Organizing Larger Programs

Austin Bingham
@austin_bingham
austin@sixty-north.com

Presenter

Robert Smallshire
@robsmallshire
rob@sixty-north.com

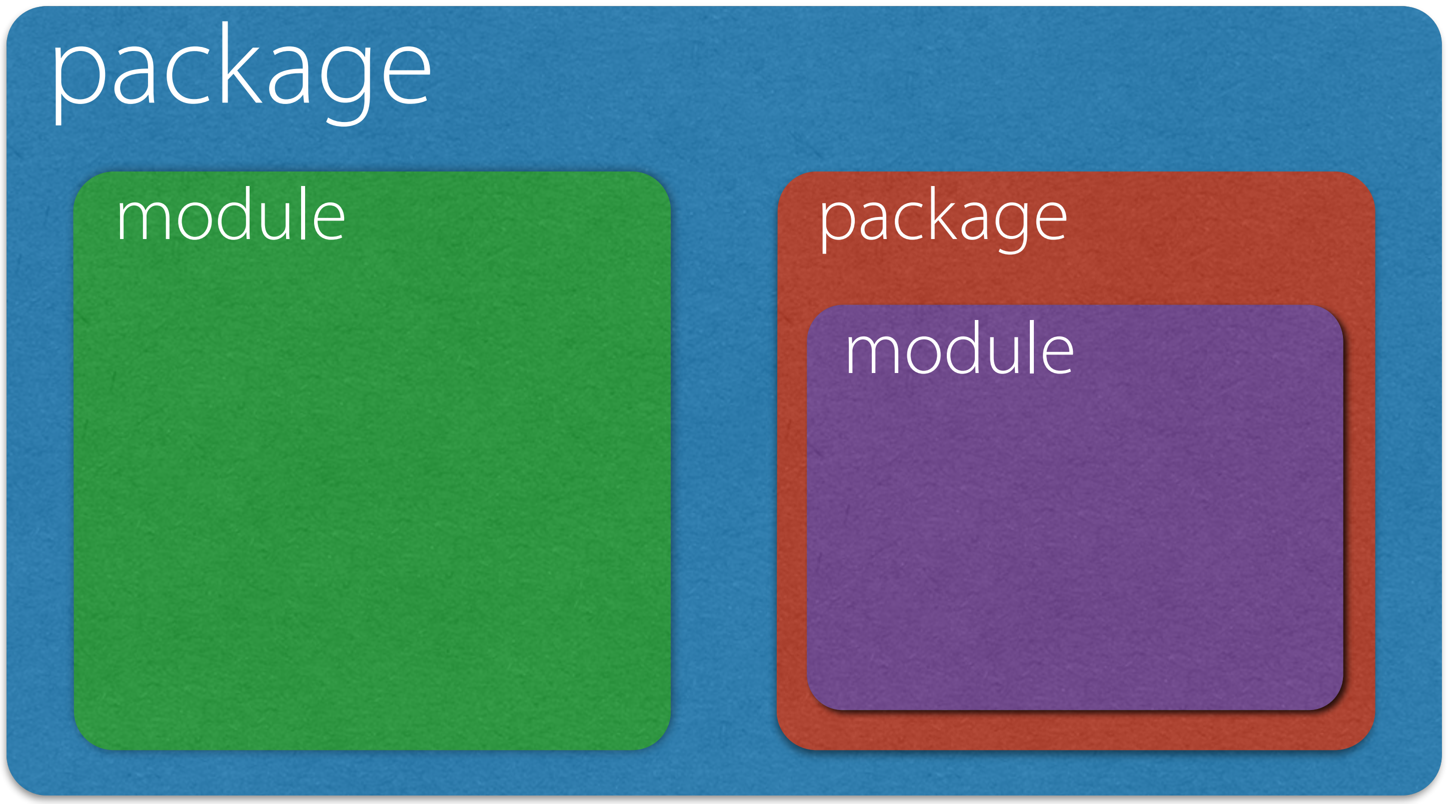
pluralsight
hardcore dev and IT training

```
>>> import my_module
>>> type(my_module)
<class 'module'>
```

