RGB-HEX Converter

In this project, we'll use Bitwise operators to build a calculator that can convert RGB values to Hexadecimal (hex) values, and vice-versa.

We'll add three methods to the project:

- A method to convert RGB to Hex
- A method to convert Hex to RGB
- A method that starts the prompt cycle

The program should do the following:

1. Prompt the user for the type of conversion they want
2. Ask the user to input the RGB or Hex value
3. Use Bitwise operators and shifting in order to convert the value
4. Print the converted value to the user

It's useful to know some background on RGB and hex values, so we recommend reading the resources we linked to.

Note: As with professional software development, you should be saving your code very often. As you code, make sure you click the "Save" button below to save your code/changes. Otherwise, you run the risk of losing all your code.

RGB-HEX Converter

1. First, we'll create the RGB to Hex method.

   On line 1, add a method called `rgb_hex()`.

2. We should expect users to make accidental errors as they input RGB values. Let's create a constant error message that will store the message we will display to users when they make an accidental error.

   Inside the method, create a variable called `invalid_msg` and set it equal to an appropriate error message. The message should say something about invalid values being entered.

3. To convert RGB values, we'll have to prompt the user to enter values for red (R), green (G), and blue (B). Let's start by asking them enter a value for red.

   On the next line, still inside of the method, prompt the user to enter a red value using `raw_input`. Wrap their input using `int()`. Finally, set the result equal to a variable called `red`.

4. We know that an RGB value contains three separate values, with each value representing an amount of red (R), green (G), and blue (B). Acceptable values are within the range of 0 to 255. Let's add error checking to make sure acceptable values are entered.

   On the next line, add an `if` statement that checks if the value of `red` is less than 0, `or` greater than 255.

5. Inside of the `if` statement, we should inform the user that an invalid value has been entered.

   Inside of the statement, print the error message that you created earlier. Then, on the next line, return.

6. Now, just as in Step 3, prompt the user to enter a value for green (G). Wrap the input using `int()` and set the result equal to a variable called `green`.

7. Next, add error checking to make sure that the value the user enters for the color green is valid. Inside of the `if` statement, print the error message and then return on the next line.

8. We'll have to follow the same process to prompt the user to enter a value for the color blue (B).

   Prompt the user, wrap their input with `int()`, and set it equal to a variable called `blue`.

☐ 9. Again, add error checking make sure the value entered for the color blue is valid. Inside of the `if` statement, print the error message and then return.

☐ 10. Now it's time to use Bitwise operators to build the rest of our method. We recommend becoming more familiar with hexadecimal numbers first so that you can understand what the Bitwise operators in the method do.

☐ 11. On the next line, outside of any `if` statement, create a variable called `val`. Set it equal to the sum of shifting `red` to left by 16 bits, shifting `green` to left by 8 bits, and blue.

☐ 12. Finally, call the `hex()` method and pass `value` in as the argument. Use list slicing to print out everything except the first two characters of that string. Also, call the `upper()` method on the result.

See if you can use string formatting to complete all of this in one line of code.

Click here to learn more about how `hex()` works.

☐ 13. Great! This method will convert an RGB value to a hex value.

Now, add a method called `hex_rgb()`. This is the method we'll use to convert the opposite way (from Hex to RGB).

☐ 14. Inside of the method, prompt the user to enter a hexadecimal value using `raw_input()`. Set the result equal to a variable called `hex_val`.

☐ 15. Let's add some error checking that will make sure the user inputs a valid hexadecimal value. Valid hexadecimal values are six characters long, so let's check for that first.

Add an `if` statement that checks if the length of `hex_val` is not equal to six.

☐ 16. Inside of the `if` statement, print a message to the user indicating that an invalid value was entered. On the next line, return.

☐ 17. Otherwise, we should accept the hex value as an integer.

Add a corresponding `else` block that sets `hex_val` equal to calling `int()` with the arguments `hex_val` and `16`.

☐ 18. Next, outside of the `else` block, but still within the method, create a variable called `two_hex_digits` and set it equal to 2 raised to the power of 8.

☐ 19. Next, we'll start calculating the RGB values.

Create a variable called `blue` and set it equal to `hex_val` modulo `two_hex_digits`.

☐ 20. Next, shift `hex_val` to the right by 8 bits.

☐ 21. Now, just as you did in Step 19, create a variable called `green` and set it equal to `hex_val` modulo `two_hex_digits`.

☐ 22. On the next line, shift `hex_val` once more to the right by 8 bits.

☐ 23. Finally, calculate the red value by creating a variable called `red` and setting it equal to `hex_val` modulo `two_hex_digits`.

☐ 24. Let's write the last line of code for this method.

On the next line, use string formatting to print out the RGB values on one line. There should be no space between the individual values.

25. Fantastic! The method you just wrote will convert a hexadecimal value to an RGB value.

    Let's add the last method that will run our program. Create a new method called `convert()`.

26. Inside the method, add a `while` loop with the Boolean `True` as the condition.

27. On the next line, inside of the `while` loop, prompt the user for input with the following message: `Enter 1 to convert RGB to HEX. Enter 2 to convert HEX to RGB. Enter X to Exit:`.

    Set the result equal to a variable called `option`.

28. Now let's handle all the cases of user input.

    Start an `if` statement that checks if `option` is equal to `'1'` (as a string).

29. Inside of the `if` statement, print the message `RGB to Hex...` to the user.

    On the following line, call the `rgb_hex()` method.

30. Add a corresponding `elif` block that checks if the option is `'2'`. If it is, print `Hex to RGB...` first. Then, on the next line, call the `hex_rgb()` method.

31. Add another `elif` statement that checks if the option is `'X'` or `'x'`. If it is, exit the loop with the `break` keyword.

32. Finally, finish the `if` statement by adding an `else` block. This part of the statement will handle any other input from the user. Inside of the `else` block, print `Error.`.

33. You're nearly done - great job! The next step is to actually call the method that will run our program.

    As the final line of your code (outside of any method), call the `convert()` method.

34. Great! Let's test out the converter.

    In the terminal, type the following and hit "Enter" on your keyboard:

    ```
    python rgb2hex.py
    ```

    Feel free to expand the functionality. Happy coding!