



C O M P A S

064-0026-00L: COMPAS II

Introduction to Computational Methods for Digital
Fabrication in Architecture

```
    if not callable(callback):
        raise Exception('Callback is not callable.')
    for key in mesh.vertices():
        if key in mesh.vertices():
            continue
        p = key_xyz[key]
        nbrs = mesh.vertex_neighbours(key, ordered=True)
        c = center_of_mass_polygon([key_xyz[nbr] for nbr in nbrs])
        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, step=1, tolerance=0.01, max_iter=100, fixed_or=[]):
    if callback:
        if not callable(callback):
            raise Exception('Callback is not callable.')
    fixed = set(fixed)
    for k in range(kmax):
        if k % step == 0:
            print(f'Iteration {k}: Length = {mesh.length()}')
```

Team



Dr. Romana
Rust



Joris
Burger



Ioanna
Mitropoulou



Gonzalo
Casas



Philippe
Fleischmann

Course goals

Gain the ability to **design, plan and execute** robotic fabrication processes (e.g. discrete assemblies and additive manufacturing) using Python

slides + code

<https://dfab.link/fs2022>

asking questions



your goals + expectations

<https://dfab.link/fs2022-goals>

TODAY

intro

course overview

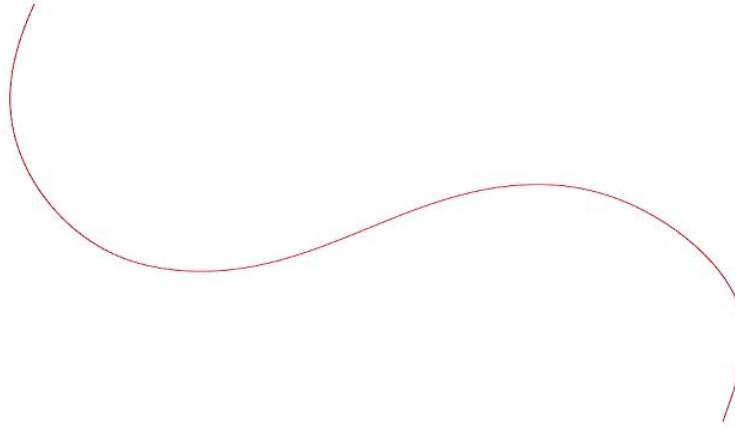
team

dfab toolbox

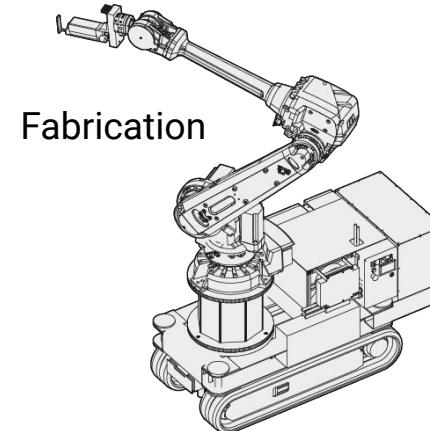
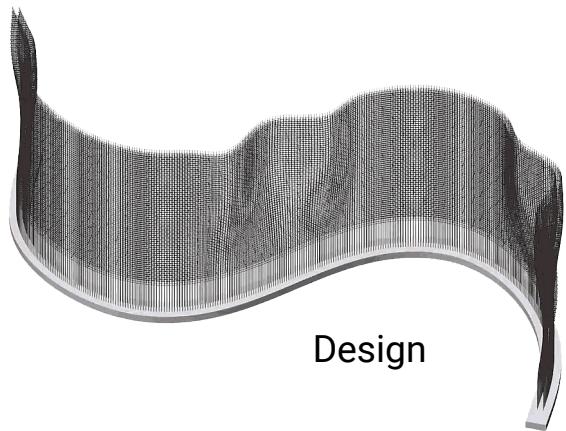
datastructures

intro
course overview
dfab toolbox
datastructures

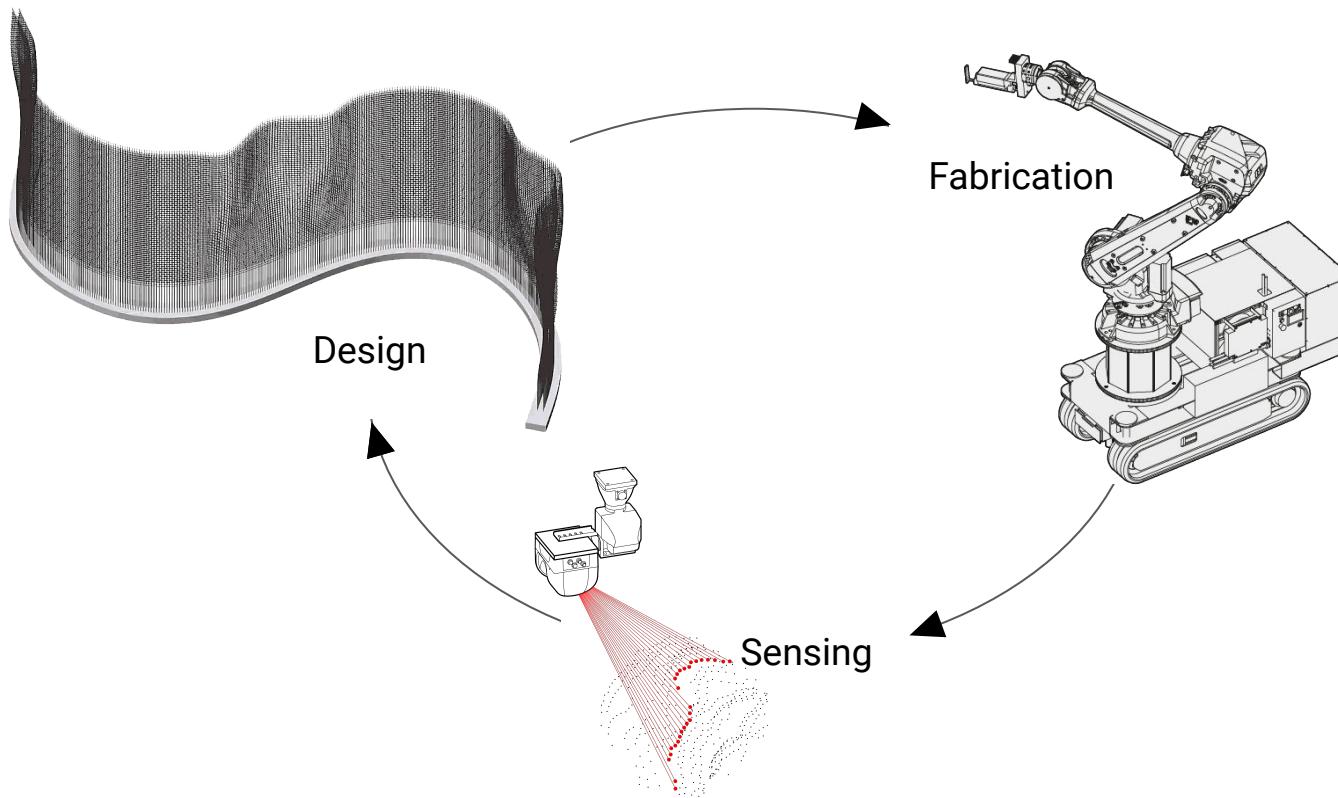
Computational design



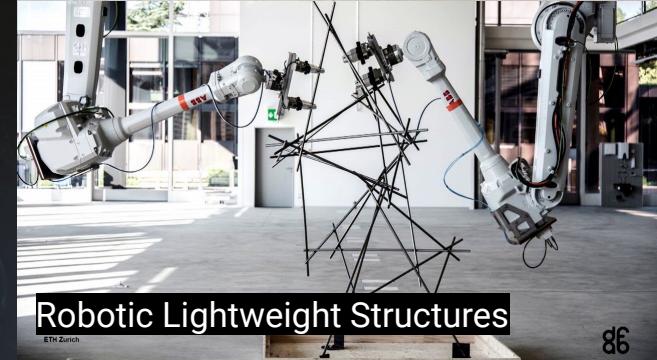
Design to fabrication



Design to fabrication

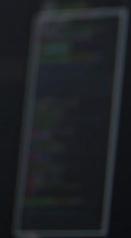


Interdisciplinarity





C O M P A S



```
    if key in mesh.vertices():
        if key in fixed:
            continue
        p = key_xyz[key]
        nbrs = mesh.vertex_neighbours(key, ordered=True)
        c = center_of_mass_polygon([key_xyz[nbr] for nbr in nbrs])
        d = np.linalg.norm(c - p)
        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):
```

intro
course overview
dfab toolbox
datastructures

Lecture	Date	Session content
01	23.02.	Introduction
02	02.03.	Robotic fundamentals
03	09.03.	Robot models
04	16.03.	ROS & MoveIt in the design environment
05	30.03.	Path planning
06	06.04.	Assembly of discrete elements I: model
07	13.04.	Assembly of discrete elements II: plan
08	27.04.	Robot control with COMPAS RRC
09	04.05.	Assembly of discrete elements III: execution
10	11.05.	COMPAS SLICER: Basics
11	18.05.	COMPAS SLICER: Advanced
12	25.05.	Advancing computational research
13	01.06.	Closing

