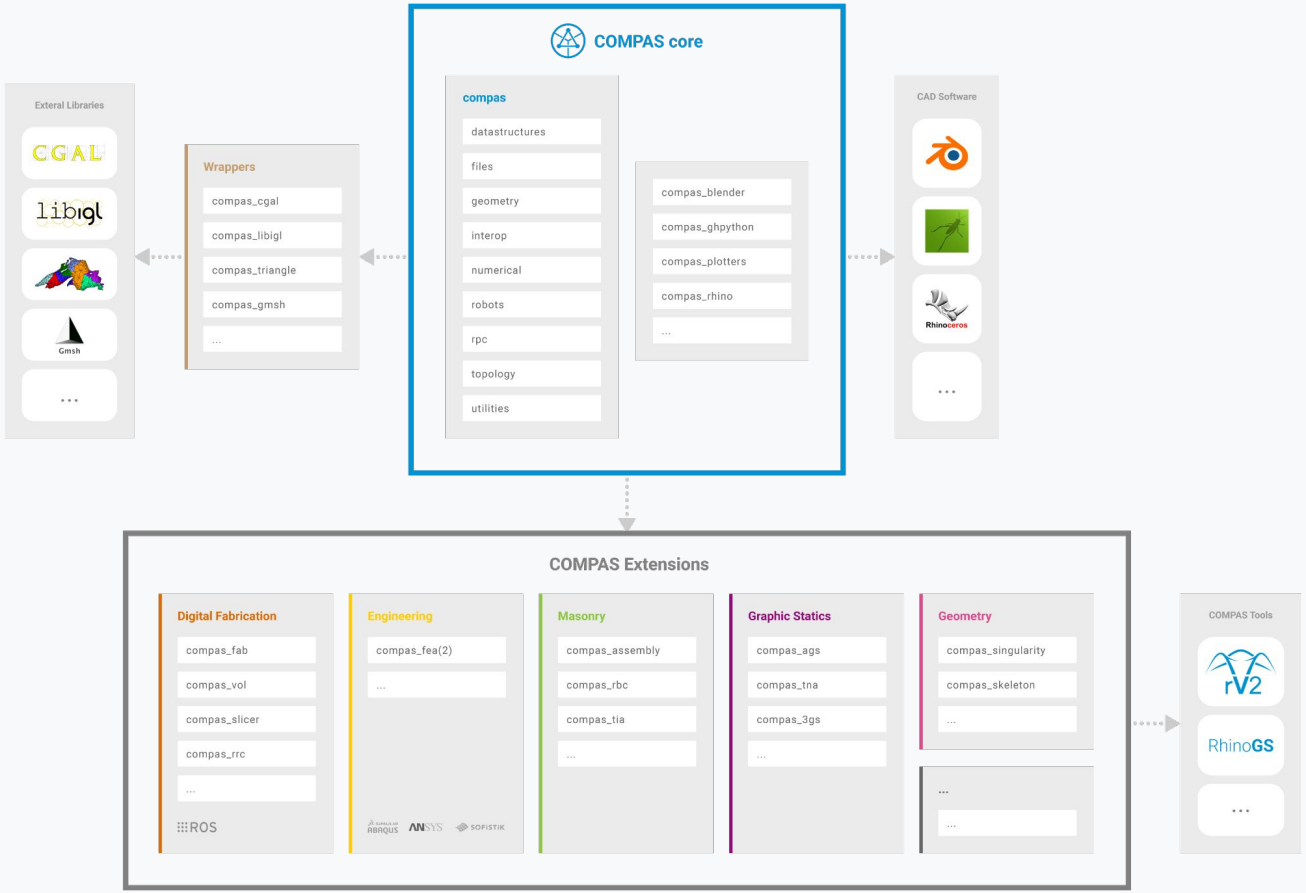
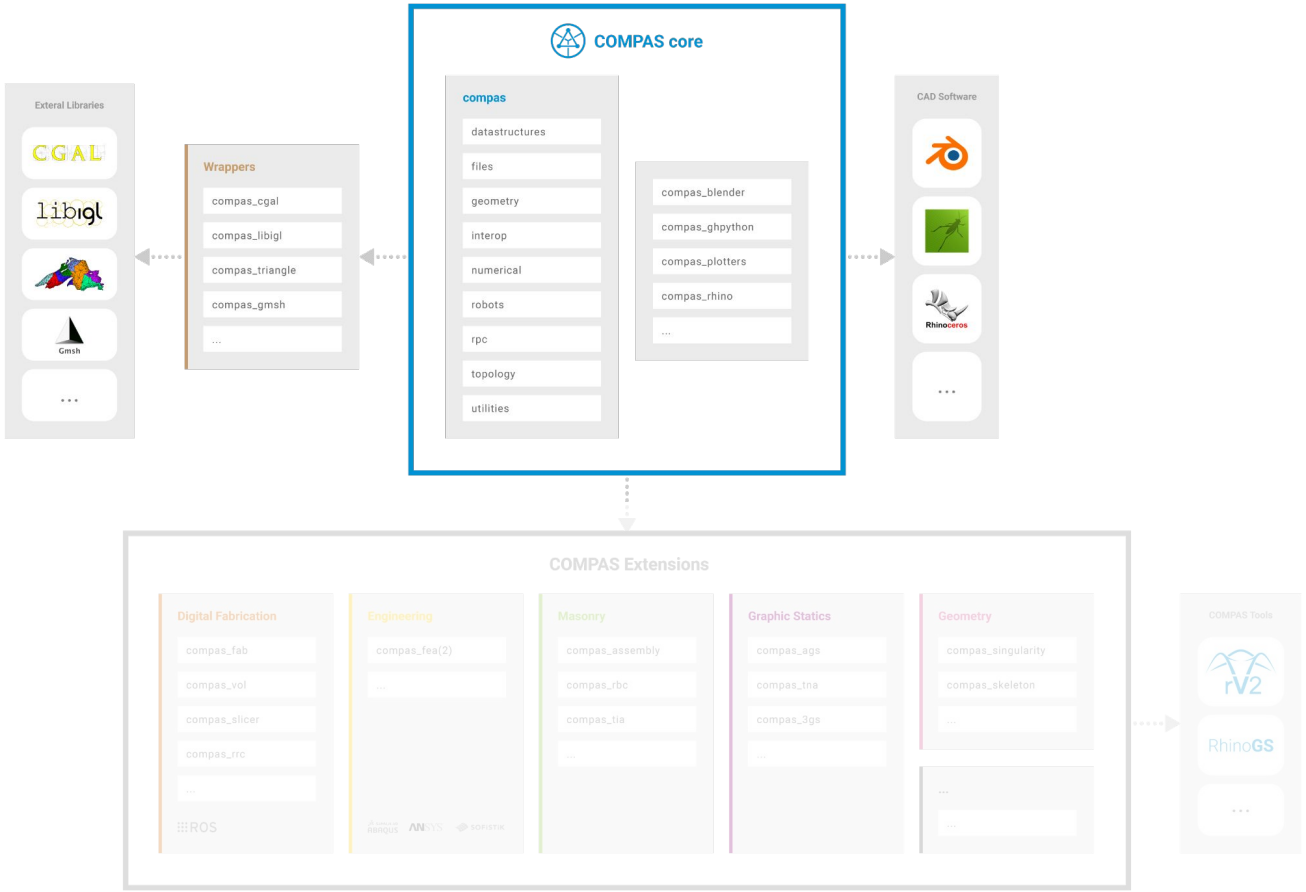
COMPAS

