

# La reproductibilité des calculs en Python

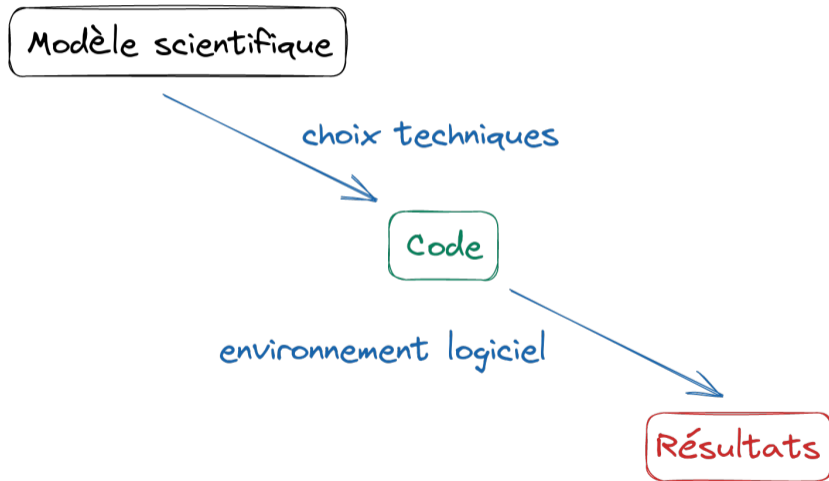
Konrad Hinsen

Centre de Biophysique Moléculaire, Orléans, France  
Synchrotron SOLEIL, Saint Aubin, France

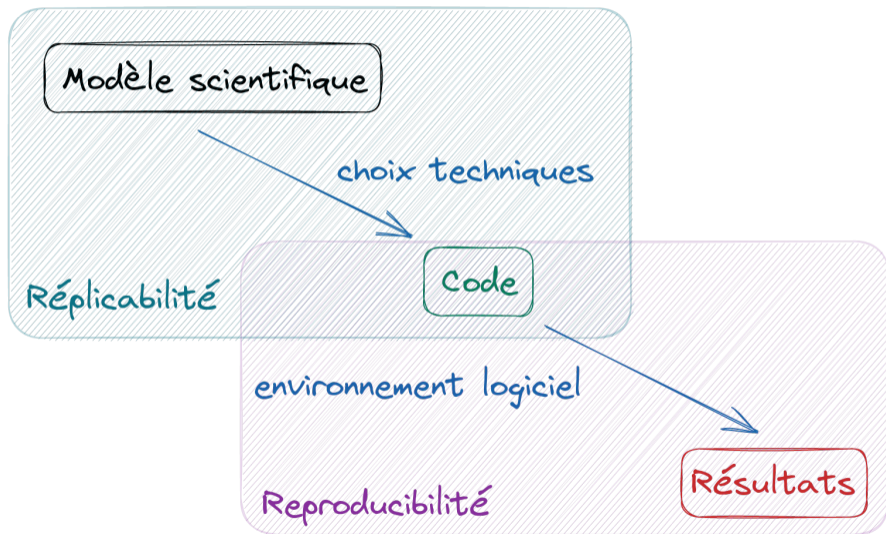
11 avril 2023

- 1 Principes de la reproductibilité computationnelle
- 2 Application à l'univers Python
- 3 Conseils pratiques

# Schéma du calcul scientifique



# Schéma du calcul scientifique



## Reproductibilité

Obtenir des résultats **identiques** (octet par octet)  
sur une **autre** machine,  
avec le **même** code et les **mêmes** données

Documentation du calcul : code, données, environnement logiciel

## Reproductibilité

Obtenir des résultats **identiques** (octet par octet)  
sur une **autre** machine,  
avec le **même** code et les **mêmes** données

Documentation du calcul : code, données, environnement logiciel

## Répliquabilité

Obtenir des résultats **équivalents**  
avec une **autre** implémentation  
du **même** modèle scientifique

Documentation du modèle scientifique

Validation des choix techniques

# What's a computation?

## Input

```
100111100001001100110101101100
001010011101010111110001001101
010111101100011110111011110001
001100001110111000100100000111
110101100111001110100000100110
110111100111000011111101101111
111001001011110001100110000101
011100001000010001011110000010
110101110011101111001010100111
111000101110011001101101001001
011001010100101011000001001100
11010011100101111100001011101
0111101111000111011110101101
000001110110011001010101011100
100010110001100000111001100010
000000111011100100100101010111
000010000001100001000010110110
101111101111000111100101110101
100101010100001001110100010001
011110011010100101111011110101
100011000110110001011101100110
110100000100000011011000001101
100000011100100111101101011011
010110010001000101110111001010
```



