Name:_____ **Section:**_____

Objectives

- Compose a function which uses a for loop.
- Understand how a computer represents colour.

Representing Colour in Hexadecimal

Let's briefly consider the hexadecimal (base-16) representation, a common way of writing binary (base-2) numbers without filling a page with 1s and 0s. In hexadecimal, we count on the basis of 16 rather than 10; thus, the second column no longer represents how many 10s we have, but how many 16s:

Decimal	Hexadecimal	Decimal	Hexadecimal
0	0	11	B
1	1	12	C
2	2	13	D
3	3	14	E
4	4	15	F
5	5	16	10
6	6	17	11
7	7	100	64
8	8	256	100
9	9	1000	3E8
10	A	1024	400

	R	G	B (forest Green)
Decimal	45	94	22
Hex	2D	5E	16

Computers typically store colors as a collectionv of three eight-bit values, representing the red, green, and blue (RGB) components on the scale 0 to 255 ($256 = 2^8$).

As 256 is conveniently divisible by 16, many platforms such as web browsers represent colors as a string with six hexadecimal digits. Consider these colors, for example:

#000000	(black, no colour present)	#888888	(a middle grey tone)
#FFFFFF	(white, all colours present)	#FF00FF	(magenta, red + blue pixels set)
#FF0000	(red, only red pixels set)	#FFFF00	(yellow, red + green pixels set)
#00FF00	(green, only green pixels set)	#DAA520	(goldenrod, a mixed yellow)
#0000FF	(blue, only blue pixels set)	#07F2CB	(turquoise, mostly green)

The tricky part of all this is to convert a decimal value into a two-digit hexadecimal value. While Python does have a built-in function hex, we will carry out this calculation manually for this class.

First, you have to divide the decimal number by 16 to get the first digit in decimal. Second, take the remainder as the second digit. For instance, to convert 16 to hexadecimal:

16 ÷ 16 = 1 rem 0 → therefore #10 is the hexadecimal equivalent (by the table above)

A couple more examples:

134 ÷ 16 = 8 rem 6 → therefore #86 is the hexadecimal equivalent of decimal 134

207 ÷ 16 = 12 rem 15 → therefore #CF is the hexadecimal equivalent of decimal 207

In this lab exercise, you will write a function rgb2hex to convert a value from an RGB tuple of decimals to a hexadecimal string. *This exercise will be completed on paper and handed in, with the rest of lab04 taking place in Jupyter and sent to Instructor by Email (zliao@zju.edu.cn)*.

1) Compose a function definition for rgb2hex which accepts a list rgb_color. (We trust that the input is valid.)
   (a) Initialize a blank string named hex.
   (b) Create a list which has the hexadecimal equivalents of each decimal value at the right index; *i.e.*, dec2hex[5] = '5', dec2hex[11] = 'B'.
2) Complete the for statement to loop over each value in the tuple rgb_color. Calculate the decimal equivalent of the first digit (divide by 16) as hex1.
3) Calculate the decimal equivalent of the second digit (remainder of division by 16) as hex2.
4) Convert the decimal value of hex1 to its hexadecimal equivalent using the dec2hex list and append the result to hex_str.
5) Convert the decimal value of hex2 to its hexadecimal equivalent using the dec2hex list and append the result to hex_str.

Return the result.

```
def rgb2hex(①_____):

    hex_str = ''

    dec2hex = ['0','1','2','3','4','5','6','7','8','9', 'A','B','C','D','E','F']

    for ②_____

        hex1 = value // 16

        hex2 = ③_____

        ④_____

        ⑤_____

    return hex_str


# example test of rgb2hex
assert rgb2hex( [ 63, 32, 255 ] ) == '3F20FF'
```