Programming Your Way up a Skyscraper

Coding in the Architecture Profession

TADEH HAKOPIAN

CONF42 PYTHON 2021

ABOUT ME

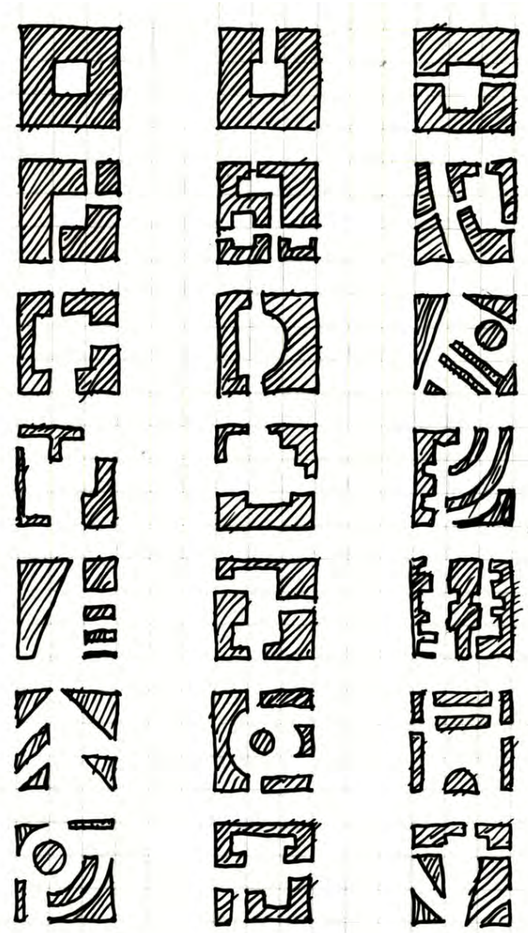
- Tadeh Hakopian
- (Todd-A) (Ha-co-pea-on)
- Design Technologist and Developer
- Background in Architecture
- Experience in Architecture, Engineering and Construction disciplines with BIM and VDC workflows
- Course Author and Speaker for BIM, Dynamo and Coding content

TAKEAWAY

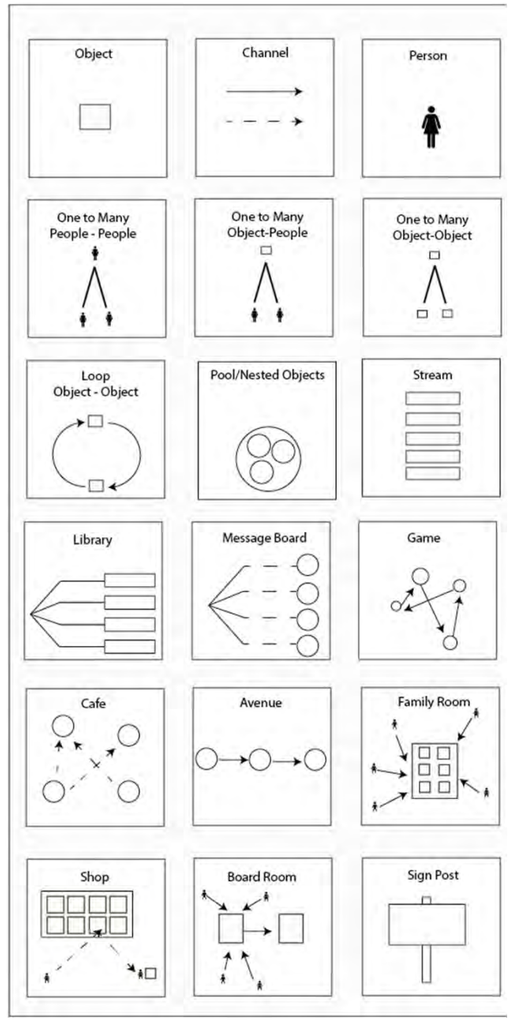
- Python Makes it possible!
- Showcase of how Python can support Architects and designers

ARCHITECTS AND ARCHITECTS

- Christopher Alexander’s Pattern Language has been a big influence
- In a nutshell a pattern can be **reused** at any **scale** without reinventing the whole process
- Used by Building Architects and Software Architects as a **philosophy for design**
- Common ground for systems architecture in both disciplines and a point of discussion