## Output

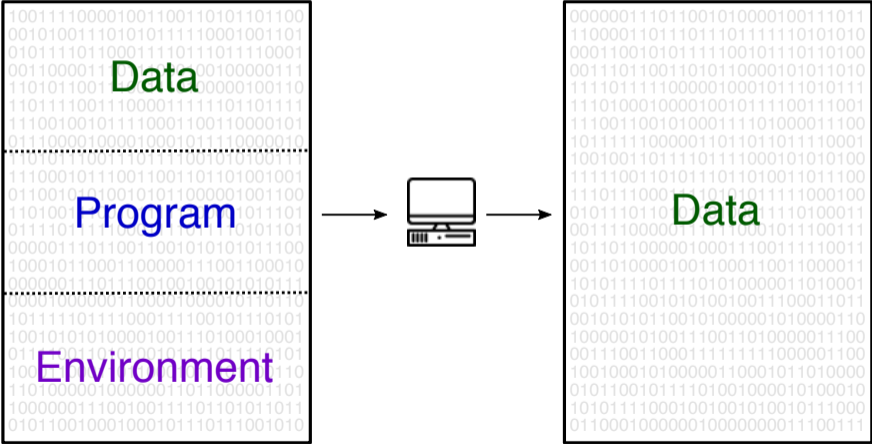
```
000000111011001010000100111011
110000110111011101111110101010
000110010101111100101110110100
001110110011010110000101011010
111101111100000100010111010111
111010001000010010111100111001
111001100101000111101000011100
101111110000011011011011110001
100100110111101111000101010100
111110011010111011010011011100
111011100011110101011111000100
010111011010100100011110100011
00111100000111110001011100111
101101100000100011100111110011
001101000010011000110011000011
101011110111101010000011010001
010111100101010010011100011011
001010101100101000001010000110
100000101001110011010000011100
001110011000111111111000001100
100100010100000110001011010000
010110010111101001000010100010
101011110001001001010010111000
01100010000001000000011100111
```

Computer by Creative Stall | from the Noun Project

# What's a computation?

Input

Output



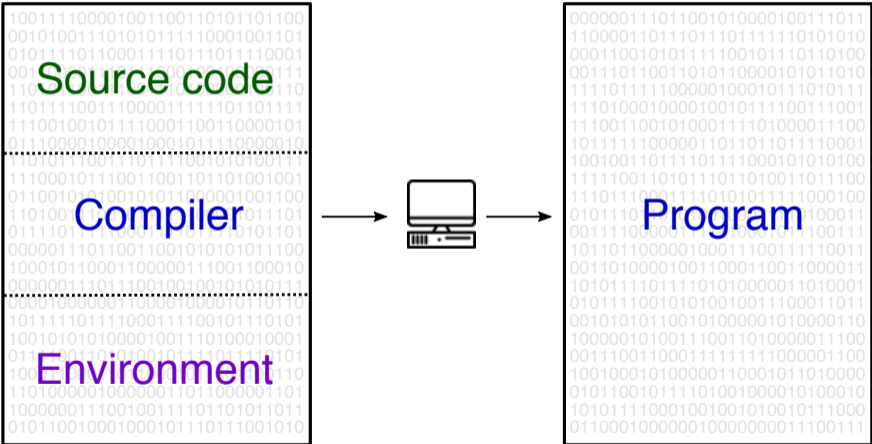
Computer by Creative Stall | from the Noun Project



# Compilation is computation

Input

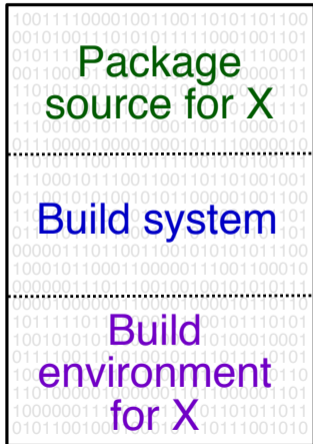
Output



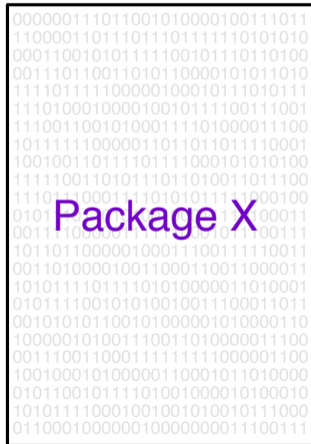
Computer by Creative Stall | from the Noun Project

