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:

```
Hello
Hi
Greetings
Hello
Hi
Greetings
Hello
Howdy
Greetings
Hello
```

```
Howdy
Howdy
Hi
Hello
Hello
Howdy
Hello
Howdy
Greetings
Hi
```

```
Hi
Howdy
Greetings
Greetings
Greetings
Greetings
Howdy
Howdy
Hello
Greetings
```

    - 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:

```
Goodbye
Goodbye
See you later
Goodbye
See you later
Have a good day!
Have a good day!
Have a good day!
Goodbye
Have a good day!
```

```
Have a good day!
See you later
Have a good day!
Bye bye.
Bye bye.
Goodbye
See you later
Have a good day!
Goodbye
Bye bye.
```

```
Have a good day!
Bye bye.
See you later
Goodbye
Have a good day!
Bye bye.
Have a good day!
Goodbye
Goodbye
Goodbye
```

- 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:

```
How are you feeling?
How are you feeling?
What do you feel is wrong?
How are you feeling?
What do you feel is wrong?
How are you feeling?
How are you feeling?
How can I help you?
What do you feel is wrong?
What is bothering you?
```

```
What is bothering you?
How are you feeling?
What do you feel is wrong?
What is bothering you?
What is bothering you?
How are you feeling?
What do you feel is wrong?
How are you feeling?
How can I help you?
How are you feeling?
```

```
How are you feeling?
What is bothering you?
What do you feel is wrong?
What do you feel is wrong?
What is bothering you?
What is bothering you?
What is bothering you?
How can I help you?
How are you feeling?
How can I help you?
```

- 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:

```python
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:

Due to the size of the database, it is impractical to show the output. However, the output is correct

    - 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:

Due to the size of the database, it is impractical to show the output. However, the output is correct

- 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: "Bot: I'm sorry, I cannot understand that sentence. Could you say it a little more simply please?". 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 goodbyeMessage() and closers of the program. If the code passed all the above tests, then the code has passed testing.
    - Code used to test:

```python
#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(f"Bot: {GreetMessage.greetMessage()}")
    userInput = input("User: ")
    while((not userInput.replace(' ','').isalpha()) or (len(userInput.split()) == (not 1)) ):
        print("Bot: Please try again, remember to use only letters.")
        userInput = input("User: ")
    print(f"Bot: {GettingStarted.gettingStarted()}")
    userWantsToTalk = True
    databaseInList = DatabaseToList.database_to_list()
    while(userWantsToTalk):
        userInputSentence = input("User: ")
        while((not userInputSentence.replace(' ','').isalpha()) or (len(userInputSentence) == 0) ):
            print("Bot: Please try again, remember to use only letters.")
            userInputSentence = input("User: ")
        if(len(userInputSentence.split())<=2):
            print(f"Bot: {GoodbyeMessage.goodbyeMessage()}")
            userWantsToTalk = False
        else:
            botAnswer,correctnessValue = BotRespons.bot_respons(userInputSentence,databaseInList)
            if correctnessValue > 1 or correctnessValue <= (1/3):
                print("Bot: I'm sorry, I cannot understand that sentence. Could you say it a little more simply please?")
            else:
                print(f"Bot: {botAnswer}")
            correctnessValue = 0
```

- Outcome of the test:

```
Bot: Hi
User: Hello2
Bot: Please try again, remember to use only letters.
User: Hello
Bot: How can I help you?
User: I feel alone 2
Bot: Please try again, remember to use only letters.
User: jdklajlkfdjskdfjsdklj kljdslk jf I jdklsdjfk jsdklfjsd klfjlskdj flksjdf lkfdjlksjdfkl jdlk jdslk jslkd jflkj kljslkdfj lkj lksjfkl js
lkf jlksdjklsdjf lkj lkfsdj lkfjdsflkjf kj klj dlkj ldsj klfdsj lksj ksj lksj jsd lkjdklfjkld jflk jdkl sjf
Bot: I'm sorry, I cannot understand that sentence. Could you say it a little more simply please?
User: I feel alone
Bot: I am sorry to hear that, why do you feel alone?
User: Goodbye
Bot: Goodbye
```

- Test success: Passed
