PICZAP

Team Name:

Pycake

Guide:

Mr. Kailash

Team Members:

- Parmeet Singh Banwait - 20BAI10141
- MVN Rajesh Reddy - 20BAI10249
- Sparsh Handa - 20BAI10295
- Mohd Mohsin Khan - 20BAI10340

OBJECTIVE

How to add watermarks on images using OpenCV in Python

Table of Contents

01

INTRODUCTION

What is a watermark ?

- ★ Watermarking is the process of superimposing a logo or piece of text atop a document or image file.
- ★ It’s an important process when it comes to both the copyright protection and marketing of digital works.
- ★ We will look at a few reasons why watermarking images and documents is important and explore how to create a watermark that’s effective for our work.

Uses of watermarks

COPYRIGHT PROTECTION

IMAGE TAMPERING DETECTION

Uses of watermarks

PROOF OF OWNERSHIP

Watermark also gives the proof of ownership. Any person can claim his/her work with his name or logo as watermark.

CONTENT AUTHENTICATION

It ensures the integrity of the shared **content**, and generally also the identity of its owner.

Uses of watermarks

PROTECTION

Watermarks can be used to protect confidential information.It prevents people from using them without your consent.

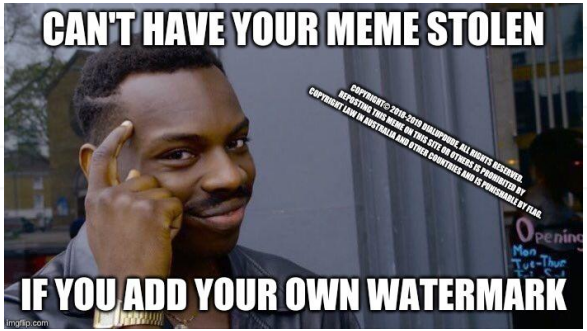
INFORMATION

Provides useful technical information about how the photo was taken, the shutter speed, aperture, whether Flash was used, brand of camera used etc.

ADD CAPTIONS

A picture is worth a thousand words, but no harm in adding a few words on top the picture that describe it in a better way and make it more meaningful.

Uses of watermarks



MEMES & COMICS

You can find the photos online, or you can use one of your own photos, add a funny caption or quote and share it with the world.

MASH IMAGES

We can mash images together. Take one photo and add another photo on top of it or we can even try adding the same photo as a watermark to produce some amazing effects.

SHOW CREATIVITY

Watermarking is all about adding text on an image, but what you do with it is completely up to you. You can be as creative as you can get.

02

EXISTING WORK WITH LIMITATIONS

Existing work

Our application is mainly divided into two parts-

★ Comparison
- It basically compares the different values and tell us how they differ from each other.

★ User Interactive phase
- It asks the user for the image and the action to be performed (like addition of watermark, removal of watermark, cropping, compressing and resizing) and the final output is being displayed it to the user.

Limitations of Watermarks

OBVIOUS DISTRACTION

One of the major complaints raised by photographers and viewers alike is the distraction a watermark can create within the image.

NOT FOOLPROOF

An unscrupulous thief may even place his own watermark on the image after removing yours. Watermarks do not offer foolproof protection against theft.

IMAGE COPYING

Watermark does not prevent image copying. Copy protection is not easy to achieve in open systems.

REDUCES SOCIAL SHARING

Social networking sites help spread information and news, but if your photos are heavily watermarked they may not receive much attention.

LIMITED SUCCESS

With current digital image software, it is very easy for someone to eliminate the watermark anyway so they can freely distribute the image.

Promotes Negative Image of you

- ★ A distracting watermark gives the impression of a photographer more focused on saving his work than showing it.
- ★ Large watermarks across a product photo may frustrate shoppers, as they can't see the product clearly without the distracting watermark over it.

03

PROPOSED WORK AND METHODOLOGY

What is PICZAP?

A modern innovation towards watermarking.

- ★ It is methodology which is used to actually help the clients/users to cope up with their work in a much more reliable manner.
- ★ It helps the user to perform several actions like addition of watermark, deletion of watermark, compressing, cropping and resizing of the image.

03

NOVELTY OF THE PROJECT

Why our software is different from others?

ONE OF ITS KIND SOFTWARE

NOT COMPLEX

EASY TO USE

04

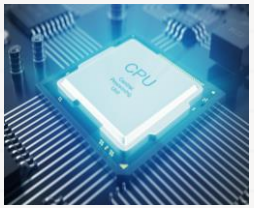
REQUIREMENTS

HARDWARE REQUIREMENTS

KEYBOARD

MONITOR

MOUSE



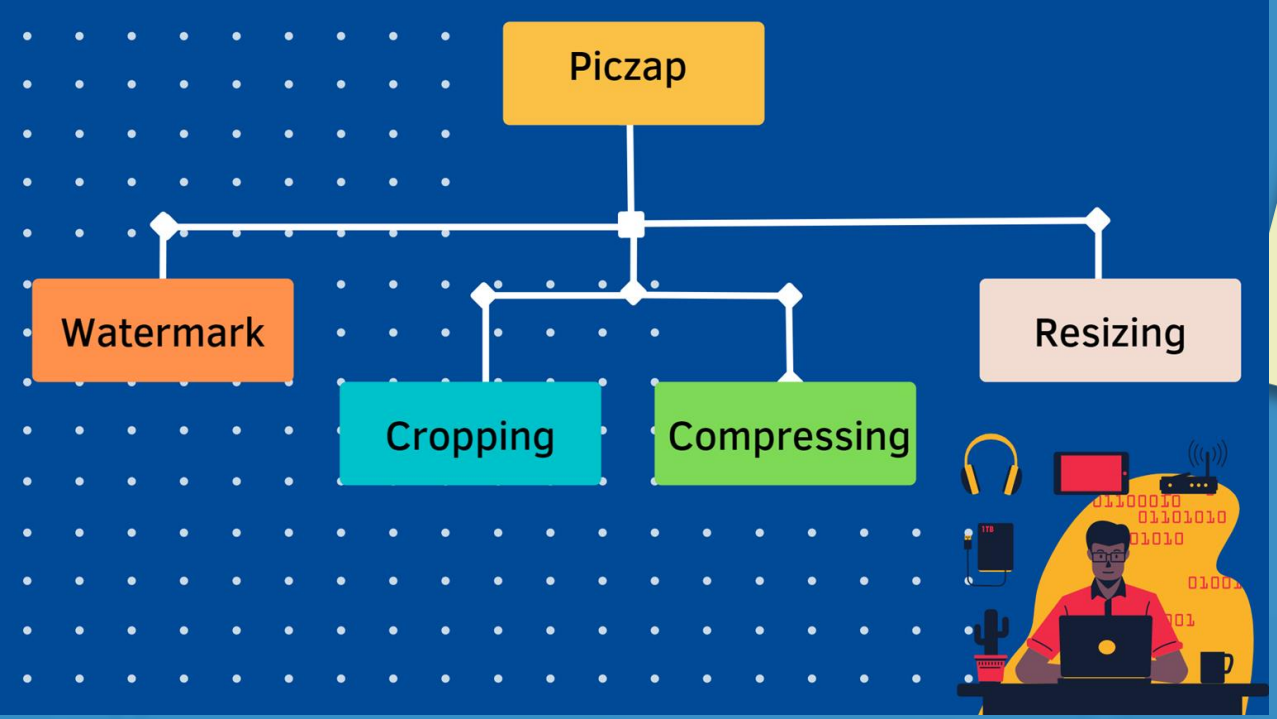
CPU

SOFTWARE REQUIREMENTS

Python

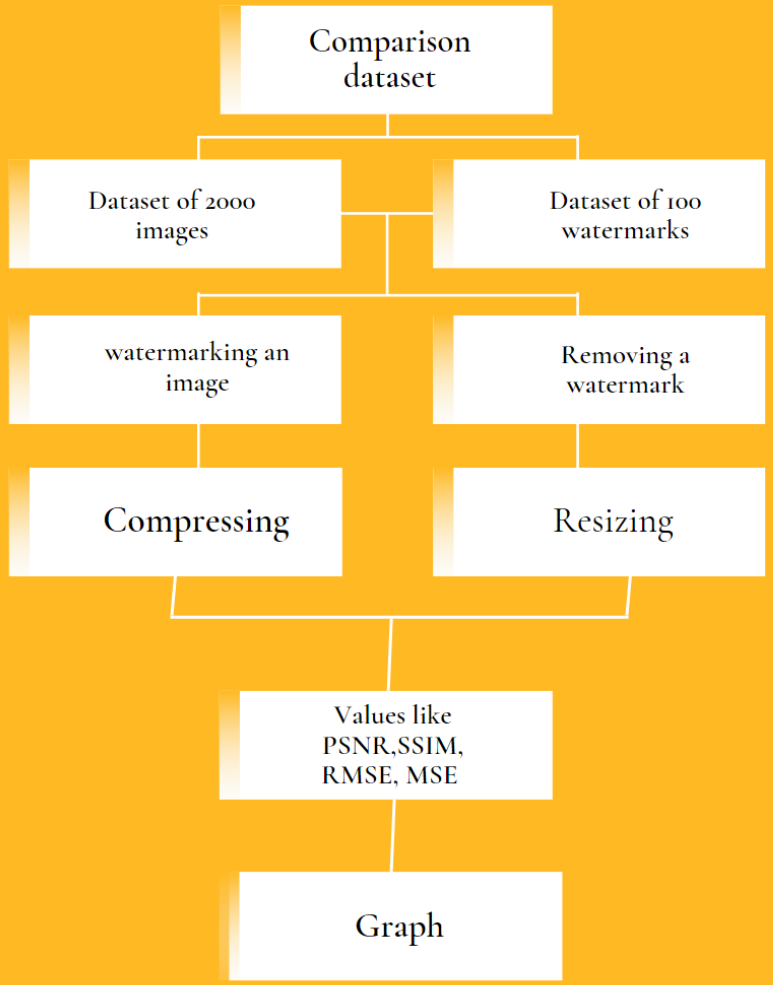
05

OVERALL SYSTEM ARCHITECTURE DIAGRAM



Software Architecture

THIS IS WHAT WE PLANNED !!



Overall Architecture Diagram

06

LITERATURE REVIEW

- ★ A watermark is an relating image or pattern in paper that appears as colorful tones of lightness/ darkness when viewed by transmitted light caused by consistence or viscosity variations in the paper. Watermarks have been used on postage prints, currency, and other government documents to discourage counterfeiting.
- ★ The origin of the water part of a watermark can be plant back when a watermark was commodity that only was in paper. At that time the watermark was created by changing the consistency of the paper and thereby creating a shadow/ lightness in the watermarked paper.This was done while the paper was still wet/ watery and thus the mark created by this process is called a watermark.

- ★ Watermarks were first introduced in Fabriano, Italy, in 1282.
- ★ The term "Digital Watermark" was chased by Andrew Tirkel and Charles Osborne in December 1992. The first successful embedding and birth of a stenographic spread diapason watermark was demonstrated in 1993 by Andrew Tirkel, Charles Osborne and Gerard Rankin.
- ★ They were used as a means to identify the paper maker or the trade council that manufactured the paper. The marks frequently were created by a line darned onto the paper earth. Watermarks continue to be used moment as manufacturer's marks and to help phony.

07

MODULE DESCRIPTION

LIBRARIES USED IN CODES

cv2

image processing and performing computer vision tasks

PIL from Image

adds support for opening, manipulating, and saving many different image file formats

os

provides functions for interacting with the operating system

sys

provides various functions and variables that are used to manipulate different parts of the Python runtime environment

random

generate random numbers. These are pseudo-random numbers means these are not truly random.

skimage

designed for image preprocessing, easy usage, it has a well-structured documentation that lists down all the modules

LIBRARIES USED IN CODES

math

provides us access to some common math functions and constants in Python

matplotlib

creates a figure, creates a plotting area in a figure, plots some lines in a plotting area, decorates the plot with labels, etc.

imagechops

contains a number of arithmetical image operations, called channel operations

operator

Operator are used to perform operations on variables and values.

argparse

The argparse module makes it easy to write user-friendly command-line interfaces.

imutils

A series of convenience functions to make basic image processing operations such as translation, rotation, resizing,and displaying Matplotlib images

METHODS USED FOR COMPARING IMAGES

MSE (Mean Square Error)

It tells you how close a regression line is to a set of points.

NRMSE (Normalised RMSE)

RMSE (Root Mean Squared Error)

It measures the amount of change per pixel due to the processing.

$$RMSE = \sqrt{MSE}$$

It allows the user to calculate the normalized root mean square error (NRMSE) as absolute value between predicted and observed values using different type of normalization methods.

METHODS USED FOR COMPARING IMAGES

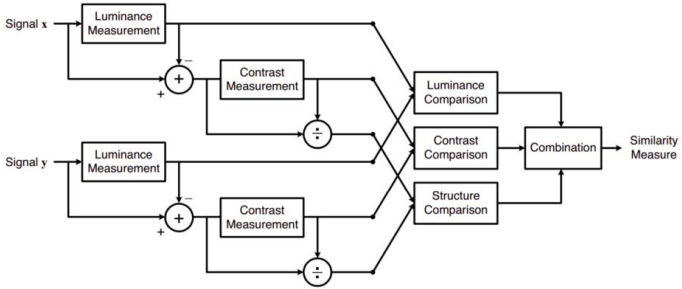
PSNR
(Peak signal-to-noise Ratio)

It computes the peak signal-to-noise ratio, in decibels, between two images. This ratio is used as a quality measurement between the original and a compressed image.

PSNR = 20 log10(MAX/(MSE)^(1/2))

SSIM
(Structural Similarity Index Measure)

It gives normalized mean value of structural similarity between the two images.



08

MODULE WORKFLOW EXPLANATION

For Watermarking of Images

Step 1: Select an image to be watermarked.

Step 2: Select the watermark to be added.

