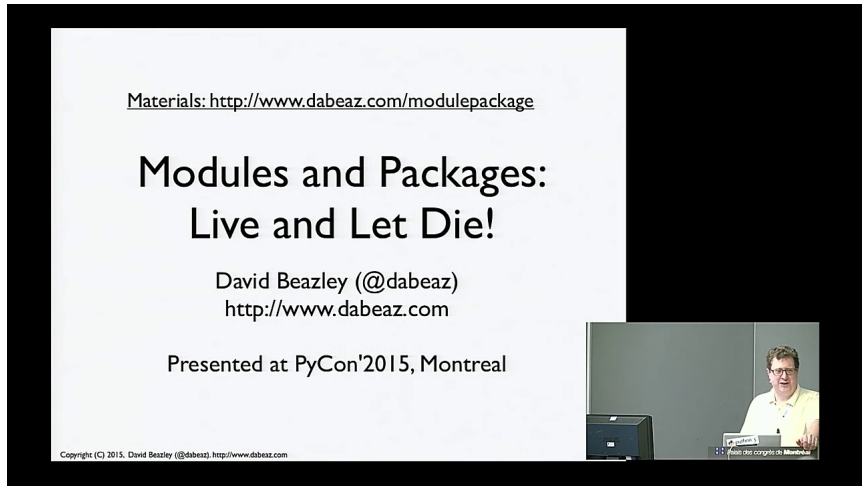
Import Deep Dive

Petr Viktorin
@encukou
encukou@gmail.com
encukou.cz

EuroPython 2015

```
import random
```

Prior art



David Beazley — Modules and Packages: Live and Let Die!

`http://pyvideo.org/video/3387`

```
import random
```

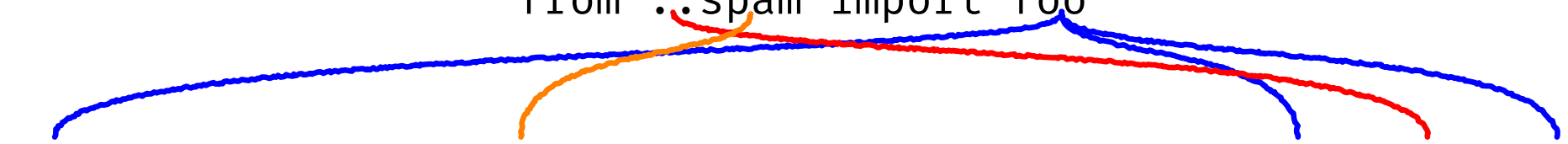
```
import random

random = __import__('random')
```

```
import random

random = __import__('random')


from ..spam import foo
```

```
import random
```

```
random = __import__('random')
```

```
from ..spam import foo
```



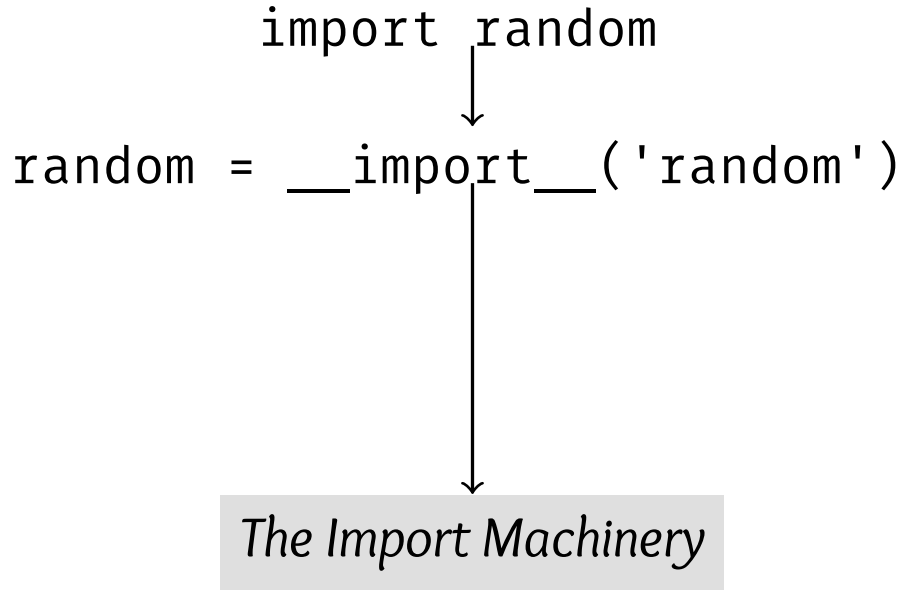
```
foo = __import__('spam', globals(), locals(), ['foo'], 2).foo
```

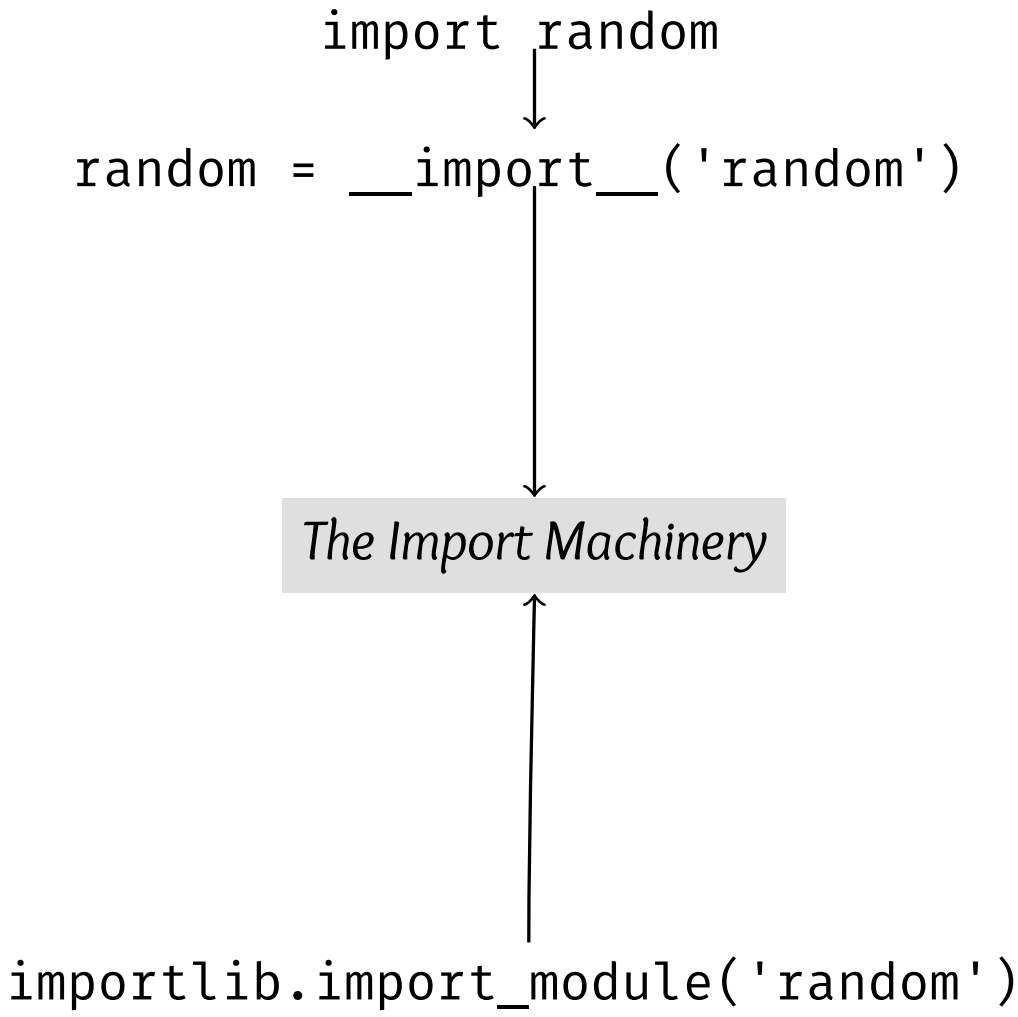
https://docs.python.org/3/library/functions.html#__import__

