Python Training.....

-- Jeetendra Bhattad

Modules

- Any python source file can be used as module.
- import <file-name> : to import python source as module
  - Creates a new namespace which serves as container for all the objects defined in that source file.
  - Executes code contained in the module within the newly created namespace.
  - Creates a name that refers to the module namespace. Usually name of namespace is same as module name.
- Multiple import : import <module-1>, <module-2> ...
- as : name used to refer a module can be changed using 'as'.
  - e.g import socket as sk (sk is local reference for the module which has renamed it, for others it remains the original one)

Modules

- Changing name of the imported module is very useful way to write extensible code.
  - e.g

```
if format == 'xml':
    import xml as reader
elif format == 'csv':
    import csv as reader
data = reader.<respective-operation>
```

- import statement can appear anywhere in program.
  - Code in each module is loaded and executed only once.
- sys.modules : shows the information about currently loaded modules. This dictionary maps module name to module objects and is used to determine whether import should load a fresh copy.
- from <module-name> import <symbol1, symbol2 ...> : used to load specific definitions. from statement & import are identical to each other.
- from <module-name> import * : loads all the definitions in a module except those that start with an underscore.
- __all__ : list of symbols that will be exported.
- importing definitions using from doesn't change their scoping rules. Refer e.g : module_packcages/ module_import_scope/foo.py & bar.py

Modules

- Module search path : interpreter searches the list of directories in sys.path
- .zip & .egg files path may be present in sys.path
- To add new location to search path add them to the sys.path
  - sys.path.append(“test_modules.zip”)
- .egg : packages created by the setuptools library.
  - .egg file is just a .zip file with additional metadata
- .py, .pyw, .pyc & .pyo files are only loaded from archive. Python doesn't create .pyc and .pyo files when .py files are loaded from archive. Thus it is important to make sure that these files are created in advance and placed in the archive in order to avoid poor performance when loading modules.
- python -O -m compileall <file-name>/<directory>

Packages

- Allow collection of modules grouped under a common Package Name.
- Resolves Namespace conflicts between module names used in different applications.
- It is defined by creating a directory with the same name as the package and creating the file __init__.py in that directory.
- Whenever any part of a package is first imported, the code in the file __init__.py is executed.
- __init__.py can be empty, usually contains code to perform package specific initialization.

distutils

- Creating distribution package
- Organize your work into a directory that contains README file, documentation file & organized code.
- Create setup.py in the working directory

```
from distutils.core import setup
setup(name = "basics", version="1.0", py_modules=['test'], packages=['history', 'operators', 'control_flow', 'data_types'])
```

py_modules : name of single python modules

packages : list of all package directories

scripts : list of script files (self executable)

```
python setup.py [sdist (source distribution)| bdist (binary distribution)]
cd dist; tar xvf basics-1.0.tar.gz; python setup.py install
```