Step 3: The image is scaled and hence downsize an image for a particular resolution or we keep the image to its original resolution when required.

Step 4: The images provided by the user is then merged and the watermarked image is being created.

Step 5: The watermarked image will be shown.

For Comparing Values of Watermarked Images

Step 1: Select database of 2000 images to be watermarked.

Step 2: Select the database of 100 watermarks to be added.

Step 3: The following operations (like watermark addition, cropping, compressing) are performed.

Step 4: After the operations are performed then the PSMR, SSIM, RMSE and MSE values are being calculated.

Step 5: Then these values are being put up in an excel sheet and plotted.

For Cropping, Resizing or compressing of Images

Step 1: Select an image to be edited.

Step 2: Give dimensions for editing of images as input.

Step 3: The software will process and will edit the image to the provided dimensions.

Step 4: Then the final edited image will be displayed.

09

SNAPSHOTS OF IMPLEMENTATION OF OUR CODE

```python
# importing cv2
from tkinter import filedialog
from tkinter import *
from skimage.morphology import binary_dilation, binary_erosion
import random
import numpy as np
import tkinter as tk
import sys
import os
import cv2
# Importing Image class from PIL module
from PIL import Image

# Opens a image in RGB mode


def watermark_image(watermark, logo):

    # calculating dimensions
    # height and width of the logo
    h_logo, w_logo, _ = logo.shape

    # height and width of the image
    h_img, w_img, _ = watermark.shape

    # calculating coordinates of center
    # calculating center, where we are going to
    # place our watermark
    center_y = int(h_img/2)
    center_x = int(w_img/2)

    # calculating from top, bottom, right and left
    top_y = center_y - int(h_logo/2)
    left_x = center_x - int(w_logo/2)
    bottom_y = top_y + h_logo
    right_x = left_x + w_logo

    # adding watermark to the image
    destination = watermark[top_y:bottom_y, left_x:right_x]
    result = cv2.addWeighted(destination, 1, logo, 0.5, 0)

    # displaying and saving image
    watermark[top_y:bottom_y, left_x:right_x] = result
    cv2.imwrite("watermarked.jpg", watermark)
    cv2.imshow("Watermarked Image", watermark)
    cv2.waitKey(0)
    cv2.destroyAllWindows()
```

```
    cv2.waitKey(0)
    cv2.destroyAllWindows()


def crop_image(z):
    im = Image.open(z)
    left = int(im.size[0]/2-224/2)
    upper = int(im.size[1]/2-100/2)
    right = left + 224
    lower = upper + 100
    im_cropped = im.crop((left, upper, right, lower))
    newsize = (im.size[0], im.size[1])
    im1 = im_cropped.resize(newsize)
    im1.save("cropped.jpg")
    im1.show


def compress_image(x):
    def initialize_K_centroids(X, K):
        m = len(X)
        return X[np.random.choice(m, K, replace=False), :]

    def find_closest_centroids(X, centroids):
        m = len(X)
        c = np.zeros(m)
        for i in range(m):
            # Find distances
            distances = np.linalg.norm(X[i] - centroids, axis=1)

        # Assign closest cluster to c[i]
            c[i] = np.argmin(distances)

        return c

    def compute_means(X, idx, K):
        _, n = X.shape
        centroids = np.zeros((K, n))
        for k in range(K):
            examples = X[np.where(idx == k)]
            mean = [np.mean(column) for column in examples.T]
            centroids[k] = mean
        return centroids

    def load_image(path):
        """ Load image from path. Return a numpy array """
        image = Image.open(path)
        return np.asarray(image) / 255
```

```
def load_image(path):
    """ Load image from path. Return a numpy array """
    image = Image.open(path)
    return np.asarray(image) / 255

def find_k_means(X, K, max_iters=10):
    centroids = initialize_K_centroids(X, K)
    previous_centroids = centroids
    # for _ in range(max_iters):
    image = load_image(x)
    w, h, d = image.shape
    print('Image found with width: {}, height: {}, depth: {}'.format(w, h, d))

    X = image.reshape((w * h, d))

    idx = find_closest_centroids(X, centroids)
    centroids = compute_means(X, idx, K)
    if (centroids == previous_centroids).all():
        return centroids
    else:
        previous_centroids = centroids

    return centroids, idx

try:
    image_path = sys.argv[1]
    assert os.path.isfile(x)
except (IndexError, AssertionError):
    print('Please specify an image')

image = load_image(x)
w, h, d = image.shape
print('Image found with width: {}, height: {}, depth: {}'.format(w, h, d))

X = image.reshape((w * h, d))
K = 20

# K = 20, the desired number of colors in the compressed image

colors, _ = find_k_means(X, K, max_iters=20)

idx = find_closest_centroids(X, colors)

idx = np.array(idx, dtype=np.uint8)
X_reconstructed = np.array(
    colors[idx, :] * 255, dtype=np.uint8).reshape((w, h, d))
```

```
    idx = np.array(idx, dtype=np.uint8)
    X_reconstructed = np.array(
        colors[idx, :] * 255, dtype=np.uint8).reshape((w, h, d))
    compressed_image = Image.fromarray(X_reconstructed)

    compressed_image.save('compressed.jpg')


def remove_watermark():
    img = cv2.imread('watermarked.jpg')
    mask = cv2.imread('Watermark.jpg', 0)
    dst = cv2.inpaint(img, mask, 20, cv2.INPAINT_TELEA)
    cv2.imshow('dst', dst)
    cv2.waitKey(0)
    cv2.destroyAllWindows()


def openFile():
    f = filedialog.askopenfilename(initialdir="D:\\Code\\Input\\2000",
                                   title="Open file okay?",
                                   filetypes=(("Image files", "*.jpg"),
                                              ("all files", "*.*")))
    # file = open(filepath, 'r')
    # print(file.read())
    # print(filepath)
    # print(f, file=open("D://Code/Path.txt", "w"))
    with open("D://Code/Path.txt", 'w') as p:
        p.write(f)


def openFile_1():
    f = filedialog.askopenfilename(initialdir="C:\\Users\\ASUS\\Pictures\\EEE Experiment\\Code",
                                   title="Open file okay?",
                                   filetypes=(("Image files", "*.jpg"),
                                              ("all files", "*.*")))
    # file = open(filepath, 'r')
    # print(file.read())
    # print(filepath)
    # print(f, file=open("D://Code/Path.txt", "w"))
    with open("D://Code/Path_1.txt", 'w') as p:
        p.write(f)

    # x = filedialog.askopenfilename()
window = Tk()
window.geometry("150x150")
window.config(background="#03071e")
```

```
                                        ("all files", "*.*")))
    # file = open(filepath, 'r')
    # print(file.read())
    # print(filepath)
    # print(f, file=open("D://Code/Path.txt", "w"))
    with open("D://Code/Path_1.txt", 'w') as p:
        p.write(f)

    # x = filedialog.askopenfilename()
window = Tk()
window.geometry("150x150")
window.config(background="#03071e")
button = Button(text="Open Image", command=lambda: openFile(),
                foreground="#000000")
button.config(background="#ffba08")
button.config(height=2, width=10)
button.pack()
window.mainloop()
x = open("D://Code/Path.txt", "r")
y = x.read()
y = str(y)

window = Tk()
window.geometry("150x150")
window.config(background="#03071e")
button = Button(text="Open Watermark", command=lambda: openFile_1(),
                foreground="#000000")
button.config(background="#ffba08")
button.config(height=2, width=20)
button.pack()
window.mainloop()
j = open("D://Code/Path_1.txt", "r")
i = j.read()
i = str(i)

# water(q)
print(y)
img = cv2.imread(y)
wat = cv2.imread(i)
# cv2.imshow("", wat)
# cv2.waitKey(0)
ch = 0
while(ch != 5):
    print("1.to watermark the image")
    print("2.to compress the image")
    print("3.to crop the image")
    print("4.to remove watermark")
```

```python
from math import log10, sqrt
import cv2
import numpy as np


def PSNR(original, compressed):
    mse = np.mean((original - compressed) ** 2)
    if(mse == 0):  # MSE is zero means no noise is present in the signal .
        # Therefore PSNR have no importance.
        return 100
    max_pixel = 255.0
    psnr = 20 * log10(max_pixel / sqrt(mse))
    return psnr


x = int(input("Enter no."))
n = 1814
z = 0

for i in range(0, x):
    n = n+1
    z = str(n)
    original = cv2.imread(
        "D://Code/Input/2000/Image ("+z+").jpg")
    compressed = cv2.imread(
        "D://Code/Compress/compressed ("+z+").jpg")
    # compressed.show()
    value = PSNR(original, compressed)
    print(value, file=open("D://Code/Compress/PSNR.txt", "a"))
    mse = np.mean((original - compressed) ** 2)
    rmse = sqrt(mse)
    nrmse = rmse/mse
    print(nrmse, file=open("D://Code/Compress/NRMSE.txt", "a"))
    print(rmse, file=open("D://Code/Compress/RMSE.txt", "a"))
    print(mse, file=open("D://Code/Compress/MSE.txt", "a"))
    print(n)
```

```
import imutils
import cv2

x = int(input("Enter no."))
n = 1814
z = 0

for i in range(0, x):
    n = n+1
    z = str(n)

    # load the two input images
    imageA = cv2.imread("D://Code/Input/2000/Image ("+z+").jpg")
    imageB = cv2.imread("D://Code/Compress/compressed ("+z+").jpg")

    # convert the images to grayscale
    grayA = cv2.cvtColor(imageA, cv2.COLOR_BGR2GRAY)
    grayB = cv2.cvtColor(imageB, cv2.COLOR_BGR2GRAY)
    # compute the Structural Similarity Index (SSIM) between the two
    # images, ensuring that the difference image is returned
    (score, diff) = structural_similarity(grayA, grayB, full=True)
    diff = (diff * 255).astype("uint8")
    print("{}".format(score), file=open("D://Code/Compress/SSIM.txt", "a"))
    print(n)
```

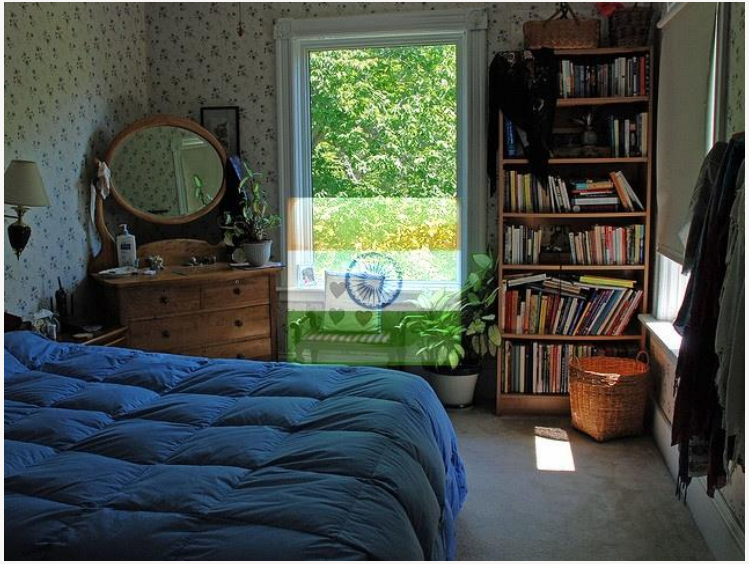
PYTHON CODE (Comparing images)

INPUT

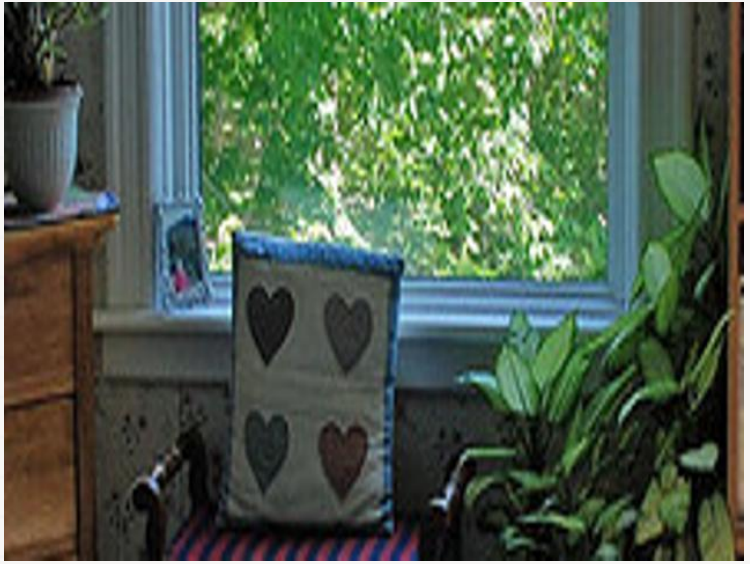
IMAGE

WATERMARK

OUTPUT

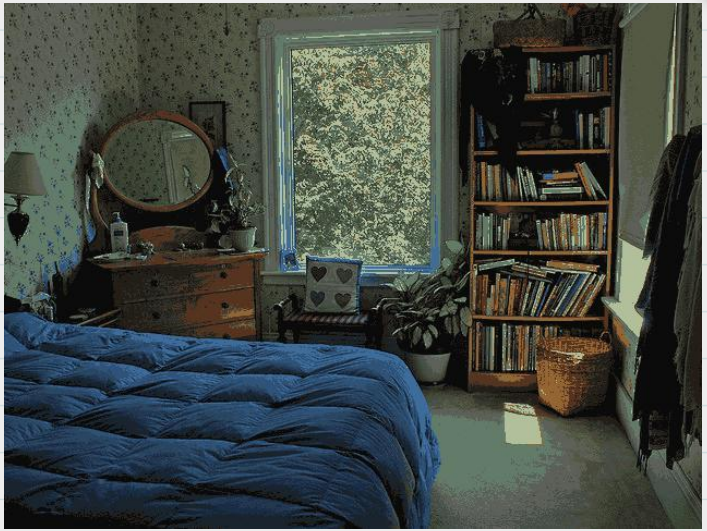


WATERMARKED IMAGE



CROPPED IMAGE

OUTPUT



COMPRESSED IMAGE

OUTPUT (Comparing Images)