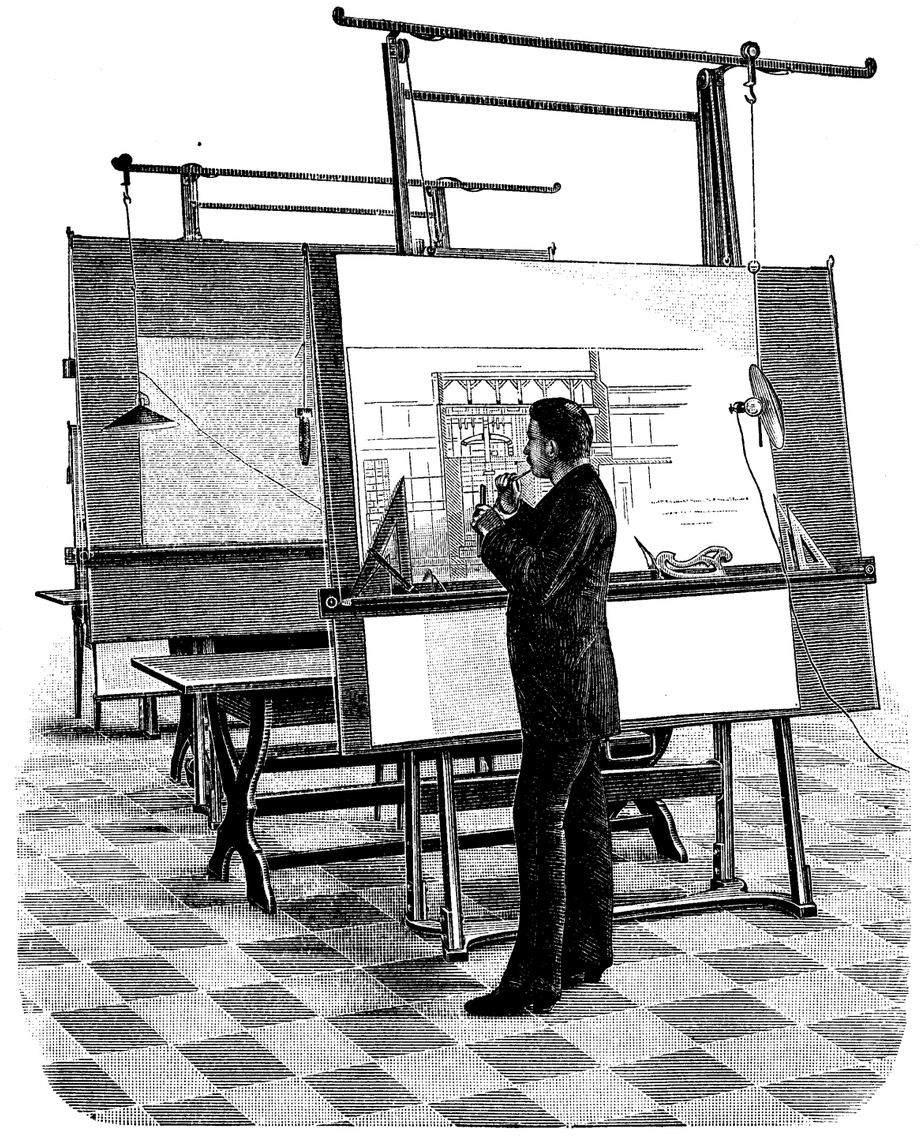
BUILDINGS



SYSTEMS

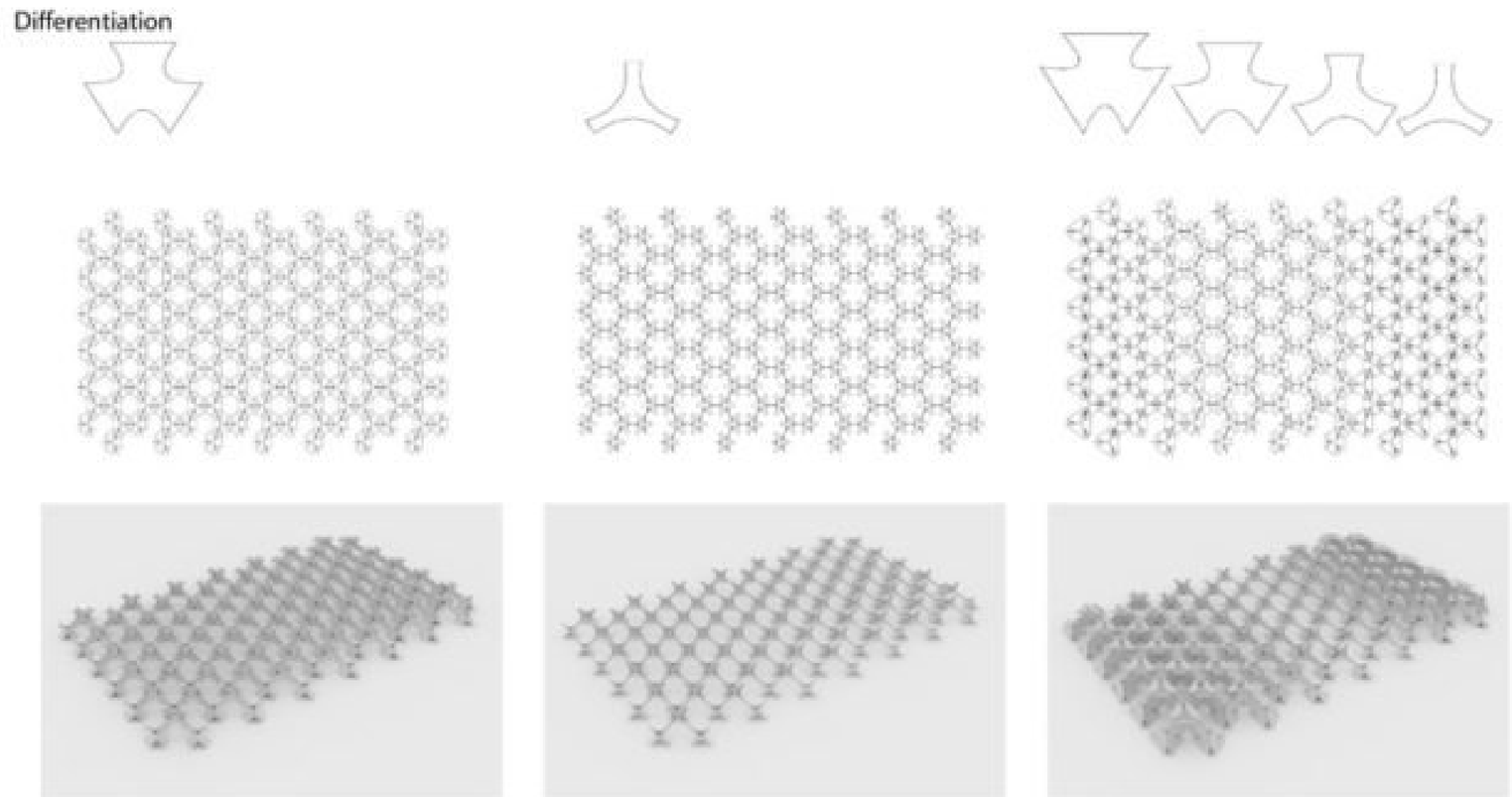
ARCHITECTURE TODAY

- How Architecture works
- Sketch an idea out
- Elaborate and draw plans for your design
- Update and keep going until you are done
- Worked back when you only needed 20 drawing plans for a house
- Not so great when you have 2000+ 30x40 inch sheets to make for today’s projects
- Wasteful and limited process



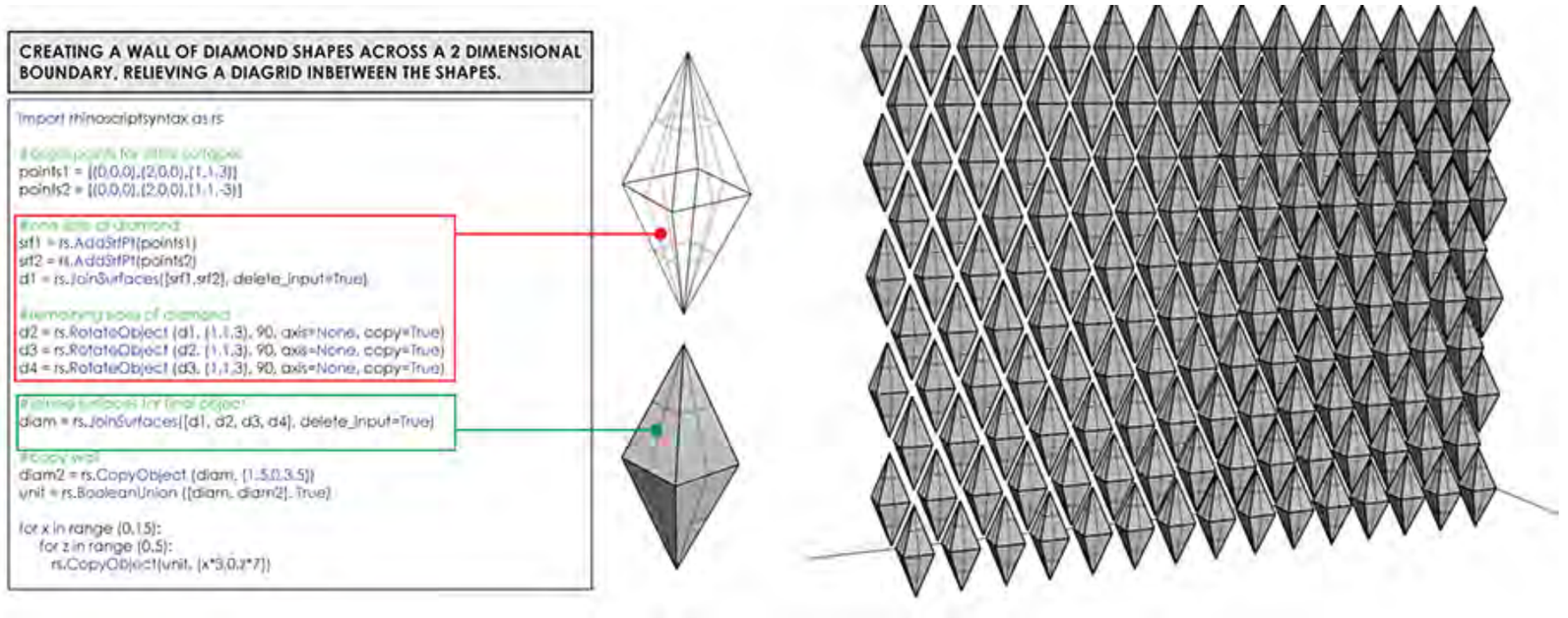
PYTHON IN ARCHITECTURE

- Architects are getting more interested in **computational** design tools
- Traditional methods of design and documentation are **not sufficient** for the needs of Architects today
- Python has been the common ground for a lot of Designers to modify the software themselves



PYTHON IN ARCHITECTURE

- Design patterns with **Code**
- Creating custom applications with **scripts**
- Patterns and Parametric designs possible through **computational** design
- **Python** is increasingly used for both practical **scripting** and intensive design with **algorithms**

