Homework #3 - Make a Fortune Teller Program

For this assignment, you will be writing a *Fortune Teller* class with the following:

- **A constructor (*__init__*) method**: The constructor will initialize a new ***Fortune_Teller*** object from the passed list of all possible answers.
  - Set ***fortunes_list*** to the passed list of possible answers.
  - Set ***questions_list*** to an empty list. This will hold all the questions that have been asked.
  - Set ***fortunes_history_list*** to an empty list. This will hold the indices of all of the answers that have been generated.

- ***__str__*** **method**: It should return a string with all of the answers in ***fortunes_list*** separated by commas, For example : "No, Yes, Hard to say."

```
Testing the __str__ method
['Yes', 'No', 'Ask again', 'Maybe', 'Not clear']
```

- ***get_fortune*** **method**: Returns a random answer from the ***fortunes_list.*** It adds the index to the end of the ***fortunes_history_list***. It returns a string containing the answer at that index (not the index).

- ***question_check*** **method**: Checks if the current question is already in the ***questions_list*** and if so returns *"I've already answered that question"*, otherwise it adds the current question to the ***questions_list*** and returns the answer from ***get_fortune***.

```
Asking the Question: Should I study today?
Not clear
```

```
Asking the Question: Should I study today? (again)
I've already answered that question
```

- ***print_questions_history*** **method**: Prints the content of the ***fortunes_history_list*** with the answer index in **[ ]** and each question and answer on a separate line. It does not return anything. If there are no items in ***fortunes_history_list*** it should print **"None yet".**

```
Printing the history
[0] Will I pass this semester? - Yes
[1] Should I study today? - No
[0] Is SI 206 the best class ever? - Yes
```

```
Printing the history when no answers have been generated yet
None yet
```

- ***main()* function**: Loops until the user types “quit” getting a question from the user, calls the ***question_check*** method, and prints the question and response from ***question_check*** as "*question* - *answer*" as shown below.

```
Ask a question or type quit: Is the answer 42?
Is the answer 42? - Yes
Ask a question or type quit: quit
```

- Example Output From HW3.py

Sample output from the main method:

```
Ask a question or type quit: Will it rain today?
Will it rain today? - No
Ask a question or type quit: Should I have pizza today?
Should I have pizza today? - Not clear
Ask a question or type quit: Will I get all As this semester?
Will I get all As this semester? - No
Ask a question or type quit: Should I have tacos today?
Should I have tacos today? - Yes
Ask a question or type quit: Will I get all As this semester?
Will I get all As this semester? - I've already answered that question
Ask a question or type quit: quit
```

Sample output from the test method:

```
Testing the __str__ method
['Yes', 'No', 'Ask again', 'Maybe', 'Not clear']

Printing the history when no answers have been generated yet
None yet

Asking the Question: Will I pass this semester?
Not clear

Asking the Question: Should I study today?
Maybe

Asking the Question: Should I study today? (again)
I've already answered that question

Asking the Question: Is SI 206 the best class ever?
Yes

Printing the history
[4] Will I pass this semester? - Not clear
[3] Should I study today? - Maybe
[0] Is SI 206 the best class ever? - Yes

Testing most_frequent method with 200 responses
Yes: 36
No: 36
Ask again: 43
Maybe: 38
Not clear: 47
The most frequent answer after 200 was Not clear
```

NOTE: Your output will not look *exactly* like this because we are using *random* and can't predict what it will return.

NOTE 2: You are welcome to replace the answers we have provided in the *main function* with your favorite responses

Grading Rubric - Total of 60 points

- 5 points - the ***__init__*** method sets the object's ***fortunes_list*** correctly to the passed ***fortunes_list*** and sets both the object's ***fortunes_history_list*** and ***questions_list*** to an empty list
- 5 points - the ***__str__*** method returns a string with all answers in ***fortunes_list*** separated by commas : "Yes, No, It depends"
- 5 points - the ***check_fortune*** method returns *"I've already answered that question"* if the question has already been asked
- 10 points - the ***check_fortune*** method calls the ***get_fortune*** method and returns the answer when the user asks a new question and adds the passed question to the ***questions_list***.
- 10 points - the ***get_fortune*** method returns a random answer and saves the index of the answer at the end of the ***fortunes_history_list***
- 5 points - the **print_questions_history** function prints **"None Yet"** when there are no items in ***fortunes_history_list***.
- 10 points - ***print_questions_history*** prints "[index] Question - Answer" for each of the questions in the ***questions_list*** and ***fortunes_history_list*** in order and on a separate line.
- 10 points - the ***main()*** function loops until the user enters "quit" and each time asks the users for a question and prints the **"*question - response*"**.

This grading rubric shows how you will gain points, but not all the ways you could lose points.

Extra Credit - 6 points

Implement the following method: Create the ***most_common*** method. It needs to find the most frequent answer after getting a fortune ***n*** times. It takes a number as an input: ***n***, Ex: 200. It resets the ***fortunes_history_list*** instance variable to an empty list, executes ***get_fortune n*** times, prints how many times each answer occurred, and prints the most frequently occurring answer. Choose any one of the top most common answers if there is a tie.

Extra Credit Example Output:

```
Testing most_frequent method with 200 responses
Yes: 36
No: 36
Ask again: 43
Maybe: 38
Not clear: 47
The most frequent answer after 200 was Not clear
```