Day Two Challenges

You have a few options for challenges today. Be sure to note the difficulty levels, but feel free to hop around to any one you want.

1. Add any feature you want to your bot!

If you can think of something really cool you want your bot to do, try to make it happen!

2. *(fairly difficult)* Add a word definition command to your bot

Users should be able to send a message of "!define <word>" and the bot should define the word they enter.

- You should define a new function that will get the definition of a word, and call it if the message from the user starts with "!define"
- There are many ways to fill out the body of your function:
  - **A:** *(easiest)* Use a lot of if statements, checking if the message equals a word and responding with the definition
  - **B:** *(more difficult)* Create a Python Dictionary, where each word is a key and each definition is a value
  - **C:** *(extremely difficult)* Use http requests and web scraping to get the definition from a website

3. *(more difficult)* Add some Magic 8 Ball functionality to your bot

Users should be able to send a message of "!ask <question>" and the bot should answer the question with a random answer.

- You should define a new function that will return the random answer, and call it if the message from the user starts with "!ask"
- Then, you will need to do a few things:
  - Create a List of the possible responses
  - Import the random library using `import random` at the top of your file
  - Call the `random.choice` function and pass in your list to extract one option
  - Return the randomly selected option

4. *(very difficult)* Update the "!repeat" command to repeat the message a certain number of times

Your bot should read a command like "!repeat <number> <message>" and repeat the given message the given number of times.

Examples

- "!repeat 4 hello" -> "hello hello hello hello"
- "!repeat 0 whatever and ever" -> "" (no message)
- "!repeat 2 something else" -> "something else something else"

Steps

- You will need to extract the number from the message
- You will need to extract the actual message from the message
- You will need to use a loop to create the response

5. *(less difficult)* Continue working on challenges from Day One

There were a **lot** of challenges on Day One. If you did not finish all of them, you can work on those.

6. *(difficult)* Turtle Graphics

Python comes with a cute little package called turtle. With this challenge you are invited to explore the universe of Turtle Graphics. Imagine you have a robotic turtle sitting in the middle of a large sheet of white paper at coordinates (0, 0). You can give your turtle commands to move forward, turn left and turn right. The command `turtle.forward(25)` will move your turtle (on-screen!) `25` pixels in the direction it is facing. Give it the command `turtle.right(45)`, and it rotates in-place `45` degrees clockwise.

Now imagine that a pen is attached to the tail of the turtle and you can tell your robotic turtle to put the pen up or put the pen down. When the pen is down, a line will be drawn as the turtle moves. If the pen is up and the turtle moves, nothing is drawn on the paper. Another cool thing is you can tell your robot turtle what color of pen to use.

Oh what fun we can have!!!

To get started, let's try something simple:

1. Setup VS Code for a new project, (note: name your file anything other than `turtle.py`)
2. In your new file, first you will need to import the Python turtle package

```
import turtle
```

3. Next you will need to create a robotic turtle and some paper on the screen
   Lets call our new turtle bob

```
bob = turtle.Turtle()
paper = turtle.Screen()
```

4. We can give a title to our screen paper with

```
paper.title("Hy-Camp Turtle Graphics")
```

5. When the turtle is first created, by default, the pen is down touching the paper. Let's change the color of the pen to "red"

```
bob.pencolor("red")
```

6. We can control the look of the turtle image on the screen, use

```
bob.shape("turtle")
```

7. When the turtle is first created, by default, the turtle is facing East. Let's tell the turtle to face north by turning to the left 90 degrees.

```
bob.left(90)
```

8. Now we can tell our turtle to move forward 100 and a red line will be drawn.

```
bob.forward(100)
```

9. Add these two lines to the end of your code.

```
paper.exitonclick()
paper.mainloop()
```

These last two lines will allow your paper and turtle to be displayed until you click the mouse on the window. Try your code.

Now copy and insert the commands to turn left and move forward three more times immediately after the first pair, for a total 8 consecutive commands. Run your program again to see what happens.

Also note, you can hide or show your turtle with the commands
`bob.hideturtle()` or `bob.showturtle()`

Instead of eight commands, put only one left turn and one forward command in a loop that executes 4 times.

Multiple squares:

Move the loop code that draws a square and place it in a function called drawSquare, and then call the function.
Also, instead of having the square always be a 100, add a function parameter to specify the square size. Then call you drawSquare function with different sizes. You can also set the pen color to different colors before the function is called.

Spin your squares:

Suppose you would like to have your turtle draw squares but at different angles around a complete circle. For instance 12 squares each offset at an angle of 30 degrees adding up to 360 degrees (a full circle)
Hint: Loop 12 times, in each loop, tell turtle to draw a square and then have the turtle turn 30 degrees.

Ask for input:

Before your program creates the turtle and paper, you could ask the user for the size of the square, the number of the squares, the degree of rotation and the color of the pen. Use the specified values when you command your robotic turtle.

Google "Python Turtle Graphic Images" to see what others have done with their robotic turtles.

Also try out these commands to draw a star

```
import turtle

# call our turtle bob
bob = turtle.Turtle()
paper = turtle.Screen()

paper.title("A Hy-Camp Turtle")

paper.bgcolor("black") # make the paper black

bob.shape("turtle")

bob.color("red", "yellow")

# Offset the turtle to center star on the paper
bob.right(180)
bob.penup()
bob.forward(200)
bob.pendown()
bob.right(180)

# draw star shape
bob.begin_fill()

while True:
    bob.forward(400)
    bob.left(170)
    if abs(bob.pos()+(200, 0)) < 1:
        break

bob.end_fill()

bob.hideturtle()

paper.exitonclick()
paper.mainloop()
```

To see a full list of turtle commands visit https://docs.python.org/3/library/turtle.html (section 24.1.2)