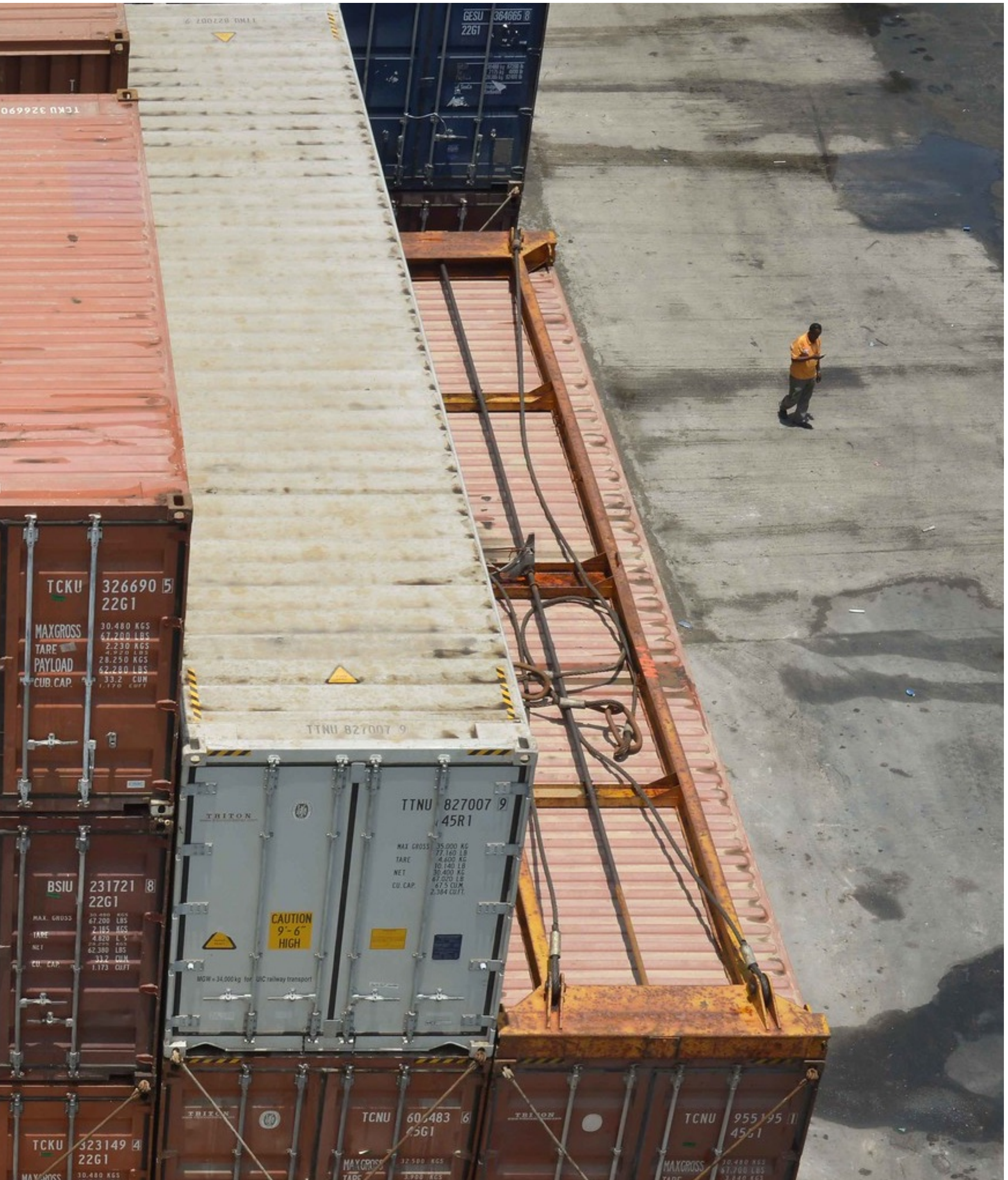


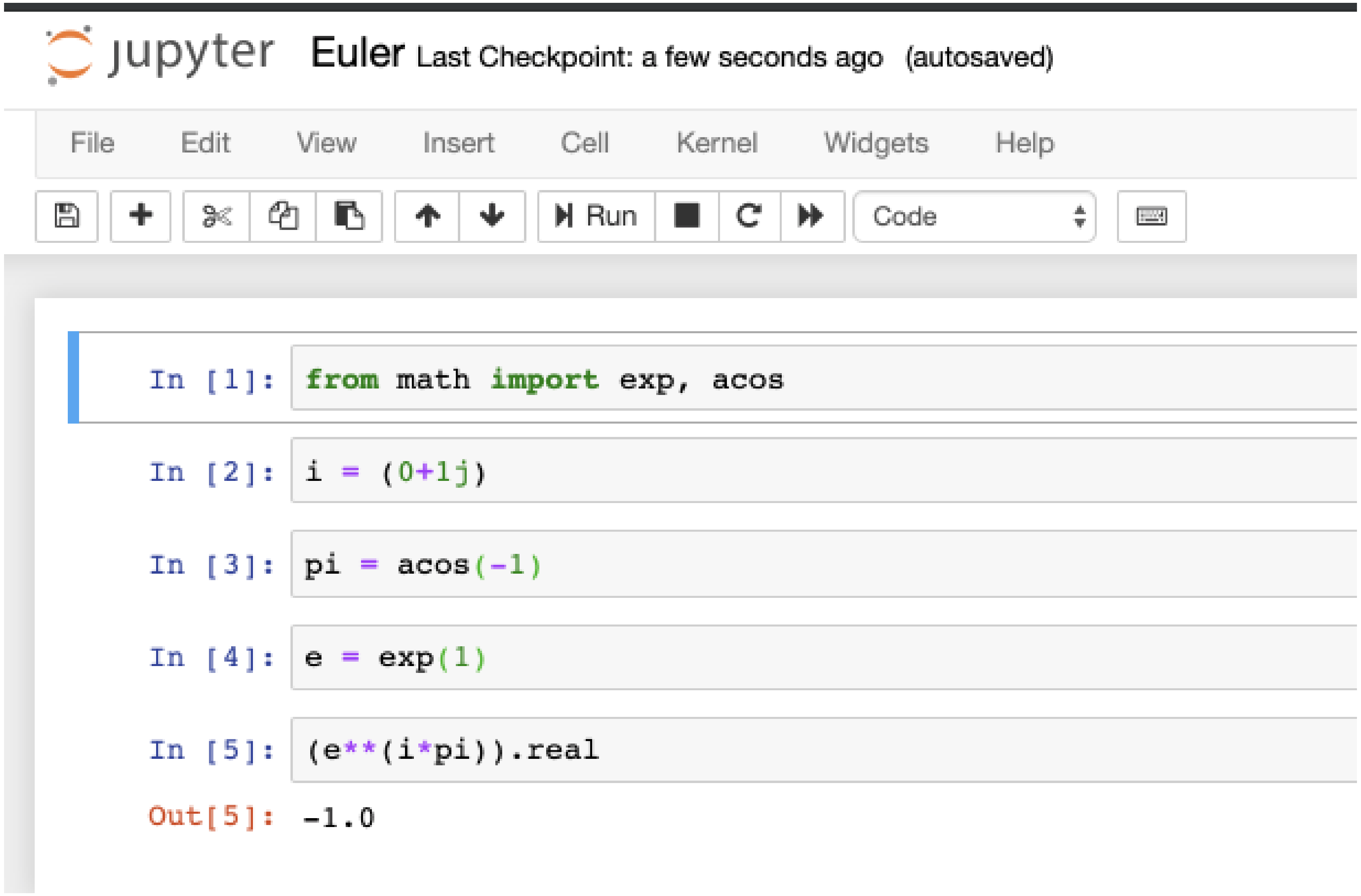
Python Imports

Vlad Korolev
vlad@dblfuzzr.com

Imports

Easy

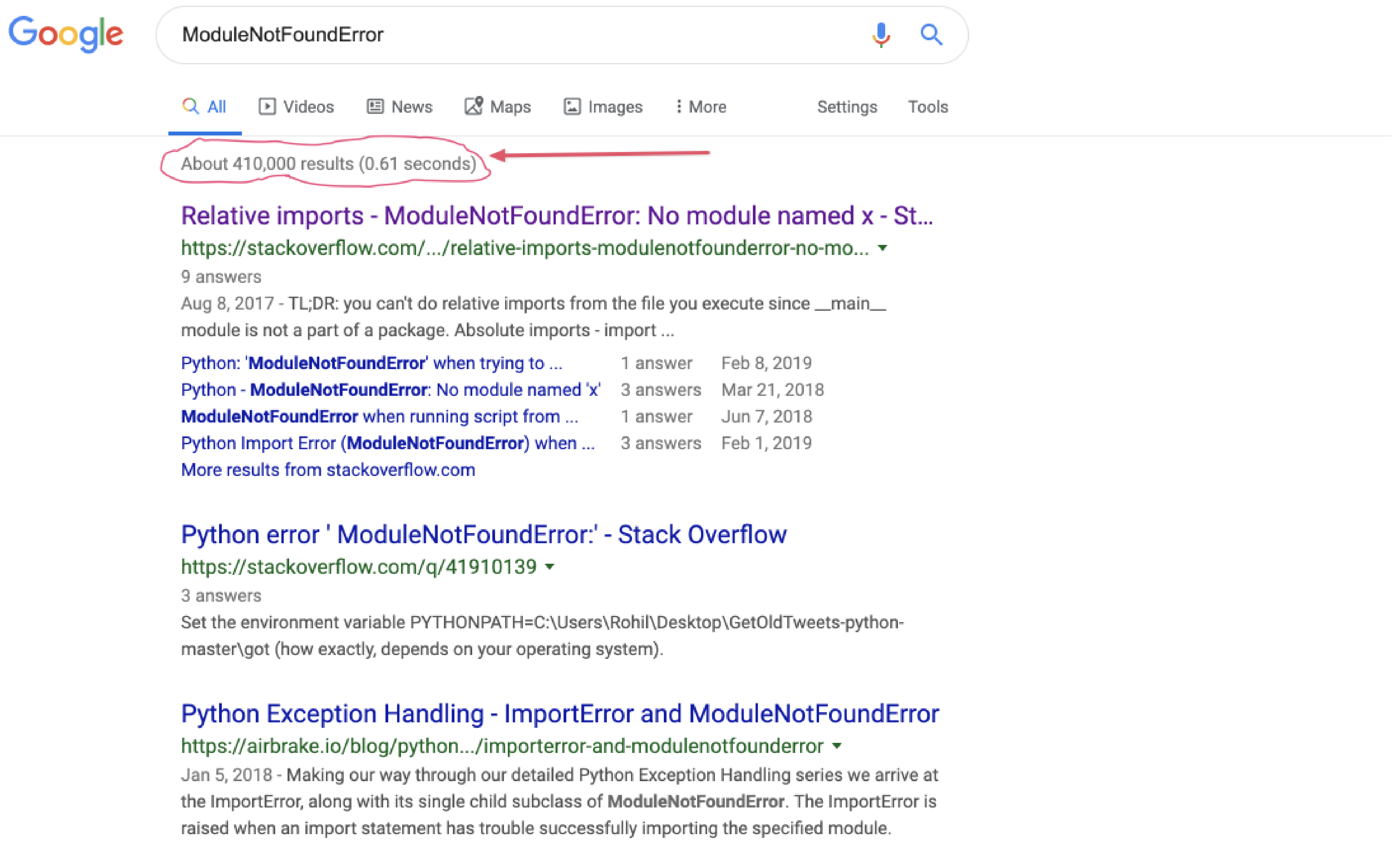
Easy



```
In [1]: from math import exp, acos

In [2]: i = (0+1j)

In [3]: pi = acos(-1)

In [4]: e = exp(1)

In [5]: (e**(i*pi)).real

Out[5]: -1.0
```

Thank you

Problems



About 410,000 results (0.61 seconds)

Relative imports - ModuleNotFoundError: No module named x - St...
https://stackoverflow.com/.../relative-imports-modulenotfounderror-no-mo...
9 answers
Aug 8, 2017 - TL;DR: you can't do relative imports from the file you execute since __main__ module is not a part of a package. Absolute imports - import ...

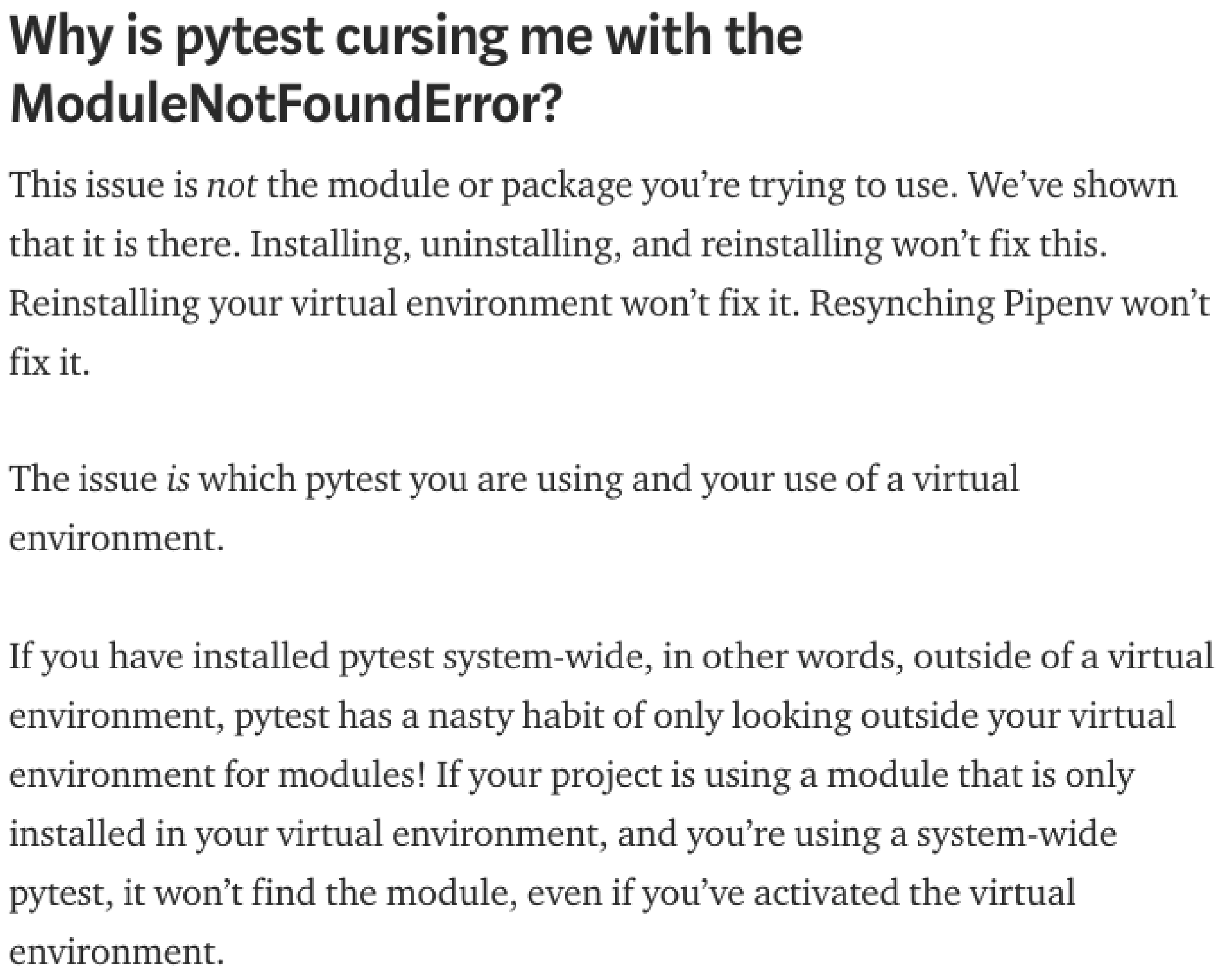
Python: '**ModuleNotFoundError**' when trying to ...	1 answer	Feb 8, 2019
Python - **ModuleNotFoundError**: No module named 'x'	3 answers	Mar 21, 2018
ModuleNotFoundError when running script from ...	1 answer	Jun 7, 2018
Python Import Error (**ModuleNotFoundError**) when ...	3 answers	Feb 1, 2019

More results from stackoverflow.com

Python error ' ModuleNotFoundError:' - Stack Overflow
https://stackoverflow.com/q/41910139
3 answers
Set the environment variable PYTHONPATH=C:\Users\Rohil\Desktop\GetOldTweets-python-master\got (how exactly, depends on your operating system).

Python Exception Handling - ImportError and ModuleNotFoundError
https://airbrake.io/blog/python.../importerror-and-modulenotfounderror
Jan 5, 2018 - Making our way through our detailed Python Exception Handling series we arrive at the ImportError, along with its single child subclass of **ModuleNotFoundError**. The ImportError is raised when an import statement has trouble successfully importing the specified module.

Problems



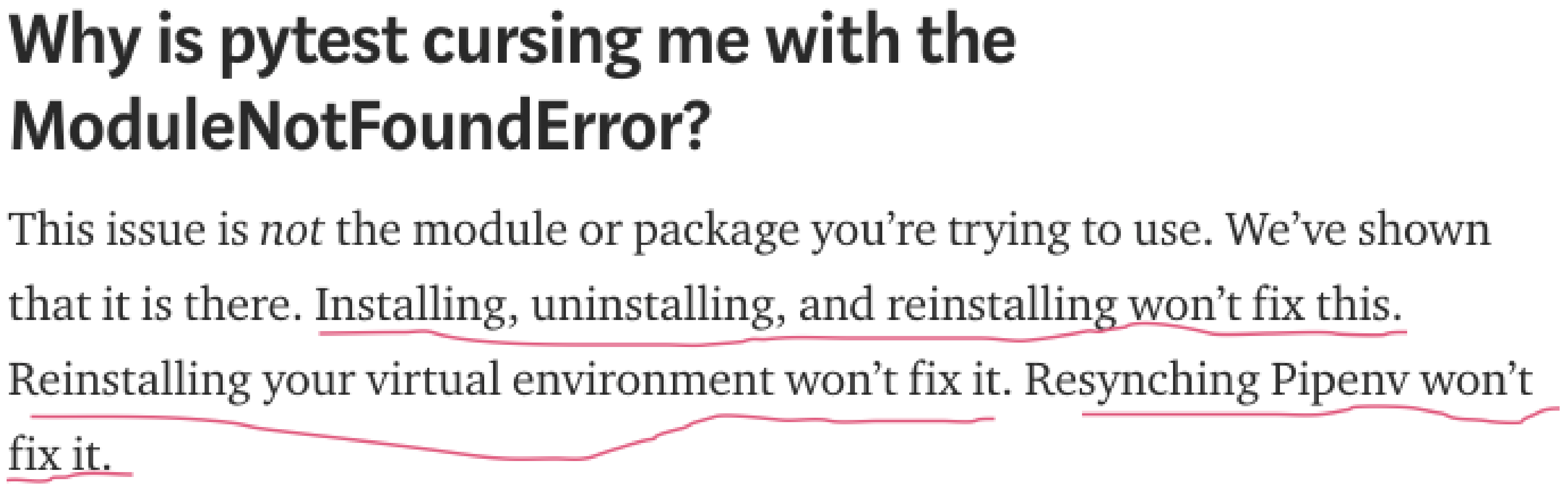
Why is pytest cursing me with the ModuleNotFoundError?

This issue is *not* the module or package you're trying to use. We've shown that it is there. Installing, uninstalling, and reinstalling won't fix this. Reinstalling your virtual environment won't fix it. Resynching Pipenv won't fix it.

The issue *is* which pytest you are using and your use of a virtual environment.

If you have installed pytest system-wide, in other words, outside of a virtual environment, pytest has a nasty habit of only looking outside your virtual environment for modules! If your project is using a module that is only installed in your virtual environment, and you're using a system-wide pytest, it won't find the module, even if you've activated the virtual environment.