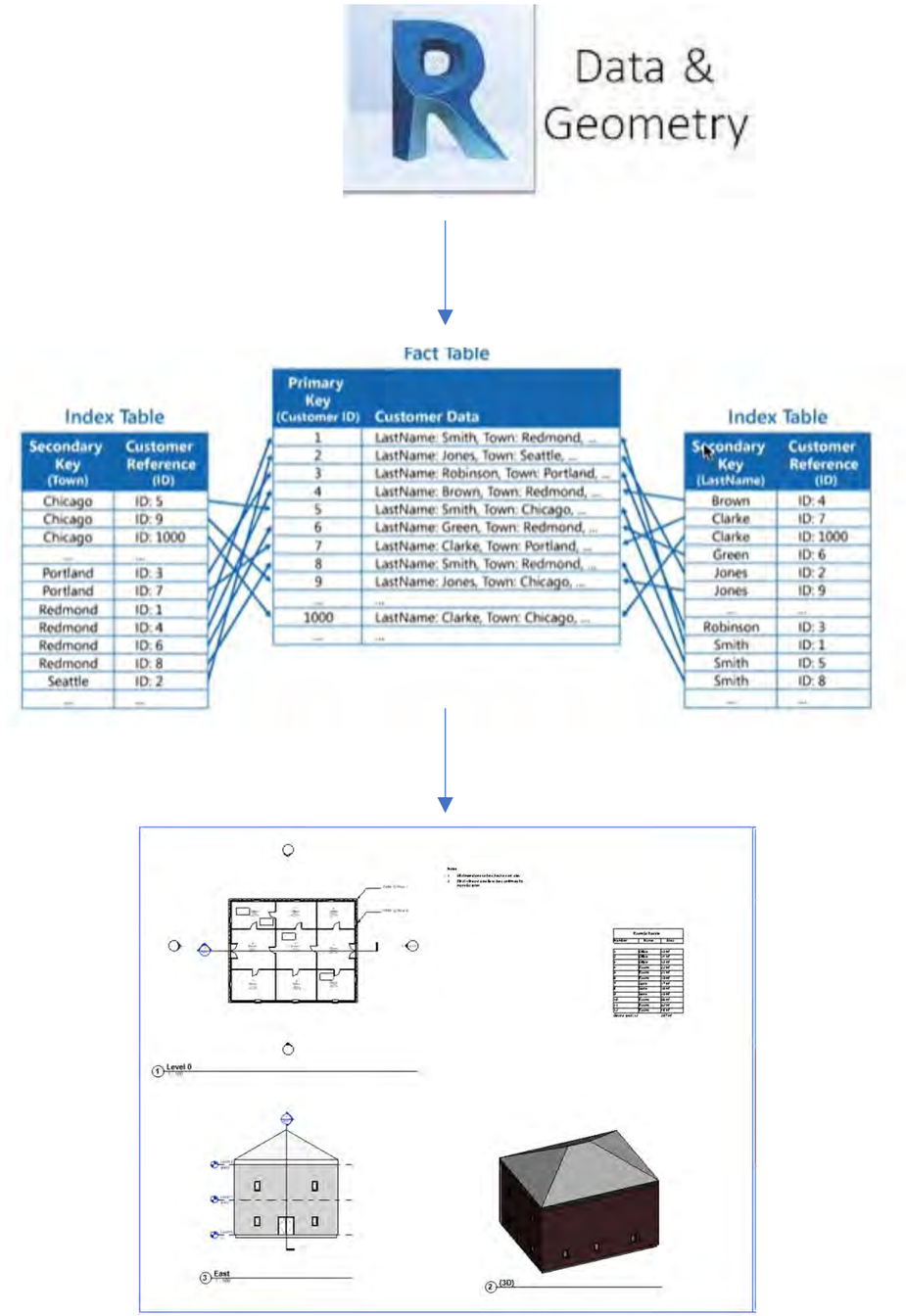


Curtain Wall geometry using Diamond shapes configured in Python script for Rhino and Grasshopper

Rhinoceros® NURBS modeling for Windows + grasshopper

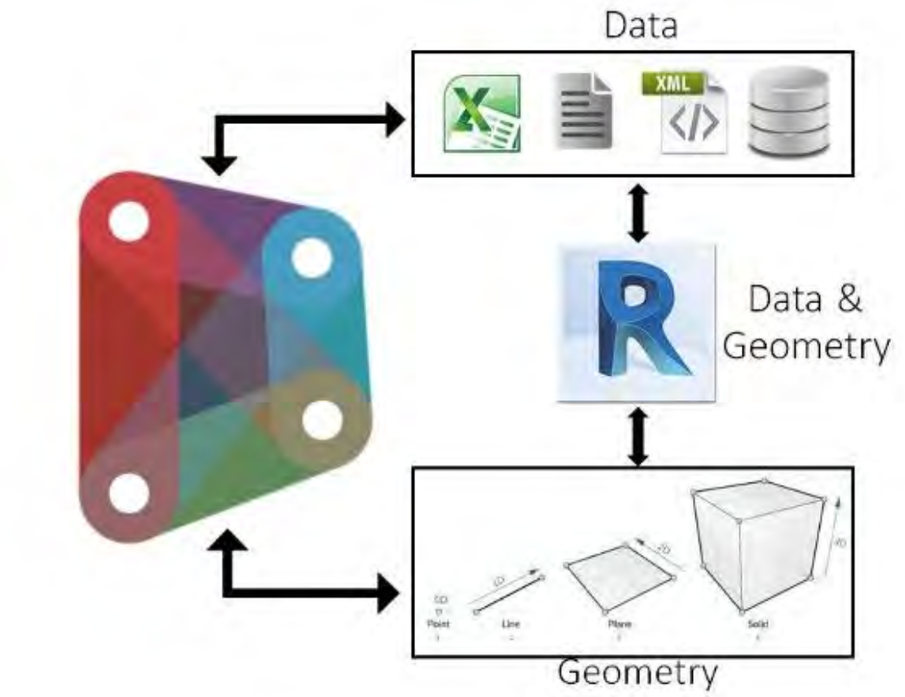
BIM

- **BIM** software is the **common data environment** for creating building components and documentation
- The model elements are embedded with parameter **data** to inform drawing information
- Revit is the largest software install base for BIM with millions of world wide users
- Can create 3D models, plans, documentations and drawings for building design and construction
- Results in a **database** you can work with



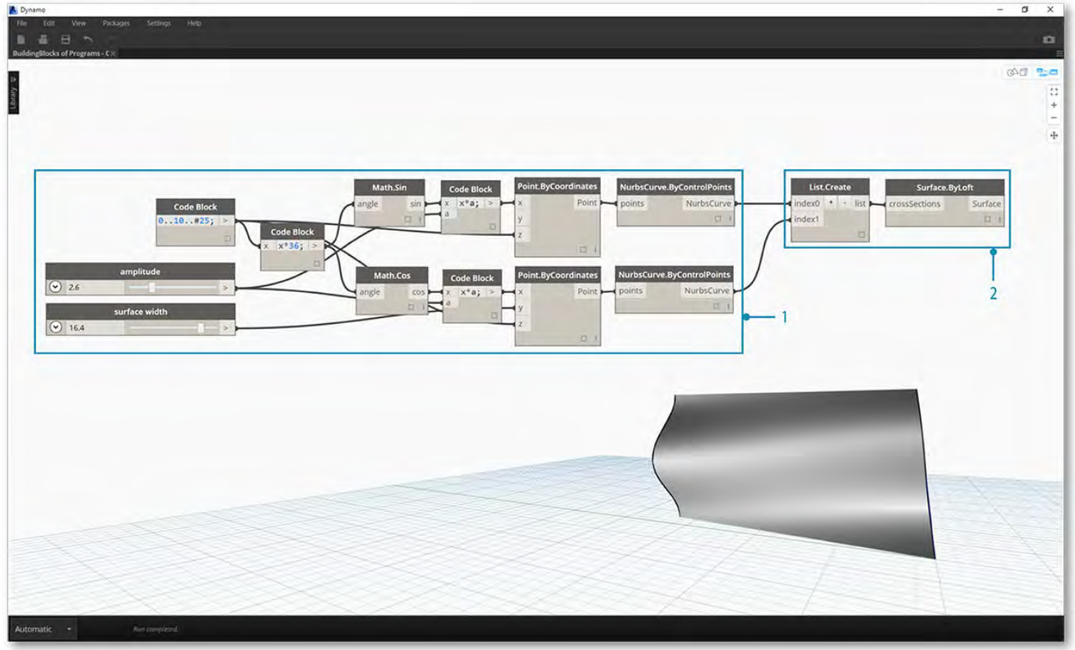
DYNAMO

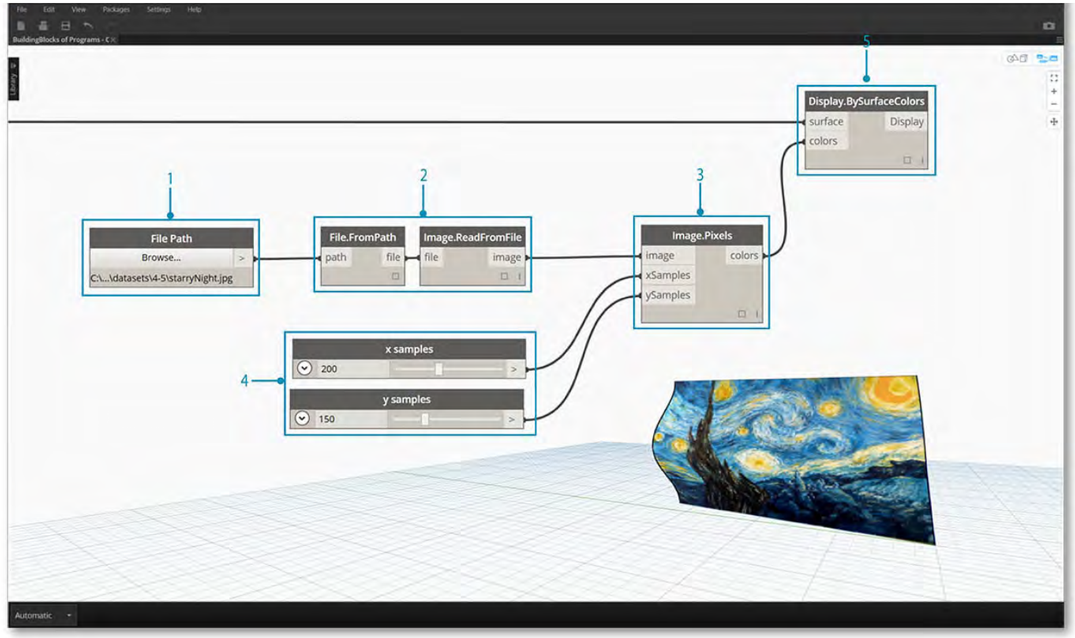
- Dynamo is from Autodesk and is a Revit add in
- Dynamo can let you **interact with Revit API** using visual scripts and Python without using a SDK
- Therefore it is a gateway for coding and opens up a lot of possibilities



