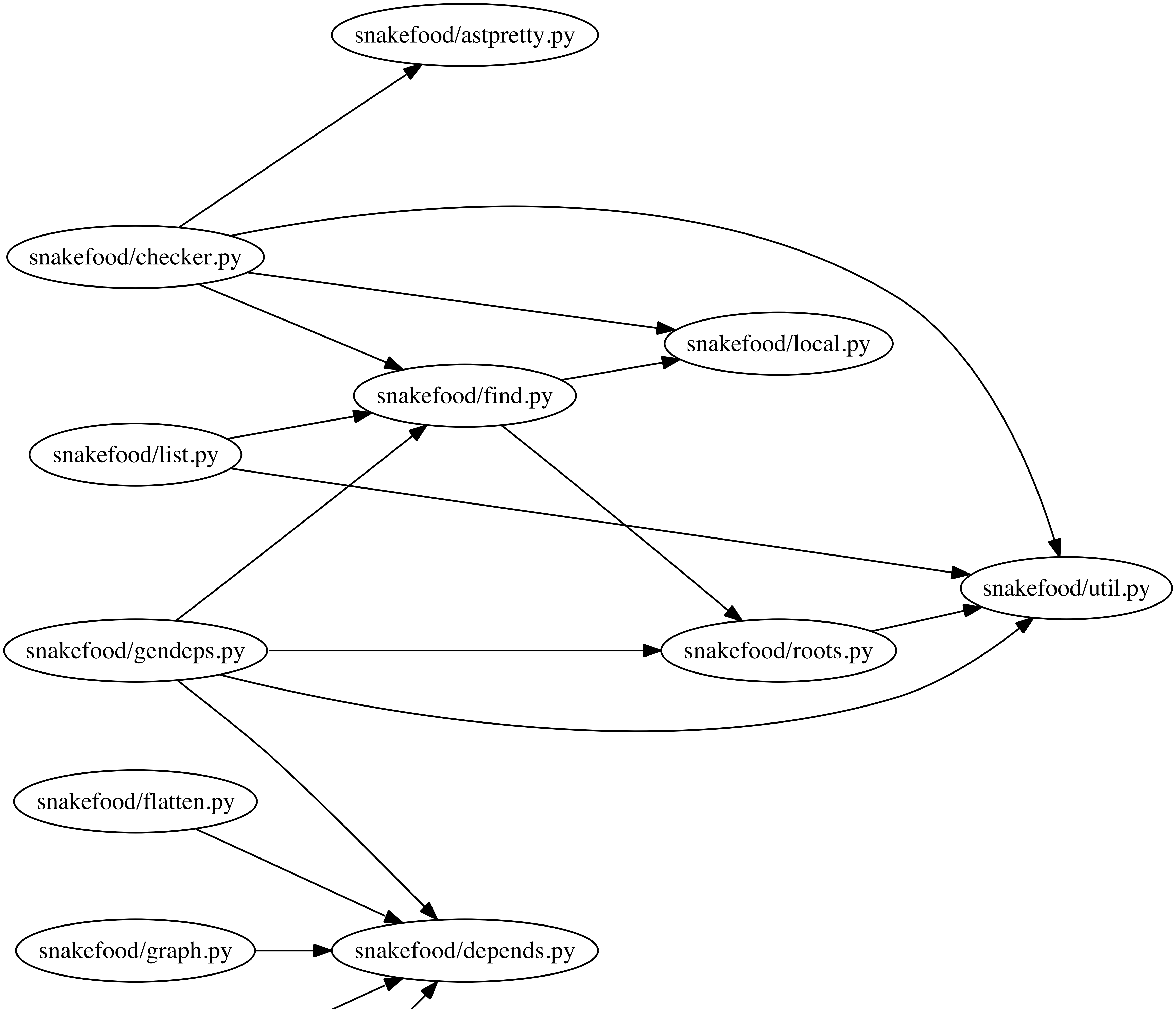
Python Dependencies with Snakefood

Martin Blais
http://furius.ca


snakefood/astpretty.py
snakefood/checker.py
snakefood/local.py
snakefood/find.py
snakefood/list.py
snakefood/util.py
snakefood/gendeps.py
snakefood/roots.py
snakefood/flatten.py
snakefood/graph.py
snakefood/depends.py

