Exercise 1:

```
# Exercise 1: Refresh NumPy
▪ Shape and type of the array
▪ Access specific elements of a 2D NumPy array
▪ Show how to slice a NumPy array to get all rows, but only the first
two columns.
▪ Reshape a 1D array of size 12 into a 2D array with 3 rows and 4
columns?
▪ Perform matrix multiplication between two 2D arrays using NumPy.
▪ Compute the mean, median, and standard deviation of a NumPy array
▪ Perform vertical and horizontal stacking in NumPy
▪ Flatten a 3 x 4 NumPy array into a 1D array.
▪ Generate a random array of size n with values drawn from a normal
distribution
▪ Perform element-wise addition, subtraction, multiplication, and
division
```

```
import numpy as np

arr = np.array([[1, 2, 3], [4, 5, 6]])
print(arr.shape)  # Output: (2, 3)
print(arr.dtype)  # Output: int64 (or int32 depending on system)
```

```
(2, 3)
int64
```

```
# Accessing the element in the 2nd row, 3rd column
element = arr[1, 2] # Returns 6
```

```
import numpy as np

# Set print options for better readability
np.set_printoptions(precision=2, suppress=True)

print("--- 1. Shape and Type ---")
arr = np.array([[1, 2, 3], [4, 5, 6]])
print(f"Array:\n{arr}")
print(f"Shape: {arr.shape}")
print(f"Data Type: {arr.dtype}\n")

print("--- 2. Access Specific Elements ---")
# Accessing row 1, column 2 (element '6')
val = arr[1, 2]
print(f"Element at [1, 2]: {val}\n")

print("--- 3. Slicing (All rows, first two columns) ---")
slice_arr = arr[:, :2]
print(f"Sliced Array:\n{slice_arr}\n")
```

```python
print("--- 4. Reshape 1D (size 12) to 2D (3x4) ---")
arr_1d = np.arange(12)
arr_reshaped = arr_1d.reshape(3, 4)
print(f"Original 1D: {arr_1d}")
print(f"Reshaped 2D (3x4):\n{arr_reshaped}\n")

print("--- 5. Matrix Multiplication ---")
mat_a = np.array([[1, 2], [3, 4]])
mat_b = np.array([[5, 6], [7, 8]])
product = mat_a @ mat_b
print(f"Matrix A:\n{mat_a}")
print(f"Matrix B:\n{mat_b}")
print(f"Result (A @ B):\n{product}\n")

print("--- 6. Mean, Median, and Std Dev ---")
stats_arr = np.array([10, 2, 38, 24, 5, 12])
print(f"Data: {stats_arr}")
print(f"Mean: {np.mean(stats_arr):.2f}")
print(f"Median: {np.median(stats_arr)}")
print(f"Std Dev: {np.std(stats_arr):.2f}\n")

print("--- 7. Vertical and Horizontal Stacking ---")
v1 = np.array([1, 2, 3])
v2 = np.array([4, 5, 6])
print(f"Vertical Stack:\n{np.vstack((v1, v2))}")
print(f"Horizontal Stack: {np.hstack((v1, v2))}\n")