DYNAMO

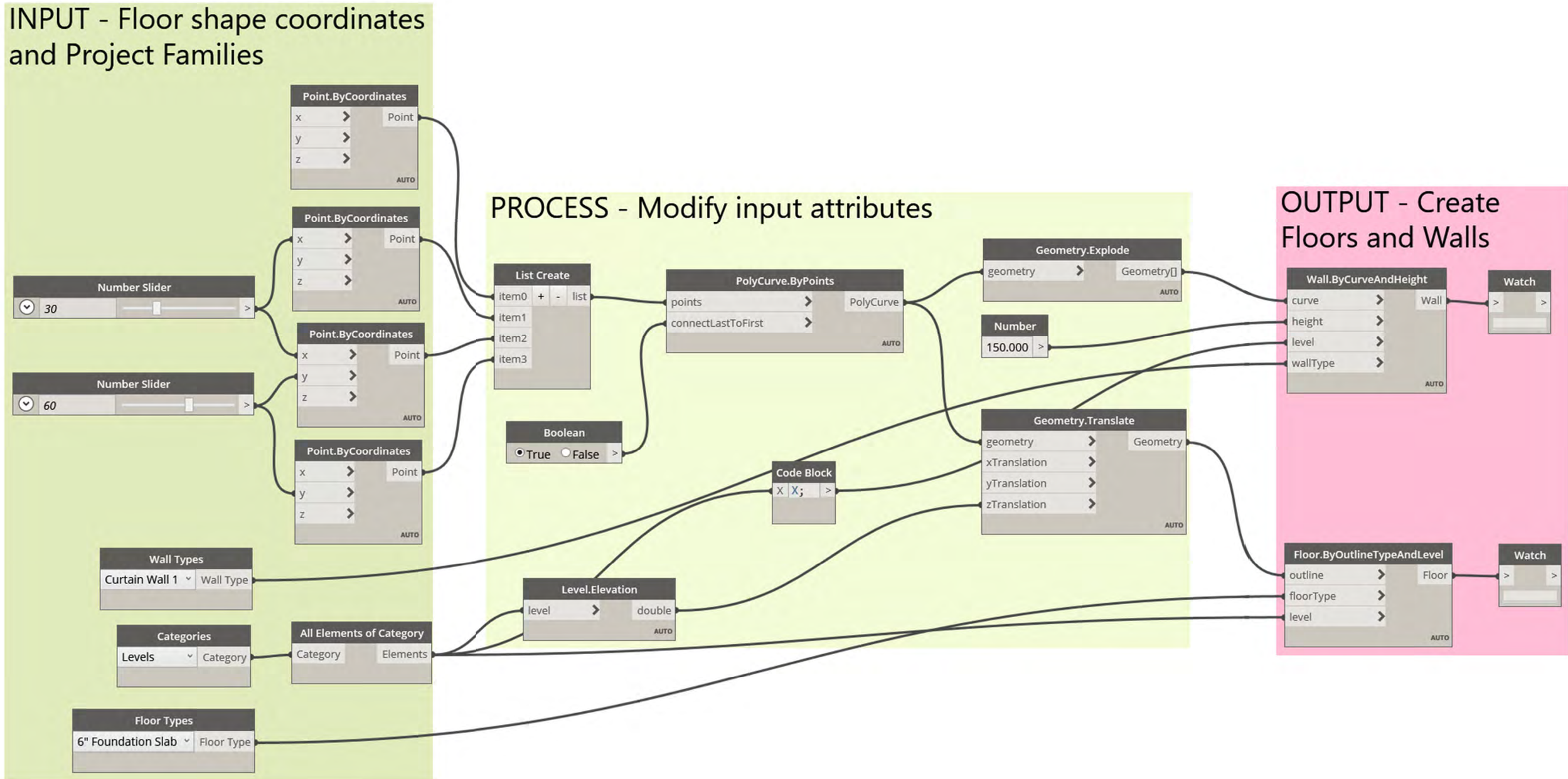
- Connect a series of nodes together into a **dataflow** to make unconventional content for Revit
- Expands the possibilities of designing in a rigid BIM format
- Can automate the creation of modeling and documentation



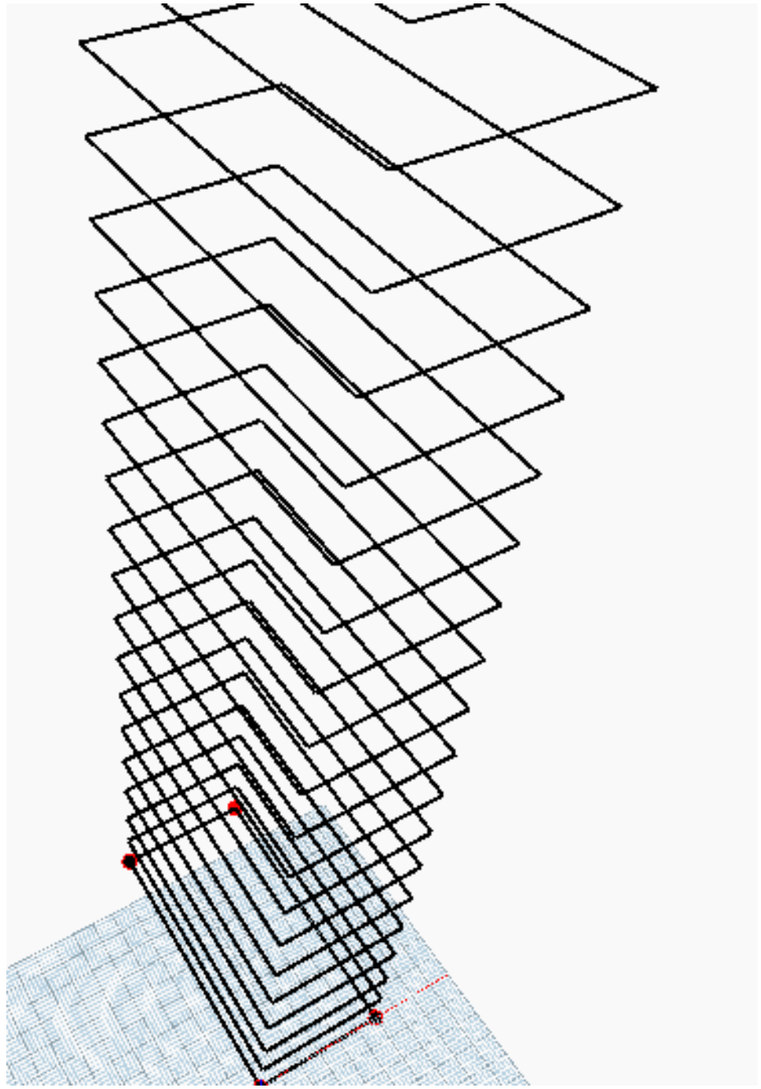


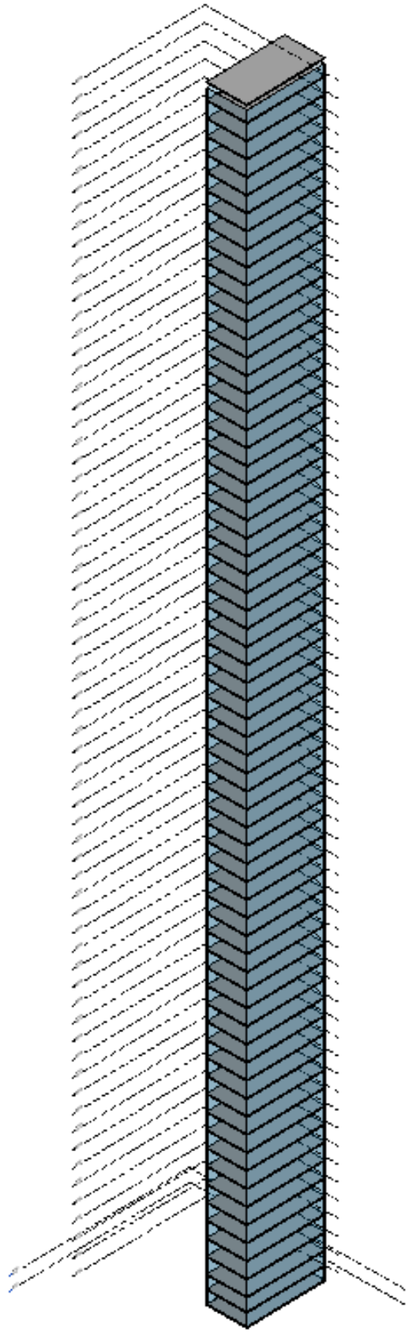
https://primer.dynamobim.org/04_The-Building-Blocks-of-Programs/4-5_color.html