Introduction

- Generates dependency graphs
- Analysis and refactoring tool
- Zero configuration
- All pure Python code
- Works on many platforms

What is a dependency?

```
a.py
...
import b
...
```

What is a dependency?

a.py

```
...
import b
...
```

b.py

```
...
import c
...
```

What is a dependency?

a.py

```
...
import b
...
```

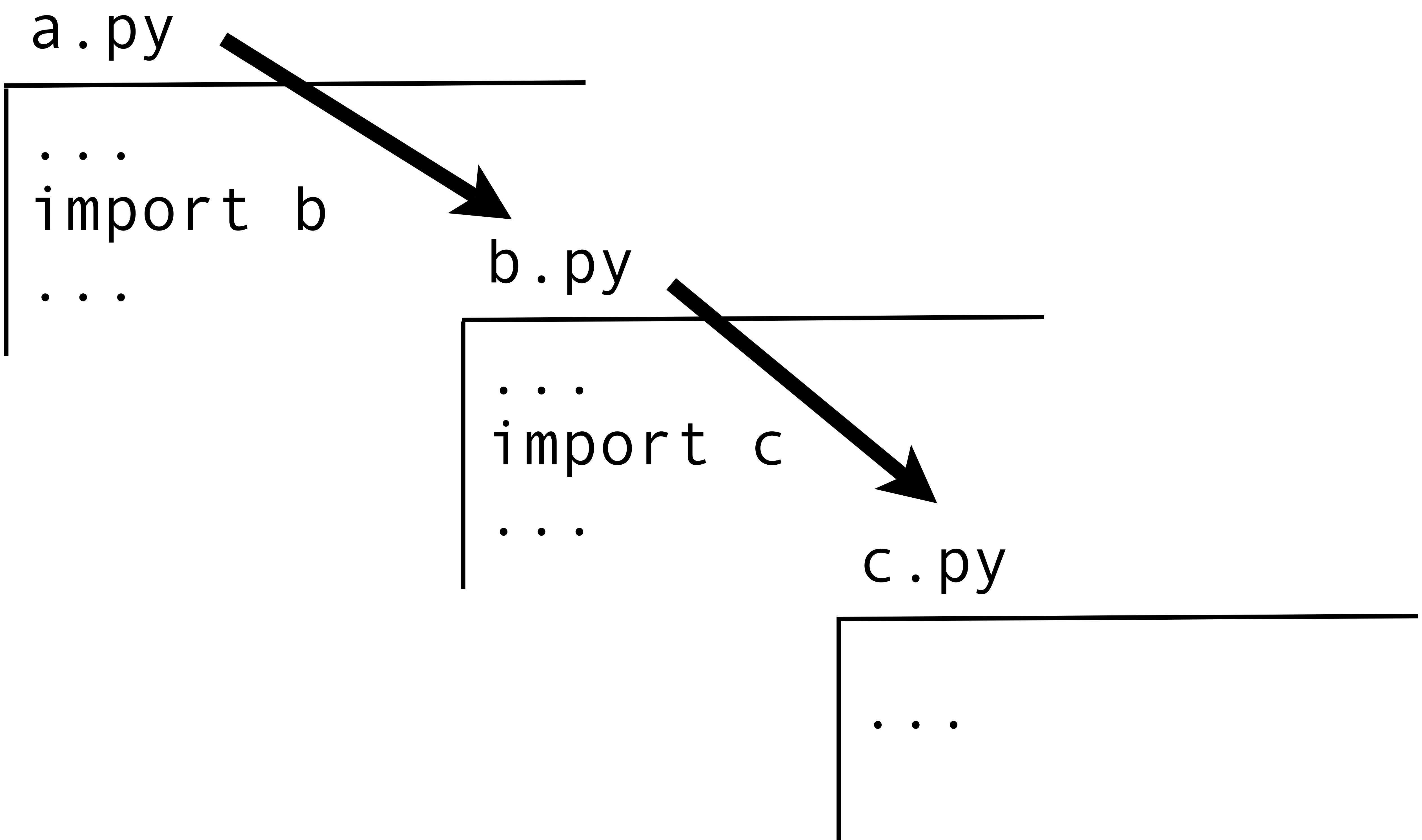
b.py

```
...
import c
...
```

c.py

```
...
```

What is a dependency?



```
...
import b
...
```

```
...
import c
...
```

```
...
```

Why?