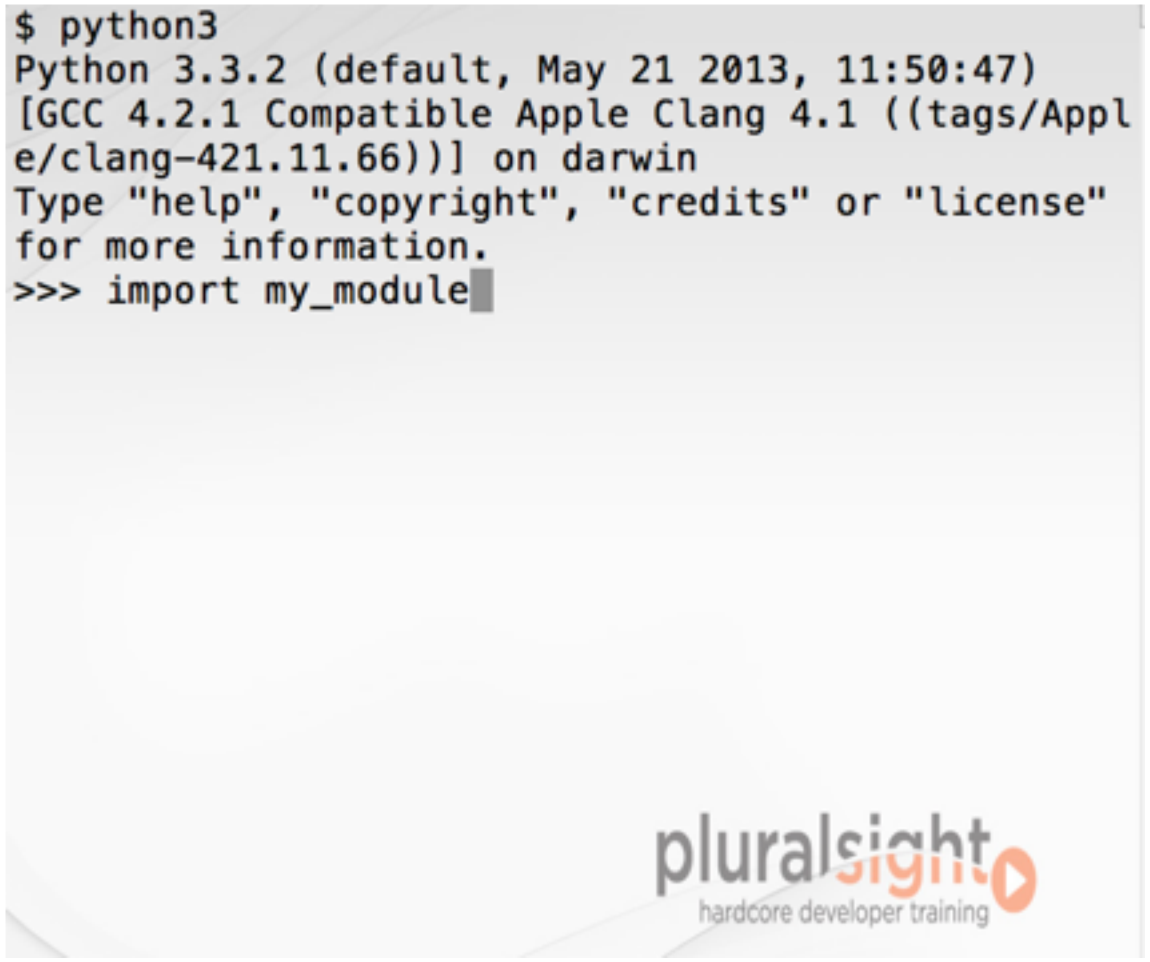
python™

package

