## Homework #3 - Make a Magic 8 Ball

### For this assignment, you will be writing a *Magic 8 Ball* class with the following:

* **A constructor (\_\_init\_\_) method**: The constructor will initialize a new *Magic\_8 object* from the passed list of all possible answers.
  + Set ***answer\_list*** to the passed list of possible answers.
  + Set ***question\_list*** to an empty list. This will hold all the questions that have been asked.
  + Set **answer\_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 **answer\_list** separated by commas, For example : "No, Yes, Better not tell you now."
* ***check\_question*** method: Checks if the current question is already in the ***question list*** and if so returns *“I already answered that question!”*, otherwise it adds the current question to the ***question\_list*** and returns the answer from ***shake\_bal****l.*
* ***shake\_ball* method**: Returns a random answer from the **answer\_list.** It randomly picks an index from 0 to the number of possible answers minus one *(hint: use the random module)*. It adds the index to the end of the **answer\_history\_list**. It returns a string containing the answer at that index (not the index).
* ***print\_history* method**: Prints the content of the **answer\_history\_list** with the index number in [] and each question and answer on a separate line. It does not return anything. If there are no items in **answer\_history\_list** it should print **"None yet".**
* ***main()* function**: Loops until the user types “quit” getting a question from the user, calls the ***check\_question*** method, and prints the question and response from **check\_question.**
* Example Output From HW3.py

**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 ***answer\_list*** correctly to the passed ***answer list***
* 5 points - the ***\_\_init\_\_*** method sets both the object's ***answer\_history\_list*** and ***question\_list*** to an empty list
* 10 points - the ***\_\_str\_\_***method returns a string with all answers in ***answer\_list*** separated by commas : "Yes, No, It depends”
* 5 points - the ***check\_answer***method returns *“I already answered that question!”* if the question has already been asked
* 10 points - the ***check\_answer***method calls the ***shake\_ball*** method and returns the answer when the user asks a new question and adds the passed question to the ***question\_list***.
* 5 points - the ***shake\_ball***method returns a random answer and saves the index of the answer at the end of the ***answer\_history\_list***
* 10 points - ***print\_history*** prints "[index] Question - Answer" for each of the questions in the **question\_list** and **answer\_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 ***generate\_n\_responses*** method. It takes a number as an input: n, Ex: 200. It generates random answers n times by calling ***shake\_ball*** and returns the index and length of the longest consecutive run for an answer index. A run is a repetition of the same number consecutively in a list.

Ex: If 10 random answers were [1,5,6,3,2,4,1,**4,4,4**] then three 4's is the longest run

Hence the function should return “**longest run was length of 3 for index 4”**

Extra Credit Example Output:

