# **Assignment 1**

# **Introduction to OpenCV, Capturing Videos, Digital Images & Color Standards**

| Total Mark: | 12 marks (6% of the total course grade)   * 9 out of 12: Learn@Seneca Submission (Due: Monday May 26 at 8:00am) * 3 out of 12: Assignment Demo (During the Lab of Week 4) |
| --- | --- |
| Submission file(s): | * Assignment1\_1.py / Assignment1\_1.ipynb * Assignment1\_2.py / Assignment1\_2.ipynb * Assignment1.docx (this document with your answers) |

Please work **within your group** to complete this assignment.

This assignment is worth 6% of the total course grade and will be evaluated through your written submission, as well as the assignment demo.

During the assignment demo, group members are *randomly* selected to explain the submitted solution. Group members who are not present during the assignment demo will lose the demo mark.

Please submit the submission file(s) through Learn@Seneca.

***Please paste the resulting images and answers in this document.***

## **Part I: A Simple OpenCV Project**

For this assignment, you need a webcam, or a digital camera connected and installed on your machine.

1. Open **Anaconda Navigator**, then on Home tab, choose **ocv/socv** environment.
2. Launch PyCharm, Visual Studio Code, or Jupyter Notebook (whichever you prefer) and select ocv as the virtual environment.
3. Create a program (save as Assignment01\_1). Copy and paste the following code in a code block (Reference: [OpenCV: Getting Started with Videos](https://docs.opencv.org/4.11.0/dd/d43/tutorial_py_video_display.html)).

|  |
| --- |
| import cv2 as cv  # Start a video capture, using device's camera  cap = cv.VideoCapture(0)  # Check if video file opened successfully  if (cap.isOpened() == False):      print("Error opening video stream or file")  frame\_width = int(cap.get(3))  frame\_height = int(cap.get(4))  print("Frame width: " , frame\_width)  print("Frame height: " , frame\_height)  # Read until video is completed  while(cap.isOpened()):      # Capture frame-by-frame      ret, frame = cap.read()      if ret == False:          break      # Display the frame      cv.imshow('frame',frame)      key = cv.waitKey(25)      # Press Q on keyboard to exit      if key & 0xFF == ord('q'):          break  # Release the video capture  cap.release()  # Close all the frames  cv.destroyAllWindows() |

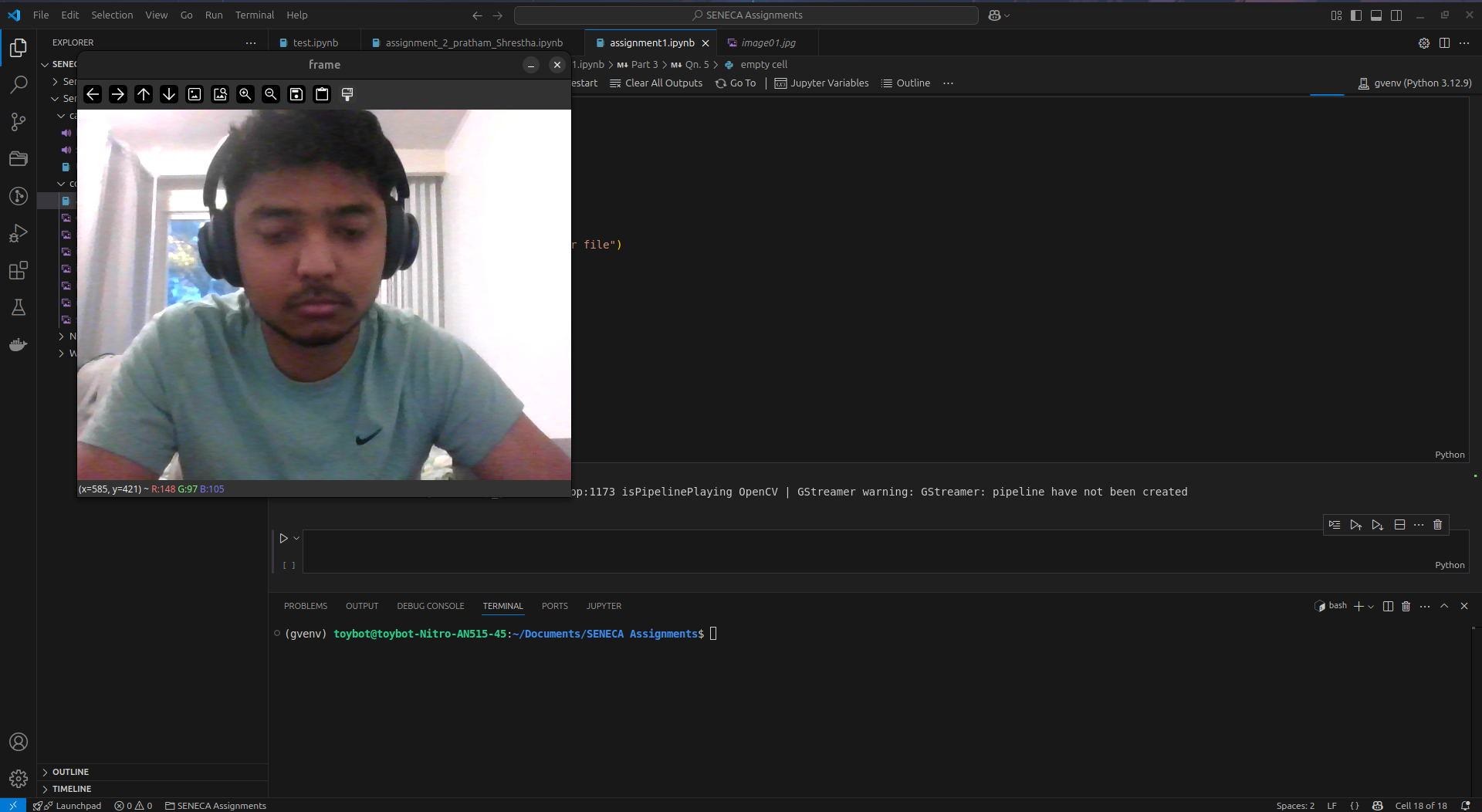
1. Run the code. You should see your webcam’s video feed. Press ‘q’ to exit.
2. Briefly explain what the code is doing.
3. What are the frame width and height values?
4. Change the parameter for cv.waitKey() from 25 to 100, and 500. What happens? Explain.

## **Part II: A Photo Booth Application (Read, Display, Write, and Pad Images)**

Modify Assignment1\_1 to:

1. Take a snapshot whenever the ‘x’ key is pressed.
2. Crop 15 pixels around the snapshot image.
3. Pad the snapshot with 40 pixels using replicate padding (or a constant boarder with the color you like) See: [OpenCV: Adding borders to your images](https://docs.opencv.org/4.11.0/dc/da3/tutorial_copyMakeBorder.html).
4. Save the snapshot. Use image names such as ‘image01.jpg’, ‘image02.jpg’, etc., automatically incrementing the filename counter. See: [OpenCV: Getting Started with Images](https://docs.opencv.org/4.11.0/db/deb/tutorial_display_image.html).
5. Show this image in a new window for 1 second. Then automatically close it and return to the camera feed.
6. Exit whenever the ‘q’ or ESC key is pressed.

Paste one of the snapshots here.



## **Part III: Calculations**

For this part, just add your answers in the document.

Calculating FPS:

1. Calculate the approximate FPS implemented in the code (use *waitKey()*). Ignore all other delays including 1 second freezes.
2. How would you change the code to have an FPS of 5? (once use *waitKey()* and once use a method with *cap*).
3. What do you observe when you apply each solution? What are the limitations of each method?

