# A11: PPM

* Assignment A11 should be completed individually or with one partner.
* If you work with a partner, be sure to follow good pair programming practices.
* Be sure to read the entire prompt and understand the problem before beginning coding.
* In this assignment, we will be working with classes as well as PPM images. You may need to install a program to view PPM images on a Windows machine. I recommend installing [Irfanview](http://www.irfanview.com/).

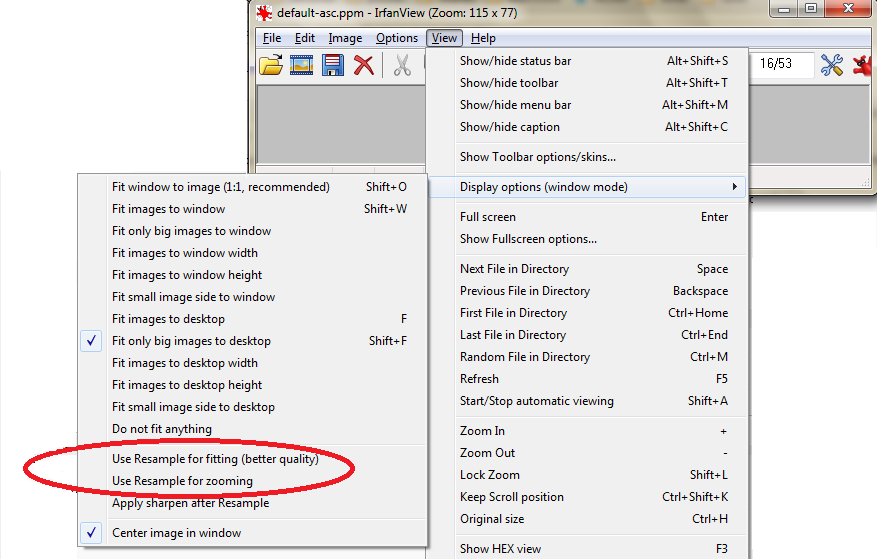
## Learning Objectives

* Navigating large code bases
* Working with classes, particularly creating new methods and calling them
* Explore different image encoding methods, and understand how computers represent images

## Screen Pixels

A **pixel** is the smallest addressable element in an image. The word pixel is based on a contraction of pix ("pictures") and el (for "element"). We will work with some very small images like: small 8x10 image. Using the zoom feature of [Irfanview](http://www.irfanview.com/) we can see the pixels of the same image to the right where zoomed in you can see it is 8 pixels wide and 10 pixels tall. The image above is a zoomed-in version of the very small image above.

We recommend that you uncheck the “Use resample” settings in Irfanview as follows:



## Image Colors

Images displayed on a screen use light for the display. Any three colors (or frequencies) of light that produce white light when combined with full intensity are called primary colors of light. The most commonly used set of primary colors of light is the set Red (R), Green (G), and Blue (B). Using a term borrowed from neuroscience, each color is typically called a color channel.

Here is a small widget for your exploration of color channels: <https://yuilibrary.com/yui/docs/color/rgb-slider.html>.

Try various RGB color channel values between 0 and 255.

## The PPM Image Format

PPM stands for "Portable Pixel Map." PPM files are an image format which is easy to view as text (using Notepad, for instance.) They are also easy to work with in code. But, because they are completely uncompressed, the files become large quickly. Most modern image formats use some kind of compression to make their size reasonable, while preserving the image appearance. However PPMs do still serve a purpose--one modern use for PPM is an intermediate format when converting images from one type to another.