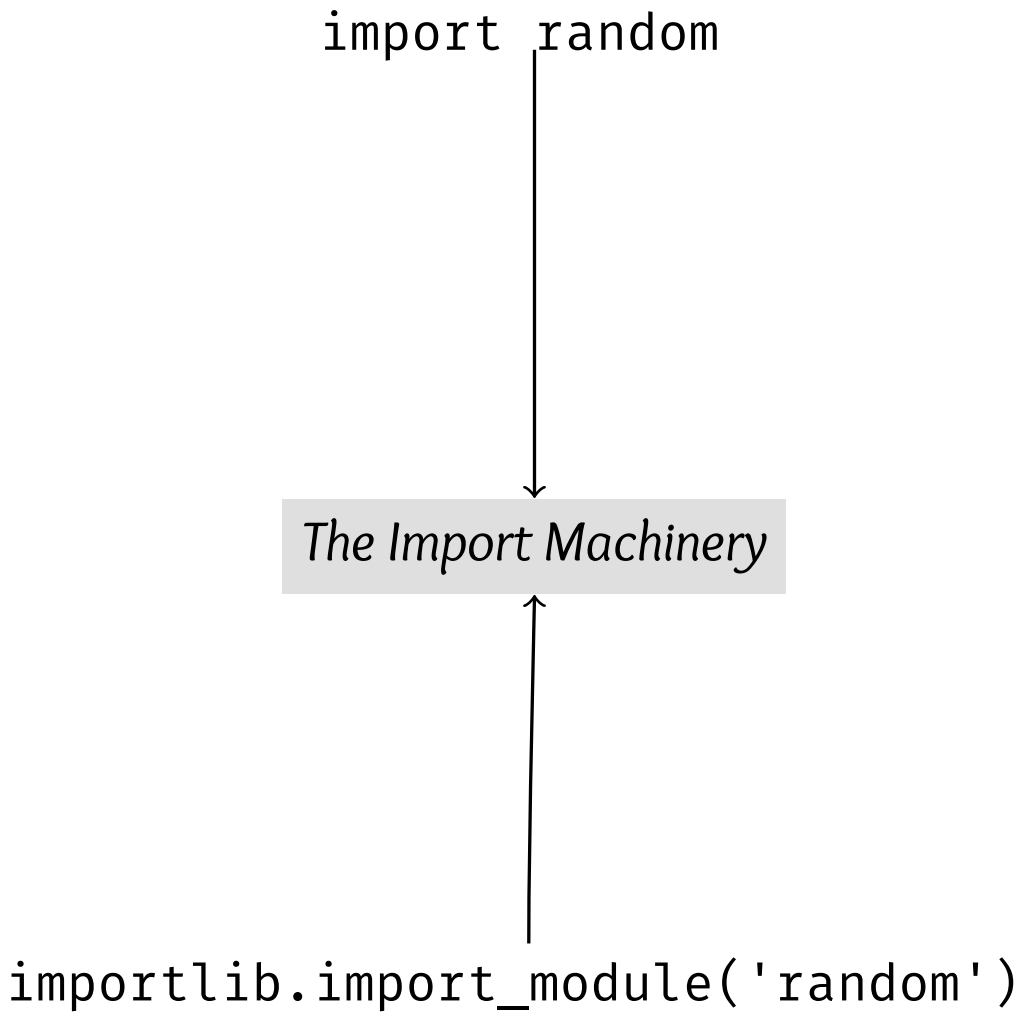
```
import random
```

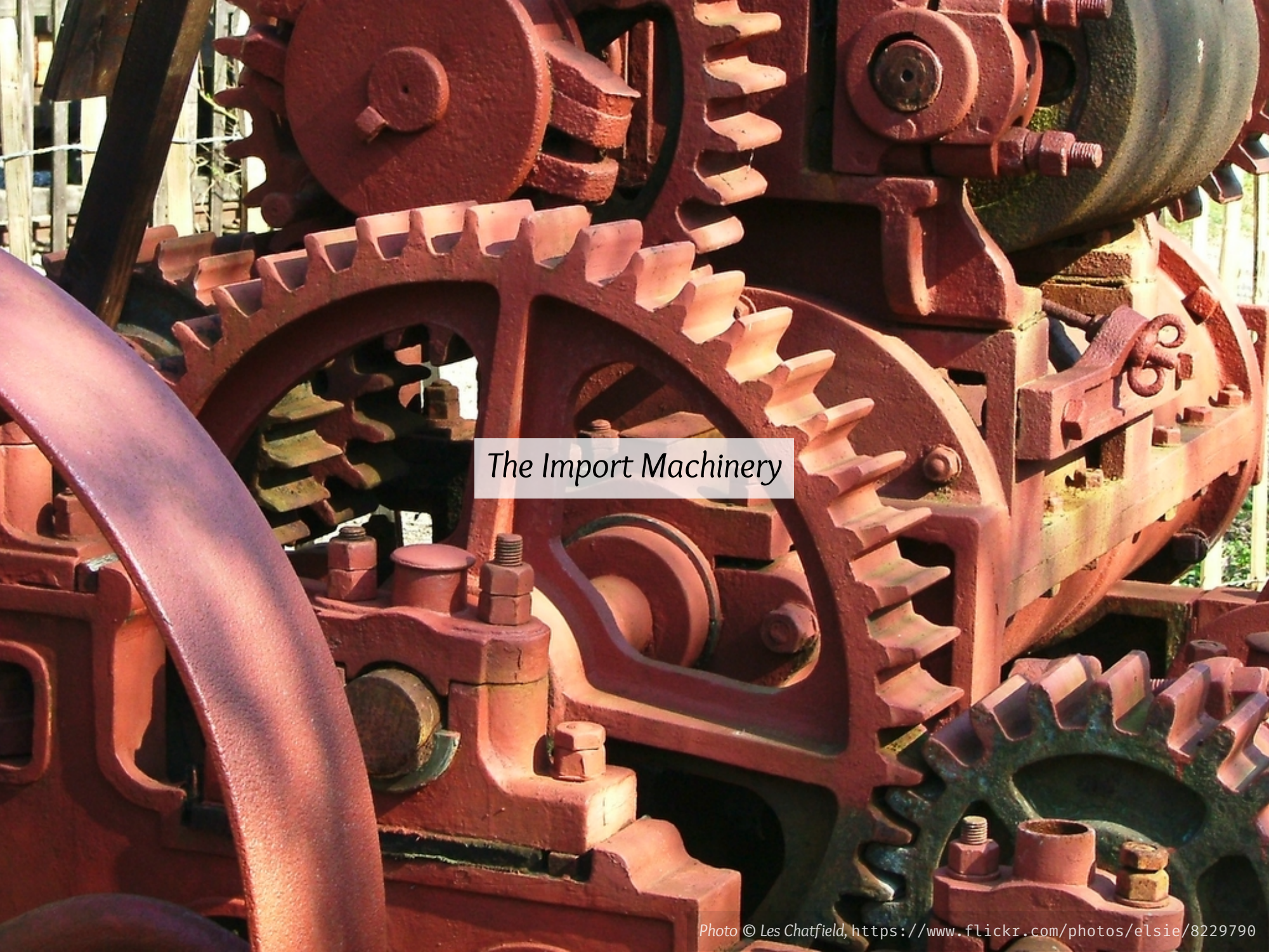


```
random = __import__('random')
```


import random
random = __import__('random')
The Import Machinery


import random
random = __import__('random')
The Import Machinery
importlib.import_module('random')


import random
The Import Machinery
importlib.import_module('random')



