



This lecture will be recorded



C O M P A S

**064-0026-00L: COMPAS II**

Introduction to Computational Methods for Digital  
Fabrication in Architecture

```
mesh.mesh.vertices[...]  
if callback:  
    if not callable(callback):  
        raise Exception('Callback is not callable.')
```

```
mesh = mesh.merge(  
    mesh, mesh.vertices[...]  
)  
for key in mesh.vertices():  
    if key in fixed:  
        continue  
  
    p = key_xyz[key]  
  
    nbs = mesh.vertex_neighbours(key, ordered=True)  
    c = center_of_mass_polygon([key_xyz[nbr] for nbr in nbs])  
  
    # update  
    attr = mesh.vertex[key]  
    attr['x'] += d * (c[0] - p[0])  
    attr['y'] += d * (c[1] - p[1])  
    attr['z'] += d * (c[2] - p[2])  
  
if callback:  
    callback(mesh, k, callback_args)
```

```
def smooth_mesh_length(mesh, lmin, lmax, fixed=None, kmax=100):  
    if callback:  
        if not callable(callback):  
            raise Exception('Callback is not callable.')
```

```
fixed = fixed or []  
fixed = set(fixed)
```

```
for k in range(kmax):
```

# Review of last assignments

Create your own function that in some way modifies the output of COMPAS\_SLICER. You can choose any model as a base for the modifications, also your own design.



Advancing computational research

# Reproducibility & Upstreaming

A definition of reproducible research

Work that can be **independently recreated** from the **same data** and the **same code** that the original team used.

*The Turing Way, <https://github.com/alan-turing-institute/the-turing-way>*

← → ↺

github.com/alan-turing-institute/the-turing-way

🔍 ☆ 🌙 ⚙️ 👤 ⋮

JavaScript 1.5%

☰ README.md

# The Turing Way

read the book

receive our newsletter

chat on gitter

DOI

TuringWay

want to contribute?

all contributors 256

Test	Status
Netlify build	🔄 CI <span>passing</span>
No Latin Phrases	🔄 Check for Latin Phrases <span>passing</span>
No Large Files	🔄 TestFileSizes <span>passing</span>
No "Lorem Ipsum"s	🔄 Check for Lorem Ipsums <span>passing</span>

This README.md file is also available in Dutch ([README-Dutch](#)), French ([README-French.md](#)), German ([README-German.md](#)), Indonesian ([README-Indonesian](#)), Italian ([README-Italian](#)), Korean ([README-Korean](#)), Portuguese ([README-Portuguese](#)), and Spanish ([README-Spanish](#)) (listed alphabetically).

The *Turing Way* is a lightly opinionated guide to reproducible data science. You can read it here: <https://the-turing-way.netlify.com> You're currently viewing the project GitHub repository where all of the bits that make up the guide live, and where the process of writing/building the guide happens.

Our goal is to provide all the information that researchers need at the start of their projects to ensure that they are easy to reproduce at the end.

This also means making sure PhD students, postdocs, PIs and funding teams know which parts of the "responsibility of reproducibility" they can affect, and what they should do to nudge data science to being more efficient, effective and understandable.