# Package building is computation

Input



Output



Computer by Creative Stall from the Noun Project

## Computation is deterministic !

- Same inputs  $\longrightarrow$  same output

## Computation is deterministic !

- Same inputs  $\longrightarrow$  same output
- Same inputs = same data, same code, same environment

## Computation is deterministic !

- Same inputs  $\rightarrow$  same output
- Same inputs = same data, same code, same environment
- **Irreproducible output  $\rightarrow$  something has changed !**

## Computation is deterministic !

- Same inputs  $\rightarrow$  same output
- Same inputs = same data, same code, same environment
- **Irreproducible output**  $\rightarrow$  **something has changed !**
- Most often it's the environment.

# Deployment vs. archiving

## Deployment

- **Goal** : facilitate the use of software tools
- Prepare for distribution
- Install
- Update
- *Newer is better*

# Deployment vs. archiving

## Deployment

- **Goal** : facilitate the use of software tools
- Prepare for distribution
- Install
- Update
- *Newer is better*

## Archiving

- **Goal** : preserve a precise software stack for reproducibility
- Make a complete snapshot
- Re-deploy a snapshot
- *Newer is different*



# Package managers

## Designed for deployment, not archiving

apt, rpm, homebrew, pip, ...

all but two : **Guix** and **Nix**

and a half : **conda**

# Package managers

## Designed for deployment, not archiving

apt, rpm, homebrew, pip, ...

all but two : **Guix** and **Nix**

and a half : **conda**

- Compatibility checks by version number and  $\geq$  relations

## Designed for deployment, not archiving

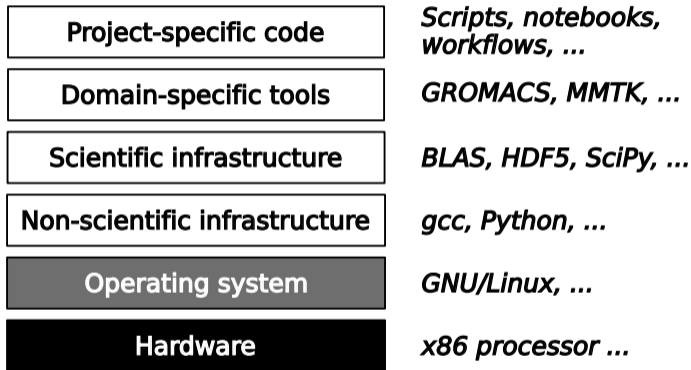
apt, rpm, homebrew, pip, ...

all but two : **Guix** and **Nix**

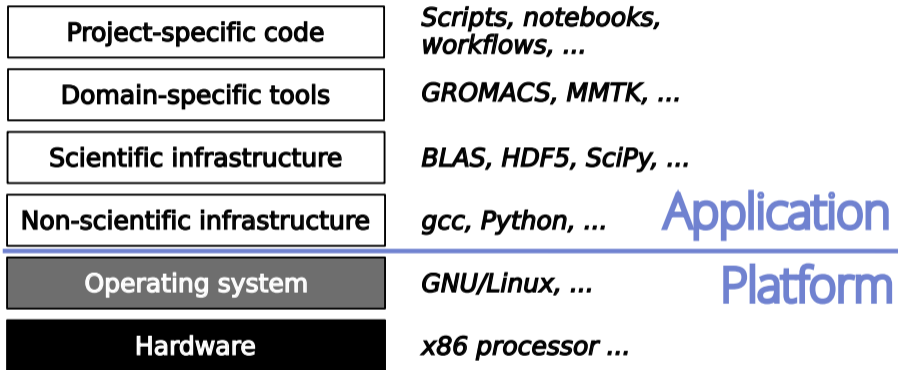
and a half : **conda**

- Compatibility checks by version number and  $\geq$  relations
- The build environment is not recorded...
- ... and is thus not reproducible ...
- ... meaning packages are not reproducible either !