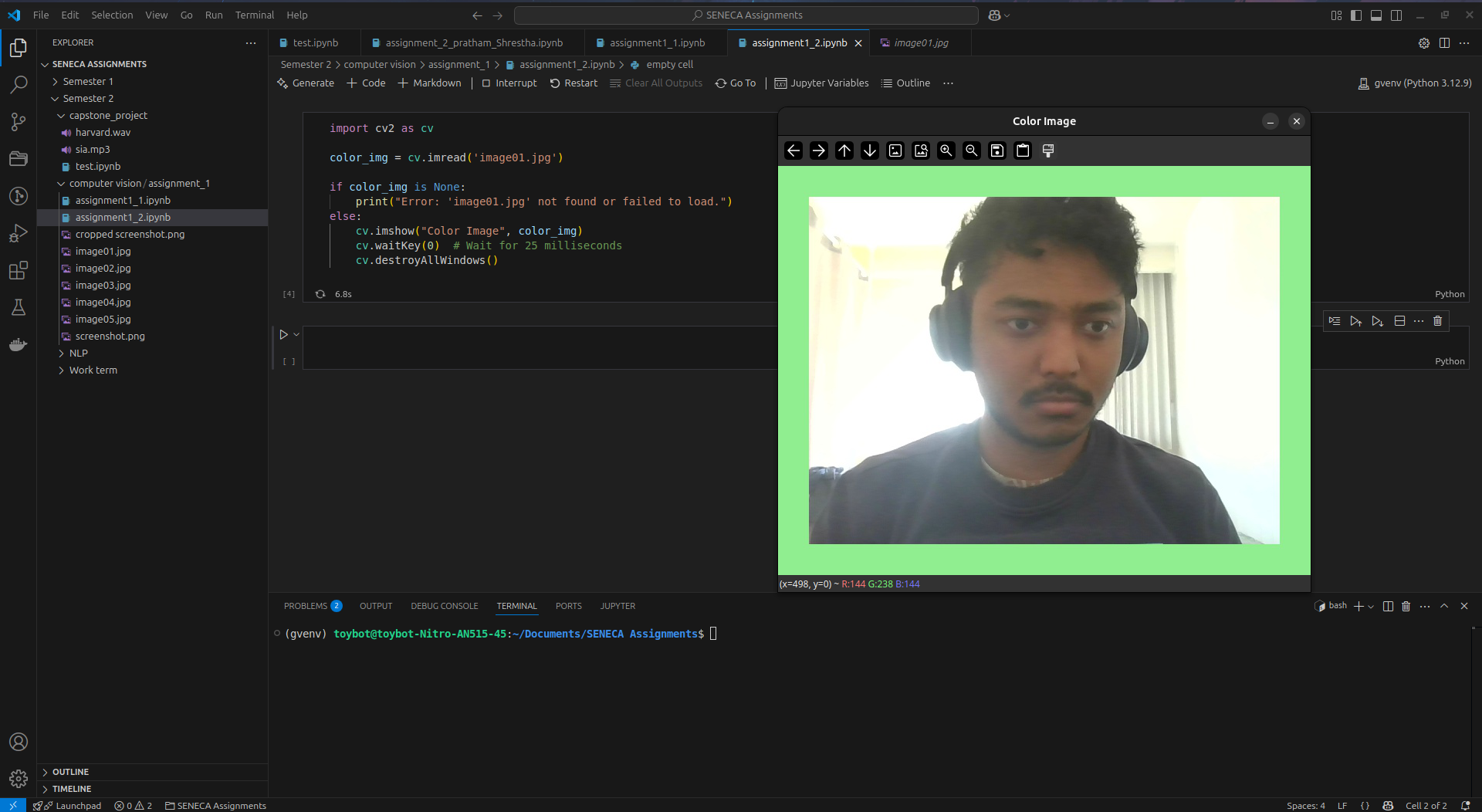
Find image01.jpg and look at its properties.

1. What is the image resolution (dimensions)?
2. Does this match the width and height output of Part I? Why is that?
3. What is the bit depth? What does bit depth show?
4. What is the file size in bytes?
5. What would the file size be if the image was not compressed? Show the calculations to support.
6. Calculate the compression ratio as the ratio between the uncompressed size and the compressed size.
7. How do you know that image is compressed? Explain.

