Building redistributables

- Objectives
  - Build redistributable libraries in Python
  - Compare modules and packages
  - Create packages and export multiple classes
  - Build packages for redistribution (possibly to PyPI)
  - Create stand-alone executables

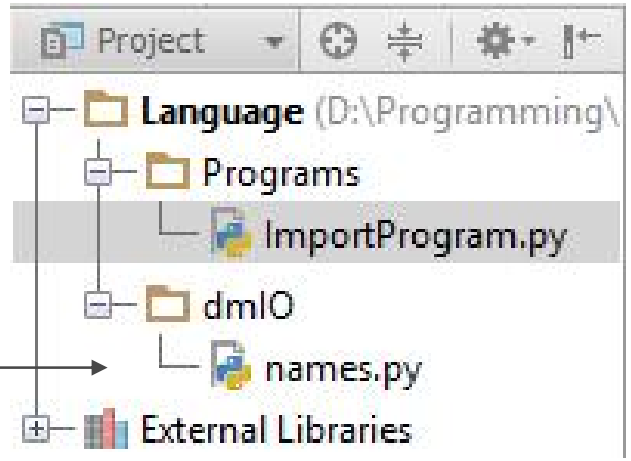
Why packaging

- Python has two main ways to reuse and share code
  - **module** – a single python file / script
  - **package** – a set of files meant to be used as a single library

Reminder: Consuming libraries [other scripts]

- Modules

Include relative paths for working folder



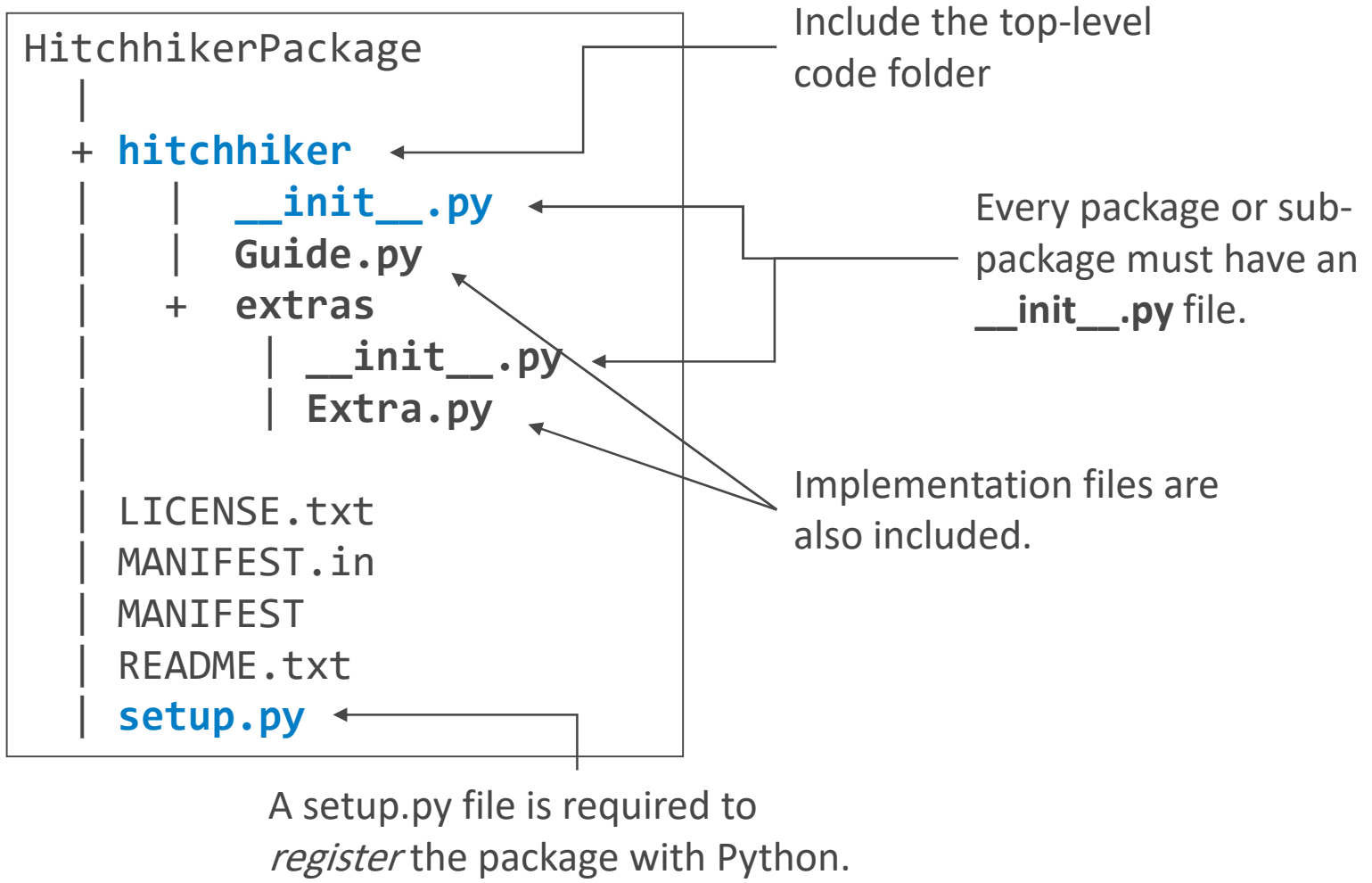

```
import dmIO.names

userName = dmIO.names.queryUserForName()
print("Nice to meet you " + userName)

# Sample execution:
# Please enter your name: Jeff
# Nice to meet you Jeff
```

