```
# !pip install trimesh
!pip install git+https://github.com/2twenity/topologicpy.git#egg=topologicpy
```

Show hidden output

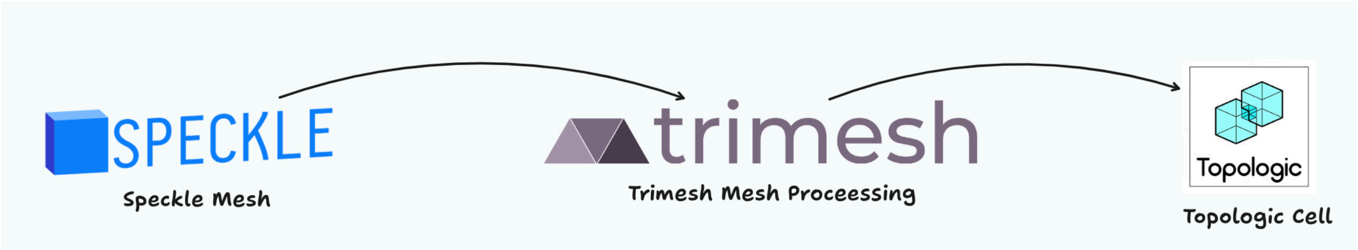
```
import numpy as np
# import trimesh


from topologicpy.Vertex import Vertex
from topologicpy.Face import Face
from topologicpy.Cell import Cell
from topologicpy.CellComplex import CellComplex
from topologicpy.Cluster import Cluster

from topologicpy.Topology import Topology
from topologicpy.Graph import Graph
from topologicpy.Dictionary import Dictionary
from topologicpy.Color import Color

from topologicpy.EnergyModel import EnergyModel
from topologicpy.Speckle import Speckle
# from topologicpy.Neo4j import Neo4j

# from scipy.spatial import ConvexHull
```



```
TOKEN = ""

client = Speckle.ClientByURL(url="https://app.speckle.systems/", token=TOKEN)

streams = Speckle.StreamsByClient(client)
stream = streams[1]

branches = Speckle.BranchesByStream(client=client, stream=stream)
branch = branches[0]

commits = Speckle.CommitsByBranch(branch=branch)
commit = commits[0]

obj = Speckle.SpeckleObject(client=client, stream=stream, branch=branch, commit=commit)

toplogy_obj = Speckle.TopologyBySpeckleObject(obj)

client = Speckle.ClientByURL(url="https://app.speckle.systems/", token=TOKEN)

streams = Speckle.StreamsByClient(client)
stream = streams[1]

branches = Speckle.BranchesByStream(client=client, stream=stream)
branch = branches[0]

commits = Speckle.CommitsByBranch(branch=branch)
commit = commits[0]

obj = Speckle.SpeckleObject(client=client, stream=stream, branch=branch, commit=commit)

topology_obj = Speckle.TopologyBySpeckleObject(obj)


first_obj = next(topology_obj)


print(first_obj)
```

```
<topologic_core.Cell object at 0x793726f479f0>
```

```
print(Dictionary.PythonDictionary(Topology.Dictionary(first_obj)))
```

```
{'applicationId': 'Object:IfcSlab/Basic Roof:RF_Stimip_50:398206', 'id': '6ed87750a5b4fbd9dea99f4c737c25fc', 'name': 'If
```

```
def get_category_from_name(name) ->  str:
        "Function is used to get clean speckle element name"

        import re
        mapping = {
            "IFC_WALL_REG": r"(Wall)",
            "IFC_SLAB_REG": r"(Slab)",
            "IFC_WINDOW_REG": r"(Window)",
            "IFC_COLUMN_REG": r"(Column)",
            "IFC_DOOR_REG": r"(Door)",
            "IFC_SPACE_REG": r"(Space)"
        }

        for expression in mapping.values():
            res = re.findall(expression, name)
            if res:
                return res[0]


data = {} #5 mins

for element in topology_obj:
  metadata = Dictionary.PythonDictionary(Topology.Dictionary(element))
  category = get_category_from_name(metadata['name'])
  metadata['category'] = category
  Topology.SetDictionary(element, metadata)


  data[metadata['name']] = element # Set full name of elements as a key for easier access
```

```
Cell.ByFaces - Error: The operation failed. Returning None.
Cell.ByFaces - Error: The operation failed. Returning None.
Cell.ByFaces - Error: The operation failed. Returning None.
Cell.ByFaces - Error: The operation failed. Returning None.
Cell.ByFaces - Error: The operation failed. Returning None.
Cell.ByFaces - Error: The operation failed. Returning None.
Cell.ByFaces - Error: The operation failed. Returning None.
Cell.ByFaces - Error: The operation failed. Returning None.
Cell.ByFaces - Error: The operation failed. Returning None.
Cell.ByFaces - Error: The operation failed. Returning None.
Cell.ByFaces - Error: The operation failed. Returning None.
Cell.ByFaces - Error: The operation failed. Returning None.
Cell.ByFaces - Error: The operation failed. Returning None.
Cell.ByFaces - Error: The operation failed. Returning None.
Cell.ByFaces - Error: The operation failed. Returning None.
Cell.ByFaces - Error: The operation failed. Returning None.
Cell.ByFaces - Error: The operation failed. Returning None.
Cell.ByFaces - Error: The operation failed. Returning None.
Cell.ByFaces - Error: The operation failed. Returning None.
```

```
     Cell.ByFaces - Error: The operation failed. Returning None.
     Cell.ByFaces - Error: The operation failed. Returning None.
     Cell.ByFaces - Error: The operation failed. Returning None.
     Cell.ByFaces - Error: The operation failed. Returning None.
     Cell.ByFaces - Error: The operation failed. Returning None.
```

```
print(data)
```

```
{'IfcSlab/Basic Roof:RF_Stimip_50:398206': <topologic_core.Cell object at 0x783178484530>, 'IfcSlab/Basic Roof:RF_Stimip
```