The Import Machinery

sans:

- Locking
- Caching
- Error handling
- Pythons older than 3.4
- Backwards compatibility shims

```
importlib.import_module('random')
```

import_module(name):
    `sys.modules[name]`*? return it*

```
    spec = find_spec(name, path)
    load(spec)
    module = sys.modules[name]
```

    return `module`

import_module(name):

`sys.modules[name]`*? return it*

```
spec = find_spec(name, path)
load(spec)
module = sys.modules[name]
```

return `module`

import_module(name):
  `sys.modules[name]`*? return it*

  `spec = find_spec(name, path)`
  `load(spec)`
  `module = sys.modules[name]`

  return `module`

```
>>> import random
>>> old = random

>>> import random
>>> random is old
True
```

import_module(name):
  `sys.modules[name]`*? return it*

  `spec = find_spec(name, path)`
  `load(spec)`
  `module = sys.modules[name]`

  return `module`

```
>>> import random
>>> old = random
>>> del sys.modules['random']
>>> import random
>>> random is old
False
>>> (random.sample is
...    old.sample)
False
```

import_module(name):

```
    sys.modules[name]? return it

    spec = find_spec(name, path)
    load(spec)
    module = sys.modules[name]

    return module
```

```
>>> import random
>>> old = random
>>> del sys.modules['random']
>>> import random
>>> random is old
False
>>> (random.sample is
...    old.sample)
False
```

```
>>> sys.modules['impostor'] = 'Not a module, is it?'
>>> import impostor
>>> impostor[:12]
'Not a module'
```

import_module(name):
    `sys.modules[name]`*? return it*

    `spec = find_spec(name, path)`
    `load(spec)`
    `module = sys.modules[name]`

    return `module`

import_module(name):
    `sys.modules[name]`*? return it*

    `spec = find_spec(name, path)`
    `load(spec)`
    `module = sys.modules[name]`

    return `module`

```
>>> sys.path
['',
 '/usr/lib64/python34.zip',
 '/usr/lib64/python3.4',
 ...]
```

import_module(name):
    sys.modules[name]? *return it*

    spec = find_spec(name, path)
    load(spec)
    module = sys.modules[name]

    return module

```
>>> sys.path
['',
 '/usr/lib64/python34.zip',
 '/usr/lib64/python3.4',
 ...]
```

```
>>> importlib.util.find_spec('antigravity')
ModuleSpec(name='antigravity',
   loader=<_frozen_importlib.SourceFileLoader ...>,
   origin='/usr/lib64/python3.4/antigravity.py')
```

import_module(name):
  `sys.modules[name]`*? return it*

  `spec = find_spec(name, path)`
  `load(spec)`
  `module = sys.modules[name]`

  return `module`

```
>>> sys.path
['',
 '/usr/lib64/python34.zip',
 '/usr/lib64/python3.4',
 ...]
```

```
>>> importlib.util.find_spec('antigravity')
ModuleSpec(name='antigravity',
   loader=<_frozen_importlib.SourceFileLoader ...>,
   origin='/usr/lib64/python3.4/antigravity.py')

>>> import io
>>> io.__spec__
ModuleSpec(name='io',
   loader=<_frozen_importlib.SourceFileLoader ...>,
   origin='/usr/lib64/python3.4/io.py')
```

import_module(name):
`sys.modules[name]`*? return it*

```
spec = find_spec(name, path)
load(spec)
module = sys.modules[name]
```

return `module`

- *Put module in* `sys.modules`
- *Initialize the module*

import_module(name):

```
   sys.modules[name]? return it


   spec = find_spec(name, path)
   load(spec)
   module = sys.modules[name]


   return module
```

- *Put module in* `sys.modules`
- *Initialize the module*

foo.py:

```
import bar

bar.do_bar()

def do_foo():
   print('Fooing!')
```

bar.py:

```
import foo

foo.do_foo()

def do_bar():
   print('Barring!')
```

import_module(name):

```
   sys.modules[name]? return it


   spec = find_spec(name, path)
   load(spec)
   module = sys.modules[name]


   return module
```

- *Put module in* `sys.modules`
- *Initialize the module*

foo.py:

```
import bar

bar.do_bar()

def do_foo():
   print('Fooing!')
```

bar.py:

```
import foo

foo.do_foo()

def do_bar():
   print('Barring!')
```

import_module(name):

```
   sys.modules[name]? return it


   spec = find_spec(name, path)
   load(spec)
   module = sys.modules[name]


   return module
```

- *Put module in* `sys.modules`
- *Initialize the module*

foo.py:

```
import bar

bar.do_bar()

def do_foo():
   print('Fooing!')
```

bar.py:

```
import foo

foo.do_foo()

def do_bar():
   print('Barring!')
```

import_module(name):

`sys.modules[name]`*? return it*

```
spec = find_spec(name, path)
load(spec)
module = sys.modules[name]
```

- *Put module in* `sys.modules`
- *Initialize the module*

return `module`

foo.py:

```
import bar

bar.do_bar()

def do_foo():
   print('Fooing!')
```

bar.py:

```
import foo

foo.do_foo()

def do_bar():
   print('Barring!')
```

import_module(name):

```
   sys.modules[name]? return it


   spec = find_spec(name, path)
   load(spec)
   module = sys.modules[name]


   return module
```

- *Put module in* `sys.modules`
- *Initialize the module*

foo.py:

```
import bar

bar.do_bar()

def do_foo():
   print('Fooing!')
```

bar.py:

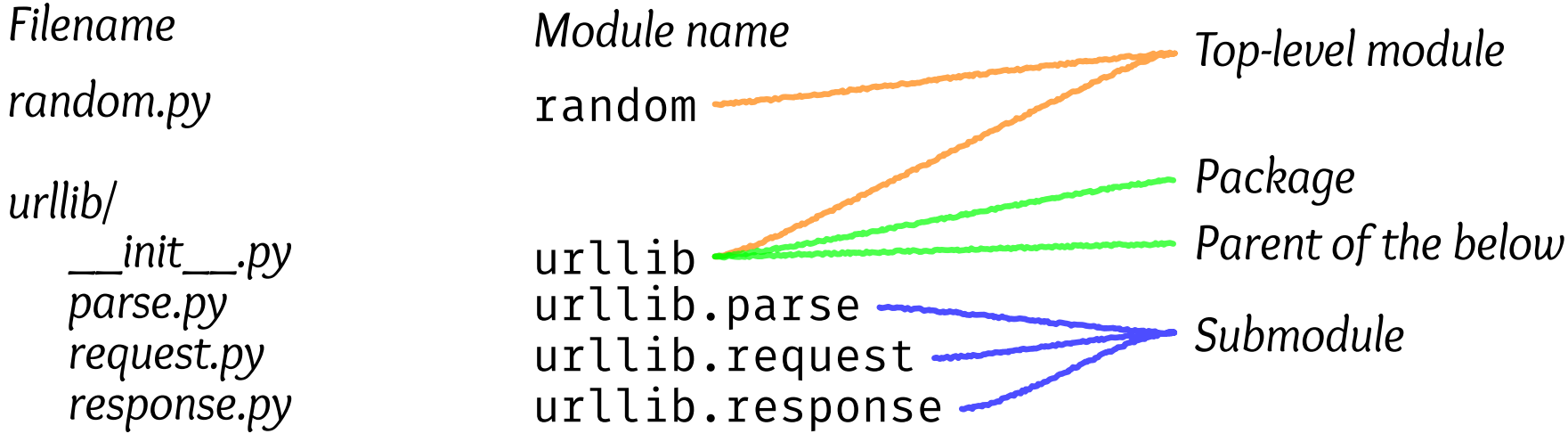
```
import foo

foo.do_foo()

def do_bar():
   print('Barring!')
```

import_module(name):
    `sys.modules[name]`*? return it*

```
    spec = find_spec(name, path)
    load(spec)
    module = sys.modules[name]
```

    return `module`

```
import_module(name):
    sys.modules[name]? return it


    spec = find_spec(name, path)
    load(spec)
    module = sys.modules[name]


    return module
```



```
import_module(name):
    sys.modules[name]? return it


    spec = find_spec(name, path)
    load(spec)
    module = sys.modules[name]


    return module
```