print("--- 8. Flatten a 3x4 Array ---")
# Using the reshaped array from step 4
flattened = arr_reshaped.flatten()
print(f"Flattened Array: {flattened}\n")

print("--- 9. Random Normal Distribution (size n=5) ---")
n = 5
random_data = np.random.randn(n)
print(f"Random Normal Array:\n{random_data}\n")

print("--- 10. Element-wise Operations ---")
x = np.array([10, 20, 30])
y = np.array([2, 4, 6])
print(f"x: {x}, y: {y}")
print(f"Add:      {x + y}")
print(f"Subtract: {x - y}")
print(f"Multiply: {x * y}")
print(f"Divide:   {x / y}")
```

```
--- 1. Shape and Type ---
Array:
[[1 2 3]
 [4 5 6]]
```

```
Shape: (2, 3)
Data Type: int64

--- 2. Access Specific Elements ---
Element at [1, 2]: 6

--- 3. Slicing (All rows, first two columns) ---
Sliced Array:
[[1 2]
 [4 5]]

--- 4. Reshape 1D (size 12) to 2D (3x4) ---
Original 1D: [ 0  1  2  3  4  5  6  7  8  9 10 11]
Reshaped 2D (3x4):
[[ 0  1  2  3]
 [ 4  5  6  7]
 [ 8  9 10 11]]

--- 5. Matrix Multiplication ---
Matrix A:
[[1 2]
 [3 4]]
Matrix B:
[[5 6]
 [7 8]]
Result (A @ B):
[[19 22]
 [43 50]]

--- 6. Mean, Median, and Std Dev ---
Data: [10  2 38 24  5 12]
Mean: 15.17
Median: 11.0
Std Dev: 12.33

--- 7. Vertical and Horizontal Stacking ---
Vertical Stack:
[[1 2 3]
 [4 5 6]]
Horizontal Stack: [1 2 3 4 5 6]

--- 8. Flatten a 3x4 Array ---
Flattened Array: [ 0  1  2  3  4  5  6  7  8  9 10 11]

--- 9. Random Normal Distribution (size n=5) ---
Random Normal Array:
[-0.22  0.83 -0.44 -0.1  -0.64]

--- 10. Element-wise Operations ---
x: [10 20 30], y: [2 4 6]
```

```
Add:      [12 24 36]
Subtract: [ 8 16 24]
Multiply: [ 20  80 180]
Divide:   [5. 5. 5.]
```

Exercise 2: Refresh Computer Vision

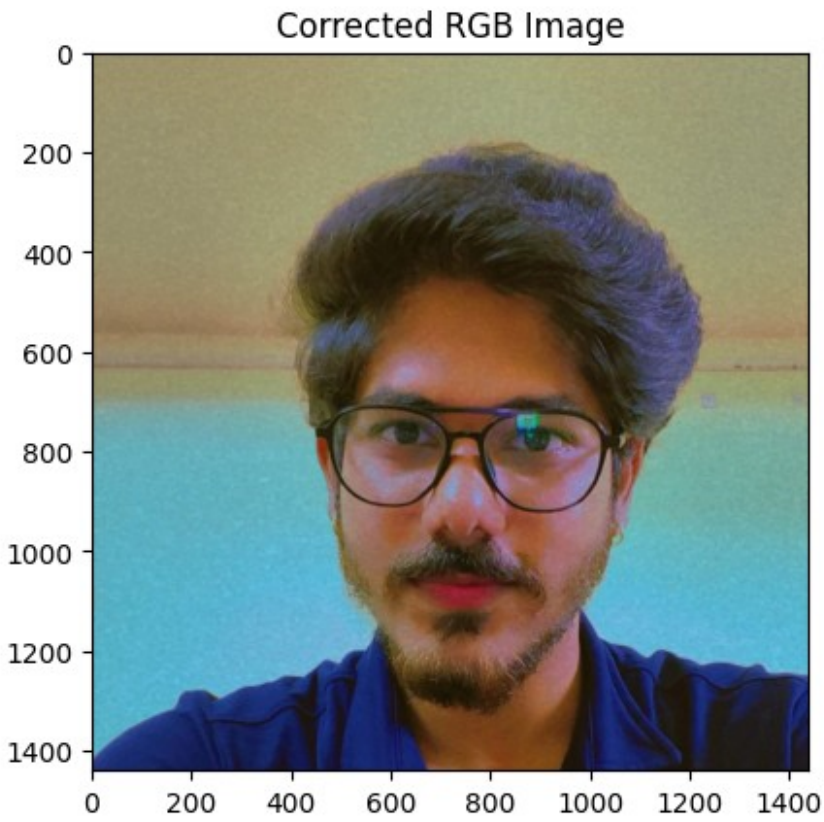
1. Load the same image using OpenCV and PIL. • Print the shape, dtype, and value range for each. • Explain any differences you observe.
2. Convert both images to grayscale. • Verify whether the outputs are numerically identical. • If not, explain why.
3. Read a color image using OpenCV and display it using matplotlib. • Why do the colors look incorrect? • Fix the issue and explain the root cause.
4. Split an RGB image into individual channels. • Visualize each channel. • Which objects appear brightest in each channel and why?
5. Convert an image from RGB to HSV. • Identify pixels corresponding to a specific color (e.g., red). • Explain why HSV is preferable to RGB for this task.
