Module

Python basic for everyone

28/06/2019

NORDIC
< CODER >

Modules

What are modules for?

Python modules are used to organize Python code.

Smaller Python scripts can have one module. But larger programs are split into several modules.

Modules are grouped together to form packages.

Modules

```
$ cat hello.py
def print_func( par ):
    print "Hello : ", par
    return
```

```
# Import module hello
import hello

# Now you can call defined function that module as follows
hello.print_func(“Earth")
```

```
Hello : Earth
```

```
>>> print __name__
__main__
>>> print hello.__name__
hello
>>>
```

Importing into the current namespace should be done with care due to name clashes

Modules

Frequently used modules

- **sys** Information about Python itself (path, etc.)
- **os** Operating system functions
- **os.path** Portable pathname tools
- **shutil** Utilities for copying files and directory trees
- **glob** Finds files matching wildcard pattern
- **re** Regular expression string matching
- **time** Time and date handling
- **datetime** Fast implementation of date and time handling
- **doctest, unittest** Modules that facilitate unit test

NORDIC
< CODER >

Introduction to language - modules

More frequently used modules

- **pdb** Debugger
- **pickle, cpickle, marshal, shelve** Used to save objects and code to files
- **getopt, optparse** Utilities to handle shell-level argument parsing
- **math, cmath** Math functions (real and complex) faster for scalars
- **random** Random generators (likewise)
- **gzip** read and write gzipped files
- **struct** Functions to pack and unpack binary data structures
- **StringIO, cStringIO** String-like objects that can be read and written as files (e.g., in-memory files)
- **types** Names for all the standard Python type

Modules

- Modules can contain any code
- Classes, functions, definitions, immediately executed code
- Can be imported in own namespace, or into the global namespace

```
>>> import math
>>> math.cos(math.pi)
-1.0
>>> math.cos(pi)
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
NameError: name 'pi' is not defined
>>> from math import cos, pi
>>> cos(pi)
-1.0
>>> from math import *
```

Introduction to language - modules

Module import

```
>>> from math import *
```

This construct will import all Python definitions into the namespace of another module. he use of this import construct may result in namespace pollution. We may have several objects of the same name and their definitions can be overridden.

```
#!/usr/bin/python
"""
names is a test module
"""

_version = 1.0

names = ["Paul", "Frank", "Jessica"]

def show_names():
   for i in names:
      print i

def _show_version():
   print _version
```

```
>>> from names import *
>>> print locals()
{'__builtins__': <module '__builtin__'
(built-in)>, '__file__': './private.py',
'show_names': <function show_names at
0xb7dd233c>,
'names': ['Paul', 'Frank', 'Jessica'],
 '__name__': '__main__', '__doc__':
None}
>>> show_names()
Paul
Frank
Jessica
```

Modules

- Inspecting module methods

```
>>> import numpy
>>> dir(numpy)
['ALLOW_THREADS', 'BUFSIZE', 'CLIP', 'ComplexWarning', 'DataSource', 'ERR_CALL',
'ERR_DEFAULT', 'ERR_DEFAULT2', 'ERR_IGNORE', 'ERR_LOG', 'ERR_PRINT', 'ERR_RAISE',
'ERR_WARN', 'FLOATING_POINT_SUPPORT', 'FPE_DIVIDEBYZERO', 'FPE_INVALID',
'FPE_OVERFLOW', 'FPE_UNDERFLOW', 'False_', 'Inf', 'Infinity', 'MAXDIMS', 'MachAr',
'NAN', 'NINF', 'NZERO', 'NaN', 'PINF', 'PZERO', 'PackageLoader', 'RAISE',
'RankWarning', 'SHIFT_DIVIDEBYZERO', 'SHIFT_INVALID', 'SHIFT_OVERFLOW',
'SHIFT_UNDERFLOW', 'ScalarType', 'Tester', 'True_', 'UFUNC_BUFSIZE_DEFAULT',
'UFUNC_PYVALS_NAME', 'WRAP', '__NUMPY_SETUP__', '__all__', '__builtins__',
'__config__', '__doc__', '__file__', '__git_revision__', '__name__', '__package__',
'__path__', '__version__', '_import_tools', '_mat', 'abs', 'absolute', 'add',
'add_docstring', 'add_newdoc', 'add_newdocs', 'alen', 'all', 'allclose', 'alltrue',
'alterdot', 'amax', 'amin', 'angle', 'any', 'append', 'apply_along_axis',
...
'typeNA', 'typecodes', 'typename', 'ubyte', 'ufunc', 'uint', 'uint0', 'uint16',
'uint32', 'uint64', 'uint8', 'uintc', 'uintp', 'ulonglong', 'unicode', 'unicode0',
'unicode_', 'union1d', 'unique', 'unpackbits', 'unravel_index', 'unsignedinteger',
'unwrap', 'ushort', 'vander', 'var', 'vdot', 'vectorize', 'version', 'void', 'void0',
'vsplit', 'vstack', 'where', 'who', 'zeros', 'zeros_like']
```

Modules

- Importing submodules

```
>>> import numpy
>>> numpy.random # Submodule
>>> numpy.random.randn() # Function in submodule
---------------------------------- (Restart Python)
>>> import numpy.random # Import submodule only
>>> numpy.random.randn()
---------------------------------- (Restart Python)
>>> from numpy import random # Alternative form
>>> random.randn()
---------------------------------- (Restart Python)
>>> from numpy.random import * # Previous warnings
>>> randn() # apply here as well!
```

Your own package

The main difference between a module and a package is that a package is a collection of modules AND it has an __init__.py file.

```
myMath/
    __init__.py
    adv/
        __init__.py
        sqrt.py
    add.py
    subtract.py
    multiply.py
    divide.py
```

```
# add.py

def add(x, y):
    """"""
    return x + y
```

```
# sqrt.py
import math

def squareroot(n):
    """"""
    return math.sqrt(n)
```

```
# outer __init__.py
from add import add
from divide import division
from multiply import multiply
from subtract import subtract
from adv.sqrt import squareroot
```

```
import mymath

print mymath.add(4,5)
print mymath.division(4, 2)
print mymath.multiply(10, 5)
print mymath.squareroot(48))
```

NORDIC
< CODER >