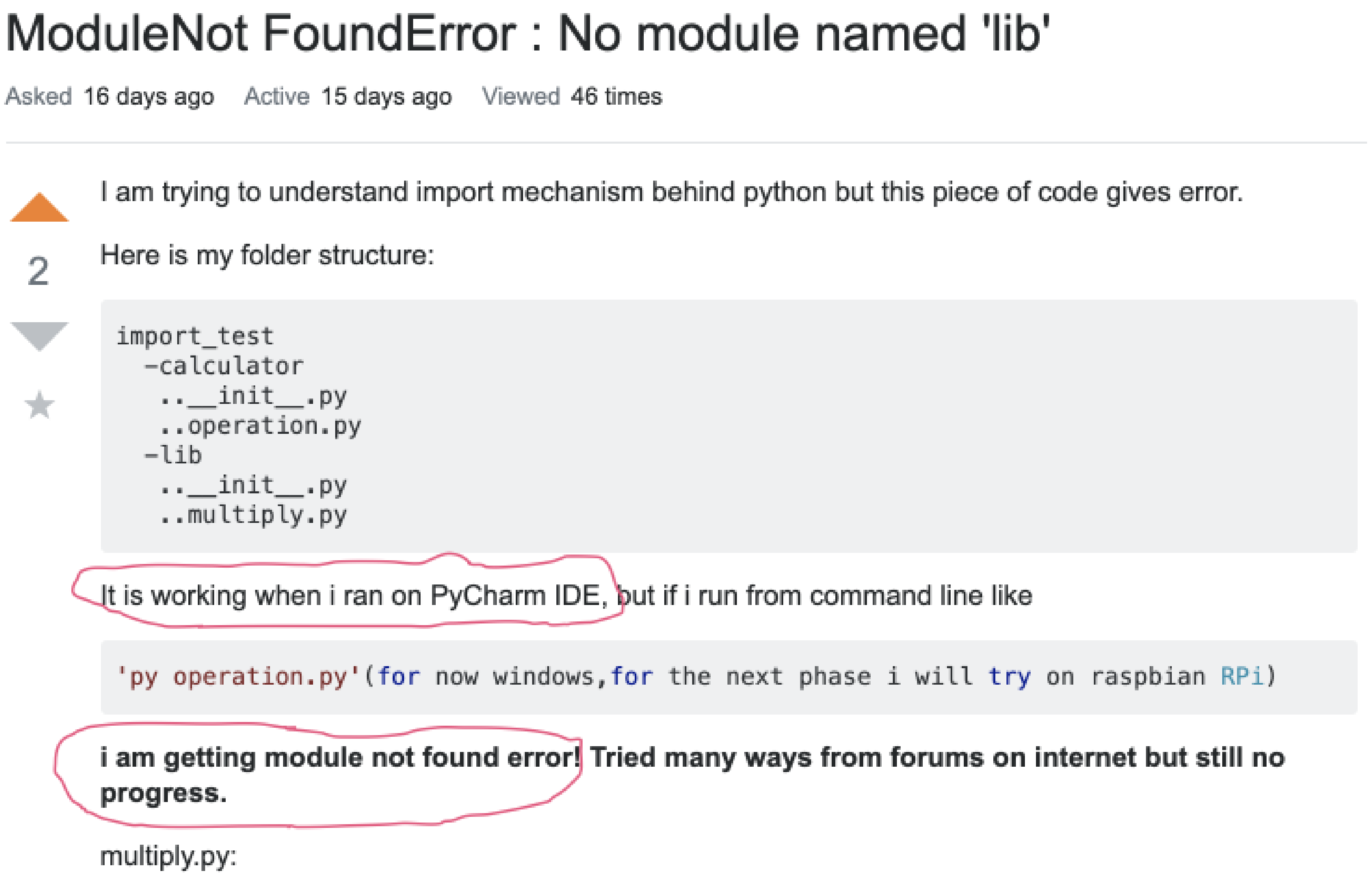
Problems



Why is pytest cursing me with the ModuleNotFoundError?

This issue is *not* the module or package you're trying to use. We've shown that it is there. Installing, uninstalling, and reinstalling won't fix this. Reinstalling your virtual environment won't fix it. Resynching Pipenv won't fix it.

Problems



ModuleNot FoundError : No module named 'lib'

Asked 16 days ago Active 15 days ago Viewed 46 times

2

I am trying to understand import mechanism behind python but this piece of code gives error.

Here is my folder structure:

```
import_test
  -calculator
   ..__init__.py
   ..operation.py
  -lib
   ..__init__.py
   ..multiply.py
```

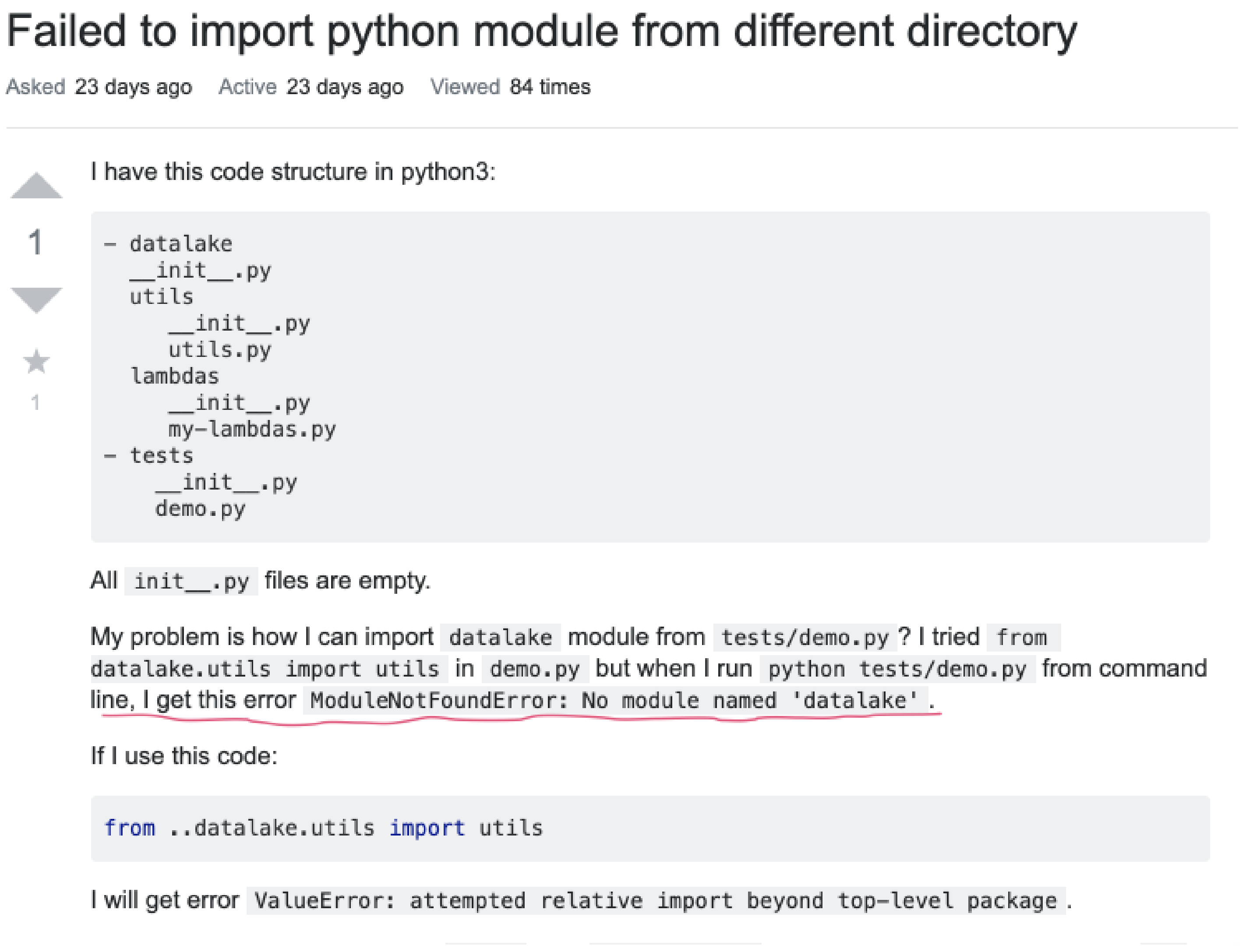
It is working when i ran on PyCharm IDE, but if i run from command line like

```
'py operation.py'(for now windows,for the next phase i will try on raspbian RPi)
```

i am getting module not found error! Tried many ways from forums on internet but still no progress.

multiply.py:

Problems



Failed to import python module from different directory

Asked 23 days ago Active 23 days ago Viewed 84 times

1

1

I have this code structure in python3:

```
- datalake
  __init__.py
  utils
     __init__.py
     utils.py
  lambdas
     __init__.py
     my-lambdas.py
- tests
    __init__.py
    demo.py
```

All `init__.py` files are empty.

My problem is how I can import `datalake` module from `tests/demo.py`? I tried `from datalake.utils import utils` in `demo.py` but when I run `python tests/demo.py` from command line, I get this error `ModuleNotFoundError: No module named 'datalake'`.

