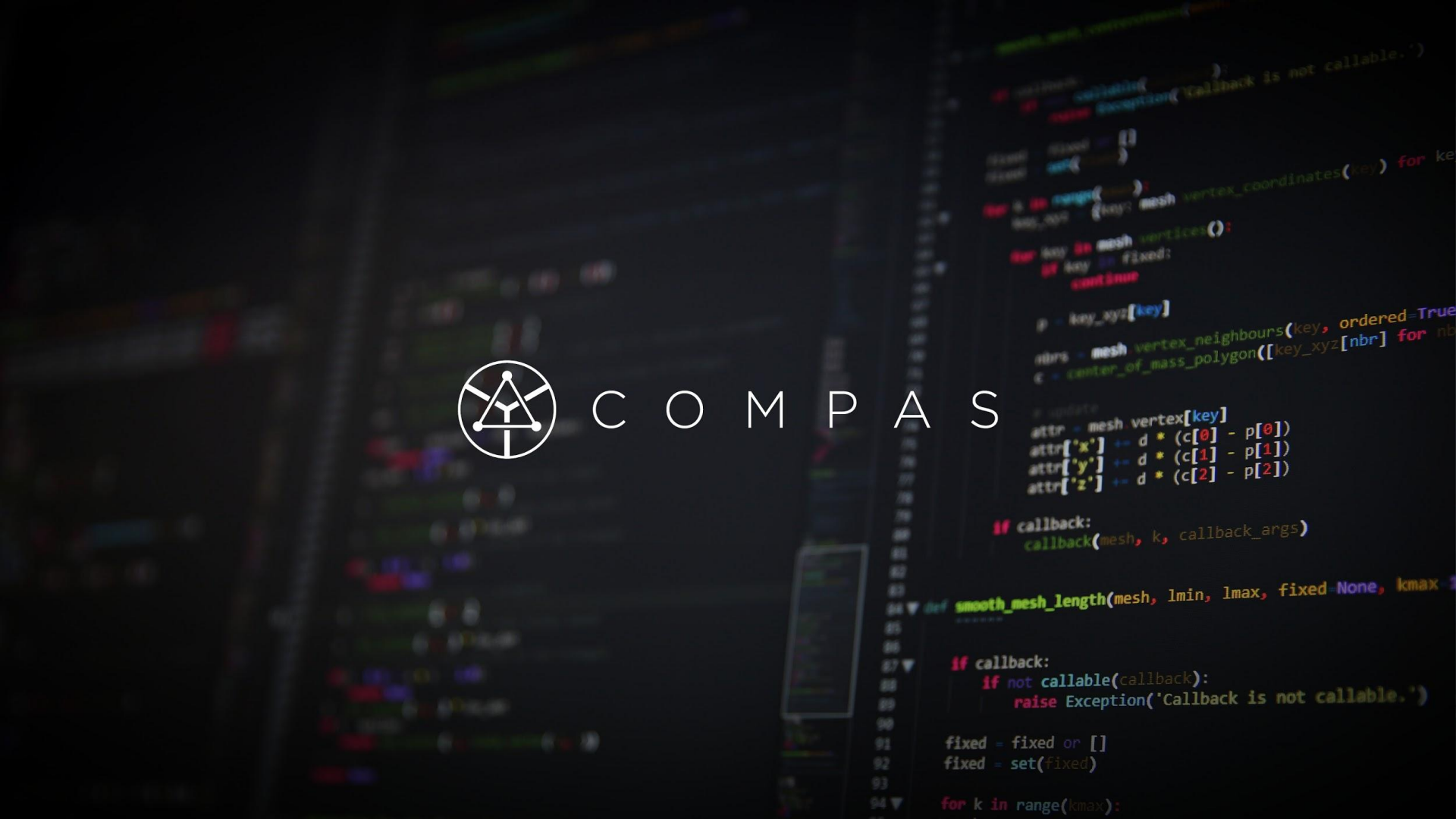

COMPAS

Visualisation

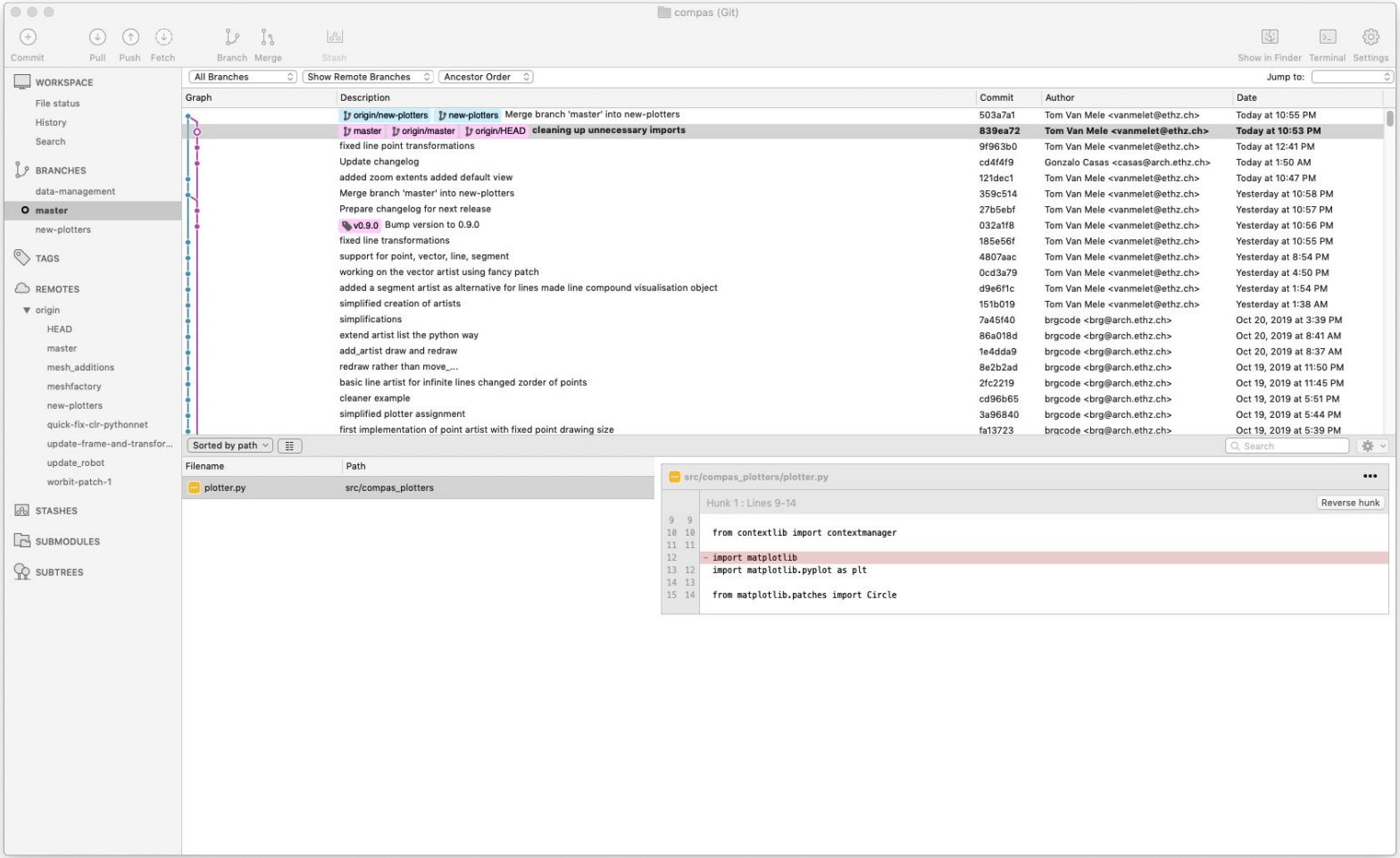
Geometry

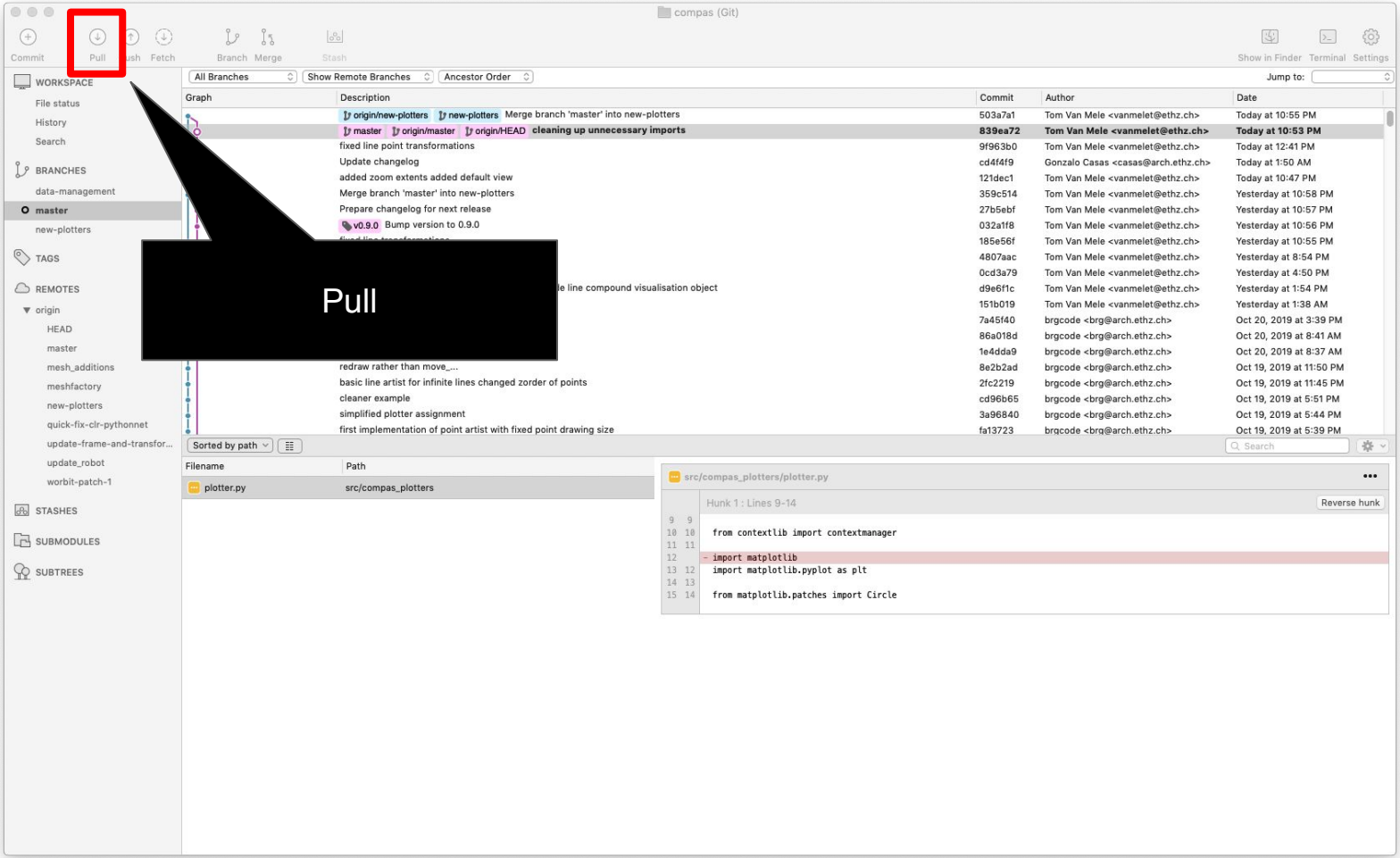
- Cross Product
- OABB using PCA

Data Structures

- Network
- Mesh
- (VolMesh)

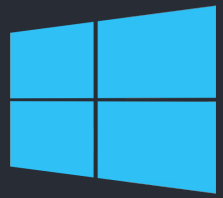
Update COMPAS repo


compas (Git)
Commit
Pull
Push
Fetch
Branch
Merge
Stash
Show in Finder
Terminal
Settings
WORKSPACE
File status
History
Search
BRANCHES
data-management
master
new-plotters
TAGS
REMOTES
origin
HEAD
master
mesh_additions
meshfactory
new-plotters
quick-fix-clr-pythonnet
update-frame-and-transfor...
update_robot
worbit-patch-1
STASHES
SUBMODULES
SUBTREES
All Branches
Show Remote Branches
Ancestor Order
Jump to:
Graph
Description
Commit
Author
Date
origin/new-plotters
new-plotters
Merge branch 'master' into new-plotters
503a7a1
Tom Van Mele <vanmelet@ethz.ch>
Today at 10:55 PM
master
origin/master
origin/HEAD
cleaning up unnecessary imports
839ea72
Tom Van Mele <vanmelet@ethz.ch>
Today at 10:53 PM
fixed line point transformations
9f963b0
Tom Van Mele <vanmelet@ethz.ch>
Today at 12:41 PM
Update changelog
cd4f4f9
Gonzalo Casas <casas@arch.ethz.ch>
Today at 1:50 AM
added zoom extents added default view
121dec1
Tom Van Mele <vanmelet@ethz.ch>
Today at 10:47 PM
Merge branch 'master' into new-plotters
359c514
Tom Van Mele <vanmelet@ethz.ch>
Yesterday at 10:58 PM
Prepare changelog for next release
27b5ebf
Tom Van Mele <vanmelet@ethz.ch>
Yesterday at 10:57 PM
v0.9.0
Bump version to 0.9.0
032a1f8
Tom Van Mele <vanmelet@ethz.ch>
Yesterday at 10:56 PM
fixed line transformations
185e56f
Tom Van Mele <vanmelet@ethz.ch>
Yesterday at 10:55 PM
support for point, vector, line, segment
4807aac
Tom Van Mele <vanmelet@ethz.ch>
Yesterday at 8:54 PM
working on the vector artist using fancy patch
0cd3a79
Tom Van Mele <vanmelet@ethz.ch>
Yesterday at 4:50 PM
added a segment artist as alternative for lines made line compound visualisation object
d9e6f1c
Tom Van Mele <vanmelet@ethz.ch>
Yesterday at 1:54 PM
simplified creation of artists
151b019
Tom Van Mele <vanmelet@ethz.ch>
Yesterday at 1:38 AM
simplifications
7a45f40
brgcode <brg@arch.ethz.ch>
Oct 20, 2019 at 3:39 PM
extend artist list the python way
86a018d
brgcode <brg@arch.ethz.ch>
Oct 20, 2019 at 8:41 AM
add_artist draw and redraw
1e4dda9
brgcode <brg@arch.ethz.ch>
Oct 20, 2019 at 8:37 AM
redraw rather than move_...
8e2b2ad
brgcode <brg@arch.ethz.ch>
Oct 19, 2019 at 11:50 PM
basic line artist for infinite lines changed zorder of points
2fc2219
brgcode <brg@arch.ethz.ch>
Oct 19, 2019 at 11:45 PM
cleaner example
cd96b65
brgcode <brg@arch.ethz.ch>
Oct 19, 2019 at 5:51 PM
simplified plotter assignment
3a96840
brgcode <brg@arch.ethz.ch>
Oct 19, 2019 at 5:44 PM
first implementation of point artist with fixed point drawing size
fa13723
brgcode <brg@arch.ethz.ch>
Oct 19, 2019 at 5:39 PM
Sorted by path
Search
Filename
Path
plotter.py
src/compas_plotters
src/compas_plotters/plotter.py
Hunk 1 : Lines 9-14
Reverse hunk
from contextlib import contextmanager
- import matplotlib
import matplotlib.pyplot as plt
from matplotlib.patches import Circle


compas (Git)
Commit
Pull
Fetch
Branch
Merge
Stash
Show in Finder
Terminal
Settings
WORKSPACE
File status
History
Search
BRANCHES
data-management
master
new-plotters
TAGS
REMOTES
origin
HEAD
master
mesh_additions
meshfactory
new-plotters
quick-fix-clr-pythonnet
update-frame-and-transfor...
update_robot
worbit-patch-1
STASHES
SUBMODULES
SUBTREES
All Branches
Show Remote Branches
Ancestor Order
Jump to:
Graph
Description
Commit
Author
Date
origin/new-plotters new-plotters Merge branch 'master' into new-plotters 503a7a1 Tom Van Mele <vanmelet@ethz.ch> Today at 10:55 PM
master origin/master origin/HEAD cleaning up unnecessary imports 839ea72 Tom Van Mele <vanmelet@ethz.ch> Today at 10:53 PM
fixed line point transformations 9f963b0 Tom Van Mele <vanmelet@ethz.ch> Today at 12:41 PM
Update changelog cd4f4f9 Gonzalo Casas <casas@arch.ethz.ch> Today at 1:50 AM
added zoom extents added default view 121dec1 Tom Van Mele <vanmelet@ethz.ch> Today at 10:47 PM
Merge branch 'master' into new-plotters 359c514 Tom Van Mele <vanmelet@ethz.ch> Yesterday at 10:58 PM
Prepare changelog for next release 27b5ebf Tom Van Mele <vanmelet@ethz.ch> Yesterday at 10:57 PM
v0.9.0 Bump version to 0.9.0 032a1f8 Tom Van Mele <vanmelet@ethz.ch> Yesterday at 10:56 PM
185e56f Tom Van Mele <vanmelet@ethz.ch> Yesterday at 10:55 PM
4807aac Tom Van Mele <vanmelet@ethz.ch> Yesterday at 8:54 PM
0cd3a79 Tom Van Mele <vanmelet@ethz.ch> Yesterday at 4:50 PM
d9e6f1c Tom Van Mele <vanmelet@ethz.ch> Yesterday at 1:54 PM
151b019 Tom Van Mele <vanmelet@ethz.ch> Yesterday at 1:38 AM
7a45f40 brgcode <brg@arch.ethz.ch> Oct 20, 2019 at 3:39 PM
86a018d brgcode <brg@arch.ethz.ch> Oct 20, 2019 at 8:41 AM
1e4dda9 brgcode <brg@arch.ethz.ch> Oct 20, 2019 at 8:37 AM
redraw rather than move_... 8e2b2ad brgcode <brg@arch.ethz.ch> Oct 19, 2019 at 11:50 PM
basic line artist for infinite lines changed zorder of points 2fc2219 brgcode <brg@arch.ethz.ch> Oct 19, 2019 at 11:45 PM
cleaner example cd96b65 brgcode <brg@arch.ethz.ch> Oct 19, 2019 at 5:51 PM
simplified plotter assignment 3a96840 brgcode <brg@arch.ethz.ch> Oct 19, 2019 at 5:44 PM
first implementation of point artist with fixed point drawing size fa13723 brgcode <brg@arch.ethz.ch> Oct 19, 2019 at 5:39 PM
Pull
Sorted by path
Search
Filename
Path
plotter.py
src/compas_plotters
src/compas_plotters/plotter.py
Hunk 1 : Lines 9-14
Reverse hunk
from contextlib import contextmanager
- import matplotlib
import matplotlib.pyplot as plt
from matplotlib.patches import Circle

ita19 vs ita19-dev

⌘ + SPACE



Terminal

Anaconda Prompt

```
conda activate ita19
python
```

```
conda activate ita19
python
```

```
conda activate ita19
python
```

```
>>> import compas
>>>
```

```
>>> import compas
>>> compas.__version__
'0.8.1'
>>>
```

```
>>> import compas
>>> compas.__version__
'0.8.1'
>>>
```

```
>>> import compas
>>> compas.__version__
'0.8.1'
>>> exit()
```

```
conda activate ita19-dev
python
```

```
conda activate ita19-dev
python
```

```
conda activate ita19-dev
python
```

```
>>> import compas
>>>
```

```
>>> import compas
>>> compas.__version__
'0.9.0'
>>>
```

```
>>> import compas
>>> compas.__version__
'0.9.0'
>>>
```