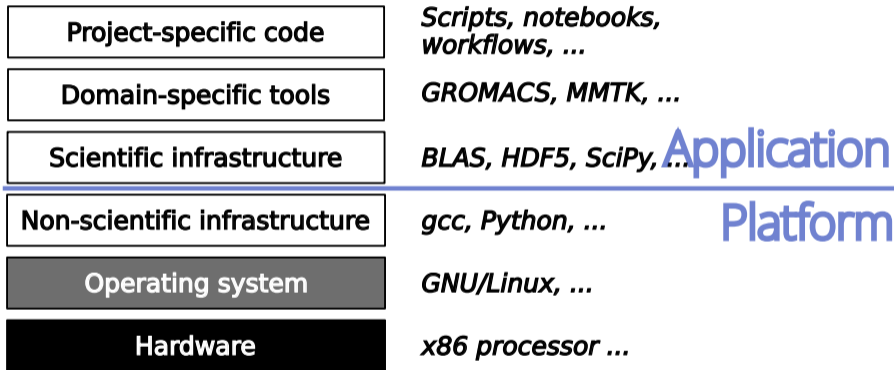
# A typical scientific software stack



# Foundation : the platform



# Foundation : the platform



Python-specific package managers  
**cannot** ensure reproducibility !

Python-specific package managers  
**cannot** ensure reproducibility !

The only OS that supports  
reproducibility is **Linux** !



# The gold standard for reproducibility

## **Platform :**

- Hardware : x86 or ARM processor
- Operating system : Linux kernel plus POSIX-compatible file system
- Package manager : Guix or Nix

# The gold standard for reproducibility

## **Platform :**

- Hardware : x86 or ARM processor
- Operating system : Linux kernel plus POSIX-compatible file system
- Package manager : Guix or Nix

## **Software environment :**

- Composed of Guix/Nix packages
- Defined by
  - A list of channels, with a commit ID for each channel
  - A list of package specifications

# The gold standard for reproducibility

## **Platform :**

- Hardware : x86 or ARM processor
- Operating system : Linux kernel plus POSIX-compatible file system
- Package manager : Guix or Nix

## **Software environment :**

- Composed of Guix/Nix packages
- Defined by
  - A list of channels, with a commit ID for each channel
  - A list of package specifications

## **Scripts, notebooks, workflows :**

- Run in Guix/Nix containers
- Stored in a version-controlled repository
- Repository can also contain pure Python dependencies as source code

# Good enough

## Platform :

- Hardware : x86 or ARM processor in a virtual machine
- Operating system and package manager : Debian snapshot

## **Platform :**

- Hardware : x86 or ARM processor in a virtual machine
- Operating system and package manager : Debian snapshot

## **Software environment :**

- Composed of Debian packages
- Defined by
  - The timestamp of the snapshot
  - A list of package names

# Good enough

## **Platform :**

- Hardware : x86 or ARM processor in a virtual machine
- Operating system and package manager : Debian snapshot

## **Software environment :**

- Composed of Debian packages
- Defined by
  - The timestamp of the snapshot
  - A list of package names

## **Scripts, notebooks, workflows :**

- Stored in a version-controlled repository
- Repository can also contain pure Python dependencies as source code

# Acceptable for short-term reproducibility

## Platform :

- Conda on top of a supported OS

# Acceptable for short-term reproducibility

## **Platform :**

- Conda on top of a supported OS

## **Software environment :**

- Composed of Conda packages
- Defined by
  - The list of channels, ordered by priority
  - A list of package with versions and hashes



## **Platform :**

- Conda on top of a supported OS

## **Software environment :**

- Composed of Conda packages
- Defined by
  - The list of channels, ordered by priority
  - A list of package with versions and hashes

## **Scripts, notebooks, workflows :**

- Stored in a version-controlled repository
- Repository can also contain pure Python dependencies as source code

# Conda vs. Guix/Nix

## Conda

- Linux, Windows, macOS

## Guix

- Linux only

# Conda vs. Guix/Nix

## Conda

- Linux, Windows, macOS
- one environment level

## Guix

- Linux only
- recursive environments

A Conda environment *with hashes* is either reproducible or fails.

## Conda

- Linux, Windows, macOS
- one environment level
- packages *not* reproducible

## Guix

- Linux only
- recursive environments
- packages reproducible

A Conda environment *with hashes* is either reproducible or fails.

A Conda environment *without hashes* is cross-platform but not reproducible (and may fail).

## **Designed for deployment, not archiving**

- Large binary files (“images”)
- In general not reproducible
- Not inspectable
- Not modifiable

Useful complement for an environment definition, but not a substitute !

- **Don't confuse reproducibility with deployment for reuse**
- Don't use Python-only package managers
- Don't use packages that you cannot build yourself from source
- Don't refer to “current” or “latest” versions
- Don't use container images without a build recipe (Dockerfile)