

Source: Tesla

ELEC 474

Prelab One: Software Installation Guide

Guide to installing Anaconda, JupyterLabs, OpenCV, NumPy, and Scikit-Learn

Kevin Huang, Michael Greenspan

Contents

1. Anaconda Installation	2
2. JupyterLab installation	3
3. OpenCV, NumPy, and Scikit-Learn Installation.....	4
3.1 OpenCV	4
3.2 NumPy	4
3.3 Scikit-Learn.....	5
4. JupyterLabs Tutorial.....	6
4.1 Starting a New Project	6
4.2 Importing Libraries.....	7
4.3 Hello World and Image Functions	8
5. Accessing and Modifying Pixels.....	10
5.1 Image Properties.....	10
5.2 Flipping and Image	10
5.3 Negating Image Pixels	11
5.4 Separating Color Channels	12

1. Anaconda Installation

This document will provide the necessary software that will be required to run and use OpenCV in a Python environment. We will be using Anaconda for the use of Python, the link to the download page of which is <https://www.anaconda.com/products/individual>. There will a greeting page with a button that says **Download**. Click this button and you will greeted with the following installation page:



Choose the version that is compatible with your computer. Most people with a reasonably current Windows PC should choose the **64-Bit Graphical Installer** version.

Once the installer has downloaded, open the **.exe** file to launch the installer, and follow these steps within the installer:

- 1) Press **Next** on the intro screen and **I Agree** with the license agreement.
- 2) Choose the option which suits your computer's use. (This setup will use the **Just Me** option. If you choose the other option, then Anaconda will be installed in **Program Files** and require admin privileges.)
- 3) You will be greeted with an installation location. If you are okay with this location, click **Next**; otherwise, click **Browse** and navigate to your desired installation folder.
- 4) Under the *Advanced Installation Options* box, leave **"Add Anaconda3 to my PATH environment variable"** **unchecked** and **"Register Anaconda3 as my default Python 3.8"** **checked**.
- 5) Click **Install** to proceed with the installation, which should be fairly quick (about two or three minutes). Once done, click **Next, Next**, and then click **Finish**.

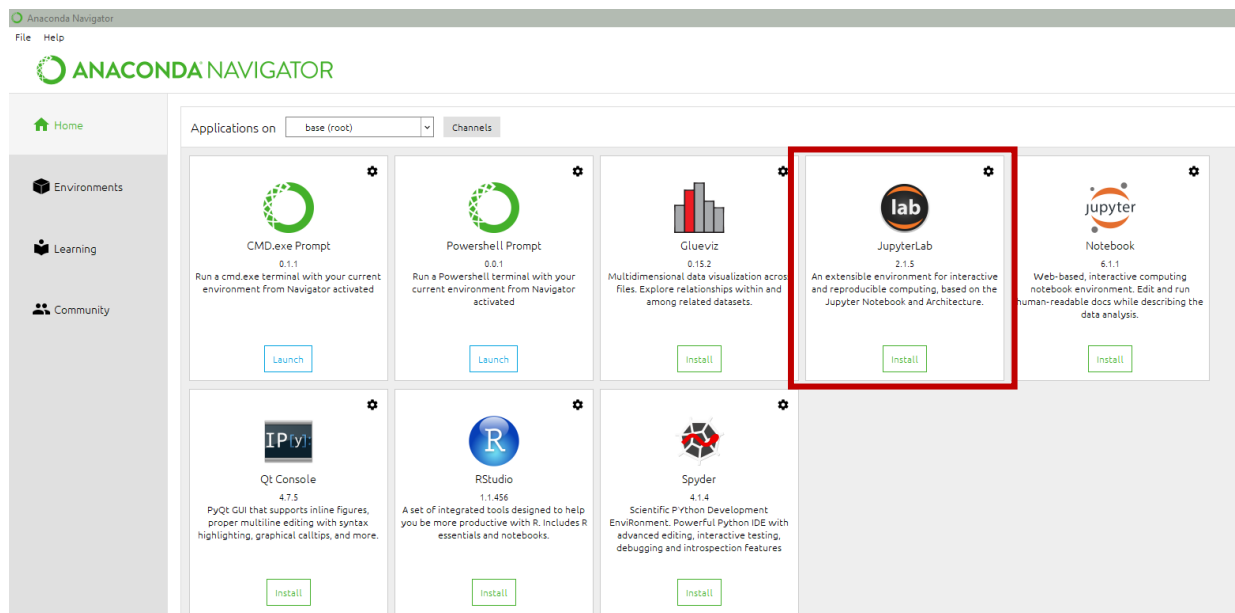
Anaconda will now be installed on your computer.

2. JupyterLab installation

The next installation will be *JupyterLab*, which will provide a flexible web-based Python development environment with text editors, terminals, and custom components. This can be installed directly from Anaconda's Navigator application which you previously installed. You can either navigate to your anaconda folder from the windows start menu, or you can type **anaconda** into your windows search bar.

(The first time that you open Anaconda Navigator, you'll be presented with a *Thanks for installing* box, with a few boxes to select to choose whether or not to send anonymized user data back to Anaconda.org. You can either check or uncheck these boxes, as you see fit.)

Once Anaconda Navigator has opened, you will be greeted with this screen:



Click the **Install** button for *JupyterLab* (highlighted in red above). Wait a second or so for the installation to finish, after which you'll be returned to the main Anaconda Navigator homepage. You'll notice now that the JupyterLab entry has an associated **Launch** button.

Close Anaconda Navigator (**File->Quit**, or **CRTL-Q**).

3. OpenCV, NumPy, and Scikit-Learn Installation

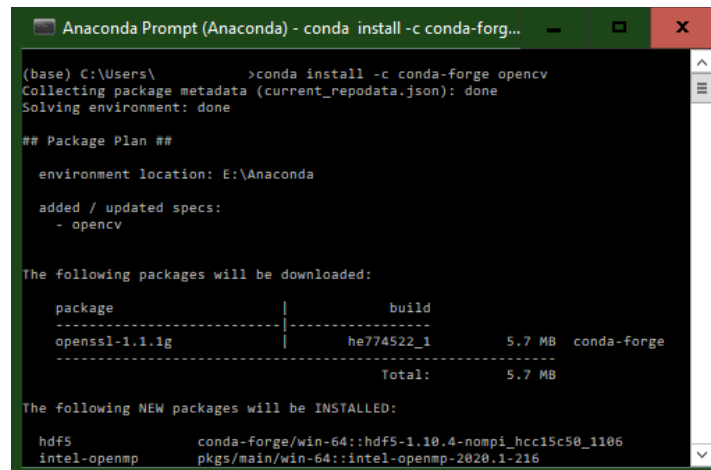
3.1 OpenCV

To install OpenCV for Python, Anaconda provides the *Anaconda Prompt* application where you can input a command to install OpenCV. There are a few ways to launch *Anaconda Prompt*:

- From the Windows Start Menu, select **Anaconda Powershell Prompt**
- From the Windows search bar, enter **Anaconda Prompt**
- From Anaconda Navigator, select **Anaconda Powershell Prompt**
- From Anaconda Navigator, select **CMD.exe Prompt**