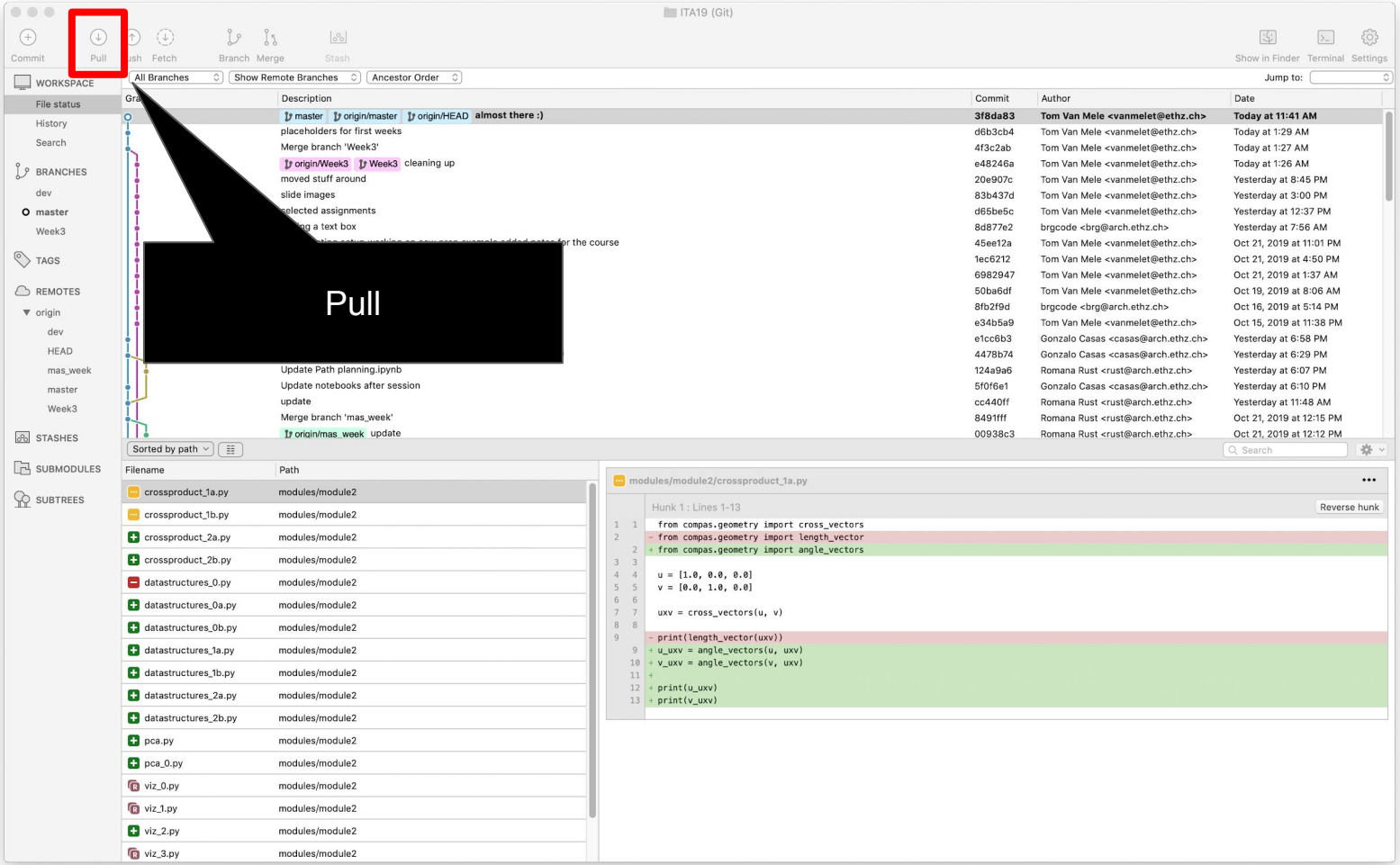
```
>>> import compas
>>> compas.__version__
'0.9.0'
>>> exit()
```

Update Rhino

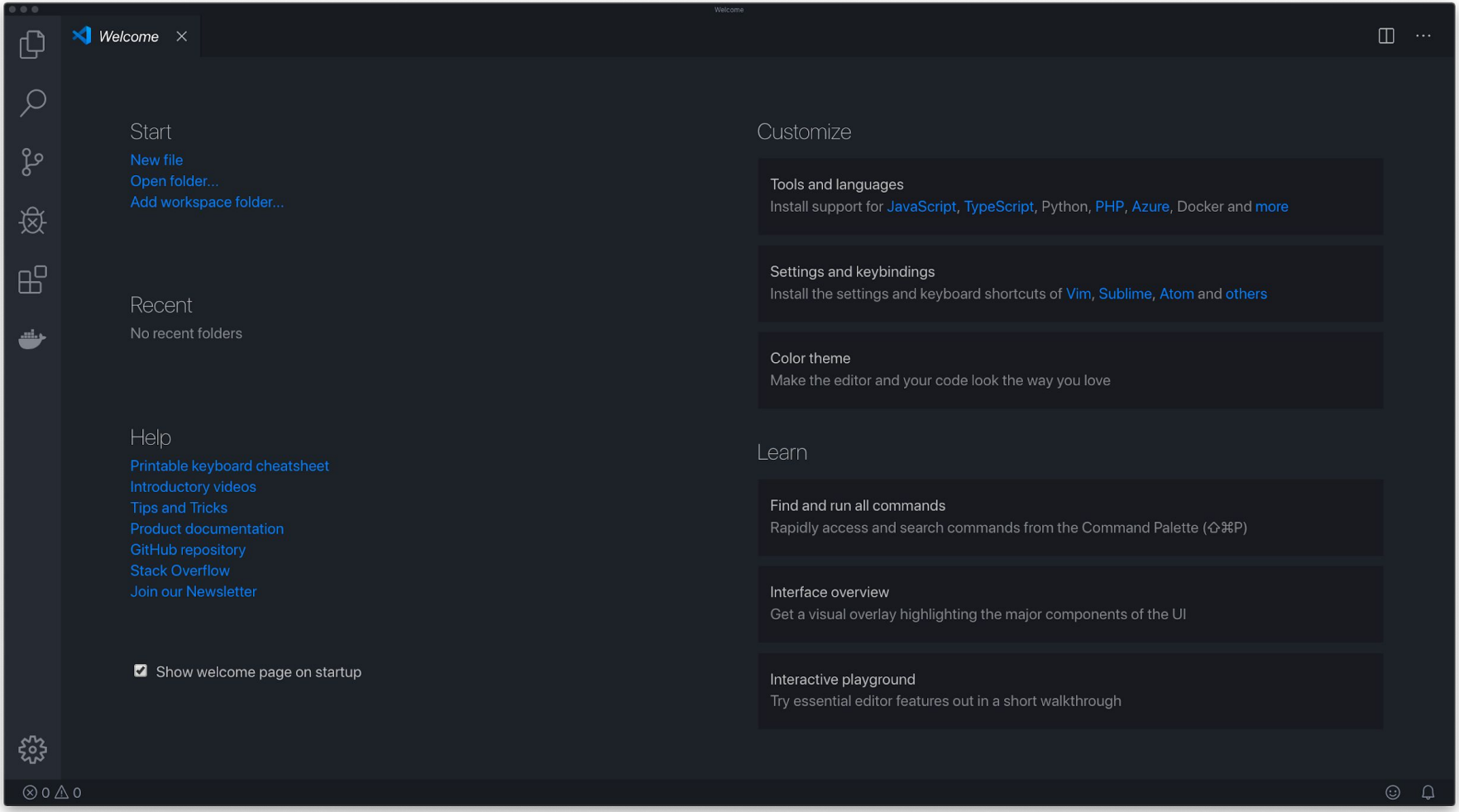
```
python -m compas_rhino.install -v 6.0 -p compas compas_rhino compas_ghpython
```

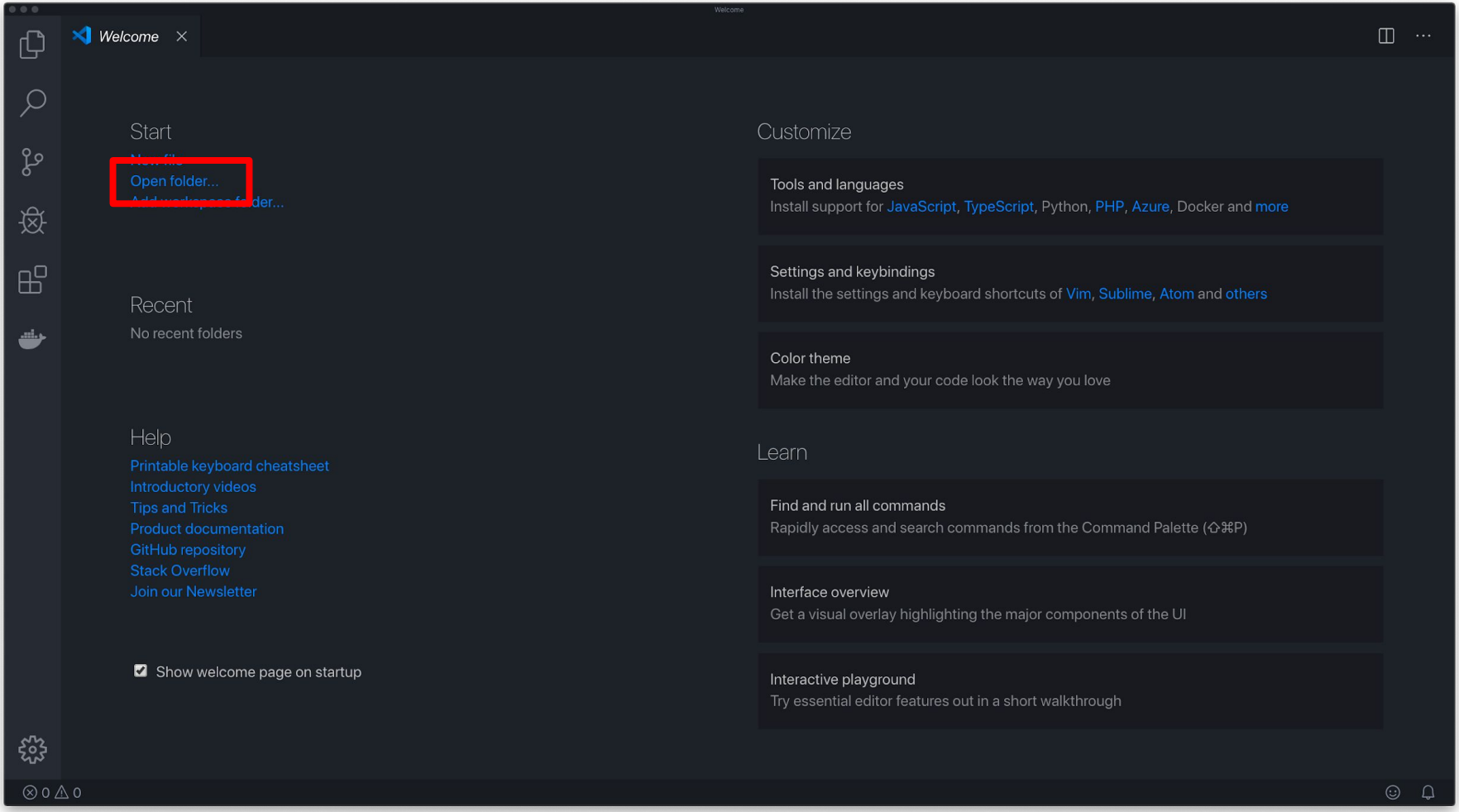

```
python -m compas_rhino.install -v 6.0 -p compas compas_rhino compas_ghpython
```

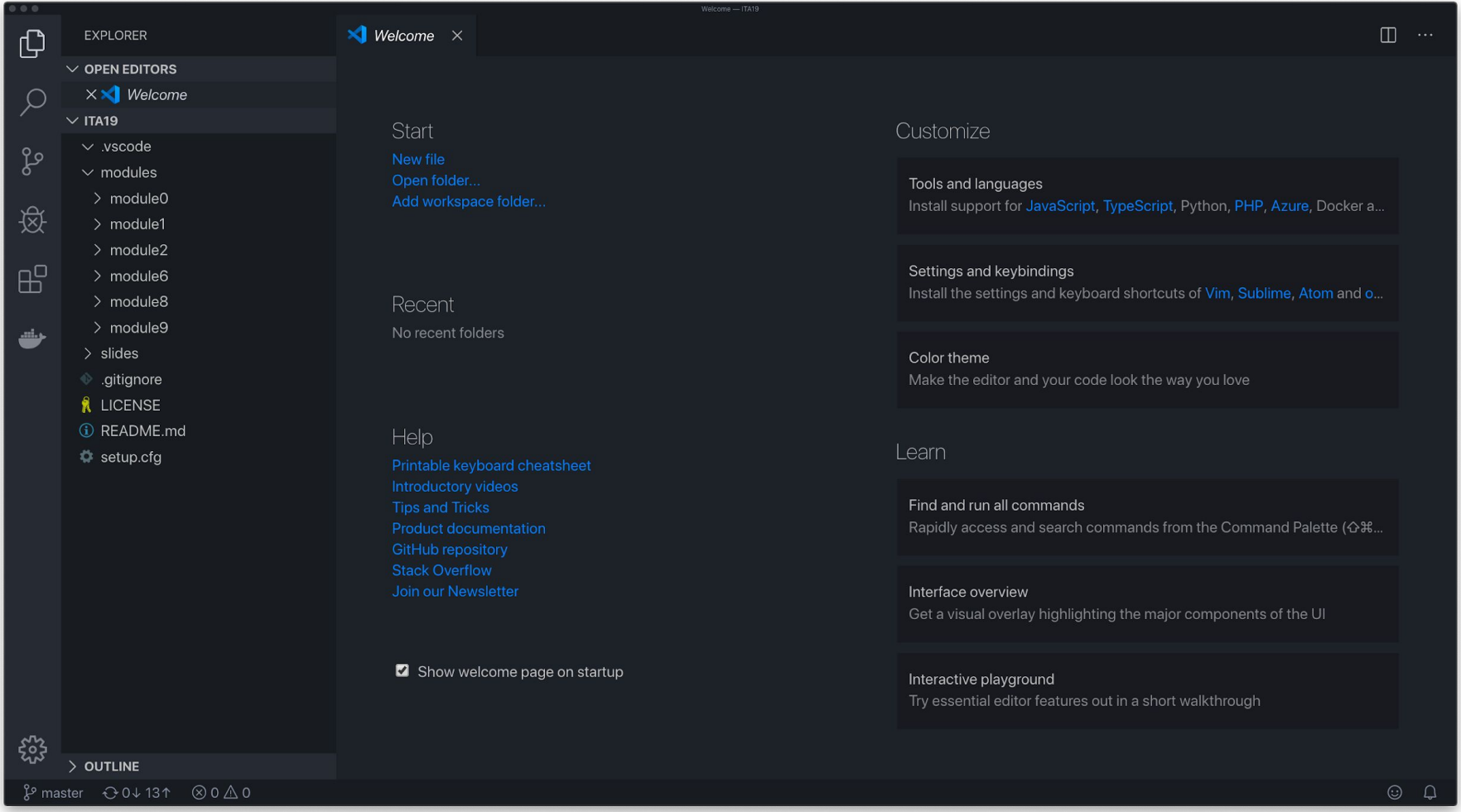
Update ITA19 repo

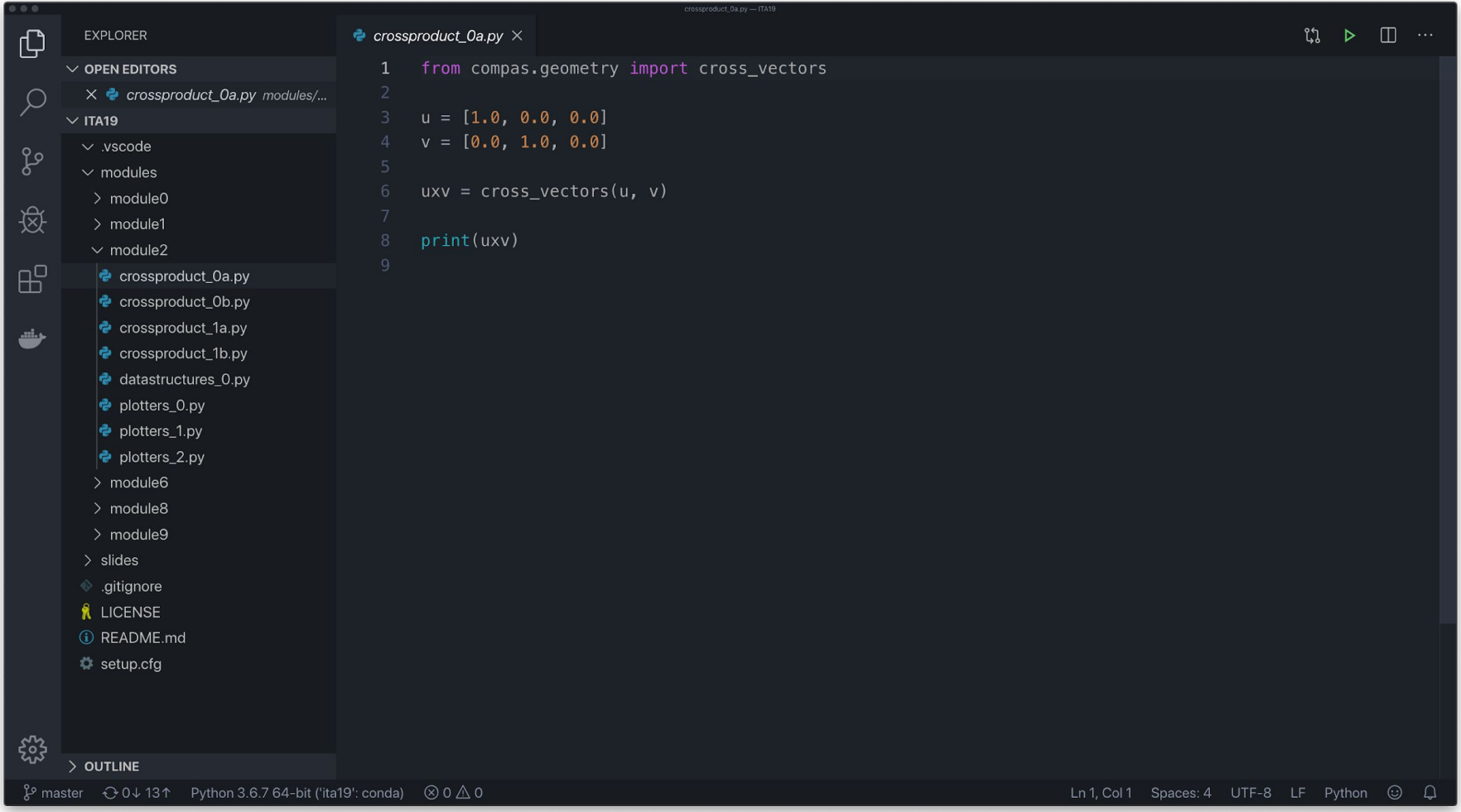

ITA19 (Git)
Commit
Pull
Fetch
Branch
Merge
Stash
Show in Finder
Terminal
Settings
WORKSPACE
File status
History
Search
BRANCHES
dev
master
Week3
TAGS
REMOTES
origin
dev
HEAD
mas_week
master
Week3
STASHES
SUBMODULES
SUBTREES
All Branches
Show Remote Branches
Ancestor Order
Jump to:
Description
Commit
Author
Date
master origin/master origin/HEAD almost there :) 3f8da83 Tom Van Mele <vanmelet@ethz.ch> Today at 11:41 AM
placeholders for first weeks d6b3cb4 Tom Van Mele <vanmelet@ethz.ch> Today at 1:29 AM
Merge branch 'Week3' 4f3c2ab Tom Van Mele <vanmelet@ethz.ch> Today at 1:27 AM
origin/Week3 Week3 cleaning up e48246a Tom Van Mele <vanmelet@ethz.ch> Today at 1:26 AM
moved stuff around 20e907c Tom Van Mele <vanmelet@ethz.ch> Yesterday at 8:45 PM
slide images 83b437d Tom Van Mele <vanmelet@ethz.ch> Yesterday at 3:00 PM
selected assignments d65be5c Tom Van Mele <vanmelet@ethz.ch> Yesterday at 12:37 PM
ng a text box 8d877e2 brgcode <brg@arch.ethz.ch> Yesterday at 7:56 AM
for the course 45ee12a Tom Van Mele <vanmelet@ethz.ch> Oct 21, 2019 at 11:01 PM
1ec6212 Tom Van Mele <vanmelet@ethz.ch> Oct 21, 2019 at 4:50 PM
6982947 Tom Van Mele <vanmelet@ethz.ch> Oct 21, 2019 at 1:37 AM
50ba6df Tom Van Mele <vanmelet@ethz.ch> Oct 19, 2019 at 8:06 AM
8fb2f9d brgcode <brg@arch.ethz.ch> Oct 16, 2019 at 5:14 PM
e34b5a9 Tom Van Mele <vanmelet@ethz.ch> Oct 15, 2019 at 11:38 PM
e1cc6b3 Gonzalo Casas <casas@arch.ethz.ch> Yesterday at 6:58 PM
4478b74 Gonzalo Casas <casas@arch.ethz.ch> Yesterday at 6:29 PM
Update Path planning.ipynb 124a9a6 Romana Rust <rust@arch.ethz.ch> Yesterday at 6:07 PM
Update notebooks after session 5f0f6e1 Gonzalo Casas <casas@arch.ethz.ch> Yesterday at 6:10 PM
update cc440ff Romana Rust <rust@arch.ethz.ch> Yesterday at 11:48 AM
Merge branch 'mas_week' 8491fff Romana Rust <rust@arch.ethz.ch> Oct 21, 2019 at 12:15 PM
origin/mas_week update 00938c3 Romana Rust <rust@arch.ethz.ch> Oct 21, 2019 at 12:12 PM
Pull
Sorted by path
Search
Filename
Path
crossproduct_1a.py modules/module2
crossproduct_1b.py modules/module2
crossproduct_2a.py modules/module2
crossproduct_2b.py modules/module2
datastructures_0.py modules/module2
datastructures_0a.py modules/module2
datastructures_0b.py modules/module2
datastructures_1a.py modules/module2
datastructures_1b.py modules/module2
datastructures_2a.py modules/module2
datastructures_2b.py modules/module2
pca.py modules/module2
pca_0.py modules/module2
viz_0.py modules/module2
viz_1.py modules/module2
viz_2.py modules/module2
viz_3.py modules/module2
modules/module2/crossproduct_1a.py
Hunk 1 : Lines 1-13
Reverse hunk
from compas.geometry import cross_vectors
- from compas.geometry import length_vector
+ from compas.geometry import angle_vectors
u = [1.0, 0.0, 0.0]
v = [0.0, 1.0, 0.0]
uxv = cross_vectors(u, v)
- print(length_vector(uxv))
+ u_uxv = angle_vectors(u, uxv)
+ v_uxv = angle_vectors(v, uxv)
+
+ print(u_uxv)
+ print(v_uxv)