Assignments

Weekly coding assignments

Submit using pull requests to course repo

Each week, the assignment of previous week is due

Submission deadline, Wednesdays 9AM

Grading

Coding assignments graded on problem solving

Coding style not part of grading, but included in feedback

Course graded as average of all assignments

Office hours

Usually on Friday afternoons, coordinate/request via Slack DM

Links

Course materials

<https://dfab.link/fs2022>

Slack

https://join.slack.com/t/compasii/shared_invite/zt-14cbtllxc-zxZRErT54F4xayA6keTb0q

Forum

<https://forum.compas-framework.org>

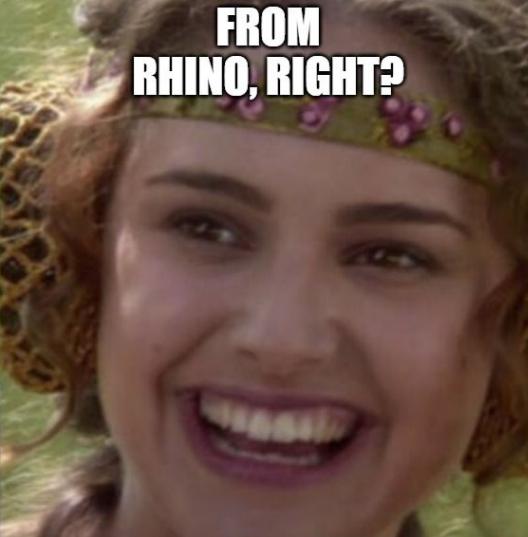
intro
course overview
dfab toolbox
datastructures



