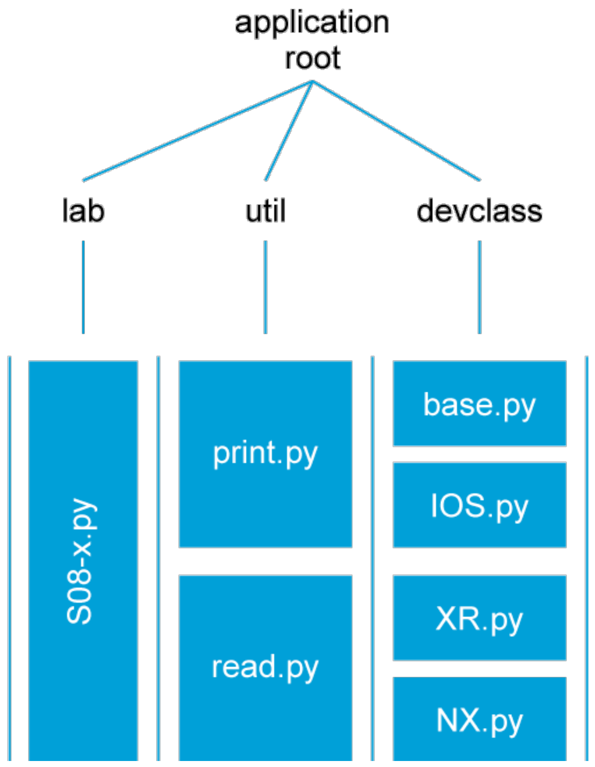
You will use packages when your project is large enough to warrant further organization of your application code. Packages are basically modules divided into separate folders or directories.

Note

Some operating systems refer their file system as containing 'folders', others use the term 'directories'. We will use 'directory' in this lesson.

- **Directories:** Packages use directory structure to organize code and modules
- Directories must have a file called `__init__.py` to identify as a package
- **Package Paths:** use PYTHONPATH environment variable and dots to specify path to modules



In the figure above, the application actually consists of Python code modules located in three different directories – 'lab', 'util', and 'devclass'. The 'lab' folder holds the main application code; the 'util' directory holds the utility modules for printing and reading; and the 'devclass' directory holds the modules for defining different device classes for use by the application.

The major points to know about using packages in Python:

- **Directories:** You use directories to organize your code modules.
- **__init__.py:** Every directory that is to become a Python package must have a file called __init__.py. This file can be completely empty, which is often the case. But it must exist.
- **Paths:** Python must know where to look on your system to find the modules you reference. If they are located in a Python package, then you must tell Python where to look. Your options are:
  - **Current directory:** If you are running your main application from a directory, and your Python package directories are located in that same directory, Python will be able to find them automatically.
  - **PYTHONPATH:** If your packages are located in another directory not directly in your application's runtime directory, you will need to set, and export, the PYTHONPATH environment variable. Python will use this variable to find your packages.