6. Resize an image to 224×224. • Identify geometric distortions. • Propose a method to preserve aspect ratio.
7. Normalize an image to the range [0,1]. • What happens if normalization is applied twice? • How does this affect visualization

```
Exercise 2: Refresh Computer Vision
1. Load the same image using OpenCV and PIL.
• Print the shape, dtype, and value range for each.
• Explain any differences you observe.
2. Convert both images to grayscale.
• Verify whether the outputs are numerically identical.
• If not, explain why.
3. Read a color image using OpenCV and display it using matplotlib.
• Why do the colors look incorrect?
• Fix the issue and explain the root cause.
4. Split an RGB image into individual channels.
• Visualize each channel.
• Which objects appear brightest in each channel and why?
5. Convert an image from RGB to HSV.
• Identify pixels corresponding to a specific color (e.g., red).
• Explain why HSV is preferable to RGB for this task.
6. Resize an image to 224×224.
• Identify geometric distortions.
• Propose a method to preserve aspect ratio.
7. Normalize an image to the range [0,1].
• What happens if normalization is applied twice?
• How does this affect visualization

import cv2
from PIL import Image
import numpy as np
```

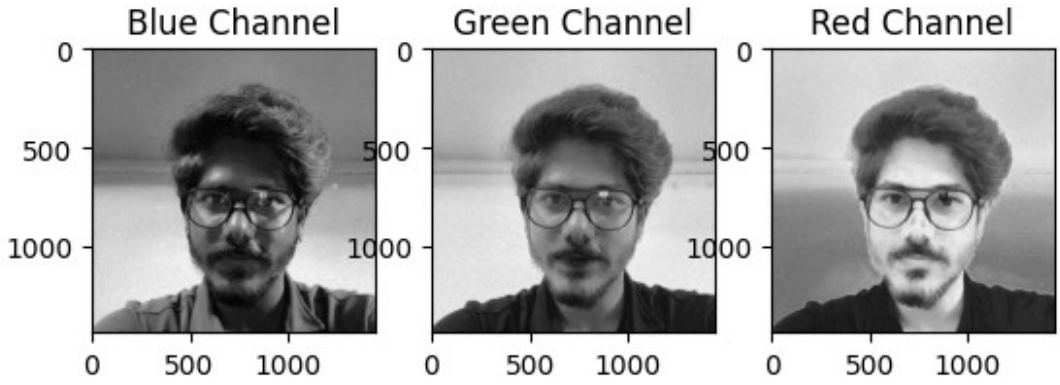
```python
# 1. Loading
img_path = '/content/profile image chnss.jpeg'
cv_img = cv2.imread(img_path)
pil_img = Image.open(img_path)

print(f"OpenCV: Shape={cv_img.shape}, Dtype={cv_img.dtype},
Range=[{cv_img.min()}, {cv_img.max()}]")
print(f"PIL: Mode={pil_img.mode}, Size={pil_img.size},
Range={pil_img.getextrema()}")

# 2. Grayscale conversion
cv_gray = cv2.cvtColor(cv_img, cv2.COLOR_BGR2GRAY)
pil_gray = np.array(pil_img.convert('L'))

# Verify identity
identical = np.array_equal(cv_gray, pil_gray)
print(f"Are grayscale outputs identical? {identical}")
```

```
OpenCV: Shape=(1440, 1440, 3), Dtype=uint8, Range=[0, 234]
PIL: Mode=RGB, Size=(1440, 1440), Range=((0, 188), (6, 201), (0, 234))
Are grayscale outputs identical? True
```

```python
import matplotlib.pyplot as plt

img_bgr = cv2.imread(img_path)