We will be using the "plain" PPM-P3 format. Here an example plain PPM-P3 of the image shown above:

P3

# Created by Dr. Scott Heggen

8 10

255

140 140 140 120 120 120 100 100 100 80 80 80 60 60 60 40 40 40 20 20 20 0 0 0

120 120 120 63 72 204 63 72 204 63 72 204 63 72 204 252 252 255 255 255 255 15 15 15

105 105 105 255 255 255 63 72 204 255 255 255 63 72 204 255 255 255 255 255 255 30 30 30

90 90 90 255 255 255 63 72 204 63 72 204 63 72 204 255 255 255 255 255 255 45 45 45

75 75 75 255 255 255 63 72 204 255 255 255 63 72 204 63 72 204 63 72 204 60 60 60

60 60 60 63 72 204 63 72 204 63 72 204 63 72 204 255 255 255 63 72 204 75 75 75

45 45 45 255 255 255 255 255 255 63 72 204 255 255 255 254 254 254 255 255 255 90 90 90

30 30 30 255 255 255 255 255 255 63 72 204 255 255 255 255 255 255 63 72 204 105 105 105

15 15 15 252 252 253 255 255 255 63 72 204 63 72 204 63 72 204 63 72 204 120 120 120

0 0 0 20 20 20 40 40 40 60 60 60 80 80 80 100 100 100 120 120 120 140 140 140

[Irfanview](http://www.irfanview.com/) will allow you to convert your own images to PPM so you can practice with your own pictures, but keep in mind that you will need to make them small or the resulting PPM will be very large!

## Image Header

You can think of the image as having two parts, a header and a body. The header consists of the following entries:

P3

# Created by Dr. Scott Heggen

8 10

255

P3 is a "magic number". It indicates what type of PPM file it is. P3 indicates full color with ASCII encoding. For this assignment, it will always be P3.

A comment line is typical after the magic number. In this example, the comment is “Created by Dr. Scott Heggen”.

Next comes the width (or number of columns) and the height (or number of rows) in the image (8w x 10h).

Finally, we have the maximum color intensity of 255, which is a common value for the magic number.

The way you see the header presented is how it should be spaced out.

### Image Body

The image body contains the actual color information. Each pixel of the image is a tiny, colored rectangle whose color is determined by how much Red (R), Green (G), and Blue (B) are present in that order. So, 63 72 204 is the first color of the image, which is the instructor's version of Berea blue. Other pixels in the image are 255 255 255, which is white, and 0 0 0 which is black. By varying the levels of the RGB values you can come up with any color.

Note that color values must be separated by a space, but after that additional white space is ignored by the image viewer. In the sample ppm above we used line breaks to format the image so that it is easy for a human to understand.

## CRC Card for the PPM class

We have created a class which can be used to manipulate and display PPM-P3 images. This is a larger class than we have seen before, so the CRC card is larger too, but like functional decomposition, class decomposition makes it easier to focus on a single method at a time.

|  |  |
| --- | --- |
| **Class name: PPM** | |
| **Class Attributes:** | **Class Collaborations (other classes):** |
| * **self.root**:   + Provided master Tkinter instance created using helper function PPM\_set\_up() * **self.inasciifile:**   + String which represents filename is used only for reading the provided PPM-P3 as input * **self.outasciifile:**   + String of filename of human readable modifications to the PPM file * **self.title:**   + Used for the title of the display window * **self.magic:**   + PPM file type is often called the "magic number." It needs to be P3 to be readable * **self.comment:**   + Creates a comment for the PPM file * **self.width:**   + Image width in pixels * **self.height:**   + Image height in pixels * **self.colormax:**   + Should be set to 255 * **self.ascii:**   + Will store the color intensities in P3 format * **self.pixellist:**   + Will store nested list containing pixel colors * **self.image:**   + Reference to image window * **self.label:**   + Used to place image in window | * global **tkintertoggle** is needed as global to ensure a single Tkinter instance which is needed to render images. |
| **Class Methods:** | **Class Collaborations (other classes):** |
| * **\_\_init\_\_()**:   + Initializer/constructor for the class, opening a provided PPM-P3 file and setting all member attributes. * **PPM\_makeoutputfiles():**   + Given self.inasciifile, sets self.ascii and creates both ascii and binary files for output * **PPM\_partition():**   + Given input parameter strng, the string to partition and ch, the character to use as the delimiter returns a triple with all characters before the delimiter, the delimiter itself if present and all of the characters after the delimiter (if any). * **PPM\_clean():** Removes all single line comments, whitespace, and newline characters present in the input parameter string strng. * **PPM\_load():**   + Takes string input parameter inasciifile as the name of the ASCII PPM-P3 (non-binary) file to load. * **PPM\_makepixellist():**   + Creates self.pixellist, a nested list of rows of [red, green, blue] pixels from input color\_list which contains an unnested list of strings. * **PPM\_updatefrompixellist():**   + Updates image object data and related files from input pixellist * **PPM\_set\_title():**   + Setter for title of display window. * **PPM\_make\_red()**:   + Colorizes current image to red by using self.pixellist. * **PPM\_grayscale():**   + TODO FIX ME: Changes the picture into a grayscale image. * **PPM\_flip\_horizontal():**   + TODO FIX ME: Flips image horizontally. * **PPM\_rotateclockwise():**   + TODO FIX ME: Rotates image clockwise. * TODO FIX ME: write at least one additional PPM class method | * **PPM\_set\_up():**   + A helper function which must be called at the beginning of any program which uses the PPM class, but it is not a part of the class. * **PPM\_render():**   + A helper function which renders all PPM images. It is not part of the class. * **PPM\_Exception:**   + A Python class which enables meaningful error messages on exceptions. |

