Image Magic CTF Writeup

This document is a walkthrough on one way to solve the **Image Magic CTF** on **CTFLearn**. The objective is to explain how I was able to solve this CTF to my future self.

General Information

- *Difficulty:* Medium
- *Category:* Programming
- *Link:* [Challenge - Image Magic - CTFlearn - CTF Practice - CTF Problems]

Introduction

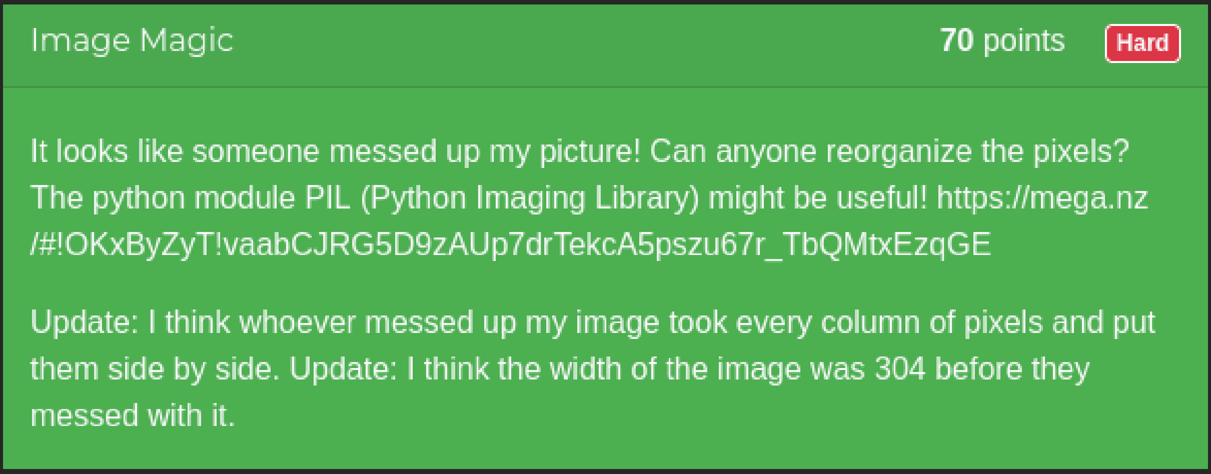
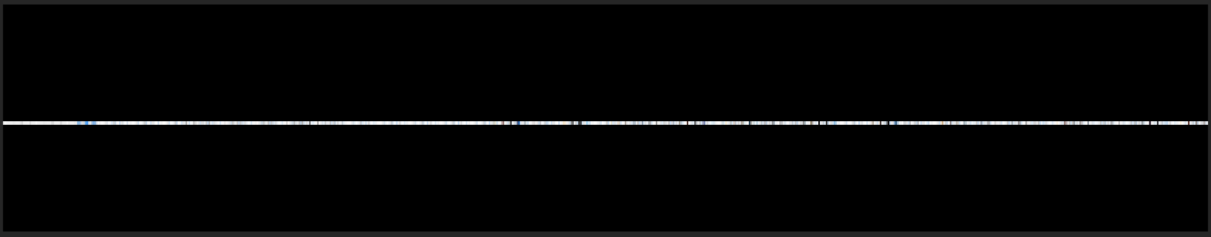


Image Magic 70 points Hard

It looks like someone messed up my picture! Can anyone reorganize the pixels? The python module PIL (Python Imaging Library) might be useful! https://mega.nz/#!OKxByZyT!vaabCJRG5D9zAUp7drTekcA5pszu67r_TbQMtxEzqGE

Update: I think whoever messed up my image took every column of pixels and put them side by side. Update: I think the width of the image was 304 before they messed with it.