Snakefood is used to answer questions

- Why does module A depend on module B?
- Why is there a circular dependency, and how can I remove it?
- How can I split a large package in parts?
- To enforce strict dependency relationships via a commit hook
- How can I produce cool looking charts of our codebase to try to look smart?

Bad Snakefood

snakefood.py:

```
# Import the module.
mod = __import__(userFilename)

# Analyse the dependencies using
# the module object.
get_dependencies(mod)
...
```

Bad Snakefood

snakefood.py:

```
# Import the module.
mod = __import__(userFilename)

# Analyse the dependencies using
# the module object.
get_dependencies(mod)
...
```

What’s wrong?

```
...
import SomeModule
...
```

Side-effects

```
...
import SomeModule
...
```

SomeModule.py

```
  ...
  # Connect to the database.
  conn = dbapi.connect(
    dbname="accounts.db",
    user="joe")
```

(and by the way... please avoid causing side-effects....)

Snakefood does not load the modules, does not catch all

...but it **ALWAYS** runs! (thanks to the **AST**), "Good enough"

What is the AST?

- “Abstract Syntax Tree”: a tree of nodes which represent the Python code, as parsed and understood by the interpreter itself
- Is include in the stdlib
- Does **NOT** require evaluating the code*

Analysis via AST

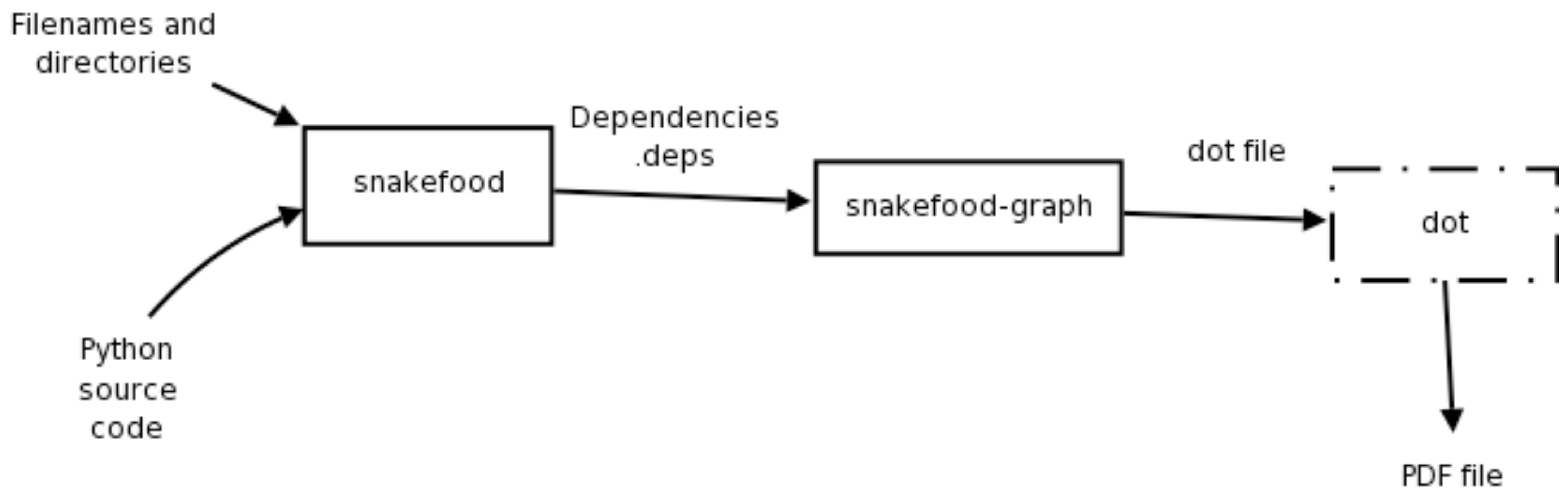
Works:

```
import Module

if __debug__:
    import Module

def foo():
    import Module
```

