

#4-2: Libraries and Documentation

CS SCHOLARS – PROGRAMMING

Hw3 Takeaways

It's okay if you struggled with this one – the difficulty ramped up too quickly

- This is my fault; I'll take this into account when designing the curriculum for next year
- In the meantime, if you're having a lot of trouble, aim to complete just a subset of problems based on your interests:
 - #7 to practice nesting
 - #4 to practice top-down design
 - #8 to practice strings
 - #9 to practice lists
 - #10 to practice text-based user interaction
 - #6 to practice GUI-based user interaction

Let's also discuss when and how to use Google/StackOverflow when working on homework assignments

Learning Goals

Install **external modules** with the **pip** command

Read **documentation** to learn how to use a new module

Python Libraries

When you install Python on a machine, it comes with a large number of built-in libraries that provide functionality beyond the built-in syntax and functions. We've already introduced some: **math**, **random**, and **tkinter**.

However, there are many other libraries that have been built by developers outside of the core Python team, to add additional functionality to the language. These modules don't come as part of the Python language, but can be added in. We call these **external modules**.

Finding Useful Modules

One of the main strengths of Python as a language is that there are thousands of external modules available, which means that you can start many projects based on work others have done instead of starting from scratch.

You can find a list of popular modules here:
wiki.python.org/moin/UsefulModules

And a more complete list of pip-installable modules here: pypi.org

Install External Modules with `pip`

In order to use an external module, you must first **install** it on your machine. To install, you'll need to download the files from the internet to your computer, then integrate them with the main Python library so that the language knows where the module is located.

It is usually possible to install modules manually, but this process can be a major pain. Luckily, Python also gives us a streamlined approach for installing modules – the **`pip` module**! This feature can locate modules that are indexed in the Python Package Index (a list of commonly-used modules), download them, and attempt to install them.

Traditionally, programmers run `pip` from the **terminal**. This is a command interface that lets you make changes directly to your computer. You can access this in the application **Terminal** (Mac/Linux) or **PowerShell** (Windows).

Running pip

To run `pip` in the terminal, use this command:

```
pip install module-name
```

This will identify the module and start the download and installation process. It may run into a **dependency error** if the module needs a second module to already be installed – in general, installing that module and then running `pip` again will fix the problem.

Note: you will not be able to run `pip` on CMU cluster machines, as these have restricted permissions. You may need to log into your main account on personal machines to run it.

Thonny Has its Own Installer

If you're using Thonny, you can install modules without using pip!

Go to **Tools > Manage Packages**. Search for the module you want to install and click on it in the results. Then click 'Install' and Thonny will handle the pip installation for you.

If you're using repl.it, the website will install packages for you! Just type `import packageName` at the top of the file and click Run, and repl.it will install the named package (in most cases).

Activity: Install an Image Library

As an example, let's install a library that lets us manipulate images.

Pillow is a lightweight and easy-to-install module that lets you manipulate images beyond .gif files. It lets you modify images or use different types of images in Tkinter.

You do: Try installing it now!

Using an Installed Module

Once you've successfully installed a module, you should be able to put

```
import module-name
```

at the top of a Python file, and it will load the module the same way it would load a built-in library. For the Pillow library, the import name is slightly different from the install name:

```
import PIL
```

Note: this may fail if you have multiple versions of Python installed on your machine. Make sure to use the `pip` associated with the version of Python you're using in your editor. You can check your editor's version in Thonny with Help > About Thonny, then call `pip` using

```
pythonversion-number -m pip install module-name
```

Learning how a Module Works

Once a new module is installed, you're still left with one major question: how do you use it?

This varies by module, but the best answer is to **read the documentation**. Most external modules have official documentation or APIs that describe which functions exist and how to use the module.

It can also be helpful to search online for other projects that have used the same module, to find examples of how to set it up. Many people have written helpful tutorials online for this exact purpose.

Two standard resources for finding help are [StackOverflow](#), a site where people can ask questions about code and get answers from other developers, and [GitHub](#), a site where people post open-source projects for others to use and contribute to.

Reminder: always cite others' work!

You'll sometimes find a useful bit of code in a StackOverflow post or a GitHub project that you'll want to use in your own project.

Whenever you copy code from online, make sure to **cite it** the same way you would cite a paragraph of text in an essay. You can do this by putting a comment above the copied code that includes a link to the URL you got the code from.

This serves two purposes. First- it gives credit to the individual who originally wrote the code. Second- if you run into a problem with the code later on, you'll be able to look back to the original source to find a solution.

Note: policies around copying code change when you're working on a commercial product. Read the fine print if you're planning to sell your code!

Learning the Pillow Library

To learn how to use the Pillow library, let's start from the documentation!

