LimboZone -?-> LimboZ0ne WRITEUP

The only thing we have is a 7zip file: `level_0.7z` , extracting the zip we obtain 4 files: `level_1.zip` , `level_0.png` , `l3vel_0.png` and `ForceLevel.py` .
In `ForceLevel.py` we have a script which bruteforce a password, from this we can acquire the method to build a password, observing carefully the images we can see that they differs only for 1 pixel, so we can find RGB values and position of this pixel and use this informations to build the password with the method we acquired from `ForceLevel.py` and extract the next zip.
From the 16th level we also have to transorm the second image in order to make him match with the first, and then apply the same method to proceed to the next level.
At the 1024th level we extract a txt file: `lev3l_1024.txt` which contains the flag: `{FLG:p1xel0ut0fBound3xcept1on_tr4p_1s_shutt1ng_d0wn}`

```python
import os
import subprocess
from PIL import Image

def extract_zip(zip_path, pwd, out):
    """
    Extract a zip archive, provided the path to the zip, password and path for tl
    and returns a list containing the filenames of the extracted files.
    """
    a = os.listdir(out)
    subprocess.run(["7z", "x", zip_path, f"-p{pwd}", f"-o{out}"], capture_output
    b = os.listdir(out)

    return list(set(b) - set(a))


def get_pwd(first_image, second_image):
    """Compare the two images provided to extract the password."""
    im1 = Image.open(first_image)
    im2 = Image.open(second_image)

    width, height = im1.size
    for x in range(width):
        for y in range(height):
            r1,g1,b1 = im1.getpixel((x,y))
            r2,g2,b2 = im2.getpixel((x,y))

            if r1 != r2 or g1 != g2 or b1 != b2:
                xy = str(x) + str(y)
                rgb1 = '{:0{}X}'.format(r1, 2) + '{:0{}X}'.format(g1, 2) + '{:0{
                rgb2 = '{:0{}X}'.format(r2, 2) + '{:0{}X}'.format(g2, 2) + '{:0{
                pwd = xy + rgb1 + rgb2
                return pwd

    return "Error"

# The following are some functions to alter the two images

def nothing(input_img):
    pass

def flip_top_bottom(input_img):
    img = Image.open(input_img)
    img = img.transpose(Image.FLIP_TOP_BOTTOM)
    img.show()
    img.save(input_img, "PNG")

def flip_left_right(input_img):
    img = Image.open(input_img)
    img = img.transpose(Image.FLIP_LEFT_RIGHT)
    img.show()
    img.save(input_img, "PNG")

def rotate_90(input_img):
    img = Image.open(input_img)
    img = img.transpose(Image.ROTATE_90)
    img.show()
    img.save(input_img, "PNG")

def rotate_180(input_img):
    img = Image.open(input_img)
    img = img.transpose(Image.ROTATE_180)
    img.show()
    img.save(input_img, "PNG")

def rotate_270(input_img):
    img = Image.open(input_img)
    img = img.transpose(Image.ROTATE_270)
    img.show()
    img.save(input_img, "PNG")

def transpose(input_img):
    img = Image.open(input_img)
    img = img.transpose(Image.TRANSPOSE)
    img.show()
    img.save(input_img, "PNG")

def transverse(input_img):
    img = Image.open(input_img)
    img = img.transpose(Image.TRANSVERSE)
    img.show()
    img.save(input_img, "PNG")

def main():
    # Arsenal of functions to manipulate the second image
    tries = [nothing, flip_top_bottom, flip_left_right, rotate_90, rotate_180, r

    extract_zip("level_0.7z", "", ".")

    i = 0
    try:
        while(1):
            for t in tries:
                # Backup the second image and try to manipulate it
                os.system(f"cp lev3l_{i}.png lev3l_{i}.png_b")
                # Manipulate second image
                t(f"lev3l_{i}.png")

                # Extraction process
                try:
                    pwd = get_pwd(f"level_{i}.png", f"lev3l_{i}.png")
                    print(pwd)

                    extracted_files = extract_zip(f"level_{i+1}.7z",pwd , ".")
                    print(extracted_files)

                    if extracted_files:
                        os.system(f"rm lev3l_{i}.png_b")
                        break
                except:
                    extracted_files = []
                # Restore the backup since the current attempt failed
                os.system(f"mv lev3l_{i}.png_b lev3l_{i}.png")
            else:
                print("Error: no method worked.")
                exit(1)

            # Remove the current level files
            os.system(f"rm level_{i+1}.7z level_{i}.png lev3l_{i}.png")
            i += 1
    except:
        os.system("cat *.txt")

if __name__ == "__main__":
    main()
```