Fails:

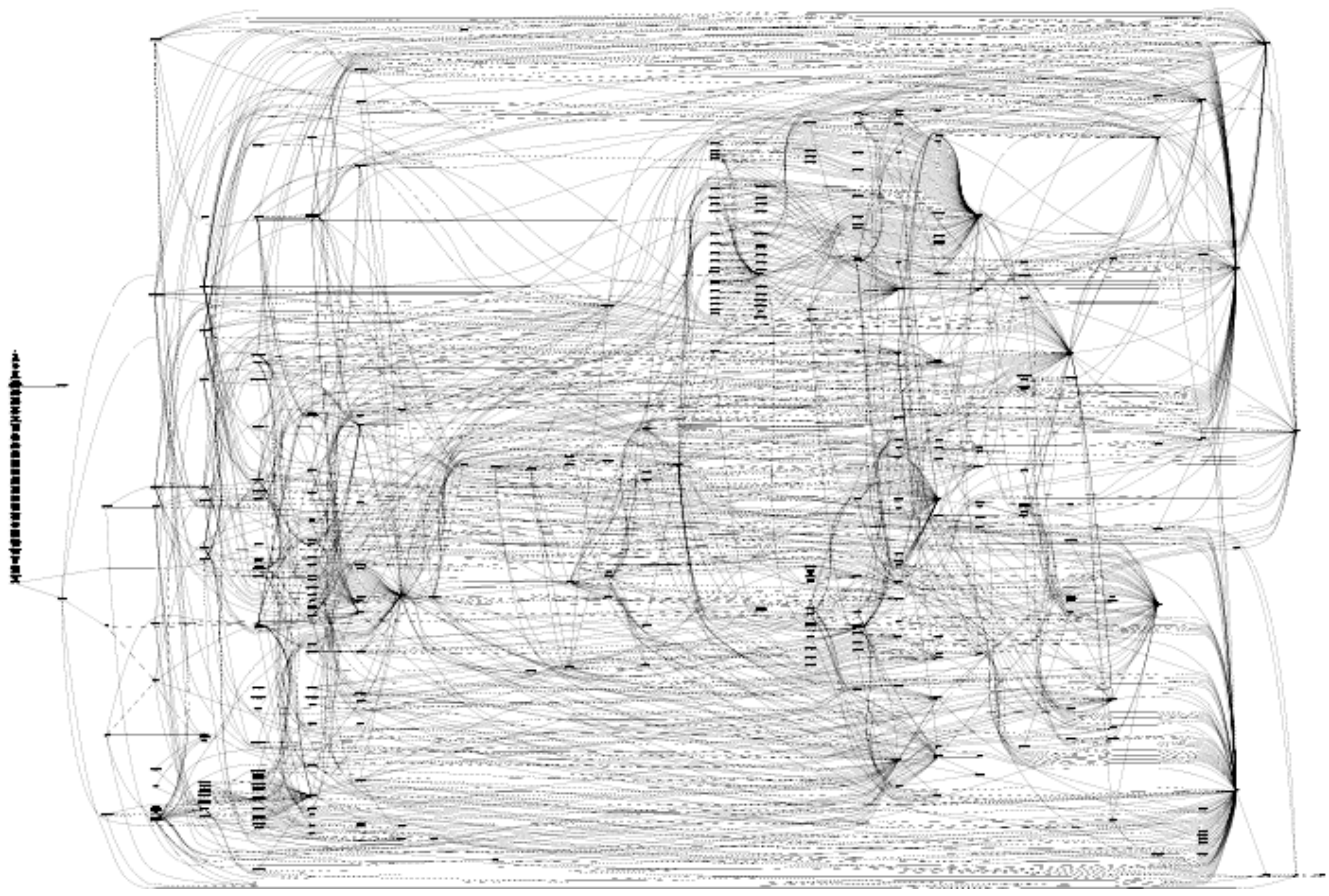
```
Module = __import__(“Module”)
```

The Food Chain



(Blitz demo)

