PROJECT

Magic 8-Ball

The Magic 8-Ball is a popular toy developed in the 1950s for fortune-telling or advice seeking.

Write a **magic8.py** Python program that can answer any “Yes” or “No” question with a different fortune each time it executes.

We’ll be using the following 9 possible answers for our Magic 8-Ball:

- `Yes - definitely.`
- `It is decidedly so.`
- `Without a doubt.`
- `Reply hazy, try again.`
- `Ask again later.`
- `Better not tell you now.`
- `My sources say no.`
- `Outlook not so good.`
- `Very doubtful.`

The output of the program will have the following format:

```
[Name] asks: [Question]
Magic 8-Ball’s answer: [Answer]
```

For example:

```
Joe asks: Is this real life?
Magic 8-Ball's answer: Better not tell you now
```

Let’s get started!

Tasks

15/15 Complete

Mark the tasks as complete by checking them off

Setting up

1.

In **magic8.py**, declare a variable `name` and assign it to the name of the person who will be asking the Magic 8-Ball.

Hint
Assign the variable `name` to a string value. In Python, a string must be wrapped in a pair of quotes.

```
name = "Joe"
```

2.
Next, declare a variable `question`, and assign it to a “Yes” or “No” question you’d like to ask the Magic 8-Ball.
Hint
Assign the variable to a string, which should be wrapped in quotes:

```
question = "Will I win the lottery?"
```

3.
We want to store the answer of the Magic 8-Ball in another variable, which we’ll call `answer`. For now, assign this variable to an empty string.
Hint
Create a variable `answer` and assign it to an empty string:

```
answer = ""
```

Generating a random number

4.
In order for the answer to be different each time we run the program, we will utilize randomly generated values.

Note: This will be something new! But don’t worry, we will try to explain this topic thoroughly and also supply the code.

In Python, we can use the `.randint()` function from the `random` module to generate a random number from a range.

But first, let’s import this module so we can use its functions. Add this line of code to the top of **magic8.py**:

```
import random
```

Hint
Make sure to do this before all the previous code that you had written.
5.
Next, we’ll create a variable to store the randomly generated value. Declare a variable called `random_number`, and assign it to the function call:

```
random.randint(1, 9)
```

which will generate a random number between `1` (inclusive) and `9` (inclusive).

Next, add a `print()` statement that outputs the value of `random_number`, and run the program several times to ensure random values are being generated as expected.

Once you're sure this is working as we expected, feel free to comment out this `print()` statement.
Hint
The variable should be assigned to the function call with the set range of values.

```
random_number = random.randint(1, 9)
```

To test that it's correctly generating random values each time, we can use `print()` to output the value that was generated.

```
print(random_number)
```

Control Flow

6.

Now that we've declared all the variables needed, it's time to implement the core logic of our program!

For this section, we'll be utilizing control flow using an `if`/`elif`/`else` statement to assign different answers for each randomly generated value.

First, write an `if` statement where if the `random_number` is equal to `1`, `answer` is assigned to the phrase "Yes - definitely."
Hint

```
if random_number == 1:
  answer = "Yes - definitely."
```

7.

Next, write an `elif` statement after the `if` statement where if the `random_number` is equal to `2`, `answer` is assigned to the phrase "It is decidedly so".

Then, continue writing `elif` statements for each of the remaining phrases for the values `3` to `9`.

Recall that the 9 possible answers of the Magic 8-Ball are:

1. `Yes - definitely.`
2. `It is decidedly so.`
3. `Without a doubt.`
4. `Reply hazy, try again.`
5. `Ask again later.`

6. `Better not tell you now.`
7. `My sources say no.`
8. `Outlook not so good.`
9. `Very doubtful.`

Hint

Add the `elif` statements for each outcome in the following way:

```
if random_number == 1:
  answer = "Yes - definitely."
elif random_number == 2:
  answer = "It is decidedly so."
elif random_number == 3:
  answer = "Without a doubt."
# ...
```

8.

Following the `if`/`elif` statements, add an `else` statement that will set `answer` to the string “Error”, if the number was accidentally assigned a value outside of our range.

Hint

```
if random_number == 1:
  answer = "Yes - definitely."
elif random_number == 2:
  answer = "It is decidedly so."
# ...
else:
  answer = "Error"
```

Seeing the result

9.

Now, let’s see our program in action! Write a `print()` statement to output the asker’s `name` and their `question`, which should be in the following format:

```
[Name] asks: [Question]
```

For example, when we run the program, the output should look something like:

```
Joe asks: Will I win the lottery?
```

Hint

```
print(name + " asks: " + question)
```

10.

