

Libraries and Documentation

15-110 – Friday 03/27

Announcements

- **Hw5 is due Monday**
 - If you haven't started yet, do so now!
- **Code reviews this weekend** – sign up now if you haven't already!
Sign-ups close **today 5pm**
 - Link: <https://www.cs.cmu.edu/~110/hw/hw4-code-review.html>
- Exam2 grades will be released Monday

Learning Goals

- Install **external libraries** and **import** them into files
- Learn how to use new libraries by using **documentation** and **tutorials**

Python Modules and Libraries

Recap: Python Modules

The Python programming language comes with a large set of built-in functions that cover a range of different purposes. However, it would take too long to load all these functions every time we want to run a program.

Python organizes its different functions into individual files called **modules**. When you run Python, it loads only a small set of functions from the built-in module. To use any other functions, you must **import** them.

Python Standard Library

We've already used a few core modules for homework assignments - mainly `math` and `tkinter`.

For a full list of modules in the Python Standard Library, look here:
<https://docs.python.org/3/library/index.html>

External Libraries

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 libraries don't come as part of the Python language, but can be added in. We call these **external libraries**.

Popular external libraries include matplotlib, scikit-learn, and pygame.



Finding Useful Libraries

One of the main strengths of Python as a language is that there are thousands of external libraries 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 libraries here:

wiki.python.org/moin/UsefulModules

And a more complete list of easily-installable libraries here: pypi.org

Install External Libraries with `pip`

In order to use an external library, 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 new library is located.

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

Running `pip`

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 **Command Prompt** (Windows). To run `pip` in the terminal, use the command:

```
pip install library-name
```

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

Thonny Has its Own Installer

If you're using Thonny, you can install libraries directly in the application.

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

You do: try installing the popular mathematics library **numpy**. Go to Tools > Manage Packages, search for numpy, and click **Install**.

Using an Installed Library

Once you've successfully installed a library, 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 regular module from the standard library. For example:

```
import numpy
```

Note: this may fail if you have multiple versions of Python installed on your machine and you install in the terminal. 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 library-name
```

Lots of Libraries

There are thousands of useful libraries out there that you can install and use!

As a starting place, check out the bonus slides on the course website for some popular libraries that CMU students tend to like to use.

Documentation and Tutorials

Learning how a Library Works

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

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

For example, numpy has a large set of tutorials, including beginner-friendly ones: <https://numpy.org/learn/>

Another Resources: Internet & AI Tutorials

Sometimes a useful library doesn't have particularly good tutorials, especially when you're still getting used to programming.

Try searching online to see if other websites have put together their own tutorials for the library. There's a large community of programmers online that help each other out – use that to your advantage.

If you can't find a tutorial, you can try using an AI tool to generate one for you.

Demo: finding a tutorial for `nltk`

Another Resource: Other Online Projects

Finally, it can be helpful to search online for other projects that have used the same library, to find examples of how they wrote their own code.

Two standard resources for finding coding information 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 piece of code online 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!

Sidebar: Advanced Python Syntax

We've taught you the core elements of programming that make up the majority of code you'll find in tutorials: data, functions, conditionals, loops, and data structures. However, there are many advanced aspects of Python that we haven't had time to cover.

If you find a bit of code that uses syntax you don't recognize, try searching it online or use an AI tool to learn what it's doing. Often you'll be able to connect it back to the core concepts you've already learned.

Example: Pillow Library

Learning the Pillow Library

Let's learn how to use documentation by learning a new library – **Pillow**. This library will let us edit images in interesting ways!

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

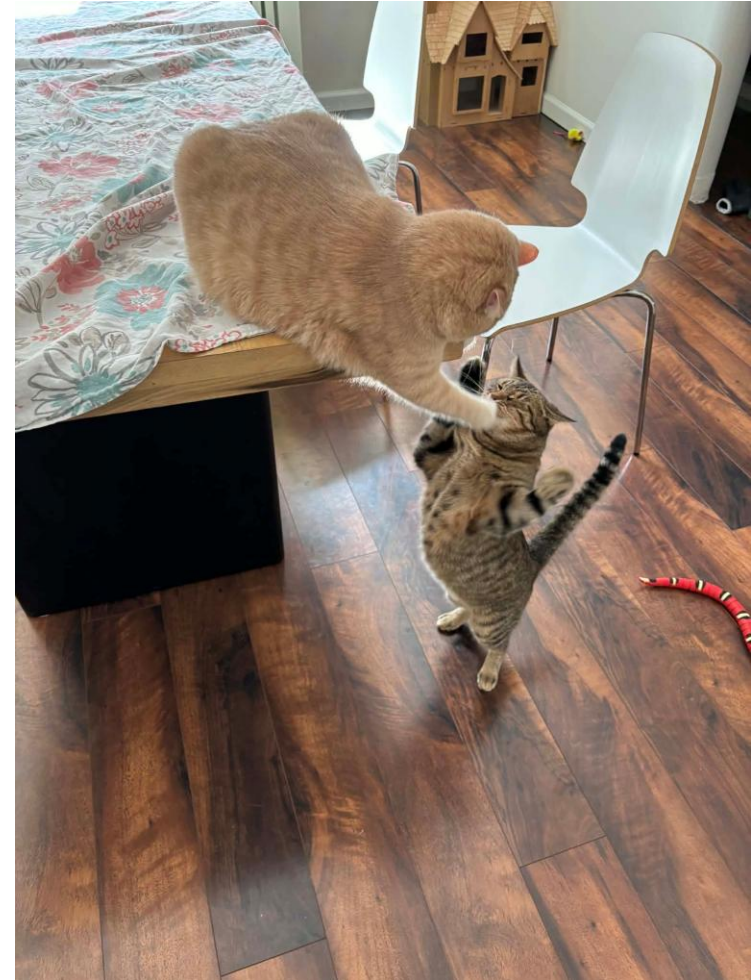
It often helps to start with a tutorial or 'start here' page. Pillow's docs have one:

<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`. All of this is explained directly in the tutorial!

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



Pillow Image Functions

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

Perhaps you want to crop an image? Try searching for the term 'crop' in the page and see what you can find!