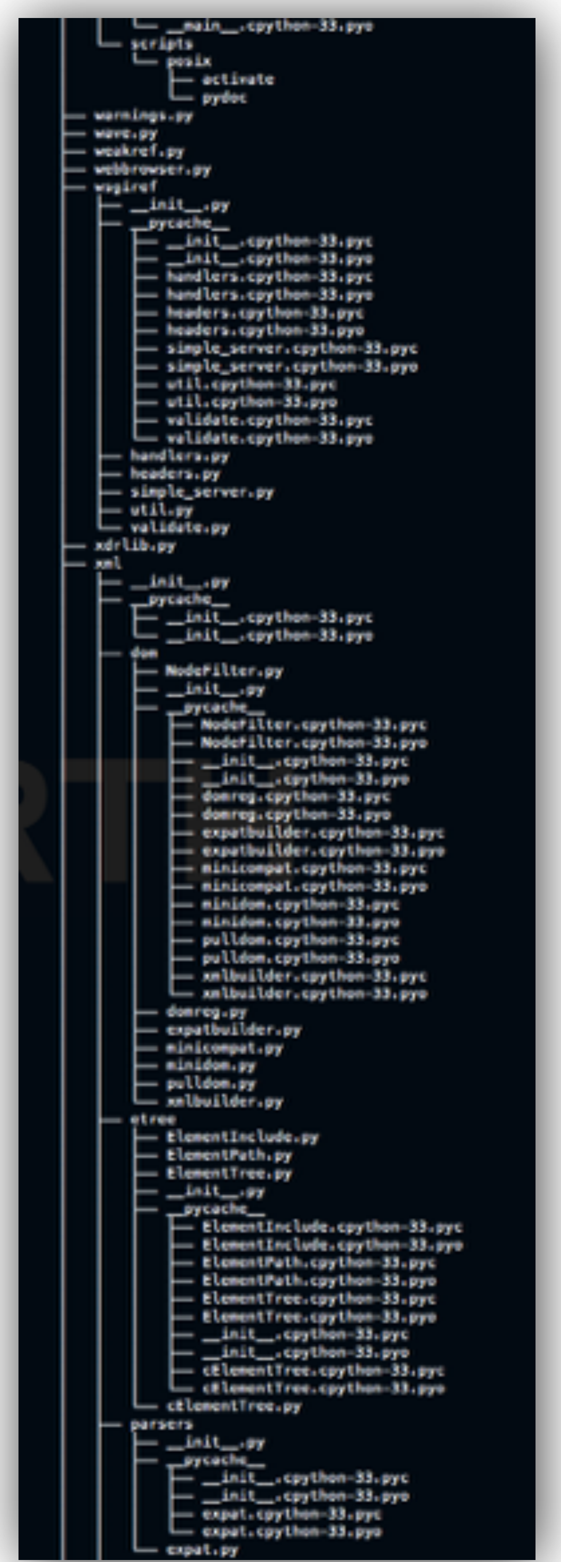
a module which can contain other modules