Clustering



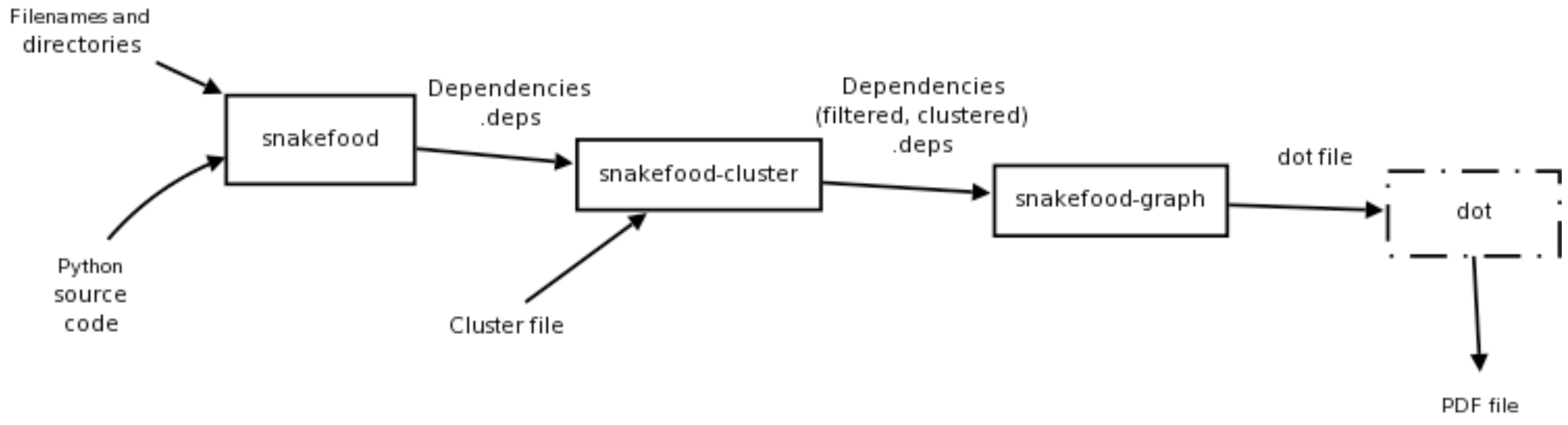
Clustering

django.clusters:

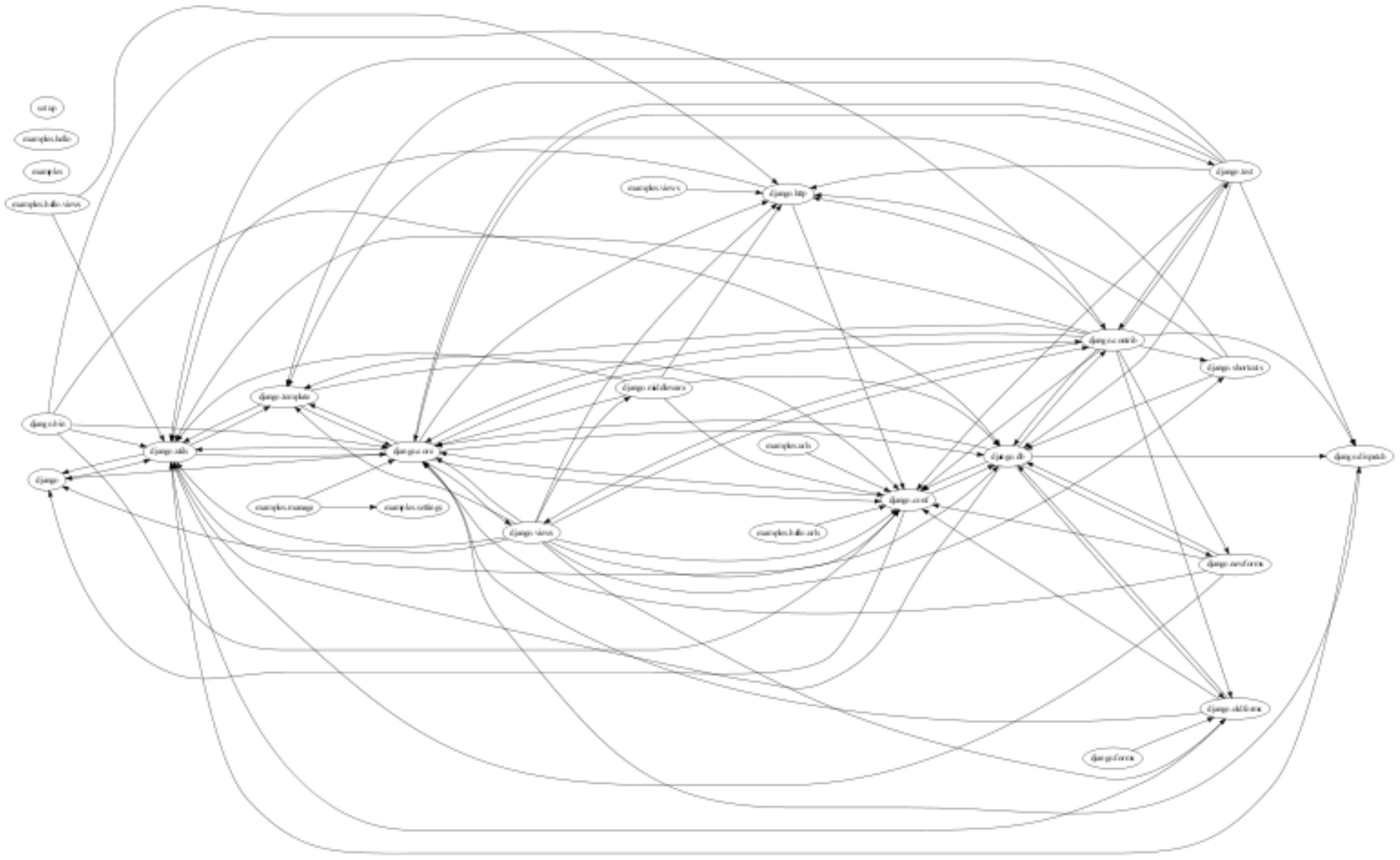
```
django/core
django/db
django/dispatch
django/forms
django/http
django/middleware
django/newforms
django/test
django/utils
django/views
...
```