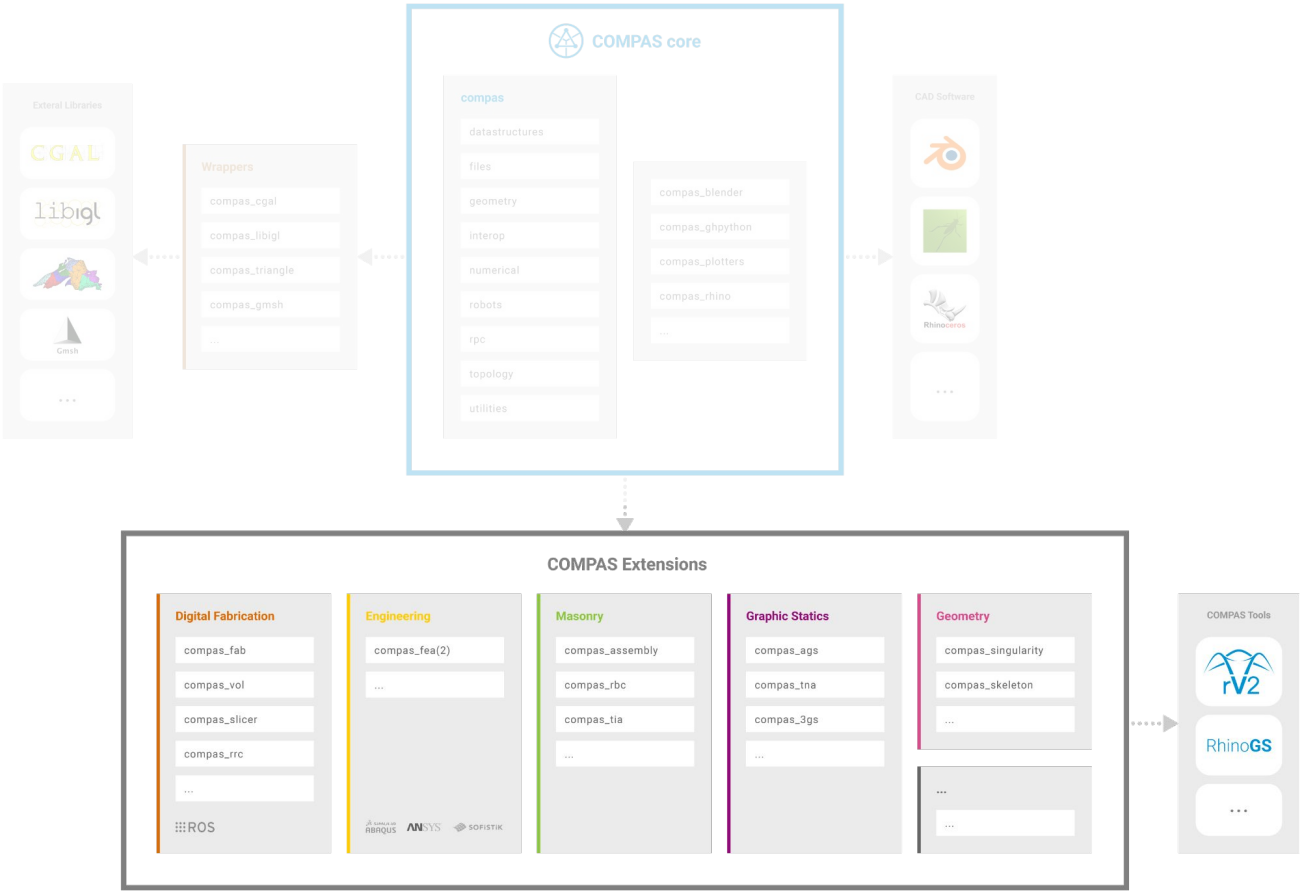
Exchange

compas.data

Tom Van Mele
Gonzalo Casas
Romana Rust
Beverly Lytle
Li Chen


Exteral Libraries
CGAL
libigl
Gmsh
...
Wrappers
compas_cgal
compas_libigl
compas_triangle
compas_gmsh
...
COMPAS core
compas
datastructures
files
geometry
interop
numerical
robots
rpc
topology
utilities
compas_blender
compas_ghpython
compas_plotters
compas_rhino
...
CAD Software
Rhinoceros
...
COMPAS Extensions
Digital Fabrication
compas_fab
compas_vol
compas_slicer
compas_rrc
...
ROS
Engineering
compas_fea(2)
...
ABAQUS
ANSYS
SOFISTIK
Masonry
compas_assembly
compas_rbc
compas_tia
...
Graphic Statics
compas_ags
compas_tna
compas_3gs
...
Geometry
compas_singularity
compas_skeleton
...
...
...
COMPAS Tools
rV2
RhinoGS
...


COMPAS core
Exteral Libraries
CGAL
libigl
Gmsh
...
Wrappers
compas_cgal
compas_libigl
compas_triangle
compas_gmsh
...
compas
datastructures
files
geometry
interop
numerical
robots
rpc
topology
utilities
compas_blender
compas_ghpython
compas_plotters
compas_rhino
...
CAD Software
Rhinoceros
...
COMPAS Extensions
Digital Fabrication
compas_fab
compas_vol
compas_slicer
compas_rrc
...
ROS
Engineering
compas_fea(2)
...
RHINO|RAHUS
ANSYS
SOFISTIK
Masonry
compas_assembly
compas_rbc
compas_tia
...
Graphic Statics
compas_ags
compas_tna
compas_3gs
...
Geometry
compas_singularity
compas_skeleton
...
...
...
COMPAS Tools
rV2
RhinoGS
...


Exteral Libraries
CGAL
libigl
Gmsh
...
Wrappers
compas_cgal
compas_libigl
compas_triangle
compas_gmsh
...
COMPAS core
compas
datastructures
files
geometry
interop
numerical
robots
rpc
topology
utilities
compas_blender
compas_ghpython
compas_plotters
compas_rhino
...
CAD Software
Rhinoceros
...
COMPAS Extensions
Digital Fabrication
compas_fab
compas_vol
compas_slicer
compas_rrc
...
ROS
Engineering
compas_fea(2)
...
ABAQUS
ANSYS
SOFISTIK
Masonry
compas_assembly
compas_rbc
compas_tia
...
Graphic Statics
compas_ags
compas_tna
compas_3gs
...
Geometry
compas_singularity
compas_skeleton
...
...
...
COMPAS Tools
rV2
RhinoGS
...

COMPAS core

- compas.data
- compas.datastructures
- compas.files
- compas.geometry
- compas.numerical
- compas.plugins
- compas.robots
- compas.rpc
- compas.scene
- compas.topology
- compas.utilities
- compas_plotters
- compas_rhino
- compas_ghpython
- compas_blender

COMPAS core extensions

- compas_cgal
- compas_cloud
- compas_dashboard
- compas_gmsh
- compas_libigl
- compas_occ
- compas_triangle
- compas_view2

COMPAS core

- compas.data
- compas.datastructures
- compas.files
- compas.geometry
- compas.numerical
- compas.plugins
- compas.robots
- compas.rpc
- compas.scene
- compas.topology
- compas.utilities
- compas_plotters
- compas_rhino
- compas_ghpython
- compas_blender

COMPAS core extensions

- compas_cgal
- compas_cloud
- compas_dashboard
- compas_gmsh
- compas_libigl
- compas_occ
- compas_triangle
- compas_view2

Data modelling & Assessment

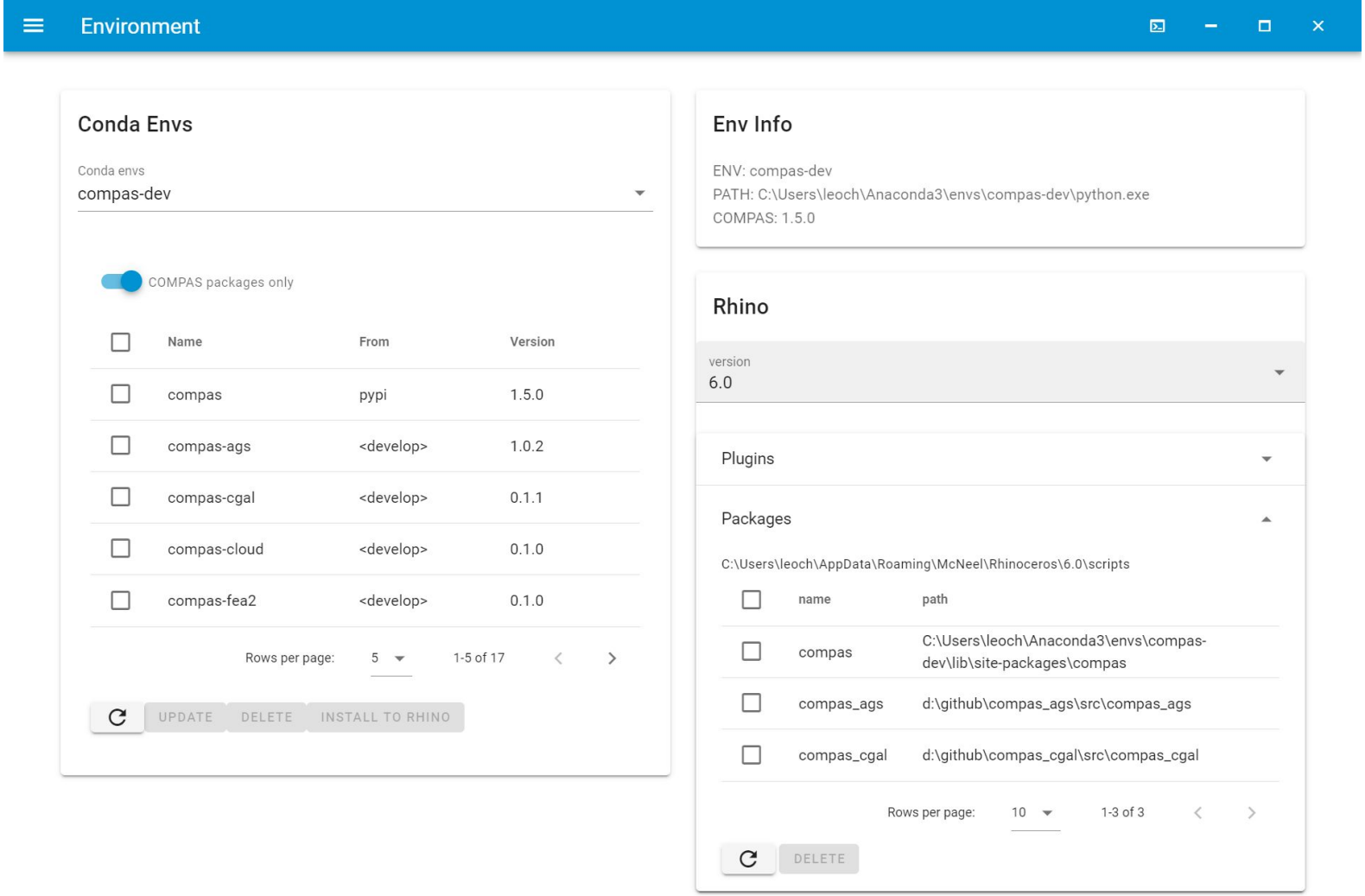
- compas_ifc
- compas_lca

Digital Fabrication

- compas_fab
- compas_slicer
- compas_rrc
- compas_vol
- compas_timber

Engineering

- compas_fea(2)