If I use this code:

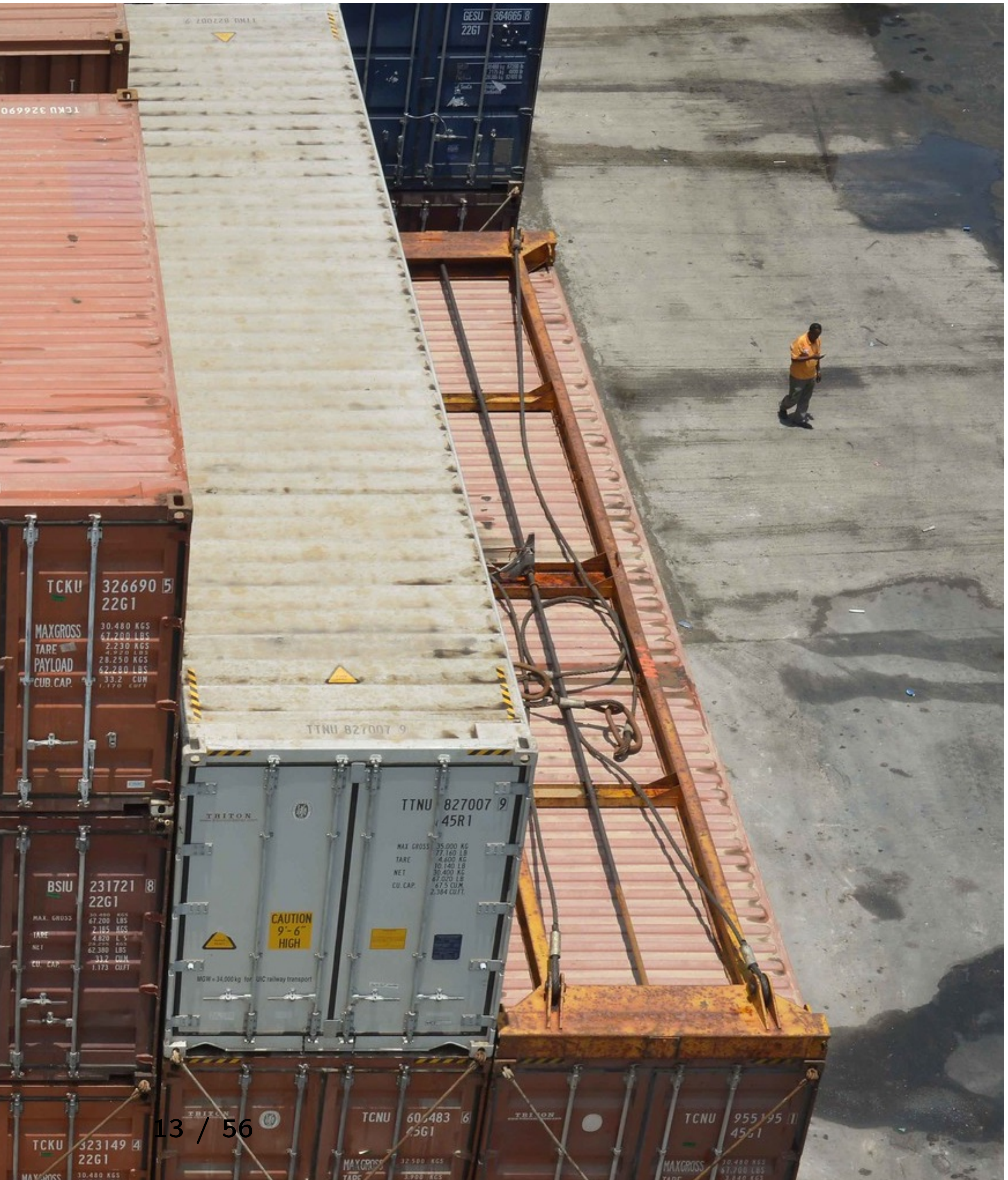
```
from ..datalake.utils import utils
```

I will get error `ValueError: attempted relative import beyond top-level package`.

Not so easy

Not so easy

In fact it's insane



Python Imports

Vlad Korolev
vlad@dblfuzzr.com

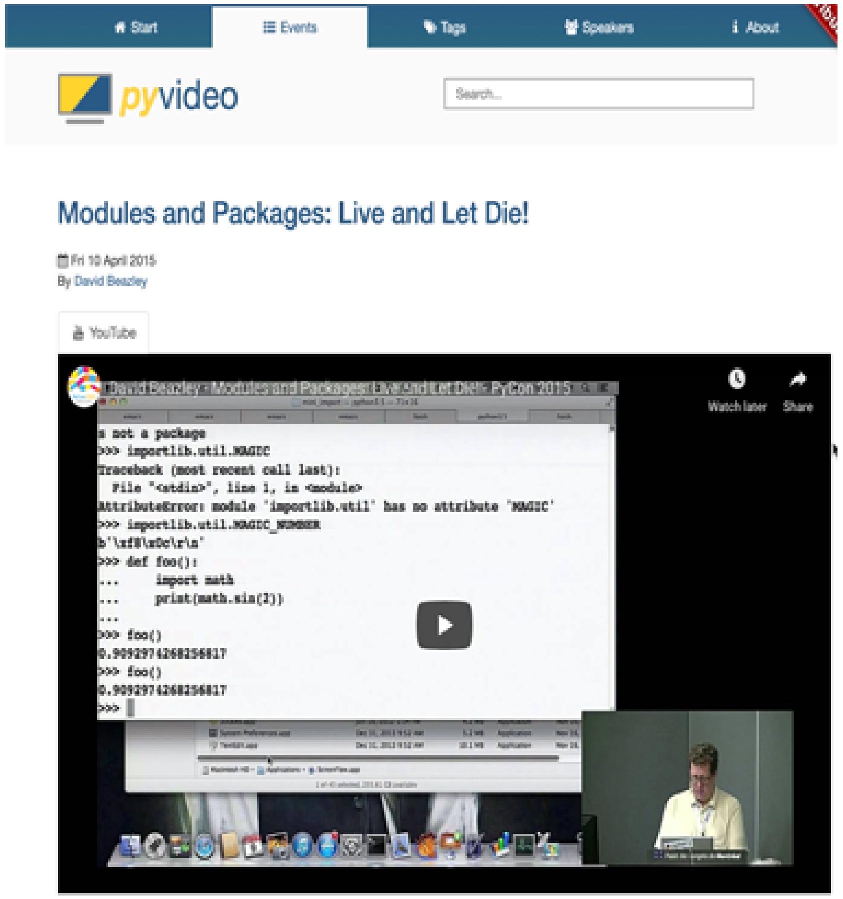
About myself

- Been using python since 1998
- Contributed to python core

About myself

- Been using python since 1998
- Contributed to python core
- Been driven crazy by import errors

Sources

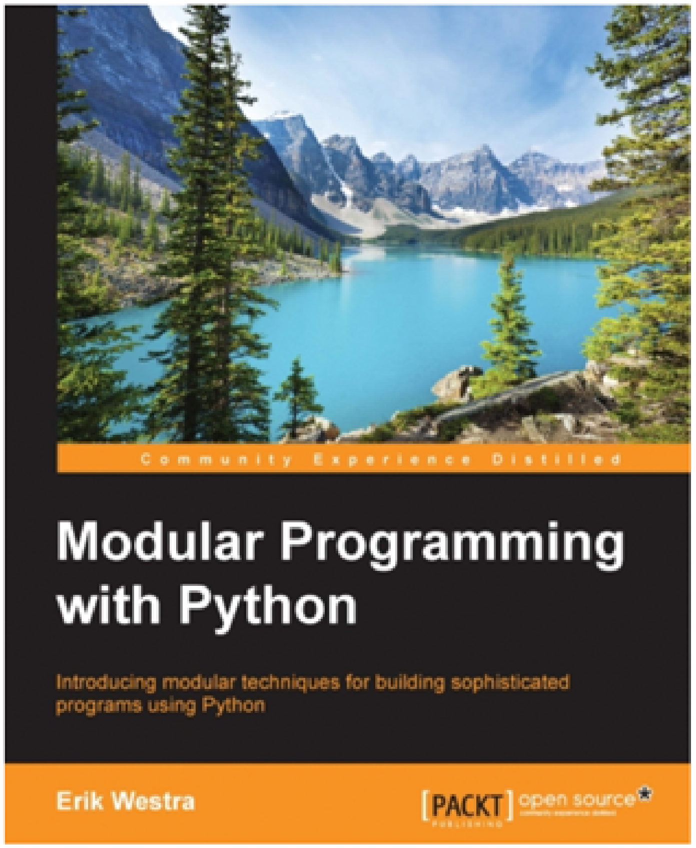


Modules and Packages: Live and Let Die!

Fri 10 April 2015
By David Beazley

David Beazley - Modules and Packages: Live And Let Die! - PyCon 2015

```
s not a package
>>> importlib.util.MAGIC
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
AttributeError: module 'importlib.util' has no attribute 'MAGIC'
>>> importlib.util.MAGIC_NUMBER
b'\xf8\x0c\r\n'
>>> def foo():
...     import math
...     print(math.sin(2))
...
>>> foo()
0.9092974268256817
>>> foo()
0.9092974268256817
>>> 
```


Community Experience Distilled
Modular Programming with Python
Introducing modular techniques for building sophisticated programs using Python
Erik Westra
[PACKT] open source

About this talk

- Summary of David's tutorial and Erik's book
- David's tutorial is 3 hrs
- Erik's book more 200 pages
- This talk is 15 minutes
- Applies to Python 3.5 and above
- Intended audience (myself)

TL;DR

- Mostly works right from Notebook or script
- Gets interesting
    - Test environments
    - Web apps running from Apache/NGINX/uWSGI
    - Serverless Lambda functions
    - Anytime you'll try to scale up your script

TL;DR : Why

- Search path confusion
- Missing or extra __init__ files
- Exotic issues
- Misunderstanding of the import and module system

Why import ?

Why import ?

- Re-use the code
- From someone else
  - Core Developers

```
from math import cos
```

  - Random people

```
import requests
```

- Ourselves

```
from x import *
```

What import?

