Refactoring COMP:107 with Python in Google Colab

Jasper Giglio

Dr. Pamela Cutter

Computer Science Department

A paper submitted in partial fulfillment of the requirements for the degree of

Bachelor of Arts at Kalamazoo College

November, 2022

Preface and Acknowledgements

Pam, I could not have done this without you and would not be where I am today without your patience and support. Alyce, for introducing me to computer science and asking me to be a TA, and moreover for putting up with me as an advisor for years later. Dr. Vargas-Perez, for the the most culturally and ecologically aware stem class I have ever taken, and always urging me to thingk about multiple problems together. Dr. Barth, for showing me how to import your own Python modules into Colab, and helping me find thins project in the first place. And finally my family for their endless support and encouragement.

Table of Contents

Jasper Giglio

Refactoring COMP:107 with Python in Google Colab

As a computer science major and former teaching assistant, it is important to me to find ways to give back and help aspiring students feel welcomed into the department. That is why I spent my SIP working with Dr. Pam Cutter to refactor the COMP 107: Pictures and Sounds class to use Python. I chose this with the goal of making the class more accessible in a way that helps instill a stronger grasp of fundamental programming skills. While the old software that the class was built on (Jes4py) was effective for beginners, some parts of it were no longer supported by the newer versions of Python. Additionally, teaching a language that is almost never seen outside of one class is not a way to set students up for success. So, I set out to find ways to avoid Jython and do the class work either natively in Python or using more mainstream libraries.

The first thing I did was look at all the assignments and compile a list of the basic functions that would be needed. This got faster as I went along, as most of the later projects are simply new ways of using earlier concepts. Once I had a firm understanding of the required functionally, it was time to research how to achieve those goals. I started my research by reading a few articles Dr. Cutter sent me, which initially led me to think Runestone Academy would have the answers. As part of one of their interactive computer science textbooks, Runestone has an image library that seemed to be capable of the desired manipulations. However, upon further inspection, the library was only usable in

their web app and could not be imported. While this might have been acceptable had we decided to run the class through Runestone, there was no apparent way to view the contents of the image library or its documentation, which severely limits its utility. Another option presented was to use Task Specific Programming (TSP) languages such as Pixel Equations, but since the primary goal was to build transferable foundational skills, it made more sense to use Python.

My research then led me to find Pillow, an updated version of the Python Image Library (PIL). This is the most popular and widely used library for any kind of image manipulation. There were many other sources I looked through, but most of them were either too technical for an introductory class, or too simplified and had no easy way to analyze pixels individually. Once I found the PIL, things started to come together. It was not all simple though; I ran into something unexpected when separating the images channels. Pillow can process several types of images, the most commonly appearing in this class is RGB. This kind of image can be broken down into three arrays, for red, green, and blue, also called channels. The values in the arrays tell you how much of that color is in the full image at that specific location. Luckily I found an easy way to split the arrays, edit one using Numpy, and then merge the new image back together. Unfortunately, whenever I tried to edit one of the single channel arrays, it would no longer be able to merge back together. I tried several ways to work around it, including recasting the array to an integer array, and limiting the values so as not to be out of the range of pixel values, but it would still not merge. Eventually, I figured out a way to access the RGB values of each pixel in a more object oriented way which, while I think it is less than ideal, still serves the purposes needed for this class.

As I moved on to the sound-related part of the class, I started with the Wave library. It has all the utility needed for the purposes of this class, but is greatly lacking in readability. Which is to say that it looks like it was written for people who know how to code, and would only cause unnecessary confusion for students learning If/Else statements for the first time. At first I thought it was perfect as is, but upon taking a second look, the syntax had too many operations that would clutter an untrained mind. For example, if one wants to copy a sound using the wave library, they would have to open the sound, get all its properties, open the new sound in write mode, copy all the properties from the old sound into the new one, and then copy each frame to the new sound, and finally close the new sound. While it is a simple enough process to learn, I/O and system operations are not the point of this class and should be handled automatically.