K2000 | fx | 0.005054334780298

	Watermark						Compress						Crop				
	MSE	PSNR	RMSE	NRMSE	SSIM		MSE	PSNR	RMSE	NRMSE	SSIM		MSE	PSNR	RMSE	NRMSE	SSIM
3	18.12863	35.54715	4.257772	0.234865	0.965864		57.39872	30.54178	7.576195	0.131992	0.855112		104.5862	27.93606	10.22674	0.097783	0.134396
4	15.58705	36.20316	3.948044	0.25329	0.945308		61.0619	30.2731	7.814211	0.127972	0.855112		103.6027	27.97709	10.17854	0.098246	0.187831
5	23.24924	34.46672	4.821747	0.207394	0.960523		70.77214	29.63218	8.412618	0.118869	0.878457		101.3915	28.07079	10.06934	0.099311	0.111223
6	12.95871	37.00519	3.599821	0.277792	0.962613		23.89553	34.34764	4.888305	0.20457	0.803012		84.46896	28.86383	9.1907	0.108806	0.402804
7	10.25103	38.02313	3.201723	0.312332	0.960079		42.43366	31.8537	6.514113	0.153513	0.888959		103.2564	27.99163	10.16152	0.098411	0.46114
8	13.60123	36.79502	3.687985	0.271151	0.972586		61.15716	30.26633	7.820304	0.127872	0.886766		104.7159	27.93068	10.23308	0.097722	0.172273
9	11.86077	37.38968	3.443946	0.290365	0.950594		31.2859	33.17732	5.59338	0.178783	0.850624		94.21863	28.38944	9.706628	0.103022	0.480387
10	33.12573	32.92915	5.755496	0.173747	0.962038		79.97667	29.10117	8.942968	0.11182	0.921954		103.7327	27.97165	10.18492	0.098184	0.176268
11	13.01028	36.98794	3.606977	0.27724	0.960103		43.49133	31.74678	6.594796	0.151635	0.817913		101.885	28.0497	10.09381	0.099071	0.122837
12	12.05716	37.31835	3.472342	0.28799	0.95915		57.79949	30.51156	7.602597	0.131534	0.885843		105.6306	27.89291	10.27767	0.097298	0.187556
13	12.73542	37.08067	3.568673	0.280216	0.922231		31.56659	33.13853	5.618415	0.177986	0.832937		89.08076	28.63296	9.438261	0.105952	0.342505
14	7.237043	39.53519	2.690175	0.371723	0.964996		102.5813	28.02012	10.12824	0.098734	0.891803		104.9837	27.91958	10.24616	0.097598	0.272612
15	10.9675	37.72973	3.311722	0.301958	0.959279		101.4079	28.07009	10.07015	0.099303	0.019551		97.67365	28.23303	9.882998	0.101184	0.119729
16	26.24508	33.94032	5.122995	0.195198	0.878076		41.05094	31.99757	6.407101	0.156077	0.000109		101.7606	28.05501	10.08765	0.099131	0.264989
17	12.6197	37.12031	3.552422	0.281498	0.962037		33.83077	32.83769	5.816422	0.171927	0.885886		99.07028	28.17137	9.953405	0.100468	0.390468
18	6.96648	39.70067	2.639409	0.378873	0.966274		49.36539	31.19658	7.026051	0.142327	0.909286		98.07552	28.2152	9.903309	0.100976	0.30972
19	9.934856	38.15919	3.151961	0.317263	0.950161		38.47895	32.27857	6.203141	0.161209	0.895624		93.92728	28.40289	9.691609	0.103182	0.272794
20	14.04596	36.65529	3.747794	0.266824	0.987649		114.8667	27.52886	10.71759	0.093305	0.892089		82.54956	28.96366	9.085679	0.110063	0.727646
21	9.629866	38.2946	3.103203	0.322248	0.96517		50.16859	31.12648	7.082979	0.141184	0.000233		100.948	28.08983	10.04729	0.099529	0.047697
22	16.21909	36.03054	4.027293	0.248306	0.959319		47.26756	31.38517	6.87514	0.145452	0.932338		106.1589	27.87124	10.30334	0.097056	0.212838
23	13.36202	36.87208	3.65541	0.273567	0.967199		57.55377	30.53007	7.58642	0.131814	0.861628		104.1975	27.95223	10.20772	0.097965	0.210125
24	8.656552	38.75735	2.942202	0.339881	0.957123		30.60408	33.27301	5.532096	0.180763	0.857473		100.4202	28.11259	10.02099	0.099791	0.471108
25	17.4539	35.71188	4.177787	0.239361	0.965136		46.16064	31.48809	6.794162	0.147185	0.893014		90.70624	28.55443	9.523983	0.104998	0.264116
26	18.7306	35.40529	4.327886	0.23106	0.956889		61.62155	30.23348	7.849939	0.12739	0.853959		102.353	28.0298	10.11696	0.098844	0.222555
27	17.59898	35.67593	4.195113	0.238373	0.951429		51.76776	30.99021	7.194982	0.138986	0.843811		104.6104	27.93506	10.22792	0.097772	0.17535
28	14.69265	36.4598	3.833099	0.260886	0.942376		103.986	27.96105	10.19735	0.098065	0.876252		104.5815	27.93626	10.22651	0.097785	0.042067
29	11.0938	37.68	3.330735	0.300234	0.964441		40.88793	32.01485	6.394367	0.156388	0.005683		104.0941	27.95654	10.20265	0.098014	0.268514
30	18.50583	35.45772	4.301841	0.232459	0.961807		27.79893	33.69052	5.272469	0.189664	0.885885		59.17682	30.40929	7.692647	0.129994	0.590232
31	15.90236	36.11619	3.987776	0.250766	0.964482		57.00585	30.57161	7.550222	0.132446	0.923254		108.891	27.76089	10.43508	0.095831	0.318049
32	15.23759	36.30164	3.903535	0.256178	0.934907		50.45149	31.10206	7.102921	0.140787	0.867688		103.0184	28.00165	10.1498	0.098524	0.380927
33	24.9847	34.15406	4.998469	0.200061	0.970375		51.19856	31.03823	7.155317	0.139756	0.806173		104.5525	27.93746	10.22509	0.097799	0.151209
34	17.96879	35.58562	4.238961	0.235907	0.966817		69.55037	29.70781	8.339686	0.119909	0.922048		103.2059	27.99376	10.15903	0.098435	0.279608

Sheet1

OUTPUT (Comparing Images)

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q
37	17.13303	35.79246	4.139207	0.241592	0.960878		112.6763	27.61248	10.61491	0.094207	0.023606		104.1273	27.95516	10.20428	0.097998	0.263883
38	7.784194	39.21867	2.790017	0.358421	0.974462		46.7094	31.43676	6.834428	0.146318	0.000217		103.6155	27.97656	10.17917	0.09824	0.275913
39	10.72002	37.82885	3.274144	0.305423	0.951348		42.42406	31.85468	6.513375	0.15353	0.870414		100.3386	28.11613	10.01691	0.099831	0.201708
40	10.22396	38.03461	3.197493	0.312745	0.948346		63.92854	30.07386	7.995532	0.12507	0.906266		103.4636	27.98293	10.1717	0.098312	0.588037
41	27.62816	33.71728	5.256249	0.19025	0.943976		65.41504	29.97403	8.087957	0.123641	0.822796		103.7636	27.97035	10.18644	0.09817	0.191556
42	19.18927	35.30022	4.380556	0.228282	0.937977		62.08552	30.2009	7.879437	0.126913	0.822713		104.2357	27.95064	10.20959	0.097947	0.231741
43	12.97771	36.99882	3.602459	0.277588	0.93759		46.61517	31.44553	6.82753	0.146466	0.812452		97.38011	28.2461	9.868136	0.101336	0.235452
44	8.387941	38.89425	2.896194	0.345281	0.969367		69.21776	29.72863	8.319721	0.120196	0.847349		84.39873	28.86744	9.186878	0.108851	0.286603
45	15.1061	36.33928	3.886656	0.257291	0.926104		45.54926	31.54599	6.749019	0.14817	0.180626		104.7653	27.92863	10.23549	0.097699	0.443317
46	6.201263	40.206	2.490234	0.401569	0.972815		28.76925	33.54152	5.363697	0.186439	0.840532		106.3596	27.86304	10.31308	0.096964	0.293704
47	29.93919	33.3684	5.471671	0.18276	0.96265		61.30945	30.25553	7.830035	0.127713	0.913294		104.2633	27.94949	10.21094	0.097934	0.195012
48	14.63595	36.47659	3.825696	0.26139	0.951744		62.51602	30.17089	7.906707	0.126475	0.802618		92.25142	28.48107	9.60476	0.104115	0.115522
49	14.4997	36.51721	3.807847	0.262616	0.962974		50.82333	31.07017	7.129048	0.140271	0.273832		104.4833	27.94034	10.22171	0.097831	0.126642
50	23.66988	34.38884	4.86517	0.205543	0.972703		65.98504	29.93635	8.123118	0.123105	0.872563		103.7764	27.96982	10.18707	0.098164	0.226273
51	14.28096	36.58323	3.779016	0.264619	0.959712		49.28164	31.20395	7.020088	0.142448	0.77881		102.0543	28.04249	10.10219	0.098988	0.20607
52	18.2176	35.52589	4.268208	0.23429	0.95696		40.43841	32.06286	6.35912	0.157254	0.870968		86.88933	28.74114	9.321444	0.10728	0.498071
53	11.34248	37.58372	3.367859	0.296925	0.929867		112.6297	27.61427	10.61272	0.094227	0.855972		107.1898	27.82927	10.35325	0.096588	0.37924
54	14.26381	36.58845	3.776746	0.264778	0.957355		44.99825	31.59885	6.708074	0.149074	0.005055		97.03796	28.26139	9.850785	0.101515	0.179465
55	33.17377	32.92286	5.759667	0.173621	0.856936		44.06291	31.69007	6.63799	0.150648	0.88617		96.32779	28.29329	9.814672	0.101888	0.383153
56	12.98577	36.99613	3.603577	0.277502	0.947844		47.56366	31.35805	6.896641	0.144998	0.895819		101.1403	28.08156	10.05685	0.099435	0.386896
57	14.95669	36.38245	3.867388	0.258572	0.963068		53.24434	30.86807	7.296872	0.137045	0.803357		104.0276	27.95932	10.19939	0.098045	0.168784
58	14.43827	36.53565	3.799772	0.263174	0.966123		61.16654	30.26566	7.820904	0.127862	0.851345		102.3253	28.03097	10.1156	0.098857	0.211873
59	17.46859	35.70822	4.179545	0.239261	0.957438		49.56506	31.17905	7.040246	0.14204	0.822668		103.3972	27.98572	10.16844	0.098343	0.147728
60	10.05298	38.10785	3.170644	0.315393	0.96016		40.6623	32.03888	6.3767	0.156821	0.889902		99.34968	28.15914	9.967431	0.100327	0.532494
61	8.56723	38.8024	2.926983	0.341649	0.958508		109.356	27.74238	10.45734	0.095627	0.871446		103.8428	27.96704	10.19033	0.098132	0.108277
62	17.48359	35.7045	4.181338	0.239158	0.92935		47.81818	31.33487	6.915069	0.144612	0.000449		100.4723	28.11034	10.02359	0.099765	0.173183
63	5.944588	40.38959	2.438152	0.410147	0.972038		55.08275	30.72065	7.421775	0.134739	0.849781		72.91624	29.50256	8.539101	0.117108	0.385828
64	18.84298	35.37931	4.34085	0.23037	0.965118		52.99939	30.8881	7.280068	0.137361	6.76E-05		98.23319	28.20822	9.911266	0.100895	0.177274
65	15.53559	36.21752	3.941522	0.253709	0.930808		47.86219	31.33088	6.918251	0.144545	0.882493		99.1429	28.16819	9.957053	0.100431	0.275575
66	12.41578	37.19106	3.523603	0.2838	0.960106		46.73508	31.43437	6.836306	0.146278	0.857682		102.4341	28.02636	10.12097	0.098805	0.329322
67	11.31123	37.5957	3.363218	0.297334	0.962907		35.44834	32.63484	5.953851	0.167959	0.88082		105.768	27.88726	10.28436	0.097235	0.409551
68	11.48947	37.5278	3.389612	0.295019	0.95913		100.6268	28.10367	10.03129	0.099688	0.872718		106.5203	27.85648	10.32087	0.096891	0.289398
69	20.93853	34.92134	4.575864	0.218538	0.96934		94.21461	28.38962	9.706421	0.103025	0.000584		105.3514	27.9044	10.26408	0.097427	0.122196
70	13.25293	36.90768	3.640458	0.274691	0.953336		100.061	28.12815	10.00305	0.09997	0.000235		105.6708	27.89125	10.27963	0.09728	0.081287

Sheet1

OUTPUT (Comparing Images)