DYNAMO SCRIPT AUTOMATION



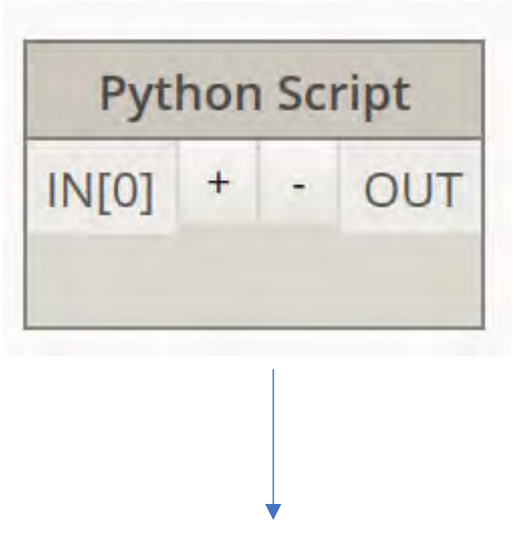
SCRIPTS





DYNAMO

- Nodes in Dynamo allow for **Python** scripting
- Uses IronPython for the compiler



Python Script

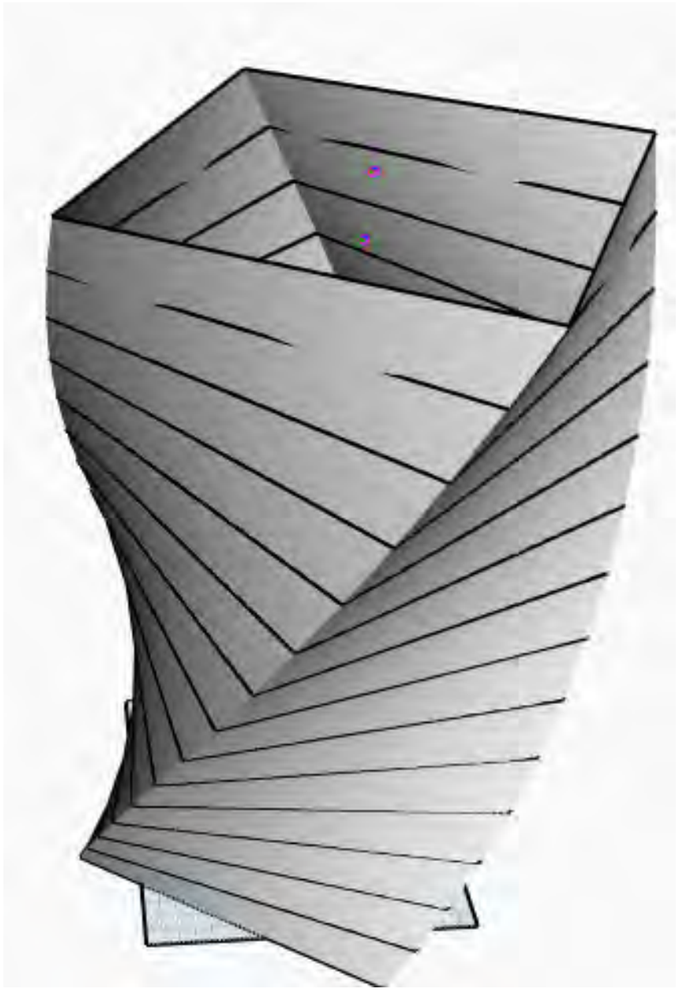
```python
# Enable Python support and load DesignScript library
import clr
clr.AddReference('ProtoGeometry')
from Autodesk.DesignScript.Geometry import *

# The inputs to this node will be stored as a list in the IN variables.
dataEnteringNode = IN

# Place your code below this line

# Assign your output to the OUT variable.
OUT = 0
```

DYNAMO

- Makes it easier to modify the **design beyond** what you can do with the default nodes

Python Script

```
# Enable Python support and load DesignScript library
import clr
clr.AddReference('ProtoGeometry')
from Autodesk.DesignScript.Geometry import *
clr.AddReference('RevitNodes')
from Revit.Elements import *

#cs = IN[0]
width = 100
length = 100
rect = Rectangle.ByWidthLength(width,length)
origin = Point.Origin()
axis = Vector.ZAxis()
degree = IN[0]

#degfloat = [float(i) for i in degree]
#degfloat = map(float, degree)
degfloat = [float(i) for i in degree]

GeoRo = Geometry.Rotate(rect,origin,axis,degfloat)

# output element based on the variable you assigned
OUT = GeoRo
```



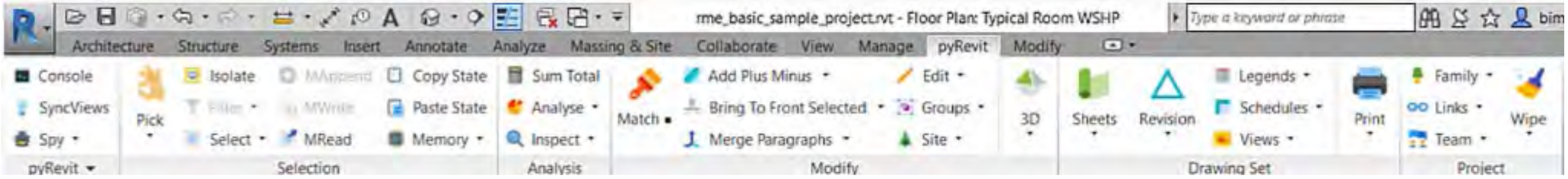
WHAT ABOUT DRAWINGS?

PYREVIT OPEN SOURCE PLUGIN DEVELOPMENT FOR REVIT

- pyRevit is a Rapid Application **Prototyping** (RAD) environment for Autodesk Revit
- Open source project for creating **custom add ins for Revit**
- Created so you **don’t need to know any C#** to create add ins to Revit
- Big un-blocker for people of non-CS backgrounds in Architecture
- Possible due to IronPython, RPS and RPW projects from other contributors

PYREVIT OPEN SOURCE PLUGIN DEVELOPMENT FOR REVIT

- Create your own tools without using the software SDK
- Uses Revit Python Wrapper (with IronPython) to use Python in a C++ runtime
- 90% of the work of Architects is **documents** so this made a big impact in productivity

