**Installing Pillow and SimpleImage on VS Code (tested on Mac but not on PC or Linux)**

**Adapted from CS106A Spring 2025**

**Install Pillow**

The SimpleImage module we are covering in class uses the "Pillow" library. A library is a body of already written code which you import and use, and in this case the Pillow library contains code to manipulate images. In order for SimpleImage to work properly, you need to install Pillow on your machine.

To install Pillow, you should first open a "terminal" window: the easiest way to do this is to use the "Terminal" tab at the bottom of VSCode or open up a separate terminal window. Press **Cmd + Space** to open **Spotlight Search**. Then type **Terminal** and hit **Enter**. You can also pin the Terminal to your dock for quick access.

Type the following command into the Terminal. (Note that "Pillow" starts with an uppercase P.) On Windows, type "py" or "python" instead of "python3"):

python3 -m pip install Pillow

...prints stuff...

Successfully installed Pillow-9.5.0

If that doesn't work, try **pip3 instead of pip**.

Note: You only have to install Pillow once. After you have successfully installed it, the library will be available for use in all future assignments and lecture examples.

To test that Pillow is working, type the following command into the VSCode terminal while you've opened any assignment or lecture example folder that contains the file simpleimage.py. On Windows, type "py" or "python" instead of "python3"):

python3 simpleimage.py

# yellow rectangle with green stripe appears

This runs the simpleimage.py code included in the project folder. When run like this, you should see a big yellow rectangle with a small green stripe on the right pop up. If you cannot get Pillow installed successfully, please check out our forum or go to the course staff office hours for help.

For the test to work, you need to have the simpleimage.py document in the same project folder you will be working on your code. Then, all you need to do is add from simpleimage import SimpleImage to the top of your code.

For ease of learning, would download the entire Week 6 Images folder onto your desktop. This folder already has the simpleimage.py file that you will need as well as the assignment, images, and starter code file.

If you are starting from scratch, simpleimage.py, your code and images MUST be in the same folder. You can navigate to this folder using cd foldername in terminal.

**Common errors:**

 **No module named 'simpleimage'**: Make sure simpleimage.py is in the same folder.

 **No image appears**: Check the filename and path.

 **VS Code settings**: Ensure Python extension is installed and an interpreter is selected.

ChatGPT is a useful resource if you run into problems with installation.

**Basic SimpleImage Functionality**

The SimpleImage code used in CS106A provides you with some basic digital image processing tools:

1. **Reading an image from a filename.** This function reads the image from the specified file and returns a SimpleImage object that can be stored in a variable to refer to the image.

image = SimpleImage(filename)

The filename is something like 'my\_image.jpg', a string that contains the name of an image file. This file should be within the project folder you're working on. You can add any photo you like to your project folder in order to open it with SimpleImage.

1. **"For-each" loop over all pixels in the image.** This gives you access to each of the pixels in the image one-by-one so that you can apply the same code to all of them.
2. for pixel in image:

# use pixel in here

1. **Accessing data inside a pixel.** Each pixel has properties that can be accessed with the "dot" syntax. These properties include its RGB (red, blue, and green) components, which are integers between 0 and 255, as well as the location of the pixel in its corresponding image. The pixel's color properties can be accessed and modified, but the pixel's location properties cannot be directly modified.
2. # color properties
3. pixel.red
4. pixel.green
5. pixel.blue
6. # location properties
7. pixel.x

pixel.y

1. **Displaying the image.** Each image can be displayed on your computer using the show function. After you've finished manipulating the image in whatever way you like, you should call this function to make the image display.

image.show()

1. **Accessing image size.** Each image has two attributes - height and width - that can be directly accessed from the image. This is useful for algorithms that rely on the size of the image.
2. # size attributes
3. image.width

image.height

**Advanced SimpleImage Functionality**

As you progress into creating your own images or working on more complicated image algorithms, there are some more advanced features of SimpleImage that you can take advantage of:

1. **Creating a blank image with a specified height and width.** This is useful for starting off with a "blank canvas" that you can color in an interesting way. Both of the lines of code below are valid ways of making a blank image with a width of 200 pixels and a height of 100 pixels.
2. # can use either
3. image = SimpleImage.blank(200, 100)

image = SimpleImage.blank(width=200, height=100)

1. **Accessing a pixel by its x, y coordinates.** The top left of the image corresponds to x=0, y=0. x-coordinates get larger as you move to the right, and y-coordinates get larger as you move down.
2. # gets the pixel from image at (x, y)

pixel = image.get\_pixel(x, y)

Note: x and y should be integers and x must be in the range 0 to width-1 (inclusive) and y must be in the range 0 to height-1 (inclusive).

1. **Setting the value of a certain pixel in an image by its x, y coordinates.** If you have a pixel and you want to place it directly into an image, you can do so with the following function.
2. # puts the pixel into image at (x, y)

image.set\_pixel(x, y, pixel)

1. **Resizing an image to be the same size as another image.** This function takes in a target\_image whose size you want to match and modifies image so that it has identical dimensions to the target image. This function is especially helpful when working on problems that involve taking pixels from one image and copying them onto another image). This function helps ensure that the sizes of the two images match up.

image.make\_as\_big\_as(target\_image)

**Range Loops with SimpleImage**

The "for each" loop described above is the easiest way to loop over all of the pixels in an image. However, sometimes you want to write loops that access pixels by their x, y coordinates. The code below demonstrates the standard nested for loop pattern to access all of the pixels in an image. The outer for loop iterates over the rows (starting with the top row (y = 0) and moving on to the next row (y = 1)), and the inner for loop iterates over the columns.

def example(filename):

image = SimpleImage(filename)

for y in range(image.height): # loop over all the rows

fox x in range(image.width): # loop over all the columns

pixel = image.get\_pixel(x, y)

# do something with pixel

return image