Reproducible Developement Environments

```
module = Module(
    code="ELEE1147",
    name="Programming for Engineers",
    credits=15,
    module_leader="Seb Blair BEng(H) PGCAP MIET MIHEEM FHEA"
)
```

UNIVERSITY OF GREENWICH

Download as a PDF

Reproducibility definition

A setup where the software environment, including all dependencies, configurations, and tools, can be consistently recreated across different machines and over time, ensuring that the code runs identically regardless of where or when it is executed.

TLDR

A reproducible development environment is a setup where the software environment can be consistently recreated across different machines and over time.

Virtual Environments with pip

- **What Are They?**
  - Isolated environments for Python projects.
- **Common Tools:**
  - **virtualenv:** Creates separate directories with their own libraries.
  - **pipenv:** Combines package management with virtual environments.
- **Benefits:**
  - Avoid package version conflicts.
  - Customize the PATH variable to prioritize local libraries.

Virutal Environment

```
❯ py -m pip list
Package  Version
-------- -------
pip      24.3.1
psutil   7.0.0
toml     0.10.2
watchdog 6.0.0
```

Host Environment

```
❯ py -m pip list
Package            Version
------------------ ----------
alabaster          1.0.0
babel              2.16.0
certifi            2024.12.14
charset-normalizer 3.4.1
colorama           0.4.6
docutils           0.21.2
idna               3.10
imagesize          1.4.1
Jinja2             3.1.5
MarkupSafe         3.0.2
packaging          24.2
pip                24.3.1
Pygments           2.19.1
requests           2.32.3
snowballstemmer    2.2.0
terminus           0.1.2
urllib3            2.3.0
```

Package Management

A package manager is a program that manages (installs, upgrades, removes) a given target software and the target’s software dependencies.

Features:

- They install packages by compiling the packages from source or downloading binary files
- They attempt to solve dependency
- They operate on a local or system-wide environment

Package Manager Types

- **Binary package managers**:
  - install software by downloading pre-compiled files and placing them in certain paths.

- **From-source package managers**:
  - install software primarily by downloading source code, compiling it, and placing it in certain paths.
  - However, from-source package managers may optionally use a binary cache, where they can download a pre-compiled binary, a fallback!

Package Managers

TABLE II: Dependency solving feature matrix for state-of-the-art package managers.

Package manager	Version scheme	Solver	Distribution granularity		Version locking	Qualif.	Dependency operators range				Range modifiers		Resolution process			Approximate solutions	
							gt/lt	and	or	not	flex patch	flex minor	correctness	completeness	user prefs	missing deps	conflict
Go (dep)	git tags	ad hoc	github	branch	yes	no	no	no	no	no	no	no	yes	yes	no	error	error
npm	semver	ad hoc	archive	package	yes	no	yes	yes	yes	no	yes	yes	?	?	no	error	keep both
Packagist	git tags	ad hoc	github	branch	yes	no	yes	yes	yes	no	yes	no	yes	?	?	?	error
opam	debian	CUDF (any)	git	package	work-around	yes	yes	yes	yes	yes	no	no	yes	yes	yes	error	error
PyPI / pip	pep-440	ad hoc	archive	package	yes	conda	yes	yes	no	yes	yes	yes	yes	yes	no	error	error
Nuget	semver	ad hoc	archive	package	yes	no	yes	yes	no	no	no	no	yes	yes	no	error	nearest wins
Paket	semver	ad hoc	archive, github	package, branch	yes	no	yes	yes	no	no	yes	no	yes	yes	no	error	error
Maven	semver	ad hoc	archive	package	no	yes	yes	yes	yes	yes	no	no	yes	yes	with plug-ins	latest	nearest wins
RubyGems	semver	ad hoc	archive	package	yes	bundler	yes	yes	no	no	yes	no	?	?	?	error	error
Cargo	semver	ad hoc	archive, git	package, branch	no	yes	yes	yes	no	no	yes	yes	yes	yes	no	latest	name mangling
CPAN	strings	ad hoc	archive	package	no	yes	yes	yes	yes	yes	no	no	no	no	no	error	error
Bower	semver	ad hoc	git	package	?	?	yes	yes	yes	no	yes	yes	yes	yes	no	error	use resolutions
Clojars	semver	ad hoc	archive	package	?	?	yes	yes	yes	yes	no	no	yes	yes	error	error	error
CRAN	debian	ad hoc	archive, git	package	?	yes	yes	yes	yes	yes	no	no	no	no	no	error	error
Hackage / cabal	semver	?	archive	package	?	no	yes	yes	yes	yes	yes	no	?	no	no	error	error
Debian (apt)	debian	CUDF (any)	package	package	pinning	yes	yes	yes	yes	yes	no	no	yes	yes	yes	error	error
RedHat (dnf)	dnf	libzypp	archive	package	?	yes	yes	yes	yes	yes	yes	yes	yes	yes	?	error	error
Eclipse P2	semver	sat4j	archive	package	?	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	error	error

Hyrums Law

With a sufficient number of users of an API, it does not matter what you promise in the contract: all observable behaviors of your system will be depended on by somebody.

https://www.hyrumslaw.com/

Sermantic Versioning

- A version number MAJOR.MINOR.PATCH:

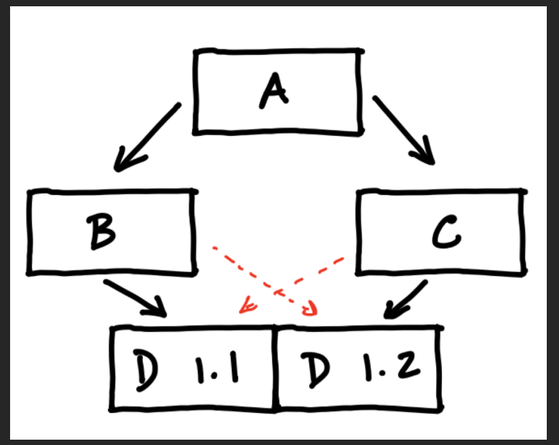

```
Django 1.3
Django 1.2.5
Django 1.2.4
Django 1.2.3
Django 1.2.2
Django 1.2.1
```

Dependency Constraints

A dependency constraint occurs when operating some versions of A requires operating B (operate usually means compile or run). Since

Diamond Dependency

- Library A depends on libraries B and C
- Both B and C depend on D
- But B requires D version 1.1,
- and C requires D version 1.2.



Automate Reproducibility

- installer scripts are great for this
- Do not rely on the human... they will make mistakes
- Most installers look like `bash install.sh`
- `curl https://github.com/<user>/<repo>/install.sh | bash`

Build our environment

Fast tracking;

- let's start off with using `curl`, pipe (`|`) and `bash`:

```
curl https://raw.githubusercontent.com/uniofgreenwich/VirtualEnv/master/install.sh | bash
```

- Pipe `|` is the key next to the `z` key, use the `shift` at the same time

Building our environment

Go to the Markdown Book start:

Virtual Environments