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q
70	13.25293	36.90768	3.640458	0.274691	0.953336		100.061	28.12815	10.00305	0.09997	0.000235		105.6708	27.89125	10.27963	0.09728	0.081287
71	10.38974	37.96476	3.223312	0.31024	0.961466		50.36666	31.10937	7.096947	0.140906	0.000685		104.5799	27.93632	10.22643	0.097786	0.225872
72	24.06792	34.31642	4.905906	0.203836	0.968326		55.77025	30.66678	7.467948	0.133906	0.832058		103.9255	27.96358	10.19439	0.098093	0.136654
73	16.71438	35.8999	4.088323	0.244599	0.960661		36.35224	32.52549	6.029282	0.165857	0.8494		83.55501	28.91108	9.140843	0.109399	0.467534
74	20.87076	34.93542	4.568453	0.218892	0.947157		59.37083	30.39507	7.705247	0.129782	0.906322		101.0523	28.08534	10.05248	0.099478	0.157932
75	13.0616	36.97084	3.614083	0.276695	0.97323		47.9964	31.31872	6.927944	0.144343	0.880272		101.6677	28.05898	10.08304	0.099176	0.267434
76	11.23459	37.62523	3.351803	0.298347	0.962588		59.94737	30.3531	7.742569	0.129156	0.870933		104.1141	27.95571	10.20363	0.098004	0.309879
77	15.10725	36.33895	3.886805	0.257281	0.947093		16.39251	35.98435	4.048766	0.246989	0.836303		85.67835	28.80209	9.25626	0.108035	0.61545
78	13.00282	36.99043	3.605942	0.27732	0.964252		52.90081	30.89618	7.273294	0.137489	0.942972		105.5275	27.89715	10.27266	0.097346	0.249562
79	11.53649	37.51007	3.396541	0.294417	0.966995		51.74368	30.99223	7.193308	0.139018	0.869095		103.9094	27.96426	10.1936	0.098101	0.17186
80	9.105563	38.53774	3.017543	0.331395	0.962944		78.36597	29.18953	8.852456	0.112963	0.856939		95.84889	28.31493	9.790245	0.102142	0.183234
81	6.166056	40.23073	2.483154	0.402714	0.966591		47.9964	30.82782	6.028482	0.094657	0.210745		102.1804	28.03713	10.10843	0.098927	0.483592
82	17.4528	35.71215	4.177655	0.239369	0.959769		46.19664	31.4847	6.796811	0.147128	0.841724		96.40857	28.28965	9.818787	0.101846	0.346304
83	24.80906	34.1847	4.980869	0.200768	0.960203		63.08606	30.13147	7.942673	0.125902	0.825451		105.116	27.91412	10.25261	0.097536	0.187533
84	13.75834	36.74514	3.709224	0.269598	0.9595		43.1713	31.77885	6.570487	0.152196	0.807669		105.1272	27.91365	10.25315	0.097531	0.358003
85	20.51678	35.00971	4.529546	0.220773	0.967567		65.61118	29.96103	8.100073	0.123456	0.84877		105.8362	27.88446	10.28767	0.097204	0.112233
86	7.088378	39.62534	2.662401	0.375601	0.96348		52.95822	30.89147	7.27724	0.137415	0.797937		102.0424	28.043	10.1016	0.098994	0.150495
87	9.901836	38.17365	3.146718	0.317791	0.956006		53.34501	30.85987	7.303767	0.136916	0.855434		101.0507	28.08541	10.0524	0.099479	0.450296
88	11.591	37.48959	3.404556	0.293724	0.97099		81.21032	29.03469	9.011677	0.110967	0.001389		93.9866	28.40014	9.694669	0.103149	0.327108
89	24.5855	34.22401	4.958377	0.201679	0.965669		87.65383	28.70309	9.362362	0.106811	0.00113		104.6932	27.93162	10.23197	0.097733	0.204998
90	9.00716	38.58492	3.001193	0.333201	0.951854		19.77036	35.17066	4.446387	0.224902	0.932079		110.2464	27.70716	10.49983	0.09524	0.480011
91	10.11297	38.08202	3.18009	0.314457	0.954088		55.86155	30.65967	7.474058	0.133796	0.84553		105.6759	27.89104	10.27988	0.097277	0.275911
92	17.73745	35.64189	4.211585	0.23744	0.958585		64.56124	30.03108	8.035001	0.124455	0.863972		102.0902	28.04096	10.10397	0.098971	0.082024
93	11.44991	37.54278	3.383772	0.295528	0.940749		35.85078	32.58582	5.987552	0.167013	0.85074		95.63384	28.32469	9.779255	0.102257	0.22755
94	14.3219	36.5708	3.784429	0.264241	0.951362		87.37028	28.71717	9.347207	0.106984	0.048493		99.81323	28.13892	9.990657	0.100094	0.29058
95	24.13064	34.30512	4.912295	0.203571	0.966058		62.60713	30.16457	7.912467	0.126383	0.840754		100.4333	28.11202	10.02164	0.099784	0.199594
96	21.53215	34.79993	4.640275	0.215504	0.957422		47.34663	31.37791	6.880889	0.14533	0.848232		102.5519	28.02137	10.12679	0.098748	0.336464
97	11.53195	37.51178	3.395872	0.294475	0.964206		91.7068	28.50679	9.576367	0.104424	0.069891		99.2942	28.16157	9.964647	0.100355	0.188912
98	36.80266	32.47201	6.066519	0.164839	0.961771		62.18312	30.19408	7.885628	0.126813	0.867561		105.0022	27.91882	10.24706	0.097589	0.124432
99	16.72497	35.89715	4.089617	0.244522	0.96938		47.92799	31.32491	6.923005	0.144446	0.882461		102.2287	28.03508	10.11082	0.098904	0.221962
100	19.24915	35.28669	4.387386	0.227926	0.962007		51.68551	30.99712	7.189263	0.139096	0.866848		102.0448	28.0429	10.10172	0.098993	0.270987
101	23.79134	34.36661	4.877637	0.205017	0.968447		72.92033	29.50232	8.53934	0.117105	0.793729		104.793	27.92748	10.23684	0.097686	0.217457
102	10.80432	37.79483	3.286992	0.304229	0.957191		60.2623	30.33035	7.76288	0.128818	0.879156		101.933	28.04765	10.09619	0.099047	0.293611
103	15.41564	36.25119	3.926275	0.254694	0.95388		44.71116	31.62664	6.68664	0.149552	0.864671		103.621	27.97632	10.17944	0.098237	0.242516

Sheet1

Ready

OUTPUT (Comparing Images)

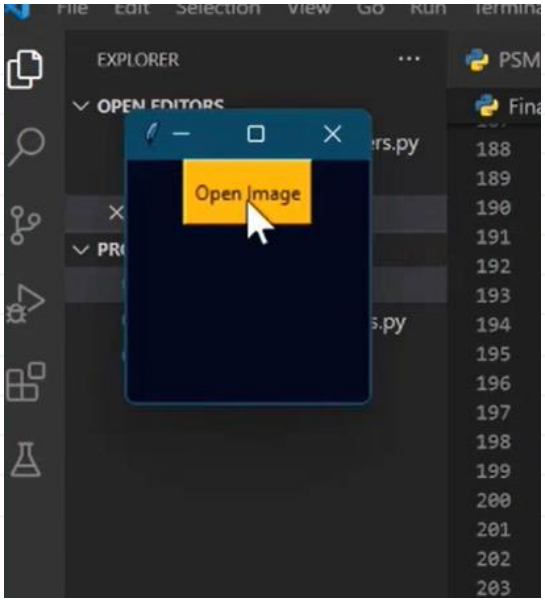
	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q
1969	23.56709	34.40774	4.854595	0.20599	0.985271		38.02386	32.33024	6.166349	0.162171	0.877921		105.0572	27.91655	10.24974	0.097563	0.033745
1970	7.856974	39.17825	2.803029	0.356757	0.962837		65.10318	29.99478	8.068654	0.123936	0.845		102.7019	28.01502	10.13419	0.098676	0.136653
1971	19.50618	35.22908	4.416581	0.22642	0.951874		58.36814	30.46905	7.639904	0.130892	0.851222		103.4756	27.98243	10.17229	0.098306	0.239541
1972	21.88455	34.72943	4.678092	0.213762	0.970328		52.75275	30.90835	7.263109	0.137682	0.89381		104.468	27.94097	10.22096	0.097838	0.105401
1973	10.21161	38.03986	3.195561	0.312934	0.969635		29.70104	33.40309	5.449866	0.183491	0.874435		103.6693	27.9743	10.18181	0.098214	0.154696
1974	13.88251	36.70612	3.725924	0.26839	0.964992		31.09348	33.20411	5.576153	0.179335	0.875082		104.8538	27.92496	10.23982	0.097658	0.215319
1975	16.28906	36.01184	4.03597	0.247772	0.96761		89.01917	28.63597	9.434997	0.105988	0.000783		105.6464	27.89226	10.27844	0.097291	0.551369
1976	29.33504	33.45694	5.416183	0.184632	0.963579		65.4272	29.97322	8.088708	0.123629	0.896816		103.8645	27.96613	10.19139	0.098122	0.179585
1977	15.769	36.15276	3.971019	0.251825	0.969707		60.23923	30.33201	7.761393	0.128843	0.788437		105.5074	27.89798	10.27168	0.097355	0.166632
1978	8.335271	38.92161	2.887087	0.34637	0.95807		110.8609	27.68302	10.52905	0.094975	0.000167		101.8361	28.05178	10.09139	0.099094	0.102487
1979	7.405933	39.43501	2.721384	0.36746	0.944918		51.74509	30.99211	7.193406	0.139016	0.841476		90.9702	28.54181	9.53783	0.104846	0.458828
1980	10.68761	37.842	3.269191	0.305886	0.955774		93.99346	28.39983	9.695022	0.103146	0.052839		78.42229	29.18641	8.855636	0.112922	0.360737
1981	24.47601	34.2434	4.947324	0.202129	0.954677		93.86376	28.40582	9.688331	0.103217	0.038346		91.44029	28.51943	9.562442	0.104576	0.177483
1982	36.75893	32.47717	6.062914	0.164937	0.939282		64.4329	30.03973	8.027011	0.124579	0.9092		99.49179	28.15293	9.974557	0.100255	0.105824
1983	19.78701	35.167	4.448259	0.224807	0.964754		39.08977	32.21017	6.252181	0.159944	0.913168		99.47557	28.15364	9.973744	0.100263	0.321601
1984	21.47434	34.8116	4.634042	0.215794	0.962545		52.99566	30.8884	7.279812	0.137366	0.851406		104.9808	27.9197	10.24601	0.097599	0.104842
1985	12.94228	37.0107	3.597538	0.277968	0.965584		23.9925	34.33005	4.898214	0.204156	0.922047		105.0271	27.91779	10.24827	0.097577	0.305313
1986	9.104388	38.5383	3.017348	0.331417	0.961862		42.69138	31.8274	6.533864	0.153049	0.845375		107.6976	27.80874	10.37775	0.09636	0.358918
1987	11.86974	37.38639	3.445249	0.290255	0.946615		71.8789	29.56479	8.478143	0.11795	0.803412		95.40819	28.33495	9.767711	0.102378	0.177406
1988	17.57798	35.68111	4.192611	0.238515	0.96773		82.96289	28.94196	9.108397	0.109789	0.01628		103.8238	27.96784	10.1894	0.098141	0.081164
1989	13.99285	36.67174	3.740702	0.26733	0.958645		49.49167	31.18548	7.035032	0.142146	0.86789		94.05539	28.39697	9.698216	0.103112	0.451796
1990	16.63595	35.92033	4.07872	0.245175	0.960489		48.0439	31.31442	6.931371	0.144272	0.874304		101.7503	28.05545	10.08714	0.099136	0.165958
1991	9.883998	38.18148	3.143883	0.318078	0.96808		42.88654	31.80759	6.548782	0.1527	0.800979		107.2863	27.82536	10.35791	0.096545	0.36169
1992	19.0021	35.34279	4.35914	0.229403	0.937617		98.39397	28.20112	9.919374	0.100813	0.013324		102.4785	28.02448	10.12316	0.098783	0.430035
1993	20.34817	35.04555	4.510895	0.221686	0.961393		66.58861	29.8968	8.160184	0.122546	0.828441		105.3857	27.90299	10.26575	0.097411	0.232796
1994	7.651632	39.29326	2.766158	0.361512	0.951022		21.80732	34.74478	4.669831	0.214141	0.000409		100.4468	28.11144	10.02231	0.099777	0.180809
1995	14.50553	36.51547	3.808613	0.262563	0.93846		48.08559	31.31065	6.934378	0.144209	0.884485		102.7404	28.01339	10.13609	0.098657	0.196252
1996	11.48318	37.53018	3.388684	0.2951	0.962808		61.25548	30.25935	7.826588	0.124276	0.890713		102.8351	28.00939	10.14076	0.098612	0.334068
1997	12.94895	37.00846	3.598464	0.277896	0.957803		93.86375	31.69589	8.478046	0.042064	0.022343		93.47229	28.42397	9.668107	0.103433	0.323134
1998	21.7024	34.76573	4.658583	0.214658	0.964246		38.02386	34.58484	7.417485	0.757154	0.822714		100.1513	28.12424	10.00756	0.099924	0.145648
1999	14.44823	36.53266	3.801082	0.263083	0.962726		68.92153	33.4795	6.15481	0.287576	0.183427		103.9288	27.96345	10.19455	0.098092	0.244613
2000	7.284465	39.50683	2.698975	0.370511	0.963527		65.10318	30.58543	7.4842	0.095363	0.005054		58.28009	30.4756	7.63414	0.130991	0.702574
2001	18.87942	35.37092	4.345045	0.230147	0.963102		46.87596	28.56846	6.414542	0.528258	0.842424		97.4281	28.24396	9.870567	0.101311	0.348785
2002	15.81323	36.1406	3.976586	0.251472	0.967716		45.22545	27.15152	9.478689	0.088991	0.004234		105.2851	27.90713	10.26085	0.097458	0.069778