Once the prompt opens, type the following command and press enter:

```
conda install -c conda-forge opencv
```



```

Anaconda Prompt (Anaconda) - conda install -c conda-forg...
(base) C:\Users\ >conda install -c conda-forge opencv
Collecting package metadata (current_repodata.json): done
Solving environment: done

## Package Plan ##

  environment location: E:\Anaconda

added / updated specs:
- opencv

The following packages will be downloaded:

package | build | size | source
-----|-----|-----|-----
opencv | 4.5.1 | 5.7 MB | conda-forge
Total: 5.7 MB

The following NEW packages will be INSTALLED:

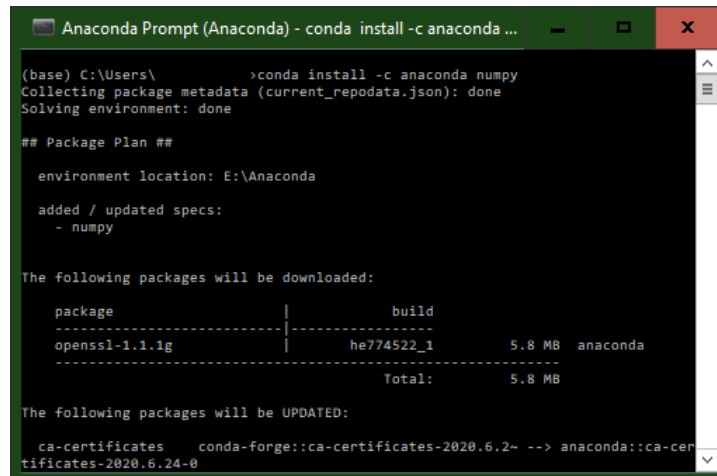
hdf5 | conda-forge/win-64::hdf5-1.10.4-nompi_hcc15c50_1106
intel-openmp | pkgs/main/win-64::intel-openmp-2020.1-216
  
```

The command will load the necessary packages and the prompt will then ask you to enter **yes** or **no (y/n)** to proceed with installing the packages. When prompted, enter **y** to install, which should take only a few seconds.

3.2 NumPy

NumPy provides very useful numerical and array operations for Python and OpenCV. As was the case with OpenCV, you can easily install NumPy in the Anaconda Prompt application, using the command:

```
conda install -c anaconda numpy
```



```

Anaconda Prompt (Anaconda) - conda install -c anaconda numpy
(base) C:\Users\ >conda install -c anaconda numpy
Collecting package metadata (current_repodata.json): done
Solving environment: done

## Package Plan ##

  environment location: E:\Anaconda

  added / updated specs:
    - numpy

The following packages will be downloaded:

  package | build | size | channel
  -----|-----|-----|-----
  openssl-1.1.1g | he774522_1 | 5.8 MB | anaconda
  -----|-----|-----|-----
  Total: | | 5.8 MB |

The following packages will be UPDATED:

  ca-certificates conda-forge::ca-certificates-2020.6.2- --> anaconda::ca-certificates-2020.6.24-0

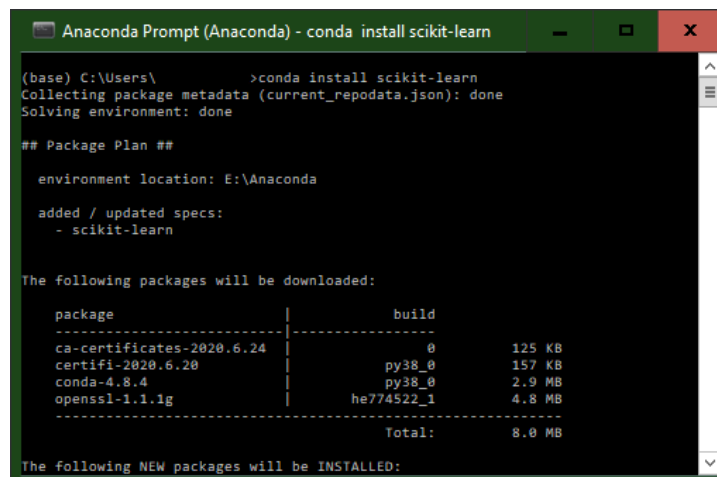
```

Press **enter** to execute the command and it will load the necessary packages and the prompt will ask you **yes or no (y/n)** to proceed with installing the packages. Enter **y** to install, which takes a few seconds.