Module

Module

- Used to resolve symbols
- Python type (dict)
- Holds names and references
- Every source file into own module
- Loaded exactly once
- Shared between importers

Module

script.py

```python
import a.a as a
import b.b as b
a.set_val(95)
print(b.get_val())
print(a.get_val())
```

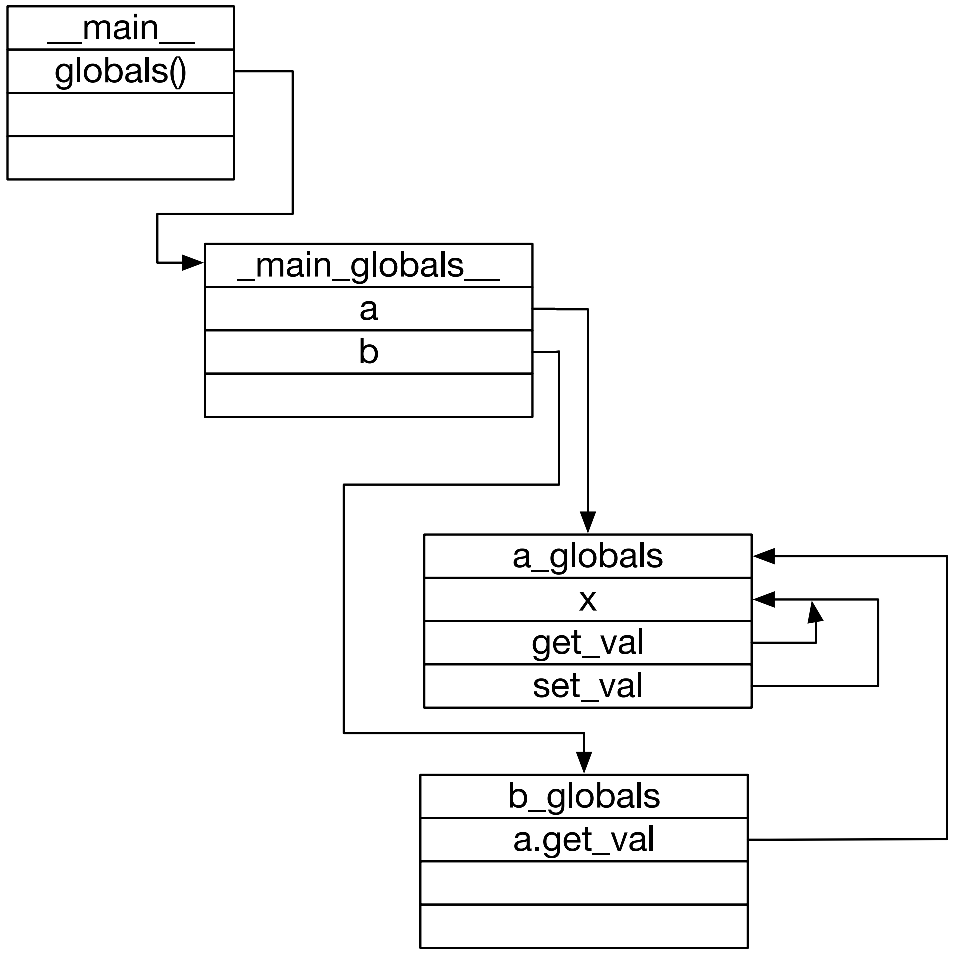
a/a.py

```python
x = 10
def set_val(a): global x ; x = a
def get_val(): return x
def globs(): return globals().keys()
```

b/b.py

```python
import a.a
def get_val(): return a.a.get_val()
def globs(): return globals().keys()
```

Module : Sharing



script.py

```
import a.a as a
import b.b as b
a.set_val(95)
print(b.get_val())
print(a.get_val())
```

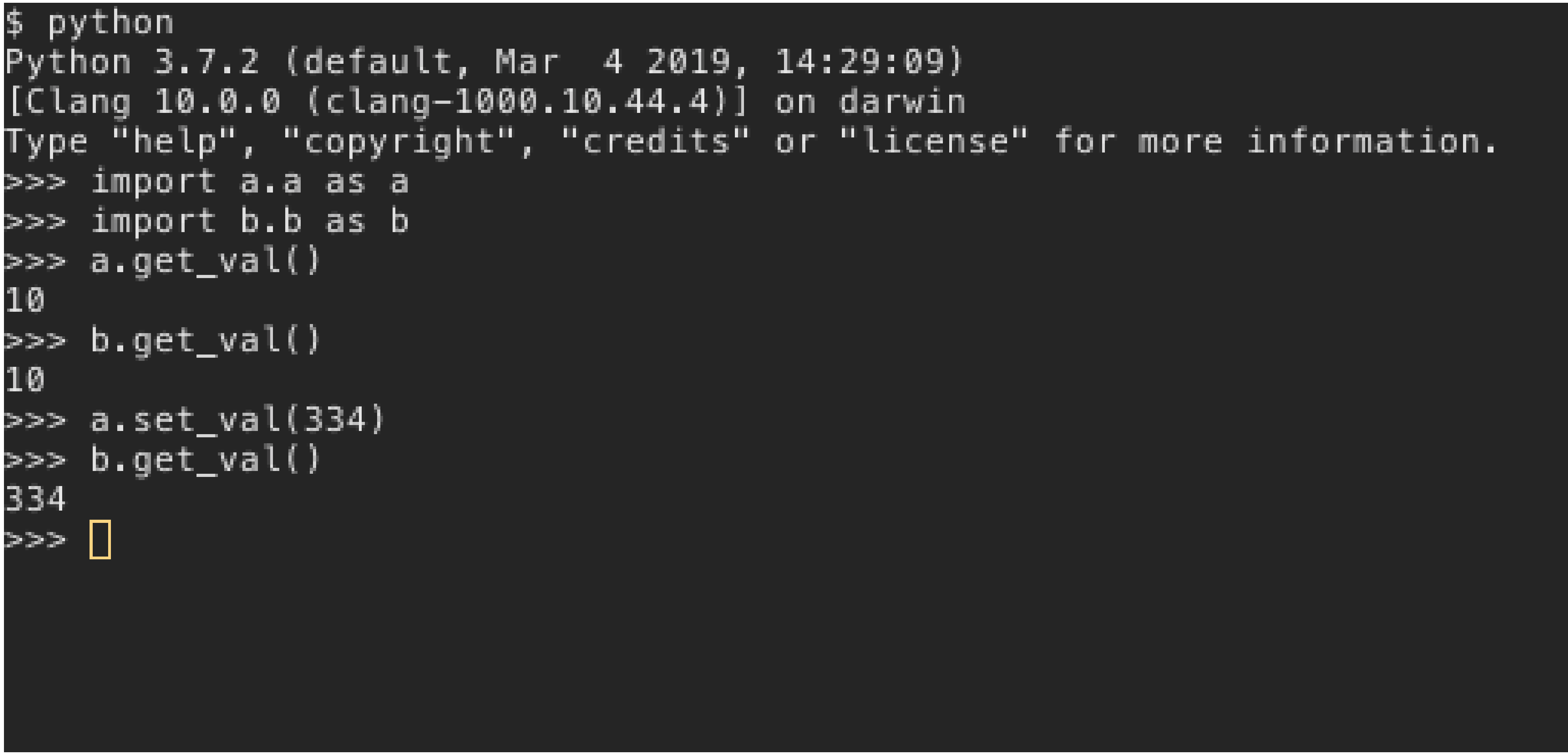
a/a.py

```
x = 10
def set_val(a): global x; x = a
def get_val(): return x
def globs(): return globals().ke
```

b/b.py

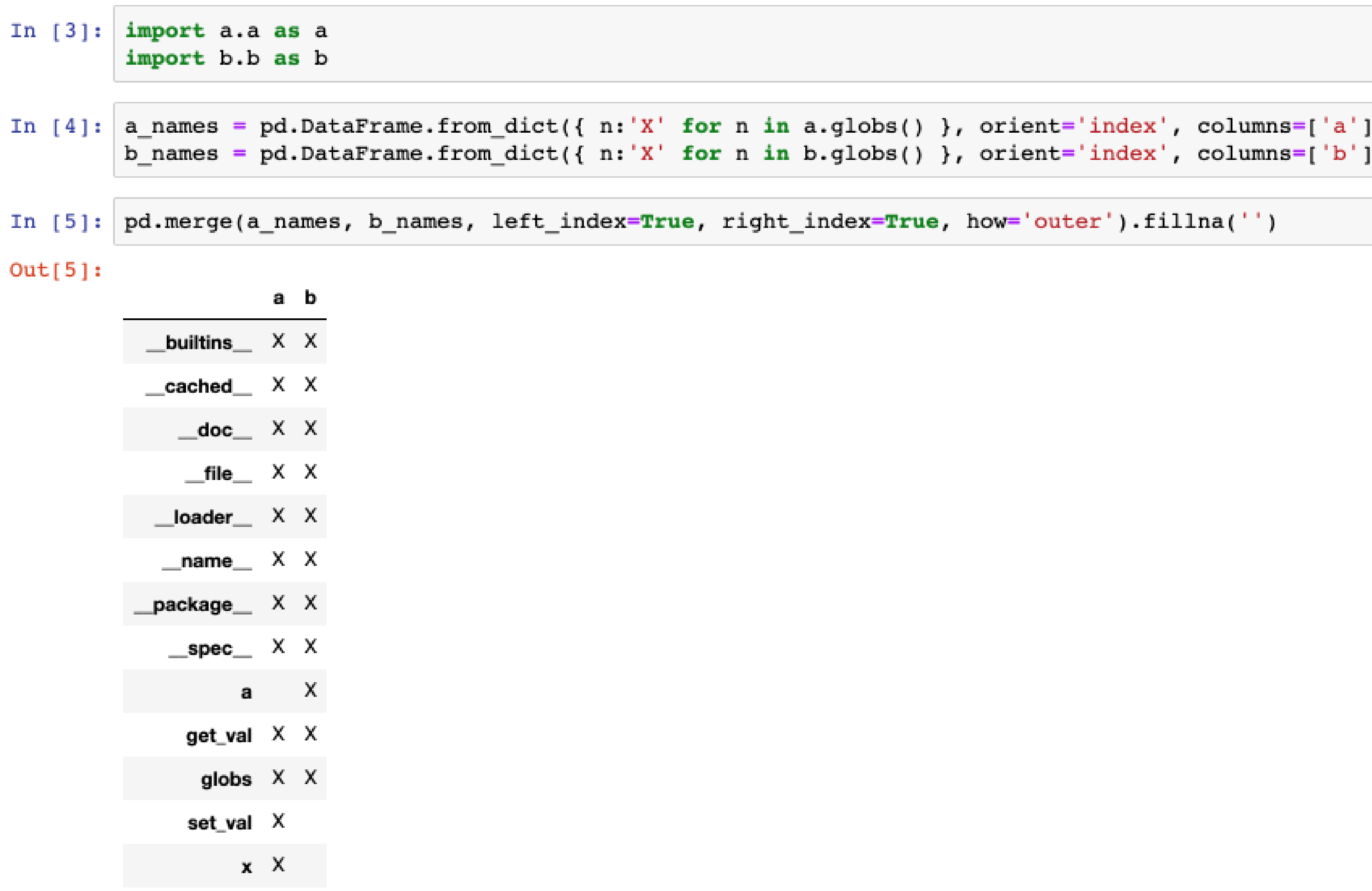
```
import a.a
def get_val(): return a.a.get_va
def globs(): return globals().ke
```