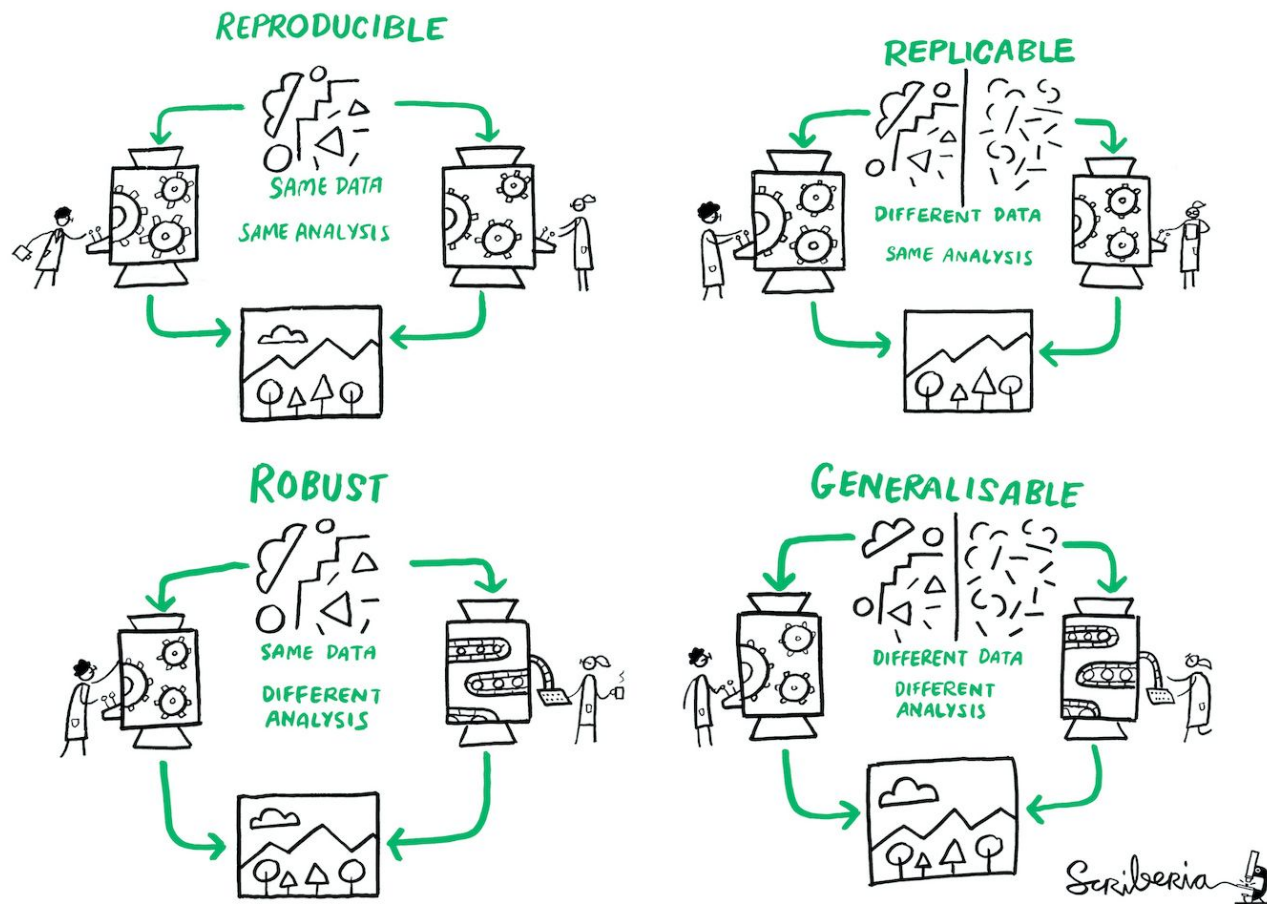
Table of contents:

- [About the project](#)
- [The team](#)
- [Contributing](#)
- [Citing \*The Turing Way\*](#)
- [Get in touch](#)
- [Contributors](#)

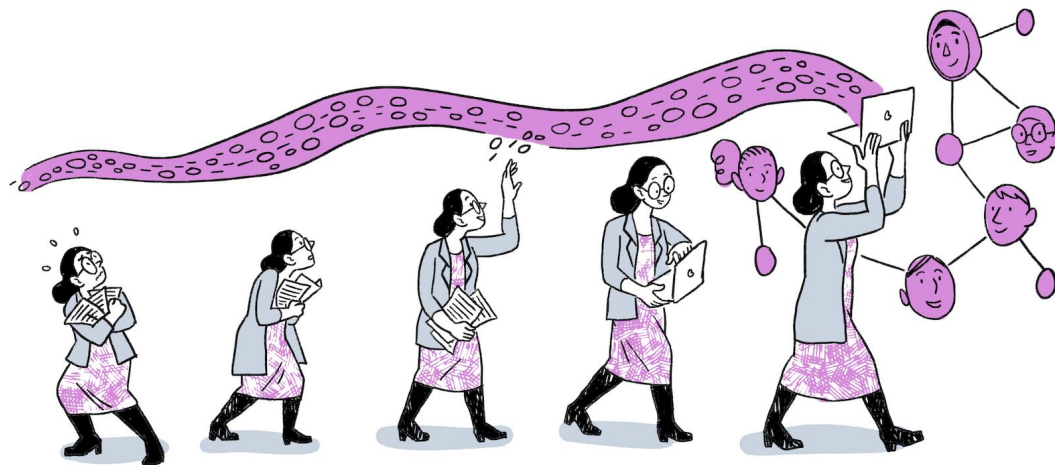
🔊 If you prefer an audio introduction to the project, our team member Rachael presented at the [Open Science Fair](#)