Sheet1

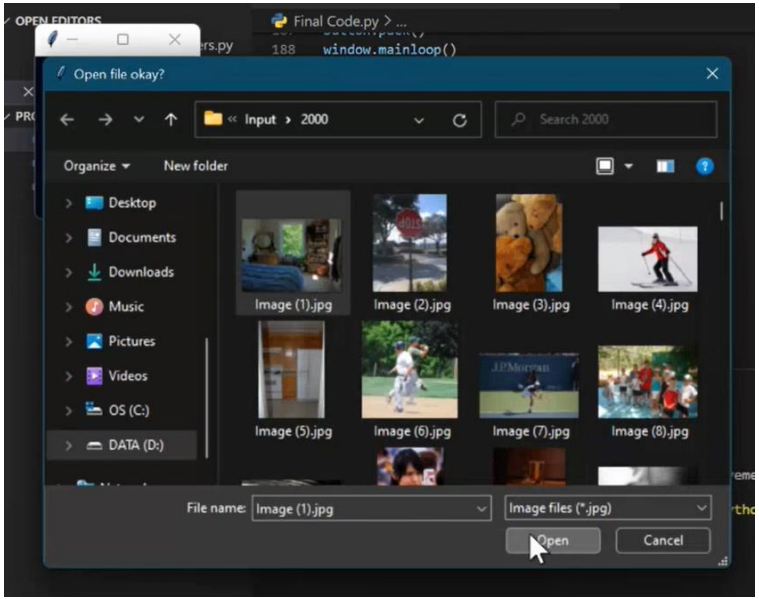
Ready

TESTING

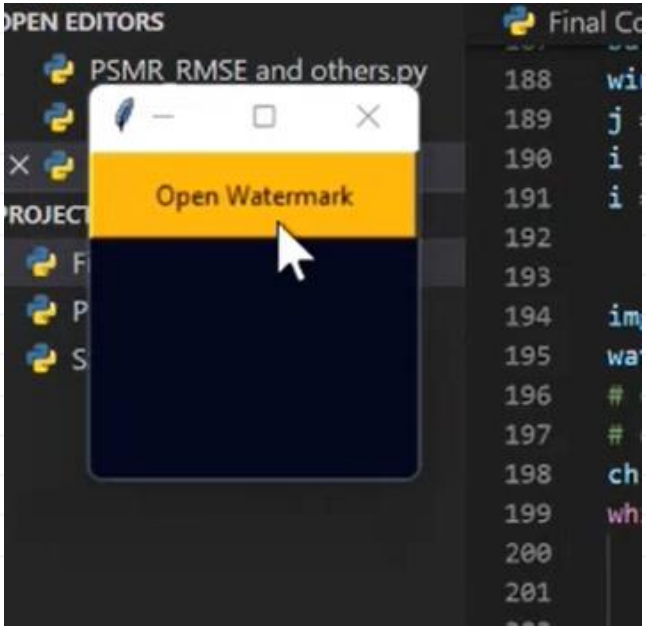
Step 1


EXPLORER
OPEN EDITORS
Open Image
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203

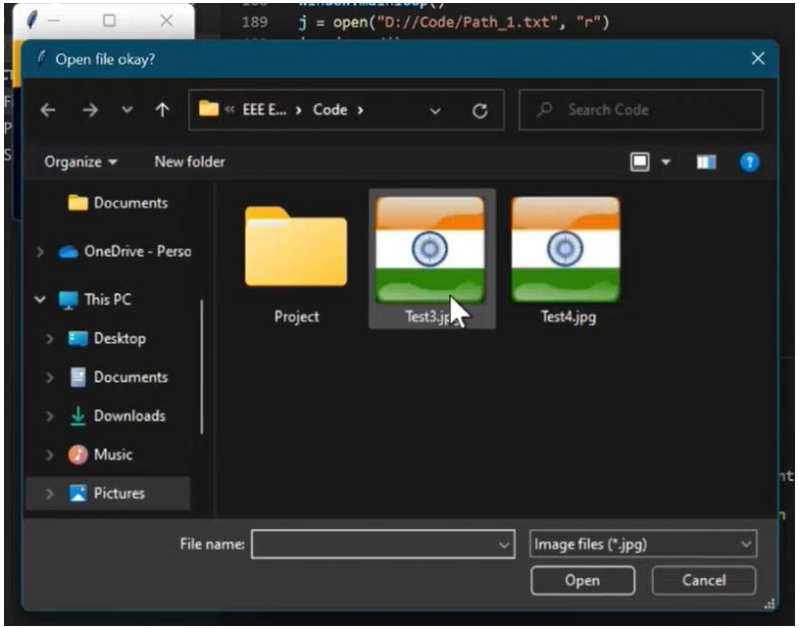
Step 2


Final Code.py > ...
188 window.mainloop()
Open file okay?
Input › 2000
Search 2000
Organize
New folder
Desktop
Documents
Downloads
Music
Pictures
Videos
OS (C:)
DATA (D:)
Image (1).jpg
Image (2).jpg
Image (3).jpg
Image (4).jpg
Image (5).jpg
Image (6).jpg
Image (7).jpg
Image (8).jpg
File name: Image (1).jpg
Image files (*.jpg)
Open
Cancel

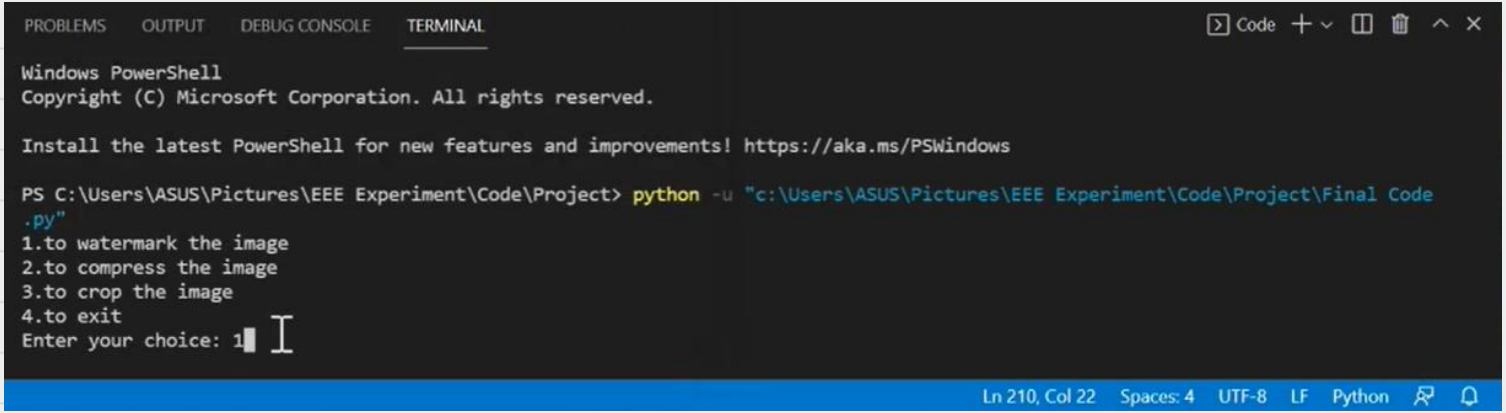
Step 3

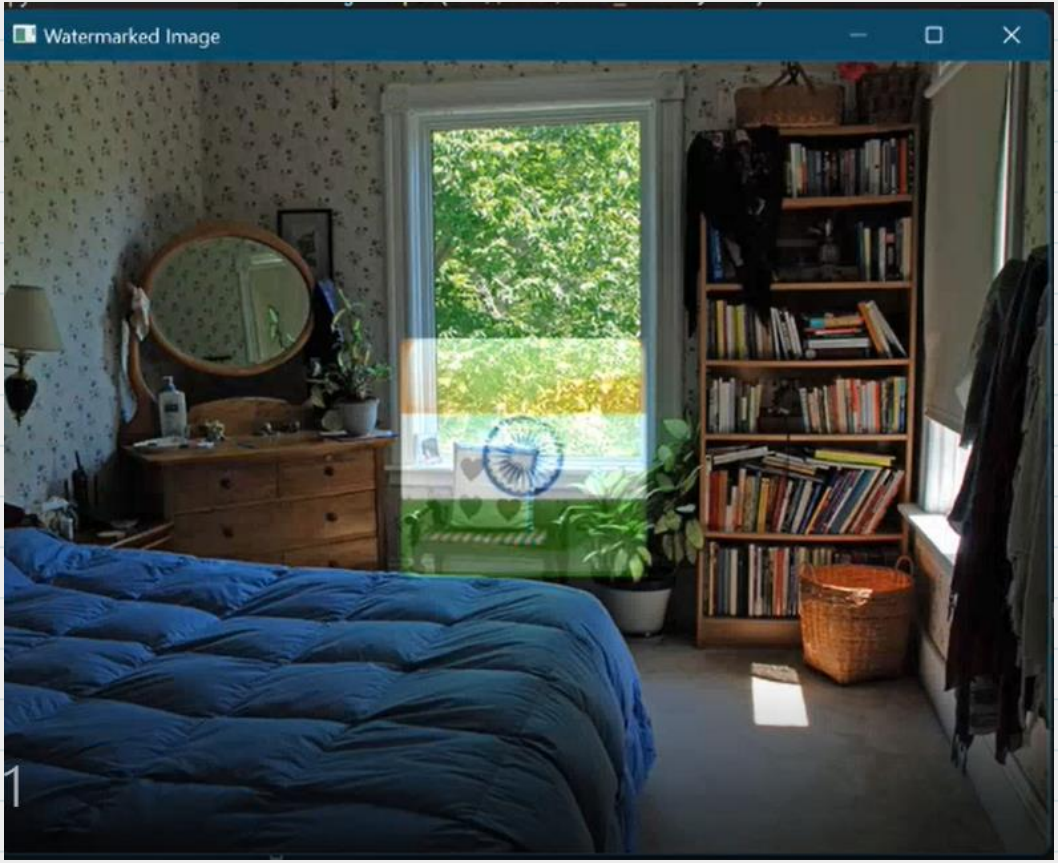

PEN EDITORS
PSMR_RMSE and others.py
Open Watermark
Final Co
188
189
190
191
192
193
194
195
196
197
198
199
200
201

Step 4


189 j = open("D://Code/Path_1.txt", "r")
Open file okay?
EEE E... › Code ›
Search Code
Organize
New folder
Documents
OneDrive - Perso
This PC
Desktop
Documents
Downloads
Music
Pictures
Project
Test3.jpg
Test4.jpg
File name:
Image files (*.jpg)
Open
Cancel

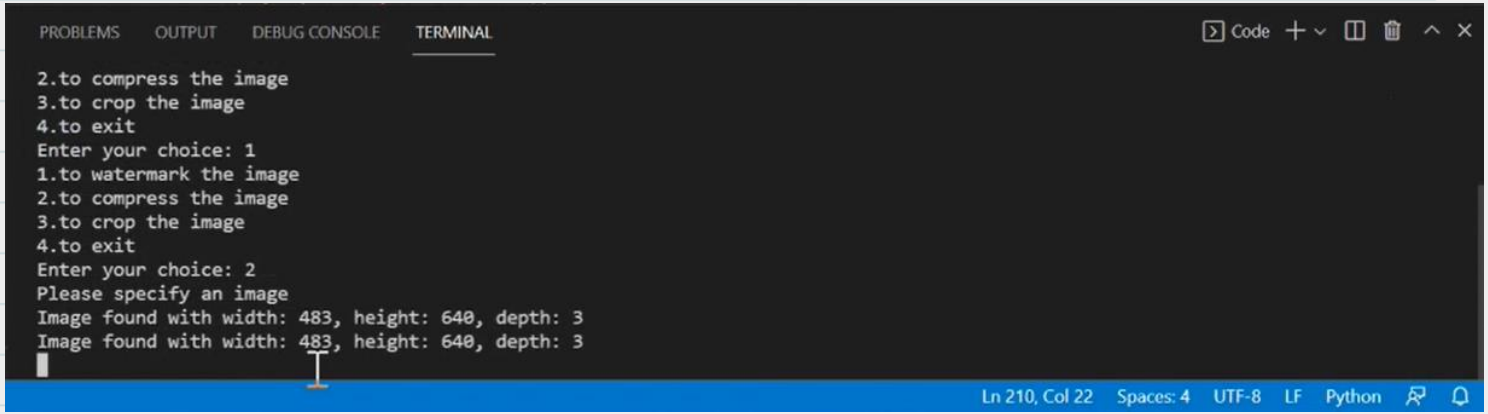
Step 5

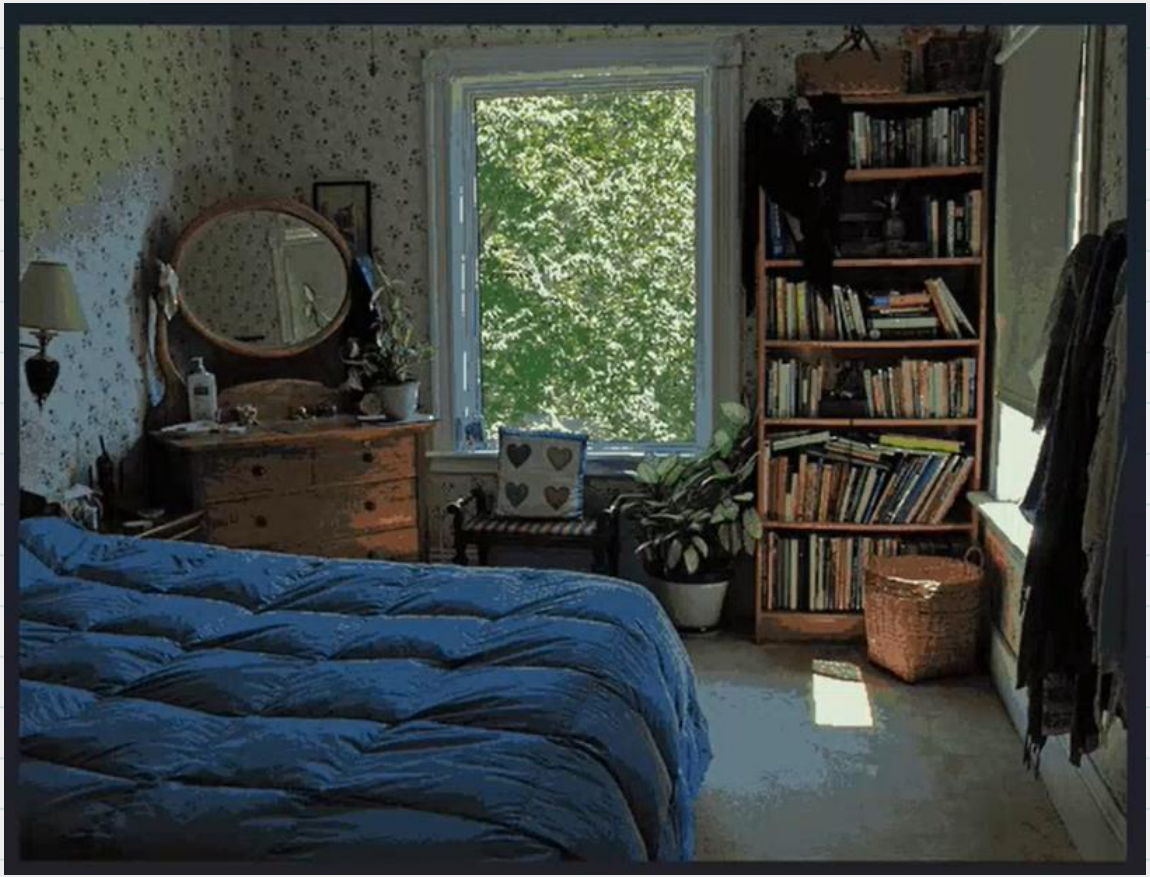

PROBLEMS
OUTPUT
DEBUG CONSOLE
TERMINAL
Code
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.
Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows
PS C:\Users\ASUS\Pictures\EEE Experiment\Code\Project> python -u "c:\Users\ASUS\Pictures\EEE Experiment\Code\Project\Final Code.py"
1.to watermark the image
2.to compress the image
3.to crop the image
4.to exit
Enter your choice: 1
Ln 210, Col 22
Spaces: 4
UTF-8
LF
Python