How does Python locate modules?



```
$ python3
Python 3.3.2 (default, May 21 2013, 11:50:47)
[GCC 4.2.1 Compatible Apple Clang 4.1 ((tags/Appl
e/clang-421.11.66))] on darwin
Type "help", "copyright", "credits" or "license"
for more information.
>>> import my_module
```



```
│   │   └── __main__.cpython-33.pyo
│   └── scripts
│       └── posix
│           ├── activate
│           └── pydoc
├── warnings.py
├── wave.py
├── weakref.py
├── webbrowser.py
├── wsgiref
│   ├── __init__.py
│   ├── __pycache__
│   │   ├── __init__.cpython-33.pyc
│   │   ├── __init__.cpython-33.pyo
│   │   ├── handlers.cpython-33.pyc
│   │   ├── handlers.cpython-33.pyo
│   │   ├── headers.cpython-33.pyc
│   │   ├── headers.cpython-33.pyo
│   │   ├── simple_server.cpython-33.pyc
│   │   ├── simple_server.cpython-33.pyo
│   │   ├── util.cpython-33.pyc
│   │   ├── util.cpython-33.pyo
│   │   ├── validate.cpython-33.pyc
│   │   └── validate.cpython-33.pyo
│   ├── handlers.py
│   ├── headers.py
│   ├── simple_server.py
│   ├── util.py
│   └── validate.py
├── xdrlib.py
├── xml
│   ├── __init__.py
│   ├── __pycache__
│   │   ├── __init__.cpython-33.pyc
│   │   └── __init__.cpython-33.pyo
│   ├── dom
│   │   ├── NodeFilter.py
│   │   ├── __init__.py
│   │   ├── __pycache__
│   │   │   ├── NodeFilter.cpython-33.pyc
│   │   │   ├── NodeFilter.cpython-33.pyo
│   │   │   ├── __init__.cpython-33.pyc
│   │   │   ├── __init__.cpython-33.pyo
│   │   │   ├── domreg.cpython-33.pyc
│   │   │   ├── domreg.cpython-33.pyo
│   │   │   ├── expatbuilder.cpython-33.pyc
│   │   │   ├── expatbuilder.cpython-33.pyo
│   │   │   ├── minicompat.cpython-33.pyc
│   │   │   ├── minicompat.cpython-33.pyo
│   │   │   ├── minidom.cpython-33.pyc
│   │   │   ├── minidom.cpython-33.pyo
│   │   │   ├── pulldom.cpython-33.pyc
│   │   │   ├── pulldom.cpython-33.pyo
│   │   │   ├── xmlbuilder.cpython-33.pyc
│   │   │   └── xmlbuilder.cpython-33.pyo
│   │   ├── domreg.py
│   │   ├── expatbuilder.py
│   │   ├── minicompat.py
│   │   ├── minidom.py
│   │   ├── pulldom.py
│   │   └── xmlbuilder.py
│   ├── etree
│   │   ├── ElementInclude.py
│   │   ├── ElementPath.py
│   │   ├── ElementTree.py
│   │   ├── __init__.py
│   │   ├── __pycache__
│   │   │   ├── ElementInclude.cpython-33.pyc
│   │   │   ├── ElementInclude.cpython-33.pyo
│   │   │   ├── ElementPath.cpython-33.pyc
│   │   │   ├── ElementPath.cpython-33.pyo
│   │   │   ├── ElementTree.cpython-33.pyc
│   │   │   ├── ElementTree.cpython-33.pyo
│   │   │   ├── __init__.cpython-33.pyc
│   │   │   ├── __init__.cpython-33.pyo
│   │   │   ├── cElementTree.cpython-33.pyc
│   │   │   └── cElementTree.cpython-33.pyo
│   │   └── cElementTree.py
│   ├── parsers
│   │   ├── __init__.py
│   │   ├── __pycache__
│   │   │   ├── __init__.cpython-33.pyc
│   │   │   ├── __init__.cpython-33.pyo
│   │   │   ├── expat.cpython-33.pyc
│   │   │   └── expat.cpython-33.pyo
│   │   └── expat.py
```