```
newImg = img.crop([0, 0,  
                  1500, 1500])  
newImg.save("cat-square.jpeg")
```



Example: Create Mosaic

Let's say we want to create a mosaic of cat 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!

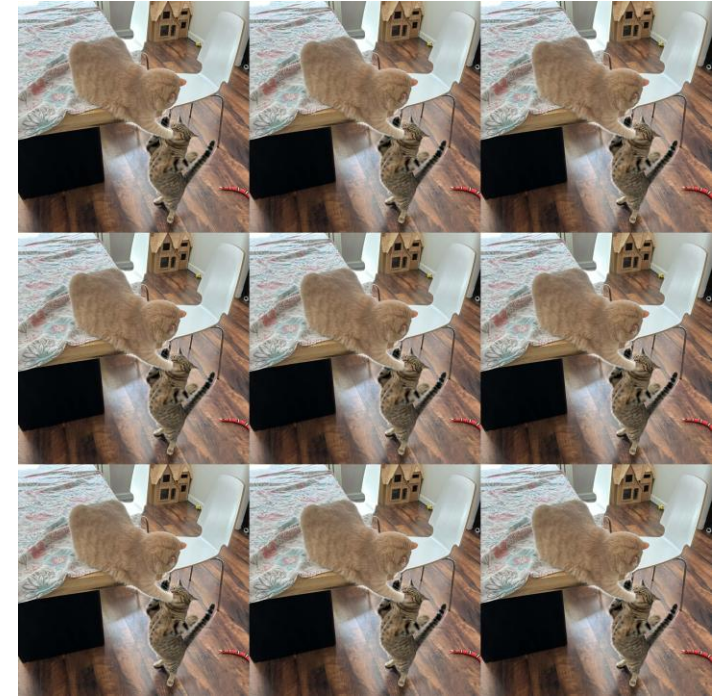
Example: 3x3 Grid

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("cats.jpeg")
print(img.size) # initially 1536 x 2048
img = img.crop([0, 0, 1500, 1500]) # make it square
small = img.resize([1000, 1000]) # make a smaller version

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



Consult Module References for Methods

We can find even more cool things that the library can do by checking the **module references**.

These are pages that describe in depth all the methods you can use when you import a module. For example, let's look at the `im.rotate` documentation in the `Image` module.

The page describes required and optional arguments, says what the function does, and gives an example of how to use the method!