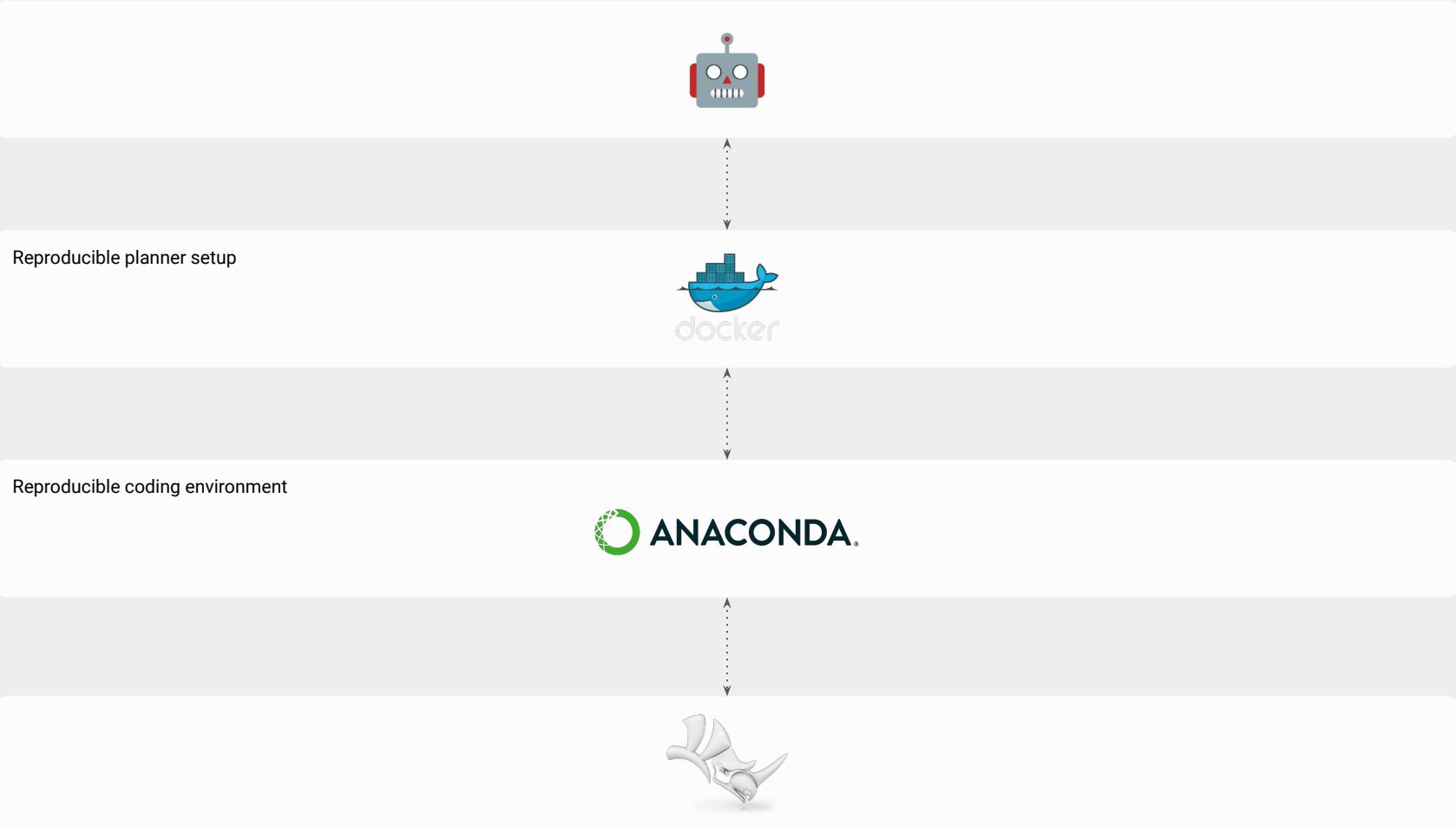
LET'S CONNECT
TO A ROBOT

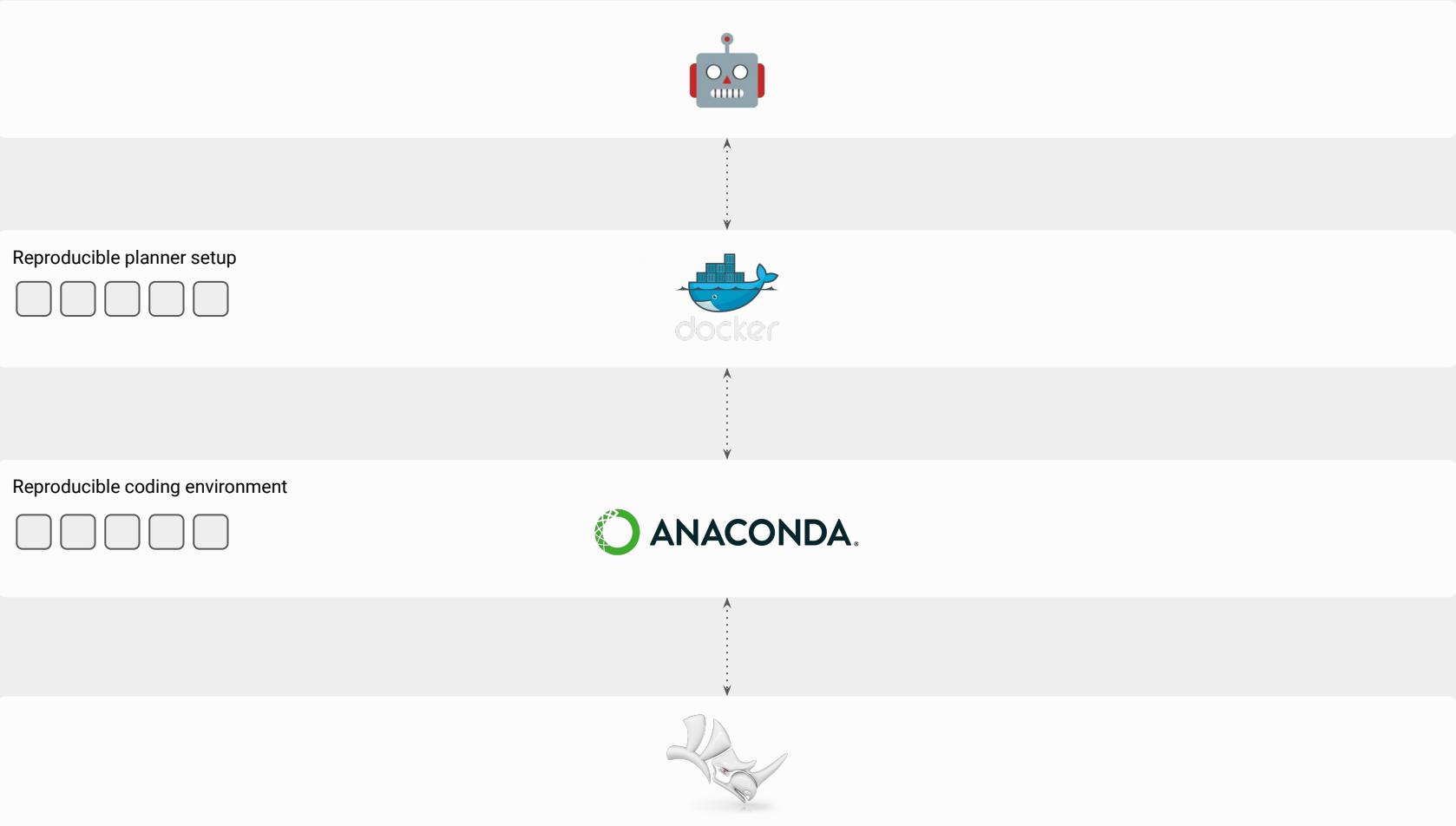


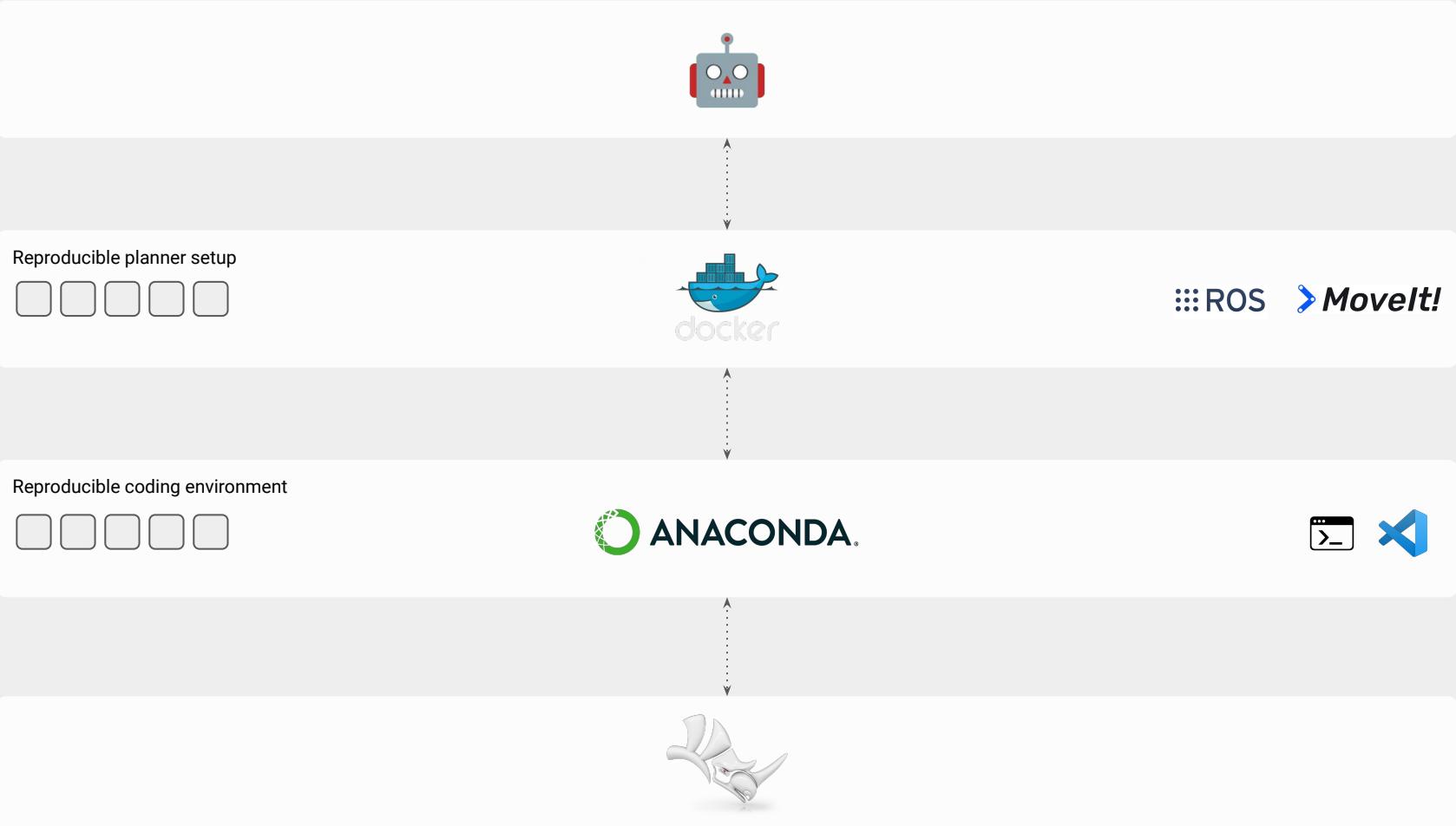
FROM
RHINO, RIGHT?

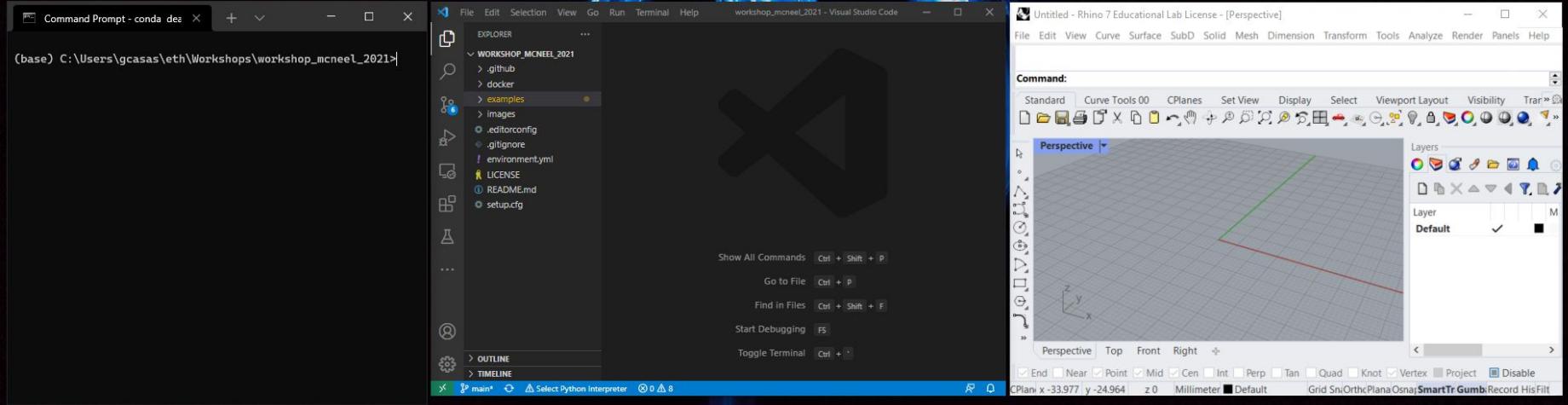


FROM
RHINO, RIGHT?





