MODULES

A QUICK RECAP

What we've seen so far…

Modules can be imported using

the `import` statement   `importlib.import_module`

When a module is imported:

system cache is checked first `sys.modules` → if in cache, just returns cached reference

otherwise:

module has to be located (found) somewhere finders e.g. `sys.meta_path`

module code has to be retrieved (loaded) loaders returned by finder → `ModuleSpec`

"empty" module typed object is created

a reference to the module is added to the system cache `sys.modules`

module is compiled

module is executed → sets up the module's namespace (`module.__dict__ is module.globals()`)

Module Finders

`sys.meta_path` →	`_frozen_importlib.BuiltinImporter`	finds built-ins, such as `math`
	`_frozen_importlib.FrozenImporter`	finds frozen modules
	`_frozen_importlib_external.PathFinder`	file-based modules

PathFinder

Finds file-based modules based on `sys.path` and package `__path__`

```
sys.path →    ['/home/fmb/my-app',
               '/usr/lib/python36.zip',
               '/usr/lib/python3.6',
               '/usr/lib/python3.6/lib-dynload',
               '/usr/local/lib/python3.6/dist-packages',
               '/usr/lib/python3/dist-packages']
```

```
collections.__path__ →    ['/usr/lib/python3.6/collections']
```

Module Properties

built-in `import math`

```
type(math)          → module

math.__spec__       →  ModuleSpec(name='math',
                                  loader=<class '_frozen_importlib.BuiltinImporter'>
                                  origin='built-in')

math.__name__       →  math

math.__package__    →  ''
```

`__file__` is not an attribute of `math`   (built-ins only)

Module Properties

standard library `import fractions`

```
type(fractions)           → module

fractions.__spec__        → ModuleSpec(name='fractions',
                                    loader=<_frozen_importlib_external.SourceFileLoader
                                           object at 0x7fa9bf7ff6d8>,
                                    origin='/usr/lib/python3.6/fractions.py')

fractions.__name__        → fractions

fractions.__package__     → ''

fractions.__file__        → /usr/lib/python3.6/fractions.py
```

Note that `fractions.__file__` was found by `PathFinder` in one of the paths listed in `sys.path`

Module Properties

custom module `import module1`

```
type(module1)          → module

module1.__spec__       → ModuleSpec(name='module1',
                                   loader=<_frozen_importlib_external.SourceFileLoader
                                          object at 0x7fd9f4c4ae48>,
                                   origin='/home/fmb/my-app/module1.py')

module1.__name__       → module1

module1.__package__    → ''

module1.__file__       → /home/fmb/my-app/module1.py
```

Note that `module1.__file__` was found by `PathFinder` in one of the paths listed in `sys.path`

Some Notes

Python docs:

https://docs.python.org/3/tutorial/modules.html
https://docs.python.org/3/reference/import.html

PEP 302

Python modules may reside

in the built-ins

in files on disk

they can even be pre-compiled, frozen, or even inside zip archives

anywhere else that can be accessed by a finder and a loader

custom finders/loaders → database, http, etc

For file based modules (`PathFinder`):

They must exist in a path specified in `sys.path`

or in a path specified by `<package>.__path__`