Step 6

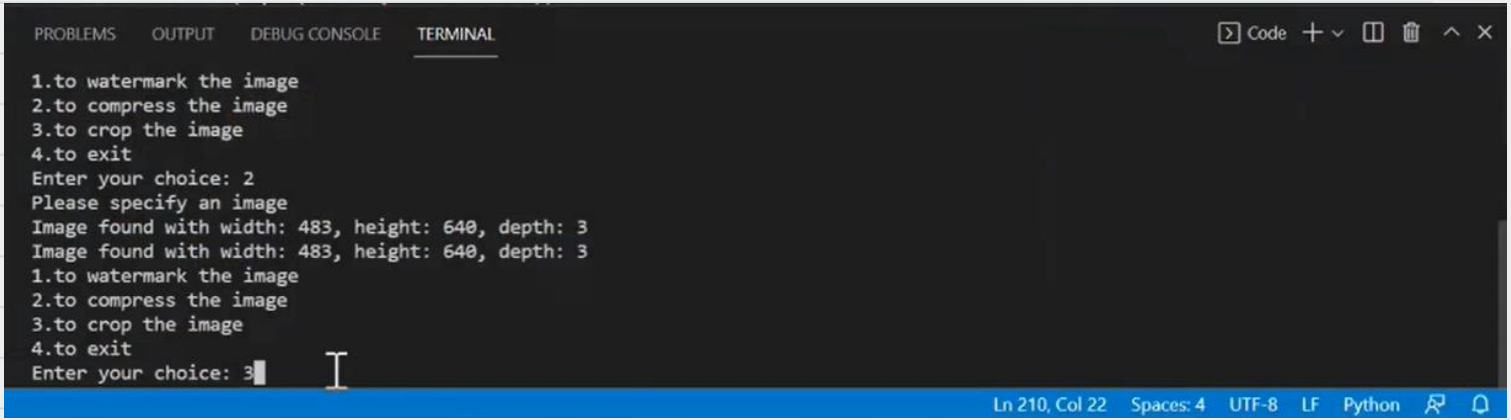
Step 7

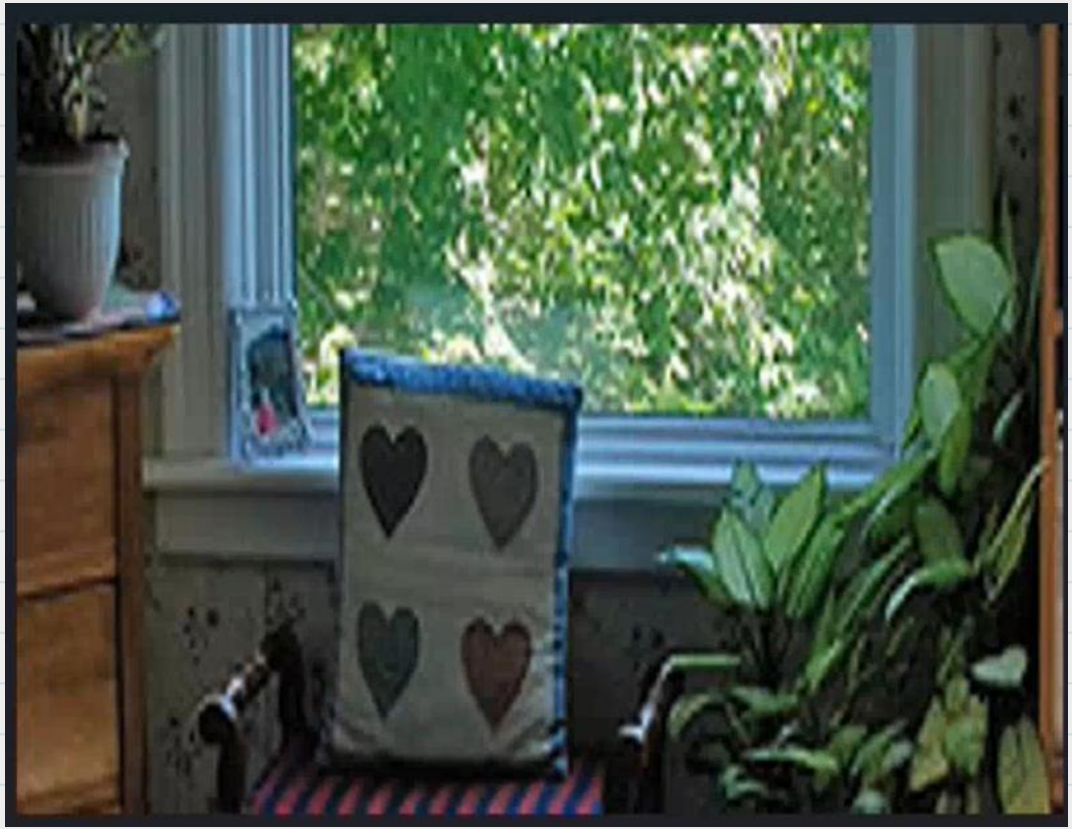

PROBLEMS
OUTPUT
DEBUG CONSOLE
TERMINAL
Code
2.to compress the image
3.to crop the image
4.to exit
Enter your choice: 1
1.to watermark the image
2.to compress the image
3.to crop the image
4.to exit
Enter your choice: 2
Please specify an image
Image found with width: 483, height: 640, depth: 3
Image found with width: 483, height: 640, depth: 3
Ln 210, Col 22
Spaces: 4
UTF-8
LF
Python



Step 8

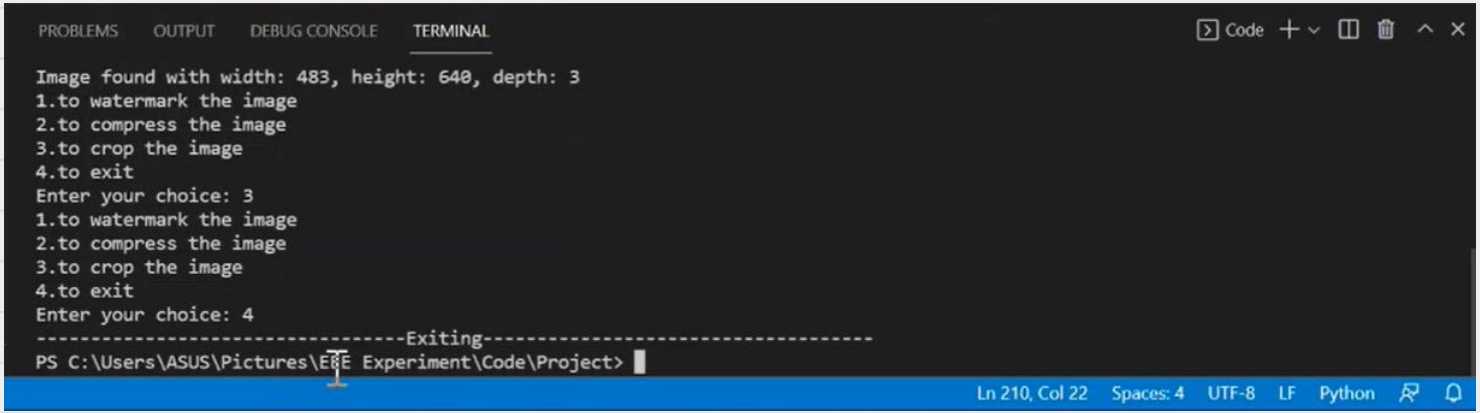
Step 9

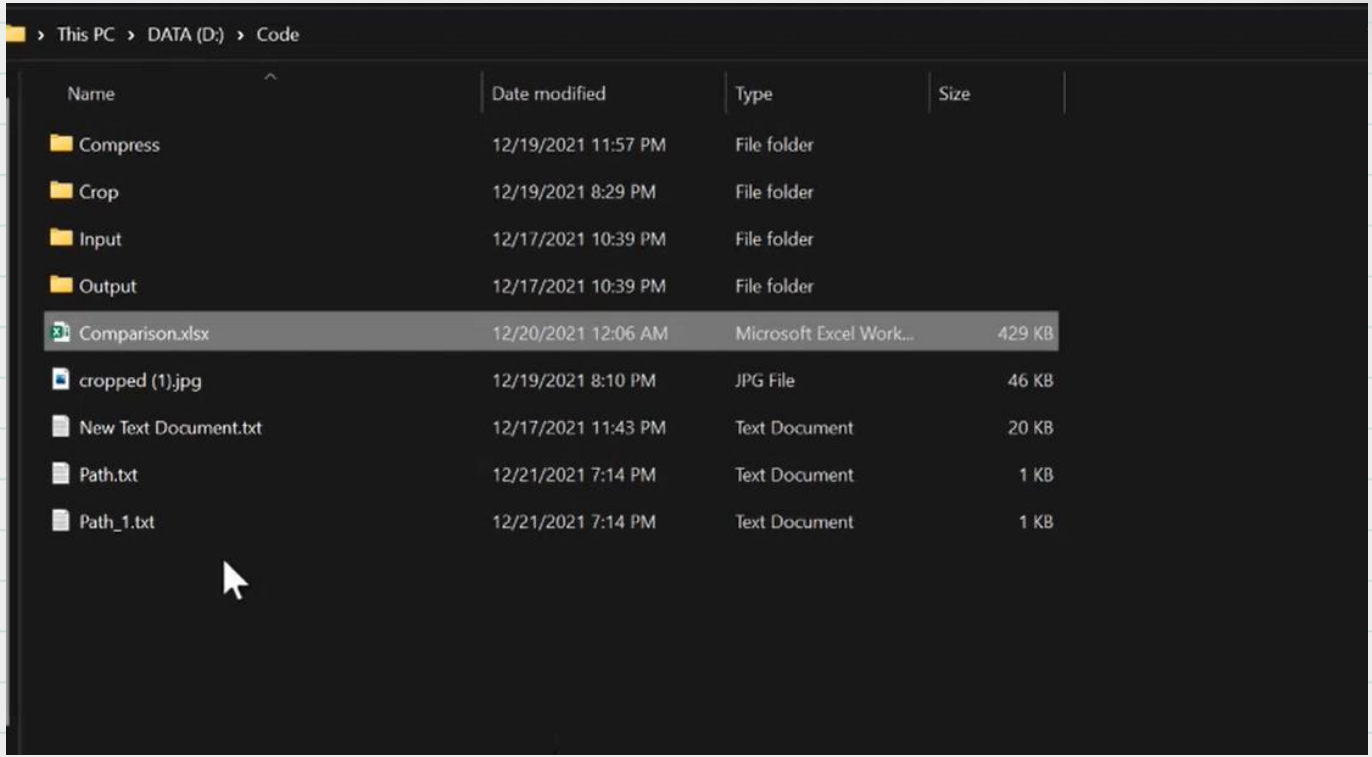

PROBLEMS
OUTPUT
DEBUG CONSOLE
TERMINAL
Code
1.to watermark the image
2.to compress the image
3.to crop the image
4.to exit
Enter your choice: 2
Please specify an image
Image found with width: 483, height: 640, depth: 3
Image found with width: 483, height: 640, depth: 3
1.to watermark the image
2.to compress the image
3.to crop the image
4.to exit
Enter your choice: 3
Ln 210, Col 22
Spaces: 4
UTF-8
LF
Python

