

Activity 1.4.4 *Python* Imaging Library API

Introduction

|  |  |
| --- | --- |
| You’ve learned the basics of writing computer programs. But most programming builds on code that has already been written. Knowing how to find and use code from other people will help make you an efficient and successful software developer.  You could create your own algorithms, for example, to rotate an image or to identify the objects in an image. But others have already solved those problems! There are many advantages to using existing code. You save time, of course. But you also connect to a community of people, making it easier for them to help you, and making it more likely they will be able to use what you create. How will you put other people’s code to use? | amusement parks,cartoons,entertainment,fat,funhouses,leisure,mirrors,people,places,recreation,thin |

Materials

* Computer with Enthought Canopy distribution of the *Python®* programming language
* Webcam or other way to capture a digital picture
* Image files and source files in *Python* for Activity 1.4.4

Procedure

1. Form pairs as directed by your teacher. Meet or greet each other to practice professional skills. Set team norms.
2. Launch Canopy. Open an editor window. Set the working directory to your folder. Create a new *Python* file. Save the file as JDoe\_JSmith\_1\_4\_4.py.
3. Some of your work in this assignment will involve trial and error using the iPython session. You might want a record of commands you have tried, so begin logging the session.

In []: %logstart -ort studentNames\_1\_4\_4.log

1. In an earlier activity, you learned how to use nested for loops to conditionally change color values of individual pixels. You did this using code from the matplotlib  and numpy libraries. You used statements like img1[r][c] = [255, 0, 255] to change the pixels in part of an image. To cause an image’s sky to turn pink, for example, the statement img1[r][c] = [255, 0, 255] was inside an if block. The if structure was inside a for loop, which was inside another for loop. Explain the purpose of these three structures (the outer for, the inner for, and the if) in the algorithm used to turn the sky pink.
2. A software developer’s job usually includes learning how to use other people’s code. It is often much quicker to find and learn how to use code that someone else has created than it is to create the code on your own. We will use code from the PIL, the Python Imaging Library, to perform image manipulations.

Recall that an **API**, or application programming interface, tells you how to use someone else’s code. PIL, like matplotlib, is an **object-oriented** library of code. Object-oriented code creates and manipulates objects, which are data structures defined in the code. PIL’s objects include images. The matplotlib library’s objects include figures and axes.

For object-oriented code, the API lists the **attributes** and **methods** for each **class**. Almost always, at least one of the methods **instantiates** the class. **Instantiation** means creating an object, also known as an instance, of the class. The instantiation methods that do this are sometimes called **constructors**.

Although you are learning to manipulate images in this activity, a primary objective is to understand how to find, select, and use API documentation. These skills will enable you to learn how to use other people’s code.

We will use nine methods and one attribute from the API for the PIL.Image class. Refer to Table 2 in the Lesson 1.4 Reference Card. In the following table, complete the nine rows that have been started. The empty row near the bottom is used in the next step.

|  |  |
| --- | --- |
| **Method** | **Short Description** |
| open() |  |
| new() |  |
| crop() | makes a PIL.Image of smaller size |
| convert() |  |
| resize() |  |
| rotate() |  |
| paste() |  |
| transform() | makes a PIL.Image by distorting an image |
| save() |  |
| Show() | Displays and image intended for debugging. |
| blend() | Interpolating an image with another and a makes a constant alpha |
| Fromstring() | Creates an image memory from a pixel data in a string as “raw” |
| **Attribute** | **Short Description** |
| **mode** | **The mode of the image as a string** |
| size | width, height |

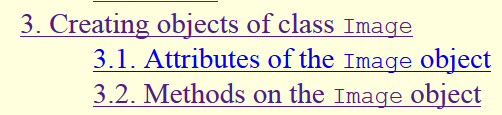
1. There are four kinds of documentation:

* Official documentation
* Third-party documentation
* Tutorials
* Problem/solution-specific documentation, often in social formats such as Stack Overflow

The official documentation is usually among the best references for using a library of code. Official documentation is produced by the person or group who created the code. Docstrings fall into this category since they are created by the developers while writing comments in their code.