Module



```
$ python
Python 3.7.2 (default, Mar  4 2019, 14:29:09)
[Clang 10.0.0 (clang-1000.10.44.4)] on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>> import a.a as a
>>> import b.b as b
>>> a.get_val()
10
>>> b.get_val()
10
>>> a.set_val(334)
>>> b.get_val()
334
>>> 
```

Module



```
In [3]: import a.a as a
        import b.b as b
```

```
In [4]: a_names = pd.DataFrame.from_dict({ n:'X' for n in a.globs() }, orient='index', columns=['a']
        b_names = pd.DataFrame.from_dict({ n:'X' for n in b.globs() }, orient='index', columns=['b']
```

```
In [5]: pd.merge(a_names, b_names, left_index=True, right_index=True, how='outer').fillna('')
```

Out[5]:

	a	b
__builtins__	X	X
__cached__	X	X
__doc__	X	X
__file__	X	X
__loader__	X	X
__name__	X	X
__package__	X	X
__spec__	X	X
a		X
get_val	X	X
globs	X	X
set_val	X	
x	X	

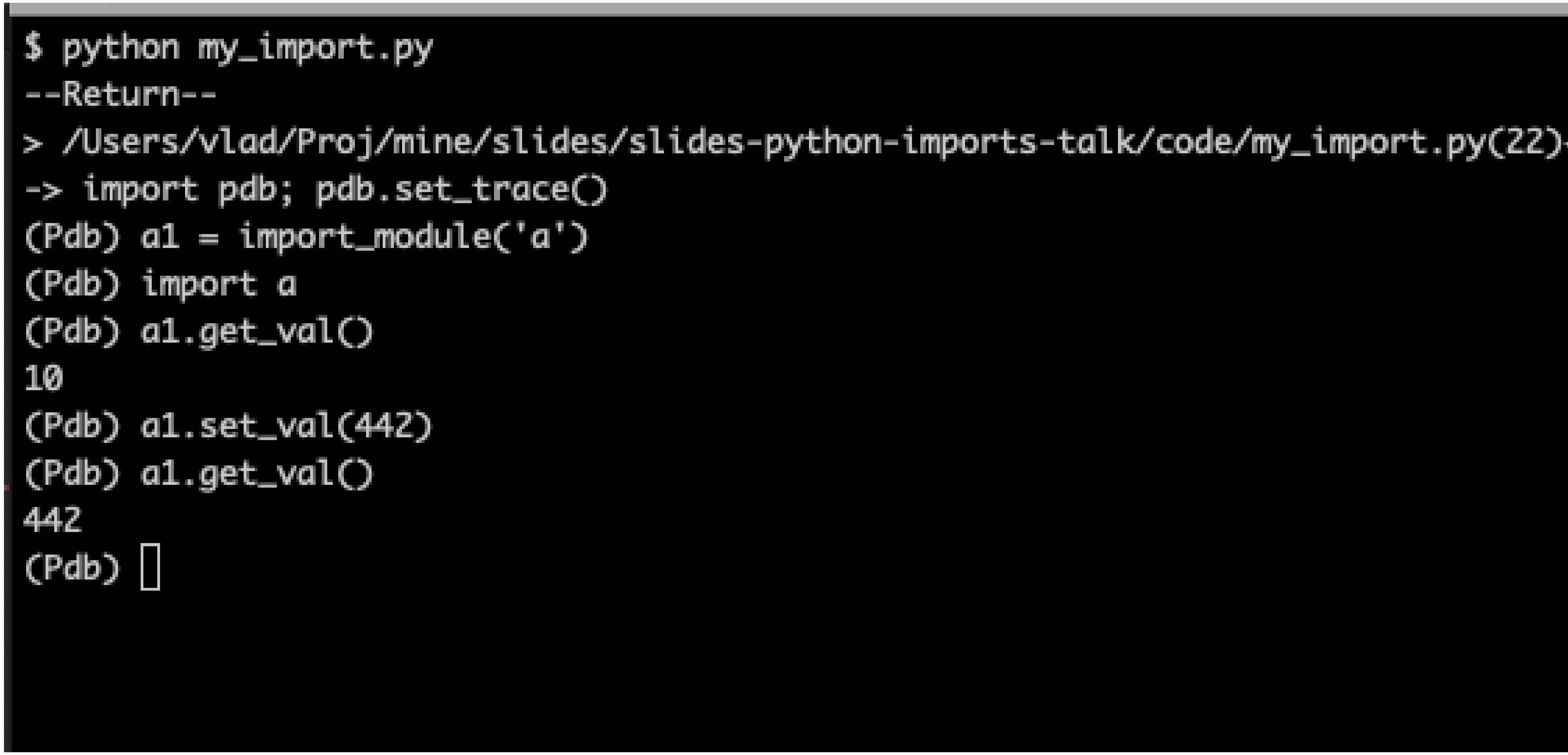
Module: Loading

- Real dumb
- Read the file
- Make namespace
- Compile
- Return result

Module: Loading

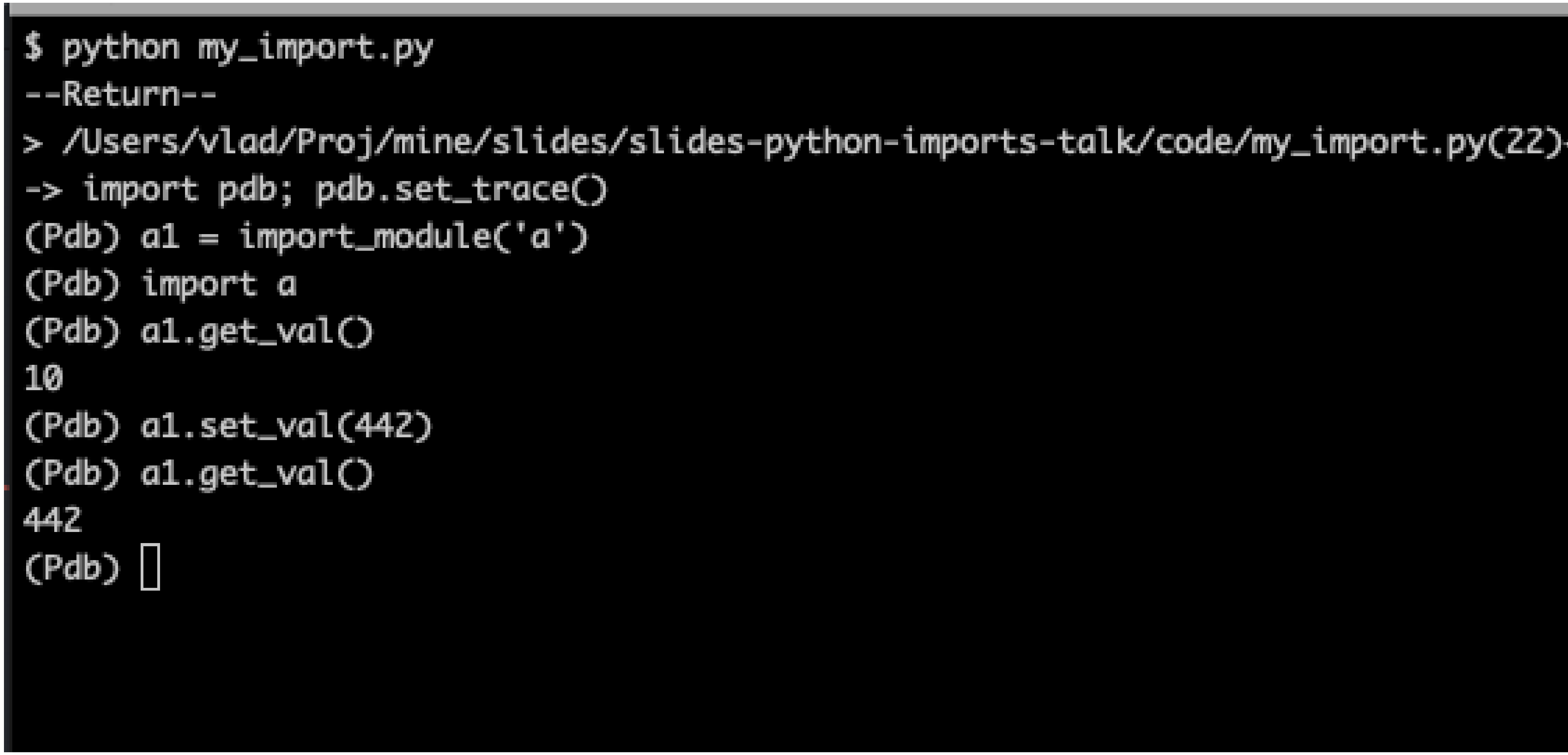
```python
def import_module(modname):
    if modname in sys.modules:
        return sys.modules[modname]
    source_path = f"share_data/{modname}/{modname}.py"
    with open(source_path, 'r') as f:
        code = f.read()
    mod = types.ModuleType(modname)
    code = compile(code, source_path, 'exec')
    sys.modules[modname] = mod
    exec(code, mod.__dict__)
    return sys.modules[modname]

import pdb; pdb.set_trace()
```

Module: Loading



```
$ python my_import.py
--Return--
> /Users/vlad/Proj/mine/slides/slides-python-imports-talk/code/my_import.py(22)
-> import pdb; pdb.set_trace()
(Pdb) a1 = import_module('a')
(Pdb) import a
(Pdb) a1.get_val()
10
(Pdb) a1.set_val(442)
(Pdb) a1.get_val()
442
(Pdb) 
```

Module: Loading



```
$ python my_import.py
--Return--
> /Users/vlad/Proj/mine/slides/slides-python-imports-talk/code/my_import.py(22)
-> import pdb; pdb.set_trace()
(Pdb) a1 = import_module('a')
(Pdb) import a
(Pdb) a1.get_val()
10
(Pdb) a1.set_val(442)
(Pdb) a1.get_val()
442
(Pdb) 
```

Module: Loading

- Real one is bit more complex
- Over 3000 lines[1]
- Essentially the same
- Real modules have `__init__.py` files

[1] `http://bit.ly/30qJ7T4`

Module loaded, now what?

