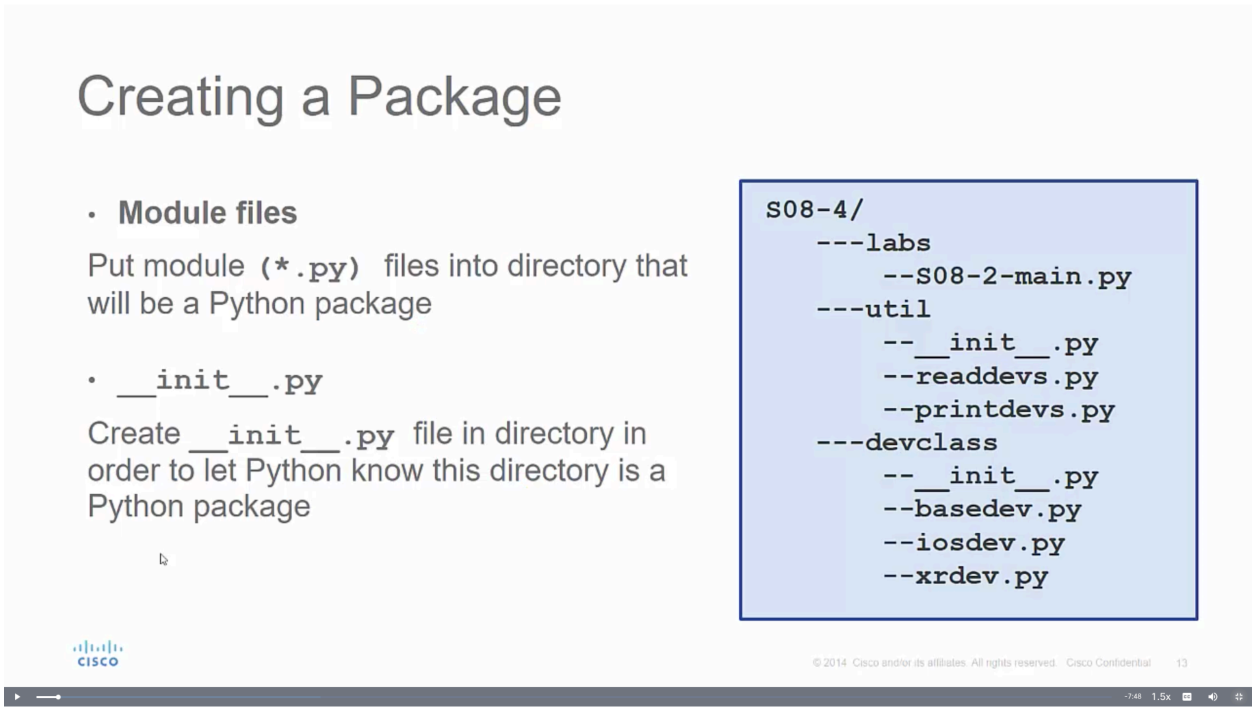


Creating a Package

- **Module files**

Put module `(*.py)` files into directory that will be a Python package

- `__init__.py`

Create `__init__.py` file in directory in order to let Python know this directory is a Python package

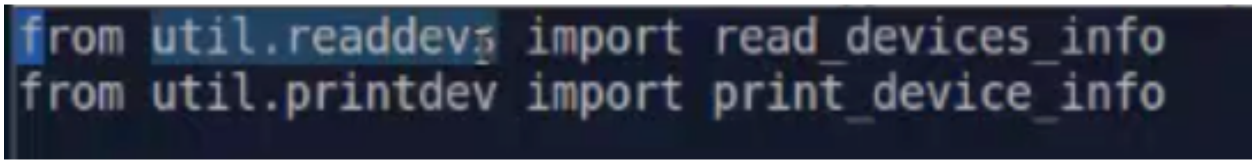
```
S08-4/
   ---labs
       --S08-2-main.py
   ---util
       --__init__.py
       --readdevs.py
       --printdevs.py
   ---devclass
       --__init__.py
       --basedev.py
       --iosdev.py
       --xrdev.py
```

© 2014 Cisco and/or its affiliates. All rights reserved. Cisco Confidential 13

“S08-2-main.py” is not considered a package and therefore does not require a __init.py__ file in the directory.

Directories with the __init.py__ file tell python to treat the directory like a package. The init.py file is a empty file.

When importing modules from another directory you may have to specify the path in dot notation -



```
from util.readdevs import read_devices_info
from util.printdev import print_device_info
```

Creating a package involves two items:

- **Modules:** You will place your module files into the appropriate directory for your package.
- **__init__.py:** You will create an __init__.py file in your directory for your package.

Once these steps are complete, you are ready to begin using your Python package.

Consider the example Python application and package directory structure below:

```
S08-3/
   main.py
   ---util
       --__init__.py
       --readdevs.py
       --printdevs.py
   ---devclass
       --__init__.py
       --basedev.py
       --iosdev.py
       --xrdev.py
```

In the example there are two directories:

- `util`: The directory that holds the Python package for the utility function modules. Note the presence of __init__.py.
- `devclass`: The directory that holds the Python package for the device class definitions. Note the presence of __init__.py.

For this example, the main application (main.py) is located in the root directory. This means that Python is able to look in the child directories 'util' and 'devclass' in order to satisfy its search for Python packages when main.py reference modules in those packages. The PYTHONPATH environment variable is not required.

Consider another example:

```
S08-3/
   ---labs
       --main.py
   ---util
       --__init__.py
       --readdevs.py
       --printdevs.py
   ---devclass
       --__init__.py
       --basedev.py
       --iosdev.py
       --xrdev.py
```

In the example, the main.py application has been moved into its own directory. There are now three directories:

- **labs:** Where the main application lives. Notice that there is no __init__.py file – it is not required. Only package folders require this file.
- **util:** The directory that holds the Python package for the utility function modules. Note the presence of __init__.py.
- **devclass:** The directory that holds the Python package for the device class definitions. Note the presence of __init__.py.

Because the application code was moved into a different directory, the package directories for util and devclass are no longer located in the directory of the main application. Therefore, the PYTHONPATH needs to be specified to let Python know where to look for the modules.

In order to successfully run the application in this environment, you would need to do the following:

```
.../S08-4/labs$ export PYTHONPATH=~/PRNE/labs/mod08/S08-4/
.../S08-4/labs$ python main.py
```