# Why it looks wrong: Matplotlib expects RGB, but OpenCV provides BGR.
# Fix:
img_rgb = cv2.cvtColor(img_bgr, cv2.COLOR_BGR2RGB)

plt.imshow(img_rgb)
plt.title("Corrected RGB Image")
plt.show()
```



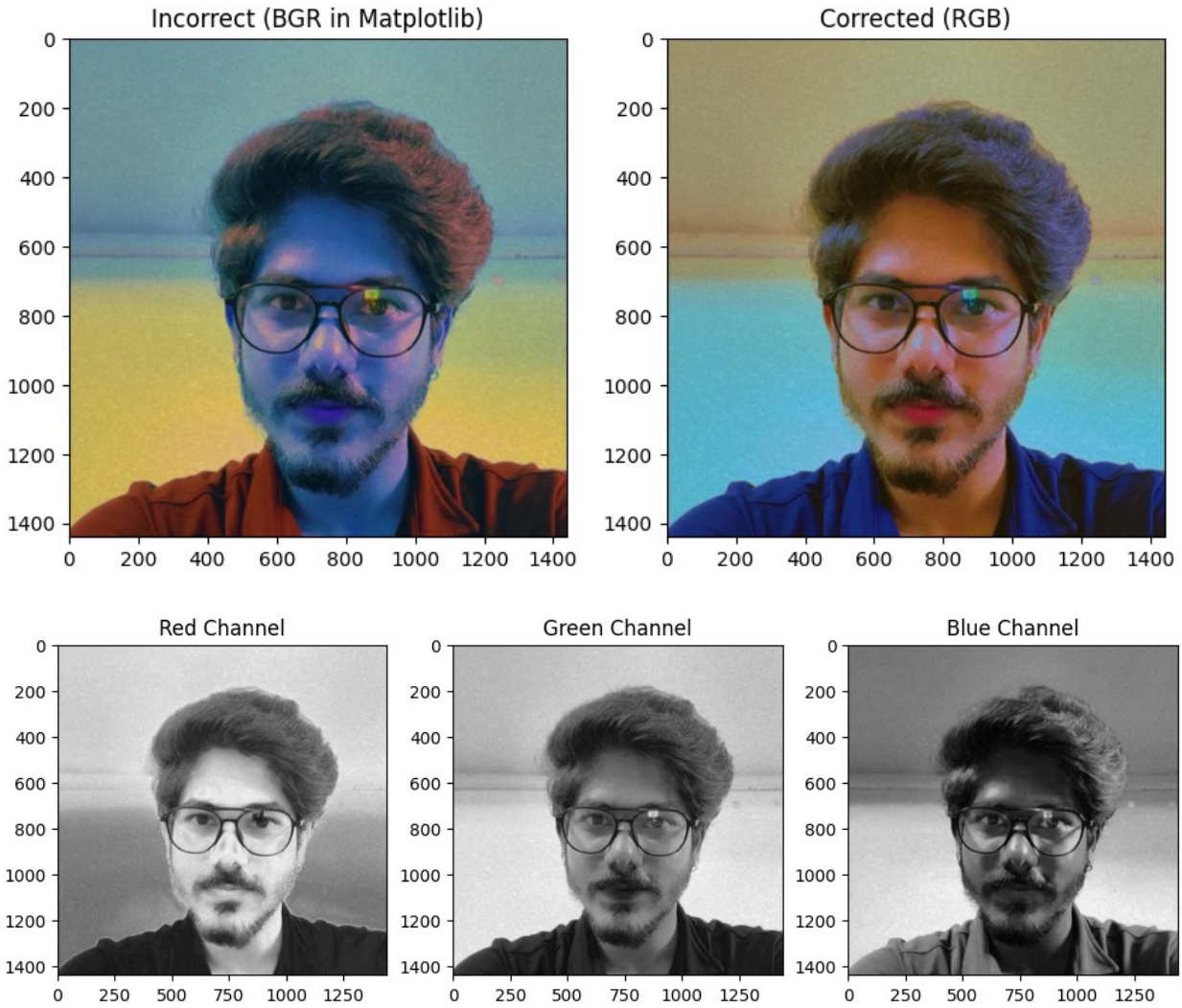

```
b, g, r = cv2.split(img_bgr) # In BGR order

# Visualization
titles = ['Blue Channel', 'Green Channel', 'Red Channel']
images = [b, g, r]

for i in range(3):
    plt.subplot(1, 3, i+1)
    plt.imshow(images[i], cmap='gray')
    plt.title(titles[i])
plt.show()
```



```
hsv = cv2.cvtColor(img_rgb, cv2.COLOR_RGB2HSV)

# Red has two ranges in HSV (at the start and end of the Hue circle)
lower_red = np.array([0, 120, 70])
upper_red = np.array([10, 255, 255])
mask = cv2.inRange(hsv, lower_red, upper_red)

norm_img = cv_img.astype(float) / 255.0
```

```
import cv2
import numpy as np
from PIL import Image
import matplotlib.pyplot as plt
# --- 1. Load Image using OpenCV and PIL ---
img_path = '/content/profile image chnss.jpeg' # Ensure you have an
image in your directory
# OpenCV (Loads as BGR by default)
cv_img = cv2.imread(img_path)
# PIL (Loads as RGB by default)
pil_img = Image.open(img_path)
print("--- 1. Metadata Comparison ---")
print(f"OpenCV: Shape={cv_img.shape}, Dtype={cv_img.dtype},
Range=[{cv_img.min()}, {cv_img.max()}]")
print(f"PIL: Mode={pil_img.mode}, Size={pil_img.size}")
# Differences: OpenCV uses BGR NumPy arrays; PIL uses its own image
object (RGB).