Environment
Conda Envs
Conda envs
compas-dev
COMPAS packages only
Name
From
Version
compas
pypi
1.5.0
compas-ags
<develop>
1.0.2
compas-cgal
<develop>
0.1.1
compas-cloud
<develop>
0.1.0
compas-fea2
<develop>
0.1.0
Rows per page:
5
1-5 of 17
UPDATE
DELETE
INSTALL TO RHINO
Env Info
ENV: compas-dev
PATH: C:\Users\leoch\Anaconda3\envs\compas-dev\python.exe
COMPAS: 1.5.0
Rhino
version
6.0
Plugins
Packages
C:\Users\leoch\AppData\Roaming\McNeel\Rhinoceros\6.0\scripts
name
path
compas
C:\Users\leoch\Anaconda3\envs\compas-dev\lib\site-packages\compas
compas_ags
d:\github\compas_ags\src\compas_ags
compas_cgal
d:\github\compas_cgal\src\compas_cgal
Rows per page:
10
1-3 of 3
DELETE

 # code.py

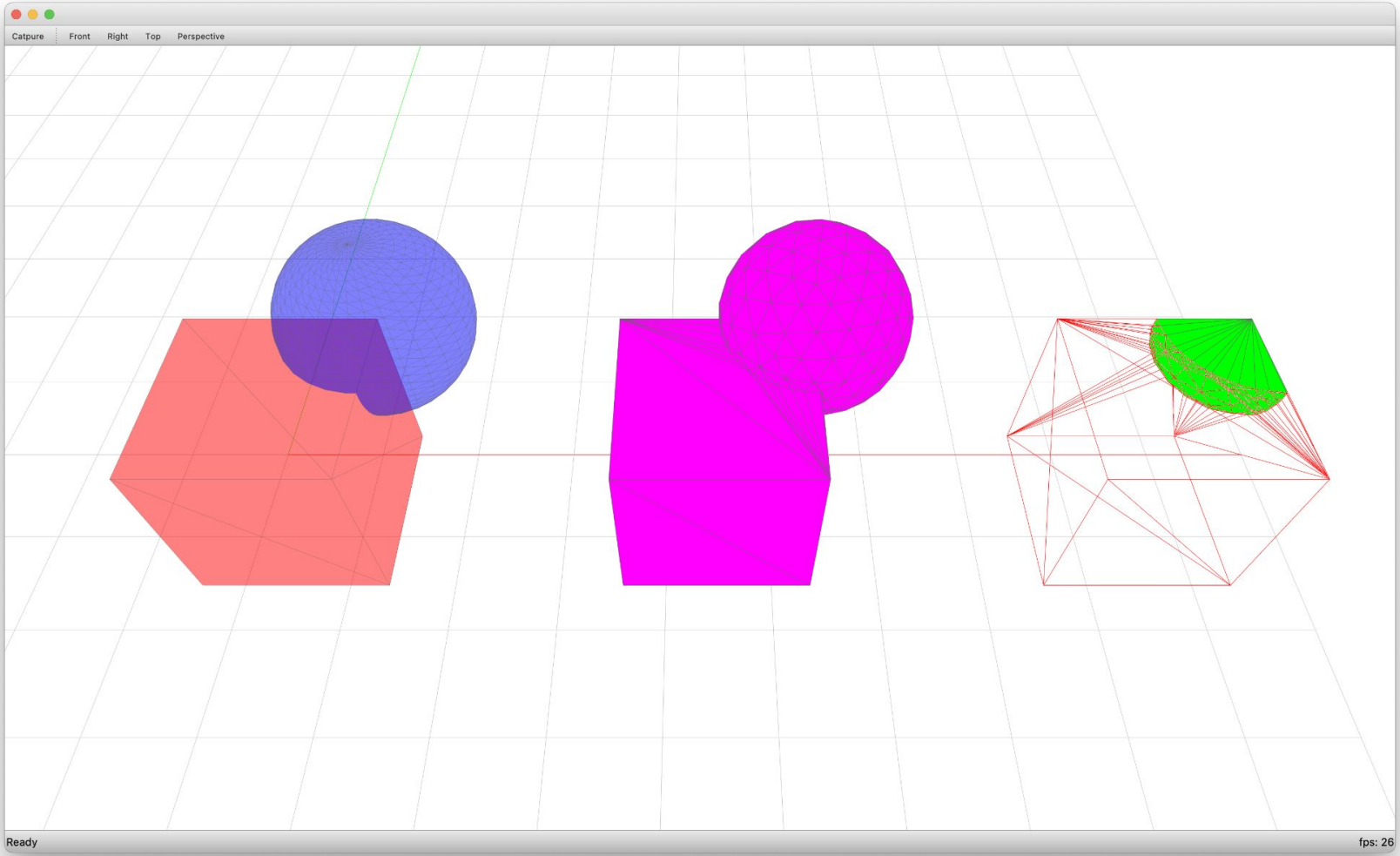
```python
class ROSRobot(component):
    def RunScript(self, ros, load):
        key = create_id(self, 'robot')

        if ros.is_connected and load:
            st[key] = ros.load_robot(True)
            artist = RobotModelArtist(st[key].model)
            st[key].artist = artist

        return st.get(key, None)
```

 # metadata.json

```json
{
    "name": "ROS Robot",
    "category": "COMPAS FAB",
    "subcategory": "ROS",
    "description": "Load robot directly from ROS.",
    "exposure": 2,

    "ghpython": {
        "isAdvancedMode": true,
        "inputParameters": [
            {
                "name": "ros",
            },
            {
                "name": "load",
                "typeHintID": "bool"
            }
        ],
        "outputParameters": [
            {
                "name": "robot",
                "description": "The robot."
            }
        ]
    }
}
```

```python
# code.py

class ROSRobot(component):
    def RunScript(self, ros, load):
        key = create_id(self, 'robot')

        if ros.is_connected and load:
            st[key] = ros.load_robot(True)
            artist = RobotModelArtist(st[key].model)
            st[key].artist = artist

        return st.get(key, None)
```


```json
# metadata.json

{
    "name": "ROS Robot",
    "category": "COMPAS FAB",
    "subcategory": "ROS",
    "description": "Load robot directly from ROS.",
    "exposure": 2,

    "ghpython": {
        "isAdvancedMode": true,
        "inputParameters": [
            {
                "name": "ros",
            },
            {
                "name": "load",
                "typeHintID": "bool"
            }
        ],
        "outputParameters": [
            {
                "name": "robot",
                "description": "The robot."
            }
        ]
    }
}
```