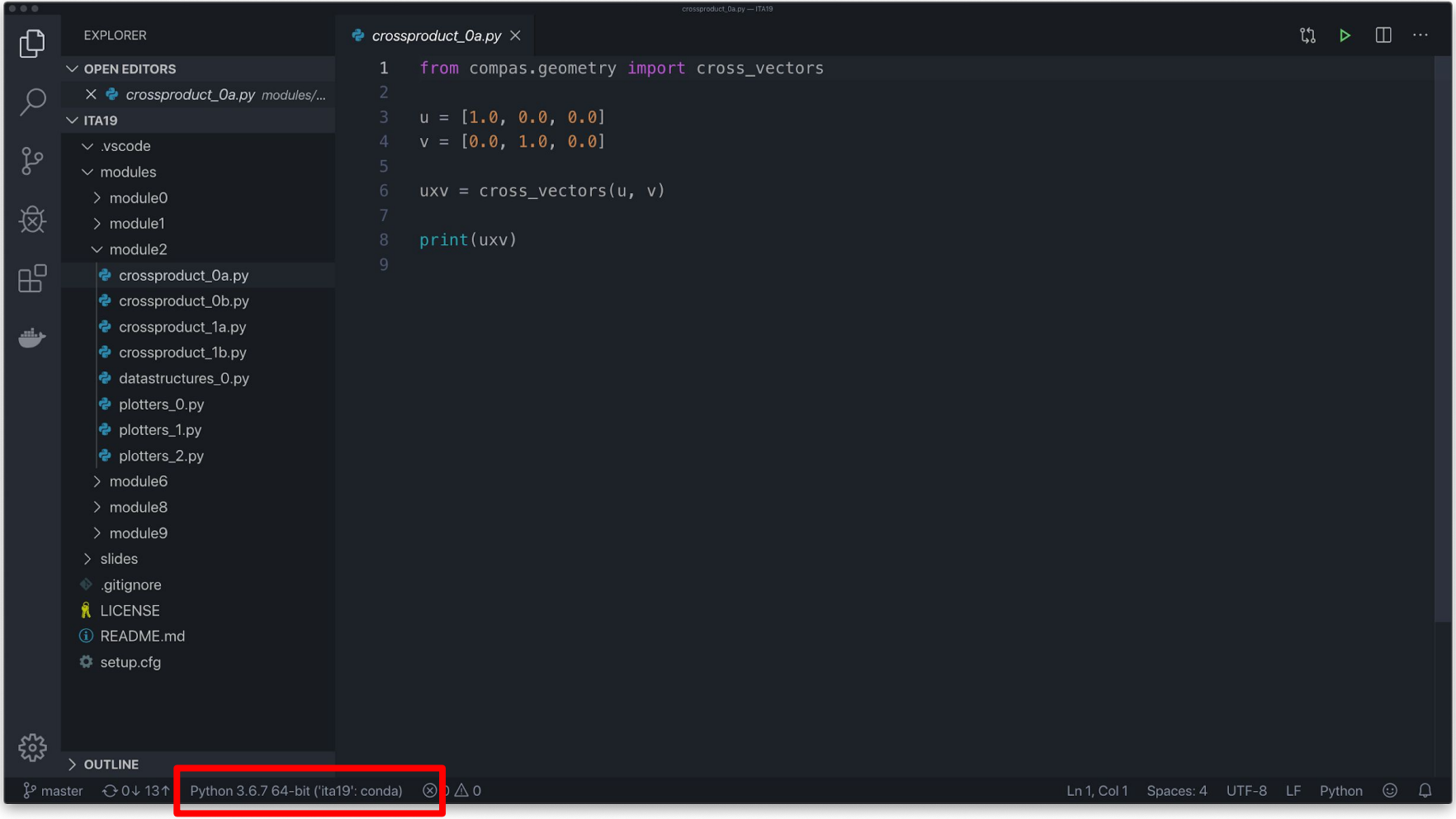
VS Code

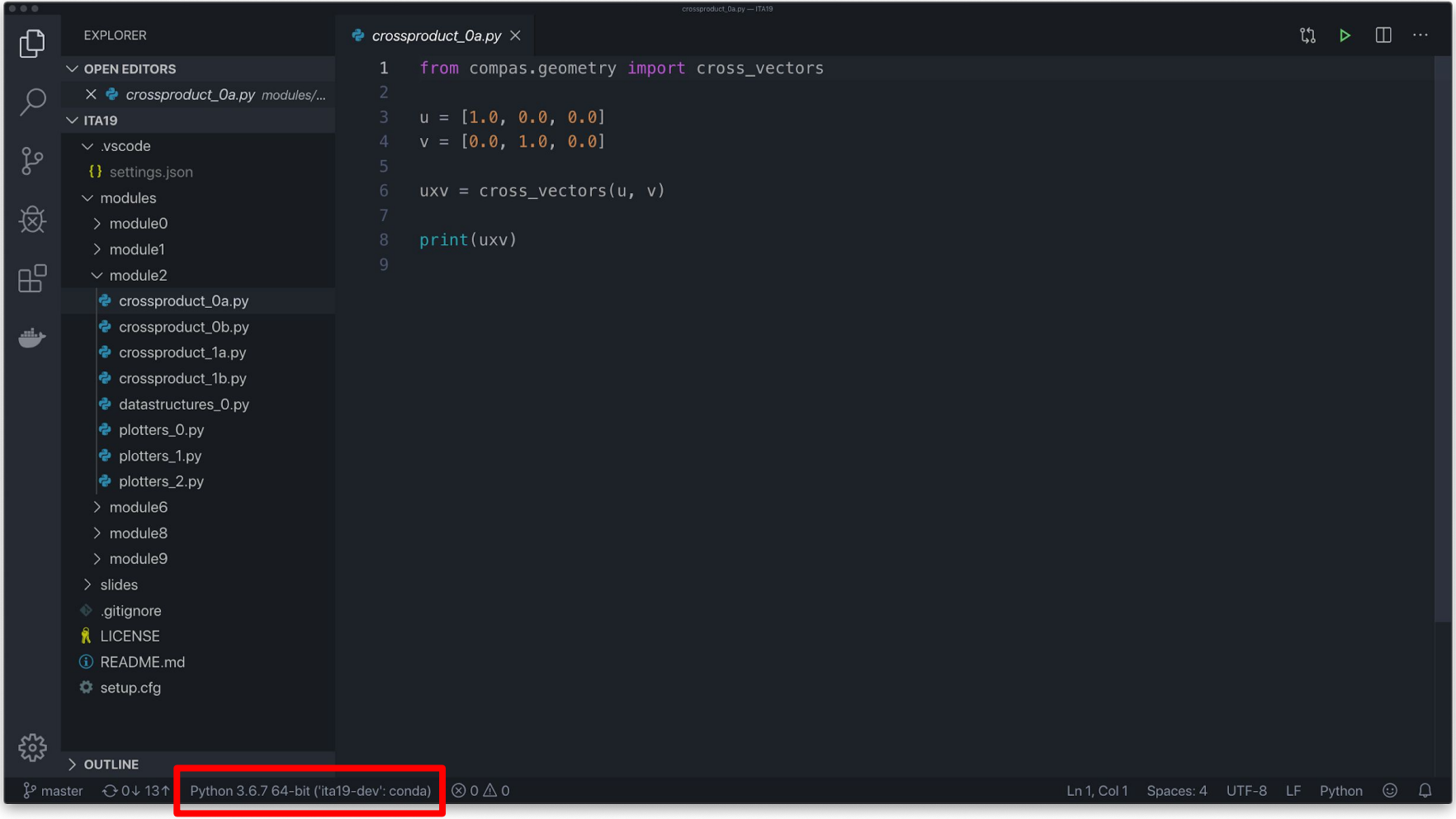

Welcome
Welcome
Start
New file
Open folder...
Add workspace folder...
Recent
No recent folders
Help
Printable keyboard cheatsheet
Introductory videos
Tips and Tricks
Product documentation
GitHub repository
Stack Overflow
Join our Newsletter
Show welcome page on startup
Customize
Tools and languages
Install support for JavaScript, TypeScript, Python, PHP, Azure, Docker and more
Settings and keybindings
Install the settings and keyboard shortcuts of Vim, Sublime, Atom and others
Color theme
Make the editor and your code look the way you love
Learn
Find and run all commands
Rapidly access and search commands from the Command Palette (⇧⌘P)
Interface overview
Get a visual overlay highlighting the major components of the UI
Interactive playground
Try essential editor features out in a short walkthrough
0 0

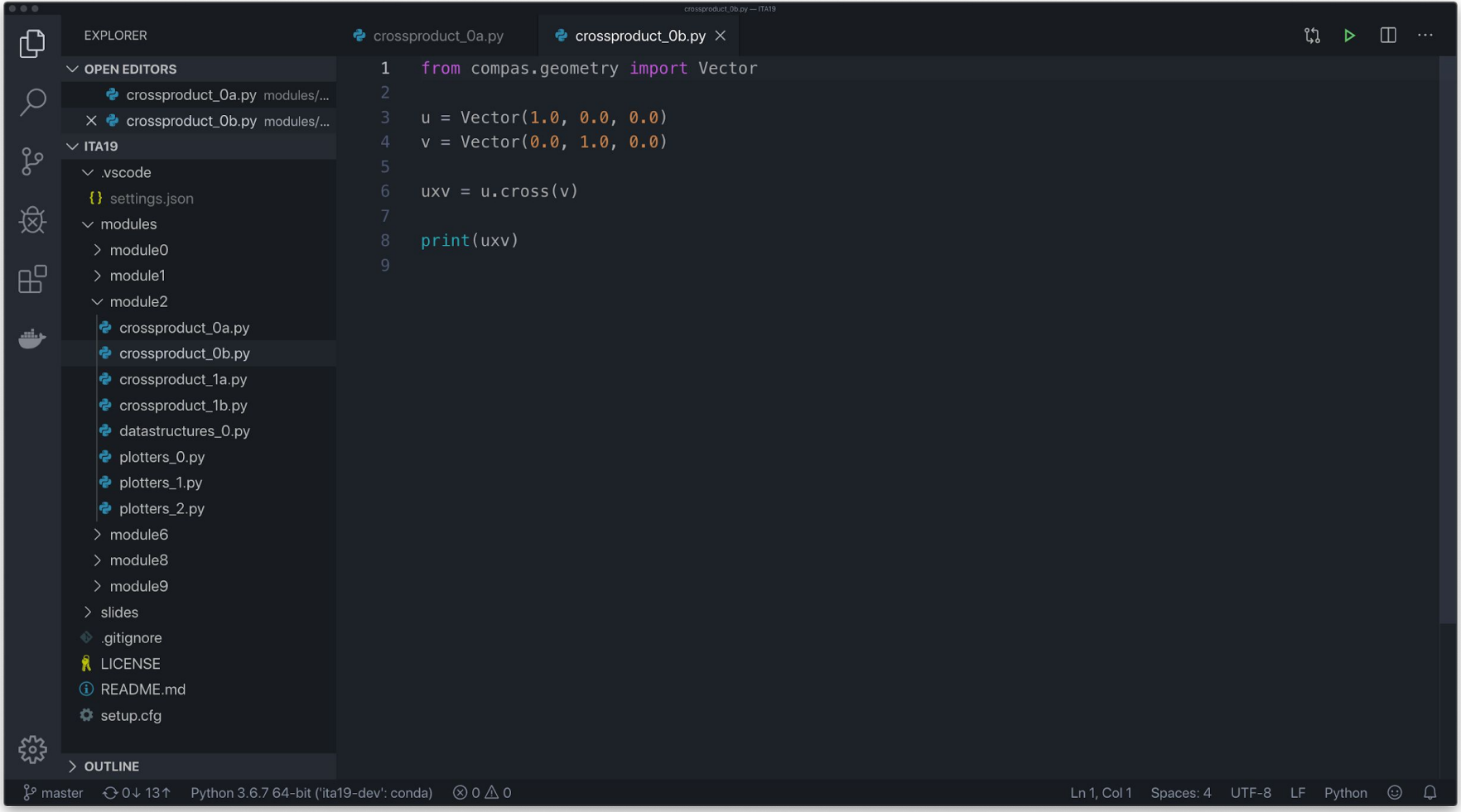

Welcome
Welcome
Start
Open folder...
Recent
No recent folders
Help
Printable keyboard cheatsheet
Introductory videos
Tips and Tricks
Product documentation
GitHub repository
Stack Overflow
Join our Newsletter
Show welcome page on startup
Customize
Tools and languages
Install support for JavaScript, TypeScript, Python, PHP, Azure, Docker and more
Settings and keybindings
Install the settings and keyboard shortcuts of Vim, Sublime, Atom and others
Color theme
Make the editor and your code look the way you love
Learn
Find and run all commands
Rapidly access and search commands from the Command Palette (⇧⌘P)
Interface overview
Get a visual overlay highlighting the major components of the UI
Interactive playground
Try essential editor features out in a short walkthrough
0 0


Welcome — ITA19
EXPLORER
OPEN EDITORS
Welcome
ITA19
.vscode
modules
module0
module1
module2
module6
module8
module9
slides
.gitignore
LICENSE
README.md
setup.cfg
OUTLINE
Welcome
Start
New file
Open folder...
Add workspace folder...
Recent
No recent folders
Help
Printable keyboard cheatsheet
Introductory videos
Tips and Tricks
Product documentation
GitHub repository
Stack Overflow
Join our Newsletter
Show welcome page on startup
Customize
Tools and languages
Install support for JavaScript, TypeScript, Python, PHP, Azure, Docker a...
Settings and keybindings
Install the settings and keyboard shortcuts of Vim, Sublime, Atom and o...
Color theme
Make the editor and your code look the way you love
Learn
Find and run all commands
Rapidly access and search commands from the Command Palette (⇧⌘...
Interface overview
Get a visual overlay highlighting the major components of the UI
Interactive playground
Try essential editor features out in a short walkthrough
master
0↓ 13↑
0 0

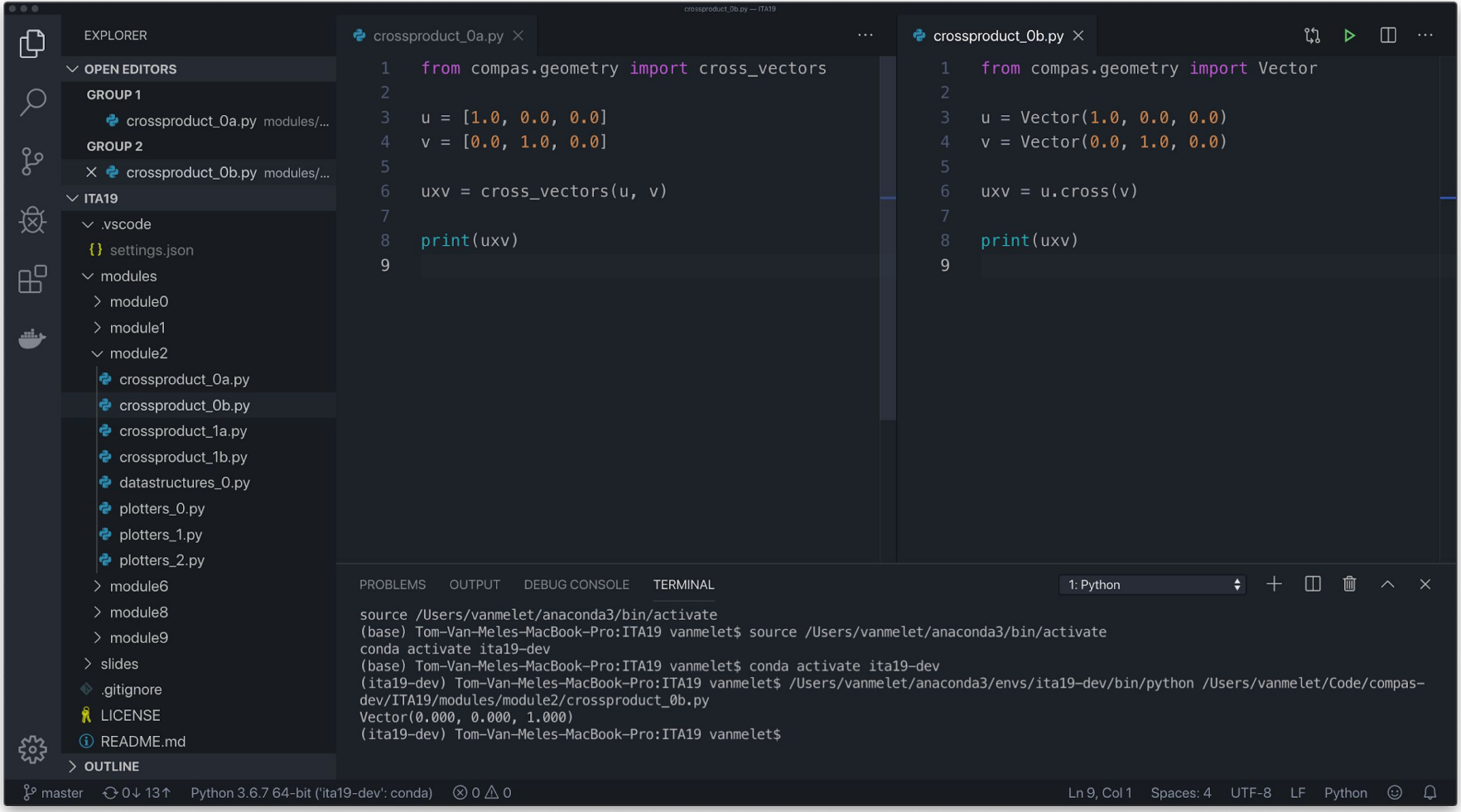

crossproduct_0a.py — ITA19
EXPLORER
OPEN EDITORS
crossproduct_0a.py modules/...
ITA19
.vscode
modules
module0
module1
module2
crossproduct_0a.py
crossproduct_0b.py
crossproduct_1a.py
crossproduct_1b.py
datastructures_0.py
plotters_0.py
plotters_1.py
plotters_2.py
module6
module8
module9
slides
.gitignore
LICENSE
README.md
setup.cfg
OUTLINE
crossproduct_0a.py
from compas.geometry import cross_vectors
u = [1.0, 0.0, 0.0]
v = [0.0, 1.0, 0.0]
uxv = cross_vectors(u, v)
print(uxv)
master
0↓ 13↑
Python 3.6.7 64-bit ('ita19': conda)
0 0
Ln 1, Col 1
Spaces: 4
UTF-8
LF
Python