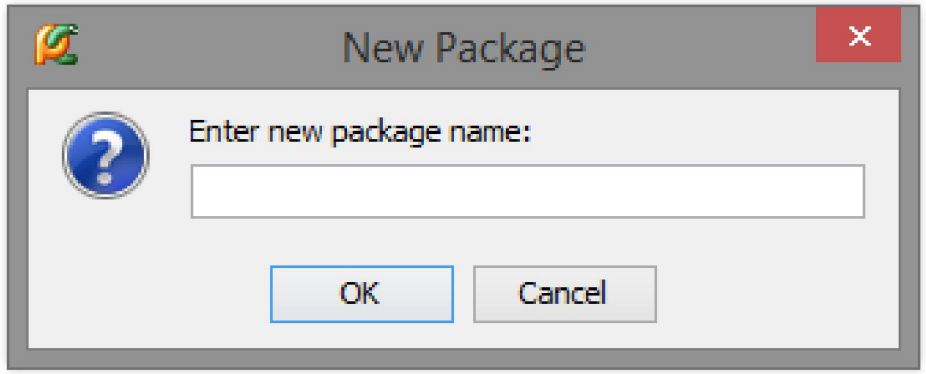
What is a package?

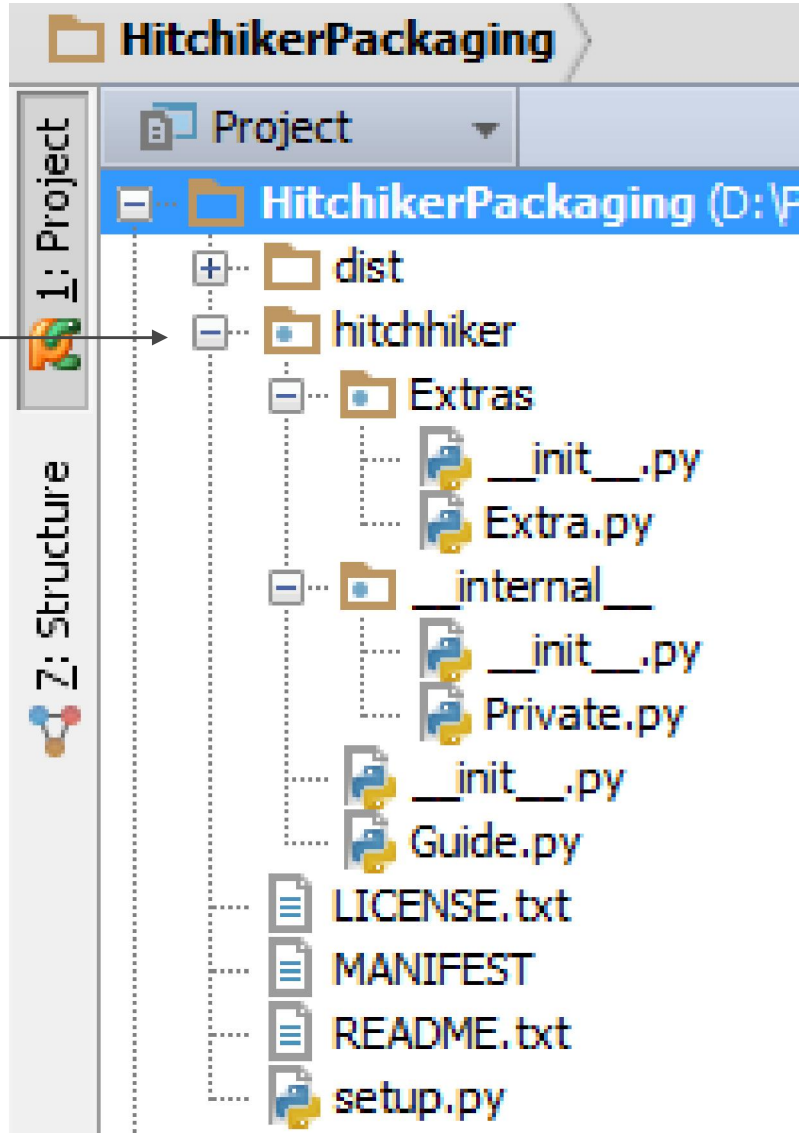
- A package is a directory with a convention-based layout



IDE support [PyCharm]

- PyCharm has direct support for packages
  - Packages have special icons





Package structure [setup.py]

- **setup.py** is the 'entry point' for **installing** your package
  - declares how to bundle package
  - declares name
  - declares requirements (e.g. dependencies)

```
# contents of setup.py
from setuptools import setup

setup(
    name="hitchhiker",
    version="1.8",
    packages=['hitchhiker', 'hitchhiker.extras'],
    license='Creative Commons Attribution',
    long_description=open('README.txt').read(),
    author='Michael Kennedy',
    author_email='michaelk@develop.com',
    url='http://blog.michaelckennedy.net'
)
```

Import and call **setup**()

name, **version**, and **packages** should be considered required.