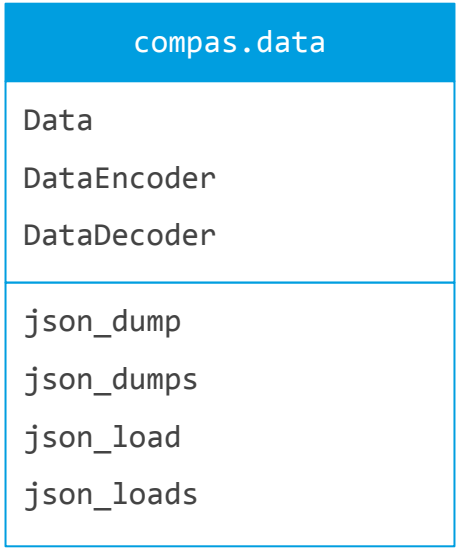

Catpure
Front
Right
Top
Perspective
Ready
fps: 26

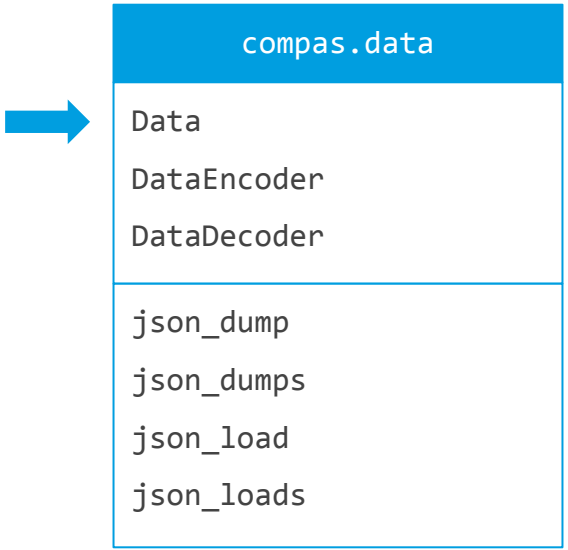
compas.data

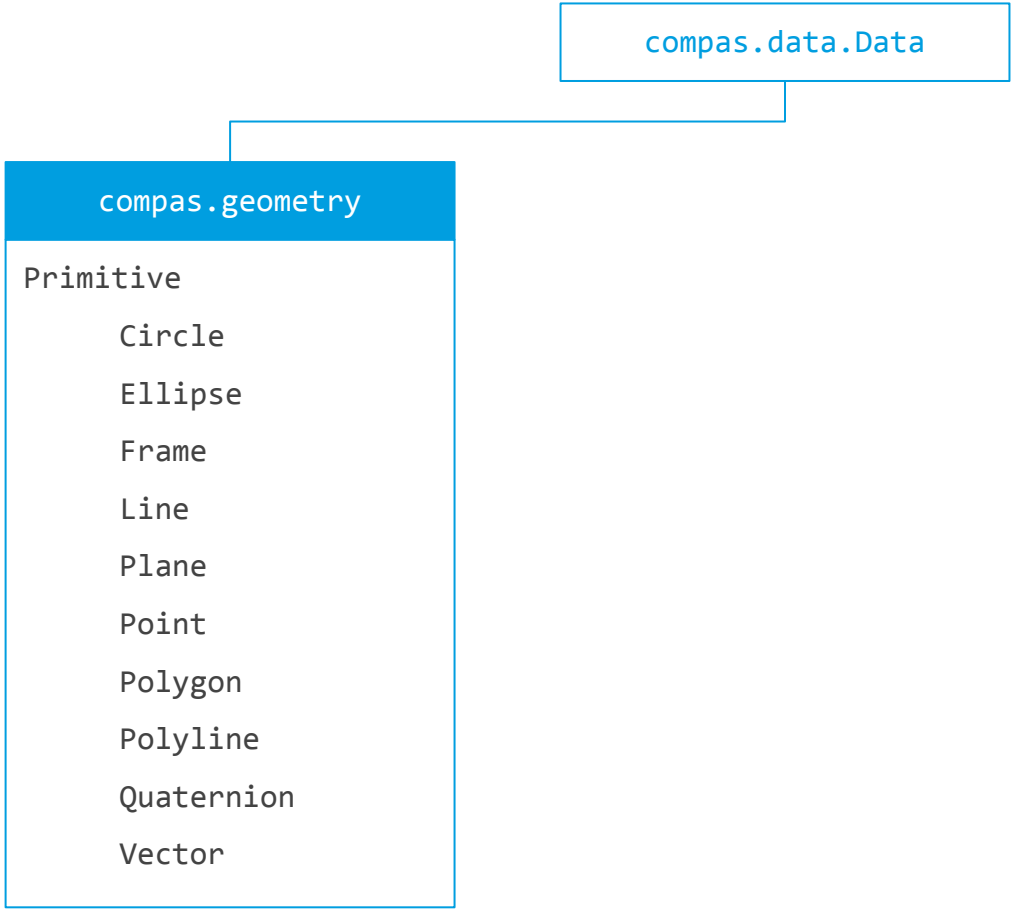
Reproducible Research is work that **can be independently recreated** from the **same data** and the **same code** that the original team used.

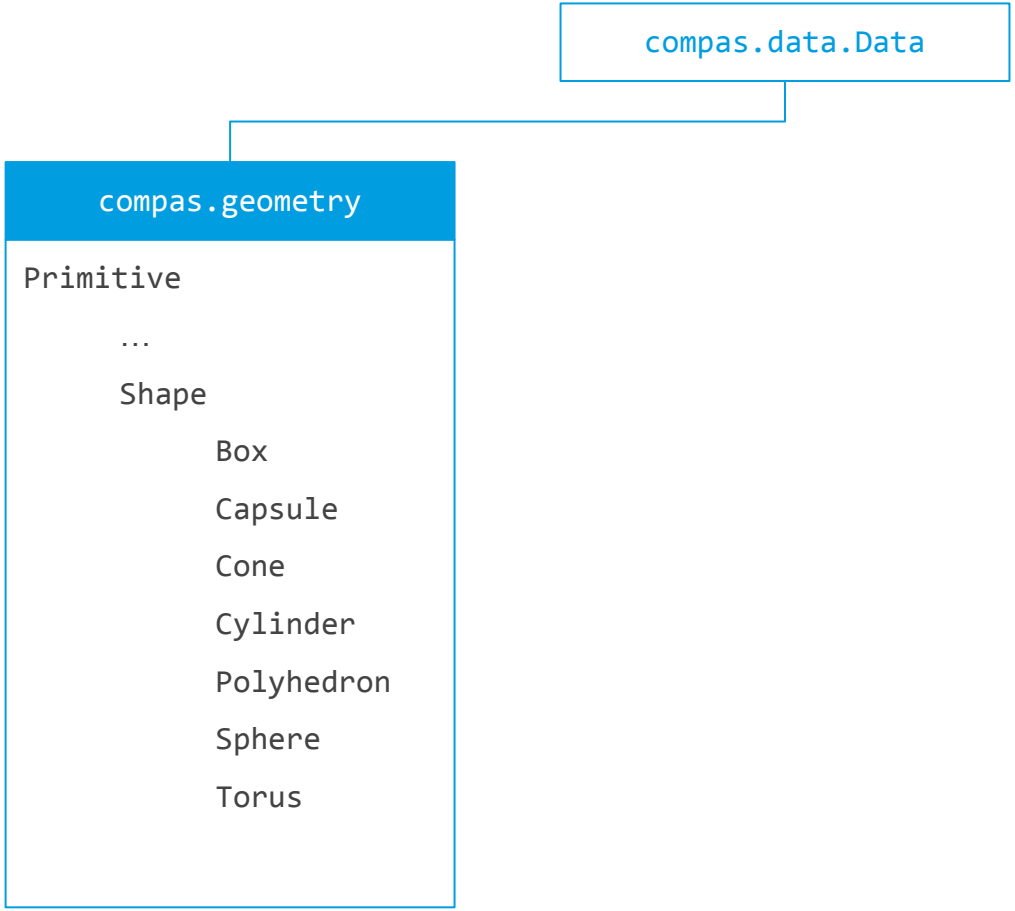
The Turing Way, https://github.com/alan-turing-institute/the-turing-way

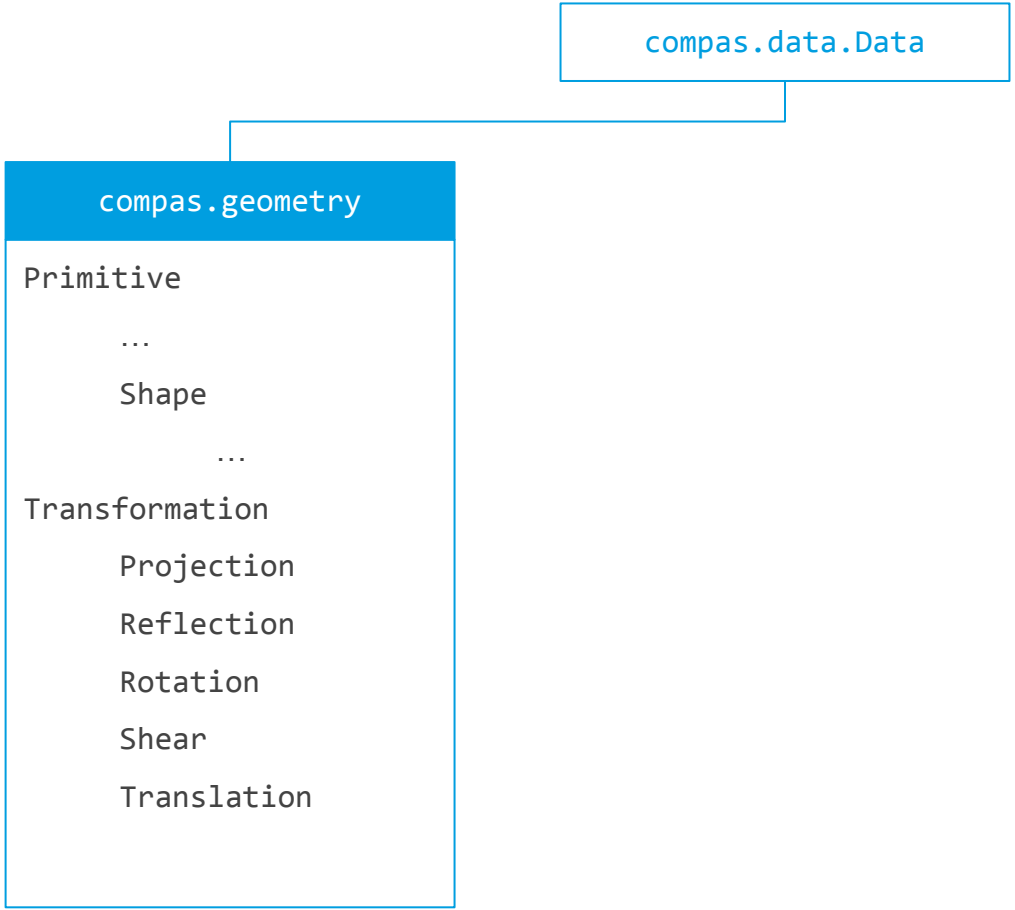
The Turing Way project illustration by Scriberia. Original version on Zenodo. http://doi.org/10.5281/zenodo.3695300