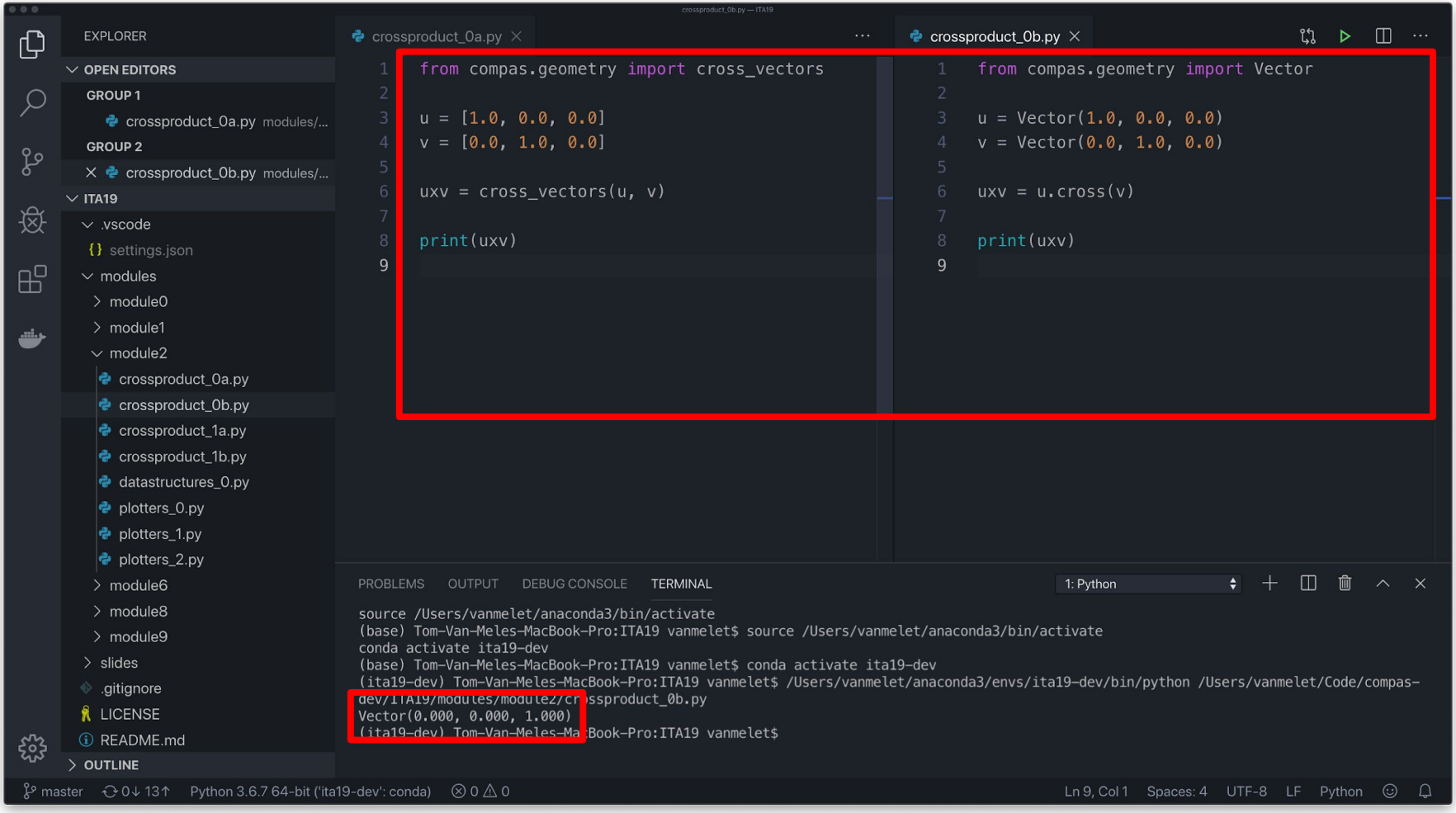

crossproduct_0a.py — ITA19
EXPLORER
OPEN EDITORS
crossproduct_0a.py modules/...
ITA19
.vscode
modules
module0
module1
module2
crossproduct_0a.py
crossproduct_0b.py
crossproduct_1a.py
crossproduct_1b.py
datastructures_0.py
plotters_0.py
plotters_1.py
plotters_2.py
module6
module8
module9
slides
.gitignore
LICENSE
README.md
setup.cfg
OUTLINE
crossproduct_0a.py
1 from compas.geometry import cross_vectors
2
3 u = [1.0, 0.0, 0.0]
4 v = [0.0, 1.0, 0.0]
5
6 uxv = cross_vectors(u, v)
7
8 print(uxv)
9
master
0↓ 13↑
Python 3.6.7 64-bit ('ita19': conda)
0
Ln 1, Col 1
Spaces: 4
UTF-8
LF
Python


crossproduct_0a.py — ITA19
EXPLORER
OPEN EDITORS
crossproduct_0a.py modules/...
ITA19
.vscode
settings.json
modules
module0
module1
module2
crossproduct_0a.py
crossproduct_0b.py
crossproduct_1a.py
crossproduct_1b.py
datastructures_0.py
plotters_0.py
plotters_1.py
plotters_2.py
module6
module8
module9
slides
.gitignore
LICENSE
README.md
setup.cfg
OUTLINE
crossproduct_0a.py
1 from compas.geometry import cross_vectors
2
3 u = [1.0, 0.0, 0.0]
4 v = [0.0, 1.0, 0.0]
5
6 uxv = cross_vectors(u, v)
7
8 print(uxv)
9
master
0↓ 13↑
Python 3.6.7 64-bit ('ita19-dev': conda)
0 0
Ln 1, Col 1
Spaces: 4
UTF-8
LF
Python


crossproduct_0b.py — ITA19
EXPLORER
crossproduct_0a.py
crossproduct_0b.py
OPEN EDITORS
crossproduct_0a.py modules/...
crossproduct_0b.py modules/...
ITA19
.vscode
settings.json
modules
module0
module1
module2
crossproduct_0a.py
crossproduct_0b.py
crossproduct_1a.py
crossproduct_1b.py
datastructures_0.py
plotters_0.py
plotters_1.py
plotters_2.py
module6
module8
module9
slides
.gitignore
LICENSE
README.md
setup.cfg
OUTLINE
1 from compas.geometry import Vector
2
3 u = Vector(1.0, 0.0, 0.0)
4 v = Vector(0.0, 1.0, 0.0)
5
6 uxv = u.cross(v)
7
8 print(uxv)
9
master
0↓ 13↑
Python 3.6.7 64-bit ('ita19-dev': conda)
0 0
Ln 1, Col 1
Spaces: 4
UTF-8
LF
Python


crossproduct_0b.py — ITA19
EXPLORER
OPEN EDITORS
GROUP 1
crossproduct_0a.py modules/...
GROUP 2
crossproduct_0b.py modules/...
ITA19
.vscode
settings.json
modules
module0
module1
module2
crossproduct_0a.py
crossproduct_0b.py
crossproduct_1a.py
crossproduct_1b.py
datastructures_0.py
plotters_0.py
plotters_1.py
plotters_2.py
module6
module8
module9
slides
.gitignore
LICENSE
README.md
OUTLINE
crossproduct_0a.py
1 from compas.geometry import cross_vectors
2
3 u = [1.0, 0.0, 0.0]
4 v = [0.0, 1.0, 0.0]
5
6 uxv = cross_vectors(u, v)
7
8 print(uxv)
9
crossproduct_0b.py
1 from compas.geometry import Vector
2
3 u = Vector(1.0, 0.0, 0.0)
4 v = Vector(0.0, 1.0, 0.0)
5
6 uxv = u.cross(v)
7
8 print(uxv)
9
master
0↓ 13↑
Python 3.6.7 64-bit ('ita19-dev': conda)
0 0
Ln 9, Col 1
Spaces: 4
UTF-8
LF
Python


crossproduct_0b.py — ITA19
EXPLORER
OPEN EDITORS
GROUP 1
crossproduct_0a.py modules/...
GROUP 2
crossproduct_0b.py modules/...
ITA19
.vscode
settings.json
modules
module0
module1
module2
crossproduct_0a.py
crossproduct_0b.py
crossproduct_1a.py
crossproduct_1b.py
datastructures_0.py
plotters_0.py
plotters_1.py
plotters_2.py
module6
module8
module9
slides
.gitignore
LICENSE
README.md
OUTLINE
crossproduct_0a.py
1 from compas.geometry import cross_vectors
2
3 u = [1.0, 0.0, 0.0]
4 v = [0.0, 1.0, 0.0]
5
6 uxv = cross_vectors(u, v)
7
8 print(uxv)
9
crossproduct_0b.py
1 from compas.geometry import Vector
2
3 u = Vector(1.0, 0.0, 0.0)
4 v = Vector(0.0, 1.0, 0.0)
5
6 uxv = u.cross(v)
7
8 print(uxv)
9
PROBLEMS
OUTPUT
DEBUG CONSOLE
TERMINAL
1: Python
source /Users/vanmelet/anaconda3/bin/activate
(base) Tom-Van-Meles-MacBook-Pro:ITA19 vanmelet$ source /Users/vanmelet/anaconda3/bin/activate
conda activate ita19-dev
(base) Tom-Van-Meles-MacBook-Pro:ITA19 vanmelet$ conda activate ita19-dev
(ita19-dev) Tom-Van-Meles-MacBook-Pro:ITA19 vanmelet$ /Users/vanmelet/anaconda3/envs/ita19-dev/bin/python /Users/vanmelet/Code/compas-dev/ITA19/modules/module2/crossproduct_0b.py
Vector(0.000, 0.000, 1.000)
(ita19-dev) Tom-Van-Meles-MacBook-Pro:ITA19 vanmelet$
master
0↓ 13↑
Python 3.6.7 64-bit ('ita19-dev': conda)
0 0
Ln 9, Col 1
Spaces: 4
UTF-8
LF
Python


crossproduct_0b.py — ITA19
EXPLORER
OPEN EDITORS
GROUP 1
crossproduct_0a.py modules/...
GROUP 2
crossproduct_0b.py modules/...
ITA19
.vscode
settings.json
modules
module0
module1
module2
crossproduct_0a.py
crossproduct_0b.py
crossproduct_1a.py
crossproduct_1b.py
datastructures_0.py
plotters_0.py
plotters_1.py
plotters_2.py
module6
module8
module9
slides
.gitignore
LICENSE
README.md
OUTLINE
crossproduct_0a.py
1 from compas.geometry import cross_vectors
2
3 u = [1.0, 0.0, 0.0]
4 v = [0.0, 1.0, 0.0]
5
6 uxv = cross_vectors(u, v)
7
8 print(uxv)
9
crossproduct_0b.py
1 from compas.geometry import Vector
2
3 u = Vector(1.0, 0.0, 0.0)
4 v = Vector(0.0, 1.0, 0.0)
5
6 uxv = u.cross(v)
7
8 print(uxv)
9
PROBLEMS
OUTPUT
DEBUG CONSOLE
TERMINAL
1: Python
source /Users/vanmelet/anaconda3/bin/activate
(base) Tom-Van-Meles-MacBook-Pro:ITA19 vanmelet$ source /Users/vanmelet/anaconda3/bin/activate
conda activate ita19-dev
(base) Tom-Van-Meles-MacBook-Pro:ITA19 vanmelet$ conda activate ita19-dev
(ita19-dev) Tom-Van-Meles-MacBook-Pro:ITA19 vanmelet$ /Users/vanmelet/anaconda3/envs/ita19-dev/bin/python /Users/vanmelet/Code/compas-dev/ITA19/modules/module2/crossproduct_0b.py
Vector(0.000, 0.000, 1.000)
(ita19-dev) Tom-Van-Meles-MacBook-Pro:ITA19 vanmelet$
master
0↓ 13↑
Python 3.6.7 64-bit ('ita19-dev': conda)
0 0
Ln 9, Col 1
Spaces: 4
UTF-8
LF
Python

```python
from compas.geometry import cross_vectors

u = [1.0, 0.0, 0.0]
v = [0.0, 1.0, 0.0]

uxv = cross_vectors(u, v)

print(uxv)
```

```
[0.0, 0.0, 1.0]
```

```python
from compas.geometry import Vector

u = Vector(1.0, 0.0, 0.0)
v = Vector(0.0, 1.0, 0.0)

uxv = u.cross(v)

print(uxv)
```

```
Vector(0.000, 0.000, 1.000)
```

Visualisation

```python
from compas.geometry import pointcloud_xy
from compas_plotters import Plotter

cloud = pointcloud_xy(20, xbounds=(0, 10), ybounds=(0, 5))

points = []
for xyz in cloud:
    points.append({'pos': xyz, 'radius': 0.1})

plotter = Plotter(figsize=(16, 10))
plotter.draw_points(points)
plotter.show()
```

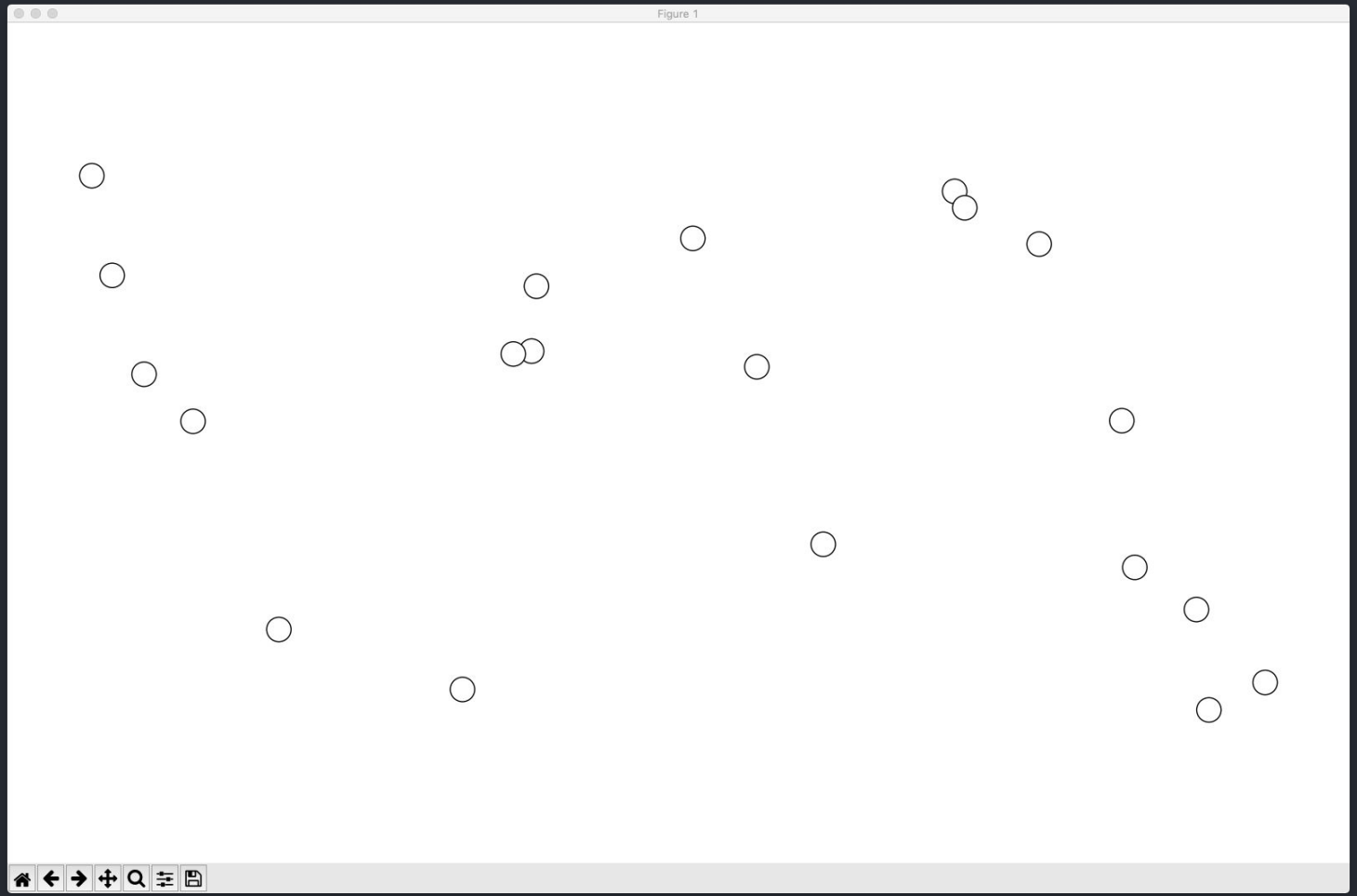


Figure 1

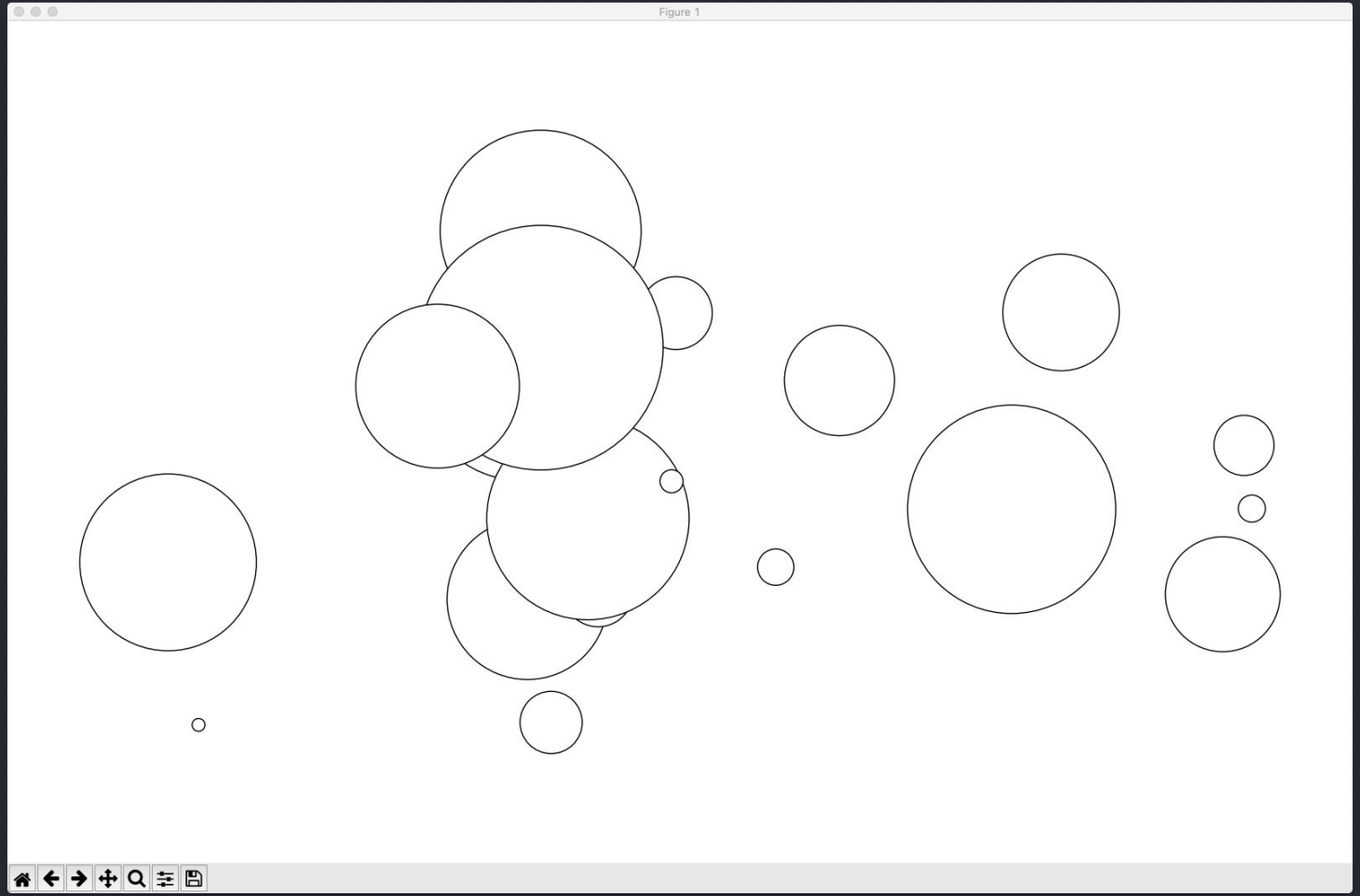


Figure 1

```
from ... import ...
from compas.geometry import pointcloud_xy
from compas_plotters import Plotter

cloud = pointcloud_xy(20, (0, 10), (0, 5))

points = []
for xyz in cloud:
    points.append({'pos': xyz, 'radius': ...})

plotter = Plotter(figsize=(8, 5))
plotter.draw_points(points)
plotter.show()
```

```python
from ... import ...
from compas.geometry import pointcloud_xy
from compas_plotters import Plotter

cloud = pointcloud_xy(20, (0, 10), (0, 5))

points = []
for xyz in cloud:
    points.append({'pos': xyz, 'radius': ...})

plotter = Plotter(figsize=(8, 5))
plotter.draw_points(points)
plotter.show()
```

```python
from random import random
from compas.geometry import pointcloud_xy
from compas_plotters import Plotter

cloud = pointcloud_xy(20, (0, 10), (0, 5))

points = []
for xyz in cloud:
    points.append({'pos': xyz, 'radius': random()})

plotter = Plotter(figsize=(8, 5))
plotter.draw_points(points)
plotter.show()
```

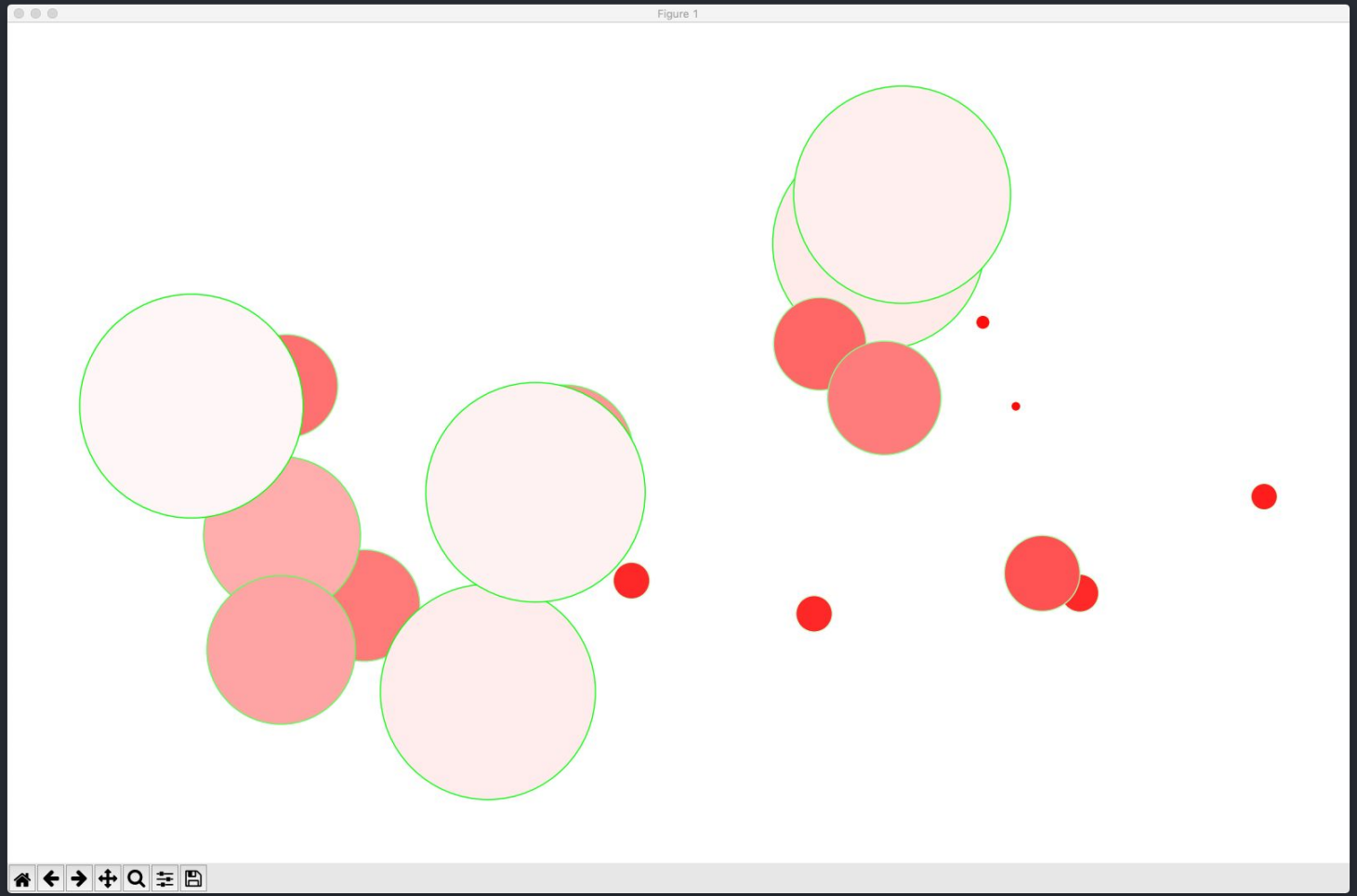


Figure 1

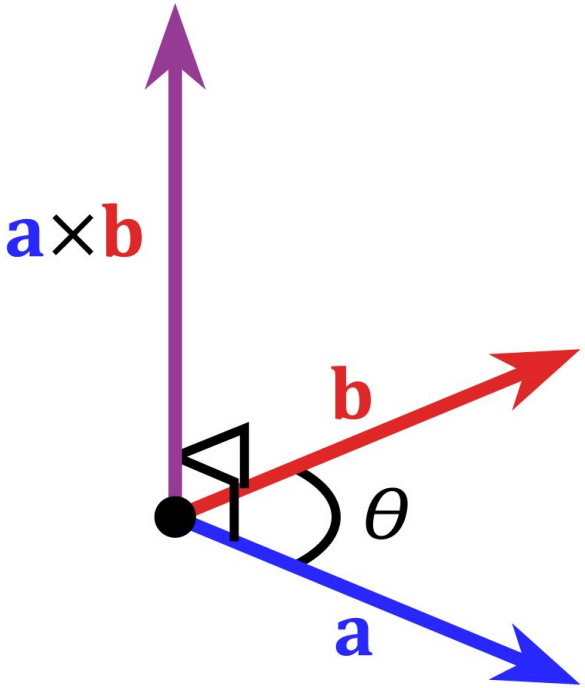
```
from random import random
from compas.geometry import pointcloud_xy
from compas_plotters import Plotter
from compas.utilities import i_to_green
from compas.utilities import ...

cloud = pointcloud_xy(20, (0, 10), (0, 5))

points = []
for xyz in cloud:
    n = ...
    points.append({
        'pos': xyz,
        'radius': ...,
        'edgecolor': ...,
        'facecolor': ...})

plotter = Plotter(figsize=(8, 5))
plotter.draw_points(points)
plotter.show()
```

```
from random import random
from compas.geometry import pointcloud_xy
from compas_plotters import Plotter
from compas.utilities import i_to_green
from compas.utilities import i_to_red

cloud = pointcloud_xy(20, (0, 10), (0, 5))

points = []
for xyz in cloud:
    n = random()
    points.append({
        'pos': xyz,
        'radius': n,
        'edgecolor': i_to_green(n),
        'facecolor': i_to_red(1 - n)})

plotter = Plotter(figsize=(8, 5))
plotter.draw_points(points)
plotter.show()
```

