

Modules

Week 2, day 3

Modules

- Hides detail
- Only load what you need
- Library

```
import re

import sys, xml

from string import
    lowercase
```

Import what?

- Depends ...
- Looks for match on PYTHONPATH (sys.path)
- Runs that file, script, library

```
import casa

casa.py
casa.pyc
casa (folder)
casa.pyo
casa.so
...
```

**if ==
 ' ',**

- Distinguishes between main / primary script & those imported
- Imported modules usually don't "do work" but only define things ...

```
# string.py  
  
def find (s, sub):  
    [...]  
  
lowercase = "ab...yz"  
  
[...]  
  
if            == '          ':  
    # test module
```

Import if needed

- Only imports if not imported already (and up to date)
- `reload` forces re-import

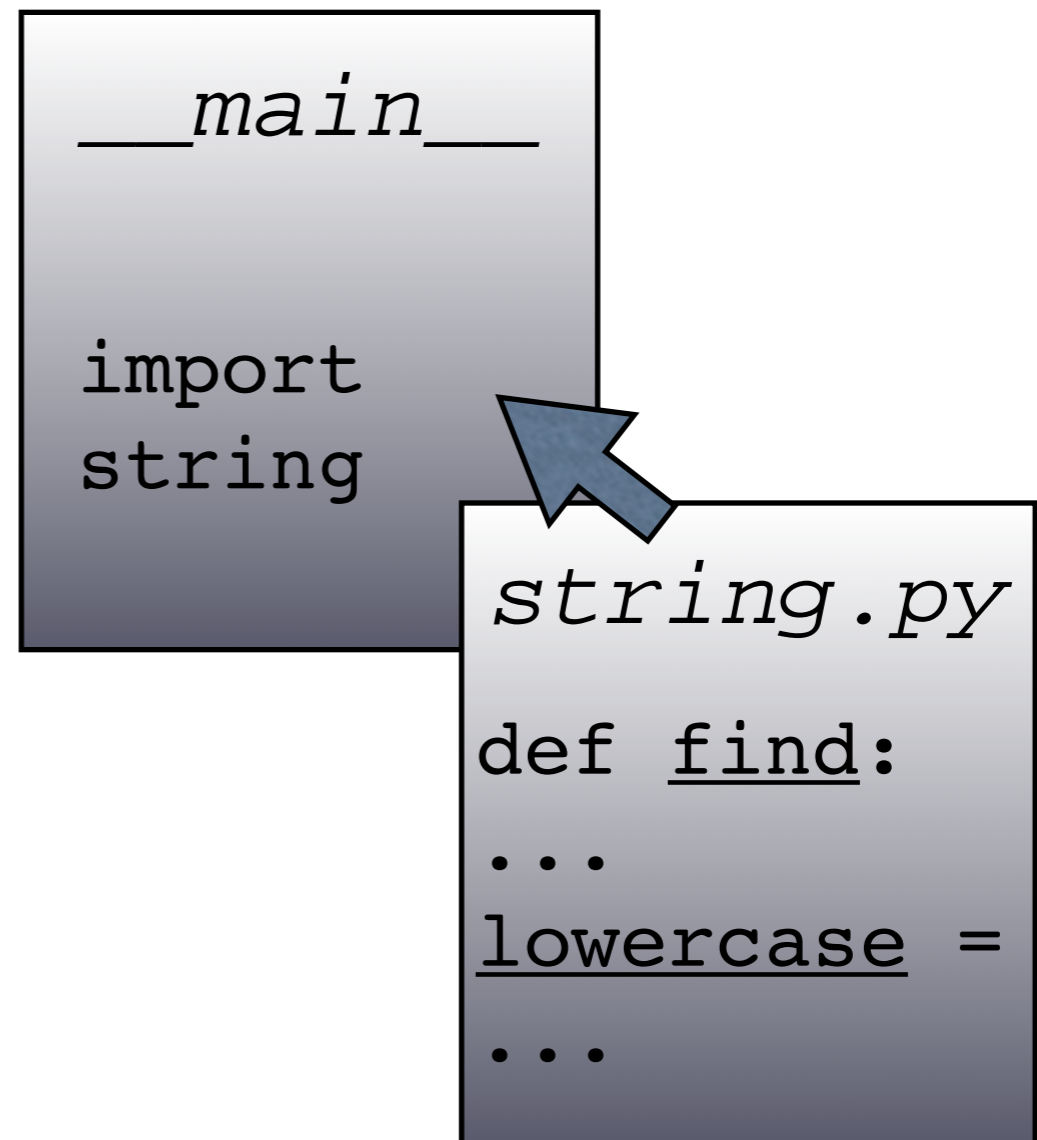
```
# hello.py  
print "Hello!"
```

```
>>> import hello  
"Hello!"  
>>> import hello
```