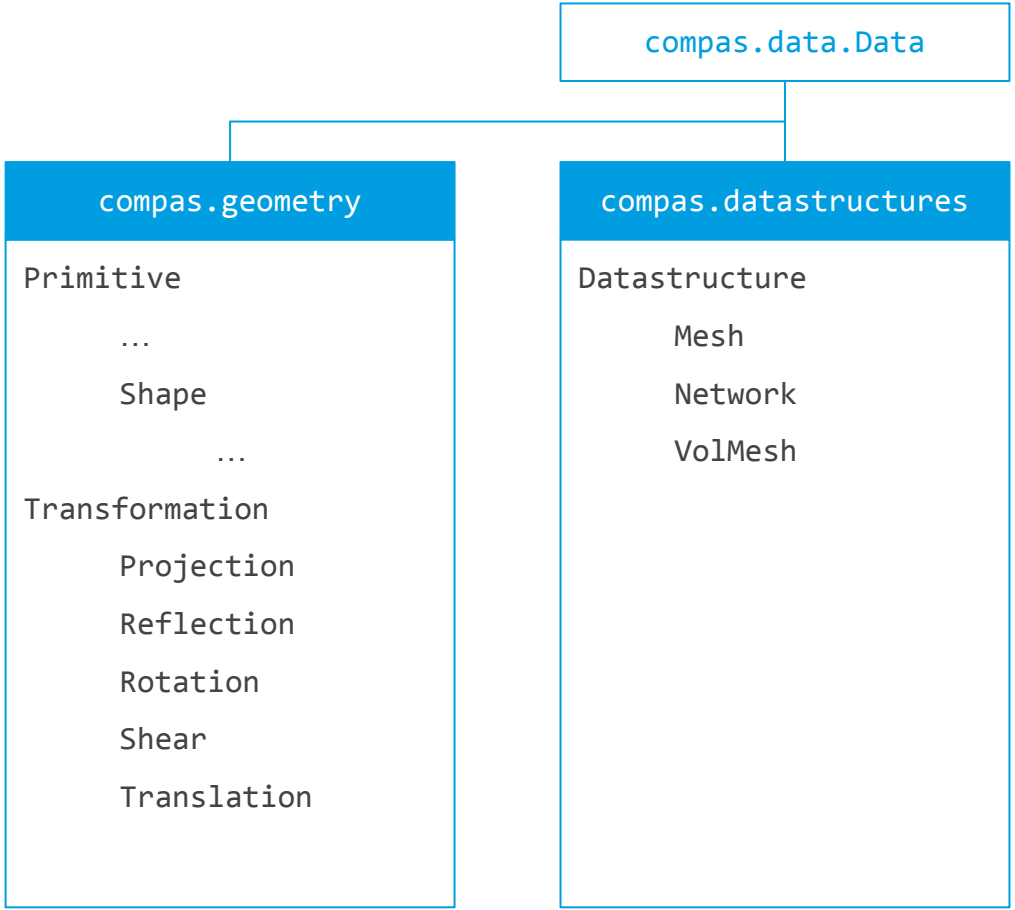

compas.data
Data
DataEncoder
DataDecoder
json_dump
json_dumps
json_load
json_loads

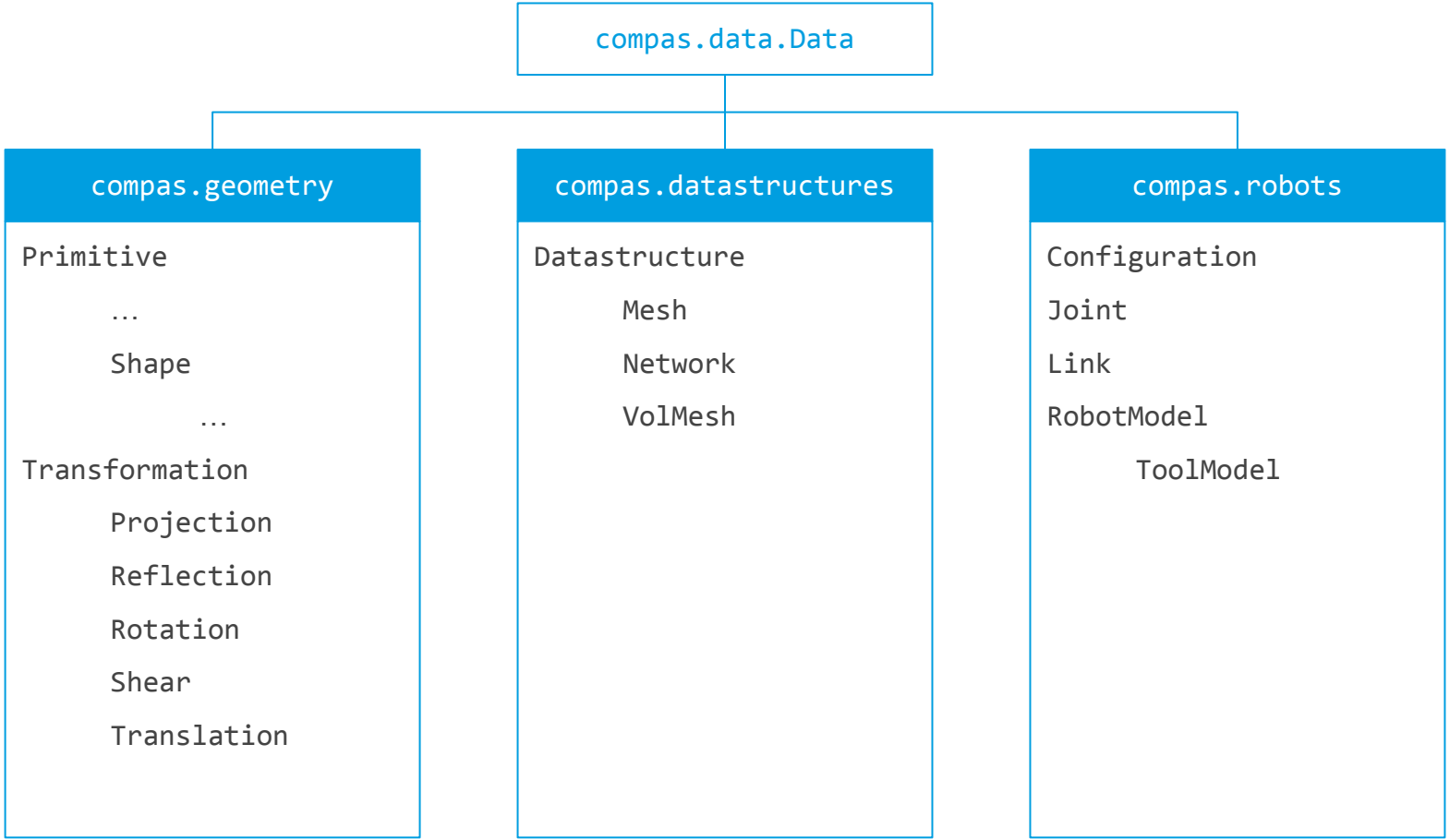

compas.data
Data
DataEncoder
DataDecoder
json_dump
json_dumps
json_load
json_loads


compas.data.Data
compas.geometry
Primitive
Circle
Ellipse
Frame
Line
Plane
Point
Polygon
Polyline
Quaternion
Vector


compas.data.Data
compas.geometry
Primitive
…
Shape
Box
Capsule
Cone
Cylinder
Polyhedron
Sphere
Torus


compas.data.Data
compas.geometry
Primitive
…
Shape
…
Transformation
Projection
Reflection
Rotation
Shear
Translation


compas.data.Data
compas.geometry
Primitive
…
Shape
…
Transformation
Projection
Reflection
Rotation
Shear
Translation
compas.datastructures
Datastructure
Mesh
Network
VolMesh


compas.data.Data
compas.geometry
Primitive
...
Shape
...
Transformation
Projection
Reflection
Rotation
Shear
Translation
compas.datastructures
Datastructure
Mesh
Network
VolMesh
compas.robots
Configuration
Joint
Link
RobotModel
ToolModel

compas.data.Data			
Attributes	Magic methods	Class methods	Methods
DATASCHEMA	__str__	from_data	to_data
JSONSCHEMA	__getstate__	from_json	to_json
data	__setstate__	from_jsonstring	to_jsonstring
dtype (read-only)			copy
guid (read-only)			validate_data
name			validate_json

```python
from compas.data import Data

from compas.geometry import Point
from compas.geometry import Box
from compas.geometry import Rotation
from compas.datastructures import Mesh
from compas.robots import RobotModel

print(issubclass(Point, Data))
print(issubclass(Box, Data))
print(issubclass(Rotation, Data))
print(issubclass(Mesh, Data))
print(issubclass(RobotModel, Data))
```

```
True
True
True
True
True
```

```python
from compas.geometry import Point

point = Point(0, 0, 0)

print(point.name)
print(point.guid)

print(point.dtype)
print(point.data)

print(point.to_jsonstring())
print(point.to_json("point.json"))
```

```
Point
d0f9f661-eccb-48eb-8425-2cd1433cda5b

'compas.geometry/Point'
[0.0, 0.0, 0.0]

'[0.0, 0.0, 0.0]'
None
```

```python
from compas.geometry import Frame

frame = Frame([0, 0, 0], [1, 0, 0], [0, 1, 0])

print(frame.name)
print(frame.guid)

print(frame.dtype)
print(frame.data)

print(frame.to_jsonstring())
print(frame.to_json("frame.json"))
```

```
Frame
18992be6-f1c7-4f90-9ca3-d52d15674fa8

'compas.geometry/Frame'
{'point': [...], 'xaxis': [...], 'yaxis': [...]}

{"point": [...], "xaxis": [...], "yaxis": [...]}
None
```

```python
from compas.geometry import Box

box = Box.from_width_height_depth(1, 1, 1)

print(box.name)
print(box.guid)

print(box.dtype)
print(box.data)

print(box.to_jsonstring())
print(box.to_json("box.json"))
```

```
Box
379a0f4d-2497-4571-b030-e312564ee8b5

'compas.geometry/Box'
{'frame': {'point': [...], 'xaxis': [...], 'yaxis':
[...]}, 'xsize': 1.0, 'ysize': 1.0, 'zsize': 1.0}

{"frame": {"point": [...], "xaxis": [...], "yaxis":
[...]}, "xsize": 1.0, "ysize": 1.0, "zsize": 1.0}
None
```

```python
print(box.to_jsonstring(pretty=True))
```

```
{
    "frame": {
        "point": [
            0.0,
            0.0,
            0.0
        ],
        "xaxis": [
            1.0,
            0.0,
            0.0
        ],
        "yaxis": [
            0.0,
            1.0,
            0.0
        ]
    },
    "xsize": 1.0,
    "ysize": 1.0,
    "zsize": 1.0
}
```

```
print(box)
```


```
{
    "frame": {
        "point": [
            0.0,
            0.0,
            0.0
        ],
        "xaxis": [
            1.0,
            0.0,
            0.0
        ],
        "yaxis": [
            0.0,
            1.0,
            0.0
        ]
    },
    "xsize": 1.0,
    "ysize": 1.0,
    "zsize": 1.0
}
```

```
>>> box
```

```
Box(Frame(Point(0.000, 0.000, 0.000),Vector(1.000,
0.000, 0.000),Vector(0.000, 1.000, 0.000)),1.0, 1.0,
1.0)
```

```
>>> box
```

```
Box(
    Frame(
        Point(0.000, 0.000, 0.000),
        Vector(1.000, 0.000, 0.000),
        Vector(0.000, 1.000, 0.000)
    ),
    1.0, 1.0, 1.0
)
```

```
print(repr(box))
```

```
Box(
    Frame(
        Point(0.000, 0.000, 0.000),
        Vector(1.000, 0.000, 0.000),
        Vector(0.000, 1.000, 0.000)
    ),
    1.0, 1.0, 1.0
)
```

```python
from compas.datastructures import Mesh

mesh = Mesh()

print(mesh.name)
print(mesh.guid)

print(point.dtype)
print(point.data)

print(mesh.to_jsonstring())
print(mesh.to_json("mesh.json"))
```

```
Mesh
7e04920d-aaca-44db-b34b-451688472f9f

'compas.datastructures/Mesh'
{'compas': '1.6.2', 'datatype': ..., 'data': ... }

{"compas": "1.6.2", "datatype": ..., "data": ... }
None
```

```python
from compas.robots import RobotModel

ur5 = RobotModel('ur5')

print(ur5.name)
print(ur5.guid)

print(point.dtype)
print(point.data)

print(ur5.to_jsonstring())
print(ur5.to_json('ur5.json'))
```

```
ur5
1859fe24-a46a-427c-bcb6-ff6b6a92a0c1

'compas.robots/RobotModel'
{'name': 'ur5', 'joints': ...}

{"name": "ur5", "joints": ...}
None
```