There are more options (docs)

Building packages

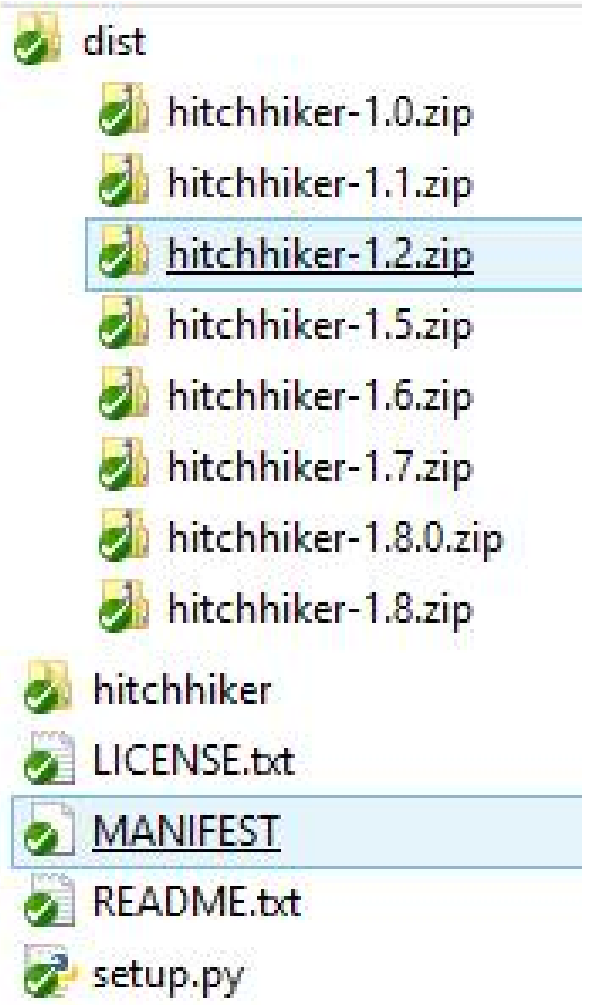
Build the package via **setup.py sdist**

```
prompt>python setup.py sdist
running sdist
running check
writing manifest file 'MANIFEST'
creating hitchhiker-1.9.1
creating hitchhiker-1.9.1\hitchhiker
creating hitchhiker-1.9.1\hitchhiker\__internal__
creating hitchhiker-1.9.1\hitchhiker\extras
making hard links in hitchhiker-1.9.1...
hard linking README.txt -> hitchhiker-1.9.1
hard linking setup.py -> hitchhiker-1.9.1
hard linking hitchhiker\Guide.py -> hitchhiker-1.9.1\hitchhiker
hard linking hitchhiker\__init__.py -> hitchhiker-1.9.1\hitchhiker
hard linking hitchhiker\__internal__\Private.py -> hitchhiker-1.9.1\hitchhiker\__internal__
hard linking hitchhiker\__internal__\__init__.py -> hitchhiker-1.9.1\hitchhiker\__internal__
...
creating 'dist\hitchhiker-1.9.1.zip' and adding 'hitchhiker-1.9.1' to it
adding 'hitchhiker-1.9.1\PKG-INFO'
adding 'hitchhiker-1.9.1\README.txt'
adding 'hitchhiker-1.9.1\setup.py'
adding 'hitchhiker-1.9.1\hitchhiker\Guide.py'
...
removing 'hitchhiker-1.9.1' (and everything under it)
```

Building packages

- Each build is compressed to a versioned zip file
- Added to a `dist` folder (created on first run)

Each version is kept and checked into source control



Installing packages

Install the package via **setup.py install**

```
prompt>python setup.py install

running install
running build
running build_py
running install_lib
copying build\lib\hitchhiker\extras\Extra.py ->
          C:\Python33\Lib\site-packages\hitchhiker\extras
...
byte-compiling C:\Python33\Lib\site-packages\hitchhiker\extras\Extra.py to
          Extra.cpython-33.pyc
...
running install_egg_info
Writing C:\Python33\Lib\site-packages\hitchhiker-1.9.0-py3.3.egg-info
```

Local / Private PyPI's

- On premise: pypiserver package

Packaging executables [cx_Freeze]

- Extension to **distutils**
- For applications not libraries
- Deployable files include
  - A full copy of Python
  - Your scripts
  - Dependent libraries
- Self contained
  - A clean, configured version of Python
  - And libraries
- More natural than a bunch of scripts (on Windows)

Summary

- Python supports redistributing code via modules or packages
- Packages are more nuanced and controlled than modules
- The __init__.py file controls which classes and modules are exported from packages
- Setup.py will build the redistributable packages