Module: Import

- Import
  - Plain

```
import x
```

  - Relative

```
import .y
```

  - Select objects

```
from math import cos, sin
```

  - Everything

```
from y import *
```

__init__.py

__init__.py

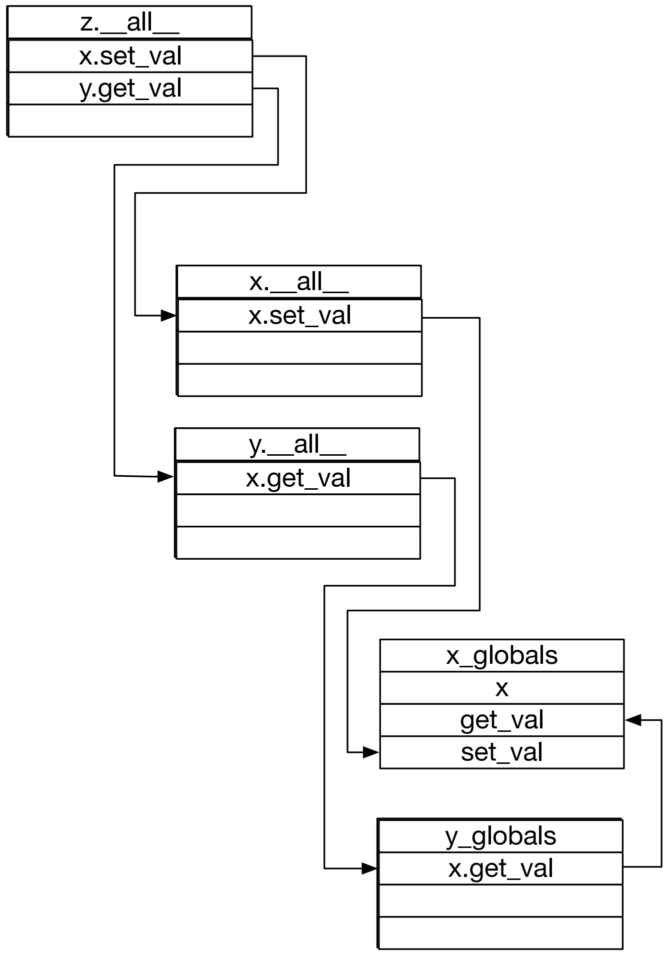
- Annoying
- Make directory into module
- Not really needed in Py3
- But better have them anyway

Module: __init__.py : what for?

- Mostly empty
- Package code (boto3)[2]
- Stitch things together
- @export decorator
- Lazy loading

[2] http://bit.ly/2L7aGuu

Module: __init__.py : stitch modules



z/x.py

```python
x = 10
def set_val(a):
    global x; x = a
def get_val():
    return x
__all__ = ('set_val',)
```

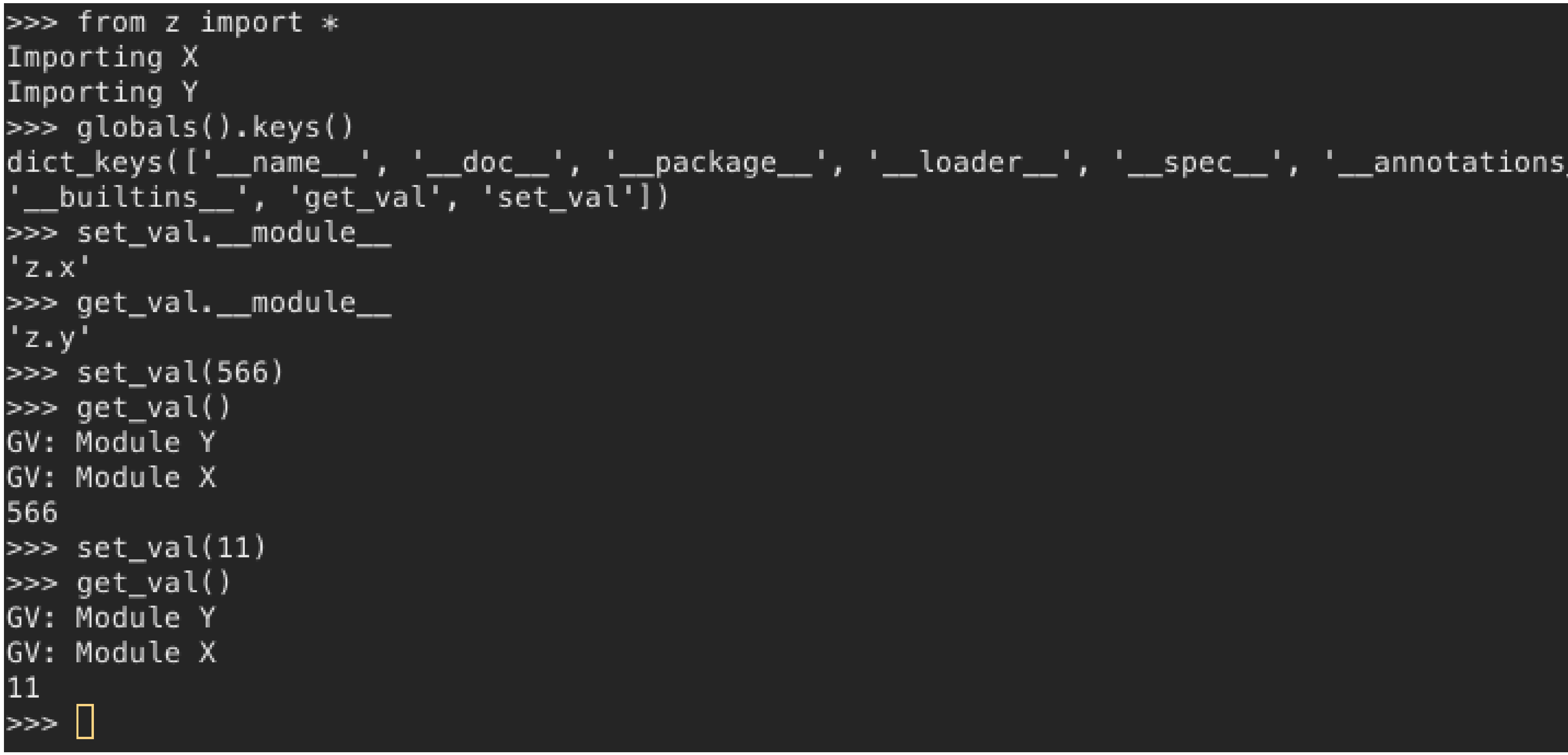
z/y.py

```python
import z.x
def get_val():
    return z.x.get_val()
__all__ = ('get_val',)
```

__init.py__

```python
from .x import *
from .y import *
__all__ = (y.__all__ + x.__all__)
```

Module: __init__.py : stitch modules



```
>>> from z import *
Importing X
Importing Y
>>> globals().keys()
dict_keys(['__name__', '__doc__', '__package__', '__loader__', '__spec__', '__annotations
'__builtins__', 'get_val', 'set_val'])
>>> set_val.__module__
'z.x'
>>> get_val.__module__
'z.y'
>>> set_val(566)
>>> get_val()
GV: Module Y
GV: Module X
566
>>> set_val(11)
>>> get_val()
GV: Module Y
GV: Module X
11
>>> 
```

What about scripts

__main__ module

- Have no module
- Auto wrapped into module named '__main__'
- Recognize this?

```
if __name__ == '__main__':
    your_code_here()
```

__main__ module

- Ways to run python code
  - As script

```
python script.py
```

  - As a module (file)

```
python -m script
```

__main__ module

- Ways to run python code
  - As a module (directory)

```
python -m dir
```

  - As a zip file

```
python -m zipfile -c hello.zip zip/*.py
python -m file.zip
```

__main__ module

- More fun with the zip file

```
python -m zipfile -c hello.zip zip/*.py
python hello.zip
```

- Self contained zip

```
python -m zipfile -c hello.zip zip/*.py
python hello.zip
cat zipstub hello.zip > hi
chmod +x hi
./hi
```

Real Module Loading

- Two step process
- Chain of searchers
- Search to get spec
- Load and link from spec

Real Module Loading

- Searchers : sys.path searcher
  - Most basic
  - Goes over the sys path and tries to resolve
  - Pays attention to .pth files and such
  - Only addresses normal dirs
- Other Searchers
  - Look in zips, eggs etc

Real Module Loading

- Custom searchers
  - You can write your own
- Auto install `http://bit.ly/2HmNMx0`