The Turing Way, <https://github.com/alan-turing-institute/the-turing-way>





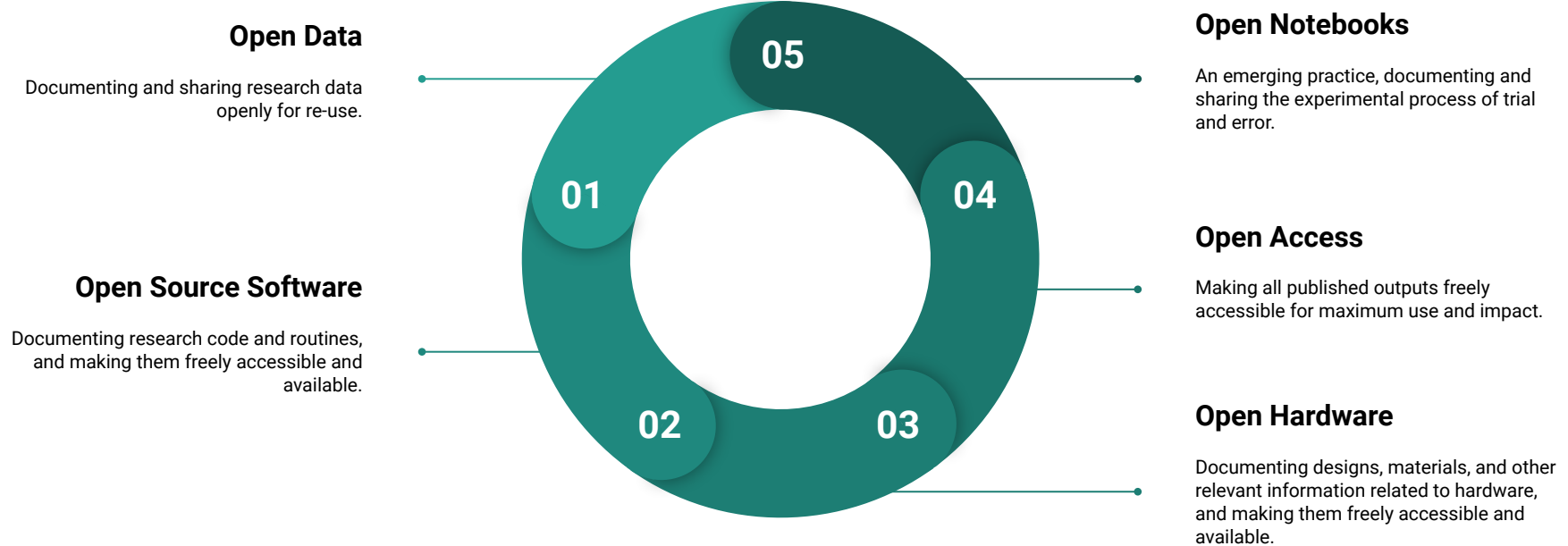




# EVOLVING TOWARDS AN ERA OF OPEN RESEARCH

Scriberia 

# Elements of open research



# Open Data

- Ensure your data is in a simple, **standard format** or formats which is machine and human-readable.
- Check, reformat or create **metadata to clearly describe what the data is**, how it was collected, and any associated strengths/weaknesses to someone that finds it.
- Identify a relevant, easily **discoverable** repository or repositories to host your data, and upload it there.
- Assign your data a **persistent identifier** such as a DOI.

# Open Source Software

- Put your code in a **freely accessible repository**.
- Include a **license** granting others the right to use, copy and modify your work.
- Include a **README** file containing useful information about a project such as what it is, how to use/install it and how to run any tests.
- If you want others to collaborate on the project include **contribution guidelines**.

# Open Hardware

- Make detailed **documentation and designs** for any hardware you develop **openly available**.
- Include a **license** granting others the right to use, copy and modify your work.
- Include a **README** file containing useful information about a project (for example, what it is and the materials used).

# Open Access

- Publish your research in an **open-access journal**.
- Store a copy or **preprint** of your work in a freely accessible public repository.

# Open Notebook

- Keep notes in an **Electronic Lab Notebook**.
- Make your notebooks **publicly accessible** online.