Part 1: Geometry

Cross Product

```
from compas.geometry import area_polygon
from compas.geometry import centroid_polygon
from compas.geometry import centroid_polyhedron
from compas.geometry import circle_from_points
from compas.geometry import convex_hull
from compas.geometry import decompose_matrix
from compas.geometry import distance_point_line
from compas.geometry import distance_line_line
from compas.geometry import intersection_line_line
from compas.geometry import intersection_line_triangle
from compas.geometry import intersection_plane_plane
from compas.geometry import is_coplanar
from compas.geometry import is_point_in_triangle
from compas.geometry import local_to_world_coords
from compas.geometry import matrix_from_basis_vectors
from compas.geometry import normal_polygon
from compas.geometry import normal_triangle
from compas.geometry import offset_line
from compas.geometry import orient_points
from compas.geometry import orthonormalize_axes
from compas.geometry import plane_from_points
from compas.geometry import reflect_line_triangle
from compas.geometry import volume_polyhedron
from compas.geometry import world_to_local_coords
...
```


a×b
b
θ
a

```
a = [ax, ay, az]
b = [bx, by, bz]


axb = [ay * bz - az * by, az * bx - ax * bz, ax * by - ay * bx]

axb = [a[1] * b[2] - a[2] * b[1],
       a[2] * b[0] - a[0] * b[2],
       a[0] * b[1] - a[1] * b[0]]
```

```
a = [ax, ay, az]
b = [bx, by, bz]


axb = [ay * bz - az * by, az * bx - ax * bz, ax * by - ay * bx]

axb = [a[1] * b[2] - a[2] * b[1],
       a[2] * b[0] - a[0] * b[2],
       a[0] * b[1] - a[1] * b[0]]


def cross_vectors(u, v):
    return [u[1] * v[2] - u[2] * v[1],
            u[2] * v[0] - u[0] * v[2],
            u[0] * v[1] - u[1] * v[0]]
```

```python
from compas.geometry import cross_vectors

u = [1.0, 0.0, 0.0]
v = [0.0, 1.0, 0.0]

uxv = cross_vectors(u, v)

print(uxv)
```

```python
from compas.geometry import Vector

u = Vector(1.0, 0.0, 0.0)
v = Vector(0.0, 1.0, 0.0)

uxv = u.cross(v)

print(uxv)
```

```python
from compas.geometry import cross_vectors

u = [1.0, 0.0, 0.0]
v = [0.0, 1.0, 0.0]

uxv = cross_vectors(u, v)

print(uxv)
```

```python
from compas.geometry import Vector

u = Vector(1.0, 0.0, 0.0)
v = Vector(0.0, 1.0, 0.0)

uxv = u.cross(v)

print(uxv)
```

```python
from compas.geometry import cross_vectors

u = [1.0, 0.0, 0.0]
v = [0.0, 1.0, 0.0]

uxv = cross_vectors(u, v)

print(uxv)
```

```python
from compas.geometry import Vector

u = Vector(1.0, 0.0, 0.0)
v = Vector(0.0, 1.0, 0.0)

uxv = u.cross(v)

print(uxv)
```

```python
from compas.geometry import cross_vectors

u = [1.0, 0.0, 0.0]
v = [0.0, 1.0, 0.0]

uxv = cross_vectors(u, v)

print(uxv)
```

```python
from compas.geometry import Vector

u = Vector(1.0, 0.0, 0.0)
v = Vector(0.0, 1.0, 0.0)

uxv = u.cross(v)

print(uxv)
```

```
[0.0, 0.0, 1.0]
```

```
Vector(0.000, 0.000, 1.000)
```

```python
from compas.geometry import cross_vectors
from compas.geometry import angle_vectors

u = [1.0, 0.0, 0.0]
v = [0.0, 1.0, 0.0]

uxv = cross_vectors(u, v)

u_uxv = angle_vectors(u, uxv)
v_uxv = angle_vectors(v, uxv)

print(u_uxv)
print(v_uxv)
```

```
1.5707963267948966
1.5707963267948966
```

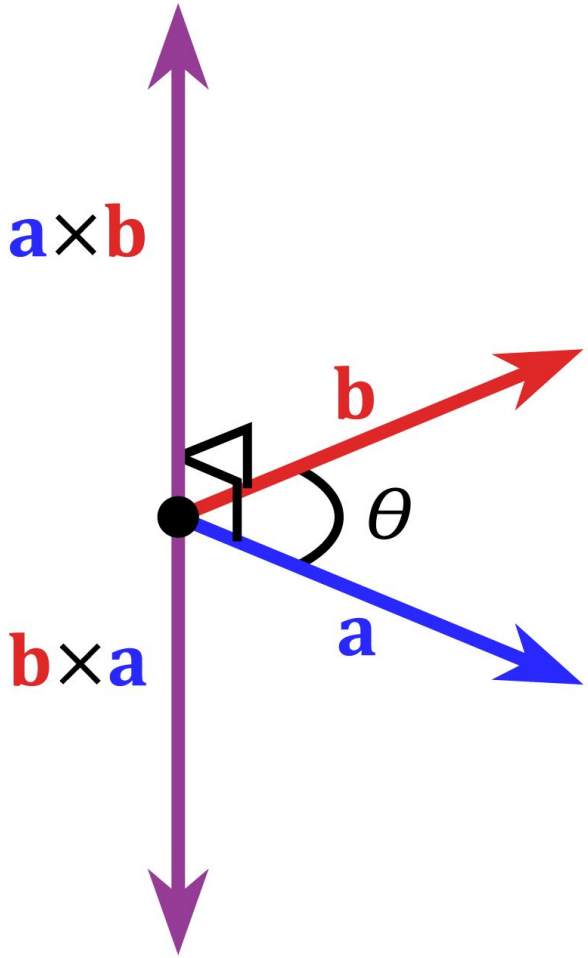
```python
from compas.geometry import Vector

u = Vector(1.0, 0.0, 0.0)
v = Vector(0.0, 1.0, 0.0)

uxv = u.cross(v)

u_uxv = u.angle(uxv)
v_uxv = v.angle(uxv)

print(u_uxv)
print(v_uxv)
```

```
1.5707963267948966
1.5707963267948966
```

Assignment

Given two vectors, use the cross product to create a set of three ortho**normal** vectors.

- orthonormal vectors are orthogonal and normalised!


a×b
b
θ
a
b×a

```python
from compas.geometry import cross_vectors

u = [1.0, 0.0, 0.0]
v = [0.0, 1.0, 0.0]

uxv = cross_vectors(u, v)
vxu = cross_vectors(v, u)

print(uxv)
print(vxu)
```

```
[0.0, 0.0, 1.0]
[0.0, 0.0, -1.0]
```

```python
from compas.geometry import Vector

u = Vector(1.0, 0.0, 0.0)
v = Vector(0.0, 1.0, 0.0)

uxv = u.cross(v)
vxu = v.cross(u)

print(uxv)
print(vxu)
```

```
Vector(0.000, 0.000, 1.000)
Vector(0.000, 0.000, -1.000)
```

Assignment (last week)

Write a function that determines if the rotation from one vector onto another is CCW or not.

```python
from compas.geometry import cross_vectors
from compas.geometry import angle_vectors

u = [1.0, 0.0, 0.0]
v = [0.0, 1.0, 0.0]

print(angle_vectors(u, v))
print(angle_vectors(v, u))
```

```python
from compas.geometry import Vector

u = Vector(1.0, 0.0, 0.0)
v = Vector(0.0, 1.0, 0.0)

print(u.angle(v))
print(v.angle(u))
```

```
1.5707963267948966
1.5707963267948966
```

```
1.5707963267948966
1.5707963267948966
```

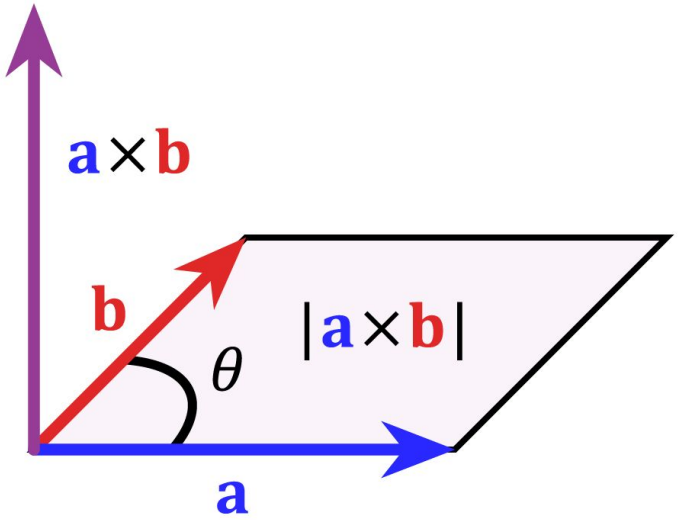
```python
from compas.geometry import cross_vectors

u = [1.0, 0.0, 0.0]
v = [0.0, 1.0, 0.0]

print(cross_vectors(u, v)[2] > 0)
print(cross_vectors(v, u)[2] > 0)
```

```
True
False
```

```python
from compas.geometry import Vector

u = Vector(1.0, 0.0, 0.0)
v = Vector(0.0, 1.0, 0.0)

print(u.cross(v)[2] > 0)
print(v.cross(u)[2] > 0)
```

```
True
False
```

```
def is_ccw_xy(a, b, c, colinear=False):
    """Determine if c is on the left of ab when looking from a to b,
    and assuming that all points lie in the XY plane.
    """
    ab_x = b[0] - a[0]
    ab_y = b[1] - a[1]
    ac_x = c[0] - a[0]
    ac_y = c[1] - a[1]

    if colinear:
        return ab_x * ac_y - ab_y  * ac_x >= 0
    return ab_x * ac_y - ab_y  * ac_x > 0
```


a×b
b
θ
|a×b|
a

```python
from compas.geometry import cross_vectors
from compas.geometry import length_vector

u = [1.0, 0.0, 0.0]
v = [0.0, 1.0, 0.0]

uxv = cross_vectors(u, v)

print(length_vector(uxv))
```

```
1.0
```

```python
from compas.geometry import Vector

u = Vector(1.0, 0.0, 0.0)
v = Vector(0.0, 1.0, 0.0)

uxv = u.cross(v)

print(uxv.length)
```

```
1.0
```

```python
from compas.geometry import subtract_vectors
from compas.geometry import cross_vectors
from compas.geometry import length_vector
from compas.geometry import area_triangle

a = [0.0, 0.0, 0.0]
b = [1.0, 0.0, 0.0]
c = [0.0, 1.0, 0.0]

ab = subtract_vectors(b, a)
ac = subtract_vectors(c, a)

L = length_vector(cross_vectors(ab, ac))
A = area_triangle([a, b, c])

print(0.5 * L == A)
```

```
True
```

```python
from compas.geometry import Vector
from compas.geometry import area_triangle

a = [0.0, 0.0, 0.0]
b = [1.0, 0.0, 0.0]
c = [0.0, 1.0, 0.0]

ab = Vector.from_start_end(a, b)
ac = Vector.from_start_end(a, c)

L = ab.cross(ac).length
A = area_triangle([a, b, c])

print(0.5 * L == A)
```

```
True
```

Assignment

Use the cross product to compute the area of a convex, 2D polygon.

- decompose the polygon into triangles from the centroid
- use the cross product to compute the total area

```
from compas.geometry import area_polygon
from compas.geometry import centroid_polygon
from compas.geometry import centroid_polyhedron
from compas.geometry import circle_from_points
from compas.geometry import convex_hull
from compas.geometry import decompose_matrix
from compas.geometry import distance_point_line
from compas.geometry import distance_line_line
from compas.geometry import intersection_line_line
from compas.geometry import intersection_line_triangle
from compas.geometry import intersection_plane_plane
from compas.geometry import is_coplanar
from compas.geometry import is_point_in_triangle
from compas.geometry import local_to_world_coords
from compas.geometry import matrix_from_basis_vectors
from compas.geometry import normal_polygon
from compas.geometry import normal_triangle
from compas.geometry import offset_line
from compas.geometry import orient_points
from compas.geometry import orthonormalize_axes
from compas.geometry import plane_from_points
from compas.geometry import reflect_line_triangle
from compas.geometry import volume_polyhedron
from compas.geometry import world_to_local_coords
...
```

Object Aligned Bounding Box using PCA

PCA of a dataset

finds the directions

along which the variation of the data

is largest,

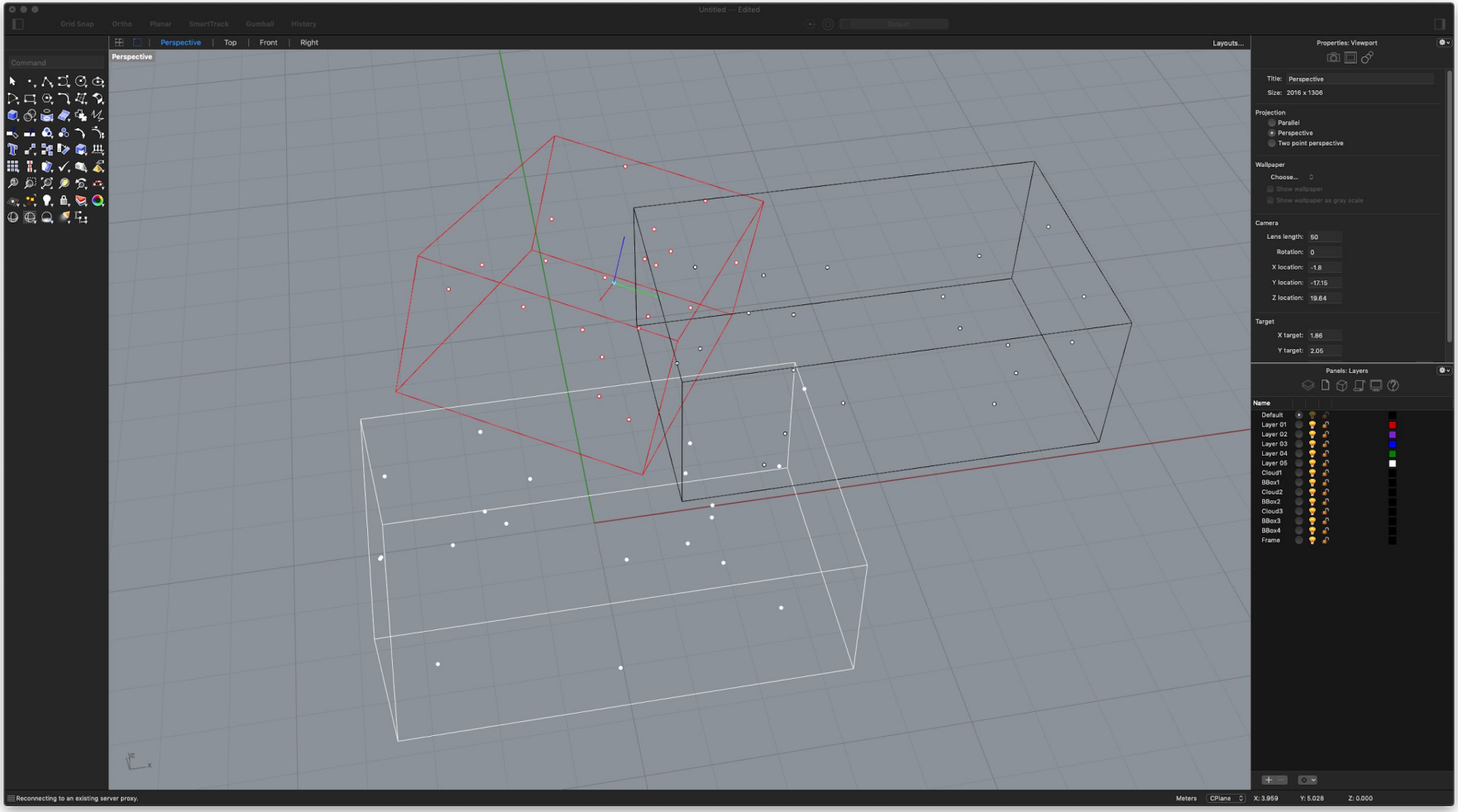
i.e. the directions along which the data is most spread out.

```python
from math import radians

import compas_rhino

from compas.geometry import pointcloud
from compas.geometry import bounding_box
from compas.geometry import Frame
from compas.geometry import Rotation
from compas.geometry import Transformation
from compas.geometry import transform_points

from compas.utilities import pairwise

# from compas.numerical import pca_numpy
from compas.rpc import Proxy

numerical = Proxy('compas.numerical')
```

```python
# ==============================================================================
# Helpers
# ==============================================================================

def draw_cloud(cloud, color, layer):
    points = []
    for xyz in cloud:
        points.append({'pos': xyz, 'color': color})
    compas_rhino.draw_points(points, layer=layer, clear=True)

def draw_bbox(bbox, color, layer):
    lines = []
    for a, b in pairwise(bbox[:4] + bbox[:1]):
        lines.append({'start': a, 'end': b, 'color': color})
    for a, b in pairwise(bbox[4:] + bbox[4:5]):
        lines.append({'start': a, 'end': b, 'color': color})
    for a, b in zip(bbox[:4], bbox[4:]):
        lines.append({'start': a, 'end': b, 'color': color})
    compas_rhino.draw_lines(lines, layer=layer, clear=True)

def draw_frame(frame, layer):
    ...
```

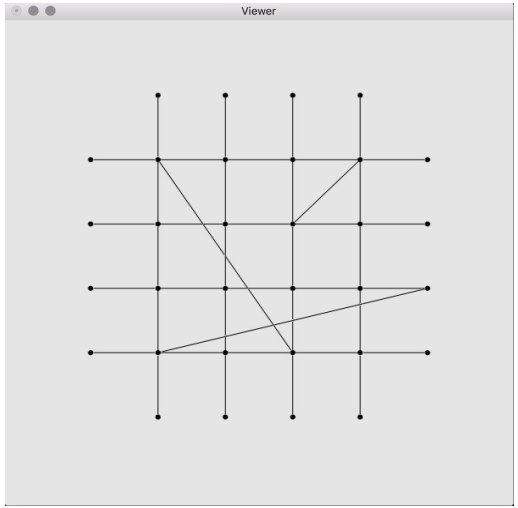
```
# ==============================================================================
# Algorithm
# ==============================================================================

cloud1 = pointcloud(20, (0, 10), (0, 5), (0, 3))
bbox1 = bounding_box(cloud1)

Rz = Rotation.from_axis_and_angle([0.0, 0.0, 1.0], radians(30))
Ry = Rotation.from_axis_and_angle([0.0, 1.0, 0.0], radians(20))
Rx = Rotation.from_axis_and_angle([1.0, 0.0, 0.0], radians(40))

R = Rz * Ry * Rz

cloud2 = transform_points(cloud1, R)
bbox2 = transform_points(bbox1, R)

average, vectors, values = numerical.pca_numpy(cloud2)

origin = average[0]
xaxis = vectors[0]
yaxis = vectors[1]
zaxis = vectors[2]
...
```

```
# ================================================================================
# Visualisation
# ================================================================================

draw_cloud(cloud1, (0, 0, 0), "Cloud1")
draw_bbox(bbox1, (0, 0, 0), "BBox1")

draw_cloud(cloud2, (255, 0, 0), "Cloud2")

draw_cloud(cloud3, (255, 255, 255), "Cloud3")
draw_bbox(bbox3, (255, 255, 255), "BBox3")

draw_bbox(bbox4, (255, 0, 0), "BBox4")

draw_frame(frame, "Frame")
```


