Python Instructions for Assignment 2

1. **Preparation of 3 photos and copy 3 photos to User-Color folder**
   - Crop in Photoshop. 900x600 px. 72 dpi.
   - Change names: *image01.jpg, image02.jpg, image03.jpg*

2. **Project Process (see detailed instruction and code)**
   1. Open Anaconda – Launch Jupyter Notebook
   2. New – Folder – Color
   3. New – Notebook – Python 3 (create a notebook)
   4. File – Rename – Color 01
   5. Follow the code

Jupyter Notebook Code – *image01.jpg , pie chart01.png*

Libraries
We will need five libraries for this project. And these libraries can be listed as following: OpenCV, Scikit-learn, NumPy, Matplotlib and Collections.

Code 1: Import libraries

```
pip install opencv-python scikit-learn numpy matplotlib
```

Code 2: Import libraries

```
from collections import Counter
from sklearn.cluster import KMeans
from matplotlib import colors
import matplotlib.pyplot as plt
import numpy as np
import cv2
```

Code 3: Reading an Image
We need to choose an image to get started.
We are using imread method by OpenCV to read the image. And then, we are converting the color format from BGR to RGB using cvtColor. The everyday images that we see on our devices are in RGB format.

```
image = cv2.imread('image01.jpg')
image = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)
plt.imshow(image)
```

Code 4: rgb_to_hex
In this function, we are converting an RGB color into Hex color format. This function will help at the end when visualizing the results of our analysis. Instead of having three different values (red, green, blue), we will have one output: hex value.

```
def rgb_to_hex(rgb_color):
    hex_color = "#"
    for i in rgb_color:
        i = int(i)
        hex_color += ("{:02x}".format(i))
    return hex_color
```

Code 5: prep_image
This function basically does the preprocessing of the image. If you want to make any changes to the picture before analyzing the colors, this is the function that you can use. We will resize and reshape the image in this step. Resizing is optional, but reshaping is needed for the color analysis model to work correctly. We will see it in the following function.

```
def prep_image(raw_img):
    modified_img = cv2.resize(raw_img, (900, 600), interpolation = cv2.INTER_AREA)
    modified_img = modified_img.reshape(modified_img.shape[0]*modified_img.shape[1], 3)
    return modified_img
```

Code 6: print value for pie chart (%)

```
def make_autopct(values):
    def my_autopct(pct):
        print(values)
        total = sum(values)
        val = int(round(pct*total/100.0))
        return '{p:.2f}%  ({v:d})'.format(p=pct,v=val)
    return my_autopct
```

Code 7: color_analysis
1) We are using k-Means to cluster the top colors. Inside the function we are passing the value of how many clusters do we want to divide. After clustering we predict the colors that weigh the most — meaning getting the most area on the image.
2)We are calling the Counter function. Counter creates a container to the elements as dictionary keys, and their value is stored as dictionary values. If you are not familiar with dictionaries, they store data in key: value pairs. They are like functions, and when you pass in the "key," you can "value" as a return. Then we order the colors according to the keys.
3)We are passing those colors in the *rgb_to_hex* function so that we can get the hex values of the colors.

4) The visualization of the result. I decided to go with a pie chart, which will be helpful to understand the weight of each color in the whole picture. After plotting the figure, I am also saving it into the computer using the *savefig* method. This way, we have a record of the result.

```
def color_analysis(img):
    clf = KMeans(n_clusters = 10)
    color_labels = clf.fit_predict(img)
    center_colors = clf.cluster_centers_

    counts = Counter(color_labels)
    ordered_colors = [center_colors[i] for i in counts.keys()]
    hex_colors = [rgb_to_hex(ordered_colors[i]) for i in counts.keys()]

    plt.figure(figsize = (16, 16))
    plt.pie(counts.values(), autopct=make_autopct(counts.values()), labels = hex_colors, colors =
hex_colors)
    plt.savefig("pie chart01.png")

    print(hex_colors)
```

Code 8: Image Color Analyzer in Action
We have the image defined earlier and assigned to the "image" variable. We will call the *prep_image* function to preprocess the image.

```
modified_image = prep_image(image)

color_analysis(modified_image)
```

Action
Top Navigation: Cell – Run All

Change to image 02
Jupyter Notebook Code – image02.jpg , pie chart02.png

Change to image 03
Jupyter Notebook Code – image03.jpg , pie chart03.png

---

3. Submission on CANVAS
Submit: pie chart 01.png, pie chart 02.png, pie chart 03.png