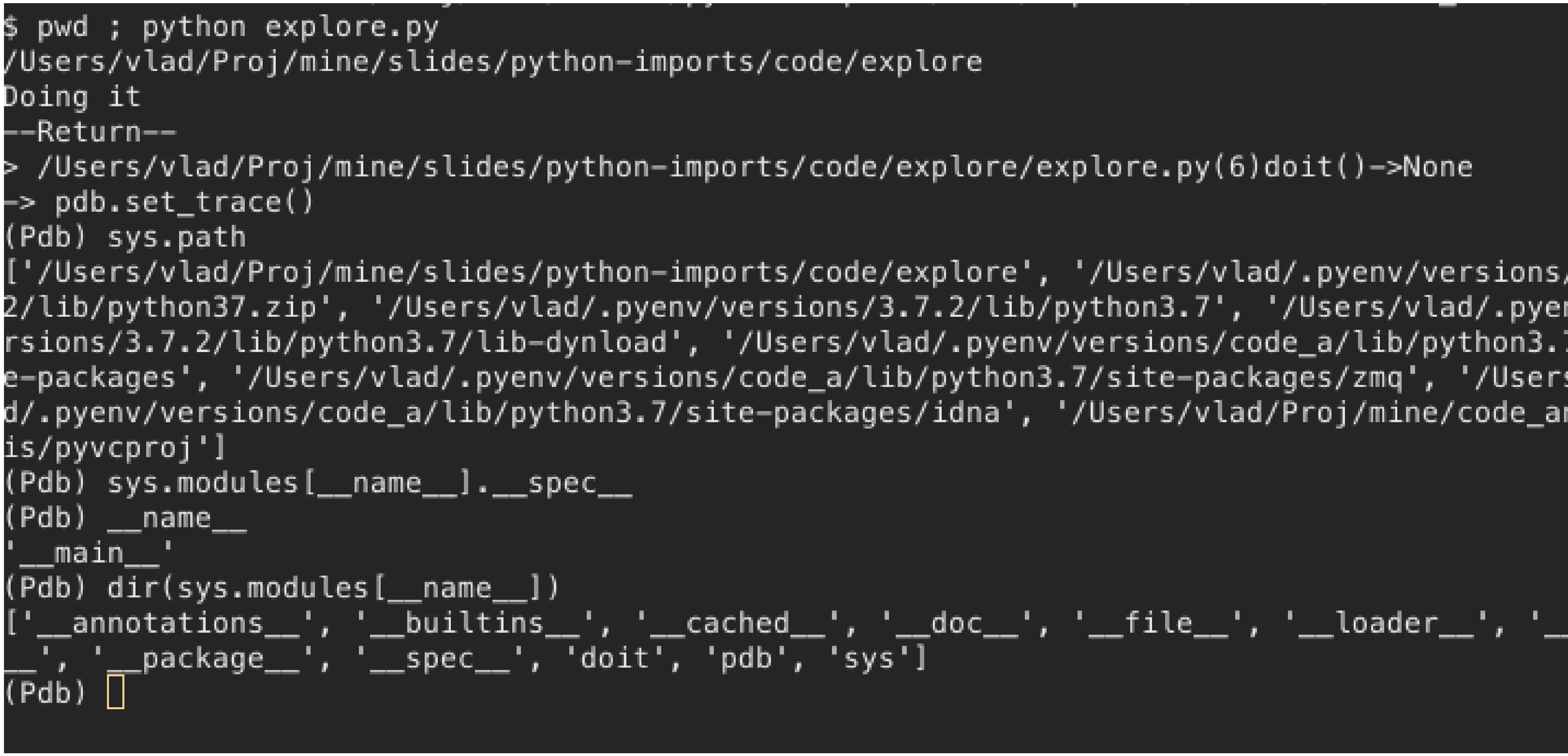
When things go wrong

When things go wrong

- Remember google search
- Common Problems
  - Search path
  - Missing init files
  - Incorrect directory
  - Running as module vs script

When things go wrong

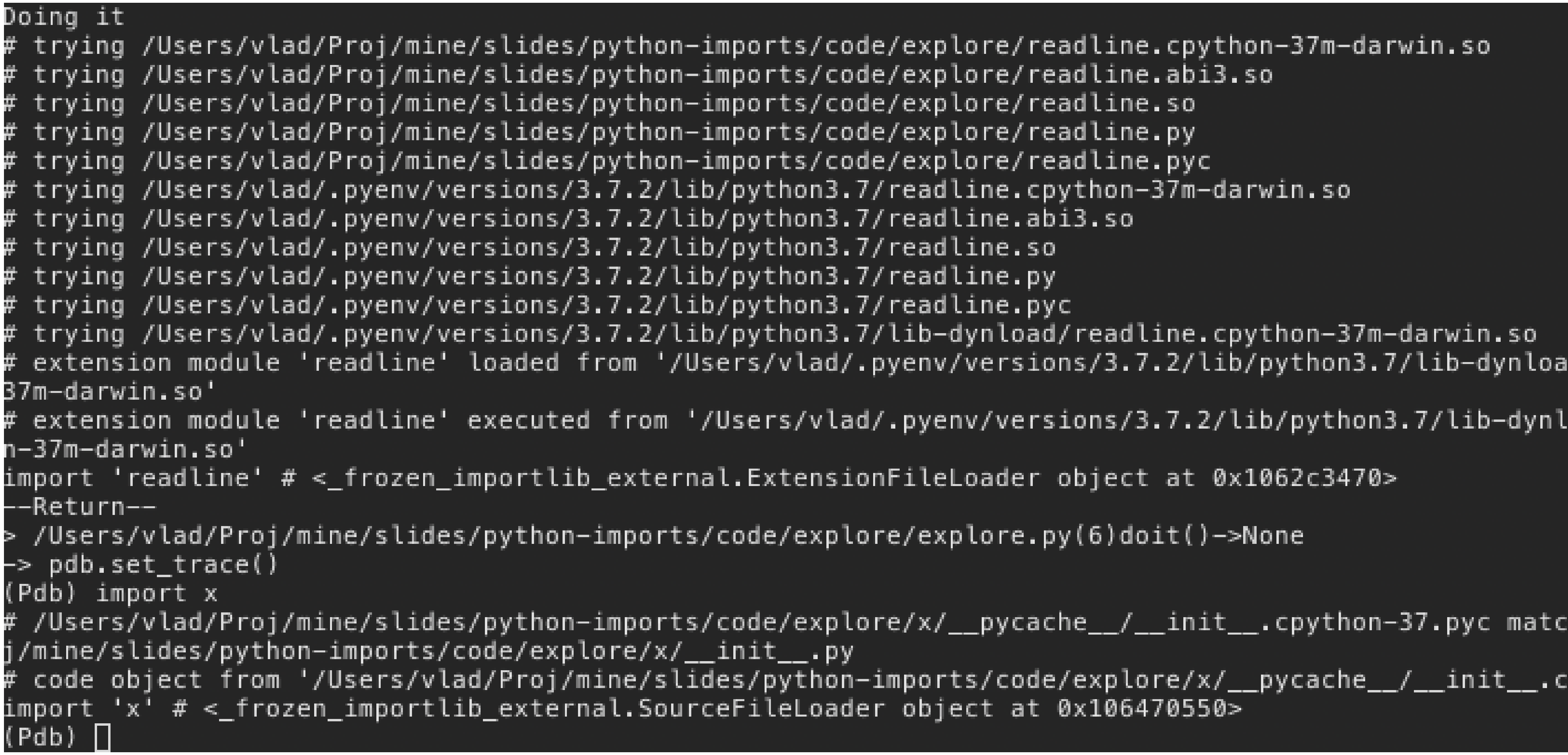
- See the context
  - sys.path
  - __spec__



```
$ pwd ; python explore.py
/Users/vlad/Proj/mine/slides/python-imports/code/explore
Doing it
--Return--
> /Users/vlad/Proj/mine/slides/python-imports/code/explore/explore.py(6)doit()->None
-> pdb.set_trace()
(Pdb) sys.path
['/Users/vlad/Proj/mine/slides/python-imports/code/explore', '/Users/vlad/.pyenv/versions
2/lib/python37.zip', '/Users/vlad/.pyenv/versions/3.7.2/lib/python3.7', '/Users/vlad/.pyen
rsions/3.7.2/lib/python3.7/lib-dynload', '/Users/vlad/.pyenv/versions/code_a/lib/python3.7
e-packages', '/Users/vlad/.pyenv/versions/code_a/lib/python3.7/site-packages/zmq', '/Users
d/.pyenv/versions/code_a/lib/python3.7/site-packages/idna', '/Users/vlad/Proj/mine/code_an
is/pyvcproj']
(Pdb) sys.modules[__name__].__spec__
(Pdb) __name__
'__main__'
(Pdb) dir(sys.modules[__name__])
['__annotations__', '__builtins__', '__cached__', '__doc__', '__file__', '__loader__', '__
__', '__package__', '__spec__', 'doit', 'pdb', 'sys']
(Pdb) 
```

When things go wrong

- python -vvv



```
Doing it
# trying /Users/vlad/Proj/mine/slides/python-imports/code/explore/readline.cpython-37m-darwin.so
# trying /Users/vlad/Proj/mine/slides/python-imports/code/explore/readline.abi3.so
# trying /Users/vlad/Proj/mine/slides/python-imports/code/explore/readline.so
# trying /Users/vlad/Proj/mine/slides/python-imports/code/explore/readline.py
# trying /Users/vlad/Proj/mine/slides/python-imports/code/explore/readline.pyc
# trying /Users/vlad/.pyenv/versions/3.7.2/lib/python3.7/readline.cpython-37m-darwin.so
# trying /Users/vlad/.pyenv/versions/3.7.2/lib/python3.7/readline.abi3.so
# trying /Users/vlad/.pyenv/versions/3.7.2/lib/python3.7/readline.so
# trying /Users/vlad/.pyenv/versions/3.7.2/lib/python3.7/readline.py
# trying /Users/vlad/.pyenv/versions/3.7.2/lib/python3.7/readline.pyc
# trying /Users/vlad/.pyenv/versions/3.7.2/lib/python3.7/lib-dynload/readline.cpython-37m-darwin.so
# extension module 'readline' loaded from '/Users/vlad/.pyenv/versions/3.7.2/lib/python3.7/lib-dynloa
37m-darwin.so'
# extension module 'readline' executed from '/Users/vlad/.pyenv/versions/3.7.2/lib/python3.7/lib-dynl
n-37m-darwin.so'
import 'readline' # <_frozen_importlib_external.ExtensionFileLoader object at 0x1062c3470>
--Return--
> /Users/vlad/Proj/mine/slides/python-imports/code/explore/explore.py(6)doit()->None
-> pdb.set_trace()
(Pdb) import x
# /Users/vlad/Proj/mine/slides/python-imports/code/explore/x/__pycache__/__init__.cpython-37.pyc matc
j/mine/slides/python-imports/code/explore/x/__init__.py
# code object from '/Users/vlad/Proj/mine/slides/python-imports/code/explore/x/__pycache__/__init__.c
import 'x' # <_frozen_importlib_external.SourceFileLoader object at 0x106470550>
(Pdb) 
```

Reloads

- Mostly loaded once
- Importing gives loaded module
- Importlib.reload
  - Mostly works
  - Unless they don't
- See here http://bit.ly/33TDpvc

What next

What's next

- Watch David's Talk
  - Make sure to take a break when they take it
- Read Eriks' book

????

Thank you