3.3 Scikit-Learn

Scikit-Learn is a data science and machine learning library and will be used in the machine learning portion of this course. This library can be installed from the Anaconda Prompt with the command:

conda install scikit-learn



```

Anaconda Prompt (Anaconda) - conda install scikit-learn
(base) C:\Users\ >conda install scikit-learn
Collecting package metadata (current_repodata.json): done
Solving environment: done

## Package Plan ##

  environment location: E:\Anaconda

  added / updated specs:
    - scikit-learn

The following packages will be downloaded:

  package | build | size | channel
  -----|-----|-----|-----
  ca-certificates-2020.6.24 | 0 | 125 KB |
  certifi-2020.6.20 | py38_0 | 157 KB |
  conda-4.8.4 | py38_0 | 2.9 MB |
  openssl-1.1.1g | he774522_1 | 4.8 MB |
  -----|-----|-----|-----
  Total: | | 8.0 MB |

The following NEW packages will be INSTALLED:

```

Press **enter** to execute the command and it will load the necessary packages and the prompt will ask you **yes or no (y/n)** to proceed with installing the packages. Enter **y** to install, which takes a few seconds.

To exit the Anaconda Prompt application, in the console type **exit** and press enter.

You now have all of the necessary software packages for ELEC 474.

4. JupyterLabs Tutorial

4.1 Starting a New Project

This section will outline a brief tutorial to the JupyterLabs environment as well as a quick “Hello World” software test. To begin, create a new directory to store your demo’s files in Anaconda Prompt. You can use this command to create a new directory:

```
mkdir <name>
```

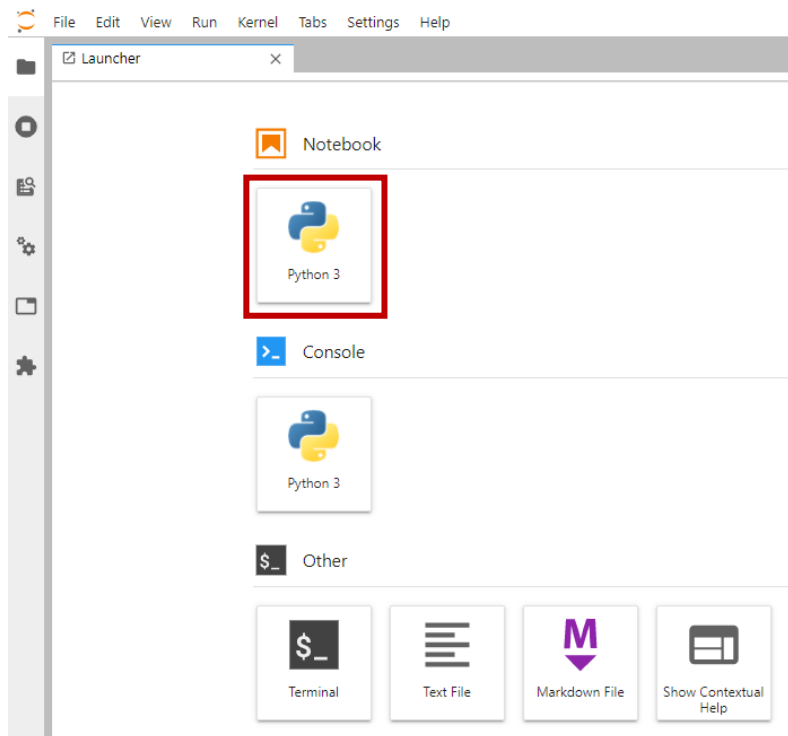
Type a desired name and press enter, navigate to this folder in the prompt by using:

```
dir <name>
```