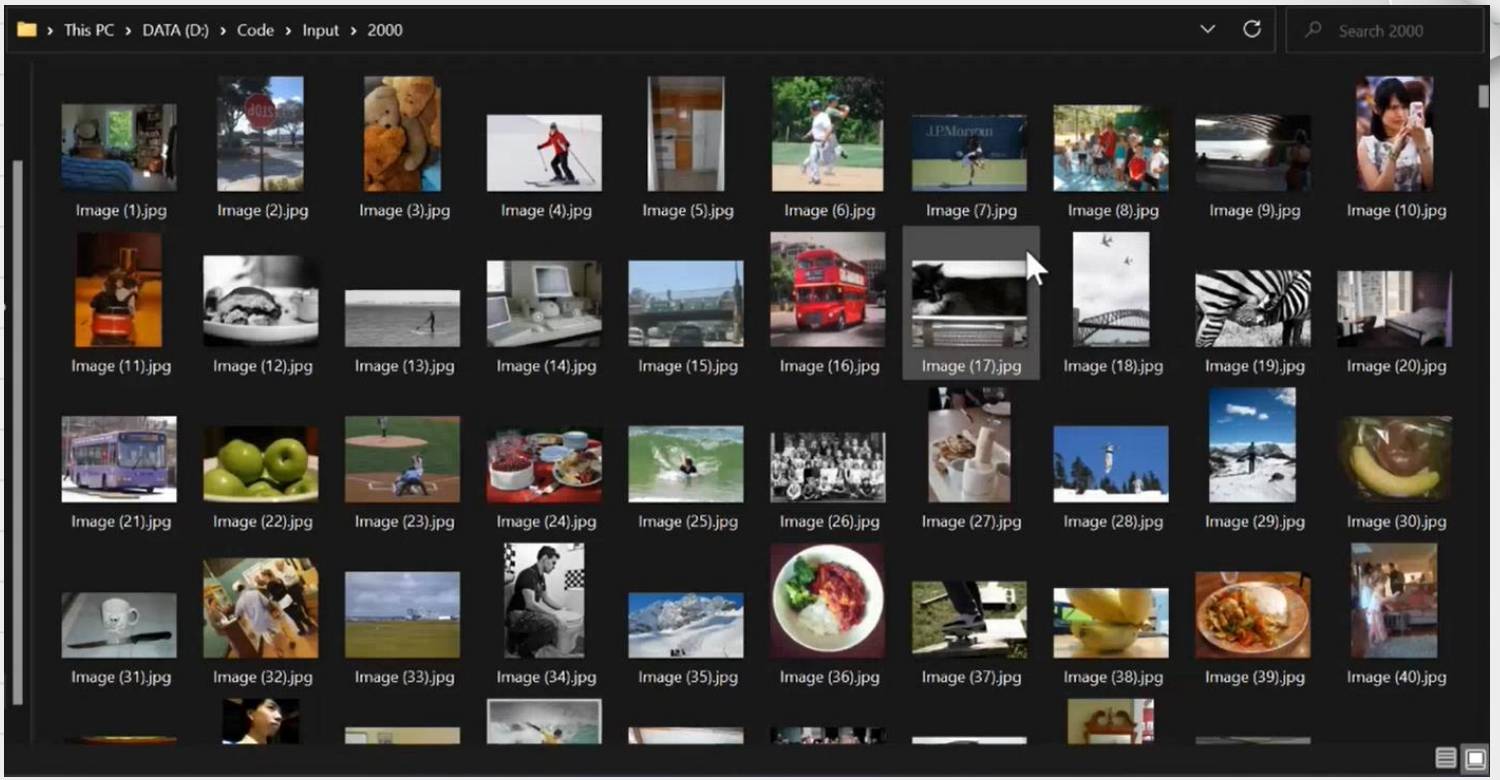


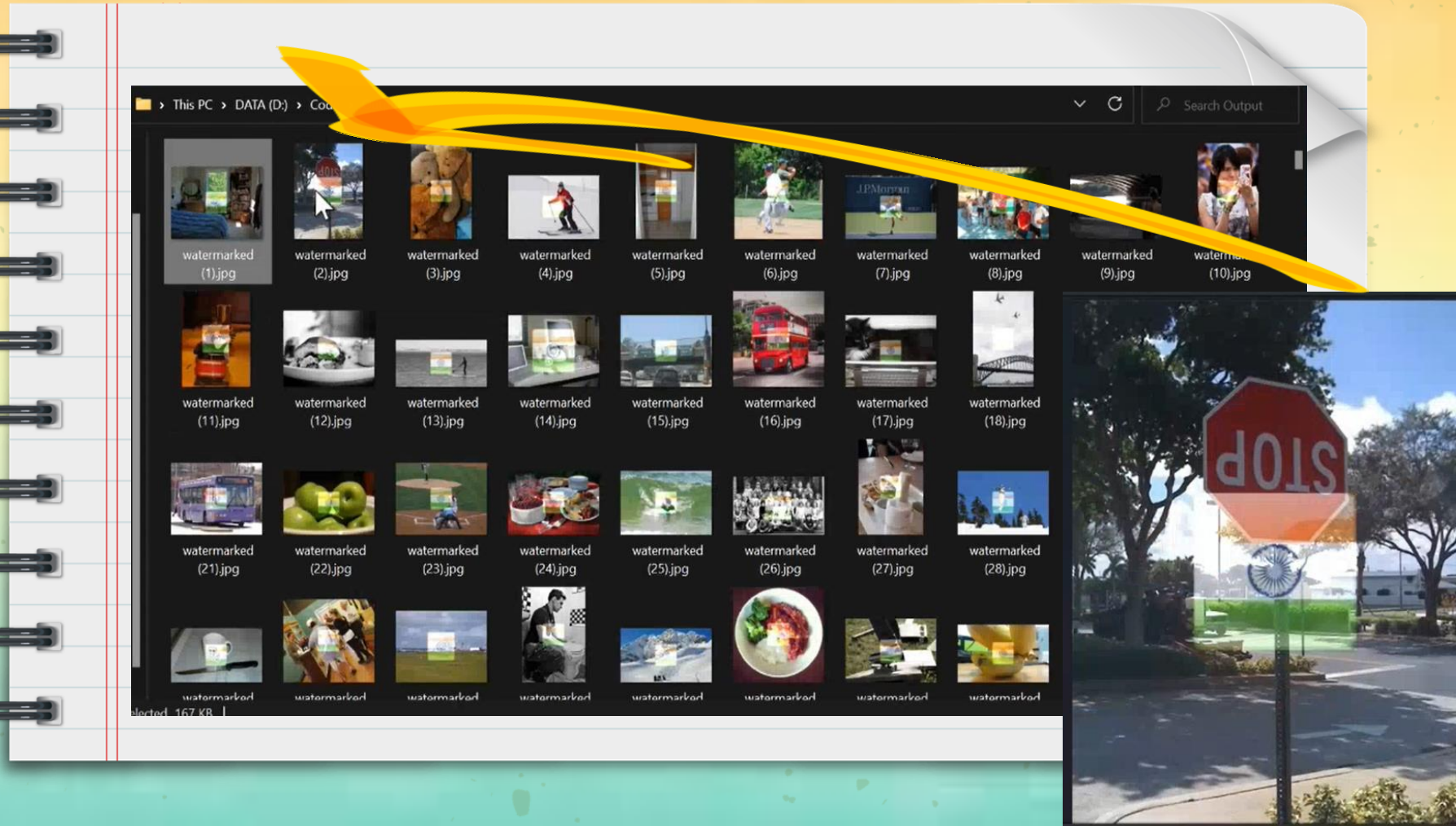
Step 10

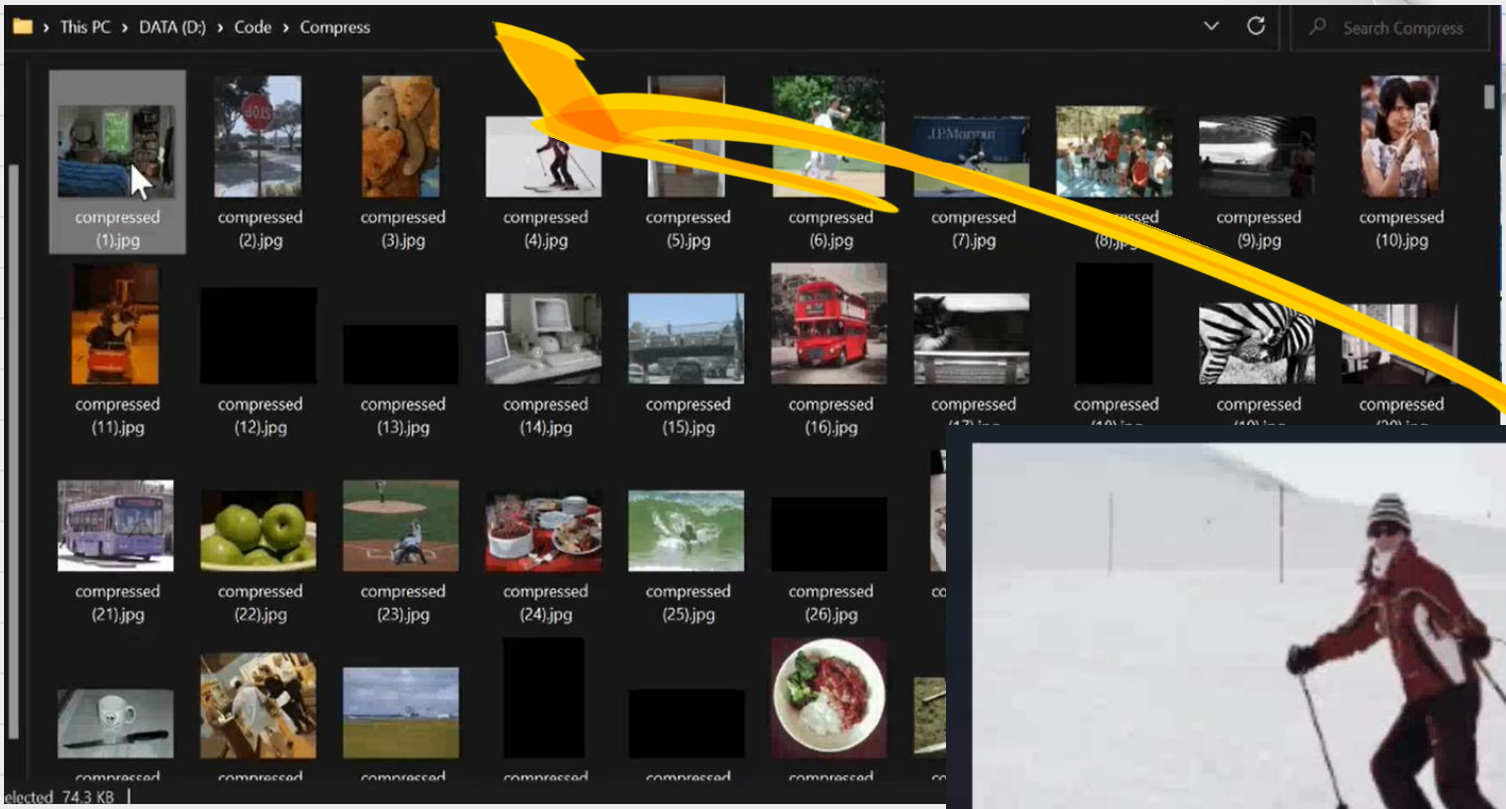
Step 11

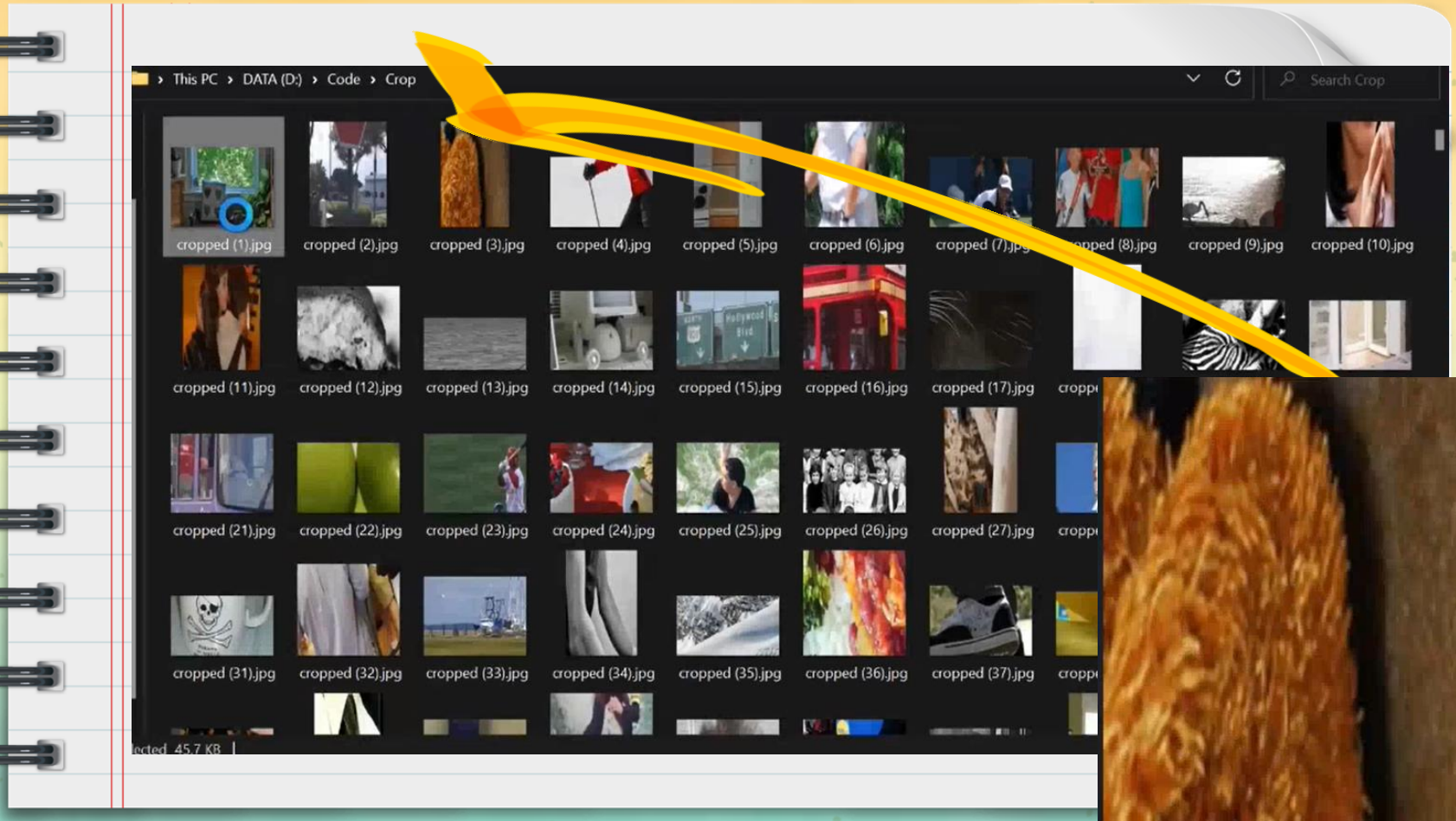

PROBLEMS
OUTPUT
DEBUG CONSOLE
TERMINAL
Code
Image found with width: 483, height: 640, depth: 3
1.to watermark the image
2.to compress the image
3.to crop the image
4.to exit
Enter your choice: 3
1.to watermark the image
2.to compress the image
3.to crop the image
4.to exit
Enter your choice: 4
----------Exiting----------
PS C:\Users\ASUS\Pictures\EBE Experiment\Code\Project>
Ln 210, Col 22
Spaces: 4
UTF-8
LF
Python

