**CS506 Programming for Computing**

**HOS05A– Manipulating Images and Headless Browser Automation**

4/10/2025 Reviewed by Clark Ngo

05/01/2025 Reviewed by Naveena Moddu

05/01/2025 Reviewed by Ashwin Paturi

School of Technology & Computing (STC) @ City University of Seattle (CityU)

**Before You Start**

* **Screenshots may be different from your environment.**
* The directory path shown in screenshots may be different from yours.
* There might be subtle discrepancies along with the steps. Please use your best judgment while going through this cookbook-style tutorial to complete each step.
* Some steps may not be explained in detail. If you are not sure what to do:
  + Consult the resources from the course.
  + If you cannot solve the problem after a few tries (usually 15 -30 minutes), ask a TA for help.

**Learning Outcomes**

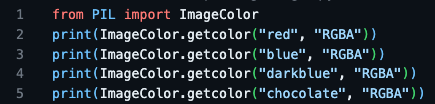
* **Section 1: Preparing your environment**
* **Section 2: Manipulating Images**
* **Section 3: Headless Browser Automation**

**Section 1: Preparing your environment:** Get started with your virtual environment here: <https://cityuseattle.github.io/docs/git/github_codepsace/#codespaces>

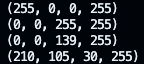
**Section 2: Manipulating Images**

**Image Colors:**

* 1. In your terminal, type: pip install pillow: 
  2. Create a file named image\_color.py and type the following:



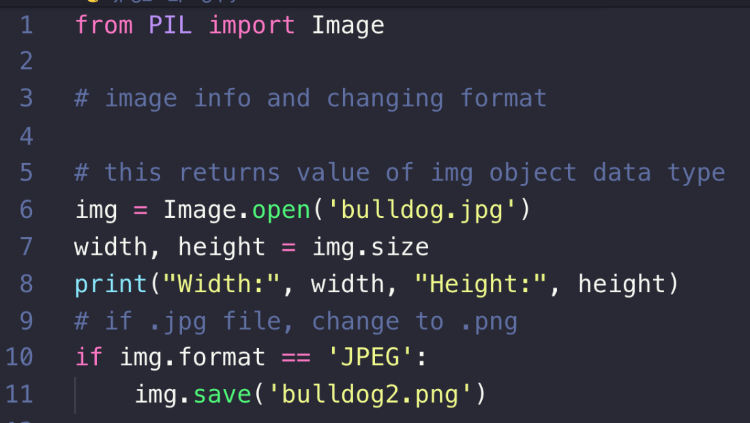
* 1. Run your code and check the output:



* 1. This is the basic understanding of computer images before using the module. An RGBA value represents color in the image (red, green, blue, and alpha(transparency)). Each value is between 0 and 255; for example, (255, 255, 255, 255) means a maximum of white and fully opaque. Pillow makes things easier for you; there is no need to memorize the RGBA value. So, (255, 255, 255, 255) equals ImageColor.getcolor(‘white’, ‘RGBA’)

**Convert image file types:**

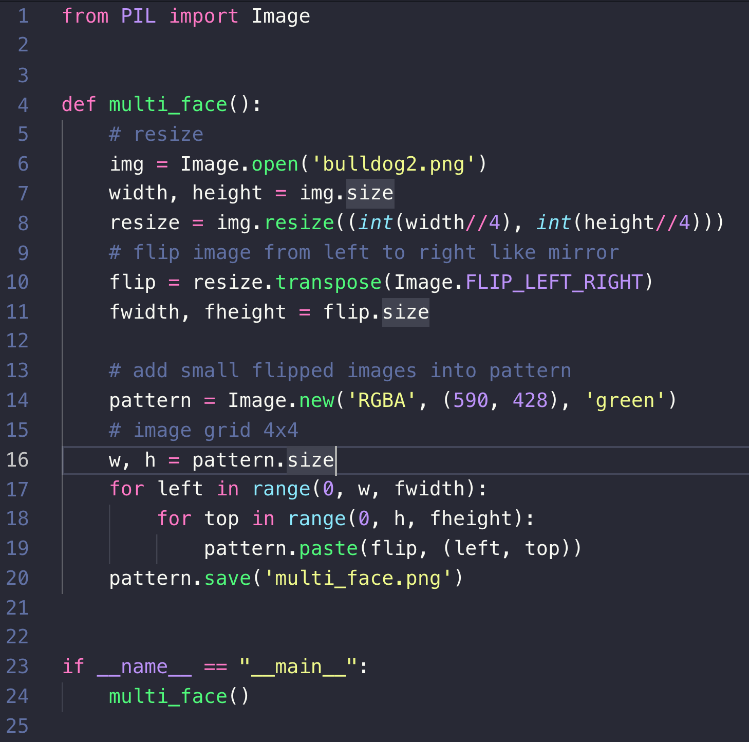
1. Create a file named jpeg\_to\_png.py and type the following:

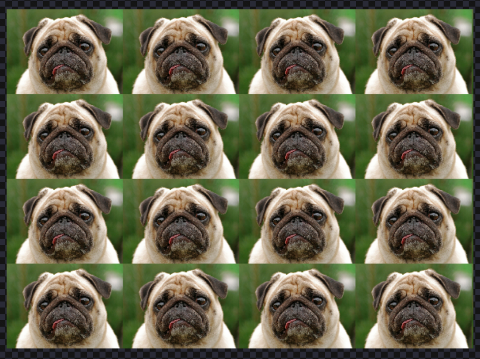


1. The output will be displayed as W: 590 H: 428. A new image file named bulldog2.png will be created.

**Repeating an image**

1. Let’s edit the image by cropping, resizing, and pasting it into another image. Create multi\_face.py with the following code. We will use the bulldog2.png created from the above code.

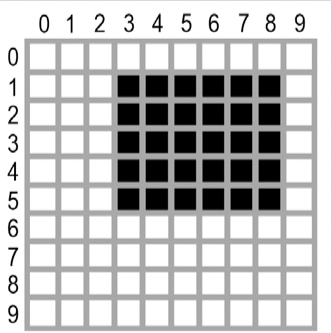




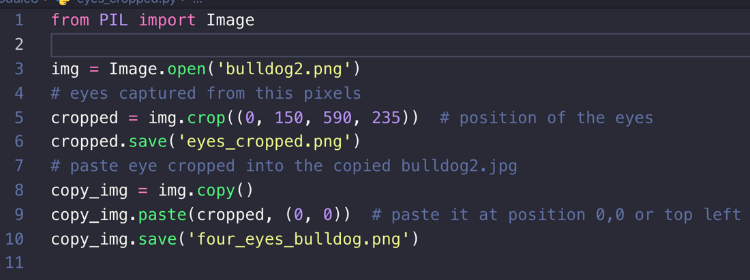
**Cropping an image**

1. Create eyes\_cropped.py and type the following. We will use bulldog2.png.

This is how we find a particular area.



The area represented by the box tuple (3, 1, 9, 6)

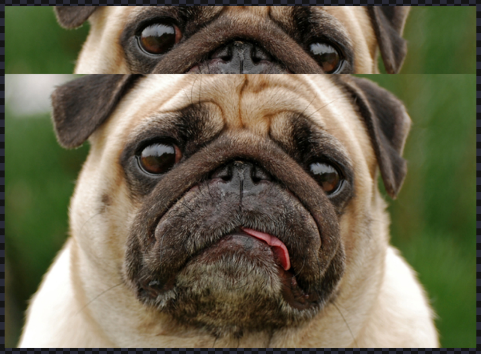


1. Run your code and view the output:

**This is what eyes\_cropped.png and four\_eyes\_bulldog.png look like.**



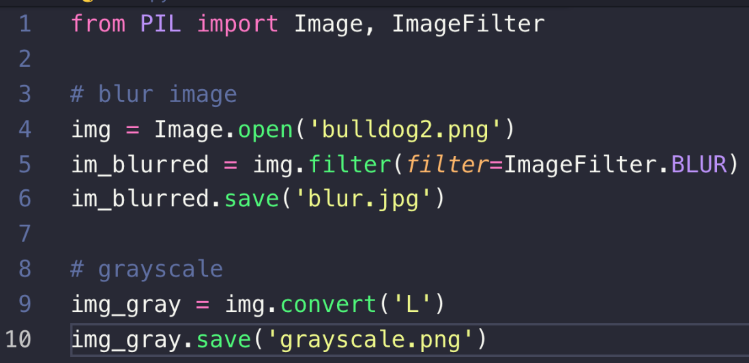
**And**



**Filtering an image**

The pillow module can also filter your image.