Filename

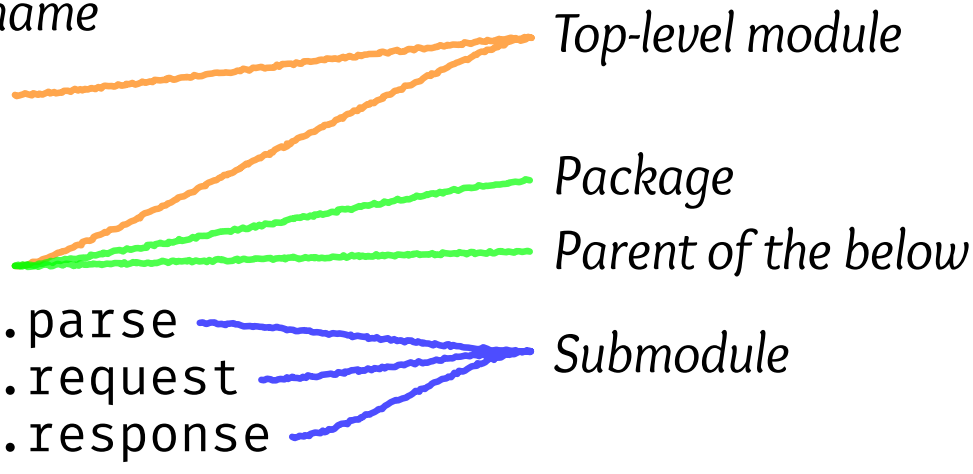
random.py

urllib/
- *__init__.py*
- *parse.py*
- *request.py*
- *response.py*

Module name

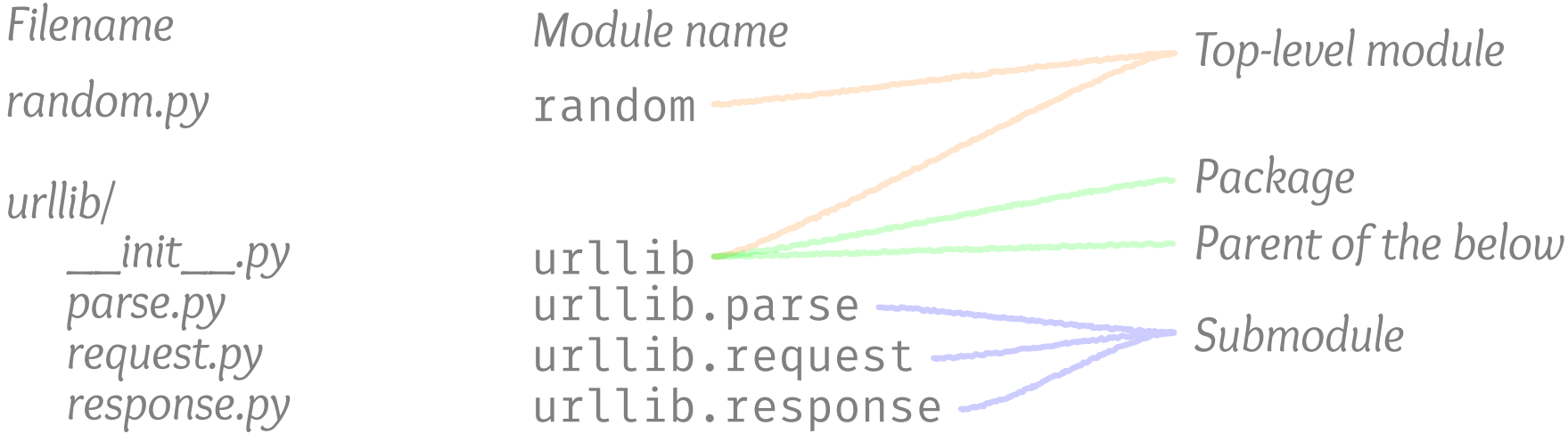
`random`

`urllib`
`urllib.parse`
`urllib.request`
`urllib.response`



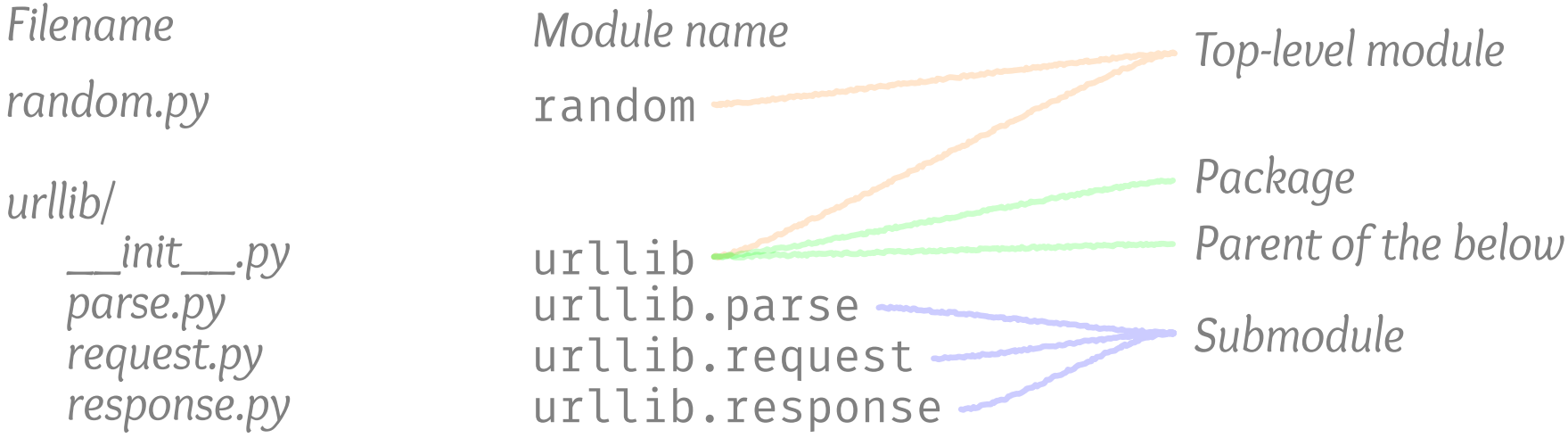
```
import_module(name):
    sys.modules[name]? return it

    spec = find_spec(name, path)
    load(spec)
    module = sys.modules[name]

    return module
```

```
import urllib.parse
>>> urllib.__path__
['/usr/lib64/python3.4/urllib']
```



```
import_module(name):
    sys.modules[name]? return it
    submodules:
        load parent module
        sys.modules[name]? return it
    spec = find_spec(name, path)
    load(spec)
    module = sys.modules[name]
    submodules:
        set module as attribute of parent
    return module
```

```
import urllib.parse
>>> urllib.__path__
['/usr/lib64/python3.4/urllib']
```



import_module(name):
    `sys.modules[name]`*? return it*
    submodules:
        load parent module
        `sys.modules[name]`*? return it*
    `spec = find_spec(name, path)`
    `load(spec)`
    `module = sys.modules[name]`
    submodules:
        set `module` *as attribute of* `parent`
    return `module`