Overall Topology

Zones Selection

[ ] ↳ *16 cells hidden*

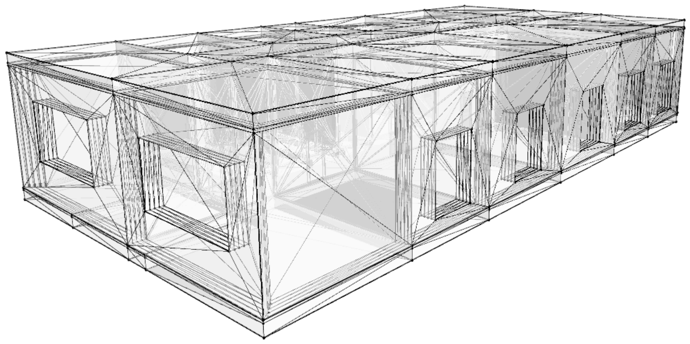
Some code

```
topologies_list = [elements_list_431 +
                   elements_list_432 +
                   elements_list_433 +
                   corridor_list_1 +
                   elements_list_434 +
                   elements_list_430 +
                   elements_list_429 +
                   elements_list_416 +
                   elements_list_400 +
                   elements_list_440 +
                   elements_list_441 +
                   elements_list_429 +
                   elements_list_439 +
                   elements_list_435 +
                   elements_list_462 +
                   elements_list_437][0]

print(len(topologies_list))
```

```
109
```

```
Topology.Show(topologies_list, renderer='colab')
```



```
cc_all = CellComplex.ByCells(topologies_list) #5mins 30sec
```

```
print(Dictionary.PythonDictionary(Topology.Dictionary(topologies_list[100])))
```

```
'Slab', 'id': '8e61aa8a0247685181429ad0f1da7215', 'name': 'IfcSlab/Floor:FL_Int_Stimip_52:397425', 'totalChildrenCount':
```

```
mapping_dict = Topology.TransferDictionaries(sources=topologies_list, sinks=CellComplex.Cells(cc_all))
```

```
print(mapping_dict.keys())
```

```
dict_keys(['sources', 'sinks'])
```

```
Topology.Show(cc_all, renderer='colab', silent=True)
```

Show hidden output

```
print(len(CellComplex.Cells(cc_all))) #107 #103
```

```
103
```

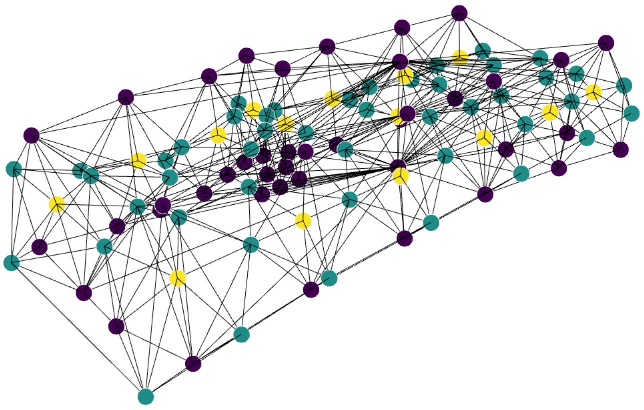
```
g = Graph.ByTopology(cc_all)
```

```
mapping_dict_2 = Topology.TransferDictionaries(sources=topologies_list, sinks=Graph.Vertices(g))
```

```
print(Dictionary.PythonDictionary(Topology.Dictionary(Graph.Vertices(g)[10])))
```

```
{'applicationId': 'Object:IfcWall/Basic Wall:WA_Ext_BrickWall_64-(2+1)_Plaster:395624', 'category': 'Wall', 'id': '36c7d
```

```
Topology.Show(g,
              renderer='colab',
              vertexSize=15,
              vertexLabelKey="full_name",
              vertexGroupKey="category",
              showVertexLabel=False,
              showVertexLegend=False,
              vertexGroups=['Slab', 'Wall', 'Space'])
```



```
print(len(Graph.Vertices(g))) #107
```

```
103
```

Fusion

Adjacent Spaces Identification

[ ] ↳ *11 cells hidden*

Fusion

[ ] ↳ *3 cells hidden*

› Neo4j

[ ] ↳ *2 cells hidden*