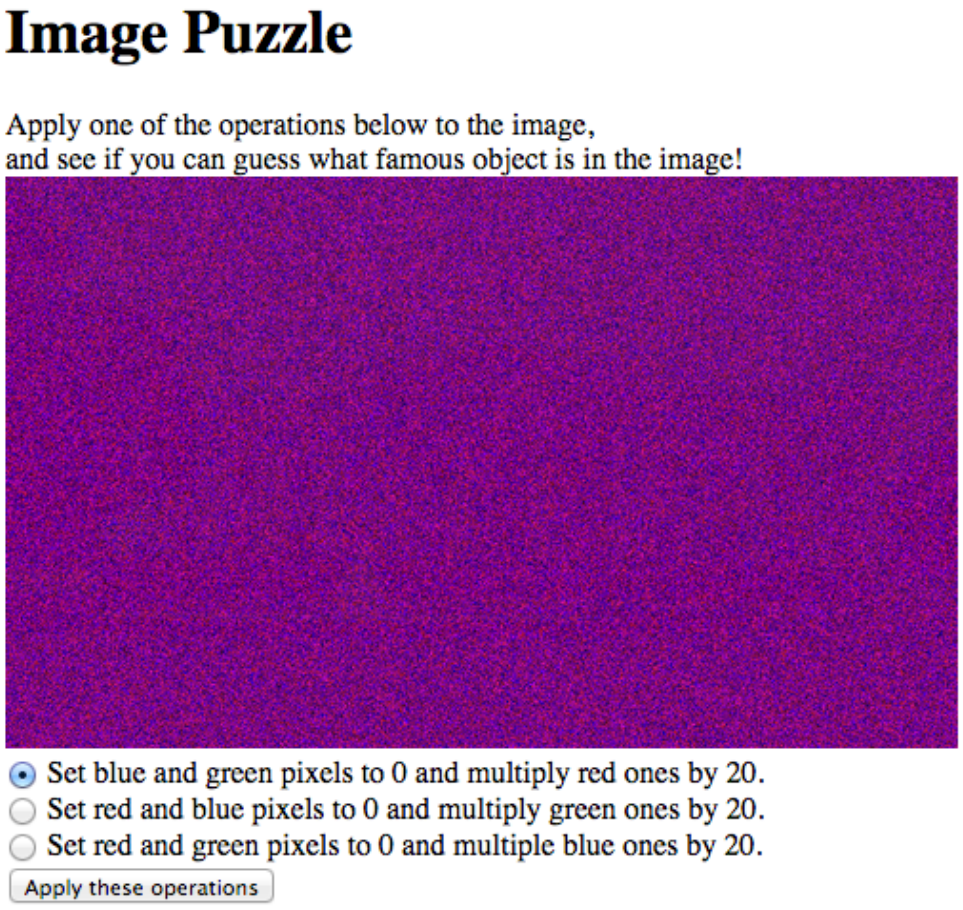
CS110 Project 2: Image Puzzles

Prof. Karpenko

Due: Saturday Nov 1st, 2014, 11:59pm

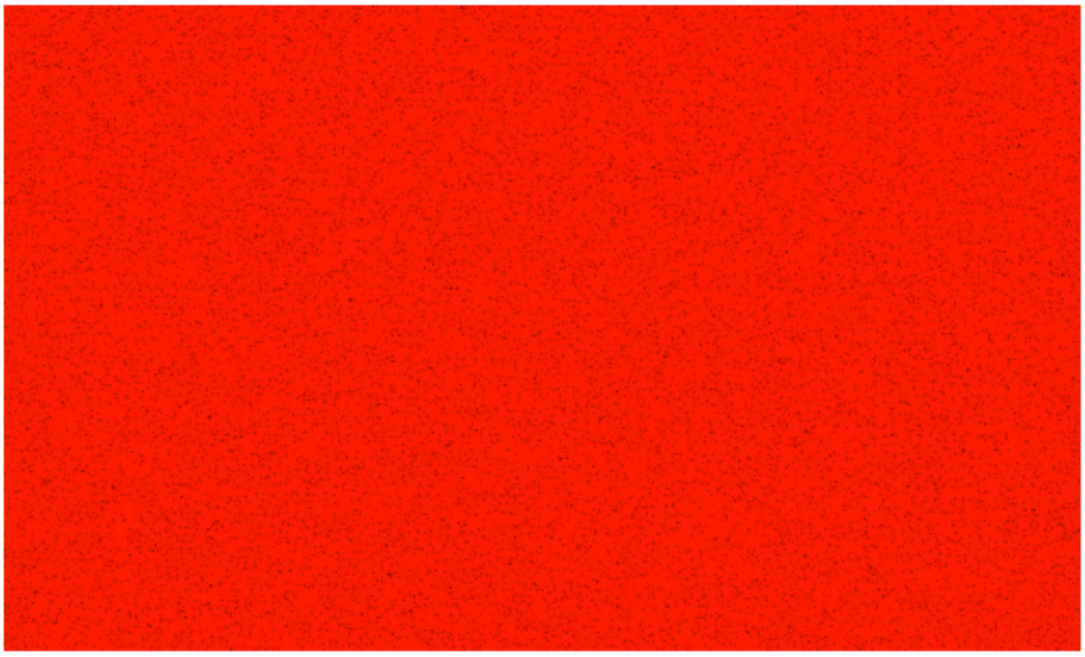
For project 2 you will develop a simple web application that displays an image puzzle like the one shown below. The image shows something famous, however it has been deliberately distorted. For example, in the example below, the image is in the green values, however the green values were divided by 20, and red and blue "noise" (random values) was added to the image. Your application will let the user "fix" the image and reveal the hidden famous object or a person. You will use a web development framework called Flask, and an image-processing package called Pillow. The webpage you will create will look similar to the image shown below:



After applying the selected operation (we chose to apply the second one in this example which is the "correct" one), the user will see another HTML page with the processed image and will be able to type his or her guess in the text field:

White house Enter

If the guess is correct, the web application will take the user to the new html page that says "Correct!", otherwise it will redirect the user to the first page and the user can try another operation on the image.
If the user chose to apply one of the "incorrect" operations, the actual image will not be revealed and it will be impossible to guess what object is in the image. For the image above, clicking on the first operation would produce the following webpage:



I can't guess! Enter

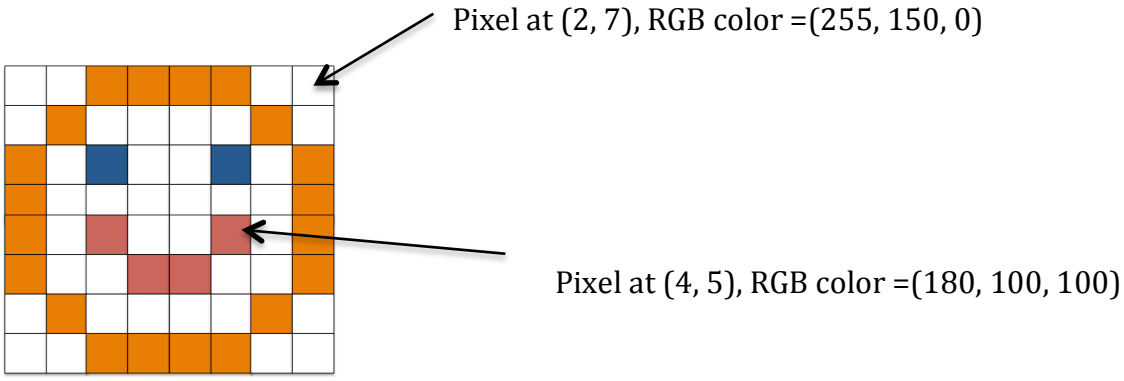
Flask

http://flask.pocoo.org/
http://en.wikipedia.org/wiki/Flask_(web_framework)
To get started, read a short introduction to Flask:
http://flask.pocoo.org/docs/0.10/quickstart/

Flask is a Python web development framework that makes creating simple web applications easy. Flask examples we discussed in class are posted on Canvas under Modules (see Lecture 23: Flask Examples). Start by running flaskHelloWorld.py example, then look at flaskAnotherSimpleEx.py, and finally at flaskHalloweenExample.py. Make sure you fully understand flaskHalloweenExample before you start working on the project.

Image Processing

An image is often represented as a grid of "pixels" (i.e. dots), where each pixel has a color. There are different ways to represent color. In the RGB model, each color is a mix of red, green and blue colors. For example, to get a while color, you would mix the three colors in equal amounts., while using the mix of red and the green will produce a yellow color.



Pillow(Imaging Library)

http://pillow.readthedocs.org/
Pillow allows a user to manipulate images easily. To process a particular image, we should first add the following statement to our python file:
from PIL import Image

I also wrote several helper functions that let you perform several basic operations on the image without having any knowledge of object-oriented programming that Pillow uses.

If you have an image called myimage.png that is stored in the same directory as your image processing code, you can open the image using the provided helper function **openImageFile:**

```
im = openImageFile("myimage.png")
```

You get the width and the height of the image using the provided function **size**:

```
(w, h) = size(im)
```

getPixel(pos, image) lets you access a given pixel in the image
and setPixel(pos, image, RGBColor) lets you set the RGB color of a particular pixel in the image.

Implementation

Before you start, install Flask and Pillow on your laptop (if you are using a lab machine, these packages are already installed there), download imageHelperFunctions module and the distorted image from Canvas. Create a project 2 folder. In that folder, create a subfolder called **static.** Place the distorted image in the static folder. Flask assumes your images are in that folder, so all your images have to be saved there.