Add a second `print()` statement that will output the Magic 8-Ball’s `answer` in the following format:

```
Magic 8-Ball's answer: [answer]
```

For example, when running the program it should look something like:

```
Magic 8-Ball's answer: My sources say no
```

Hint

```
print("Magic 8-Ball's answer: " + answer)
```

11.

Great job! You've successfully utilized your knowledge of conditionals and previous fundamental Python concepts to create a program that generates different fortunes.

Run your program several times to see that it's working as expected.

Hint

There should be a different output every time you press Run.

Optional Challenges

12.

If you're up for some more challenges, try implementing the following features to your program.

So far, the Magic 8-Ball provides 9 possible fortunes. Try to add a few more possible answers to the program.

To do this, you will need to increase the range of randomly generated numbers and add additional `elif` statements for each new answer.

Hint

For example, if we add one more answer to the Magic 8-Ball, there will be a total of `10` outcomes, we'll need to expand the range of randomly generated values to `10` instead of `9`.

```
random_number = random.randint(1, 10)
```

Then we'll need to add another `elif` statement, which will assign the variable `answer` to the new possible phrase, like the following:

```
elif random_number == 10:
  answer = "Signs point to yes"
```

13.

What if the asker does not provide a name, such that the value of `name` is an empty string? If the `name` string is empty, the output of the program looks like the following:

```
 asks: Will I win the lottery?
Magic 8 Ball's answer: Outlook not so good
```

As you can see, the formatting of the output can use some improvement when there is no name provided.

We can address this by printing out just the question, such that it looks like the following:

```
Question: Will I win the lottery?
Magic 8-Ball's answer: Outlook not so good
```

You can implement this by creating an `if`/`else` statement such that:

- If the name is an empty string, it will only print the question.
- Else, the player’s name and question are both printed.

Hint

We can write an `if` statement that checks if the `name` is set to an empty string. In Python, you can do this a few ways, including:

```
if name == ""
```

or,

```
if len(name) == 0
```

Within the `if` statement, when there is no name provided, we can just print the `question`:

```
if name == "":
  print("Question: " + question)
```

Otherwise, we can print the `name` and the `question` within an `else` statement block:

```
else:
  print(name + " asks: " + question)
```

14.

What if the `question` string is empty? If the user does not provide any question, then the Magic 8-Ball cannot provide a fortune, otherwise, the fabric of reality is at risk!

To ensure that the fabric of reality is safe, we can create an `if`/`else` statement where:

- If the question is an empty string, print out a message to the user.
- Else, print the name and question, with the Magic 8-Ball’s answer.

Hint

If the `question` is an empty string, we want to let the user know, and prevent the other `print()` statements from being run.

To check if the `question` is set to an empty string, we can use one of the following conditions for the `if` statement:

```
if question == "":
  # do something
```

or

```
if len(question) == 0:
  # do something
```

Using either of these conditions to check that the `question` is empty, we will write a `print()` statement within the `if` statement block that will output a message, like so:

```
if question == "":
  print("The Magic 8-Ball cannot provide a fortune unless you ask it something.")
```

Else, if a question is provided, we can run the other `print()` statements:

```
else:
  print(name + " asks: " + question)
  print("Magic 8-Ball's answer: " + answer)
```

Solution

15.

Don’t forget to check off all the tasks before moving on.

Sample solutions:

- **magic8.py**

P.S. If you make something cool, share it with us!
Hint
Tag @Codecademy on Twitter or make a Pull Request to the Learn Python GitHub repo for a chance to be featured in the course!

magic8.py

```
import random
name = "Andres"
question = "Will I win the lottery in 2023?"
answer = ""
random_number = random.randint(1, 10)
#print(random_number)
if random_number == 1:
  answer = "Yes - definitely."
elif random_number == 2:
  answer = "It is decidedly so."
elif random_number == 3:
  answer = "Without a doubt."
elif random_number == 4:
  answer = "Reply hazy, try again."
elif random_number == 5:
  answer = "Ask again later."
elif random_number == 6:
  answer = "Better not tell you now."
elif random_number == 7:
  answer = "My sources say no."
elif random_number == 8:
  answer = "Outlook not so good."
elif random_number == 9:
  answer = "Very doubtful."
elif random_number == 10:
  answer = "Signs point to yes."
else:
  answer = "Error"

if question == "":
  print("The Magic 8-Ball cannot provide a fortune unless you ask it something.")
else:
  if name == "":
    print("Question: " + question)
  else:
    print(name + " asks: " + question)
  print("Magic 8-Ball's answer: " + answer)
```