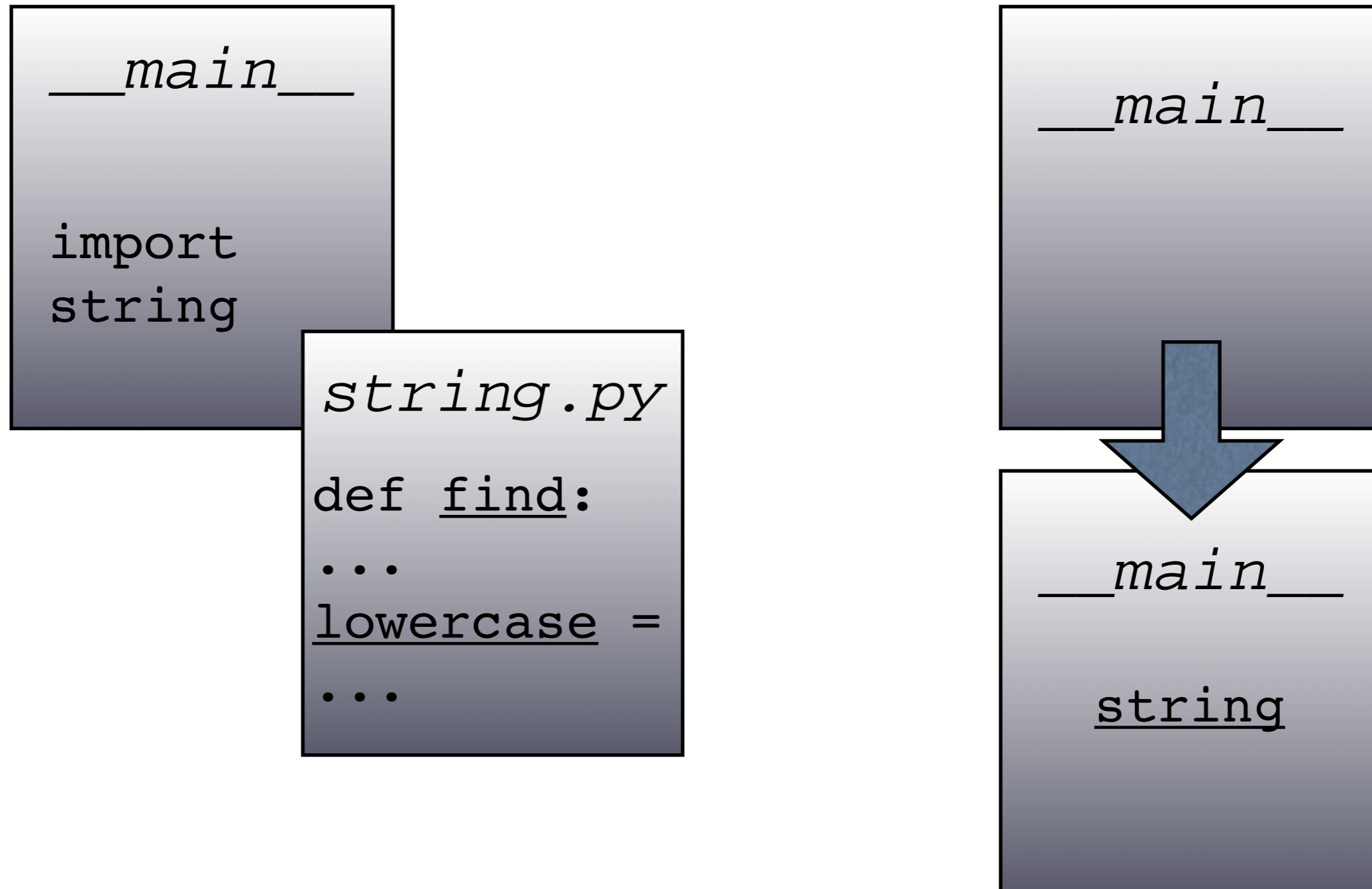
```
>>> reload (hello)  
"Hello!"
```

Importing defines and names things

- `import` runs the module ...
- Running the module defines names ...
- `import` makes the names available ...



Before & after import



Many ways of importing

```
import re
# "re" is now defined. Everything in re
# appears as "re.<name>", e.g. re.compile
```

```
from re import compile
# re.compile is now available as "compile".
# Nothing else in re or even re is visible.
```

```
from re import compile as mycomp
# re.compile is now available as "mycomp".
# Nothing else in re or even re is visible.
```

from *module* import *

- Imports everything that doesn't begin with underscore
- Can still import directly
- Single underscore “careful”, double “beware”

```
>>> import mod

>>> dir (mod)
['__builtins__',
 '__doc__', '__file__',
 '__name__', 'newfunc',
 '_xchar']

>>> from mod import *
>>> dir()
[... ,
 newfunc]
```

A folder can be a module

- Need to organise / group files & scripts
- Place in a folder
- An `__init__.py` file makes it a *package*