We're given a PNG file, which looks like this:



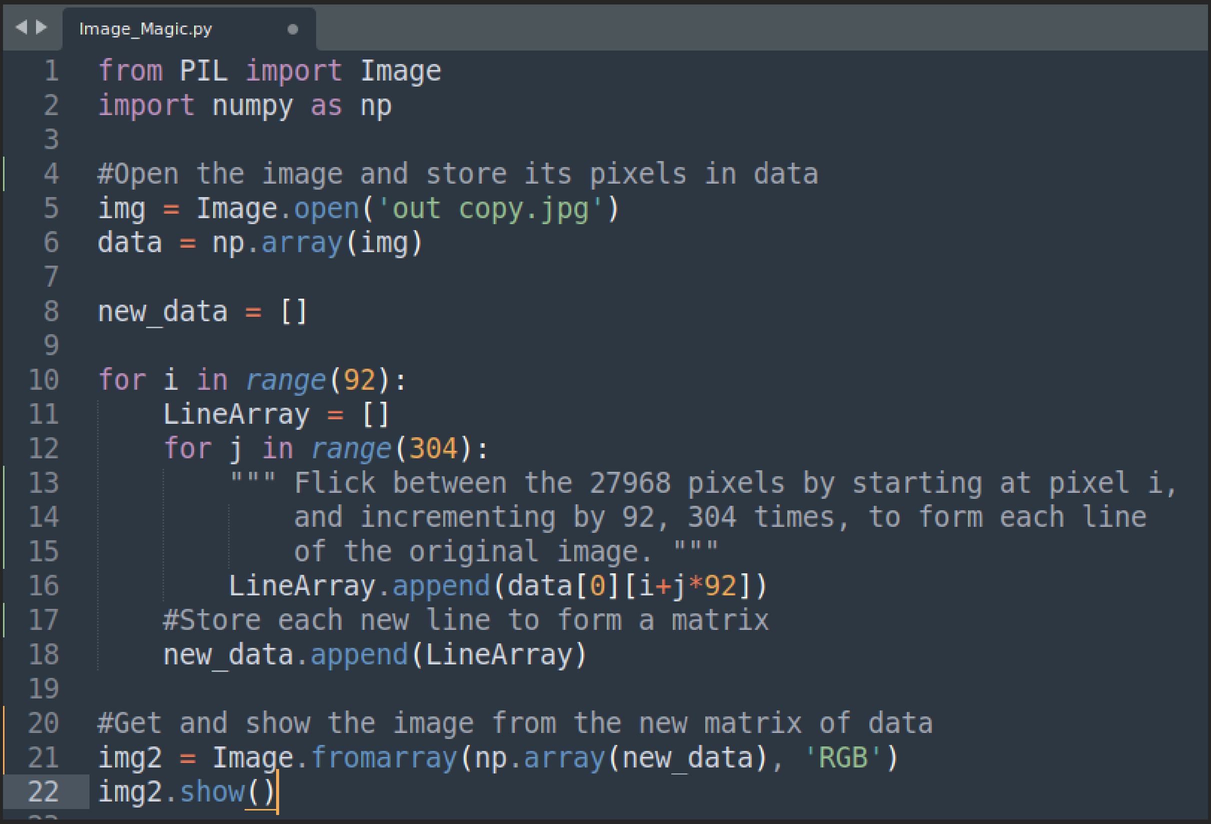
It's just a line of 27968 pixels, so it looks like the original image was jumbled in some way.

Now, reading the challenge description, it seems like the person who disorganised the image *"took every column of pixels and put them side by side".*

Moreover, if the image's width is 304, then its height is 27968 / 304 = 92

That being said, to solve this challenge, we're going to use the PIL python module, in order to transform the image into an array of numbers, and reorganise them in a specific way to recover the original.

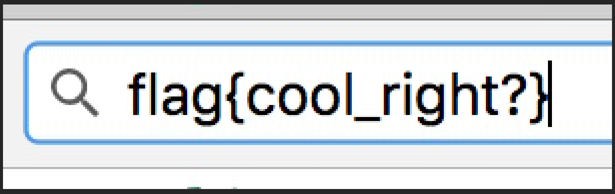
Let’s consider the following python script:


Image_Magic.py

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

#Open the image and store its pixels in data
img = Image.open('out copy.jpg')
data = np.array(img)

new_data = []

for i in range(92):
    LineArray = []
    for j in range(304):
        """ Flick between the 27968 pixels by starting at pixel i,
            and incrementing by 92, 304 times, to form each line
            of the original image. """
        LineArray.append(data[0][i+j*92])
    #Store each new line to form a matrix
    new_data.append(LineArray)

#Get and show the image from the new matrix of data
img2 = Image.fromarray(np.array(new_data), 'RGB')
img2.show()
```

Reading the comments will help us understand how exactly we’re getting the original image

Executing this script, we get the following image:


flag{cool_right?}

We get the following flag: **flag{cool_right?}**