Computer Science 1 — CSci 1100
Lab 5 — Images and Modules

This lab explores the use of images and the use of external modules to obtain data, in this case images. You can find all image functions at:

`http://pillow.readthedocs.org/en/latest/handbook/tutorial.html`

and some examples at:

`http://www.cs.rpi.edu/~sibel/csci1100/fall2014/course_notes/lec08_modules.html`

In this lab, you have two mandatory checkpoints that everyone must complete. For the third checkpoint, we will give you a couple of options. You must complete at least one of these, but you are welcome to try all of them and get help. You are given a module called `panoramio.py` which will be used in one of the possible checkpoint 3 options.

Checkpoint 1: Two by two wallpaper

To start this part, first download the ZIP folder from Piazza that contains many images and a file called `check1.py`. Unzip this folder in your Dropbox folder for Lab5. Make sure that programs are saved in the same directory as these images. You must save your file first before executing.

In this first part of the lab, you are going to create two by two a wallpaper using any four of the images. We recommend the first four images in the given list, but you can use any image you wish.

You program must first create a `512x512` blank document.

Then, you must open each image, resize it to size `256x256` and paste them into this new image.

You can check the similar example from class to see how each function is called. Remember, paste function modifies an image but does not return anything.

When done, call the `show()` function to check that the wallpaper looks correct. You will see that the images are distorted because the original images are not square and become distorted when resized. We will fix that in the next part.

Recall a few image functions that might help you accomplish this:

```
from PIL import Image    ##must import Image first to be able to use it
im = Image.new(mode, size, color)  # use mode 'RGB', color 'white'
im = im.resize((width,height)) # resize to the given width/height
im.paste(pasted_image, (x,y)) # (x,y) coordinates of upper left corner
im.save(filename)
im.show()
```

Note that the size is always a tuple.

To complete Checkpoint 1, show a TA or a Mentor your code and the output of your program.

Checkpoint 2: Image correction

To start this part, create a new file called `check2_helper.py`. You will write a function in this file which will later be used as a module.

What we want to accomplish in this checkpoint is to crop images so that they are square so that resizing does not distort them. If the image is too wide, then you must crop it along the x-axis. If the image is too long, then you must crop it along the y-axis. Always start from the top left corner.

Check the course notes for finding the size of an image and cropping. Remember that the crop function returns a new image.

Write a function that takes as input an image object, crops it to make it square and returns the resulting object. You must write a function to accomplish this.

Now, write code in `check2_helper.py` to test this function with an example image. Open the image, crop it to a square using the function and then show the result. Are you convinced that it works? Then, remove all the test code, leaving only the function.

Copy your solution from checkpoint 1 to a new file called `check2.py`. Modify your code so that:

- You import your own module `check2_helper.py`
- You use your own function from this module to make each image in the wallpaper square before resizing them.

To complete Checkpoint 2, show a TA or a Mentor your code and the output of your program. We will check that you are calling your module correctly, and that your program has correct structure.

Checkpoint 3 (option 1): Use an external source for images

Instead of using the files given to you, you can use external sources. To accomplish this, we have written a module called `panoramio.py` that finds images taken at a specific address (using images from Google maps). You have a few functions that will be useful to you in this part:

```
import panoramio as pan
urls = pan.getPhotos('Eiffel Paris France',5) # Return 5 pictures of Eiffel Tower
im = pan.openphoto(urls[0]) # Open the first image URL
```

The function `getPhotos()` will take an address and the number of desired images (5 in the above example) as input. It will find all photos near the given address (up to the given number), and return a list of URLs of these images. There may be no photos at a given address, especially if your address is not understood by the program. In this case, you will get an empty list.

For each photo, the URL is a string like this:

`+u'http://mw2.google.com/mw-panoramio/photos/medium/59461095.jpg'+`

Don't worry about the `u` prefix, it just means the string is encoded in Unicode. It should not matter to you.

To open an image from a URL, we will use a different method than Image.open(). We provided a function for you called `openphoto` that does this as shown above.

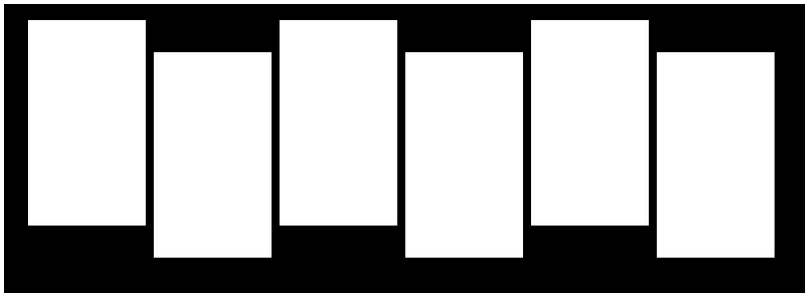
Your task now in this checkpoint is easy:

- Ask the user for an address
- Read the user input, and find 4 images at this address
- If the returned list has at least 4 images, then construct the wallpaper using these images and show the result.

To complete Checkpoint 3, show a TA or a Mentor your code and the output of your program.

Checkpoint 3 (option 2): A different layout for wallpaper

Here is a bit of challenge for you. We will change the wallpaper layout so that you put 6 images in a line, alternating the vertical location as shown in the figure below. Each white box is an image. I recommend you use the last 6 images (`1.jpg,..,6.jpg`) in the list for this.



To accomplish this:

- You will need to resize the images (no cropping) proportional to their original size so that their height is 256.
- Paste them in a wallpaper of size 1000, 360.
- The first image starts at (30,20).
- Images have 10 pixels in between along the horizontal axis.
- The images move up and down 40 pixels along the vertical axis.

If you do this, I promise you will get a cool wallpaper!

To complete Checkpoint 3, show a TA or a Mentor your code and the output of your program.