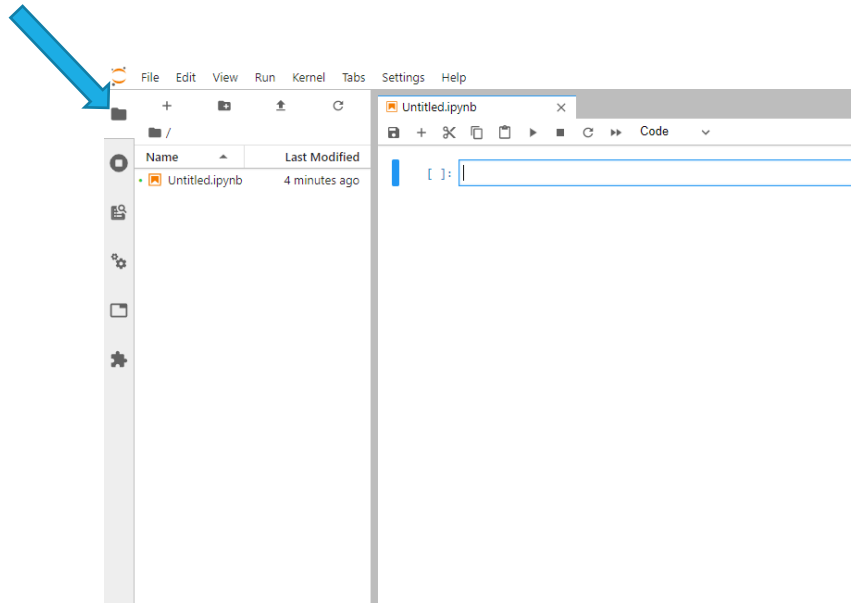
Once you are in your desired folder, you can now open JupyterLab to this folder in Anaconda Prompt using:

```
Jupyter lab
```

This command will open JupyterLab in the desired folder on your default web browser. Open a new Notebook by clicking on the **Notebook->Python 3** button:



This will open a new *Jupyter Notebook* where you can write your Python code. Click on the top left folder to open a view of the directory:



You can rename your notebook by right-clicking the .ipynb file and selecting the Rename option. Included in this tutorial is the standard test image for image processing, *Lena*. Drag and drop this image into the folder section to include it in your workspace.

On the right side of the notebook you can now write Python code. Jupyter Notebooks uses a code cell configuration where you can type multiple code segments and run them separately or simultaneously (more information about this structure is provided on Jupyter's own documentation, <https://Jupyter-notebook.readthedocs.io/en/stable/notebook.html>).

4.2 Importing Libraries

Let's start by importing the necessary libraries and check their versions. Type this code segment into the code cell:

```
import cv2
from matplotlib import pyplot as plt
import numpy as np
import sklearn

print(cv2.__version__)
print(np.__version__)
print(sklearn.__version__)
```

This code will import the necessary libraries required for OpenCV, NumPy for OpenCV array operations, and matplotlib (which is pre-installed) for displaying images in OpenCV. The code also imports Scikit-learn but for this setup tutorial, Scikit-learn will only be checked for its version. The code will also print the versions for both OpenCV, NumPy, and Scikit-learn.

You can run this code cell, first make sure that the cell is selected, by clicking the mouse anywhere within the cell. Once selected, the cell will display a blue bar on the margin. Once selected, the code in the cell can be executed a number of ways:

- Using the play button in the toolbar;
- Using the keyboard shortcut, Shift+Enter;
- From the **Run->Run Selected Cells** tab.

Once this cell has been run, your Notebook should look like this:

```
[3]: import cv2
      from matplotlib import pyplot as plt
      import numpy as np
      import sklearn

      print(cv2.__version__)
      print(np.__version__)
      print(sklearn.__version__)

4.0.1
1.18.5
0.23.1
```

The versions printed and displayed when run shows that OpenCV, NumPy, and Scikit-learn have been successfully installed.

4.3 Hello World and Image Functions

Now let's do a simple hello world program with some of OpenCV's image loading, type conversion, and display methods. Type this code segment into the second code cell:

```
print("Hello World!")
img = cv2.imread("lena.png")
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
```