Refer to [http://effbot.org/imagingbook](http://effbot.org/imagingbook/), the official documentation for PIL. Identify three methods of objects in the PIL.Image class that are not yet listed in the table in Step 5. Add the three methods to the table in Step 5. Record both the method name and a short description of the method from the documentation.

1. Another place to learn about a library is from third-party documentation. New Mexico Tech produced some excellent documentation of PIL. Refer to Section 3 of that documentation at <http://infohost.nmt.edu/tcc/help/pubs/pil/>



The attribute size is listed in the table above in Step 5. What is another attribute of every PIL.Image object?

mode

1. You may be able to find tutorials created specifically to teach, as opposed to reference materials created for later reference. Tutorials might be part of official documentation or developed by third parties. Visit <http://effbot.org/imagingbook/introduction.htm>.

How is this different than the reference material linked in Step 6?

In step 6 it only tells you what it does. For this it tells you how it works and in what way you can use it.

1. For this activity you have been provided with the Lesson 1.4 Reference Card and three of the best references available online for PIL. Usually when developers find a module of code to use, they still have to identify useful documentation. Find one additional PIL documentation resource with an Internet search. Good search terms are “PIL,” “Python,” and the name of a method for which you want more information. Another technique is to include an error message in your search terms. Record the URL of a site you select from your search results. Summarize the author’s purpose in terms of the categories of documentation listed in Step 6.

URL: https://pillow.readthedocs.org/

Summary of author’s intent:

They want to tell you how to install it and how to use it. They also want to say how it can be used tell you all the modules.

1. As you complete the rest of this activity, continue to use the various sources of documentation linked above and others as appropriate. Which do you think will be the most useful to you and why?

Effbot.org because it’s the official website. Doing so it has everything and it doesn’t leave anything out.

1. Figure 1 in the Lesson 1.4 Reference Card has reference material about three *Python* code libraries. Based on your teacher’s description, summarize the purpose of these three libraries:

* matplotlib.pyplot (plt) – collection of functions that can create a figure and plot figures graphs etc. (http://matplotlib.org/users/pyplot\_tutorial.html)
* numpy (np) – a fundamental package for scientific computing in python. It can integrate c/C++, has broadcasting function and array objects. (http://www.numpy.org/)
* PIL – Adds image processing capabilities to your python interpreter. It adds file format support and “powerful” image processing capabilities. Fast access to data and stored in few ebasic pixel formats. http://effbot.org/imagingbook/overview.htm

1. Obtain the code and images provided for Activity 1.4.4 in 1.4.4 SourceFiles.zip as directed by your teacher. Open and execute earthEyes.py. This program should display three figures as shown below. The program pastes an image of the Earth over one eye in an image of a student.

The figures below are each labeled with three variables from earthEyes.py. The first variable in each case is a matplotlib Figure. The second and third variables are the AxesSubplot objects.

|  |  |  |
| --- | --- | --- |
| fig | fig2 | fig3 |
| axes[0] axes[1] | axes2[0] axes2[1] | axes3[0] axes3[1] |
|  |  |  |

To bring the figures to the front of the Windows environment, click on pyplot icons in the Windows taskbar.



Examine all three figures on your screen. The three figures show the identification of the eye’s coordinates, the resizing of the Earth’s image, and the pasting of the Earth on top of the eye.

1. The code creating the figure above on the left is shown below.

|  |  |
| --- | --- |
| 14  15  16  17  18  19  20  21  22  23  24 | *# Open and show the student image in a new Figure window*  student\_img = PIL.Image.open(student\_file)  fig, ax = plt.subplots(1, 2)  ax[0].imshow(student\_img, interpolation='none')  *# Display student in second axes and set window to the right eye*  ax[1].imshow(student\_img, interpolation='none')  ax[1].set\_xticks(range(1050, 1410, 100))  ax[1].set\_xlim(1050, 1400) *# Measure in plt, experiment in iPython*  ax[1].set\_ylim(1100, 850)  fig.show() |

1. Line 15 calls the function open() from the PIL.Image library. The function is being called with one argument: student\_file. The function returns one object, which is being assigned to student\_img.