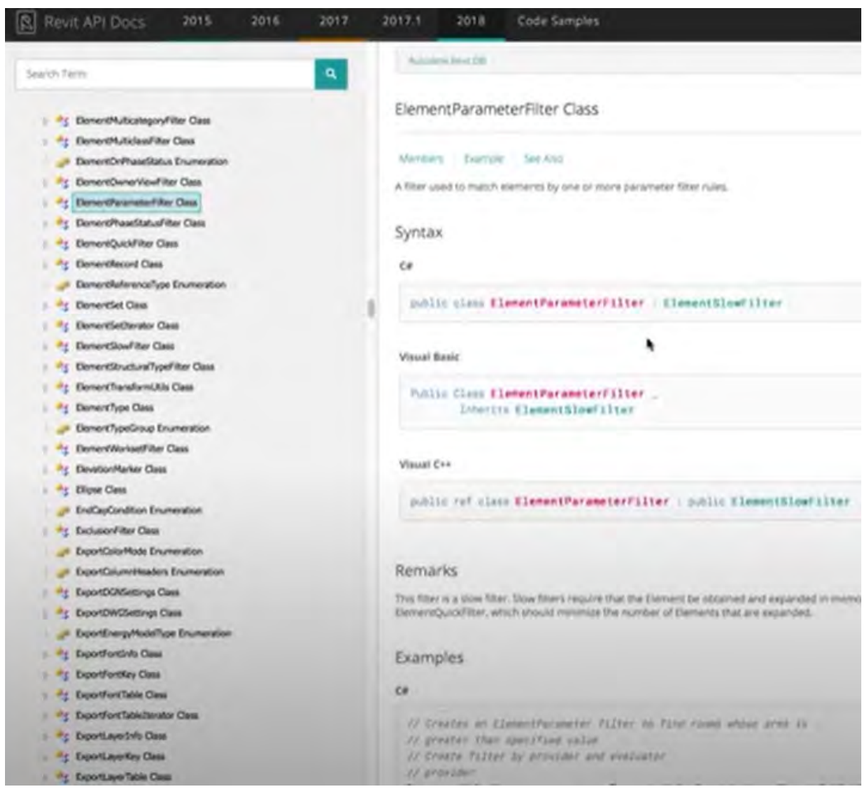


PYREVIT

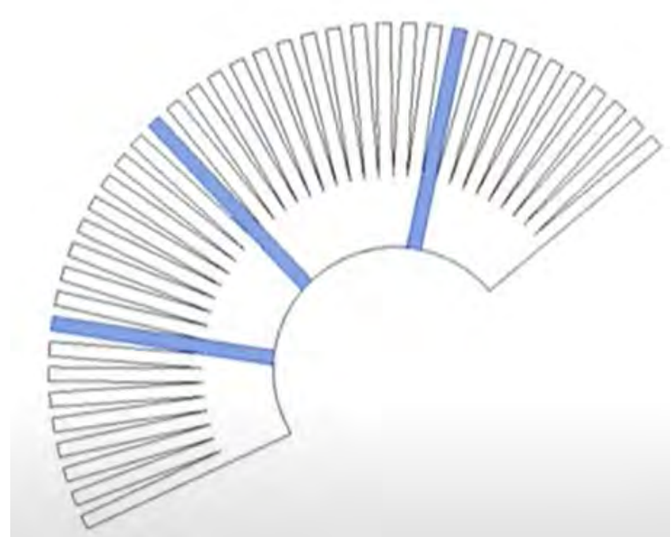
API calls in C# requires SDK kit to run in .NET

PyRevit takes the calls and can write them in Python code to create custom applications

Allows for editing in the Revit model environment based on custom tools

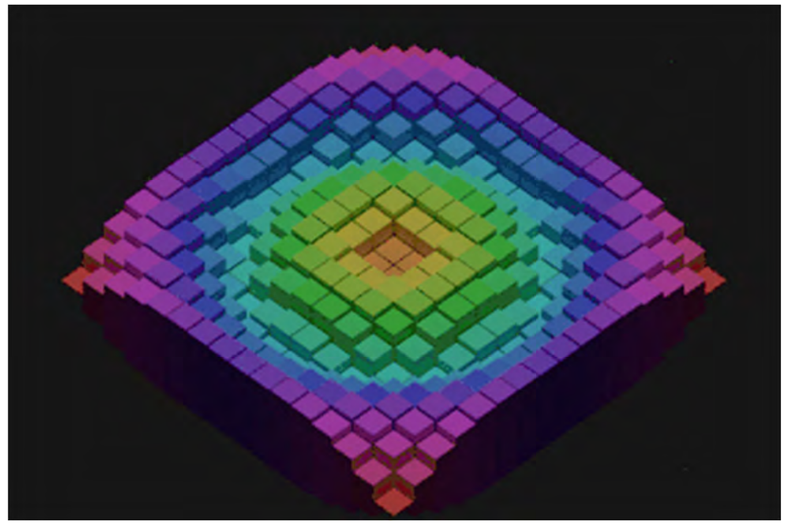


```
"""Advanced Collection of Data: Collects all the walls of height 10"""

__author__ = 'Ehsan Iran-Nejad'

# for timing ------------------------------------------------------------
from pyrevit.coreutils import Timer
timer = Timer()
# ---------------------------------------------------------------------


import Autodesk.Revit.DB as DB


doc = __revit__.ActiveUIDocument.Document
uidoc = __revit__.ActiveUIDocument

height_param_id = DB.ElementId(DB.BuiltInParameter.WALL_USER_HEIGHT_PARAM)

height_param_prov = DB.ParameterValueProvider(height_param_id)

param_equality = DB.FilterNumericEquals()

heigh_value_rule = DB.FilterDoubleRule(height_param_prov,
                                       param_equality,
                                       10.0,
                                       1E-6)

param_filter = DB.ElementParameterFilter(heigh_value_rule)

walls = DB.FilteredElementCollector(doc) \
          .WherePasses(param_filter) \
          .ToElementIds()


uidoc.Selection.SetElementIds(walls)


# for timing ------------------------------------------------------------
endtime = timer.get_time()
print(endtime)
# ---------------------------------------------------------------------
```



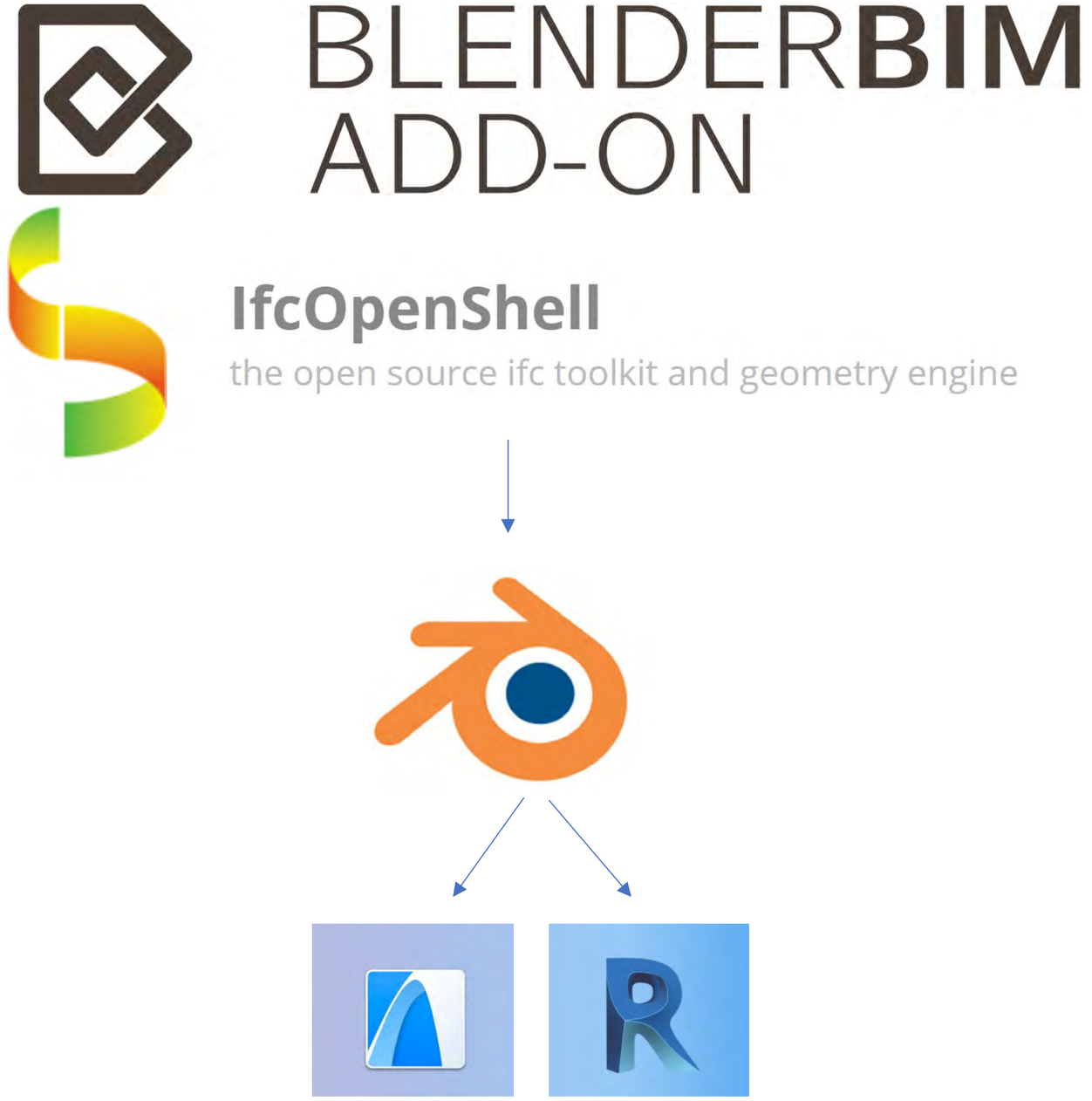
BLENDER 3D IN ARCHITECTURE DESIGN

- Blender is gaining a lot of ground with **designers**
- Blender itself is written in Python, C and C++ code with add-ons written in Python being increasingly **supported**