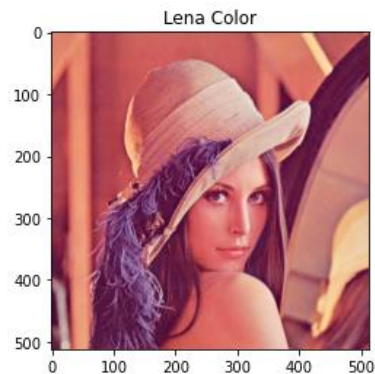
The first line is straight forward as we are simply printing a string of characters. The second line uses OpenCV's ***imread*** function and loads into the ***img*** variable the image of lena. The third line uses OpenCV's ***cvtColor*** function which is used to convert an image's color space to another. In this case the first argument is the ***img*** variable which is a BGR (blue, green, red) color image of lena and ***cvtColor*** will convert it to a grayscale image by specifying the color conversion code ***COLOR_BGR2GRAY***. Press the play button or Shift+Enter to execute this code cell. You should see ***Hello World!*** printed.

Now let's display our original and grayscale images. Into the next cell, type the following statements:

```
plt.imshow(cv2.cvtColor(img, cv2.COLOR_BGR2RGB))
plt.title("Lena Color")
```

This code segment uses matplotlib ***imshow*** function to display the RGB image of lena. Note that ***cvtColor*** is called to convert ***img*** to RGB (red, green, blue): this is because OpenCV by default loads images in BGR mode and matplotlib by defaults displays images in RGB mode. The second line simply adds a title to the plot. Press the play button or Shift+Enter to execute this code cell. Your output should look like this:

```
[5]: plt.imshow(cv2.cvtColor(img, cv2.COLOR_BGR2RGB))  
plt.title("Lena Color")  
[5]: Text(0.5, 1.0, 'Lena Color')
```

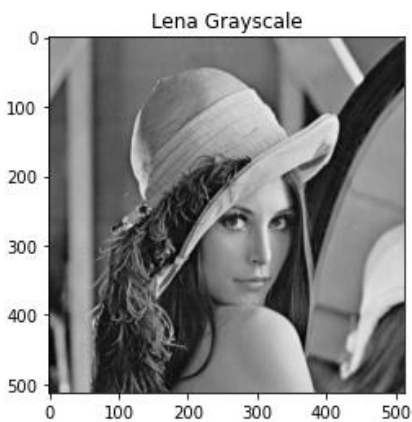


Next you can display the grayscale image:

```
plt.imshow(gray, cmap = 'gray')  
plt.title("Lena Grayscale")
```

This code segment is very similar to above but with the only slight difference being that this plot's color map (cmap) is set to grayscale instead of its default RGB color map. Again, by pressing the play button or Shift+Enter your output should look like this:

```
[6]: plt.imshow(gray, cmap = 'gray')  
plt.title("Lena Grayscale")  
[6]: Text(0.5, 1.0, 'Lena Grayscale')
```



5. Accessing and Modifying Pixels

In this section you will implement code to access and modify pixels. Accessing and modifying pixels allows for the manipulation of images and can be used to perform operations such as flipping, negating, and separating the image to each of its color channels.

5.1 Image Properties

We can first start with accessing an image's properties.

There are three important function you can use to access image properties, these consist of:

```
print(img.shape)
print(img.size)
print(img.dtype)
```

press Shift+Enter and your output will be:

```
print (img.shape)
print (img.size)
print (img.dtype)

(512, 512, 3)
786432
uint8
```

The `shape` property represents the image's dimensions and channel size. Its format is: *rows*, *columns*, and *channels*. If the image is grayscale, then there is only one channel, so only the *rows* and *columns* properties are output.

The `size` property represents the total number of pixels in the image.

The `dtype` property is the image's data type. It is often important to know this property, as a large number of errors in OpenCV can result from incorrect use of the data types.

With this information, you can now implement a few image modifications based on pixel access.

