Trick: importing from a path

- Inline code:

```
import importlib.machinery
import sys

spec = importlib.machinery.PathFinder.find_spec('foo', ['/tmp'])
foo = importlib.util.module_from_spec(spec)  # make it a module
spec.loader.exec_module(fff)  # let the module run
sys.modules['foo'] = foo  # make future imports sane
```

- `importlib.util.spec_from_file_location` also works
  - It requires knowing packages from files (__init__.py)
- This is different in Python2.x

Trick: importing from a path

- Helper function:

```
def import_path(path, override_name=None):
    path = os.path.realpath(path)
    name = os.path.splitext(os.path.basename(path))[0]
    basedir = os.path.dirname(path)

    if (override_name or name) in sys.modules:
        return sys.modules[override_name or name]

    spec = importlib.machinery.PathFinder.find_spec(name, [basedir])
    if not spec:
        raise ImportError(
            "No module named '%s' found in: %s" % (name, basedir))
    module = importlib.util.module_from_spec(spec)
    spec.loader.exec_module(module)  # run module
    sys.modules[override_name or name] = module  # make imports sane
```