```
Image.rotate(angle, resample=Resampling.NEAREST, expand=0, center=None, translate=None, fillcolor=None) \[source\]
```

Returns a rotated copy of this image. This method returns a copy of this image, rotated the given number of degrees counter clockwise around its centre.

PARAMETERS:

- **angle** – In degrees counter clockwise.
- **resample** – An optional resampling filter. This can be one of `Resampling.NEAREST` (use nearest neighbour), `Resampling.BILINEAR` (linear interpolation in a 2x2 environment), or `Resampling.BICUBIC` (cubic spline interpolation in a 4x4 environment). If omitted, or if the image has mode "1" or "P", it is set to `Resampling.NEAREST`. See [Filters](#).
- **expand** – Optional expansion flag. If true, expands the output image to make it large enough to hold the entire rotated image. If false or omitted, make the output image the same size as the input image. Note that the expand flag assumes rotation around the center and no translation.
- **center** – Optional center of rotation (a 2-tuple). Origin is the upper left corner. Default is the center of the image.
- **translate** – An optional post-rotate translation (a 2-tuple).
- **fillcolor** – An optional color for area outside the rotated image.

RETURNS:

An `Image` object.

This rotates the input image by `theta` degrees counter clockwise:

```
from PIL import Image

with Image.open("hopper.jpg") as im:

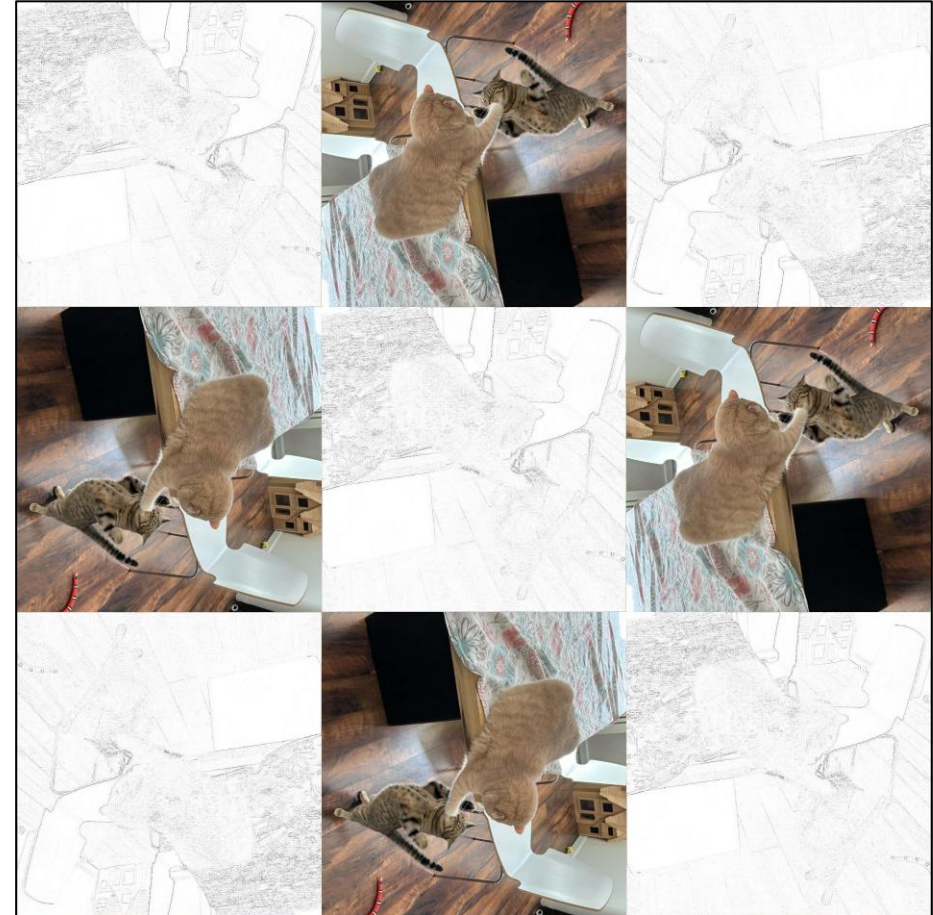
    # Rotate the image by 60 degrees counter clockwise
    theta = 60
    # Angle is in degrees counter clockwise
    im_rotated = im.rotate(angle=theta)
```

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



Learning Goals

- Install **external libraries** and **import** them into files
- Learn how to use new libraries by using **documentation** and **tutorials**