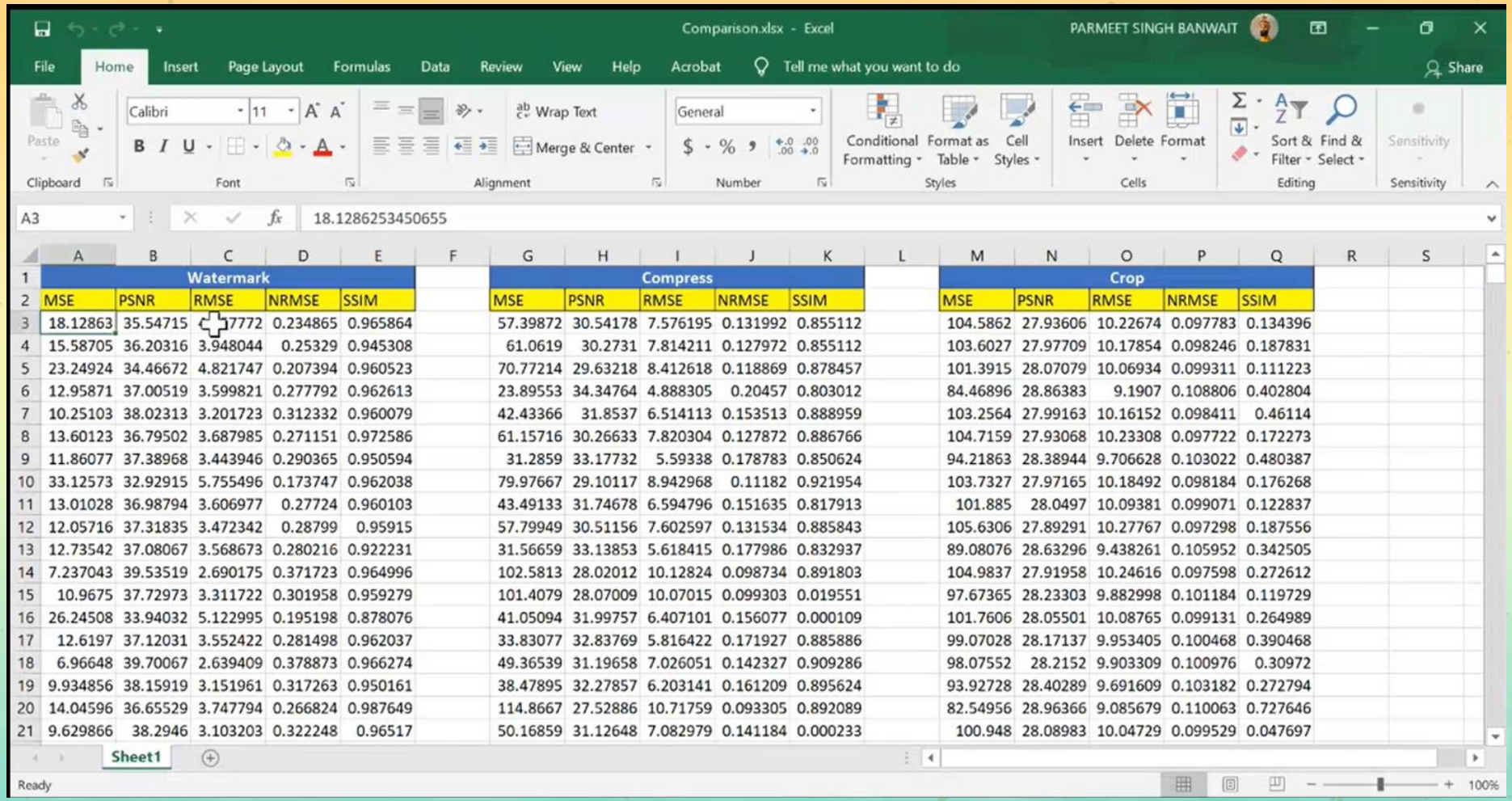

This PC › DATA (D:) › Code
Name
Date modified
Type
Size
Compress
12/19/2021 11:57 PM
File folder
Crop
12/19/2021 8:29 PM
File folder
Input
12/17/2021 10:39 PM
File folder
Output
12/17/2021 10:39 PM
File folder
Comparison.xlsx
12/20/2021 12:06 AM
Microsoft Excel Work...
429 KB
cropped (1).jpg
12/19/2021 8:10 PM
JPG File
46 KB
New Text Document.txt
12/17/2021 11:43 PM
Text Document
20 KB
Path.txt
12/21/2021 7:14 PM
Text Document
1 KB
Path_1.txt
12/21/2021 7:14 PM
Text Document
1 KB


This PC › DATA (D:) › Code › Input › 2000
Search 2000
Image (1).jpg
Image (2).jpg
Image (3).jpg
Image (4).jpg
Image (5).jpg
Image (6).jpg
Image (7).jpg
Image (8).jpg
Image (9).jpg
Image (10).jpg
Image (11).jpg
Image (12).jpg
Image (13).jpg
Image (14).jpg
Image (15).jpg
Image (16).jpg
Image (17).jpg
Image (18).jpg
Image (19).jpg
Image (20).jpg
Image (21).jpg
Image (22).jpg
Image (23).jpg
Image (24).jpg
Image (25).jpg
Image (26).jpg
Image (27).jpg
Image (28).jpg
Image (29).jpg
Image (30).jpg
Image (31).jpg
Image (32).jpg
Image (33).jpg
Image (34).jpg
Image (35).jpg
Image (36).jpg
Image (37).jpg
Image (38).jpg
Image (39).jpg
Image (40).jpg


This PC › DATA (D:) ›
Search Output
watermarked (1).jpg
watermarked (2).jpg
watermarked (3).jpg
watermarked (4).jpg
watermarked (5).jpg
watermarked (6).jpg
watermarked (7).jpg
watermarked (8).jpg
watermarked (9).jpg
watermarked (10).jpg
watermarked (11).jpg
watermarked (12).jpg
watermarked (13).jpg
watermarked (14).jpg
watermarked (15).jpg
watermarked (16).jpg
watermarked (17).jpg
watermarked (18).jpg
watermarked (21).jpg
watermarked (22).jpg
watermarked (23).jpg
watermarked (24).jpg
watermarked (25).jpg
watermarked (26).jpg
watermarked (27).jpg
watermarked (28).jpg
STOP


This PC › DATA (D:) › Code › Compress
Search Compress
compressed (1).jpg
compressed (2).jpg
compressed (3).jpg
compressed (4).jpg
compressed (5).jpg
compressed (6).jpg
compressed (7).jpg
compressed (9).jpg
compressed (10).jpg
compressed (11).jpg
compressed (12).jpg
compressed (13).jpg
compressed (14).jpg
compressed (15).jpg
compressed (16).jpg
compressed (21).jpg
compressed (22).jpg
compressed (23).jpg
compressed (24).jpg
compressed (25).jpg
compressed (26).jpg
elected 74.3 KB


This PC › DATA (D:) › Code › Crop
Search Crop
cropped (1).jpg
cropped (2).jpg
cropped (3).jpg
cropped (4).jpg
cropped (5).jpg
cropped (6).jpg
cropped (7).jpg
cropped (8).jpg
cropped (9).jpg
cropped (10).jpg
cropped (11).jpg
cropped (12).jpg
cropped (13).jpg
cropped (14).jpg
cropped (15).jpg
cropped (16).jpg
cropped (17).jpg
cropped (21).jpg
cropped (22).jpg
cropped (23).jpg
cropped (24).jpg
cropped (25).jpg
cropped (26).jpg
cropped (27).jpg
cropped (31).jpg
cropped (32).jpg
cropped (33).jpg
cropped (34).jpg
cropped (35).jpg
cropped (36).jpg
cropped (37).jpg



Comparison.xlsx - Excel

PARMEET SINGH BANWAIT

File Home Insert Page Layout Formulas Data Review View Help Acrobat Tell me what you want to do Share

Calibri 11 Wrap Text General Merge & Center Conditional Formatting Format as Table Cell Styles Insert Delete Format Sort & Filter Find & Select Sensitivity

Clipboard Font Alignment Number Styles Cells Editing Sensitivity

A3 18.1286253450655

	Watermark					Compress					Crop				
	MSE	PSNR	RMSE	NRMSE	SSIM	MSE	PSNR	RMSE	NRMSE	SSIM	MSE	PSNR	RMSE	NRMSE	SSIM
3	18.12863	35.54715	[illegible]7772	0.234865	0.965864	57.39872	30.54178	7.576195	0.131992	0.855112	104.5862	27.93606	10.22674	0.097783	0.134396
4	15.58705	36.20316	3.948044	0.25329	0.945308	61.0619	30.2731	7.814211	0.127972	0.855112	103.6027	27.97709	10.17854	0.098246	0.187831
5	23.24924	34.46672	4.821747	0.207394	0.960523	70.77214	29.63218	8.412618	0.118869	0.878457	101.3915	28.07079	10.06934	0.099311	0.111223
6	12.95871	37.00519	3.599821	0.277792	0.962613	23.89553	34.34764	4.888305	0.20457	0.803012	84.46896	28.86383	9.1907	0.108806	0.402804
7	10.25103	38.02313	3.201723	0.312332	0.960079	42.43366	31.8537	6.514113	0.153513	0.888959	103.2564	27.99163	10.16152	0.098411	0.46114
8	13.60123	36.79502	3.687985	0.271151	0.972586	61.15716	30.26633	7.820304	0.127872	0.886766	104.7159	27.93068	10.23308	0.097722	0.172273
9	11.86077	37.38968	3.443946	0.290365	0.950594	31.2859	33.17732	5.59338	0.178783	0.850624	94.21863	28.38944	9.706628	0.103022	0.480387
10	33.12573	32.92915	5.755496	0.173747	0.962038	79.97667	29.10117	8.942968	0.11182	0.921954	103.7327	27.97165	10.18492	0.098184	0.176268
11	13.01028	36.98794	3.606977	0.27724	0.960103	43.49133	31.74678	6.594796	0.151635	0.817913	101.885	28.0497	10.09381	0.099071	0.122837
12	12.05716	37.31835	3.472342	0.28799	0.95915	57.79949	30.51156	7.602597	0.131534	0.885843	105.6306	27.89291	10.27767	0.097298	0.187556
13	12.73542	37.08067	3.568673	0.280216	0.922231	31.56659	33.13853	5.618415	0.177986	0.832937	89.08076	28.63296	9.438261	0.105952	0.342505
14	7.237043	39.53519	2.690175	0.371723	0.964996	102.5813	28.02012	10.12824	0.098734	0.891803	104.9837	27.91958	10.24616	0.097598	0.272612
15	10.9675	37.72973	3.311722	0.301958	0.959279	101.4079	28.07009	10.07015	0.099303	0.019551	97.67365	28.23303	9.882998	0.101184	0.119729
16	26.24508	33.94032	5.122995	0.195198	0.878076	41.05094	31.99757	6.407101	0.156077	0.000109	101.7606	28.05501	10.08765	0.099131	0.264989
17	12.6197	37.12031	3.552422	0.281498	0.962037	33.83077	32.83769	5.816422	0.171927	0.885886	99.07028	28.17137	9.953405	0.100468	0.390468
18	6.96648	39.70067	2.639409	0.378873	0.966274	49.36539	31.19658	7.026051	0.142327	0.909286	98.07552	28.2152	9.903309	0.100976	0.30972
19	9.934856	38.15919	3.151961	0.317263	0.950161	38.47895	32.27857	6.203141	0.161209	0.895624	93.92728	28.40289	9.691609	0.103182	0.272794
20	14.04596	36.65529	3.747794	0.266824	0.987649	114.8667	27.52886	10.71759	0.093305	0.892089	82.54956	28.96366	9.085679	0.110063	0.727646
21	9.629866	38.2946	3.103203	0.322248	0.96517	50.16859	31.12648	7.082979	0.141184	0.000233	100.948	28.08983	10.04729	0.099529	0.047697

Sheet1

Ready

100%

11

DEMO VIDEO

VIDEO EXPLANATION OF OUR CODE

https://drive.google.com/file/d/1NHdDcJciqj21rLpAeUau419XDJzF7mrz/view?usp=sharing
You can find the video here.

12

RESULTS & CONCLUSION

Result

★ So, here within we can see that we have successfully added upon the compared and performed certain actions upon the images.

12

REFERENCES

★ https://docs.python.org/3/library/

★ https://www.geeksforgeeks.org/libraries-in-python/

★ **https://www.python.org/doc/essays/blurb/**

★ **https://experience.dropbox.com/resources/what-is-watermarking**


Thank
you!!!