Untitled — Edited
Grid Snap
Ortho
Planar
SmartTrack
Gumball
History
Command
Perspective
Top
Front
Right
Layouts...
Perspective
Properties: Viewport
Title: Perspective
Size: 2016 x 1306
Projection
Parallel
Perspective
Two point perspective
Wallpaper
Choose...
Show wallpaper
Show wallpaper as gray scale
Camera
Lens length: 50
Rotation: 0
X location: -1.8
Y location: -17.15
Z location: 19.64
Target
X target: 1.86
Y target: 2.05
Panels: Layers
Name
Default
Layer 01
Layer 02
Layer 03
Layer 04
Layer 05
Cloud1
BBox1
Cloud2
BBox2
Cloud3
BBox3
BBox4
Frame
Reconnecting to an existing server proxy.
Meters
CPlane
X: 3.959
Y: 5.028
Z: 0.000

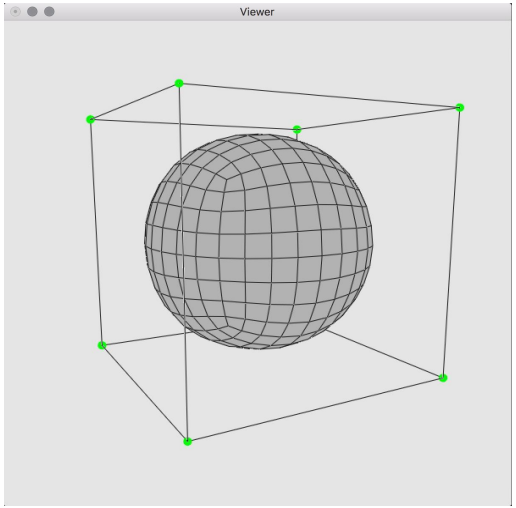
Part 2: Data Structures

Network



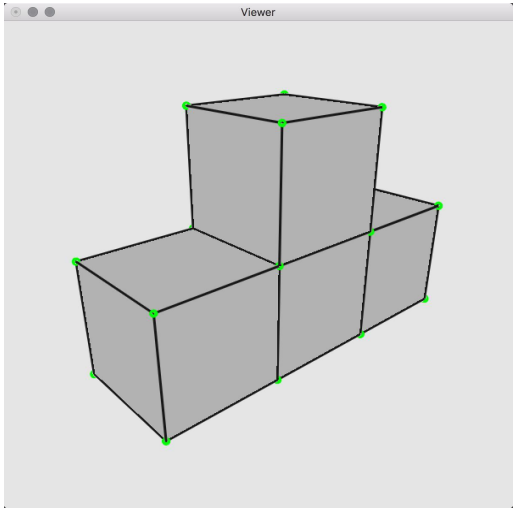
- General networks
- Edge graph

Mesh

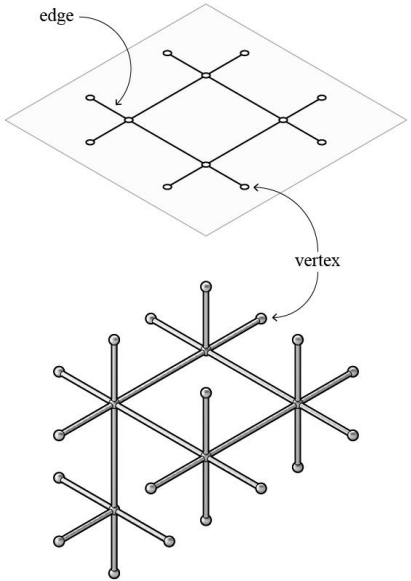


- Surface meshes
- Half-edge

VolMesh

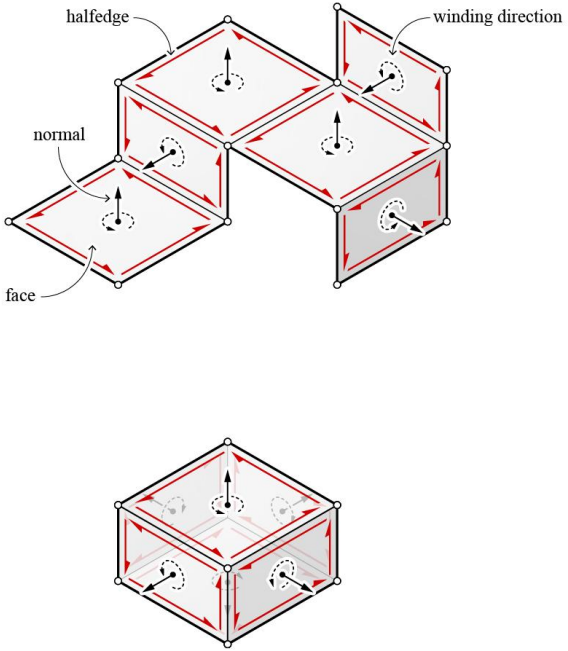


- Cellular meshes
- Half-plane



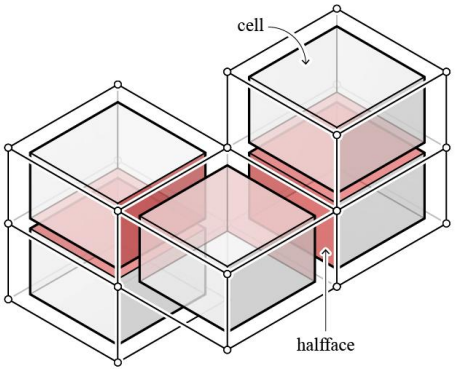
network

a network of vertices,
with vertex connectivities defined by directed edges



mesh

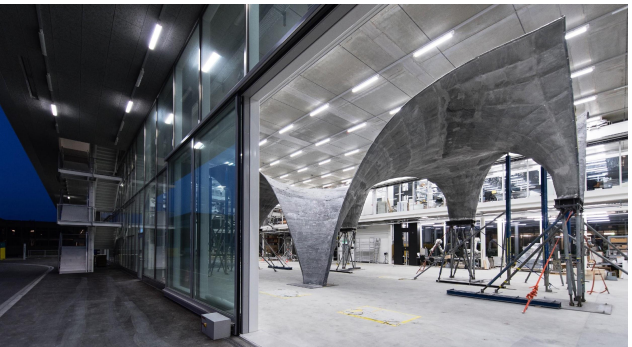
a network of faces,
with face connectivities defined by halfedge adjacencies

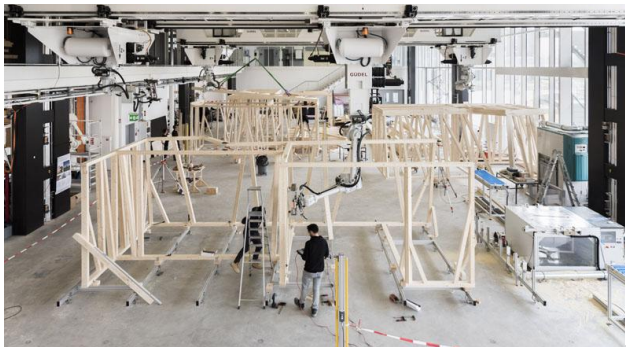


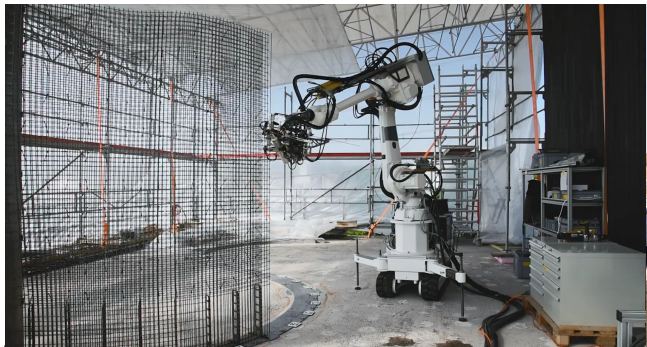
volmesh

a network of cells,
with cell connectivities defined by halfface adjacencies


FIRST
STREET
GARDEN
IDEAS
CITY
MAY 28-30
FESTIVAL FOR
THE FUTURE
ideas-city.org







Constructors

```python
import os
import compas
from compas.datastructures import Network

HERE = os.path.dirname(__file__)
DATA = os.path.join(HERE, 'data')
FILE = os.path.join(DATA, 'lines.obj')

network = Network.from_obj(FILE)

print(network.summary())
```

```
- name: Network
- vertices: 32
- edges: 43
- vertex degree: 1/5
```

```python
import os
import compas
from compas.datastructures import Mesh

HERE = os.path.dirname(__file__)
DATA = os.path.join(HERE, 'data')
FILE = os.path.join(DATA, 'faces.obj')

mesh = Mesh.from_obj(FILE)

print(mesh.summary())
```

```
- name: Mesh
- vertices: 36
- edges: 60
- faces: 25
- vertex degree: 2/4
- face degree: 2/4
```

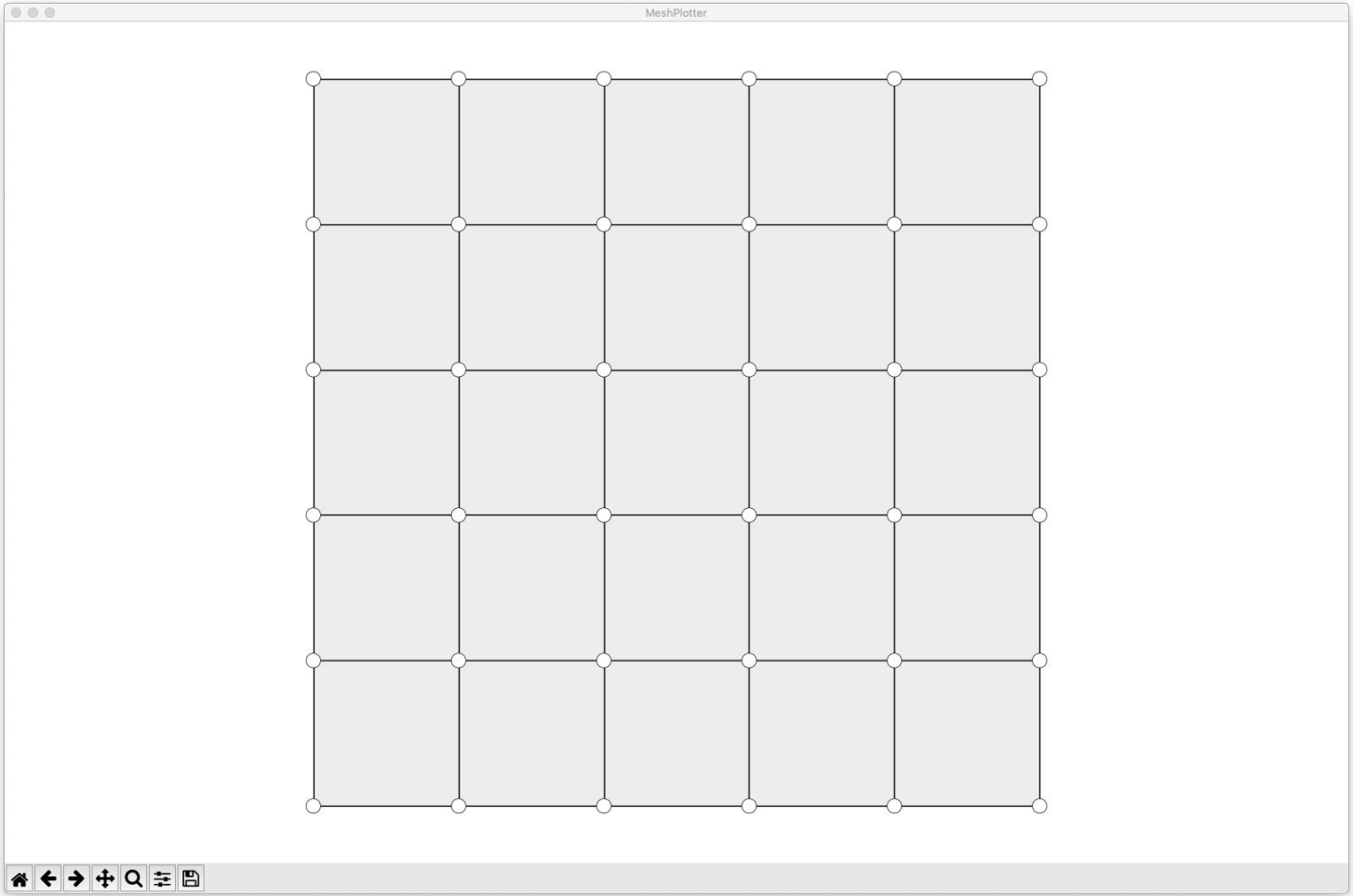
```python
from compas.datastructures import Network


network = Network.from_vertices_and_edges()

network = Network.from_lines()

network = Network.from_data()
network = Network.from_json()

network = Network.from_obj()
```

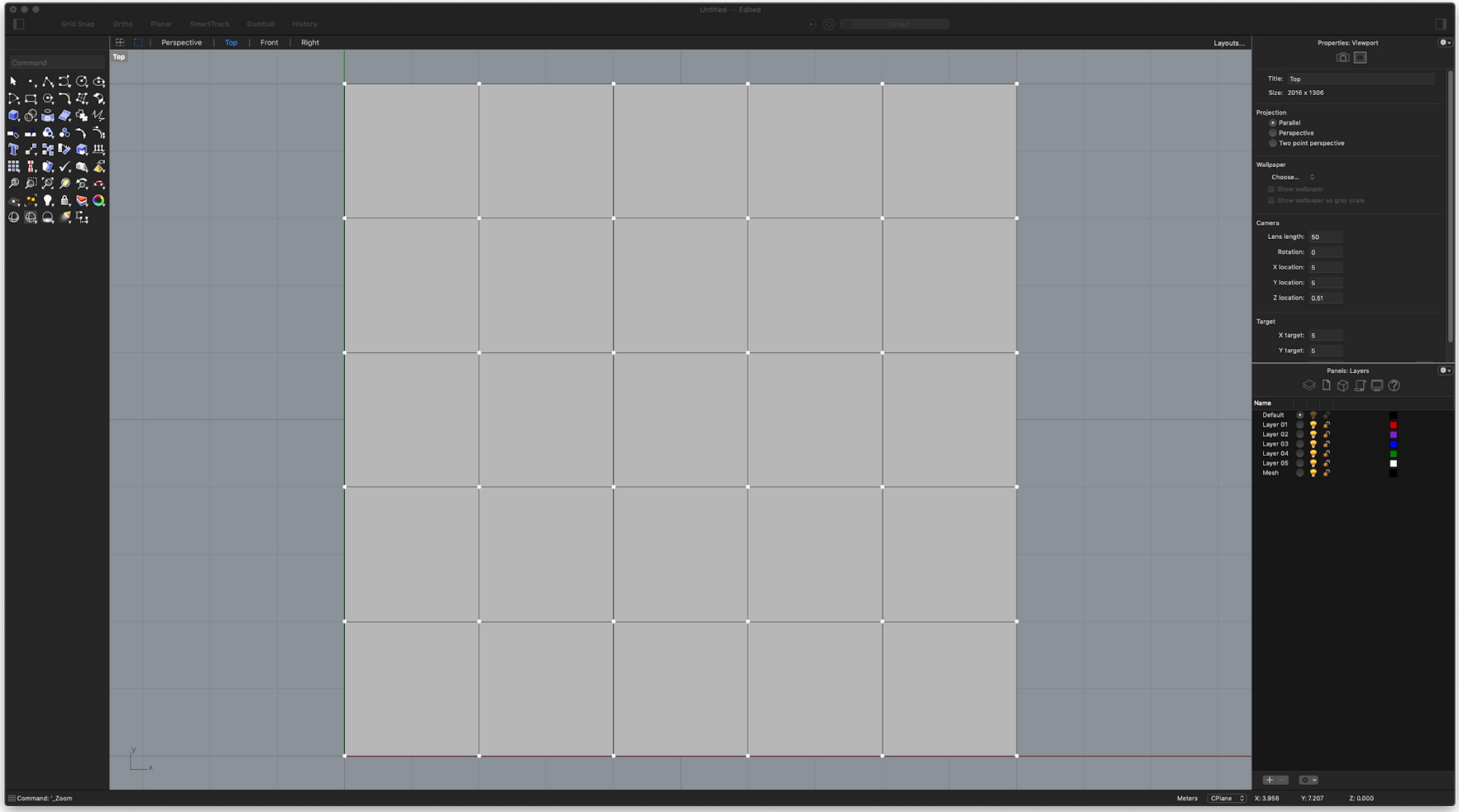
```python
from compas.datastructures import Mesh


mesh = Mesh.from_vertices_and_faces()

mesh = Mesh.from_lines()

mesh = Mesh.from_data()
mesh = Mesh.from_json()

mesh = Mesh.from_obj()
mesh = Mesh.from_off()
mesh = Mesh.from_stl()
mesh = Mesh.from_ply()
```

Visualisation

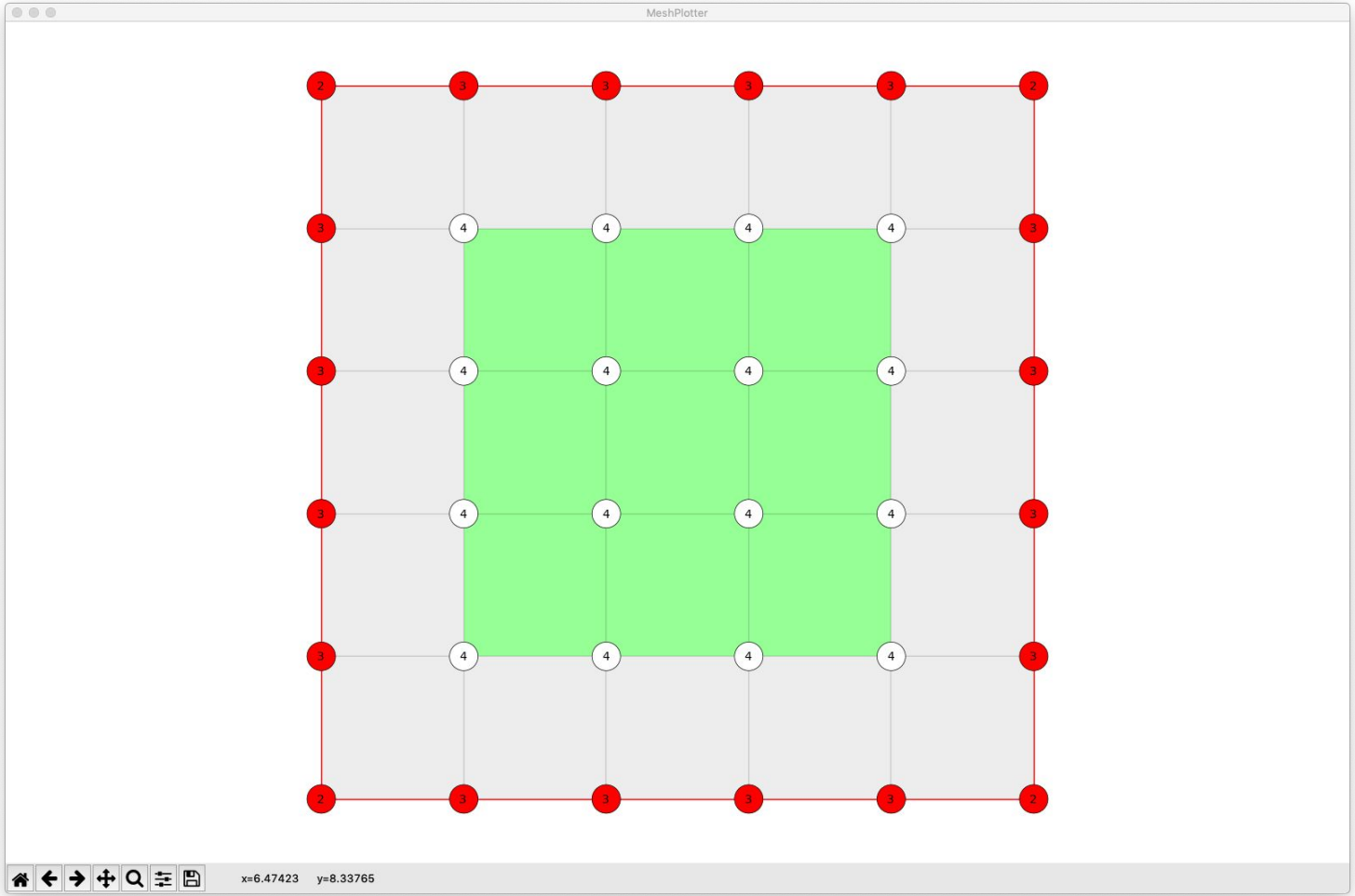
```python
import os
import compas

from compas.datastructures import Mesh
from compas_plotters import MeshPlotter

HERE = os.path.dirname(__file__)
DATA = os.path.join(HERE, 'data')
FILE = os.path.join(DATA, 'faces.obj')

mesh = Mesh.from_obj(FILE)

plotter = MeshPlotter(mesh, figsize=(8, 5))

plotter.draw_vertices()
plotter.draw_edges()
plotter.draw_faces()

plotter.show()
```

```python
import os
import compas

from compas.datastructures import Mesh
from compas_rhino.artists import MeshArtist

HERE = os.path.dirname(__file__)
DATA = os.path.join(HERE, 'data')
FILE = os.path.join(DATA, 'faces.obj')

mesh = Mesh.from_obj(FILE)

artist = MeshArtist(mesh, layer="Mesh")

artist.draw_vertices()
artist.draw_edges()
artist.draw_faces()
```