- *Put module in* `sys.modules`
- *Initialize the module*

import_module(name):

```
    sys.modules[name]? return it
    submodules:
        load parent module
        sys.modules[name]? return it
    spec = find_spec(name, path)
    load(spec)
    module = sys.modules[name]
    submodules:
        set module as attribute of parent
    return module
```

- *Put module in* `sys.modules`
- *Initialize the module*

foo/__init__.py:

```
from foo import main

from foo import consts
```

foo/consts.py:

```
value = 42
```

foo/main.py:

```
import foo

use(foo.consts.value)
```

import_module(name):

```
    sys.modules[name]? return it
    submodules:
        load parent module
        sys.modules[name]? return it
    spec = find_spec(name, path)
    load(spec)
    module = sys.modules[name]
    submodules:
        set module as attribute of parent
    return module
```

- *Put module in* `sys.modules`
- *Initialize the module*

foo/__init__.py:

```
from foo import main

from foo import consts
```

foo/main.py:

```
import foo

use(foo.consts.value)
```

foo/consts.py:

```
value = 42
```

import_module(name):

```
    sys.modules[name]? return it
    submodules:
        load parent module
        sys.modules[name]? return it
    spec = find_spec(name, path)
    load(spec)
    module = sys.modules[name]
    submodules:
        set module as attribute of parent
    return module
```

- *Put module in* `sys.modules`
- *Initialize the module*

foo/__init__.py:

```
from foo import main

from foo import consts
```

foo/main.py:

```
import foo

use(foo.consts.value)
```

foo/consts.py:

```
value = 42
```

import_module(name):

```
    sys.modules[name]? return it
    submodules:
        load parent module
        sys.modules[name]? return it
    spec = find_spec(name, path)
    load(spec)
    module = sys.modules[name]
    submodules:
        set module as attribute of parent
    return module
```

- *Put module in* `sys.modules`
- *Initialize the module*

foo/__init__.py:

```
from foo import main

from foo import consts
```

foo/main.py:

```
import foo

use(foo.consts.value)
```

foo/consts.py:

```
value = 42
```

import_module(name):

```
    sys.modules[name]? return it
    submodules:
        load parent module
        sys.modules[name]? return it
    spec = find_spec(name, path)
    load(spec)
    module = sys.modules[name]
    submodules:
        set module as attribute of parent
    return module
```

- *Put module in* `sys.modules`
- *Initialize the module*

foo/__init__.py:

```
from foo import main

from foo import consts
```

foo/main.py:

```
import foo

use(foo.consts.value)
```

foo/consts.py:

```
value = 42
```

import_module(name):

```
    sys.modules[name]? return it
    submodules:
        load parent module
        sys.modules[name]? return it
    spec = find_spec(name, path)
    load(spec)
    module = sys.modules[name]
    submodules:
        set module as attribute of parent
    return module
```

- *Put module in* `sys.modules`
- *Initialize the module*

foo/__init__.py:

```
from foo import main

from foo import consts
```

foo/main.py:

```
import foo

use(foo.consts.value)
```

foo/consts.py:

```
value = 42
```

import_module(name):

```
    sys.modules[name]? return it
    submodules:
        load parent module
        sys.modules[name]? return it
    spec = find_spec(name, path)
    load(spec)
    module = sys.modules[name]
    submodules:
        set module as attribute of parent
    return module
```

- *Put module in* `sys.modules`
- *Initialize the module*

foo/__init__.py:

```
from foo import main

from foo import consts
```

foo/main.py:

```
import foo

use(foo.consts.value)
```

foo/consts.py:

```
value = 42
```

`__init__` should:

- *import from submodules*
- *set `__all__`*
- *nothing else*

submodules should:

- *not import directly from `__init__`*
- *not have internal import cycles*

import_module(name):
    `sys.modules[name]`*? return it*
    submodules:
        load parent module
        `sys.modules[name]`*? return it*
    `spec = find_spec(name, path)` *?*
    `load(spec)`
    `module = sys.modules[name]`
    submodules:
        set `module` *as attribute of* `parent`
    return `module`

Where's the module?

```
>>> import random
```

Where's the module?

```
>>> import random
>>> random
<module 'random' from '/usr/lib64/python3.4/random.py'>
```

Where's the module?

```
>>> import random
>>> random
<module 'random' from '/usr/lib64/python3.4/random.py'>
>>> random.__file__
'/usr/lib64/python3.4/random.py'
```

Where's the module?

```
>>> import random
>>> random
<module 'random' from '/usr/lib64/python3.4/random.py'>
>>> random.__file__
'/usr/lib64/python3.4/random.py'


>>> import sys
```

Where's the module?

```
>>> import random
>>> random
<module 'random' from '/usr/lib64/python3.4/random.py'>
>>> random.__file__
'/usr/lib64/python3.4/random.py'


>>> import sys
>>> sys
<module 'sys' (built-in)>
```

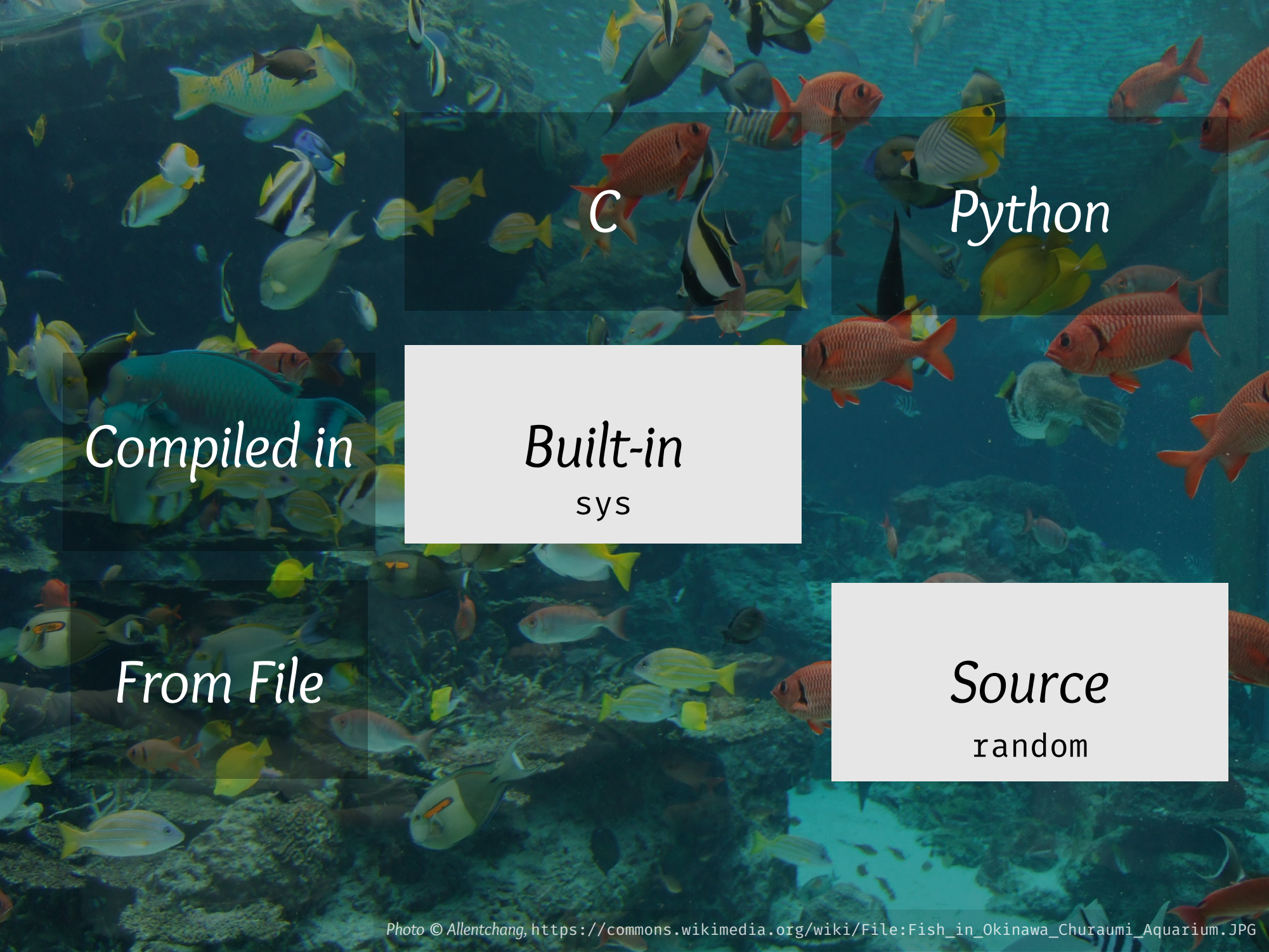
Where's the module?