The Food Chain

(with Clustering)



Clustering


setup
examples.hello
examples
examples.hello.views
django.bin
django.utils
django
django.template
django.core
examples.manage
examples.settings
django.views
django.middleware
examples.views
django.http
examples.urls
django.conf
examples.hello.urls
django.db
django.contrib
django.shortcuts
django.test
django.dispatch
django.newforms
django.oldforms
django.forms

(Blitz demo,
with clustering)

Zero Configuration

```
/project/
    lib/                          <--- ROOT
        booze/
            __init__.py
            scotch.py
            whiskey.py
            rhye.py
            test/                 <--- ROOT
                test_liquor.py
                test_perf.py
```

File Format

```
((ROOT, FILENAME), (ROOT, FILENAME))
((ROOT, FILENAME), (ROOT, FILENAME))
((ROOT, FILENAME), (ROOT, FILENAME))
...

Existence of a node:
((ROOT, FILENAME), (None, None))
```

```
(('/home/lib', 'booze/whiskey.py'), ('/home/lib', 'booze/scotch.py'))

(('/home/lib', 'booze/whiskey.py'), (None, None))
(('/home/lib', 'booze/scotch.py'), (None, None))
```

Parsing

```
for (root1, p1), (root2, p2) in
    map(eval, sys.stdin):
        ...
```

--follow and --internal

$$\begin{tikzcd}\text{Files specified by the user} \arrow[r] & \text{Files imported by code (dependencies)}\end{tikzcd}$$

Follow: recursively analyse files depended upon

Internal: filter out all the files that are not in the *roots* of files specified by the user

Codebases

4suite cgkit django docutils
enthought genshi gnosis
mailman matplotlib neoui
numpy pypy pythonweb pyutil
reportlab scipy sqlobject stdlib
twisted webstack zope

sfood-checker

```
import sys, os, re, StringIO, \
       datetime, math, urllib, \
       too_many_imports...
```

```
$ sfood-checker FILE.py
FILE.py:11:9: Unused import ‘os’
FILE.py:12:9: Unused import ‘datetime’
FILE.py:12:16: Unused import ‘math’
```

(Results)

Links

Full documentation at:

http://furius.ca/snakefood/

Questions?