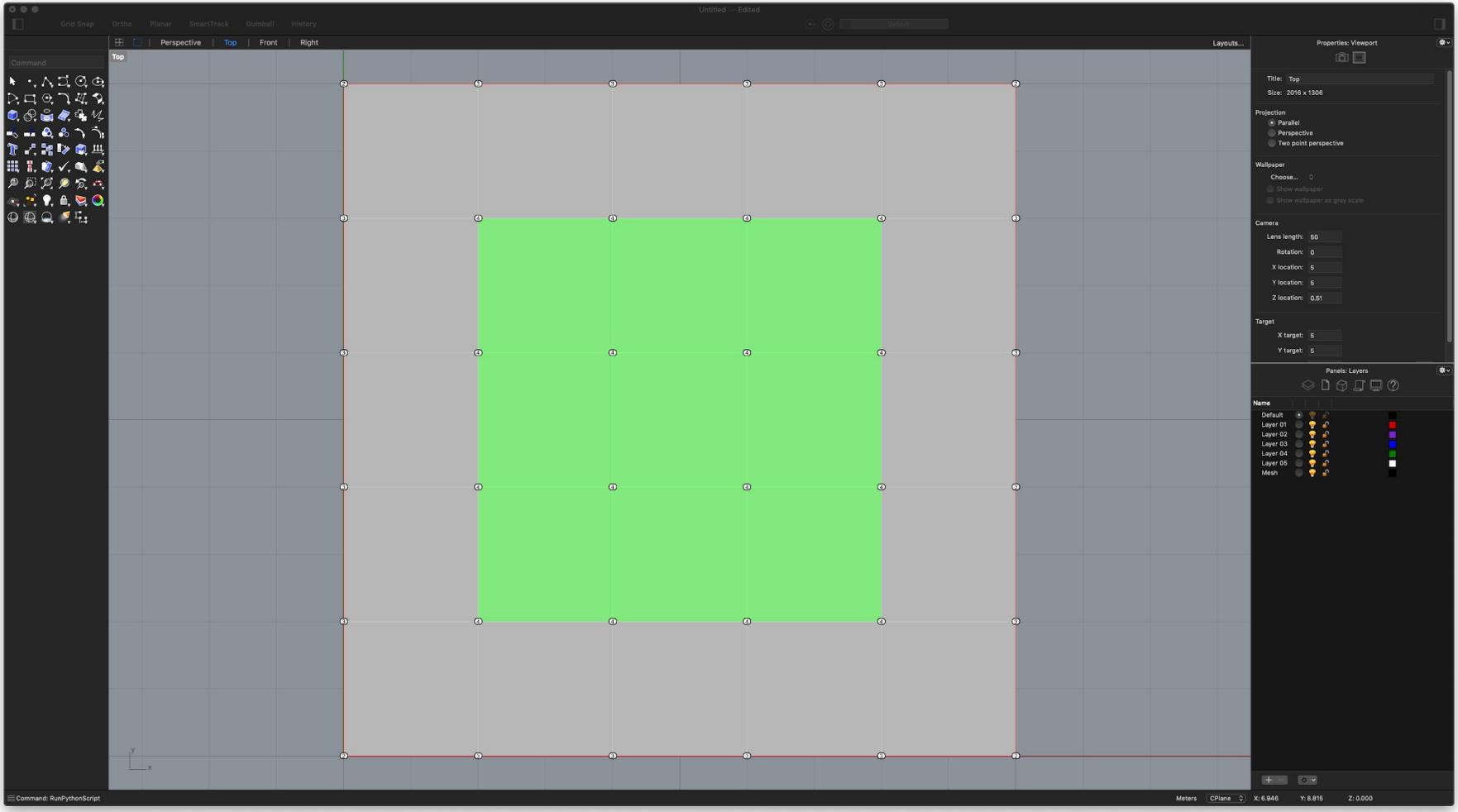

MeshPlotter


Untitled — Edited
Grid Snap
Ortho
Planar
SmartTrack
Gumball
History
Perspective
Top
Front
Right
Layouts...
Top
Command
Properties: Viewport
Title: Top
Size: 2016 x 1306
Projection
Parallel
Perspective
Two point perspective
Wallpaper
Choose...
Show wallpaper
Show wallpaper as gray scale
Camera
Lens length: 50
Rotation: 0
X location: 5
Y location: 5
Z location: 0.51
Target
X target: 5
Y target: 5
Panels: Layers
Name
Default
Layer 01
Layer 02
Layer 03
Layer 04
Layer 05
Mesh
Command: '_Zoom
Meters
CPlane
X: 3.956
Y: 7.207
Z: 0.000

```
mesh = Mesh.from_obj(FILE)

plotter = MeshPlotter(mesh, figsize=(16, 10))

plotter.draw_vertices(
    facecolor={key: (255, 0, 0) for key in mesh.vertices_on_boundary()},
    text={key: str(mesh.vertex_degree(key)) for key in mesh.vertices()},
    radius=0.2)

plotter.draw_edges(
    keys=list(mesh.edges_on_boundary()),
    color=(255, 0, 0))

plotter.draw_faces(
    facecolor={key: (150, 255, 150) for key in mesh.faces() if not mesh.is_face_on_boundary(key)})

plotter.show()
```


MeshPlotter
x=6.47423 y=8.33765

```
mesh = Mesh.from_obj(FILE)

artist = MeshArtist(mesh, layer="Mesh")

artist.draw_vertices(
    color={key: (255, 0, 0) for key in mesh.vertices_on_boundary()})

artist.draw_vertexlabels(
    text={key: str(mesh.vertex_degree(key)) for key in mesh.vertices()})

artist.draw_edges(
    keys=list(mesh.edges_on_boundary()),
    color=(255, 0, 0))

artist.draw_faces(
    color={key: (150, 255, 150) for key in mesh.faces() if not mesh.is_face_on_boundary(key)})
```


Untitled — Edited
Grid Snap
Ortho
Planar
SmartTrack
Gumball
History
Perspective
Top
Front
Right
Layouts...
Top
Command
Properties: Viewport
Title: Top
Size: 2016 x 1306
Projection
Parallel
Perspective
Two point perspective
Wallpaper
Choose...
Show wallpaper
Show wallpaper as gray scale
Camera
Lens length: 50
Rotation: 0
X location: 5
Y location: 5
Z location: 0.51
Target
X target: 5
Y target: 5
Panels: Layers
Name
Default
Layer 01
Layer 02
Layer 03
Layer 04
Layer 05
Mesh
Command: RunPythonScript
Meters
CPlane
X: 6.946
Y: 8.815
Z: 0.000

Accessing the data

```
mesh = Mesh.from_obj(FILE)

print(mesh.vertices())

print(mesh.faces())

print(mesh.edges())
```

```
<generator object Mesh.vertices at ... >
<generator object Mesh.faces at ... >
<generator object Mesh.edges at ... >
```

```
mesh = Mesh.from_obj(FILE)

for key in mesh.vertices():
    print(key)

for key in mesh.faces():
    print(key)

for key in mesh.edges():
    print(key)
```

```
0                0                (0, 1)
1                1                (0, 6)
2                2                (1, 7)
...              ...              ...
```

```
mesh = Mesh.from_obj(FILE)

print(list(mesh.vertices()))

print(list(mesh.faces()))

print(list(mesh.edges()))
```

```
[0, 1, 2, ... ]

[0, 1, 2, ... ]

[(0, 1), (0, 6), (1, 7), ... ]
```

```
mesh = Mesh.from_obj(FILE)

for key in mesh.vertices():
    print(key)

for key in mesh.faces():
    print(key)

for key in mesh.edges():
    print(key)
```

```
0                0                (0, 1)
1                1                (0, 6)
2                2                (1, 7)
...              ...              ...
```

Attributes

```python
mesh = Mesh.from_obj(FILE)

print(mesh.default_vertex_attributes)

print(mesh.default_face_attributes)

print(mesh.default_edge_attributes)
```

```
{'x': 0.0, 'y': 0.0, 'z': 0.0}

{}

{}
```

```python
mesh = Mesh.from_obj(FILE)

for key, attr in mesh.vertices(data=True):
    print(key, attr)

for key, attr in mesh.faces(data=True):
    print(key, attr)

for u, v, attr in mesh.edges(data=True):
    print((u, v), attr)
```

```
0 {'x': 0.0, 'y': 0.0, 'z': 0.0}
1 {'x': 2.0, 'y': 0.0, 'z': 0.0}
2 {'x': 4.0, 'y': 0.0, 'z': 0.0}
...
```

```python
mesh = Mesh.from_obj(FILE)

print(mesh.default_vertex_attributes)

print(mesh.default_face_attributes)

print(mesh.default_edge_attributes)
```

```
{'x': 0.0, 'y': 0.0, 'z': 0.0}

{}

{}
```

```python
mesh = Mesh.from_obj(FILE)

for key, attr in mesh.vertices(data=True):
    print(key, attr)

for key, attr in mesh.faces(data=True):
    print(key, attr)

for u, v, attr in mesh.edges(data=True):
    print((u, v), attr)
```

```
0 {}
1 {}
2 {}
...
```

```python
mesh = Mesh.from_obj(FILE)

print(mesh.default_vertex_attributes)

print(mesh.default_face_attributes)

print(mesh.default_edge_attributes)
```

```
{'x': 0.0, 'y': 0.0, 'z': 0.0}

{}

{}
```

```python
mesh = Mesh.from_obj(FILE)

for key, attr in mesh.vertices(data=True):
    print(key, attr)

for key, attr in mesh.faces(data=True):
    print(key, attr)

for u, v, attr in mesh.edges(data=True):
    print((u, v), attr)
```

```
(0, 1) {}
(0, 6) {}
(1, 7) {}
...
```

```python
mesh = Mesh.from_obj(FILE)

mesh.update_default_edge_attributes({
    'q': 1.0,
    'f': 0.0})

for u, v, attr in mesh.edges(data=True):
    print((u, v), attr)
```

```
(0, 1) {'q': 1.0, 'f': 0.0}
(0, 6) {'q': 1.0, 'f': 0.0}
(1, 7) {'q': 1.0, 'f': 0.0}
...
```

```python
mesh = Mesh.from_obj(FILE)

print(mesh.get_vertex_attribute(0, 'x'))

print(mesh.get_vertex_attributes(0, 'xyz'))

print(mesh.get_vertices_attribute('x'))

print(mesh.get_vertices_attributes('xyz'))
```

```
0.0

[0.0, 0.0, 0.0]

[0.0, 2.0, 4.0, ... ]

[[0.0, 0.0, 0.0], [2.0, 0.0, 0.0], [4.0, 0.0, 0.0], ... ]
```

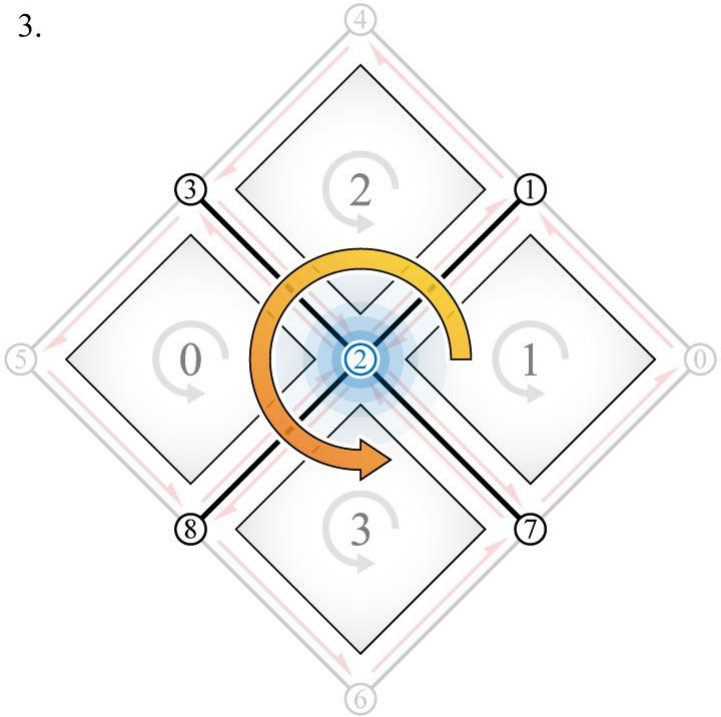
Topology vs Geometry

```python
mesh = Mesh.from_obj(FILE)

for key in mesh.vertices():
    print(mesh.vertex_coordinates(key))
    print(mesh.vertex_normal(key))
    print(mesh.vertex_area(key))

for fkey in mesh.faces():
    print(mesh.face_coordinates(fkey))
    print(mesh.face_normal(fkey))
    print(mesh.face_area(fkey))
```

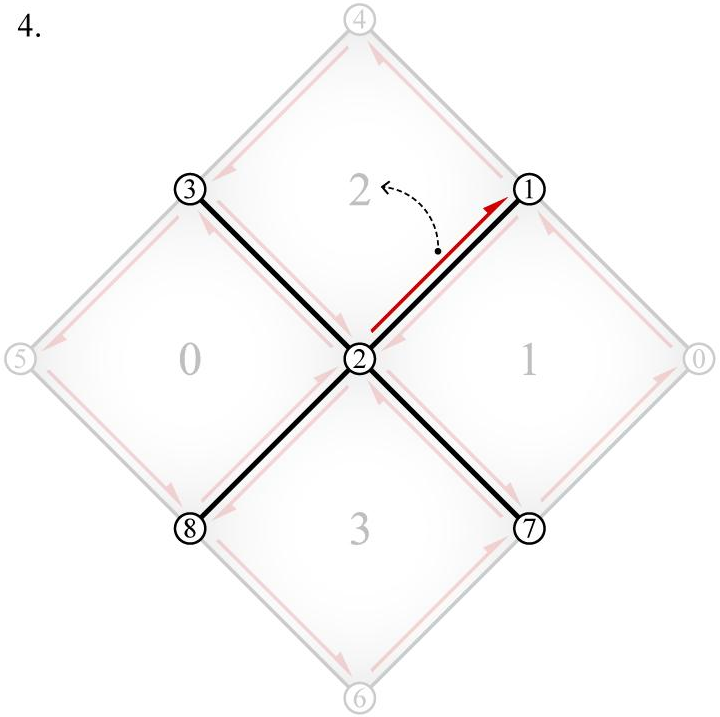
```python
mesh = Mesh.from_obj(FILE)

for key in mesh.vertices():
    print(mesh.vertex_neighbors(key))
    print(mesh.vertex_degree(key))
    print(mesh.vertex_neighborhood(key))
    print(mesh.vertex_faces(key))

for fkey in mesh.faces():
    print(mesh.face_vertices(fkey))
    print(mesh.face_neighbors(fkey))
    print(mesh.face_halfedges(fkey))

    for key in mesh.face_vertices(fkey):
        print(mesh.face_vertex_ancestor(fkey, key))
        print(mesh.face_vertex_descendant(fkey, key))
```

Mesh topology

3.

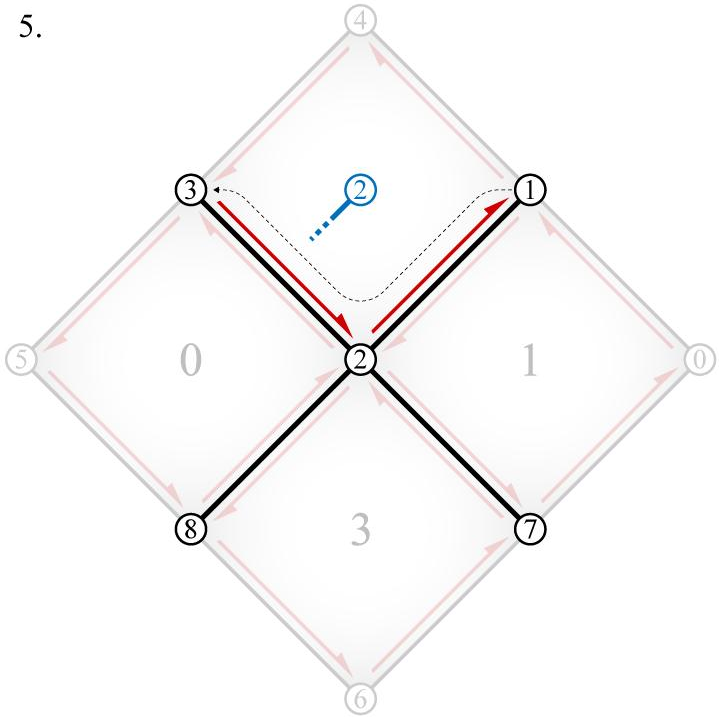


4.

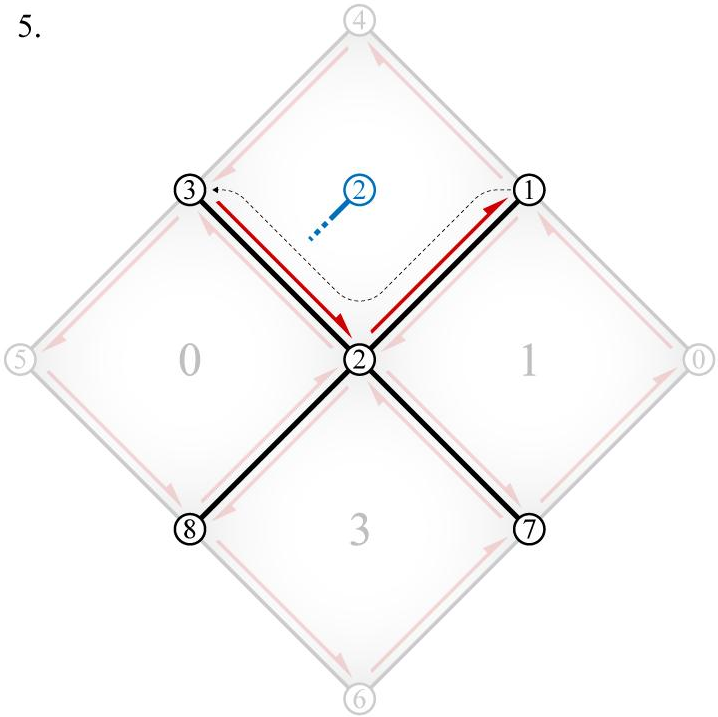


Mesh topology

5.

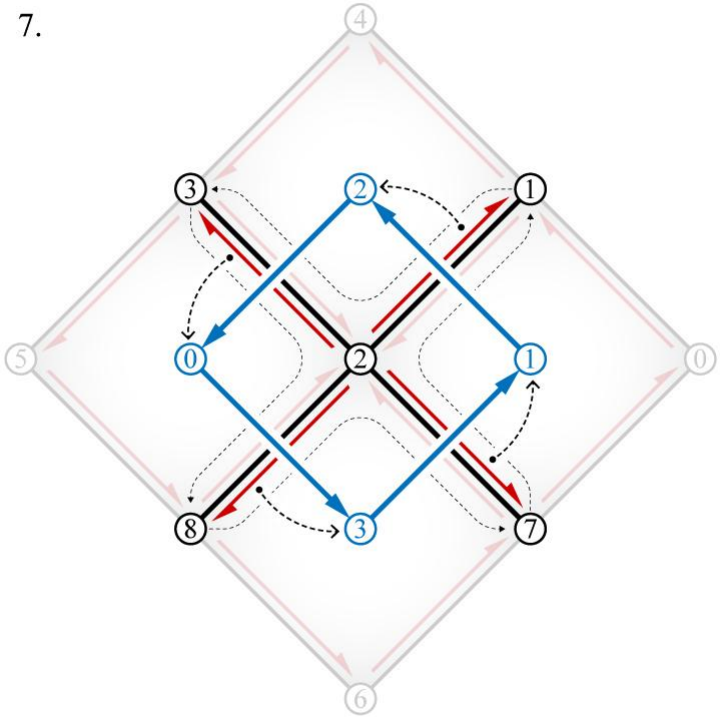


5.