Project 2 template file has been provided to you (it is under Assignments/Project 2 on Canvas). All the function signatures have already been written, you just need to fill in the code in the body of each function. Please note that ***you do not have to use*** the provided template file. The instructions below assume you are using the provided template, but it is easy to modify them to work with your own code. I recommend taking the following steps to complete the project:

1. Start by writing a function called **editImage** that takes three parameters: an integer "option" which is 1, 2, or 3; the filename of the original image, and the filename where the edited image is going to be saved. This function should open the original image, iterate over the pixels and set the color of each pixel depending on the value of the "option" parameter:
    - If option == 1, then set the green and blue values of the pixel to 0, and multiply the red value by 20.
    - If option == 2, then set the red and blue values to 0, and multiply the green value by 20
    - If option == 3, then set the red and green values to 0, and multiply the blue value by 20.

   Then this function should save the edited image in the file with the name provided in the parameter filename2.

Note: You would need to copy imageHelperFunctions module (on Canvas/Project2) to your project directory and import the imageHelperFunctions module. You laptop needs to have Pillow installed for the module to compile. You can start with your code from homework 8 and modify it according to the description above. Test this

function on different files with different options. Do not move onto other functions until this function is working fine.

2. Write **displayPuzzle** function that returns an html string corresponding to the html document shown on top of this document. The html document should have:
    - the heading "Image Puzzle" and the text below it asking the user to select one of the radio buttons to "decode" the image.
    - the image with the noise (red and blue) that is provided to you : https://usfca.instructure.com/courses/1298637/files/58737185/download?wrap=1
      As mentioned above, the image should be in the subfolder called **static**, and the path to the image that you specify in the image HTML tag will be something like:
      "/static/distortedImage.png"
    - three radio buttons as shown in the image above and one submit button.

Once you run your flask application, this function will be invoked when you go to: http://127.0.0.1:5000/ URL in your browser.
Test this part first to make sure the page is displayed correctly. You might want to play with flaskHalloweenExample before working on this part, it is very similar to what you are trying to do.

Next, change the **action** field of the HTML form that has the submit button so that when the button is clicked, flask takes you to the showResult page (and the corresponding showEditedImage function gets invoked). Test that this works before moving to the next step.

When the user selects one of the three image editing operations, we would like to show the corresponding image. The simplest thing to do is pre-compute these images when we first load the page. So in this function we will call editImage function you wrote earlier with three different options: 1, 2, 3 and save three different files in the static folder. The first image, let me call it **image1.png**, will store the result of applying the first operation to the original image (setting the green and blue values of the pixel to 0, and multiplying the red value by 20.) The second image, image2.png, will store the result of applying the second operation to the original image, and the third image, image3.png – the result of the third operation. We only need to save these images once, so we can first check if, say, image1.png, already exists (see the code below), and if not, call editImage:

```
if not os.path.exists('static/image1.png'):
     editImage(1,'static/distortedImage.png','static/image1.png')
```

The same needs to be done for the other two image editing operations. All of the images will be saved in the static subfolder.
Test function displayPuzzle thoroughly before moving onto the next step. This is the main function of the project.

3. Write a function **showEditedimage** that gets invoked when you the user clicks on the submit button the main html page. In this function you should check what radio button the user selected (refer to flaskHalloweenExample to see how to do that). Then you should return the HTML string that displays one of the images (image1, image2, image3) depending on which operation the user selected.

   The returned HTML document should also have an HTML form with the text field where the user can type their guess. The action field of the HTML form should redirect the user to the third html page (/guessImage):
   action="guessImage"

4. Finally, write **guessImage** function, that gets invoked when the app gets redirected to /guessImage. The function should get the value the user typed in the text box, and check if it's a "white house" (for now we assume we have only the image of a white house). If the value the user typed is correct (== "white house"), then return "Correct!", otherwise redirect the user to the main page for our application:
   return redirect("/")

5. **(Optional, Extra credit, 2 pts)** If you would like, you can deploy your web application at:
www.pythonanywhere.com

Grading:

This project is worth 10% of your total grade. **You are not allowed to use any code from the web or collaborate on the project with anybody.** You may receive help only from the instructor, the TAs or the CS tutors. I will randomly select several people from each section of cs110, and ask them to come for an interactive code-walkthrough. If you submit the code that you cannot explain to me during the code review, you will get a 0 for the project.

Submission: Save your file in **project2.py** . Thoroughly test your code before submitting it and **add comments to your code**. Upload all your files to Canvas.

Installation instructions:

Flask and Pillow are already installed on the lab machines. If you prefer to have these packages installed on your personal laptops, follow the installation instructions that were emailed to you (the instructor emailed the Mac installation instructions, and Carlos posted Windows instructions on Canvas under Announcements).