Practice using this vocabulary by describing line 16: Line 16 calls the function \_\_\_subplots\_\_\_\_\_\_ from the \_\_\_matplotlib.pyplot\_\_\_\_ library. The function is being called with \_2\_ argument(s): \_\_\_1, 2\_\_\_\_\_\_\_\_\_. The function returns \_\_1\_\_ object(s), which is/are being assigned to \_\_\_\_\_axes\_\_\_\_\_\_\_\_.

1. In line 17 the imshow() method is called on the object ax[0]. Recite to your partner the methods called on each object in lines 20-24:

Line 17 calls \_\_imshow()\_ on \_\_\_ax[0]\_\_\_\_

Line 20 calls \_\_imshow()\_\_\_\_\_\_\_\_ on \_\_\_\_\_ax[1]\_\_\_\_\_\_

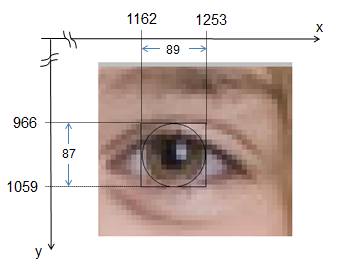
Line 21 calls \_\_\_set\_xticks()\_\_\_\_\_\_\_ on \_\_\_ax[1]\_\_\_\_\_\_\_\_

Line 22 calls \_\_\_set\_xlim()\_\_\_\_\_\_\_ on \_\_\_ax[1]\_\_\_\_\_\_\_\_

Line 23 calls \_\_\_set\_ylim()\_\_\_\_\_\_\_ on \_\_\_ax[1]\_\_\_\_\_\_\_\_

Line 24 calls \_\_\_show()\_\_\_\_\_\_\_ on \_\_\_\_figure\_\_\_\_\_\_\_

1. Lines 21-23 change the axes displaying the single right eye in the figure above. This figure was used to identify the upper left coordinates (1162, 966) of a **bounding box** containing the eye. A bounding box is a rectangle containing a particular part of an image. The bounding box for the right eye’s **iris** (the colored part of the eye) is illustrated below.

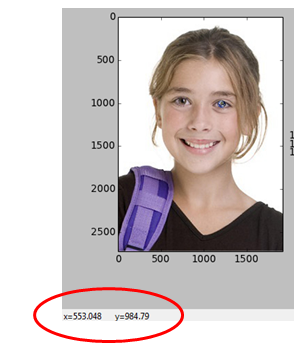


What are the (x, y) coordinates of the upper left corner of the bounding box?

1050,1100

1. In the next several steps, you will use *Python* to paste an image of the Earth over the other eye in the image of the student. We will call this the left eye of the image, though it is the student’s right eye. First, identify the coordinates of the iris of the image’s left eye.
2. Mouse over the eye and take note of the coordinates. 700-800 Y

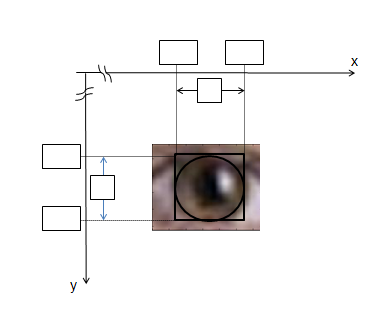
920-1020 X



1. Zoom in on the image’s left eye by using the figure window’s pan/zoom mode.
2. The pan/zoom mode can be toggled on and off using the pan/zoom button.