#### 

## The instructionsBC sign

We have created a ppm.py module which contains the PPM class and works with PPM images. Download all of the following files and put them into a single folder:

* [ppm.py](https://drive.google.com/open?id=0B0J8Yj0B6KRSd0NJeHlNZk5fTU0)
* [a11\_ppm.py](https://drive.google.com/open?id=0B0J8Yj0B6KRSd0I3cXNFSDA2aW8)
* [default.ppm](https://drive.google.com/open?id=0B0J8Yj0B6KRSWlhPbGJ5ckRKcXc)
* [bc-sign.ppm](https://drive.google.com/open?id=0B0J8Yj0B6KRSOS0xLUY4Y01yYk0)
* [bc-flowers.ppm](https://drive.google.com/open?id=0B0J8Yj0B6KRSU09hT1gwUlpiV28)

You will be extending the PPM class, and testing your extensions.

**Hint: Be careful to remember that making copies of a data structure which is a list of rows which are lists which is a list of color pixels (i.e., a list of lists of lists) will require you to make deep copies.**

The user will specify the name of the image file. The file needs to be a text file in PPM format as described in the discussion above.

Add the following methods into the PPM class:

* PPM\_grayscale(self):
  + Changes the picture from color to grayscale. This is done by averaging the values for red, green, and blue for each pixel. The new value of each pixel is the average of the three values for that original color pixel. For example, if the three colors were 10, 40, and 200, the average would be 83.3333. Truncate that to an integer: 83.
* PPM\_flip\_horizontal(self):
  + Flip the image on the horizontal axis. After the flip, the far right end pixels will be on the far left, and vice versa. Be sure to preserve RGB order.
* PPM\_rotate\_clockwise(self):
  + Rotate the image clockwise 90 degrees. This one is a bit tricky because you have to make a deep copy that is sideways.
* At least one additional method. Consider such ideas as negative, add a frame, turn sepia, colorize, add noise, extreme contrast, soft blur, or other functions which appeal to you. Be sure to explain these in the docstring.

Create new PPM objects in the **a11\_ppm.py** file, and call your new functions. Be sure to display and save the output.

## Requirements:

* Be sure to include an appropriate docstring for each of your methods and to add informative comments in both files.
* Be sure to modify the standard header at the top of your program with name, username, assignment number, purpose and acknowledgements.
* The highest level of the program (i.e., no indenting) **must only** contain the following:
  + The header
  + Any import statements
  + Class definitions
  + Function definitions
  + A call to the main() function

## Submission Instructions

1. Review the requirements above to ensure you have completed everything that was required of you.
2. Save your code as **A11\_ppm\_*username*.py and ppm\_username.py**. Replace *username* with your Berea usernames. For example, the TA Bianca Marrero’s file would be **A11\_ppm\_marrerob.py and ppm\_marrerob.py.** Be sure to fix imports so your code works!
3. Zip the code files together.
4. Upload the Zip file to Moodle by the due date listed on the course website: <https://trello.com/b/w7bIrLoV/>.
5. If you worked with a partner, your partner should upload a file named **A11\_ppm\_*usernames*.txt** and include both partner’s name in the document.