666… (iii) str_1 = "I am quite happy" and str_2 = "I am happy". Output expected: 0 (iv) str_1 = "I'm happy" and str_2 = "I am happy". Output expected:  0. If outputs come out correctly, then the class SimilarityOfTwoSentences has passed the test
    - Code used to test:



```
from SimilarityOfTwoSentences import SimilarityOfTwoSentences

str_1 = "I am happy"
str_2 = "I am happy"
print(SimilarityOfTwoSentences.sentence_similarity(str_1, str_2))
str_1 = "I am happy"
str_2 = "I am sad"
print(SimilarityOfTwoSentences.sentence_similarity(str_1, str_2))
str_1 = "I am quite happy"
str_2 = "I am happy"
print(SimilarityOfTwoSentences.sentence_similarity(str_1, str_2))
str_1 = "I'm happy"
str_2 = "I am happy"
print(SimilarityOfTwoSentences.sentence_similarity(str_1, str_2))
```

- Outcome of the test:

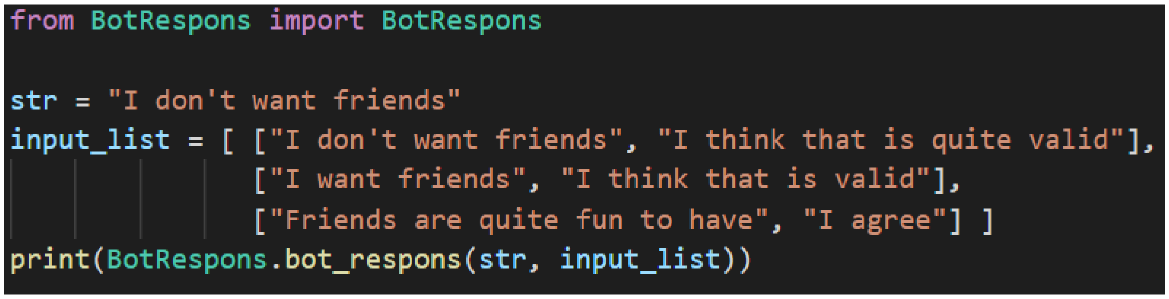


```
1.0
0.667
0.0
0.0
```

- Test success: Passed

---

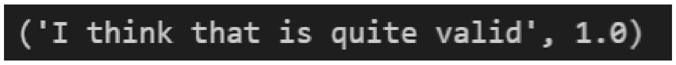
- BotsRespons:
  - Test: Create a new class. Import the class BotsRespons, and call the function, bot_respons (str, input_list); where a case is tested: str = "I don't want friends" input_list = [ ["I don't want friends", "I think that is quite valid"], ["I want friends", "I think that is valid"], ["Friends are quite fun to have", "I agree"] ] Output expected: "I think that is quite valid", and 1. If the output comes out as expected, then BotsRespons has passed the test
    - Code used to test:



```python
from BotRespons import BotRespons

str = "I don't want friends"
input_list = [ ["I don't want friends", "I think that is quite valid"],
               ["I want friends", "I think that is valid"],
               ["Friends are quite fun to have", "I agree"] ]
print(BotRespons.bot_respons(str, input_list))
```

- Outcome of the test:



```
('I think that is quite valid', 1.0)
```

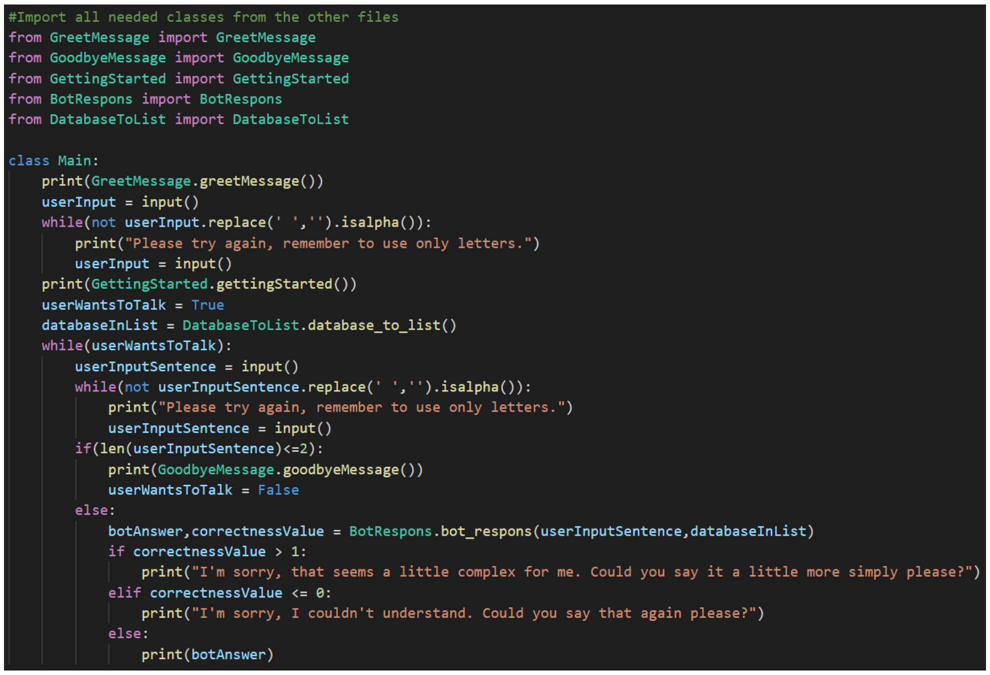
- Test success: Passed

---

- Main:
  - Test: First enter an input that does contain a char that is not a letter, expected output should be: an appropriate output informing the user that the input is invalid. Then, input something appropriate. Next, enter an input that does contain a char that is not a letter, expected output should be: print an appropriate output informing the user that the input is invalid. Next, input a sentence that is larger than any "prompt" in the database; the  expected output should be: a message informing the user, that the bot could not understand. Next, input a sentence that is at least 3 words long, but is smaller or equal to the largest "prompt" in the database. The expected output should be an answer that is appropriate. To determine an appropriate answer, apply the method bot_respons (str, list). After doing so, if the output is appropriate and matches the manually calculated answer,

then the case is cleared. Next, enter in a sentence that is less, in terms of length, than or equal to 2; expected output is an output from the method goodbye_message() and closers of the program. If the code passed all the above tests, then the code has passed testing.

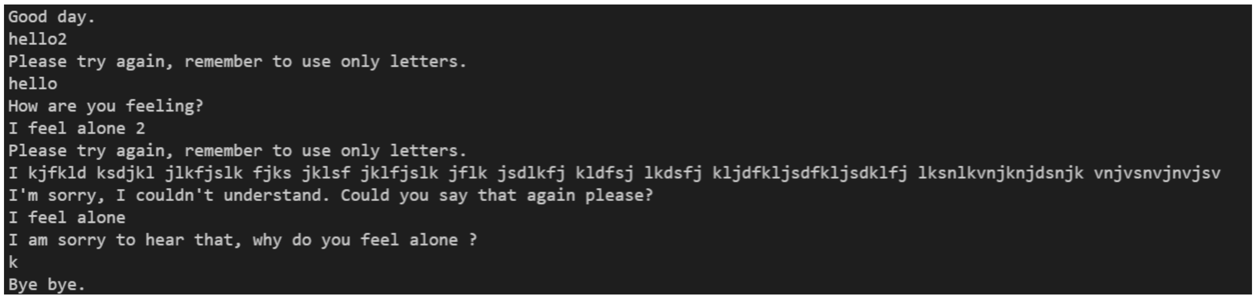
- Code used to test:



```
#Import all needed classes from the other files
from GreetMessage import GreetMessage
from GoodbyeMessage import GoodbyeMessage
from GettingStarted import GettingStarted
from BotRespons import BotRespons
from DatabaseToList import DatabaseToList

class Main:
    print(GreetMessage.greetMessage())
    userInput = input()
    while(not userInput.replace(' ','').isalpha()):
        print("Please try again, remember to use only letters.")
        userInput = input()
    print(GettingStarted.gettingStarted())
    userWantsToTalk = True
    databaseInList = DatabaseToList.database_to_list()
    while(userWantsToTalk):
        userInputSentence = input()
        while(not userInputSentence.replace(' ','').isalpha()):
            print("Please try again, remember to use only letters.")
            userInputSentence = input()
        if(len(userInputSentence)<=2):
            print(GoodbyeMessage.goodbyeMessage())
            userWantsToTalk = False
        else:
            botAnswer,correctnessValue = BotRespons.bot_respons(userInputSentence,databaseInList)
            if correctnessValue > 1:
                print("I'm sorry, that seems a little complex for me. Could you say it a little more simply please?")
            elif correctnessValue <= 0:
                print("I'm sorry, I couldn't understand. Could you say that again please?")
            else:
                print(botAnswer)
```

- Outcome of the test:



```
Good day.
hello2
Please try again, remember to use only letters.
hello
How are you feeling?
I feel alone 2
Please try again, remember to use only letters.
I kjfkld ksdjkl jlkfjslk fjks jklsf jklfjslk jflk jsdlkfj kldfsj lkdsfj kljdfkljsdfkljsdklfj lksnlkvnjknjdsnjk vnjvsnvjnvjsv
I'm sorry, I couldn't understand. Could you say that again please?
I feel alone
I am sorry to hear that, why do you feel alone ?
k
Bye bye.
```

- Test success: Passed