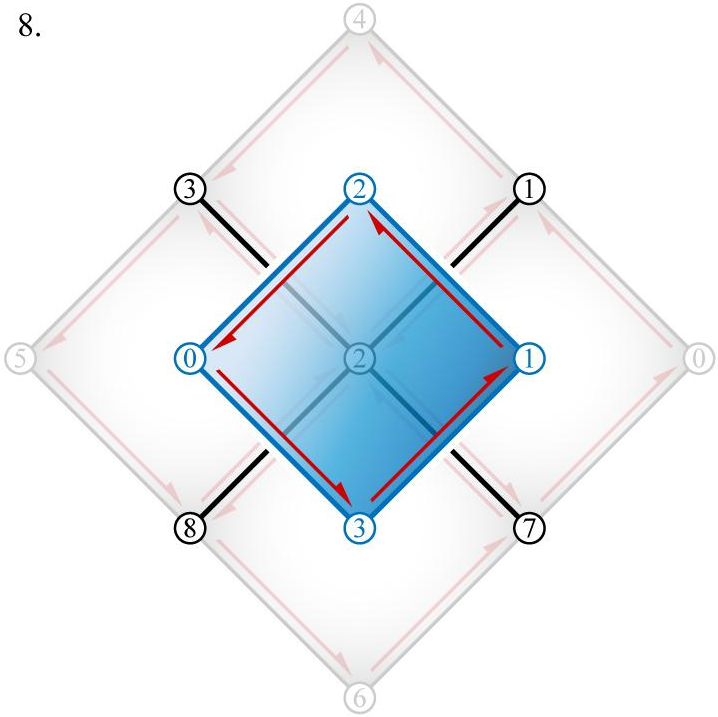


Mesh topology

7.

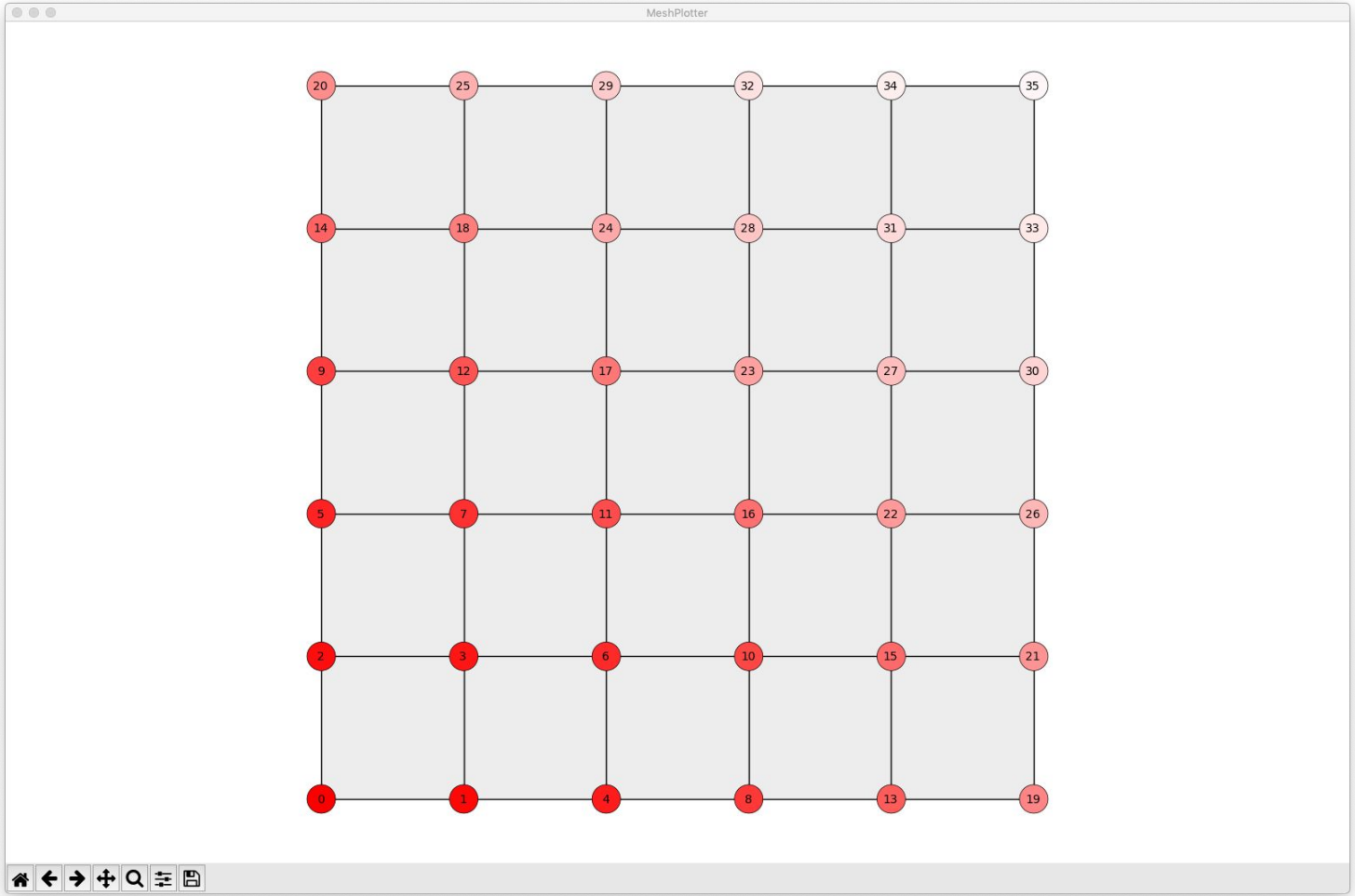


8.



```python
from compas.datastructures import Mesh
from compas.topology import breadth_first_ordering
from compas.utilities import i_to_red
from compas_plotters import MeshPlotter

HERE = os.path.dirname(__file__)
DATA = os.path.join(HERE, 'data')
FILE = os.path.join(DATA, 'faces.obj')

mesh = Mesh.from_obj(FILE)

bfo = breadth_first_ordering(mesh.adjacency, 0)

print(bfo)
```

```
[0, 1, 6, 7, ..., 35]
```


MeshPlotter
20
25
29
32
34
35
14
18
24
28
31
33
9
12
17
23
27
30
5
7
11
16
22
26
2
3
6
10
15
21
0
1
4
8
13
19

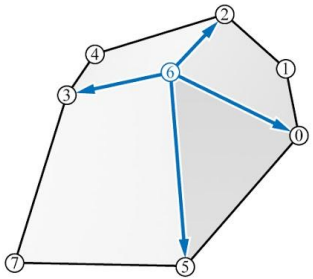
```python
from compas.datastructures import Mesh
from compas.topology import breadth_first_ordering
from compas.utilities import i_to_red
from compas_plotters import MeshPlotter

HERE = os.path.dirname(__file__)
DATA = os.path.join(HERE, 'data')
FILE = os.path.join(DATA, 'faces.obj')

mesh = Mesh.from_obj(FILE)

mesh.set_vertices_attributes('xyz', [0, 0, 0])

bfo = breadth_first_ordering(mesh.adjacency, 0)

print(bfo)
```

```
[0, 1, 6, 7, ..., 35]
```

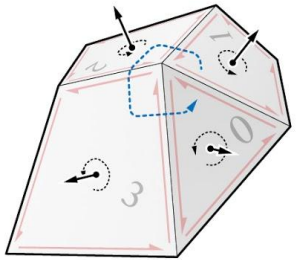
```python
from compas.datastructures import Mesh
from compas.topology import breadth_first_ordering
from compas.utilities import i_to_red
from compas_plotters import MeshPlotter

mesh = Mesh.from_obj(FILE)

bfo = breadth_first_ordering(mesh.adjacency, 0)

v = mesh.number_of_vertices()

plotter = MeshPlotter(mesh, figsize=(16, 10))

plotter.draw_vertices(
    text={key: index for index, key in enumerate(bfo)},
    radius=0.2,
    facecolor={key: i_to_red(1 - index / v) for index, key in enumerate(bfo)})

plotter.draw_edges()
plotter.draw_faces()

plotter.show()
```

Applications

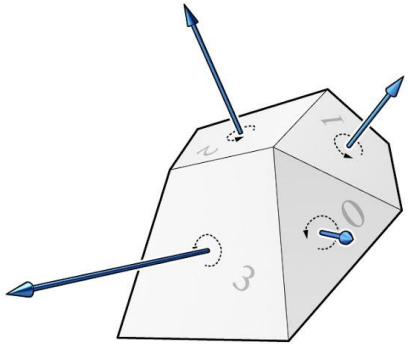
Mesh vertex normal



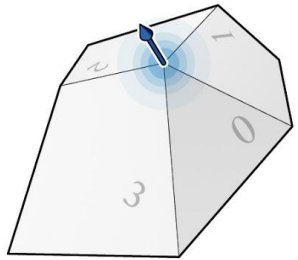
1. vertex neighbors



2. vertex faces

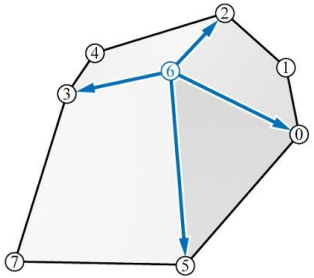


3. weighted face normals

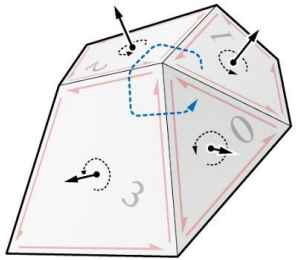


4. vertex normal
(unitized average of the weighted face normals)

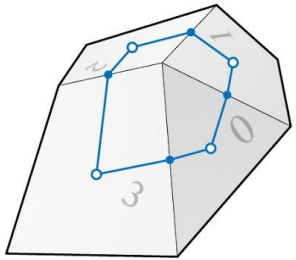
Mesh vertex tributary area



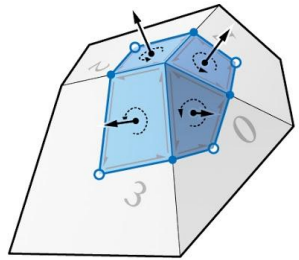
1. vertex neighbors



2. vertex faces

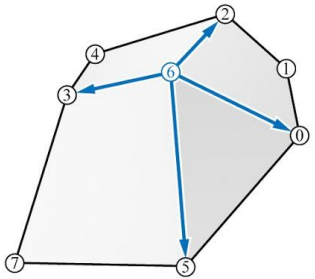


3. face centroids

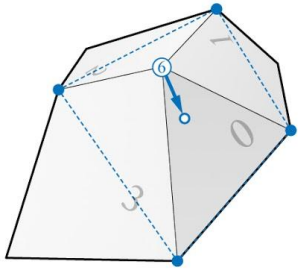


4. tributary area

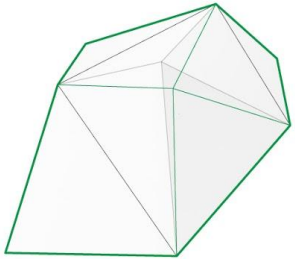
Mesh centroidal smoothing



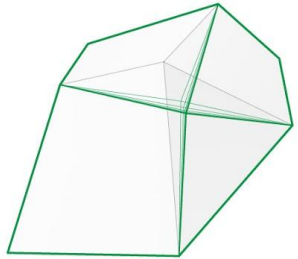
1. vertex neighbors



2. vertex neighbor centroid



3. k=1



4. k=10

Assignments

Geometry

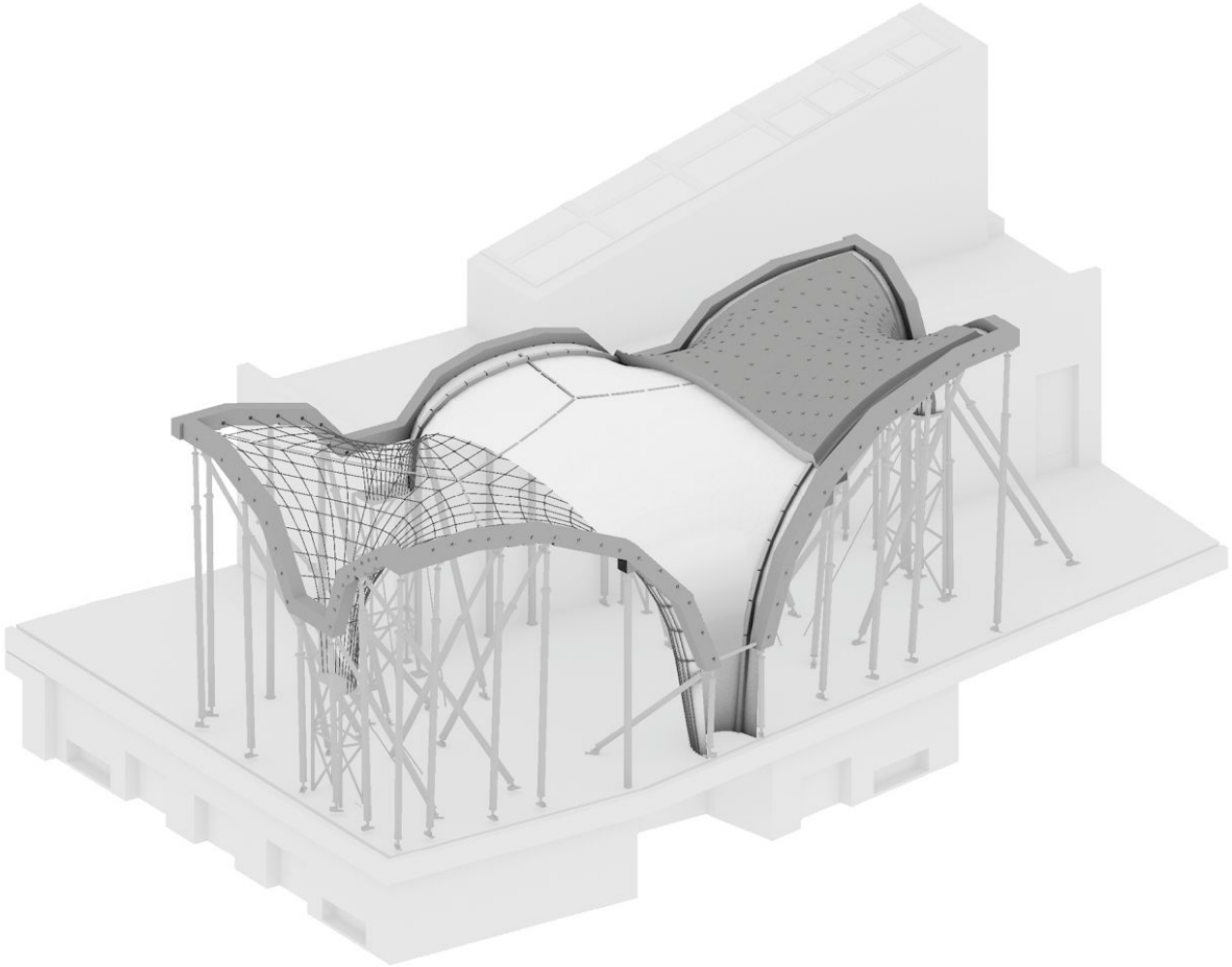
1. Given two vectors, use the cross product to create a set of three ortho**normal** vectors.
2. Use the cross product to compute the area of a convex, 2D polygon.

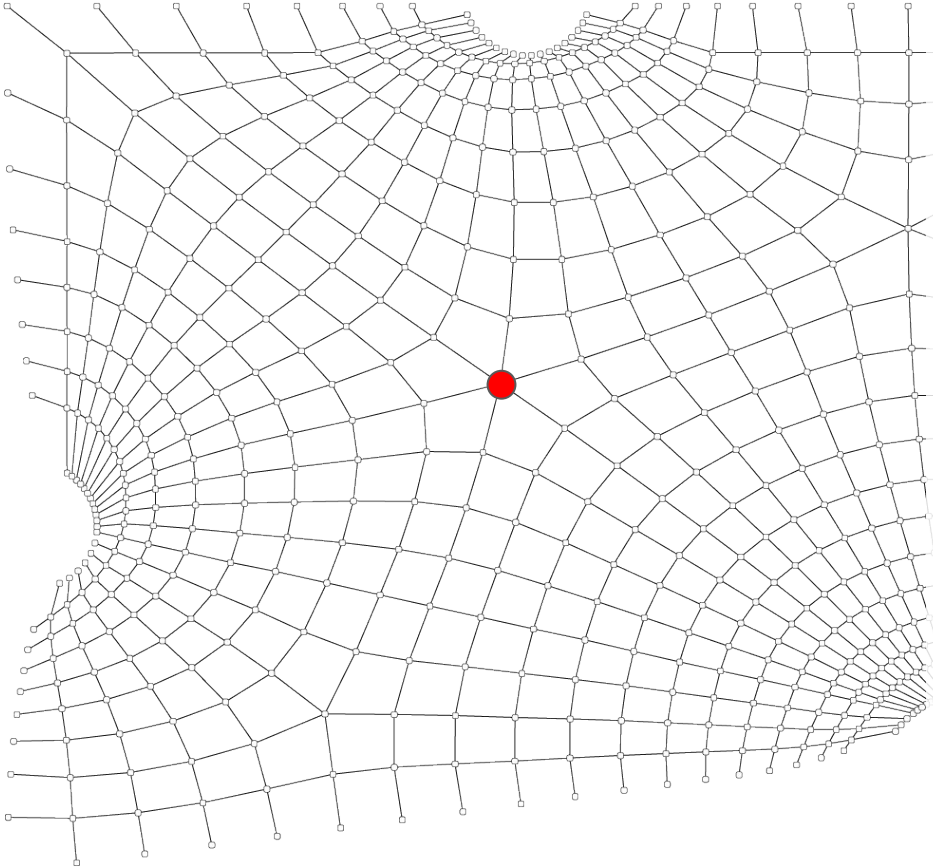
3. Define a function for computing the cross products of two arrays of vectors.
    a. The input arrays have the same length (same number of vectors).
    b. Prototype in pure Python (loop over the arrays).
    c. Make Numpy equivalent without loops.

Data Structures

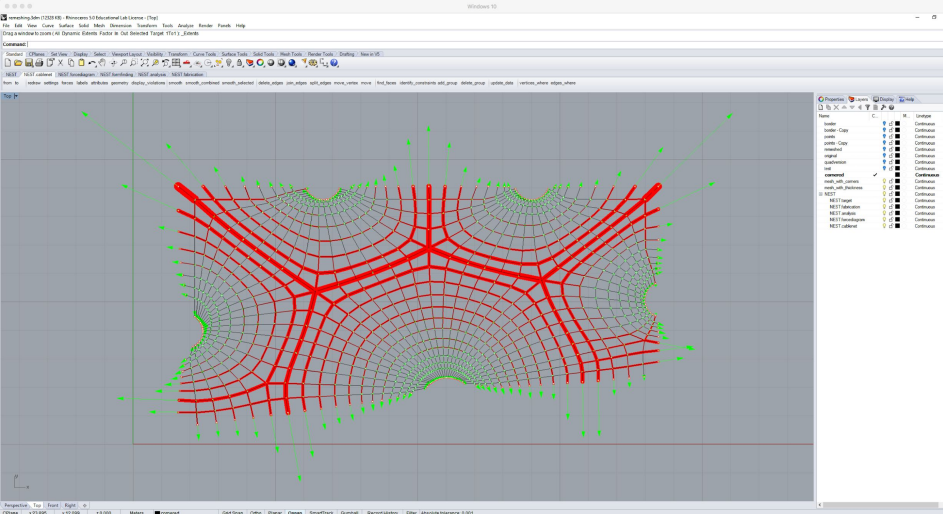
1. Using **faces.obj**
    a. Define a function for traversing the mesh from boundary to boundary in a “straight” line.
    b. Visualise the result.

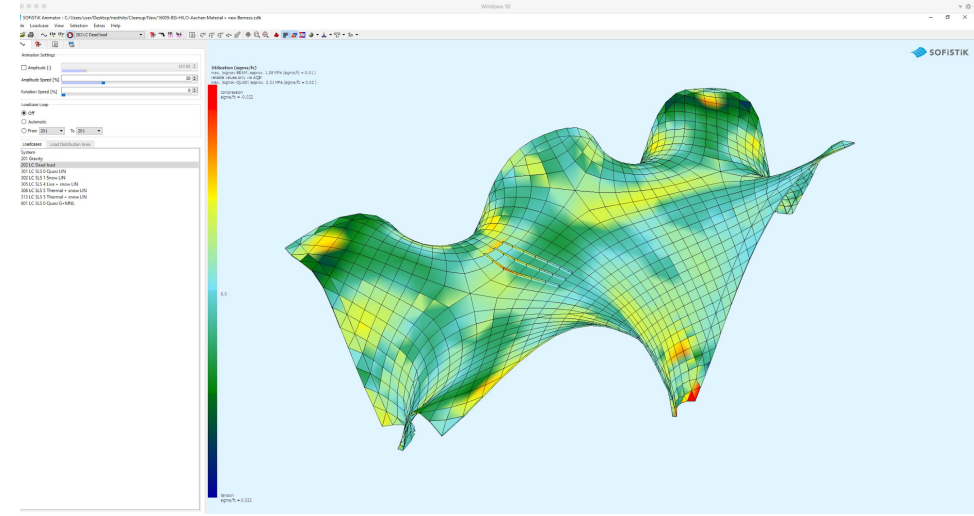
Next Week





```
cablenet.vertex[745] => {

  'constraint_name': 'target',
  'constraint_type': None,
  'cx'             : 0.950,
  'cy'             : 0.950,
  'damping'        : 1,
  'is_anchor'      : False,
  'is_constrained' : True,
  'is_fixed'       : False,
  'is_ridge'       : False,
  'px'             : 0.000,
  'py'             : 0.000,
  'pz'             : 0.000,
  'ring_type'      : 'XL',
  'rx'             : 0.000,
  'ry'             : 0.000,
  'rz'             : 0.000,
  'sag'            : 0.024,
  'sw1'            : 0.672,
  'sw2'            : 0.981,
  'x'              : 6.405,
  'y'              : 5.366,
  'z'              : 5.653,
```




SOFISTIK