5.2 Flipping and Image

For this you will need to implement code to:

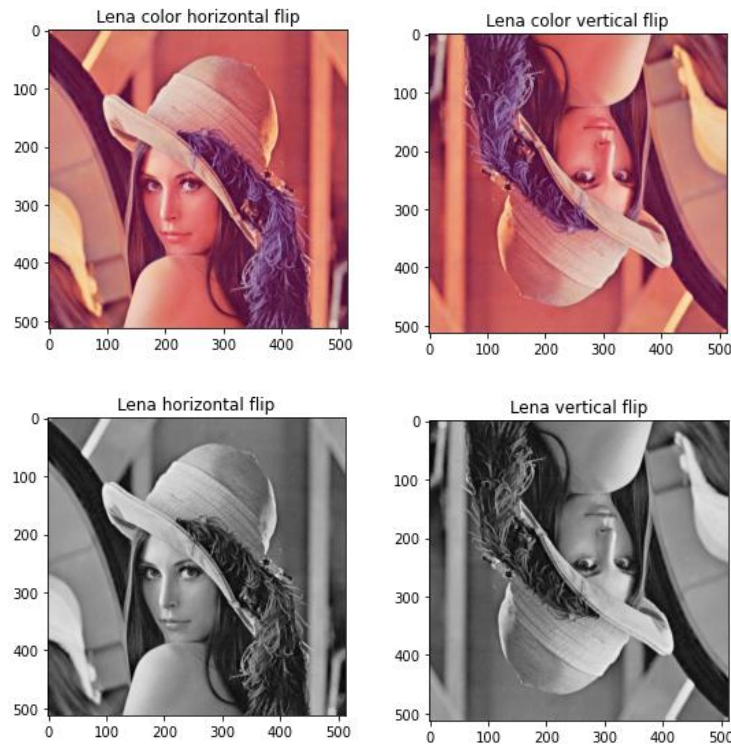
- 1) Loop through an image's pixels
- 2) Place pixels in their corresponding flipped (horizontal and vertical) positions
- 3) Output the image in Jupyter Labs

Here are some things you should consider:

- Use `shape` to access the image's row and column sizes
- Create an empty image for the flipped pixels that has the same properties (i.e. `shape` and `data type`) as the input image

- Use *for loops* to loop through the image's pixels, and place the pixels in their corresponding flipped positions

Flip both of the color and gray lena images, first horizontally and then vertically. Your output results should look like this:



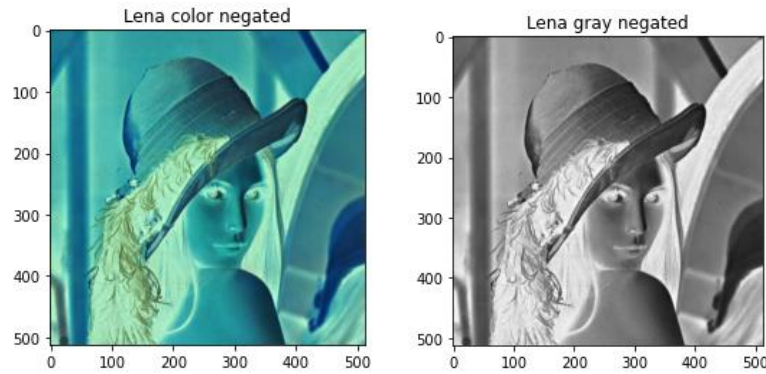
5.3 Negating Image Pixels

For this you will need to implement

- 1) Loop through each image pixel
- 2) Negate each pixels based on its **data type**
- 3) Output the negated image in Jupyter Lab

It is important to note the image's **data type**. Lena's data type is **uint8**, which is an unsigned integer of 8 bits, with values in the range of [0, 255]. Be sure to take this into account when negating your image.

Negate both the color and grayscale images of lena. Your outputs should look like this:



5.4 Separating Color Channels

For this you will need to implement

- 1) Loop through color image pixels
- 2) Place into three different images: Blue channel pixels, green channel pixels, red channel pixels
- 3) Output the channel separated images in Jupyter Lab
- 4) Merge the separated images

Here are some things you should consider:

- Create three empty color images of the original color image's shape and size for visualization
- Be sure to access the correct color channel - Remember, OpenCV loads images in BGR format

Your output should look like this:

Three color channel images:



These three images can then be merged by creating an empty color image and with correct channel and pixel manipulation your merged output should be the original color image:

