```
#Color Theme Maker
import tkinter
from tkinter import BOTH, IntVar, DISABLED, filedialog

#Define window
root = tkinter.Tk()
root.title('Color Theme Maker')
root.iconbitmap('color_wheel.ico')
root.geometry('450x500')
root.resizable(0,0)

#Define fonts and colors
#NONE:  Using system defaults

#Define functions
def get_red(slider_value):
    """Turn current slider value for red into a hex value and update color.
    The scale value is passed automatically when the scale is moved calling the get_red function."""
    global red_value

    #Turn the slider value into an int and hex value. Strip leading chars so only two remain
    red_value = hex(int(slider_value))
    red_value = red_value.lstrip("0x")

    #If hex value is single digit, lead with a 0 such that d becomes 0d
    while len(red_value) < 2:
        red_value = "0" + str(red_value)

    update_color()


def get_green(slider_value):
    """Turn current slider value for green into a hex value and update color.
    The scale value is passed automatically when the scale is moved calling the get_green function."""
    global green_value

    #Turn the slider value into an int and hex value. Strip leading chars so only two remain
    green_value = hex(int(slider_value))
    green_value = green_value.lstrip("0x")

    #If hex value is single digit, lead with a 0 such that d becomes 0d
    while len(green_value) < 2:
        green_value = "0" + str(green_value)

    update_color()


def get_blue(slider_value):
    """Turn current slider value for blue into a hex value and update color.
    The scale value is passed automatically when the scale is moved calling the get_blue function."""
    global blue_value

    #Turn the slider value into an int and hex value. Strip leading chars so only two remain
    blue_value = hex(int(slider_value))
    blue_value = blue_value.lstrip("0x")

    #If hex value is single digit, lead with a 0 such that d becomes 0d
    while len(blue_value) < 2:
        blue_value = "0" + str(blue_value)

    update_color()
```

```
62
63
64  def update_color():
65      """UPdate the current color box based on the slider values.  Display tuple and hex
        values of the current color"""
66      #Make the color box smaller than the original due to ipadx and ipday on the
        original color box
67      color_box = tkinter.Label(input_frame, bg="#" + red_value + green_value +
        blue_value, height=6, width=15)
68      color_box.grid(row=1, column=3, columnspan=2, padx=35, pady=10)
69
70      #Display the tuple and hex value for the given color
71      color_tuple.config(text='(' + str(red_slider.get()) + '),' + '(' +
        str(green_slider.get()) + '),' + '(' + str(blue_slider.get()) + ')')
72      color_hex.config(text="#" + red_value + green_value + blue_value)
73
74
75  def set_color(r,g,b):
76      """Set a given color"""
77      red_slider.set(r)
78      green_slider.set(g)
79      blue_slider.set(b)
80
81
82  def store_color():
83      """Store the current color tuple value and display color"""
84      global stored_colors
85
86      #Get the current value of each slider and append 0's to keep formatting
87      red = str(red_slider.get())
88      while len(red) < 3:
89          red = "0" + red
90
91      green = str(green_slider.get())
92      while len(green) < 3:
93          green = "0" + green
94
95      blue = str(blue_slider.get())
96      while len(blue) < 3:
97          blue = "0" + blue
98
99      #Keep a reference of the current color
100     stored_red = red_slider.get()
101     stored_green = green_slider.get()
102     stored_blue = blue_slider.get()
103
104     #Create new widgets for the stored color.
105     recall_button = tkinter.Button(output_frame, text="Recall Color",
        command=lambda:set_color(stored_red, stored_green, stored_blue))
106     new_color_tuple = tkinter.Label(output_frame, text='(' + red + '),' + '(' + green +
        '),' + '(' + blue + ')')
107     new_color_hex = tkinter.Label(output_frame, text='#' + red_value + green_value +
        blue_value)
108     new_color_black_box = tkinter.Label(output_frame, bg='black', width=3, height=1)
109     new_color_box = tkinter.Label(output_frame, bg="#"+ red_value + green_value +
        blue_value, width=3, height=1)
110
111     #Put new widgets on the screen
112     recall_button.grid(row=stored_color.get(), column=1, padx=20)
113     new_color_tuple.grid(row=stored_color.get(), column=2, padx=20)
114     new_color_hex.grid(row=stored_color.get(), column=3, padx=20)
115     new_color_black_box.grid(row=stored_color.get(), column=4, pady=2, ipadx=5, ipady=5)
116     new_color_box.grid(row=stored_color.get(), column=4)
117
118     #Update the dict stored_colors with the new color tuple and hex values
119     stored_colors[stored_color.get()] = [new_color_tuple.cget("text"),
        new_color_hex.cget("text")]
```

```
120
121         #Move the radio button stored colors_ to the next value if available
122         if stored_color.get() < 5:
123             stored_color.set(stored_color.get() + 1)
124
125
126     def save_colors():
127         """Output the chosen colors to a txt file."""