# Licenses

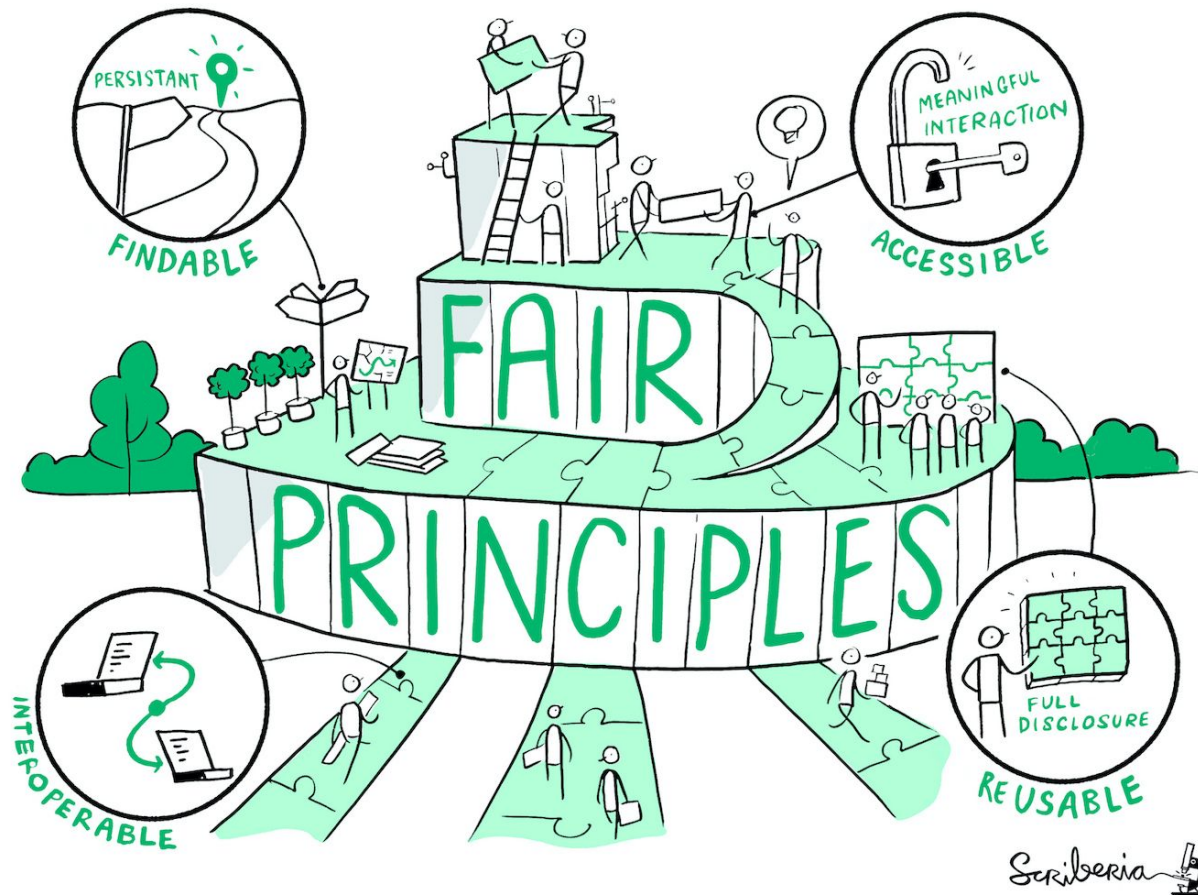


# Spectrum of OSS licenses



**[choosealicense.com](https://choosealicense.com)**

Data



# Software

# Code management



# Reproducible environments

- Tools for reproducible environments:
  - Package managers (conda/pip)
  - Containers (Docker)
- Use and rely on semantic versioning: [semver.org](https://semver.org)
  - MAJOR.MINOR.PATCH
  - e.g. 1.5.0
- Describe your dependencies meticulously



# Code quality

- **Always optimize for readability**
- Follow naming & style conventions (PEP8)
- Naming is hard, take your time
- Encapsulate complexity: solve it just once
- Don't reinvent the wheel, reuse encapsulated code
- Less code is always more
- **Leverage tools to assist your coding (linters, formatters, static analyzers, etc)**