# --- 2. Grayscale Comparison ---
cv_gray = cv2.cvtColor(cv_img, cv2.COLOR_BGR2GRAY)
pil_gray = np.array(pil_img.convert('L'))
identical = np.array_equal(cv_gray, pil_gray)
print(f"\n--- 2. Grayscale Identity ---")
print(f"Are outputs identical? {identical}")
# --- 3. Matplotlib Display (The BGR vs RGB issue) ---
print(f"\n--- 3. Displaying Images (Check the pop-up window) ---")
img_rgb_corrected = cv2.cvtColor(cv_img, cv2.COLOR_BGR2RGB)
plt.figure(figsize=(10, 5))
plt.subplot(1, 2, 1)
plt.imshow(cv_img) # Will look "blue"
plt.title("Incorrect (BGR in Matplotlib)")
plt.subplot(1, 2, 2)
plt.imshow(img_rgb_corrected) # Will look correct
plt.title("Corrected (RGB)")
plt.show()
# --- 4. Channel Splitting ---
b, g, r = cv2.split(cv_img)
channels = [r, g, b]
titles = ['Red Channel', 'Green Channel', 'Blue Channel']
plt.figure(figsize=(12, 4))
for i in range(3):
    plt.subplot(1, 3, i+1)
```

```
    plt.imshow(channels[i], cmap='gray')
    plt.title(titles[i])
plt.show()
# --- 5. RGB to HSV ---
hsv_img = cv2.cvtColor(cv_img, cv2.COLOR_BGR2HSV)
lower_blue = np.array([100, 50, 50])
upper_blue = np.array([130, 255, 255])
mask = cv2.inRange(hsv_img, lower_blue, upper_blue)
print(f"\n--- 5. HSV Filtering ---")
print("HSV is better because Hue (color) is separate from Value
(brightness).")
# --- 6. Resizing & Aspect Ratio ---
distorted = cv2.resize(cv_img, (224, 224))
h, w = cv_img.shape[:2]
aspect = w / h
new_h = 224
new_w = int(new_h * aspect)
resized_aspect = cv2.resize(cv_img, (new_w, new_h))
print(f"\n--- 6. Resizing ---")
print(f"Distorted Shape: {distorted.shape}")
print(f"Aspect Preserved Shape: {resized_aspect.shape}")
# --- 7. Normalization ---
norm_1 = cv_img.astype(float) / 255.0
norm_2 = norm_1 / 255.0 # Applying twice
print(f"\n--- 7. Normalization ---")
print(f"Single Norm Range: [{norm_1.min():.2f}, {norm_1.max():.2f}]")
print(f"Double Norm Range: [{norm_2.min():.4f}, {norm_2.max():.4f}]")
# Double normalization makes the image appear black as values approach
0.

--- 1. Metadata Comparison ---
OpenCV: Shape=(1440, 1440, 3), Dtype=uint8, Range=[0, 234]
PIL: Mode=RGB, Size=(1440, 1440)

--- 2. Grayscale Identity ---
Are outputs identical? True

--- 3. Displaying Images (Check the pop-up window) ---
```




```
--- 5. HSV Filtering ---
HSV is better because Hue (color) is separate from Value (brightness).

--- 6. Resizing ---
Distorted Shape: (224, 224, 3)
Aspect Preserved Shape: (224, 224, 3)

--- 7. Normalization ---
Single Norm Range: [0.00, 0.92]
Double Norm Range: [0.0000, 0.0036]
```