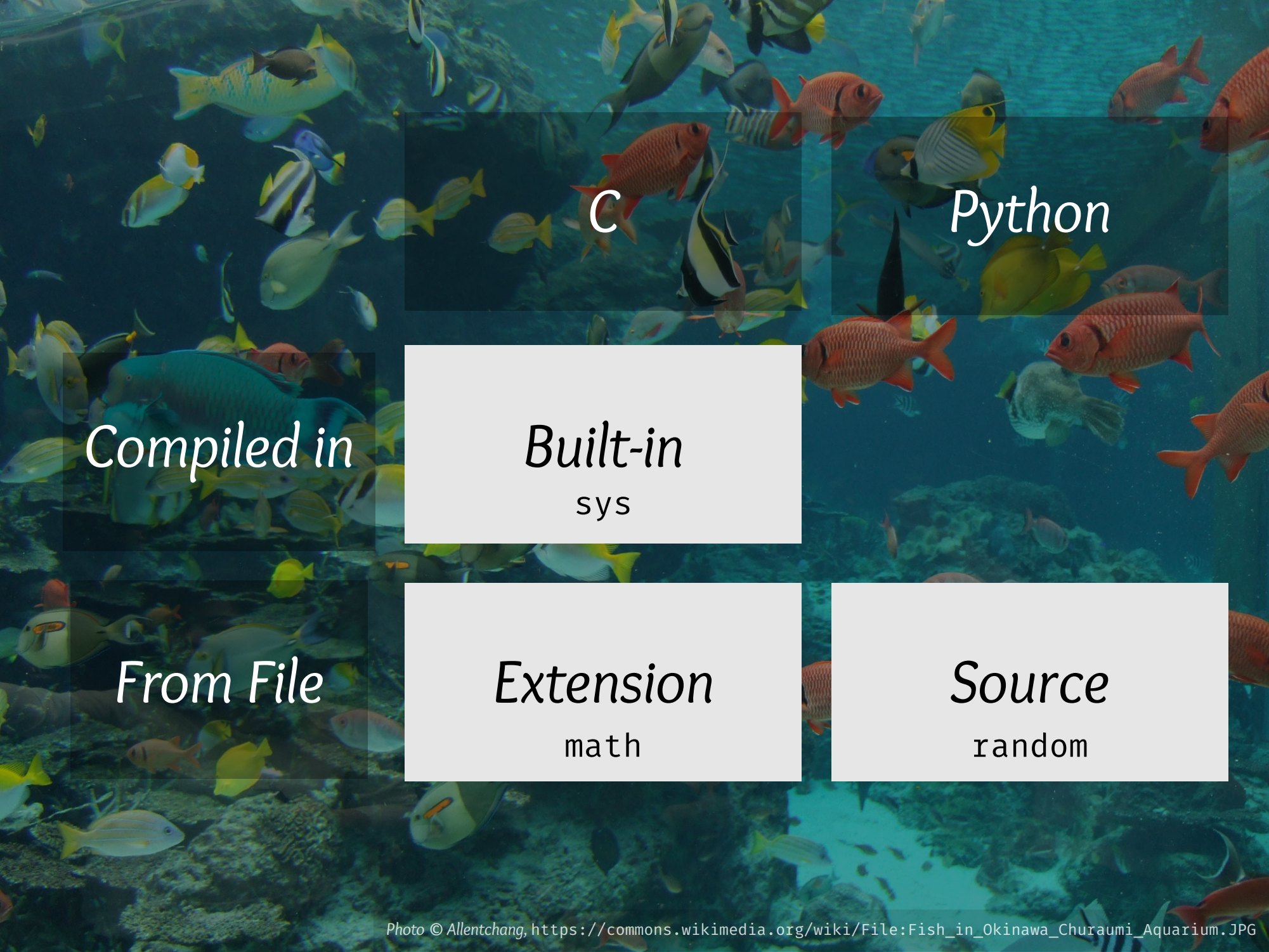
```
>>> import random
>>> random
<module 'random' from '/usr/lib64/python3.4/random.py'>
>>> random.__file__
'/usr/lib64/python3.4/random.py'


>>> import sys
>>> sys
<module 'sys' (built-in)>
>>> sys.__file__
Traceback (most recent call last):
   File "<stdin>", line 1, in <module>
AttributeError: 'module' object has no attribute '__file__'
```

Where's the module?

```
>>> import random
>>> random
<module 'random' from '/usr/lib64/python3.4/random.py'>
>>> random.__file__
'/usr/lib64/python3.4/random.py'


>>> import sys
>>> sys
<module 'sys' (built-in)>
>>> sys.__file__
Traceback (most recent call last):
   File "<stdin>", line 1, in <module>
AttributeError: 'module' object has no attribute '__file__'
```