copying

```python
from compas.geometry import Point

point = Point(0, 0, 0)

other = Point.from_data(point.data)

print(point == other)
print(point is other)
```

```
True
False
```

```python
from compas.geometry import Point

point = Point(0, 0, 0)

other = Point.from_jsonstring(point.to_jsonstring())

print(point == other)
print(point is other)
```

```
True
False
```

```python
from compas.geometry import Point

point = Point(0, 0, 0)

other = point.copy()

print(point == other)
print(point is other)
```

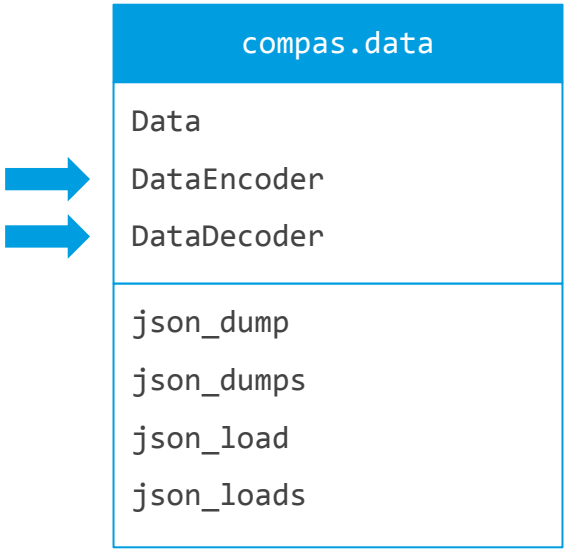
```
True
False
```

```python
import json
from compas.geometry import Point, Frame
from compas.datastructures import Mesh

p1 = Point(0, 0, 0)
f1 = Frame.worldXY()
m1 = Mesh()

data1 = [p1.data, f1.data, m1.data]
string = json.dumps(data1)
data2 = json.loads(string)

p2 = Point.from_data(data2[0])
f2 = Frame.from_data(data2[1])
m2 = Mesh.from_data(data2[2])
```

```python
import json
from compas.geometry import Point, Frame
from compas.datastructures import Mesh

p1 = Point(0, 0, 0)
f1 = Frame.worldXY()
m1 = Mesh()

data1 = [p1.data, f1.data, m1.data]
string = json.dumps(data1)
data2 = json.loads(string)

p2 = Point.from_data(data2[0])
f2 = Frame.from_data(data2[1])
m2 = Mesh.from_data(data2[2])
```

```python
import json
from compas.geometry import Point, Frame
from compas.datastructures import Mesh

p1 = Point(0, 0, 0)
f1 = Frame.worldXY()
m1 = Mesh()

data1 = [p1.data, f1.data, m1.data]
string = json.dumps(data1)
data2 = json.loads(string)

p2 = Point.from_data(data2[0])
f2 = Frame.from_data(data2[1])
m2 = Mesh.from_data(data2[2])
```

```python
import json
from compas.geometry import Point, Frame
from compas.datastructures import Mesh

p1 = Point(0, 0, 0)
f1 = Frame.worldXY()
m1 = Mesh()

data1 = [p1.data, f1.data, m1.data]
string = json.dumps(data1)
data2 = json.loads(string)

p2 = Point.from_data(data2[0])
f2 = Frame.from_data(data2[1])
m2 = Mesh.from_data(data2[2])
```


compas.data
Data
DataEncoder
DataDecoder
json_dump
json_dumps
json_load
json_loads

```python
import json
from compas.geometry import Point, Frame
from compas.datastructures import Mesh


p1 = Point(0, 0, 0)
f1 = Frame.worldXY()
m1 = Mesh()

data1 = [p1.data, f1.data, m1.data]
string = json.dumps(data1)
data2 = json.loads(string)

p2 = Point.from_data(data2[0])
f2 = Frame.from_data(data2[1])
m2 = Mesh.from_data(data2[2])
```

```python
import json
from compas.geometry import Point, Frame
from compas.datastructures import Mesh
from compas.data import DataEncoder, DataDecoder

p1 = Point(0, 0, 0)
f1 = Frame.worldXY()
m1 = Mesh()

data = [p1, f1, m1]
string = json.dumps(data, cls=DataEncoder)


p2, f2, m2 = json.loads(string, cls=DataDecoder)
```

```python
import json
from compas.geometry import Point, Frame
from compas.datastructures import Mesh

p1 = Point(0, 0, 0)
f1 = Frame.worldXY()
m1 = Mesh()

data1 = [p1.data, f1.data, m1.data]
string = json.dumps(data1)
data2 = json.loads(string)

p2 = Point.from_data(data2[0])
f2 = Frame.from_data(data2[1])
m2 = Mesh.from_data(data2[2])
```

```python
import json
from compas.geometry import Point, Frame
from compas.datastructures import Mesh
from compas.data import DataEncoder, DataDecoder

p1 = Point(0, 0, 0)
f1 = Frame.worldXY()
m1 = Mesh()

data = [p1, f1, m1]
string = json.dumps(data, cls=DataEncoder)

p2, f2, m2 = json.loads(string, cls=DataDecoder)
```

```python
import json
from compas.geometry import Point, Frame
from compas.datastructures import Mesh

p1 = Point(0, 0, 0)
f1 = Frame.worldXY()
m1 = Mesh()

data1 = [p1.data, f1.data, m1.data]
string = json.dumps(data1)
data2 = json.loads(string)

p2 = Point.from_data(data2[0])
f2 = Frame.from_data(data2[1])
m2 = Mesh.from_data(data2[2])
```

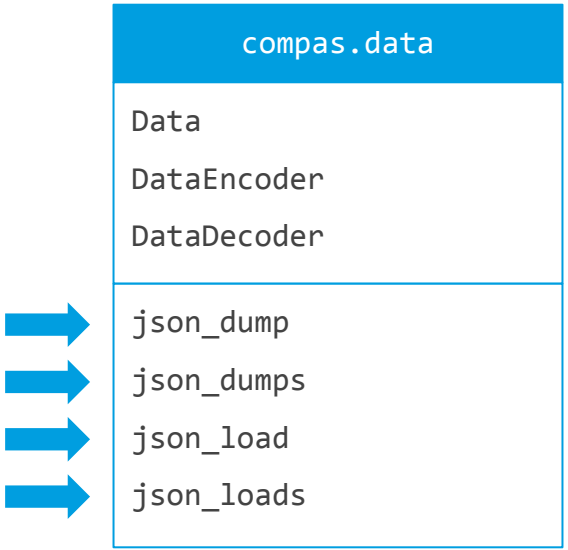
```python
import json
from compas.geometry import Point, Frame
from compas.datastructures import Mesh
from compas.data import DataEncoder, DataDecoder

p1 = Point(0, 0, 0)
f1 = Frame.worldXY()
m1 = Mesh()

data = [p1, f1, m1]
string = json.dumps(data, cls=DataEncoder)

p2, f2, m2 = json.loads(string, cls=DataDecoder)
```

```python
import json
from compas.geometry import Point, Frame
from compas.datastructures import Mesh


p1 = Point(0, 0, 0)
f1 = Frame.worldXY()
m1 = Mesh()

data1 = [p1.data, f1.data, m1.data]
string = json.dumps(data1)
data2 = json.loads(string)

p2 = Point.from_data(data2[0])
f2 = Frame.from_data(data2[1])
m2 = Mesh.from_data(data2[2])
```

```python
import json
from compas.geometry import Point, Frame
from compas.datastructures import Mesh
from compas.data import DataEncoder, DataDecoder

p1 = Point(0, 0, 0)
f1 = Frame.worldXY()
m1 = Mesh()

data = [p1, f1, m1]
string = json.dumps(data, cls=DataEncoder)

p2, f2, m2 = json.loads(string, cls=DataDecoder)
```

```python
import json
from compas.geometry import Point, Frame
from compas.datastructures import Mesh


p1 = Point(0, 0, 0)
f1 = Frame.worldXY()
m1 = Mesh()

data1 = [p1.data, f1.data, m1.data]
string = json.dumps(data1)
data2 = json.loads(string)

p2 = Point.from_data(data2[0])
f2 = Frame.from_data(data2[1])
m2 = Mesh.from_data(data2[2])
```

```python
import json
from compas.geometry import Point, Frame
from compas.datastructures import Mesh
from compas.data import DataEncoder, DataDecoder

p1 = Point(0, 0, 0)
f1 = Frame.worldXY()
m1 = Mesh()

data = [p1, f1, m1]
string = json.dumps(data, cls=DataEncoder)

p2, f2, m2 = json.loads(string, cls=DataDecoder)
```


compas.data
Data
DataEncoder
DataDecoder
json_dump
json_dumps
json_load
json_loads

```python
import json
from compas.geometry import Point, Frame
from compas.datastructures import Mesh
from compas.data import DataEncoder, DataDecoder

p1 = Point(0, 0, 0)
f1 = Frame.worldXY()
m1 = Mesh()

data = [p1, f1, m1]
string = json.dumps(data, cls=DataEncoder)

p2, f2, m2 = json.loads(string, cls=DataDecoder)
```

```python
from compas.geometry import Point, Frame
from compas.datastructures import Mesh
from compas.data import json_dumps, json_loads

p1 = Point(0, 0, 0)
f1 = Frame.worldXY()
m1 = Mesh()

data = [p1, f1, m1]
string = json_dumps(data)

p2, f2, m2 = json_loads(string)
```

```python
import json
from compas.geometry import Point, Frame
from compas.datastructures import Mesh
from compas.data import DataEncoder, DataDecoder

p1 = Point(0, 0, 0)
f1 = Frame.worldXY()
m1 = Mesh()

data = [p1, f1, m1]
string = json.dumps(data, cls=DataEncoder)

p2, f2, m2 = json.loads(string, cls=DataDecoder)
```

```python
from compas.geometry import Point, Frame
from compas.datastructures import Mesh
from compas.data import json_dumps, json_loads

p1 = Point(0, 0, 0)
f1 = Frame.worldXY()
m1 = Mesh()

data = [p1, f1, m1]
string = json_dumps(data)

p2, f2, m2 = json_loads(string)
```

```python
import json
from compas.geometry import Point, Frame
from compas.datastructures import Mesh
from compas.data import DataEncoder, DataDecoder

p1 = Point(0, 0, 0)
f1 = Frame.worldXY()
m1 = Mesh()

data = [p1, f1, m1]
string = json.dumps(data, cls=DataEncoder)

p2, f2, m2 = json.loads(string, cls=DataDecoder)
```

```python
from compas.geometry import Point, Frame
from compas.datastructures import Mesh
from compas.data import json_dumps, json_loads

p1 = Point(0, 0, 0)
f1 = Frame.worldXY()
m1 = Mesh()

data = [p1, f1, m1]
string = json_dumps(data)

p2, f2, m2 = json_loads(string)
```

```python
from compas.data import json_dumps, json_loads
from compas.geometry import Point, Frame
from compas.datastructures import Mesh

p1 = Point(0, 0, 0)
f1 = Frame.worldXY()
m1 = Mesh()

data = [p1, f1, m1]
string = json_dumps(data)

p2, f2, m2 = json_loads(string)
```

```python
import compas
from compas.geometry import Point, Frame
from compas.datastructures import Mesh

p1 = Point(0, 0, 0)
f1 = Frame.worldXY()
m1 = Mesh()

data = [p1, f1, m1]
string = compas.json_dumps(data)

p2, f2, m2 = compas.json_loads(string)
```

```python
from compas.data import json_dumps, json_loads
from compas.geometry import Point, Frame
from compas.datastructures import Mesh

p1 = Point(0, 0, 0)
f1 = Frame.worldXY()
m1 = Mesh()

data = [p1, f1, m1]
string = json_dumps(data)

p2, f2, m2 = json_loads(string)
```

```python
import compas
from compas.geometry import Point, Frame
from compas.datastructures import Mesh

p1 = Point(0, 0, 0)
f1 = Frame.worldXY()
m1 = Mesh()

data = [p1, f1, m1]
string = compas.json_dumps(data)

p2, f2, m2 = compas.json_loads(string)
```

```python
from compas.data import json_dumps, json_loads
from compas.geometry import Point, Frame
from compas.datastructures import Mesh

p1 = Point(0, 0, 0)
f1 = Frame.worldXY()
m1 = Mesh()

data = [p1, f1, m1]
string = json_dumps(data)

p2, f2, m2 = json_loads(string)
```

```python
import compas
from compas.geometry import Point, Frame
from compas.datastructures import Mesh

p1 = Point(0, 0, 0)
f1 = Frame.worldXY()
m1 = Mesh()

data = [p1, f1, m1]
string = compas.json_dumps(data)

p2, f2, m2 = compas.json_loads(string)
```

```python
import compas
from compas.geometry import Point
from compas.datastructures import Network

p1 = Point(0, 0, 0)

network = Network()

a = network.add_node(point=p1)

string = compas.json_dumps(network)
network = compas.json_loads(string)

p2 = network.node_attribute(a, 'point')

print(repr(p1))
print(repr(p2))
print(p1 == p2)
```

```
import compas
from compas.geometry import Point
from compas.datastructures import Network

p1 = Point(0, 0, 0)

network = Network()

a = network.add_node(point=p1)

string = compas.json_dumps(network)
network = compas.json_loads(string)

p2 = network.node_attribute(a, 'point')

print(repr(p1))
print(repr(p2))
print(p1 == p2)
```

```python
import compas
from compas.geometry import Point
from compas.datastructures import Network

p1 = Point(0, 0, 0)

network = Network()

a = network.add_node(point=p1)

string = compas.json_dumps(network)
network = compas.json_loads(string)

p2 = network.node_attribute(a, 'point')

print(repr(p1))
print(repr(p2))
print(p1 == p2)
```

```python
import compas
from compas.geometry import Point
from compas.datastructures import Network

p1 = Point(0, 0, 0)

network = Network()

a = network.add_node(point=p1)

string = compas.json_dumps(network)
network = compas.json_loads(string)

p2 = network.node_attribute(a, 'point')

print(repr(p1))
print(repr(p2))
print(p1 == p2)
```

```python
import compas
from compas.geometry import Point
from compas.datastructures import Network

p1 = Point(0, 0, 0)

network = Network()

a = network.add_node(point=p1)

string = compas.json_dumps(network)
network = compas.json_loads(string)

p2 = network.node_attribute(a, 'point')

print(repr(p1))
print(repr(p2))
print(p1 == p2)
```

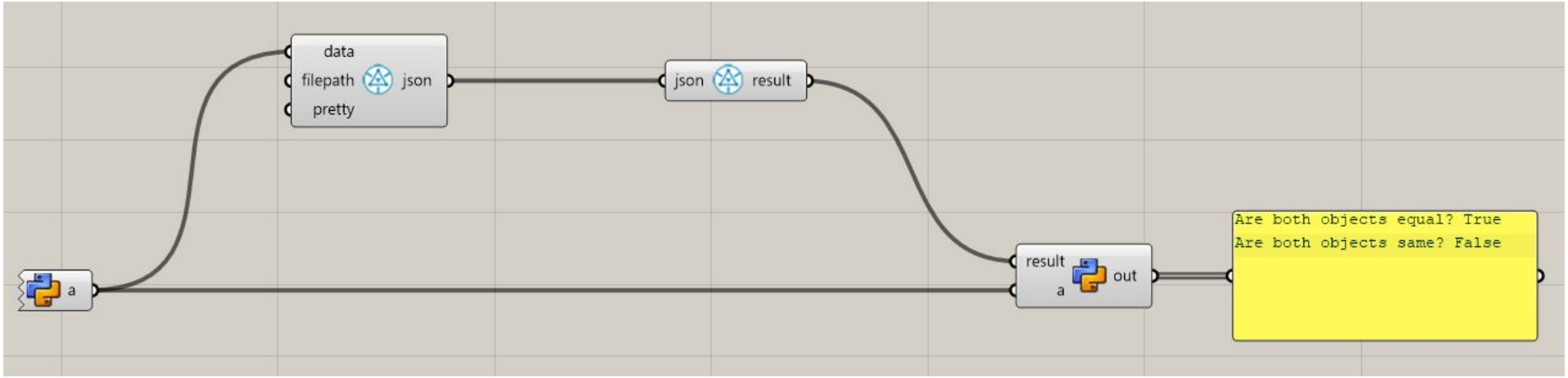
```python
import compas
from compas.geometry import Point
from compas.datastructures import Network

p1 = Point(0, 0, 0)

network = Network()

a = network.add_node(point=p1)

string = compas.json_dumps(network)
network = compas.json_loads(string)

p2 = network.node_attribute(a, 'point')

print(repr(p1))
print(repr(p2))
print(p1 == p2)
```

```
Point(0.000, 0.000, 0.000)
Point(0.000, 0.000, 0.000)
True
```

compas_ghpython.components

ToJson

FromJson


data
filepath
pretty
json
json
result
a
result
a
out
Are both objects equal? True
Are both objects same? False

