

# Modules and Packages

# Modules

- **Modules allow you to organize your code**
- **Place related functions in a file such as 'spam.py'**

```
# spam.py
```

```
a = 37
```

```
def foo():
```

```
    print 'you requested foo function' #statements
```

```
def bar():
```

```
    print 'you requested bar function' #statements
```

# Using Modules

Use `import` to load and execute

```
import spam
```

```
print spam.a  
spam.foo()
```

Comments:

- A module defines a **namespace**
- This namespace is the global namespace for everything in the module.
- Modules are **only loaded once** (even for repeated import statements).

# Modules

Large programs can be broken into modules

```
# numbers.py
```

```
def divide(a,b):
```

```
    q = a/b
```

```
    r = a - q*b
```

```
    return q,r
```

```
def gcd(x,y):
```

```
    g = y
```

```
    while x > 0:
```

```
        g = x
```

```
        x = y % x
```

```
        y = g
```

```
    return g
```

## The import statement

```
import numbers  
x,y = numbers.divide(42,5)  
n = numbers.gcd(7291823, 5683)
```

**import** creates a namespace and  
executes a file.

# Variations on Import

---

```
from module import name
```

This imports selective symbols from a module into the current namespace

```
from spam import foo, bar  
foo()      # No longer need module prefix
```

Or you can import all of the symbols into the current namespace

```
from spam import *  
foo()
```

Note: But this does not import symbols starting with an underscore (`_`)

```
from spam import fooooooooooooo as f  
f()
```

# No longer need module prefix

# Reloading Module

`reload module`

For example:

`reload spam`

- Useful for debugging, but somewhat problematic if you aren't careful.
- References to classes and other objects in the old module are not updated.
- Generally doesn't work with extension modules (written in C).

# Module Search Path

- **To look for modules on import...**

**The interpreter searches the directories in `sys.path`**

```
['',  
'/usr/local/lib/python1.5/',  
'/usr/local/lib/python1.5/plat-sunos5',  
'/usr/local/lib/python1.5/lib-tk',  
'/usr/local/lib/python1.5/lib-dynload',  
'/usr/local/lib/python1.5/site-packages',  
'/usr/local/lib/site-python/']
```

- **New directories can be added to the path as needed**

```
import sys  
sys.path.append('/home/myname/python')
```

# Module Loading

- On "import spam", the interpreter looks for the following files:

spam.so, spammodule.so, spammodule.sl, or

spammodule.dll (compiled extensions)

spam.pyo (optimized bytecode, only with -O option)

spam.pyc (bytecode)

spam.py (source)

- Creation of .pyc and .pyo files

Created automatically by the interpreter on module import

Contain compiled byte-code for the module.

.pyo files are optimized in the sense that they do not contain debugging information.

Regenerated if the modification date of the matching .py file is newer.

For distribution, only the .pyc or .pyo files are needed (the .py file can be omitted).

# Packages

- **Motivation**

Sometimes it is useful to break an application into submodules

**Example: A Graphics package**

**Graphics primitives**

**2d plotting**

**3d plotting**

**Image file formats**

- **Packages can be used to create a hierarchical module namespace**

**Graphics**

**Graphics.Primitives**

**Graphics.Primitives.lines**

**Graphics.Plot2D.xyplot**

**Graphics.Formats.png**

**Graphics.Formats.jpeg**

**...**

# Packages

## How to create a package?

Organize .py files in a subdirectory structure

Include special `__init__.py` files.

Graphics/

`__init__.py`

Primitives/

`__init__.py`

`lines.py`

`fill.py`

contains **floodfill**

...

Plot2D/

`__init__.py`

`xyplot.py`

`logplot.py`

...

Formats/

`__init__.py`

`png.py`

`jpeg.py`

...

## Packages

### Importing from a package

```
import Graphics.Primitives.fill
```

```
Graphics.Primitives.fill.floodfill(img,x,y,color)
```

```
from Graphics.Primitives import fill
```

```
fill.floodfill(img,x,y,color)
```

```
from Graphics.Primitives.fill import floodfill
```

```
floodfill(img,x,y,color)
```

### Note:

On import, the `__init__.py` files in each subdirectory are executed.

Can be used to perform initialization of a subcomponent.

**The End**