sys.path

list of directories Python searches for modules

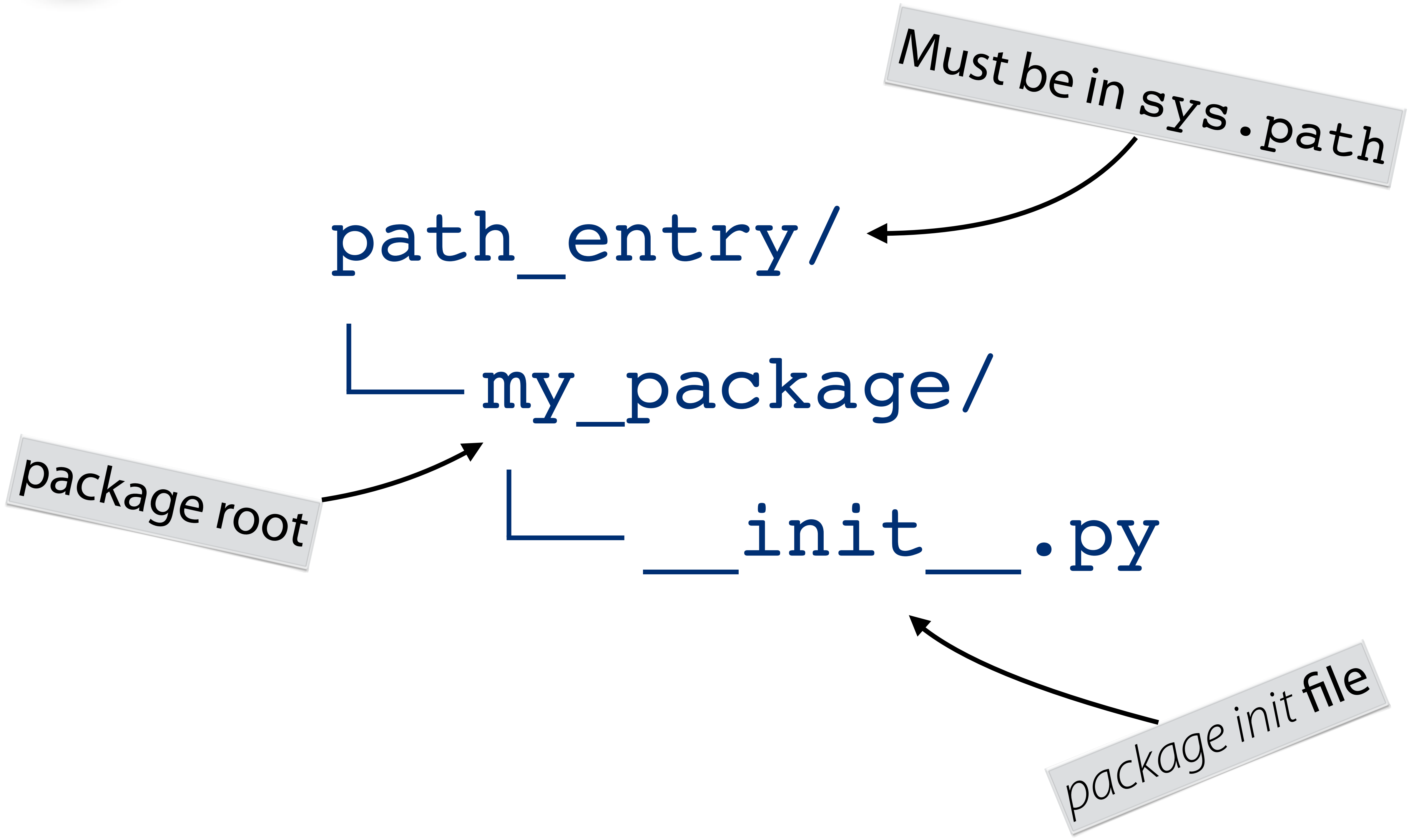
python™

PYTHONPATH

Environment variable listing paths added to sys.path

python™

Basic package structure



```
path_entry/
└── my_package/
    └── __init__.py
```

Package review

1. Packages are modules that contain other modules.
2. Packages are generally implemented as directories containing a special `__init__.py` file.
3. The `__init__.py` file is executed when the package is imported.
4. Packages can contain sub packages which themselves are implemented with `__init__.py` files in directories.
5. The `module` objects for packages have a `__path__` attribute.