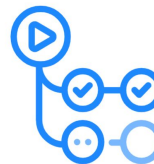


# Code testing

- There are lots of test types
  - Unit tests
  - Integration tests
  - Performance tests
- But at least, write unit tests. Please, pretty please! 😊
- Test algorithms, not UI
- If you fix a bug, write a test to verify and make sure you don't break it again.
- Run them automatically

# Continuous Integration

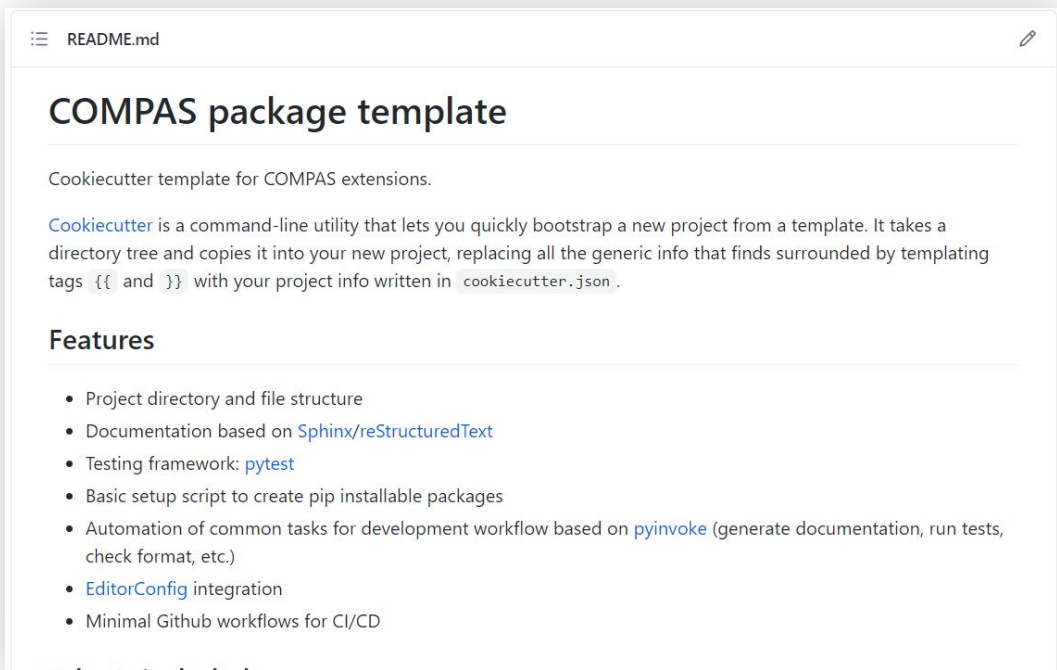
- Automate all the thingz!
  - Testing
  - Building/compiling
  - Publishing
  - Style / consistency checks
  - Anything repetitive!
- Makes sure you discover regressions when they occur



GitHub Actions

*How do I do all that?*

# COMPAS Templates



New extension/package: <https://github.com/compas-dev/tpl-extension>

# COMPAS Actions

## compas-actions.ghpython\_components

Trying to make Grasshopper development version-control friendlier since 1337.