1. When pan/zoom mode is on, drag with the left mouse button to pan. Drag with the right mouse button to zoom.
2. In the following figure, record the coordinates of the upper left and lower right for the bounding box of the iris for the image’s left eye. Also record the width and height of the bounding box. 920,800



1. The following part of the source code creates the figure that shows two Earths. Read and analyze lines 26 - 33 of the code.

|  |  |
| --- | --- |
| 26  27  28  29  30  31  32  33 | *# Open, resize, and display earth*  earth\_file = os.path.join(directory, 'earth.png')  earth\_img = PIL.Image.open(earth\_file)  earth\_small = earth\_img.resize((89, 87)) *# w and h measured in plt*  fig2, axes2 = plt.subplots(1, 2)  axes2[0].imshow(earth\_img)  axes2[1].imshow(earth\_small)  fig2.show() |

* 1. Line 27 uses the join() method from the os.path module. It is being passed \_2\_\_ arguments. The value it returns is being assigned to the variable \_\_\_\_earth\_file\_\_\_\_\_\_\_\_\_\_.
  2. In line 28 the open() function of the PIL.Image module returns a new PIL.Image object, which is being assigned to the variable \_\_earth\_img\_\_\_\_\_\_\_\_\_\_.
  3. In line 29 the resize() method takes only one argument: a 2-tuple. Explain why there are two sets of parentheses in this line.

Because the resize is a parantheses and inside is the size

* 1. Refer to the bounding box shown in Step 13c. Explain the purpose of the (89, 87) argument in line 29.

It says it’s resized to 89x87

* 1. Practice describing code with your partner by reciting the appropriate one of the following two sentences for each line of code in lines 30-33:

Pick one (function or method) for each line:

* Line \_30\_ calls the function \_\_\_\_subplots\_\_\_\_\_ from the \_\_matplotlib.pyplot\_\_\_\_\_ library with \_2\_ argument(s): \_\_\_\_1,2\_\_\_\_\_\_\_\_. The function returns \_\_1\_\_ object(s), which is/are being assigned to \_\_\_axes2\_\_\_\_\_\_\_\_\_\_.
* Line \_31\_ calls the method \_\_\_\_imshow\_\_\_\_\_ on the object \_\_earth\_img\_\_\_\_\_ with \_1\_ argument(s): \_\_earth\_img\_\_\_\_\_\_\_\_\_\_.

Line 30 calls the… subplots function

Line 31 calls the… imshow function

Line 32 calls the…imshow function

Line 33 calls the… show function

* 1. Reading documentation can be difficult because the words are unfamiliar, even to the expert. Also, the information you need might be found in more than one place. Be persistent! As an example to show you how to piece together information from different places, refer to both of the following sources of PIL documentation to learn how to reduce the size of an image with the resize() method of PIL.Image objects.

[http://effbot.org/imagingbook](http://effbot.org/imagingbook/)

Lundh, Fredrik. (2008). *Python Imaging Library.*

<http://infohost.nmt.edu/tcc/help/pubs/pil/>

Shipman, John. (2013). *Python Imaging Library.*

* + 1. What is an additional argument that can be passed to the resize() method?

Size & filter

* + 1. What is the default value of that argument?

NEAREST

* + 1. Our code is **downsampling** the image from NASA. Downsampling means using fewer bytes to represent the data. What value of the optional argument is recommended for downsampling?

ANTIALIAS

* 1. Refer again to the two sources of PIL documentation listed above. What is represented by the size attribute of an Image object?

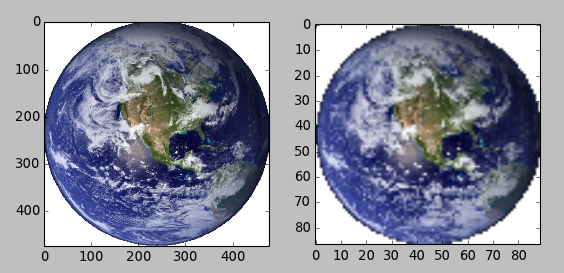
Width and height

* 1. Try the following in the iPython shell and explain the output.

|  |  |
| --- | --- |
| In []:  In []:  In []: | earth\_img.size the size of the picture in the plot is 479 x 475  earth\_small.size the size of the picture in the plot is 89x87  earth\_img.size[1] the size of the earth in the pixels from 479x475 it takes the column |

* 1. Examine the two images in the figure produced by the code. The two images are displayed on the screen using the same number of screen pixels. How can you tell that the two images contain a different number of image pixels?

One is more blurry and one Is way more defined.