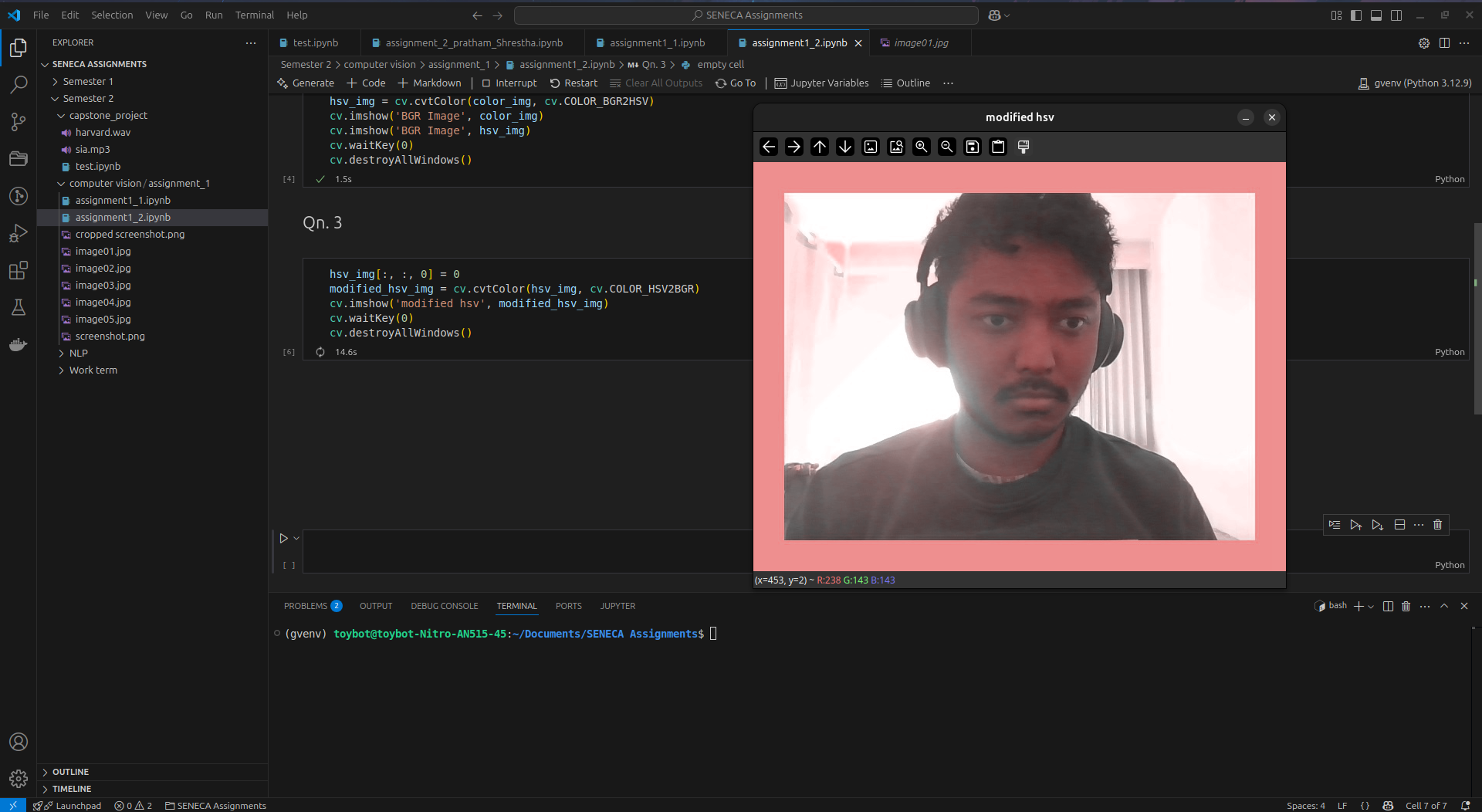
## **Part IV: Color Standards**

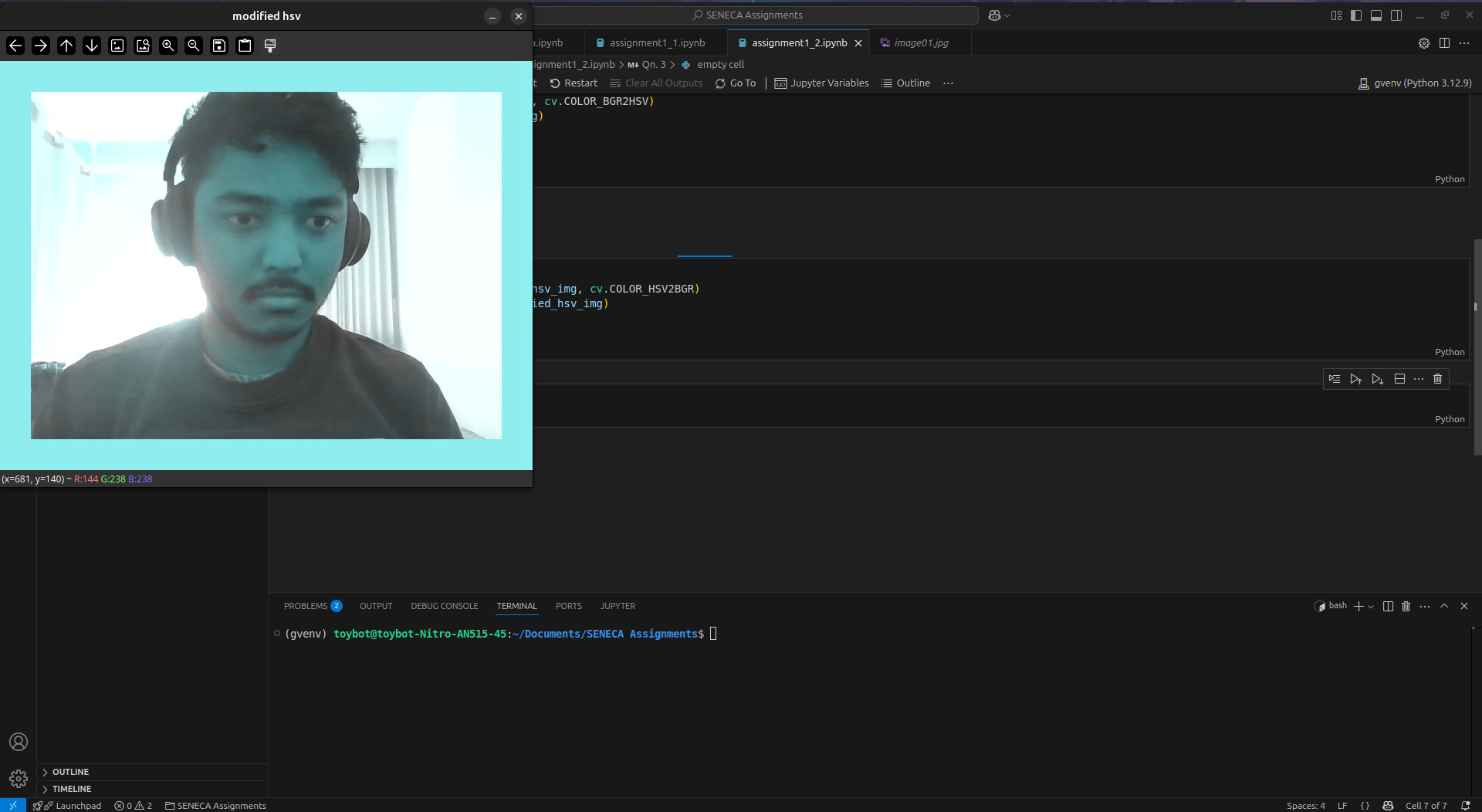
Create a program (save as Assignment01\_2). Include code to:

1. Open a color image and display. Paste a sample here.



1. Convert it from BGR to HSV color-space (See: [OpenCV: Color Space Conversions](https://docs.opencv.org/3.4/d8/d01/group__imgproc__color__conversions.html" \l "ga397ae87e1288a81d2363b61574eb8cab)).
2. Set the H once to 0 and once to 90. Paste the results here.





1. What is being done in this part? Explain.

Changing Hue from 0 to 90 changes the color tone from redish to cyan blueish.

Please note that, for HSV color-space in OpenCV, hue range is [0,179], saturation range is [0,255], and value range is [0,255].

## **Part V: Group Work**

Add this declaration to your file:

We, ------------ (mention assigned group number and your names), declare that the attached assignment is our own work in accordance with the Seneca Academic Policy. We have not copied any part of this assignment, manually or electronically, from any other source including web sites, unless specified as references. We have not distributed our work to other students.

Specify what each member has done towards the completion of this assignment:

|  |  |  |
| --- | --- | --- |
|  | **Name** | **Task(s)** |
| **1** |  |  |
| **2** |  |  |
| **3** |  |  |