Python MIT 1 3 0 0 Updated 17 hours ago

## compas-actions.docversions

0 0 0 0 Updated 16 days ago

## compas-actions.docs

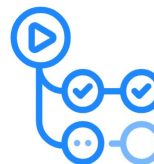
0 0 0 0 Updated 21 days ago

## compas-actions.publish

0 0 0 0 Updated 23 days ago

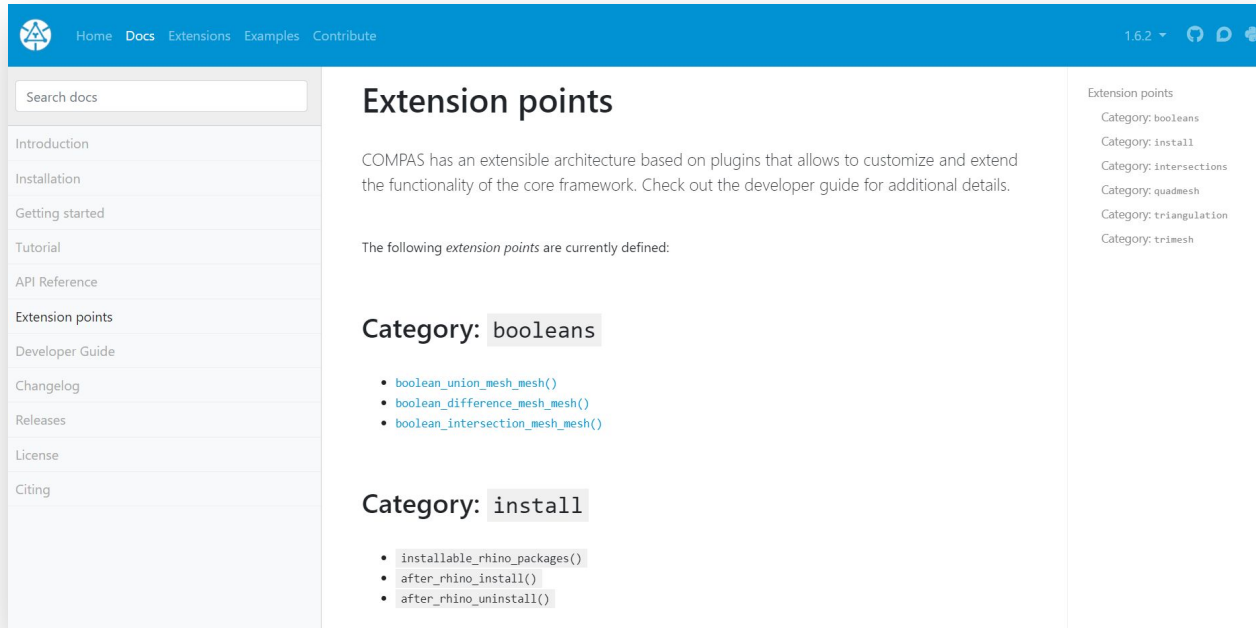
## compas-actions.build

0 0 0 0 Updated on Apr 14



GitHub Actions

# COMPAS Plugins



The screenshot shows the COMPAS Plugins website. The header is blue with the COMPAS logo on the left and navigation links (Home, Docs, Extensions, Examples, Contribute) in the center. On the right of the header, the version '1.6.2' and social media icons are displayed. A left sidebar contains a search bar and a list of navigation links. The main content area is titled 'Extension points' and contains a paragraph about the extensible architecture, a list of defined extension points, and two sections for 'Category: booleans' and 'Category: install' with their respective function lists. A right sidebar lists all extension point categories.

COMPAS Plugins Website Screenshot:

**Header:** COMPAS logo, Home, Docs, Extensions, Examples, Contribute, 1.6.2, GitHub, Discord, Telegram icons.

**Left Sidebar:**

- Search docs
- Introduction
- Installation
- Getting started
- Tutorial
- API Reference
- Extension points
- Developer Guide
- Changelog
- Releases
- License
- Citing

**Main Content:**

## Extension points

COMPAS has an extensible architecture based on plugins that allows to customize and extend the functionality of the core framework. Check out the developer guide for additional details.

The following *extension points* are currently defined:

### Category: booleans

- `boolean_union_mesh_mesh()`
- `boolean_difference_mesh_mesh()`
- `boolean_intersection_mesh_mesh()`

### Category: install

- `installable_rhino_packages()`
- `after_rhino_install()`
- `after_rhino_uninstall()`