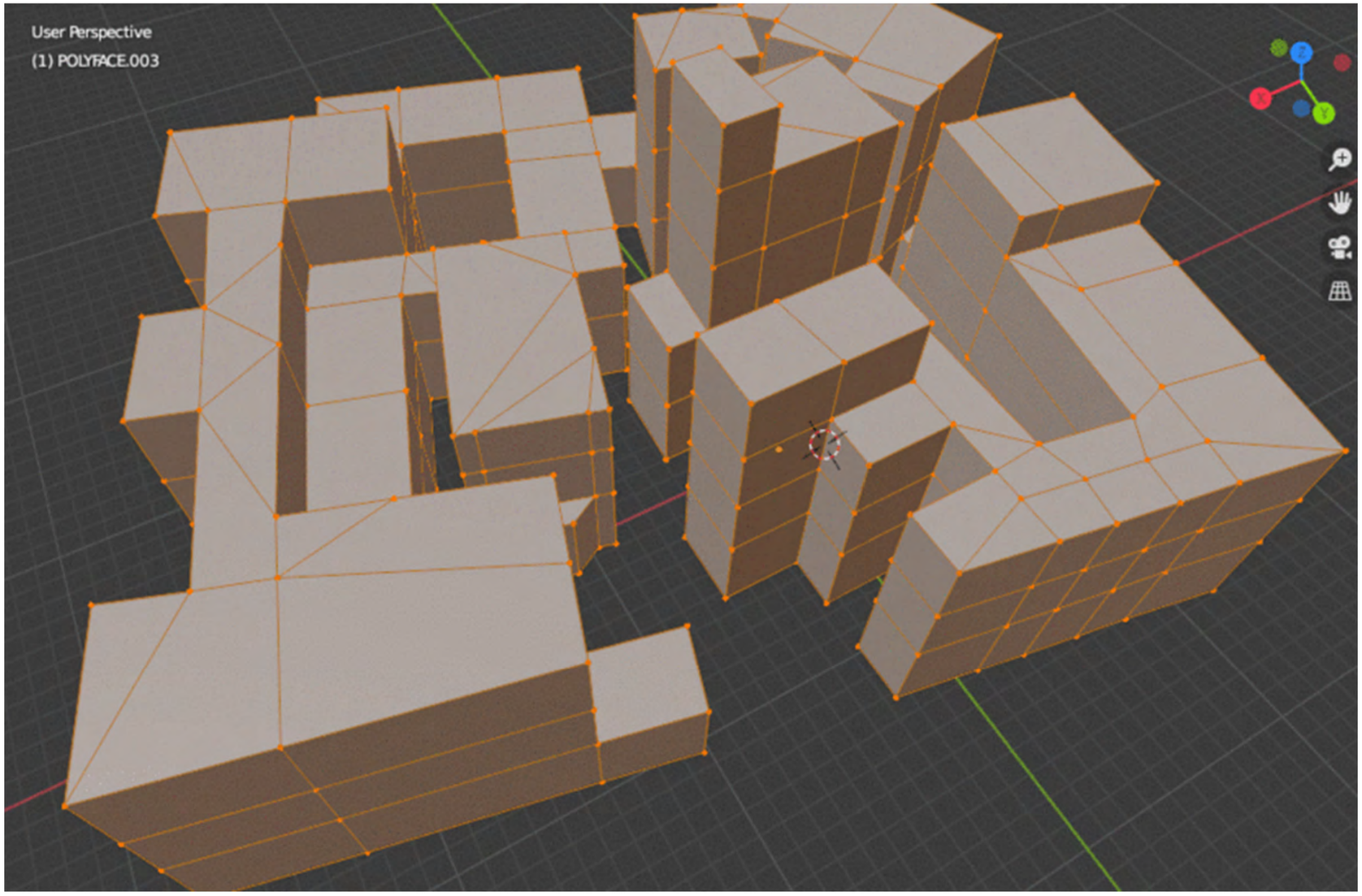


BLENDER BIM BRIDGING THE GAP

- BlenderBIM makes the Blender geometry **writeable to a BIM format** for parametric element modeling
- Add on allows data with **IFC file format** which is interchangeable with many BIM software like Revit or Archicad
- Provides a future where designers can use **open source tools** outside of enterprise suites



