Built-In Libraries vs. External Libraries

As we covered in an earlier course, the Python Standard Library comes as part of the Python installation and includes modules for the most common tasks you can do with Python. But there's tons of other things you might want to do in your scripts, and not all of them are in the standard library. This is where external modules come into play. When developers write a Python module that they think others might find useful, they publish it in ***PyPI*** -- also known as the ***Python Package Index*** (https://pypi.org). We can browse this repository of Python modules to find the module we need. It includes thousands of projects, which are classified by different categories, like topic, development status, and intended audience.

In this module, we're going to be ***transforming*** and ***converting*** images. To do that, we'll be using a popular library for image manipulation: the ***Python Imaging Library (PIL)***. The original PIL library hasn't been updated since 2009 and does not support Python 3. Fortunately, there's a current *fork* of PIL called ***Pillow***, that properly supports Python 3 and is kept up-to-date. The Pillow library is packaged with the name **pillow**, but the module name in Python is still **PIL**.

If you try to import the PIL module on a computer that doesn't have pillow (or PIL) installed, you might get an error like this:

```
>>> import PIL
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
ModuleNotFoundError: No module named 'PIL'

```

Okay, looks like I don't have that module yet! As we covered in an earlier course, there are several ways to add external modules to your Python environment. PIL is a pretty common library, and on Linux it's usually available as a native package. For example:

```
user@ubuntu:~$ sudo apt install python3-pil
Reading package lists... Done
Building dependency tree
(...)
Unpacking python3-pil:amd64 (4.3.0-2) ...
Setting up python3-pil:amd64 (4.3.0-2) ...

```

For other environments, you should use Python's package installer, ***pip3***. Like this:

```
$ pip3 install pillow
Collecting pillow
  Downloading https://files.pythonhosted.org/packages/85/28/2c72ba965b52884a0bd71e4
     |████████████████████████████████| 3.9MB 1.7MB/s
Installing collected packages: pillow
Successfully installed pillow-6.2.1

```

Once we've done that, we can try to import the module again. And this time it should succeed with no errors:

```
>>> import PIL
```

That's better!

Now, how do you learn to use a library that you've never worked with before? It's time to get familiar with the library's ***Application Programming Interface (API)***!