There was also the question of which development environment and paradigm to use. Previously the class had used Jython- a language that combines the simplicity of Python with the speed of Java. Now that the course was moving to Python, we faced many options. We decided to try the Jupyter notebook format in Google Colab. This was for a few reasons, but primarily we were impressed by the ability to collaborate on projects in real time without needing to be sitting at the same desk. Additionally, the format allows the code to be more readable and interactive, which can make a huge difference in how well students can learn. Notebooks are in my opinion one of the best Computer Science learning tools, since they combine text cells with code, and with some preliminary structure and good documentation they can be as readable as any textbook.

While I have learned volumes about Python and Colab, there are still some areas that give me trouble, most notably when working with sound. The Wave module only allows you to open files as byte strings, which are immutable. This means that in order to edit a sound, one must read the file, convert it to a list, manipulate and then put back into bytes, and finally back into a playable sound. However, in Google Colab, sometimes the operations are too large or have other errors. For example, when I tried to print the bytes from a ten second .wav file, it gave an error message citing config variables. While it is not necessary to show all the 775,000 bytes in a wave file, in order to reverse and otherwise manipulate the sounds it is important to be able to loop through lists of that length. Additionally, a simple function written to reverse the bytes would never complete when executed. Since the same code works flawlessly in VS Code, it seems like there are some default limits to the scale of processing done by the free version of Colab, which may or may not have easy workarounds. A future endeavor would be to more thoroughly isolate these issues.

This project was a great way to learn that not all problems will be solved in the order that you would like. For example, one of the earliest problems I came across became one of the last for which I found a viable solution. Specifically, that is how to open media files in Colab. Since it is an online service, any media you wish to be included in the Python runtime needs to be uploaded in every session. Furthermore, the system dialogs are not easily accessible to open files since you are not technically running the code on your local machine. After scouring the internet and finding many articles with the same solution to a different problem, I finally came across a way to mount your Google Drive to the Colab session, and easily access any files or shared

drives attached to your account. This was one of the many reminders of how both programming and research are nonlinear processes and prosper under rigorous re-evaluation and revision. Despite having found a reliable way to access files, I feel that having a way to access to a local system (e.g. Windows Media Player) would add to the versatility of this class.

I have learned a lot from this project, and while the work is not yet complete, it has been invaluable to experience part of what goes into the creation of a class and how to troubleshoot and solve problems without being given a clearly defined structure or approach. My foremost goal was to let my endeavors lay a framework on which the COMP107 class can be built, and hopefully I have done so in a way which will enable the future classes to feel even more confident in their coding knowledge and transferability of skills. The availability of Google Colab and collaborative potential will make the class much more accessible to all and by using popular libraries like Pillow and Wave, we can set students up for success in future classes or professional settings.

References:

Tenacious39, *How to manipulate the pixel values of an image using Python?* GeeksforGeeks. https://www.geeksforgeeks.org/how-to-manipulate-the-pixel-values-of-an-image-using-python/

Aclark, Stevejohnson, Wiredfool, Hugovk, Radarhere, (2022). *Tutorial*. https://pillow.readthedocs.io/en/stable/handbook/tutorial.html

Guzdial, M. (2021). *Media Computation today: Runestone, Snap!, Python 3, and a Teaspoon Language*. https://computinged.wordpress.com/2021/09/06/media-computation-today-runestone-snap-python-3-and-a-teaspoon-language/

Jsenning (2021). *Jes4py* https://github.com/gordon-cs/JES4py/tree/master/jes4py

Miller, B. (2022). *Using Repitition With Images*. https://runestone.academy/ns/books/published/StudentCSP/CSPRepeatImages/repeatimages.html

Zhao, J. (2022b, February 13). *How To Split, Merge & Blend Images Use Python Pillow*. https://www.dev2qa.com/how-to-split-merge-blend-images-use-python-pillow/

Rodgers, B. (2022) http://www.cs.kzoo.edu/cs107/schedule.html

Spraug, N. (2010). *COMP107 Support Code*.

http://www.cs.kzoo.edu/cs107/supportCode_jes4py.html