absolute imports

imports which use a full path to the module

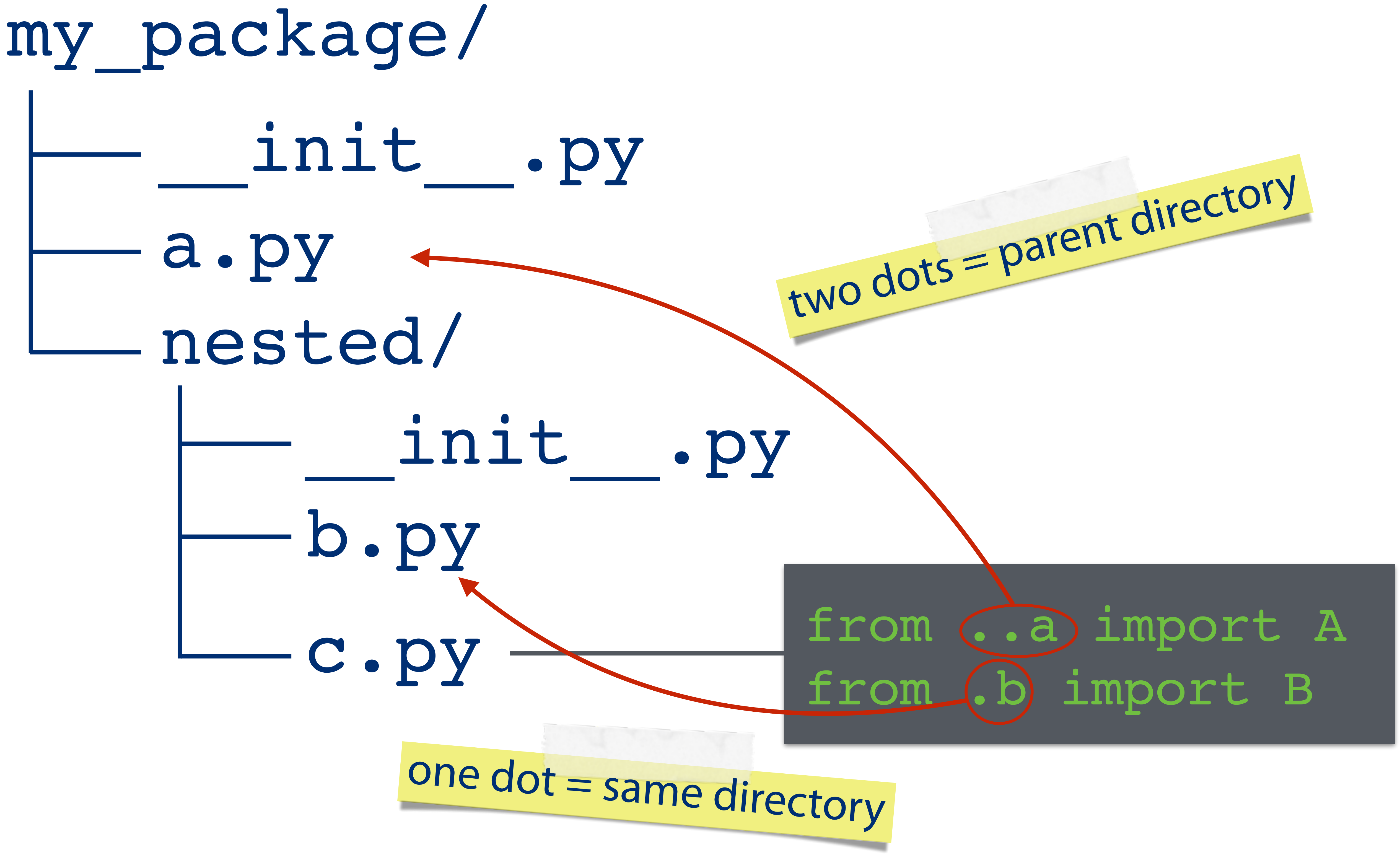
```
from reader.reader import Reader
```

relative imports

imports which use a relative path
to modules in the same package

```
from .reader import Reader
```

Relative imports



```
my_package/
├── __init__.py
├── a.py
└── nested/
    ├── __init__.py
    ├── b.py
    └── c.py
```

```python
from ..a import A
from .b import B
```

Relative imports

```
farm/
├── __init__.py
├── bird/
│   ├── __init__.py
│   ├── chicken.py
│   └── turkey.py
└── bovine/
    ├── __init__.py
    ├── cow.py
    ├── ox.py
    └── common.py
```

relative import, but requires use of `common.ruminate()`

```
from . import common
```

Relative imports

1. Can reduce typing in deeply nested package structures
2. Promote certain forms of modifiability
3. Can aid package renaming and refactoring
4. General advice is to avoid them in most cases

__all__

list of attribute names imported via `from module import *`

Local Variables

The `locals()` built-in function returns a dictionary mapping local variable **names** to their **values**.

```
from module import *
```

The `__all__` attribute should be a **list** of **strings** containing names available in the module.

namespace packages

packages split across several directories

Defined in PEP420

Namespace packages

Namespace packages have **no `__init__.py`**.