**Right Sidebar:**

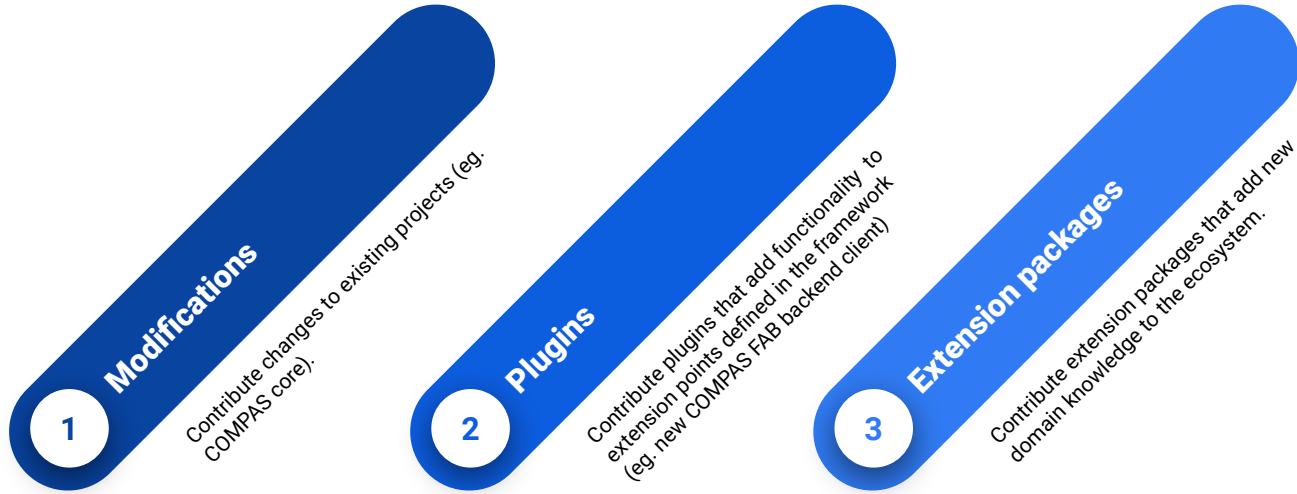
Extension points

- Category: booleans
- Category: install
- Category: intersections
- Category: quadmesh
- Category: triangulation
- Category: trimesh

Upstreaming research

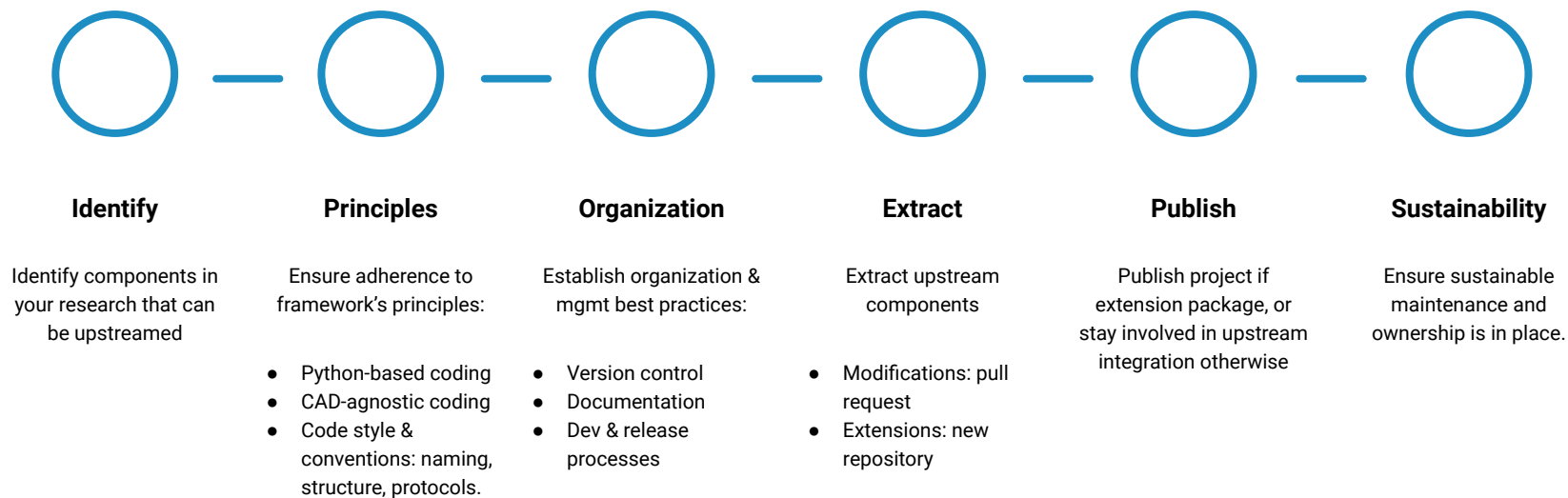
...is the process of **contributing** private research source code **back to the open source sphere**.

# Upstream contribution types





# Upstream process



# More resources

The Turing Way

<https://the-turing-way.netlify.app>

Awesome Reproducible Research (and other awesome-\* lists)

<https://github.com/leipzig/awesome-reproducible-research>

COMPAS Exchange (Tomorrow, 10-12am, registration required):

[https://ethz.zoom.us/meeting/register/u5Eocu2oqjkqGdVZ0m7Fz0dX\\_FaZIKLjvHaC](https://ethz.zoom.us/meeting/register/u5Eocu2oqjkqGdVZ0m7Fz0dX_FaZIKLjvHaC)

# Next week

# Thanks!

