* Course Overview
* Prerequisites
  + Imported nested packages
  + Packages are modules
  + Locating modules for import
  + Modularization
    - A -> B -> C
  + Prerequisites
  + Main Block()
    - Differentiate Module execution from module import
  + \_\_<method-name>\_\_
    - Special methods
    - Dunder(double underscore) method name
* Introduction to Packages
  + Modules
    - Python’s basic tool for organizing code
    - Normally a single python source file
    - Load modules with import
    - Represented by module objects
  + Packages: Modules that contain other modules
  + urllib is a package
    - has a \_\_path\_\_ member
  + urllib.request is a module
    - doesn’t have a \_\_path\_\_ member
  + packages are generally directories
  + modules are generally files
* Locating Modules
  + Python looks on filesystem for corresponding python file
  + Python use sys.path
    - List of directories
    - Then searched in order in import
    - First match provides module
    - ImportError when there is no match
  + sys.path[0] is empty
    - when you start interpreter with no arguments
  + can add places for python to search
  + ex)
    - import sys
    - sys.path.append(‘not\_searched’)
  + PYTHONPATH
    - Environment variable
    - Lists of paths added to sys.path when python starts
    - Windows: set PYTHONPATH=path1;path2;path3
    - Linux/macOS: export PYTHONPATH=path1:path2:path3
* Summary
  + Importing nested packages
  + All modules in hierarchy are imported
    - Only the first name is bound
    - Use fully-qualified names for submodules
  + Package directory paths are stored in \_\_path\_\_
  + Sys.path contains module search
    - It is initialized from PYTHONPATH
* Creating Packages
  + create directory in sys.path
  + create a file called ‘\_\_init\_\_.py’
    - its what make the package a module
    - often empty
    - optional in python 3.3+
    - still required in earlier python versions
    - powerful initialization tool
    - explicit is better than implicit
  + module.\_\_file\_\_
    - returns path to \_\_init\_\_.py file
  + a package is a directory container \_\_init\_\_.py
* Creating a Subpackage
  + create a directory within a directory
  + the sub directory must also include \_\_init\_\_.py
  + import gzip to work with compressed file
  + gzip.open
    - decompresses during reading
  + python -m [full path module name] [sys.argv[1]] [sys.argv[2]...sys.argv[5]
    - -m: run module flag
  + ex)
    - python -m demo\_reader.compressed.bzipped test.bz2 data compressed with bz2
* Relative Imports
  + Absolute Imports
    - specify all ancestors modules of modules you want to import
  + ex)
    - import demo\_reader.compressed.bzipped
    - from demo\_reader.compressed import bzipped
  + Relative Import Syntax
    - use shortened paths to modules and packages
  + ex)
    - from ..module\_name import name
  + each dot before module\_name stands for an ancestor package of module doing the import
  + you can only use relative imports with ‘from module import name’ form of import
  + relative imports can only be used to import modules within the current top-level package
  + can reduce typing in deeply nested package structures
  + promote a certain form of modifiability
  + in general prefer absolute import
* Using \_\_all\_\_
  + module-level attribute
  + controls from module import \* behavior
  + if not specified, imports all public names
  + must be a list of string
    - each entry is a name to import
  + ex in \_\_init\_\_.py
    - from demo\_reader.compressed.bzipped import opener as bz2\_opener
    - from demo\_reader.compressed.gzipped import opener as gzip\_opener
    - \_\_all\_\_ = [‘bz2\_opener’, ‘gzip\_opener’]
  + while \_\_all\_\_ can be useful
    - we recommend avoiding import \* in general
  + packages are modules which can container other modules
  + directories containing \_\_init\_\_.py
  + technically optional
    - but its presence is an explicit signal to developers
    - executed at package import
  + packages can contain subpackages
  + \_\_all\_\_ controls import \* behavior
* Namespace Packages