This avoids **complex initialization** ordering **problems**.

Importing namespace packages

1. Python scans all entries in `sys.path.`
2. If a matching directory with `__init__.py` is found, a normal package is loaded.
3. If `foo.py` is found, then it is loaded.
4. Otherwise, all matching directories in `sys.path` are considered part of the namespace package.

Namespace packages

```
path1
└── split_farm
    └── bovine
        ├── __init__.py
        ├── common.py
        ├── cow.py
        ├── ox.py
        └── string.py
path2
└── split_farm
    └── bird
        ├── __init__.py
        ├── chicken.py
        └── turkey.py
```

executable directories

directories containing an entry point for Python execution

Executable directories

```
reader
├── __main__.py
└── reader
    ├── __init__.py
    ├── compressed
    │   ├── __init__.py
    │   ├── bzipped.py
    │   └── gzipped.py
    └── reader.py
```

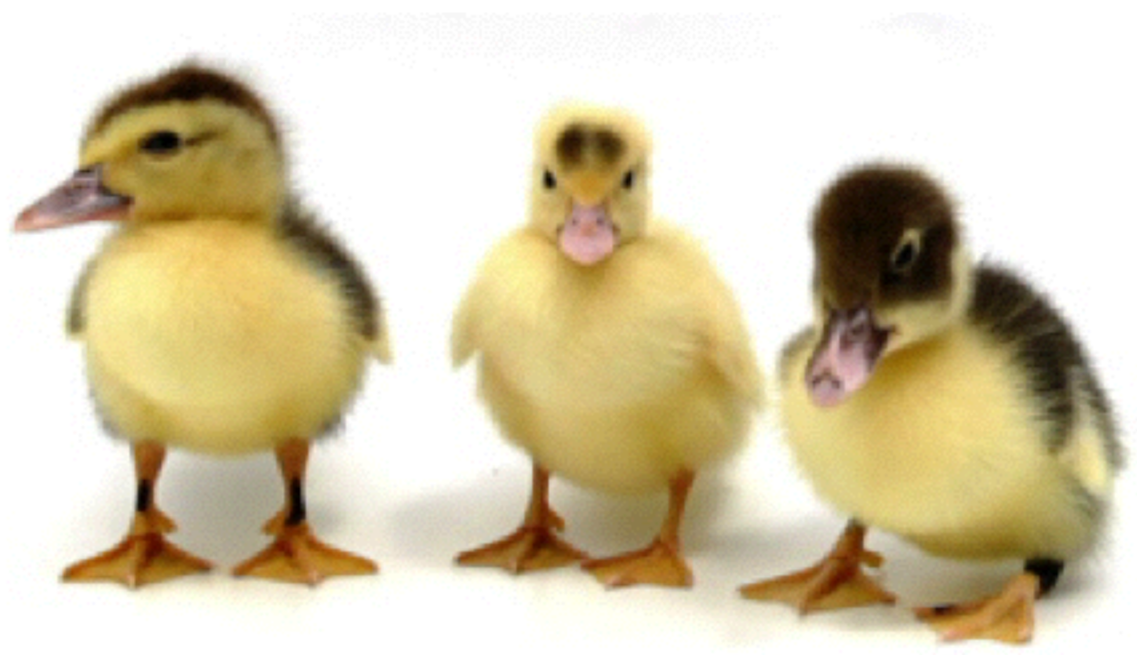
executable zip file

zip file containing an entry point for Python execution

Recommended project structure

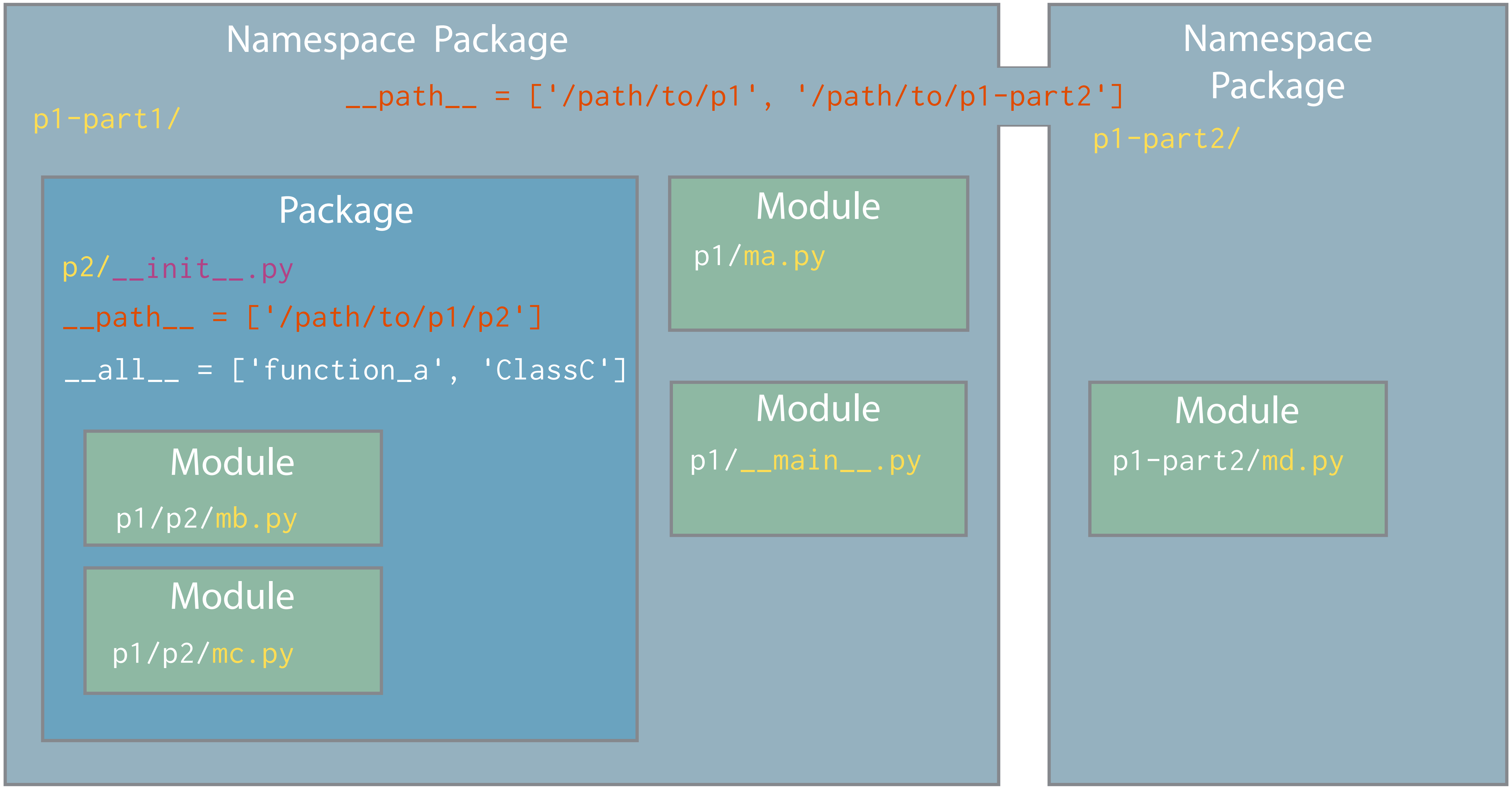
```
project_name
├── __main__.py
├── project_name
│   ├── __init__.py
│   ├── more_source.py
│   ├── subpackage1
│   │   └── __init__.py
│   └── test
│       ├── __init__.py
│       └── test_code.py
└── setup.py
```

Duck Tails



Organizing Larger Programs

```
python3 -m p1.mb
```



```
sys.path = ['', dir_1, dir_2, dir_n]
```

PYTHONPATH

```
from .mb import some_function
from ..ma import some_other_function
```