1. Create a file named filter.py and type the following code:



1. Run the code, and you should see two new files named **blur.jpg**and **grayscale.png**created.

**blur.jpg**

A close up of a dog looking at the camera

Description automatically generated

**grayscale.png**

A close up of a dog looking at the camera

Description automatically generated

**Section 3: Headless Browser Automation**

**What does it mean:**

* "Headless" = running a browser without a graphical user interface (GUI).
* "Browser automation" = programmatically controlling a browser (e.g., navigating pages, clicking buttons, scraping data).

So, headless browser automation refers to automating browser tasks without rendering a visible window — perfect for environments like:

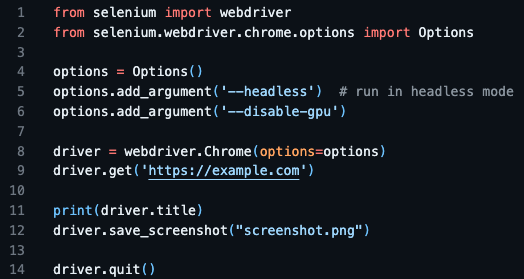
* CI/CD pipelines
* Cloud IDEs (like GitHub Codespaces)
* Web scraping tools
* Server-side tests

**Selenium for Automating Web Browsers:** Selenium is a powerful tool for automating web browsers. It allows developers to simulate user interactions like clicking, typing, and navigating through pages.

**Selenium in Headless mode:** When run in headless mode, Selenium operates without opening a visible browser window, making it ideal for use in environments without a graphical interface, like servers or GitHub Codespaces. This mode is perfect for tasks such as automated testing, web scraping, or background data collection, offering a full browser's functionality with improved speed and resource efficiency.

**Taking a screenshot of a website**

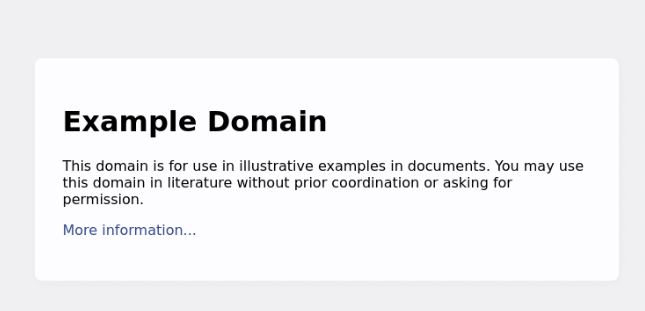
1. In your terminal, type: pip install selenium
2. Test an example website and view: <https://example.com>
3. Create a file named browser\_automation.py.
4. Type the following:



This code will access the website and extract the website’s title. It will also take a screenshot of the current page it's in.

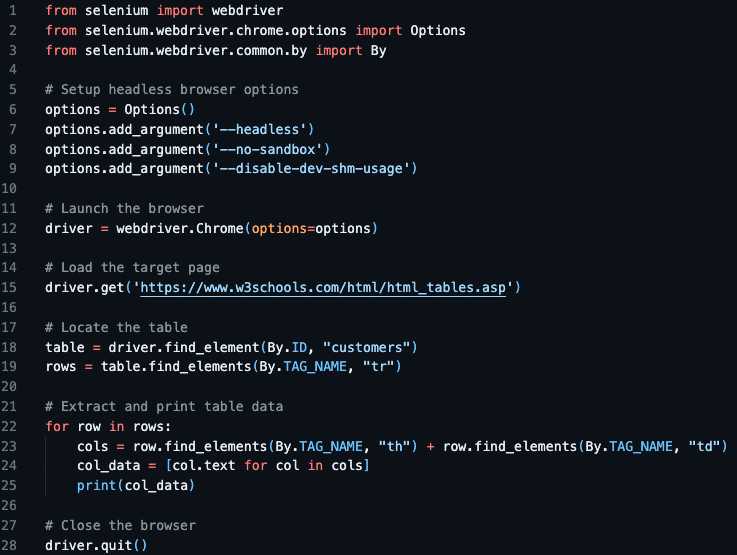
1. View the output and the screenshot.png file. The screenshot should be the same as the website’s page.





**Extracting a table from a website**

1. Test a website with a table and view it: <https://www.w3schools.com/html/html_tables.asp>
2. Create a file named extract\_tables.py.
3. Type the following:



1. View the output:



**Push your work to GitHub Codespaces.**

Follow instructions here: <https://cityuseattle.github.io/docs/git/codespaces_submission/>