1. Explain the algorithm you think resize() might be using. You can explain it using English sentences, pseudocode, commented code, or any other appropriate method. However, your explanation should address details related to the three or four bytes accounting for each pixel in the Image object returned by the resize() method.

It zooms in the picture throwing away some other pixels. The height and width is changed by scrunching it up and the height is squishing it.

1. The following portion of the source code creates the rightmost figure above, fig3, showing the girl and a close-up of her pair of eyes. Read and analyze lines 35 - 44 of the code.

|  |  |
| --- | --- |
| 35  36  37  38  39  40  41  42  43  44 | *# Paste earth into right eye and display*  *# Uses alpha from mask*  student\_img.paste(earth\_small, (1162, 966), mask=earth\_small)  *# Display*  fig3, axes3 = plt.subplots(1, 2)  axes3[0].imshow(student\_img, interpolation='none')  axes3[1].imshow(student\_img, interpolation='none')  axes3[1].set\_xlim(500, 1500)  axes3[1].set\_ylim(1130, 850)  fig3.show() |

1. In line 37 the command refers to two PIL.Image objects:

student\_img is the object on which paste() is being called.

earth\_small is passed as the first argument to the paste() method.

These two PIL.Image objects have different “mode” attributes.

student\_img is RGB, while

earth\_small is RGBA.

As a consequence one image uses three bytes per pixel, while the other image uses four bytes per pixel. Use the iPython session as in Step 15h to determine the size of each image.

Calculate the number of bytes used for the representation of each image, using the following formula. Ignore compression and metadata.

*Image bytes = width • height • bytes/pixel*

student\_img bytes = 15667200

earth\_small bytes = 30972

1. The RGBA file type includes a fourth byte called the alpha channel. The alpha channel controls how transparent or opaque an image is. Only some file types can store an alpha channel. PNG files can store the alpha channel. JPG files cannot. The save() method of PIL.Image recognizes filetype extensions in the filename you provide, so transparency is retained if you name the file with PNG. Save earth\_small.png with the following code in the iPython session.

|  |  |
| --- | --- |
| In []: | earth\_small.save('smallEarth.png') |

1. Examine the file size of these two files in operating system and record the file sizes. Note the other metadata at the bottom of the Windows Explorer window.



student.jpg bytes = 206066

smallEarth.png bytes = 18774

1. Explain the discrepancy between your answers in step b and your answers in step d.

The windows doesn’t count white. Also it is significantly smaller due to loss of quality.

1. Refer to the two sources of PIL documentation listed previously to learn about the  paste()  method of PIL.Image objects. These sources of documentation indicate that the first argument to paste() can be either a color or an image. Here we are using an image. According to the documentation, what happens if a color is used for the first argument?

Same as above, but fills the region with a single colour. The colour is given as a single numerical value for single-band images, and a tuple for multi-band images.

1. The documentation also describes what happens if the modes of two images in step 17b are different from each other. According to the documentation, what happens?

It adds the alpha channel or it retains the alpha channel

1. Explain the purpose of each argument being passed to paste() in line 32.

The earth\_small image is being pasted onto the student\_img on those co-ordinates and the mask of that is the x and y

1. Add lines of code in the code editor to paste the Earth image onto the girl’s other iris. Save your code as directed by your teacher.

student\_img.paste(earth\_small, (700, 935), mask=earth\_small)

1. Save your Python file in the code editor. If you were logging the iPython session, save it with %logstop.

In []: %logstop

Conclusion

1. Describe the classes, methods and attributes used in code in this lesson.

They were using the PIL, Matplotlib.pyplot, and os.path classes. We used the methods such as set\_ylim, imshow, and figure. We were using the student\_img variable earth\_file, earth\_img, and earth\_small.

1. Consider the statement

*Abstraction is a strategy for handling complexity.*

By giving a name to a complex task and encapsulating the details inside that method. Describe a complex computational task you performed in this activity. Describe the abstraction you used from the PIL library that helped you perform this task.

I pasted the earth eye on top of the girls left eye. The abstraction I used was set\_ylim and set\_xlim and that helped me set the X and Y of the small earth that was made using the resize function of the PIL library.