128         #Get the directory where the user would like to save
129         file_name = filedialog.asksaveasfilename(initialdir='./', title='Save Colors',
            filetypes=(('Text', '.txt'),('All Files', '*.*')))
130
131         #open the new file as write
132         with open(file_name, "w") as f:
133             f.write("Color Theme Maker Output\n")
134             for saved_entry in stored_colors.values():
135                 f.write(saved_entry[0] + "\n" + saved_entry[1] + "\n\n")
136
137
138     #Define Layout
139     input_frame = tkinter.LabelFrame(root, padx=5, pady=5)
140     output_frame = tkinter.LabelFrame(root, padx=5, pady=5)
141     input_frame.pack(fill=BOTH, expand=True, padx=5, pady=5)
142     output_frame.pack(fill=BOTH, expand=True, padx=5, pady=5)
143
144     #Setting up the input frame.
145     #Create the labels, sliders, and buttons for each color RGB
146     red_label = tkinter.Label(input_frame, text="R")
147     red_slider = tkinter.Scale(input_frame, from_=0, to=255, command=get_red)
148     red_button = tkinter.Button(input_frame, text="Red", command=lambda:set_color(255,0,0))
149     green_label = tkinter.Label(input_frame, text="G")
150     green_slider = tkinter.Scale(input_frame, from_=0, to=255, command=get_green)
151     green_button = tkinter.Button(input_frame, text="Green",
        command=lambda:set_color(0,255,0))
152     blue_label = tkinter.Label(input_frame, text="B")
153     blue_slider = tkinter.Scale(input_frame, from_=0, to=255, command=get_blue)
154     blue_button = tkinter.Button(input_frame, text="Blue", command=lambda:set_color(0,0,255))
155
156     #Create buttons for each complimentary color
157     yellow_button = tkinter.Button(input_frame, text="Yellow",
        command=lambda:set_color(255,255,0))
158     cyan_button = tkinter.Button(input_frame, text="Cyan",
        command=lambda:set_color(0,255,255))
159     magenta_button = tkinter.Button(input_frame, text="Magenta",
        command=lambda:set_color(255,0,255))
160
161     #Create utility buttons
162     store_button = tkinter.Button(input_frame, text="Store Color", command=store_color)
163     save_button = tkinter.Button(input_frame, text="Save", command=save_colors)
164     quit_button = tkinter.Button(input_frame, text="Quit", command=root.destroy)
165
166     #Put labels, sliders, and buttons on to the frame....Use ipadx with rbg buttons to
        define column width, then use sticky on others
167     red_label.grid(row=0, column=0, sticky='W')
168     red_slider.grid(row=1, column=0, sticky='W')
169     red_button.grid(row=2, column=0, padx=1, pady=1, ipadx=20)
170     green_label.grid(row=0, column=1, sticky='W')
171     green_slider.grid(row=1, column=1, sticky='W')
172     green_button.grid(row=2, column=1, padx=1, pady=1, ipadx=15)
173     blue_label.grid(row=0, column=2, sticky='W')
174     blue_slider.grid(row=1, column=2, sticky='W')
175     blue_button.grid(row=2, column=2, padx=1, pady=1, ipadx=18)
176     yellow_button.grid(row=3, column=0, padx=1, pady=1, sticky="WE")
177     cyan_button.grid(row=3, column=1, padx=1, pady=1, sticky="WE")
178     magenta_button.grid(row=3, column=2, padx=1, pady=1, sticky="WE")
179     store_button.grid(row=4, column=0, columnspan=3, padx=1, pady=1, sticky="WE")
180     save_button.grid(row=4, column=3, padx=1, pady=1, sticky="WE")
```

```
quit_button.grid(row=4, column=4, padx=1, pady=1, sticky="WE")

#Create the color box and color labels
color_box = tkinter.Label(input_frame, bg='black', height=6, width=15)
color_tuple = tkinter.Label(input_frame, text='(0), (0), (0)')
color_hex = tkinter.Label(input_frame, text='#000000')

#Put the color box and labels on the frame.
color_box.grid(row=1, column=3, columnspan=2, padx=35, pady=10, ipadx=10, ipady=10)
color_tuple.grid(row=2, column=3, columnspan=2)
color_hex.grid(row=3, column=3, columnspan=2)

#Setting up the output frame
#Initialize a dictionary to hold all stored colors
stored_colors = {}
stored_color = IntVar()

#Create radio buttons to select stored colors and populate each row with placeholder values
for i in range(6):
    radio = tkinter.Radiobutton(output_frame, variable=stored_color, value=i)
    radio.grid(row=i, column=0, sticky='W')

    recall_button = tkinter.Button(output_frame, text="Recall Color", state=DISABLED)
    new_color_tuple = tkinter.Label(output_frame, text="(255), (255), (255)")
    new_color_hex = tkinter.Label(output_frame, text="#ffffff")
    new_color_black_box = tkinter.Label(output_frame, bg="black", width=3, height=1)
    new_color_box = tkinter.Label(output_frame, bg='white', width=3, height=1)

    recall_button.grid(row=i, column=1, padx=20)
    new_color_tuple.grid(row=i, column=2, padx=20)
    new_color_hex.grid(row=i, column=3, padx=20)
    new_color_black_box.grid(row=i, column=4, pady=2, ipadx=5, ipady=5)
    new_color_box.grid(row=i, column=4)

    #.cget() returns the value of a specific option.  Store the text value of the tuple label and hex label
    stored_colors[stored_color.get()] = [new_color_tuple.cget('text'), new_color_hex.cget('text')]

#Initialize the starting values for the color box display
red_value = "00"
green_value = "00"
blue_value = "00"

#Run the root window's main loop
root.mainloop()
```