storage

```python
# script A

import compas
from compas.datastructures import Mesh

mesh = Mesh()

compas.json_dump(mesh, "mesh.json")
```

```python
# script B

import compas

mesh = compas.json_load("mesh.json")
```

```python
# script A

import compas
from compas.geometry import Point, Frame
from compas.datastructures import Mesh

point = Point(0, 0, 0)
frame = Frame.worldXY()
mesh  = Mesh()

data = [point, frame, mesh]

compas.json_dump(data, "session.json")
```

```python
# script B

import compas

data = compas.json_load("session.json")

point = data[0]
frame = data[1]
mesh  = data[2]
```

validation (experimental)

```
Schema(compas.data.is_float3)
```

```
{
   "$schema": "http://json-schema.org/draft-07/schema#",
   "$id": "gh:compas-dev/compas/schemas/point.json",
   "$compas": "1.6.2",

   "type": "array",
   "minItems": 3,
   "maxItems": 3,
   "items": {"type": "number"}
}
```

```
Schema({
    "point": is_float3,
    "xaxis": is_float3,
    "yaxis": is_float3
})
```

```
{
   "$schema": "http://json-schema.org/draft-07/schema#",
   "$id": "gh:compas-dev/compas/schemas/point.json",
   "$compas": "1.6.2",

   "type": "array",
   "minItems": 3,
   "maxItems": 3,
   "items": {"type": "number"}
}
```

```
Schema({
    "matrix": compas.data.is_float4x4
})
```

```
{
   "$schema": "http://json-schema.org/draft-07/schema#",
   "$id": "gh:compas-dev/compas/schemas/point.json",
   "$compas": "1.6.2",

   "type": "array",
   "minItems": 3,
   "maxItems": 3,
   "items": {"type": "number"}
}
```

```
Schema({
    "compas": str,
    "datatype": str,
    "data": {
        "attributes": dict,
        "dva": dict,
        "dea": dict,
        "dfa": dict,
        "vertex": And(
            dict,
            lambda x: all(isinstance(key, int) for key in x)
        ),
        "face": And(
            dict,
            lambda x: all(isinstance(key, int) for key in x),
            lambda x: all(all(isinstance(item, int) for item in value) for value in x.values())
        ),
        "facedata": And(
            dict,
            lambda x: all(isinstance(key, int) for key in x),
            lambda x: all(isinstance(value, dict) for value in x.values())
        ),
        "edgedata": And(
            dict,
            lambda x: all(isinstance(key, str) for key in x),
            lambda x: all(isinstance(value, dict) for value in x.values())
        ),
        "max_vertex": schema.And(int, lambda x: x >= -1),
        "max_face": schema.And(int, lambda x: x >= -1)
    }
})
```

```
{
   "$schema": "http://json-schema.org/draft-07/schema#",
   "$id": "https://github.com/compas-dev/compas/schemas/halfedge.json",
   "$compas": "1.6.2",

   "type": "object",
   "poperties": {
       "compas": {"type": "string"},
       "datatype": {"type": "string"},
       "data": {
           "type": "object",
           "properties": {
               "attributes":   {"type": "object"},
               "dva":          {"type": "object"},
               "dea":          {"type": "object"},
               "dfa":          {"type": "object"},
               "vertex":       {"type": "object"},
               "face":         {"type": "object"},
               "facedata":     {"type": "object"},
               "edgedata":     {"type": "object"},
               "max_int_key":  {"type": "number"},
               "max_int_fkey": {"type": "number"}
           },
           "required": ["attributes", "dva", "dea", "dfa", "vertex", "face", "facedata", "edgedata", "max_int_key", "max_int_fkey"]
       }
   },
   "required": ["compas", "datatype", "data"]
}
```

```python
from compas.geometry import Point

point = Point(0, 0, 0)

print(point.validate_data())
print(point.validate_json())
```

```
[0.0, 0.0, 0.0]
'[0.0, 0.0, 0.0]'
```

```python
from compas.datastructures import Mesh

mesh = Mesh()

print(mesh.validate_data())
print(mesh.validate_json())
```

```
{'compas': ..., 'datatype': ..., 'data': ...}
{"compas": ..., "datatype": ..., "data": ...}
```

```python
from compas.data import validate_data
from compas.datastructures import Mesh

data = validate_data(..., cls=Mesh)
mesh = Mesh.from_data(data)
```

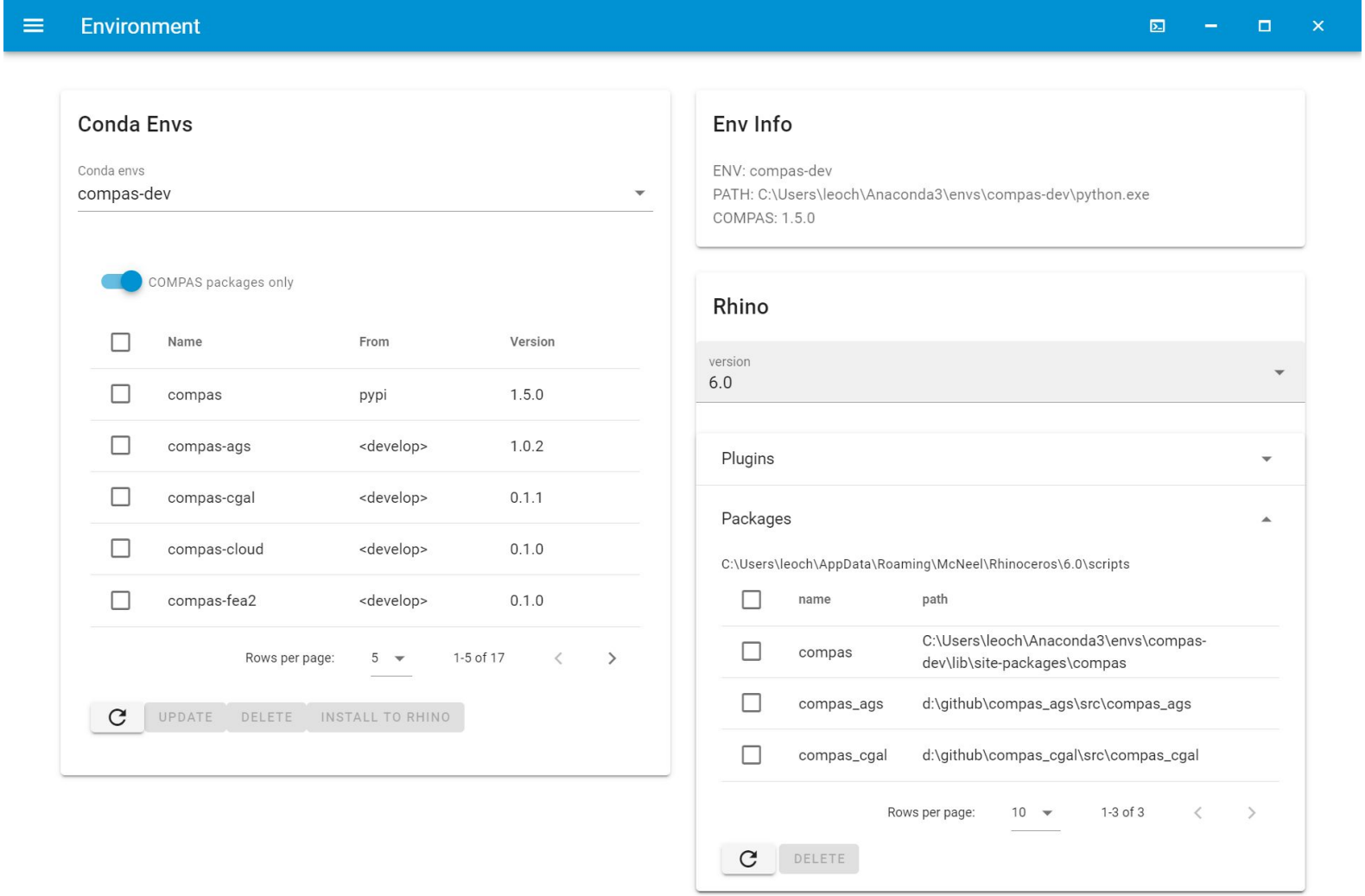
COMPAS core

- compas.data
- compas.datastructures
- compas.files
- compas.geometry
- compas.numerical
- compas.plugins
- compas.robots
- compas.rpc
- compas.scene
- compas.topology
- compas.utilities

COMPAS core extensions

- compas_cgal
- compas_cloud
- compas_dashboard
- compas_gmsh
- compas_libigl
- compas_occ
- compas_triangle
- compas_view2

- compas_plotters
- compas_rhino
- compas_ghpython
- compas_blender


Environment
Conda Envs
Conda envs
compas-dev
COMPAS packages only
Name
From
Version
compas
pypi
1.5.0
compas-ags
<develop>
1.0.2
compas-cgal
<develop>
0.1.1
compas-cloud
<develop>
0.1.0
compas-fea2
<develop>
0.1.0
Rows per page:
5
1-5 of 17
UPDATE
DELETE
INSTALL TO RHINO
Env Info
ENV: compas-dev
PATH: C:\Users\leoch\Anaconda3\envs\compas-dev\python.exe
COMPAS: 1.5.0
Rhino
version
6.0
Plugins
Packages
C:\Users\leoch\AppData\Roaming\McNeel\Rhinoceros\6.0\scripts
name
path
compas
C:\Users\leoch\Anaconda3\envs\compas-dev\lib\site-packages\compas
compas_ags
d:\github\compas_ags\src\compas_ags
compas_cgal
d:\github\compas_cgal\src\compas_cgal
Rows per page:
10
1-3 of 3
DELETE