/usr/bin/python3





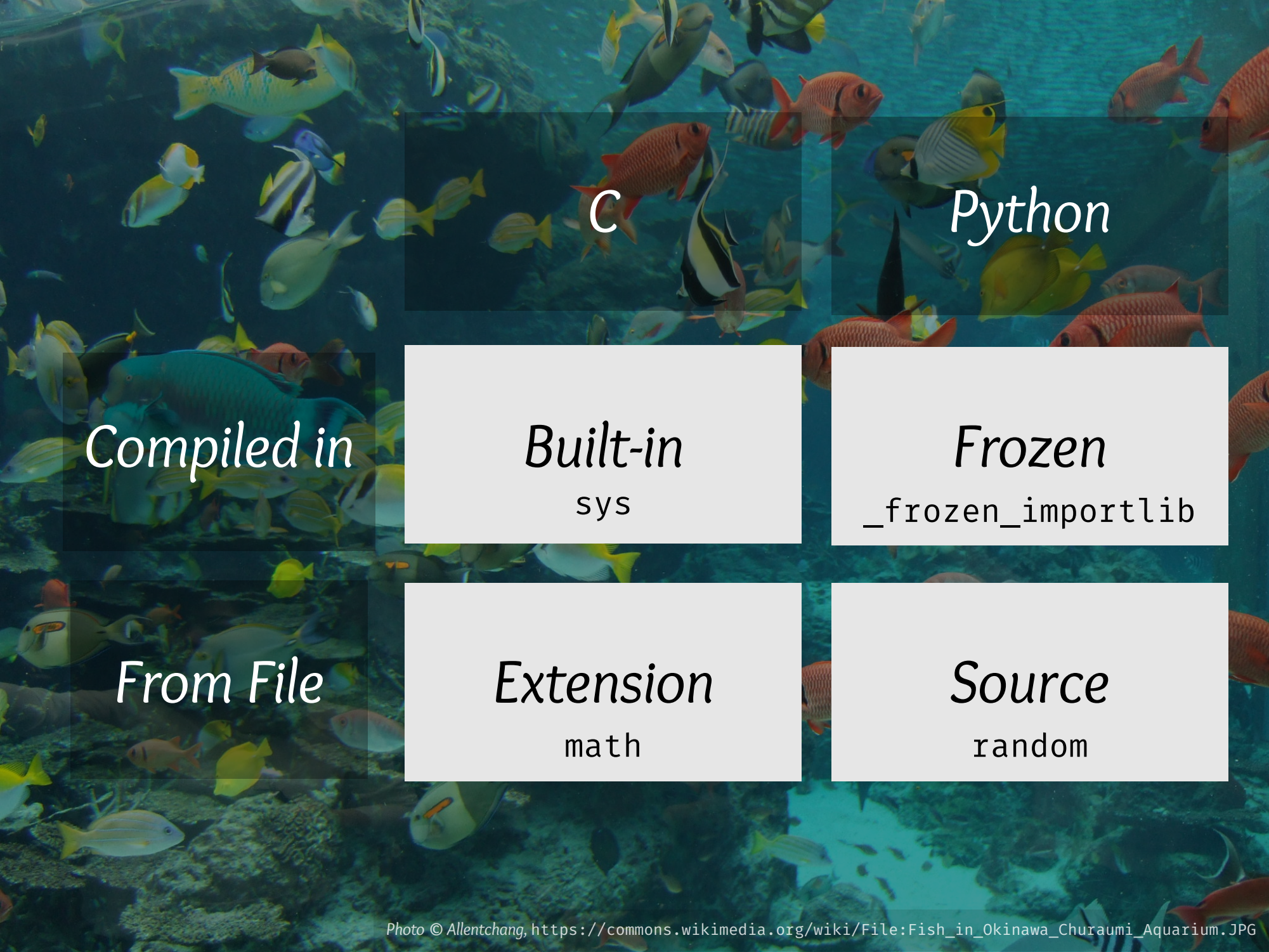


Photo © Allentchang, https://commons.wikimedia.org/wiki/File:Fish_in_Okinawa_Churaumi_Aquarium.JPG

```
>>> sys.meta_path
[<BuiltinImporter>,
 <FrozenImporter>,
 <PathFinder>]
```

find_spec(name, path):

```
    for finder in sys.meta_path:
```

call its `find_spec`*(name, path)*

return spec if successful

```
>>> sys.meta_path
[<BuiltinImporter>,
 <FrozenImporter>,
 <PathFinder>]
```

find_spec(name, path):
   `for finder in sys.meta_path:`
      call its `find_spec`*(name, path)*
      return spec if successful

```
>>> sys.meta_path
[<BuiltinImporter>,
 <FrozenImporter>,
 <PathFinder>]
```

```
>>> sys.path
['', '/usr/lib64/python34.zip', '/usr/lib/python3.4', ...]
```

find_spec(name, path):
   `for finder in sys.meta_path:`
      call its `find_spec`*(name, path)*
      return spec if successful

```
>>> sys.meta_path
[<BuiltinImporter>,
 <FrozenImporter>,
 <PathFinder>]


>>> sys.path_hooks
[<zipimporter>,
 <path_hook_for_FileFinder>]
```

```
>>> sys.path
['', '/usr/lib64/python34.zip', '/usr/lib/python3.4', ...]
```

find_spec(name, path):
   `for finder in sys.meta_path:`
      call its `find_spec`*(name, path)*
      return spec if successful

PathFinder.find_spec(name, path):
   `for directory in path:`
      get `sys.path_hooks` *entry*
      call its `find_spec`*(name)*
      return spec if successful

```
>>> sys.meta_path
[<BuiltinImporter>,
 <FrozenImporter>,
 <PathFinder>]
```

```
>>> sys.path_hooks
[<zipimporter>,
 <path_hook_for_FileFinder>]
```

```
>>> sys.path
['', '/usr/lib64/python34.zip', '/usr/lib/python3.4', ...]
```

find_spec(name, path):
   `for finder in sys.meta_path:`
      call its `find_spec`*(name, path)*
      return spec if successful

PathFinder.find_spec(name, path):
   `for directory in path:`
      get `sys.path_hooks` *entry*
      call its `find_spec`*(name)*
      return spec if successful

```
>>> sys.meta_path
[<BuiltinImporter>,
 <FrozenImporter>,
 <PathFinder>]
```

```
>>> sys.path_hooks
[<zipimporter>,
 <path_hook_for_FileFinder>]
```

```
>>> sys.path
['', '/usr/lib64/python34.zip', '/usr/lib/python3.4', ...]
```

`random.cpython-34m.so`	*(Extension module)*
`random.abi3.so`	*(Extension module)*
`random.so`	*(Extension module)*
`random.py`	*(Source module)*
`random.pyc`	*(Sourceless module)*

find_spec(name, path):
   `for finder in sys.meta_path:`
      call its `find_spec`*(name, path)*
      return spec if successful

PathFinder.find_spec(name, path):
   `for directory in path:`
      get `sys.path_hooks` *entry*
      call its `find_spec`*(name)*
      return spec if successful

`ModuleSpec:`
   `name`
    random
   `origin`
    /usr/lib64/python3.4/random.py
   `cached`
    /usr/lib64/python3.4/__pycache__/random.cpython-34.pyc
   `loader`
    importlib.machinery.SourceFileLoader
   `loader_state, parent, submodule_search_locations`

import_module(name):
    `sys.modules[name]`*? return it*
    submodules:
        load parent module
        `sys.modules[name]`*? return it*
    `spec = find_spec(name, path)`
    `load(spec)`
    `module = sys.modules[name]`
    submodules:
        set `module` *as attribute of* `parent`
    return `module`

?

```
load(spec):
    module = spec.loader.create_module(spec)
    if module is None:
        module = types.ModuleType(spec.name)
    set initial module attributes
    sys.modules[spec.name] = module
    spec.loader.exec_module(module, spec)
```

```
load(spec):
    module = spec.loader.create_module(spec)
    if module is None:
        module = types.ModuleType(spec.name)
    set initial module attributes
    sys.modules[spec.name] = module
    spec.loader.exec_module(module, spec)
```

```
load(spec):
    module = spec.loader.create_module(spec)
    if module is None:
        module = types.ModuleType(spec.name)
    set initial module attributes
    sys.modules[spec.name] = module
    spec.loader.exec_module(module, spec)
```

```
spec                    → __spec__
spec.name               → __name__
spec.loader             → __loader__
spec.parent             → __package__
spec.origin             → __file__
spec.cached             → __cached__
spec.submodule_search_locations
                        → __path__
```

```
load(spec):
    module = spec.loader.create_module(spec)
    if module is None:
        module = types.ModuleType(spec.name)
    set initial module attributes
    sys.modules[spec.name] = module
    spec.loader.exec_module(module, spec)
```

```
spec                    → __spec__
spec.name               → __name__
spec.loader             → __loader__
spec.parent             → __package__
spec.origin             → __file__
spec.cached             → __cached__
spec.submodule_search_locations
                        → __path__
```

```
>>> import __main__
>>> a = 'hello'
>>> __main__.a
'hello'
>>> __main__.a = 'world'
>>> a
'world'
```

load(spec):
    module = `spec.loader.create_module(spec)`
    if module is None:
        module = `types.ModuleType(spec.name)`
    set initial module attributes
    `sys.modules[spec.name]` = *module*
    `spec.loader.exec_module`*(module, spec)*