<https://pillow.readthedocs.io/en/stable/index.html>

It often helps to start with a tutorial or 'start here' page. Pillow's docs have one:

<https://pillow.readthedocs.io/en/stable/handbook/tutorial.html>

Pillow Images

Pillow makes it very easy to open image files, using the `Image.open` function. You can even display those images with `image.show`, or save them with `image.save`.

```
from PIL import Image
img = Image.open("stella.jpg")
img.show()
img.save("new-stella.jpg")
```



Pillow Image Functions

You can provide Pillow images to the Tkinter `create_image` function, but you can also manipulate them directly!

Scrolling through the tutorial a bit shows that there's a method to crop an image. We can probably find a lot more too!

```
newImg = img.crop([200, 200, 3500,  
2500])  
newImg.save("new-stella.jpg")
```



Example: Create Mosaic

Let's say we want to create a mosaic of Stella pictures using Pillow. We can **alternate between coding and reading the documentation** to put together the code that we need.

First, we want to figure out how to change the image's size. Browsing the tutorial shows...

- We can get the height and width from `img.size`
- We can call `img.resize([newHeight, newWidth])` to create a new image of that size

Example: Combining Images

Next, we want to determine how to combine multiple images together.

More browsing shows that:

- We can create an empty image with `Image.new("RGBA", [width, height])`
- We can paste images into the empty image, along with coordinates, using `img.paste`

Now we can start creating something real!

Example: 3x3 Grid

We can combine regular Python code (in this case, loops) with the library functions to make a cool result!

```
from PIL import Image

img = Image.open("stella.jpg")
print(img.size) # initially 3024 x 4032
img = img.crop([24, 24, 3024, 3024]) # make it square
smallStella = img.resize([1000, 1000]) # make a small version

result = Image.new("RGB", [3000, 3000])
for row in range(3):
    for col in range(3):
        result.paste(smallStella, [col*1000, row*1000, (col+1)*1000, (row+1)*1000])
result.save("stella-grid.jpg")
```

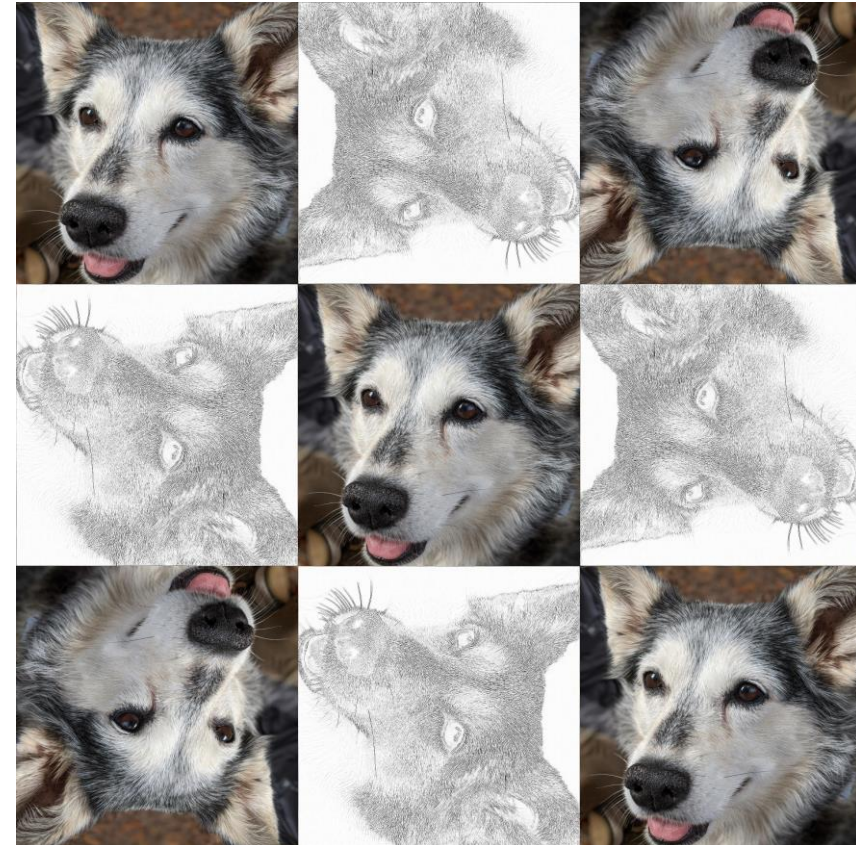


Example: Add Some Flair

We can keep exploring what's possible in the library to add even more customization!

- `img.rotate` lets us rotate the image by a given number of degrees
- `img.filter` lets us apply special filters, like contour or embossing

This is the real power of libraries – we can use these special features directly without having to implement them ourselves



Activity: Learn a New Library

You do: work with a small group of students to install and learn about a new library. Try to create a small example using the library to practice how it works!

After you've had some time to research and work, we'll go around the room and share what we've found.

Here are some recommended libraries to try out:

- NumPy/SciPy (math and science functions)
- Matplotlib (graphs and charts)
- scikit-learn (machine learning)
- BeautifulSoup (webscraping)
- Pydub (audio)
- Pygame (game development)