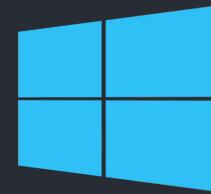




Installation

<https://dfab.link/fs2022>

⌘ + SPACE

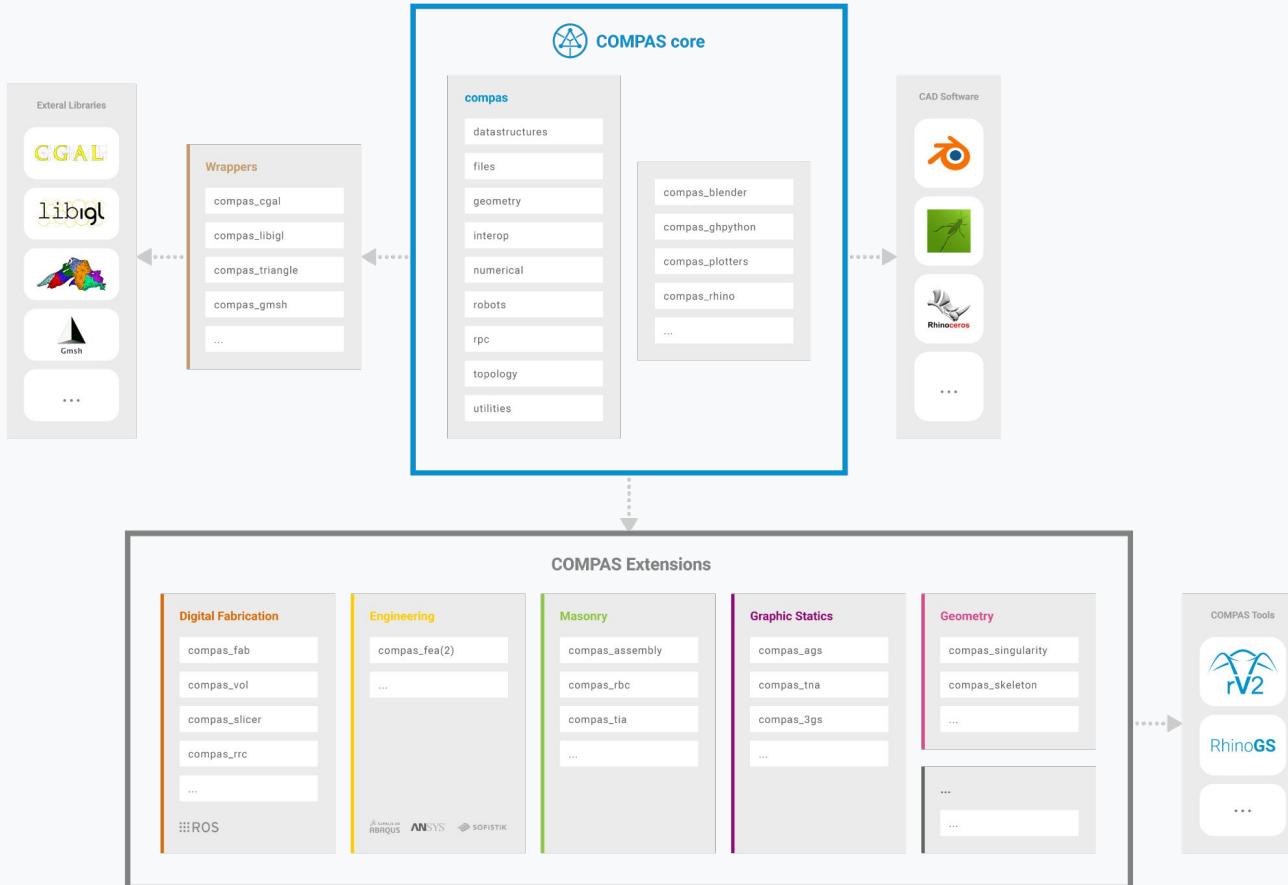


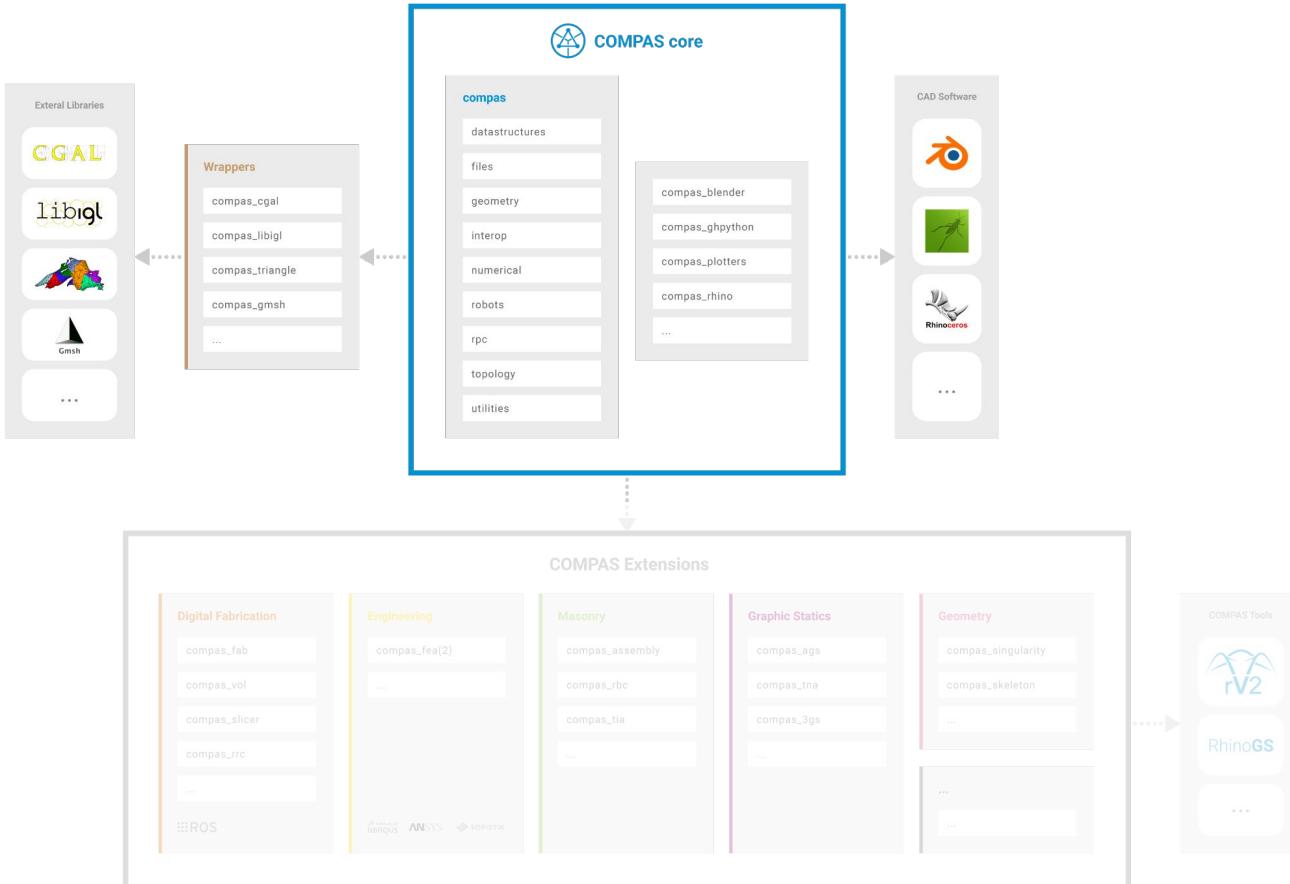
Terminal

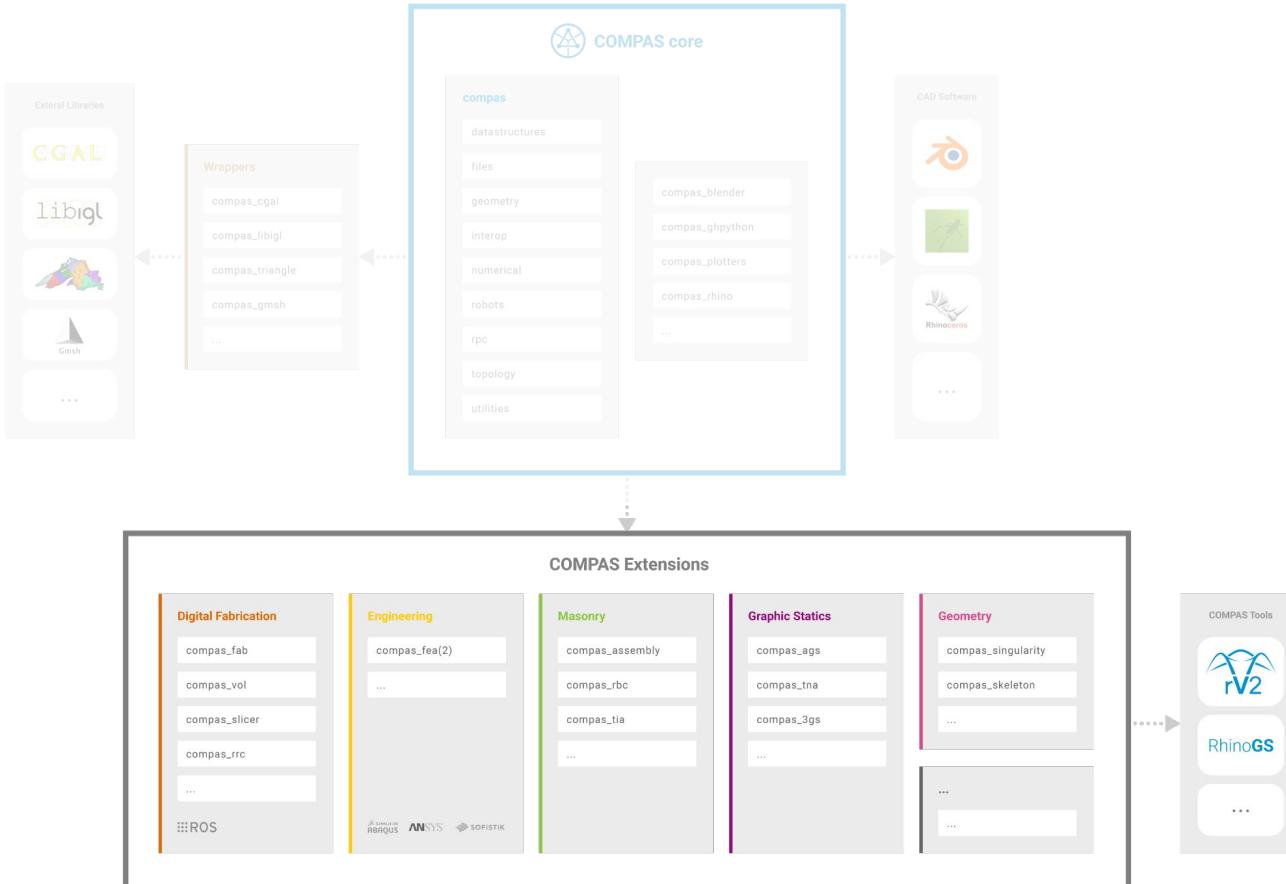
Anaconda Prompt

Create
environment

```
(base) conda config --add channels conda-forge  
(base) conda env create -f https://dfab.link/fs2022.yml
```







COMPAS core

compas.data

compas.scene

compas.datastructures

compas.topology

compas.files

compas.utilities

compas.geometry

compas.numerical

compas_plotters

compas.plugins

compas_rhino

compas.robots

compas_ghpython

compas.rpc

compas_blender

COMPAS core extensions

compas_cgal

compas_cloud

compas_dashboard

compas_gmsh

compas_libigl

compas_occ

compas_triangle

compas_view2

Data modelling & Assessment

compas_ifc

compas_lca

Digital Fabrication

compas_fab

compas_slicer

compas_rrc

compas_vol

compas_timber

Engineering

compasfea(2)

Data modelling & Assessment

`compas_ifc`

`compas_lca`

Digital Fabrication

`compas_fab`

`compas_slicer`

`compas_rrc`

`compas_vol`

`compas_timber`

Engineering

`compasfea(2)`

```
(compas-dev-38) C:\Users\gcasas>python
Python 3.8.10 | packaged by conda-forge | (default, May 11 2021, 06:25:23) [MSC v.1916 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> import compas
>>> from compas.datastructures import Mesh
>>> mesh = Mesh.from_obj(compas.get('faces.obj'))
>>> print(mesh.summary())
Mesh summary
=====
- vertices: 36
- edges: 60
- faces: 25
>>> |
```

CAD-agnostic and cross-platform

Usage from the CPython interpreter

dr_numpy.py — compas-dev

```

FOLDERS
├── compas-dev
│   ├── compas
│   ├── build
│   ├── data
│   ├── dist
│   ├── docs
│   ├── libs
│   └── src
│       ├── compas
│       │   ├── __init__.py
│       │   ├── __pycache__
│       │   ├── com
│       │   ├── datastructures
│       │   ├── files
│       │   ├── geometry
│       │   ├── interop
│       │   ├── numerical
│       │   │   ├── __pycache__
│       │   │   ├── alglib
│       │   │   ├── algorithms
│       │   │   ├── descent
│       │   │   ├── devo
│       │   │   ├── dr
│       │   │   ├── drs
│       │   │   │   ├── __pycache__
│       │   │   │   ├── __init__.py
│       │   │   │   ├── drs_numpy.py
│       │   │   ├── fd
│       │   │   ├── ga
│       │   │   ├── lma
│       │   │   ├── mma
│       │   │   ├── pca
│       │   │   ├── solvers
│       │   │   ├── topop
│       │   │   └── __init__.py
│       │   ├── linalg.py
│       │   ├── matrices.py
│       │   ├── operators.py
│       │   ├── utilities.py
│       │   ├── plotters
│       │   ├── robots
│       │   ├── rpc
│       │   ├── topology
│       │   ├── utilities
│       │   ├── viewers
│       │   └── __init__.py
│       ├── COMPAS-app-info
│       ├── compas.blender
│       ├── compas.ghpython
│       ├── compas.hpc
│       ├── compas.revit
│       ├── compas_rhino
│       ├── temp
│       └── tests
│           ├── bumpversion.cfg
│           ├── editconfig
│           ├── ignore
│           ├── travis.yml
│           └── CONTRIBUTORS.md
└── LICENSE

Git branch: master, index: <-, working: 10y, Line 33, Column 1

```

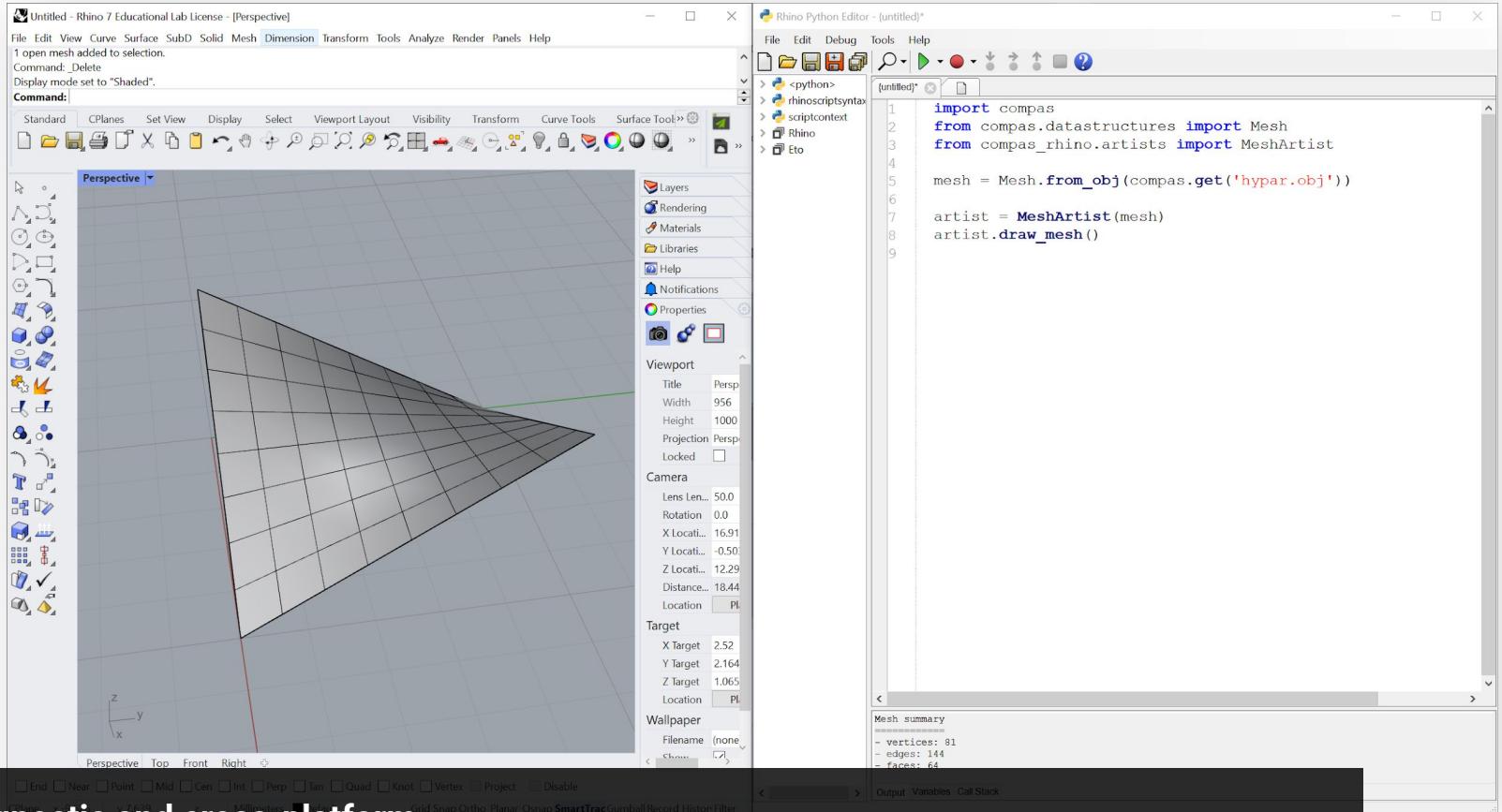
```

dr_numpy.py x
1  from __future__ import absolute_import
2  from __future__ import division
3  from __future__ import print_function
4
5  import compas
6
7  try:
8      from numpy import array
9      from numpy import meshgrid
10     from numpy import isnan
11     from numpy import ones
12     from numpy import zeros
13     from scipy.linalg import norm
14     from scipy.sparse import diags
15
16     except ImportError:
17         raise_if_not_ipython()
18
19     from compas.numerical import connectivity_matrix
20     from compas.numerical import normrow
21
22
23     __all__ = ['dr_numpy']
24
25
26 K = [
27     [0.5, 0.5],
28     [0.5, 0.5],
29     [0.5, 0.0, 0.5],
30     [[1.0, 0.0, 0.0], 1.0],
31 ]
32
33
34 class Coeff():
35     def __init__(self, c):
36         self.c = c
37         self.a = (1 - c * 0.5) / (1 + c * 0.5)
38         self.b = 0.5 / (1 + self.a)
39
40
41 def dr_numpy(vertices, edges, fixed, loads, ure, fpre, lpre, limt, E, radius,
42             callback=None, callback_args=None, **kwargs):
43     """Implementation of the dynamic relaxation method for form finding and analysis
44     of articulated networks of axial-force members.
45
46     Parameters
47
48     vertices : list
49         XYZ coordinates of the vertices.
50     edges : list
51         Connectivity of the vertices.
52     fixed : list
53         Indices of the fixed vertices.
54     loads : list
55         XYZ components of the loads on the vertices.
56     qpre : list
57         Prescribed force densities in the edges.
58     fpre : list
59         Prescribed forces in the edges.
60     lpre : list
61         Prescribed lengths of the edges.
62     limt : float
63         Initial length of the edges.
64     E : float
65         Stiffness of the edges.
66     radius : list
67         Radius of the edges.
68     callback : callable, optional
69         User-defined function that is called at every iteration.
70     callback_args : tuple, optional
71         Additional arguments passed to the callback.
72
73 Notes
74 -----
75 For more info, see [1]_.
76
77 References
78 .. [1] De Lant L., Veenendaal P., Van Hee T., Hollart H. and Block P.,
79    .. «Bending incorporated: designing tension structures by integrating
80    .. Proceedings of Tensioni Symposium 2013, Istanbul, Turkey, 2013.
81
82 Examples
83 -----
84 ... plot:
85 ... travis:
86 ... include-source:
87
88     import compas
89     from compas.datastructures import Network

```

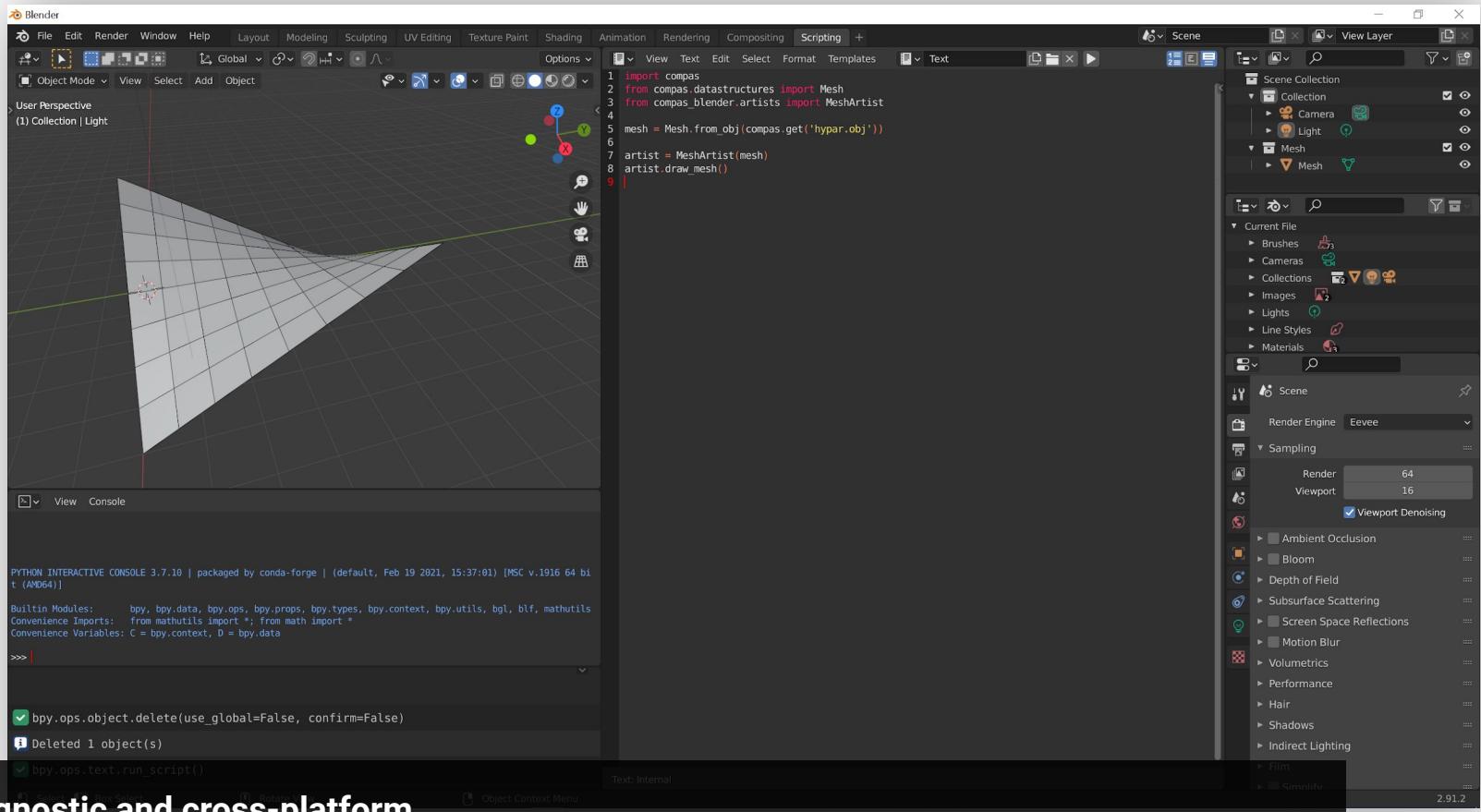
CAD-agnostic and cross-platform

Usage with standalone viewer



CAD-agnostic and cross-platform

Usage from Rhino on IronPython



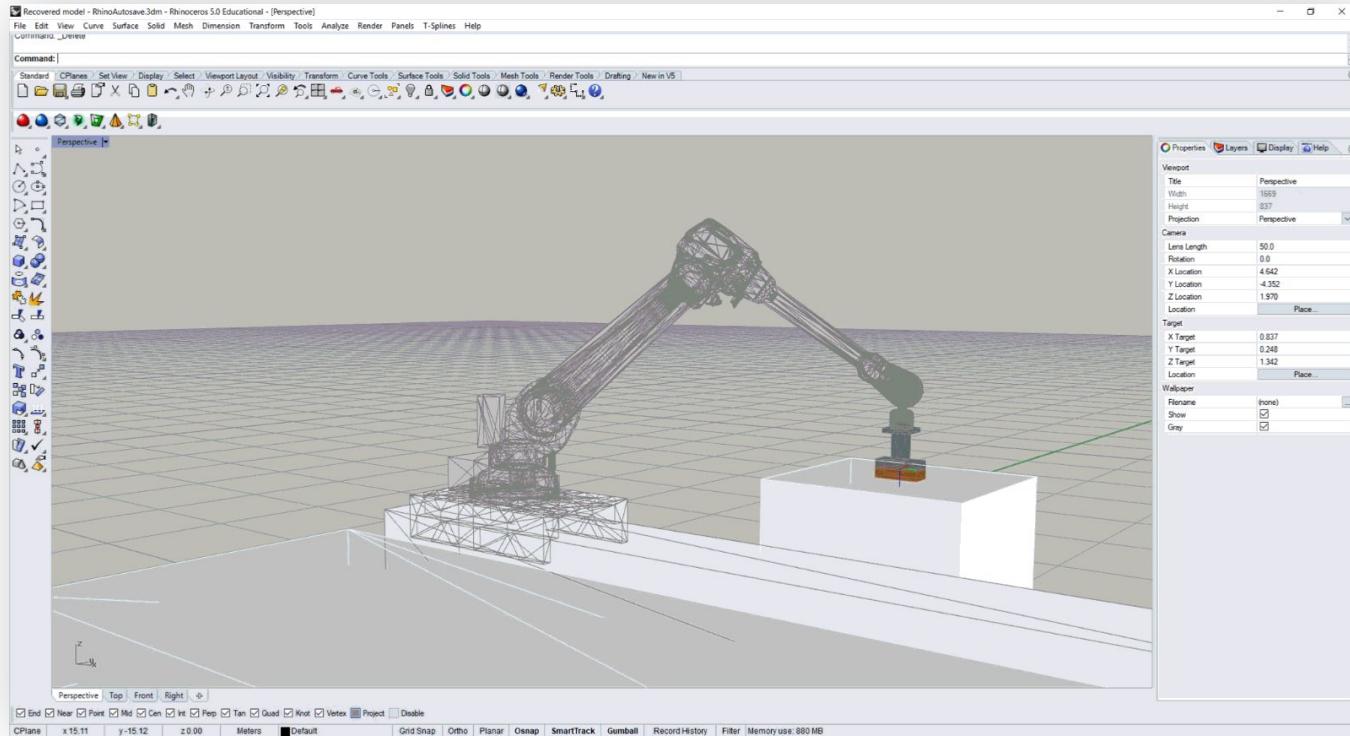
CAD-agnostic and cross-platform

Usage from Blender on CPython

```
# Any supported CAD
import compas
from compas.artists import Artist
from compas.datastructures import Mesh

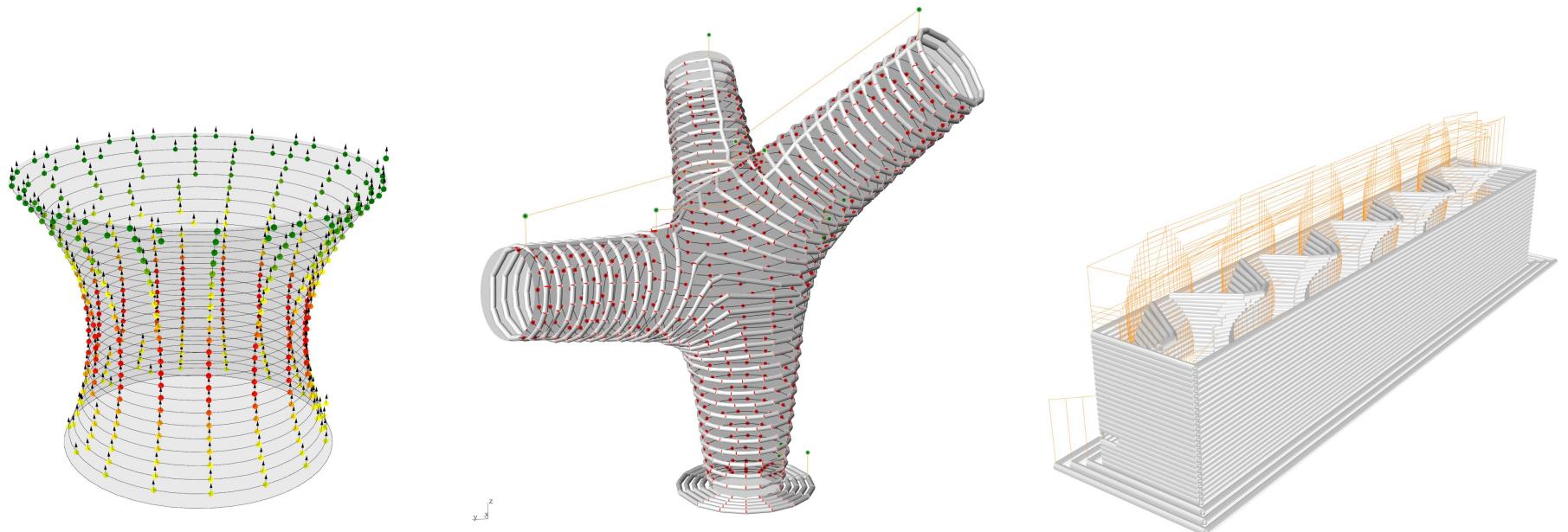
mesh = Mesh.from_obj(compas.get('hypar.obj'))

artist = Artist(mesh)
artist.draw_mesh()
```



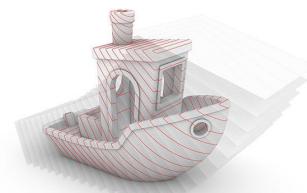
COMPAS FAB

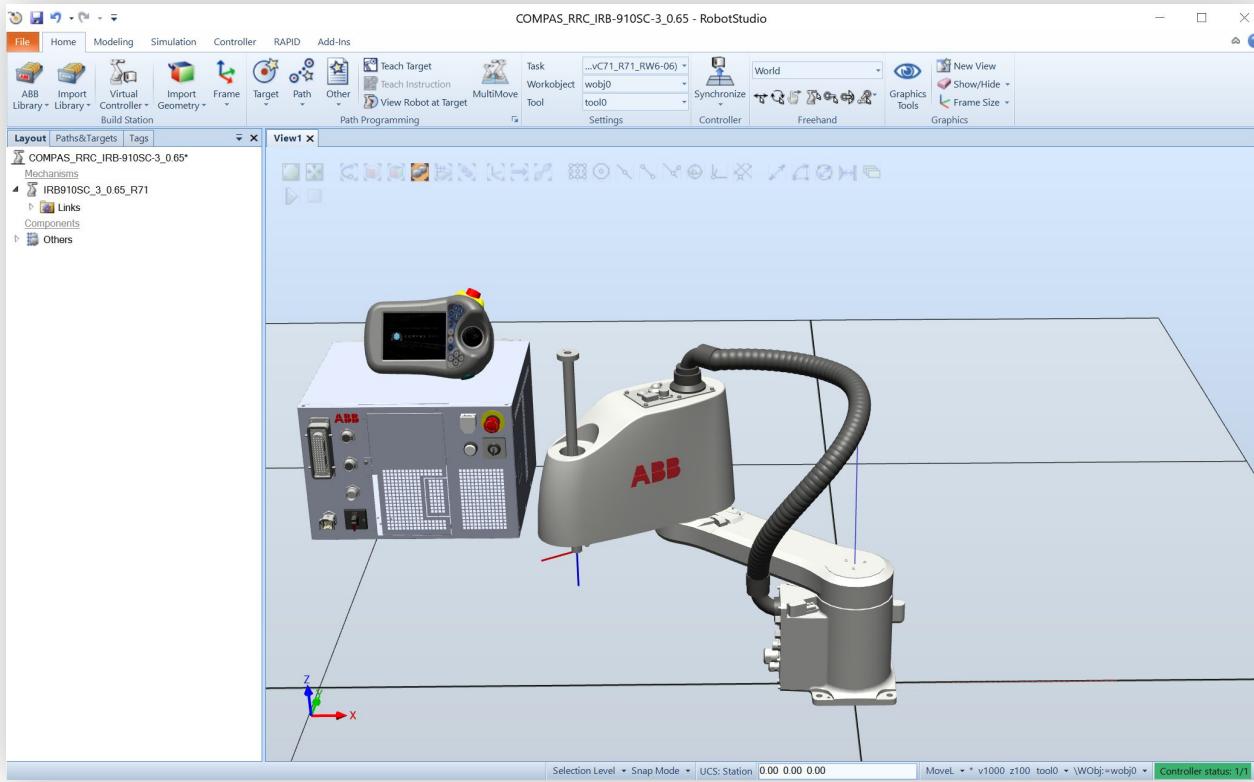
Robotic fabrication package



COMPAS SLICER

Slicing package for FDM 3D Printing





COMPAS RRC

Robot control for ABB robots

Installation

https://dfab.link/fs2022

Activate

```
(base) conda activate fs2022
```

```
(fs2022) python -m compas_rhino.install -v 7.0
```

```
(fs2022) cd Documents
```

```
(fs2022) git clone https://github.com/compas-teaching/COMPAS-II-FS2022
```

Restart Rhino

```
(base) conda activate fs2022
(fs2022) python -m compas_rhino.install -v 7.0
(fs2022) cd Documents
(fs2022) git clone https://github.com/compas-teaching/COMPAS-II-FS2022
```

[compas-teaching / COMPAS-II-FS2022](#) (Public)

Code Issues Pull requests Actions Projects Wiki Security Insights Settings

main 1 branch 0 tags

Add file ▾ <> Code ▾

gonzalocasas Update README.md ctab5b 1 hour ago 2 commits

.github	Initial commit	28 days ago
.editorconfig	Initial commit	28 days ago
.gitignore	Initial commit	28 days ago
LICENSE	Initial commit	28 days ago
README.md	Update README.md	1 hour ago
environment.yml	Initial commit	28 days ago
setup.cfg	Initial commit	28 days ago

README.md

COMPAS II

064-0026-00L Introduction to Computational Methods for Digital Fabrication in Architecture

Pin Unwatch 5 Fork 0 Star 3

About
No description, website, or topics provided.

Readme
MIT License
3 stars
5 watching
0 forks

Releases
No releases published
Create a new release

Packages
No packages published
Publish your first package

Choose any target
folder

```
(base) conda activate fs2022
(fs2022) python -m compas_rhino.install -v 7.0
(fs2022) cd Documents
(fs2022) git clone https://github.com/compas-teaching/COMPAS-II-FS2022
```

Clone repository

```
(base)    conda activate fs2022
(fs2022) python -m compas_rhino.install -v 7.0
(fs2022) cd Documents
(fs2022) git clone https://github.com/compas-teaching/COMPAS-II-FS2022
```

Select environment

The screenshot shows the Visual Studio Code interface with the following details:

- File Explorer:** Shows a folder named "WORKSHOP_SWINBURNE_2021" containing numerous Python files (e.g., 018.mesh_info_vertex_neighbors.py, 019.mesh_info_edge_strip.py, etc.).
- Terminal:** The title bar says "101_several_ways_to_construct_frame.py - workshop_swinburne_2021 - Visual Studio Code". A dropdown menu titled "Select your default terminal profile" is open, listing several options:
 - PowerShell C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe
 - Git Bash C:\Program Files\Git\bin\bash.exe - login
 - Command Prompt C:\Windows\System32\cmd.exe** (selected)
 - Ubuntu (WSL) C:\Windows\System32\wsl.exe -d Ubuntu
 - Windows PowerShell C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe
- Terminal Content:** The terminal window shows the following text:

```
Microsoft Windows [Version 10.0.19042.985]
(c) Microsoft Corporation. All rights reserved.

C:\Users\gcasas\eth\Workshops\workshop_swinburne_2021>C:/Users/gcasas/anaconda3/Scripts/activate
base) C:\Users\gcasas\eth\Workshops\workshop_swinburne_2021>conda activate swinburne
(swinburne) C:\Users\gcasas\eth\Workshops\workshop_swinburne_2021>
```
- Bottom Status Bar:** Shows "Python 3.8.10 64-bit (swinburne: conda)" and other status indicators.

Select Command Prompt profile

A screenshot of the Visual Studio Code interface. The top right corner features a blue callout box with the text "Select Command Prompt profile". A red rectangle highlights the "profiles" dropdown menu in the terminal bar, which lists several terminal profiles: "PowerShell", "Git Bash", "Command Prompt" (which is selected), "Ubuntu (WSL)", and "Windows PowerShell". An arrow points from the callout box towards this highlighted area. The main workspace shows an "EXPLORER" sidebar with a file tree for "WORKSHOP_SWINBURNE_2021" containing numerous Python files. The "TERMINAL" tab is active, displaying a Microsoft Windows command prompt window. The command history includes activating a conda environment named "swinburne" and navigating to its directory. The status bar at the bottom indicates the terminal is running "Python 3.8.10 64-bit ('swinburne': conda)" and shows line 3, column 34.

```
101_several_ways_to_construct_frame.py
101_several_ways_to_construct_frame.py
examples > 101_several_ways_to_construct_frame.py
1    """There are several ways to construct frames in compas.
2
3    from compas.geometry import Point
4    from compas.geometry import Vector
5    from compas.geometry import Frame
6    from compas.geometry import Plane
7
8    # Frame autocorrects axes to be orthonormal
9    F = Frame(Point(1, 0, 0), Vector(-0.45, 0.1, 0.3), Vector(1, 0, 0))
10
11   F = Frame([1, 0, 0], [-0.45, 0.1, 0.3], [1, 0, 0])
12
13   F = Frame.from_points([1, 1, 1], [2, 3, 6], [6, 3, 0])
14   F = Frame.from_plane(Plane([0, 0, 0], [0.5, 0.2, 0.1]))
15   F = Frame.from_euler_angles([0.5, 1., 0.2])
16   F = Frame.worldXY()
17
18
101_several_ways_to_construct_frame.py
102_point_in_frame.py
103_frame_in_frame.py
104_box_from_the_world_to_local.py
105_box_from_the_world_to_local.r...
106_examples_transformation.py
107_inverse_transformation.py
108_premultiply_transformations.py
109_pre_vs_post_multiplication.py
110_decompose_transformation.py
111_transform_point_and_vector.py
112_transform_multiple.py
113_change_basis_transformation.py
114_transformation_between_fram...
115_box_from_the_world_to_local.r...
116_several_ways_to_construct_rota...
117_robot_tcp_orientations.py
118_euler_angles.py
119_axis_angle.py
120_quaternion.py
201_visualize_model_rhino.py ... M
> OUTLINE
> TIMELINE
```

File Edit Selection View Go Run Terminal Help

101 several ways to construct frame.py - workshop_swinburne_2021 - Visual Studio Code

select your default terminal profile

- PowerShell C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe
- Git Bash C:\Program Files\Git\bin\bash.exe -login
- Command Prompt** C:\Windows\System32\cmd.exe
- Ubuntu (WSL) C:\Windows\System32\wsl.exe -d Ubuntu
- Windows PowerShell C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe

cmd + ^ x

Microsoft Windows [Version 10.0.19042.985]
(c) Microsoft Corporation. All rights reserved.
C:\Users\gcasas\eth\Workshops\workshop_swinburne_2021>C:/Users/gcasas/anaconda3/Scripts/activate
(base) C:\Users\gcasas\eth\Workshops\workshop_swinburne_2021>conda activate swinburne
(swinburne) C:\Users\gcasas\eth\Workshops\workshop_swinburne_2021>

main* Python 3.8.10 64-bit ('swinburne': conda) 0 △ 0

Ln 3, Col 34 Spaces: 4 UFT-8 CRLF Python R L

intro
course overview
dfab toolbox
datastructures

Primitives & Shapes

compas.geometry

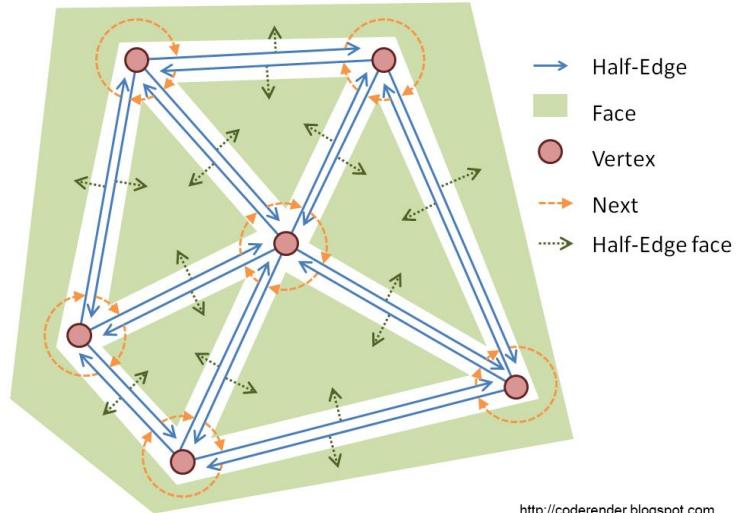
- point, vector, line, plane, circle, polyline, polygon, frame
- transformation, translation, rotation, scale, reflection, projection, shear
- box, sphere, polyhedron, torus, cone, capsule

Object	Python	COMPAS
point	<code>point = [0, 0, 0]</code>	<code>point = Point(0, 0, 0)</code>
vector	<code>vector = [0, 0, 1]</code>	<code>vector = Vector(0, 0, 1)</code>
line	<code>line = [0, 0, 0], [1, 0, 0]</code>	<code>line = Line(point, point)</code>
plane	<code>plane = [0, 0, 0], [0, 0, 1]</code>	<code>plane = Plane(point, vector)</code>
circle	<code>circle = ([0, 0, 0], [0, 0, 1]), 1.0</code>	<code>circle = Circle(plane, radius)</code>
polyline	<code>polyline = [0, 0, 0], [1, 0, 0], [1, 1, 0], [0, 0, 0]</code>	<code>polyline = Polyline(points)</code>
polygon	<code>polygon = [0, 0, 0], [1, 0, 0], [1, 1, 0]</code>	<code>polygon = Polygon(points)</code>
frame	<code>frame = [0, 0, 0], [1, 0, 0], [0, 1, 0]</code>	<code>frame = Frame(point, xaxis, yaxis)</code>

Mesh

compas.datastructures

- **CAD-agnostic**
- half-edge data structure
- support for n-sided polygonal faces
- open or closed, polygonal surfaces
- custom attributes at mesh, vertex, edge and face

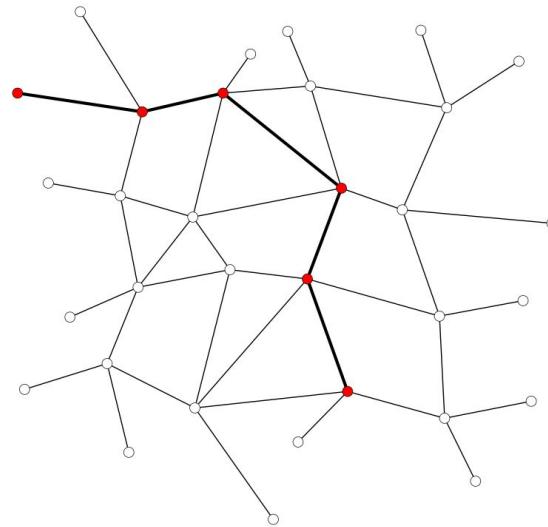


<http://coderender.blogspot.com>

Network

compas.datastructures

- directed edge graph data structure
- graph: topological
- network: geometric implementation of graph
- custom attributes at network, node and edge
- networkx support



Remote Procedure Calls

compas.rpc



Next week

- Make sure examples run on your machine
- Ask for help if needed
 - Slack
 - Forum
- Next week:
 - Robotic fundamentals

Thanks!