`spec`	→ *__spec__*
`spec.name`	→ *__name__*
`spec.loader`	→ *__loader__*
`spec.parent`	→ *__package__*
`spec.origin`	→ *__file__*
`spec.cached`	→ *__cached__*
`spec.submodule_search_locations`	→ *__path__*

```
>>> import __main__
>>> a = 'hello'
>>> __main__.a
'hello'
>>> __main__.a = 'world'
>>> a
'world'
```

```
if __name__ == '__main__'
```

```
load(spec):
    module = spec.loader.create_module(spec)
    if module is None:
        module = types.ModuleType(spec.name)
    set initial module attributes
    sys.modules[spec.name] = module
    spec.loader.exec_module(module, spec)
```

```
spec                    → __spec__
spec.name               → __name__
spec.loader             → __loader__
spec.parent             → __package__
spec.origin             → __file__
spec.cached             → __cached__
spec.submodule_search_locations
                        → __path__
```

```
>>> import __main__
>>> a = 'hello'
>>> __main__.a
'hello'
>>> __main__.a = 'world'
>>> a
'world'
```

```
if __name__ == '__main__'
```

```
load(spec):
    module = spec.loader.create_module(spec)
    if module is None:
        module = types.ModuleType(spec.name)
    set initial module attributes
    sys.modules[spec.name] = module
    spec.loader.exec_module(module, spec)
```

- *Put module in* `sys.modules`
- *Initialize the module*

get_code(module, spec):
    if `spec.cached` *exists, and matches* `origin` *stats,*
        return it!
    load source from `origin` *and compile it*
    write bytecode to `spec.cached`

```
ModuleSpec:
    name
      random
    origin
      /usr/lib64/python3.4/random.py
    cached
      /usr/lib64/python3.4/__pycache__/random.cpython-34.pyc
   ...
```

get_code(module, spec):
    if `spec.cached` *exists, and matches* `origin` *stats,*
        return it!
    load source from `origin` *and compile it*
    write bytecode to `spec.cached`

Python 2

```
somemodule.py
somemodule.pyc
```

get_code(module, spec):
    if `spec.cached` *exists, and matches* `origin` *stats,*
        return it!
    load source from `origin` *and compile it*
    write bytecode to `spec.cached`

Python 2

`somemodule.py`
`somemodule.pyc`

get_code(module, spec):
    if `spec.cached` *exists, and matches* `origin` *stats,*
        return it!
    load source from `origin` *and compile it*
    write bytecode to `spec.cached`

Python 2

```
somemodule.py
somemodule.pyc
```

Python 3

```
somemodule.py
__pycache__/
    somemodule.cpython-34.pyc
```

get_code(module, spec):
    if `spec.cached` *exists, and matches* `origin` *stats,*
        return it!
    load source from `origin` *and compile it*
    write bytecode to `spec.cached`

Python 2

```
somemodule.py
somemodule.pyc
```

Python 3

```
somemodule.py
__pycache__/
    somemodule.cpython-34.pyc
```

Or:

```
somemodule.py
somemodule.pyc
```

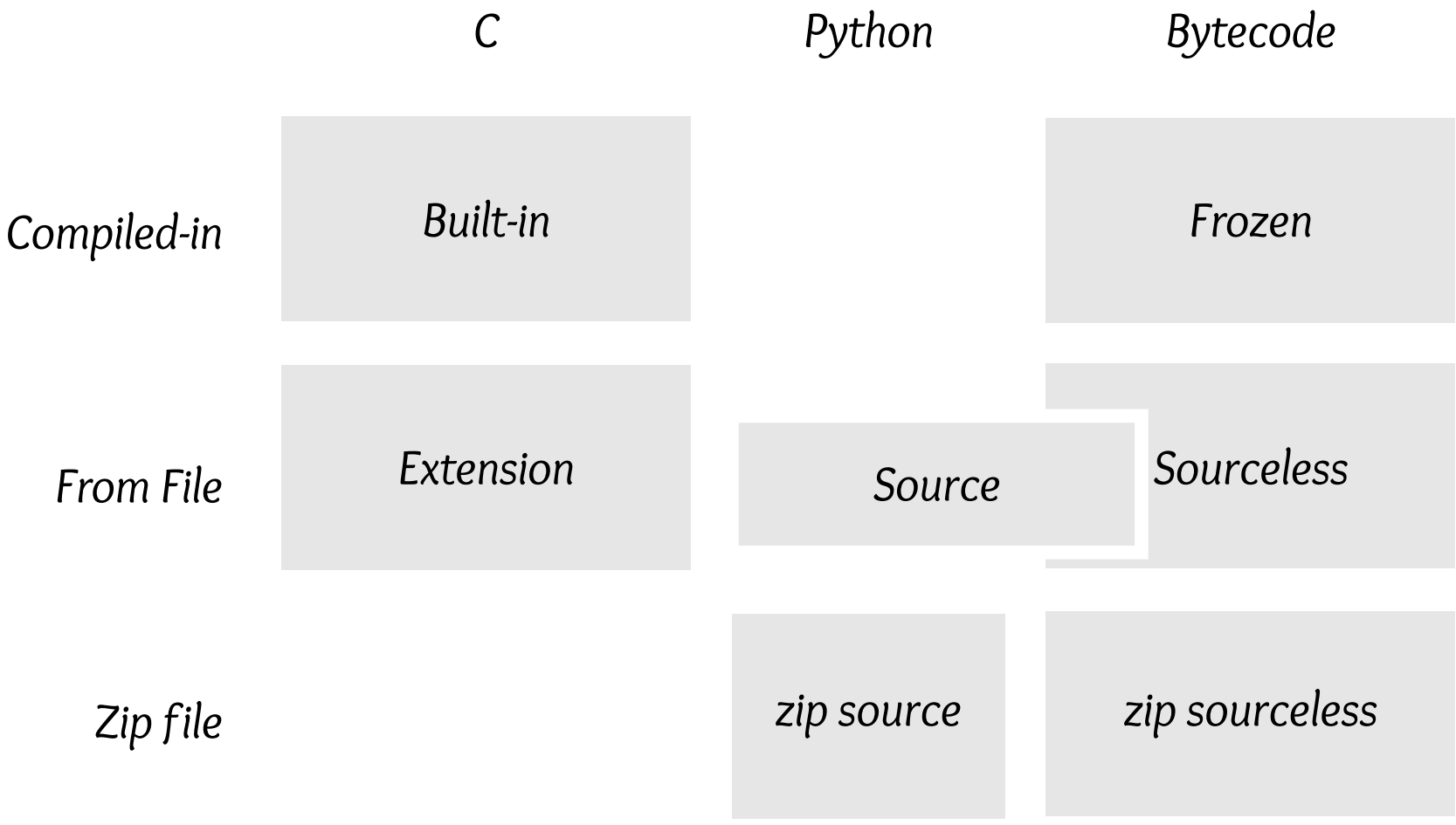
import_module(name):
    `sys.modules[name]`*? return it*
    submodules:
        load parent module
        `sys.modules[name]`*? return it*
    `spec = find_spec(name, path)`
    `load(spec)`
    `module = sys.modules[name]`
    submodules:
        set `module` *as attribute of* `parent`
    return `module`

find_spec(name, path):
    `for finder in sys.meta_path:`
        call its `find_spec`*(name, path)*
        return spec if successful

PathFinder.find_spec(name, path):
    `for directory in path:`
        get `sys.path_hooks` *entry*
        call its `find_spec`*(name)*
        return spec if successful

load(spec):
    module = `spec.loader.create_module(spec)`
    if module is None:
        module = `types.ModuleType(spec.name)`
    set initial module attributes
    `sys.modules[spec.name]` *= module*
    `spec.loader.exec_module`*(module, spec)*

get_code(module, spec):
    if `spec.cached` *exists, and matches* `origin` *stats,*
        return it!
    load source from `origin` *and compile it*
    write bytecode to `spec.cached`


C
Python
Bytecode
Compiled-in
Built-in
Frozen
From File
Extension
Source
Sourceless
Zip file
zip source
zip sourceless