PROCEDURAL MODELING



Source: brunopostle

https://community.osarch.org/discussion/comment/2489/#Comment_2489

GENERATIVE TOWER DESIGN

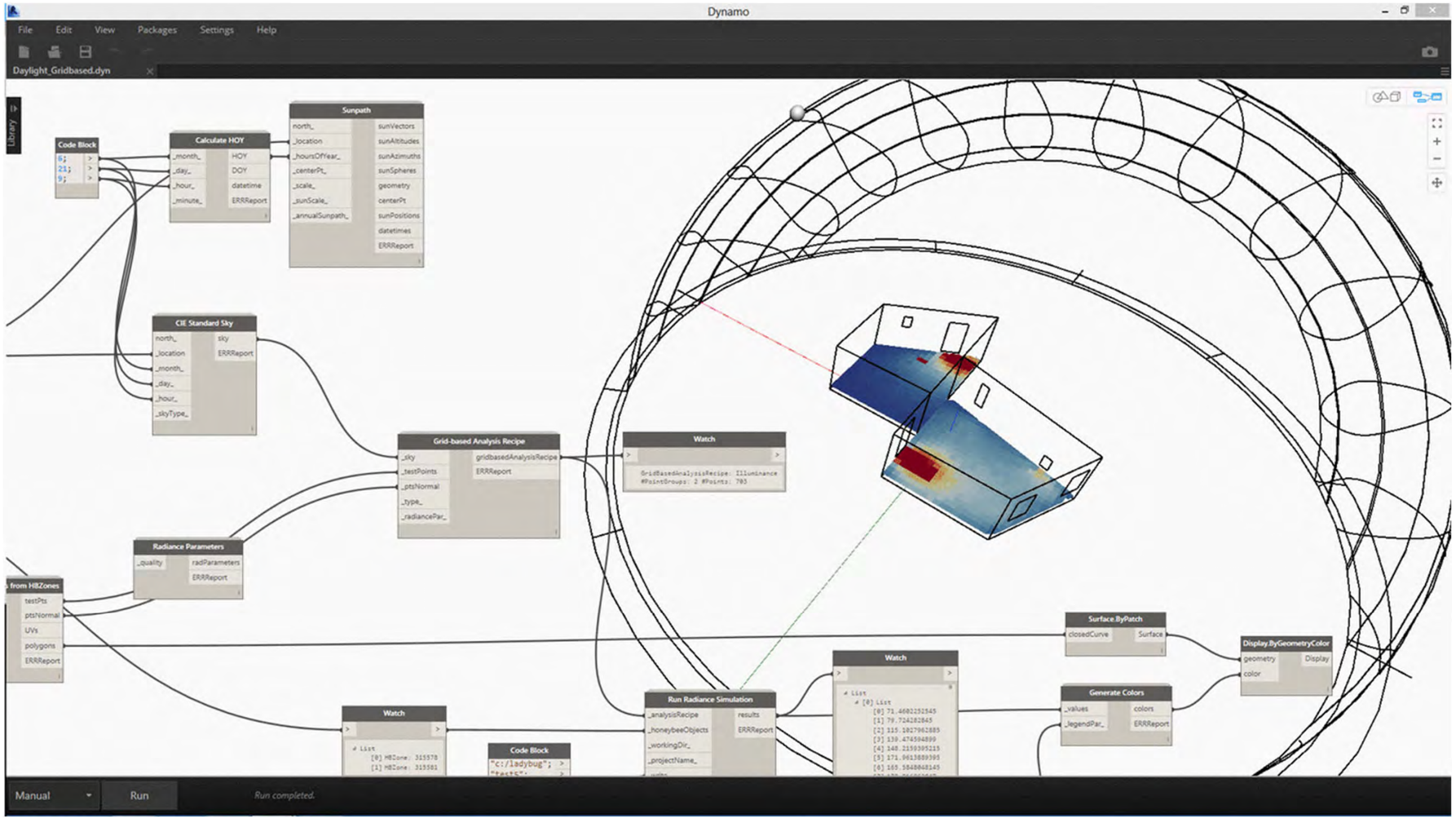
- From Blender to BIM to design

Source: UH Studio

https://www.youtube.com/watch?v=4LGw1g5sGEo&feature=emb_logo&ab_channel=UHStudio

ENERGY SIMULATION

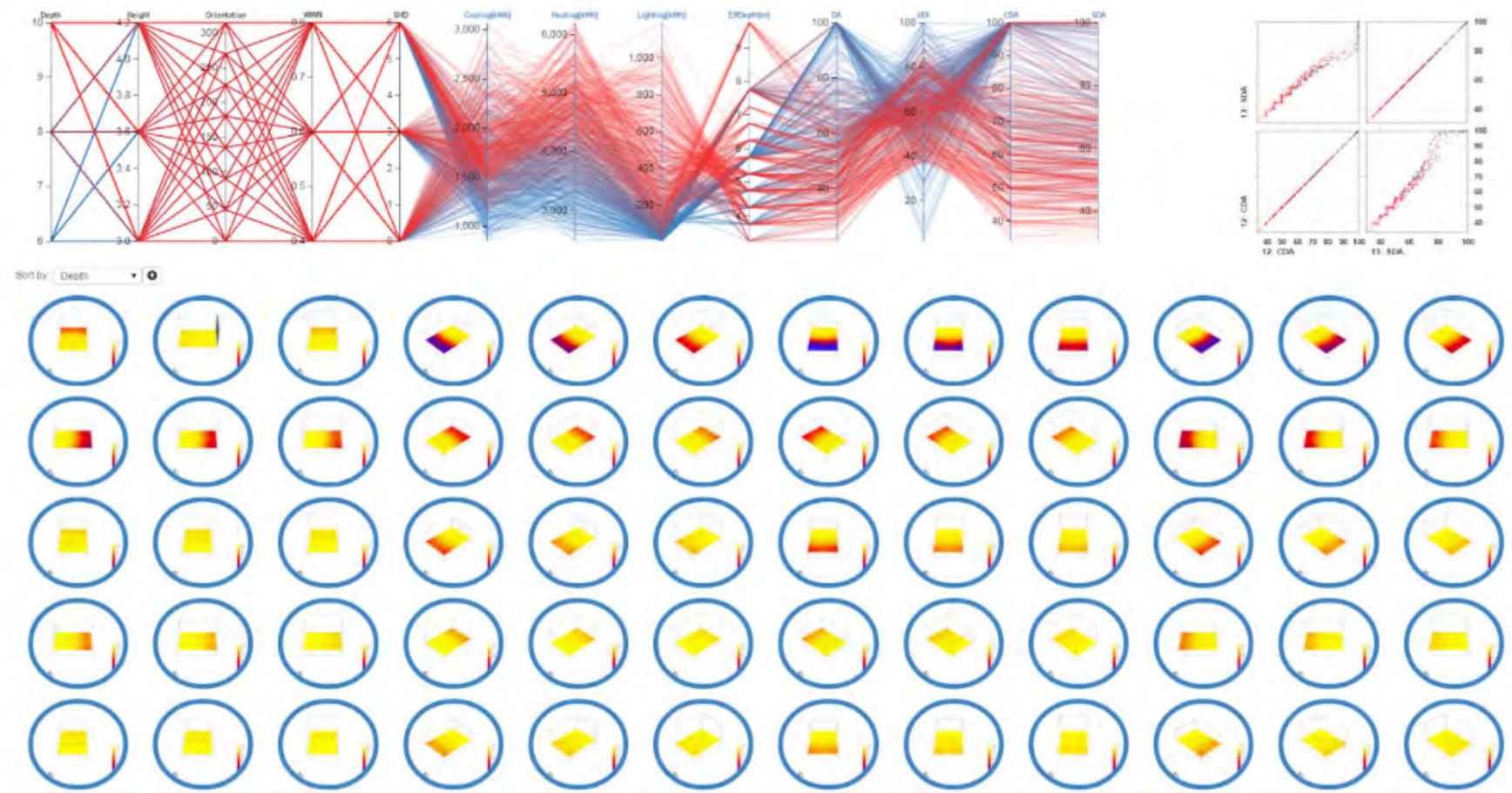
- Ladybug Tools is written in Python, which can be run on virtually any operating system and plugged into **any geometry engine**
- pip install lbt-ladybug



Mostapha Sadeghipour Roudsari

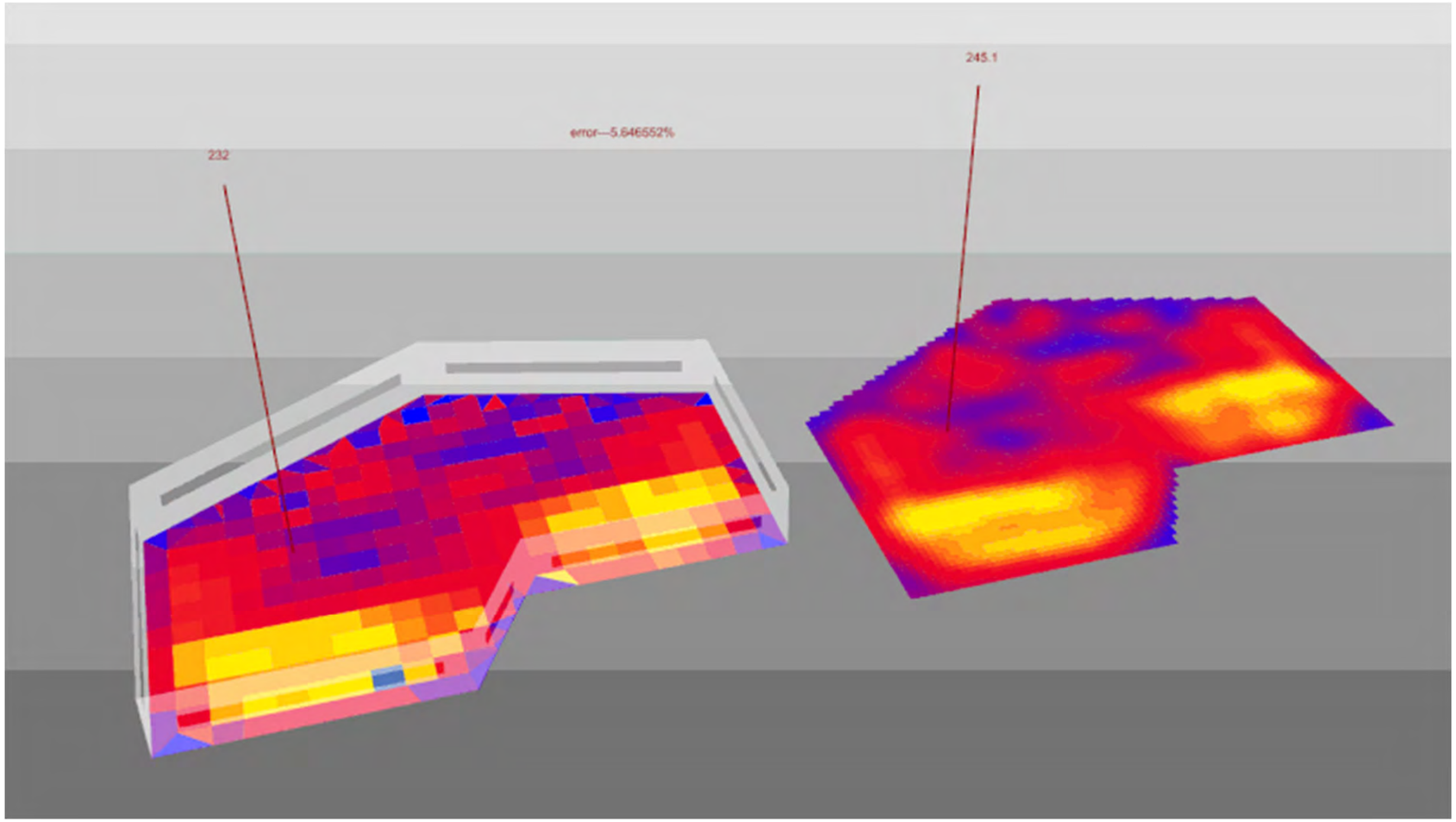
https://docs.google.com/presentation/d/1gP5rRtU_DMiP53koQZJrqHoN6q3zaKBEQDPk7CoUkEo/edit#slide=id.g4515a63eb8_0_622

ENERGY SIMULATION - OPTIONEERING



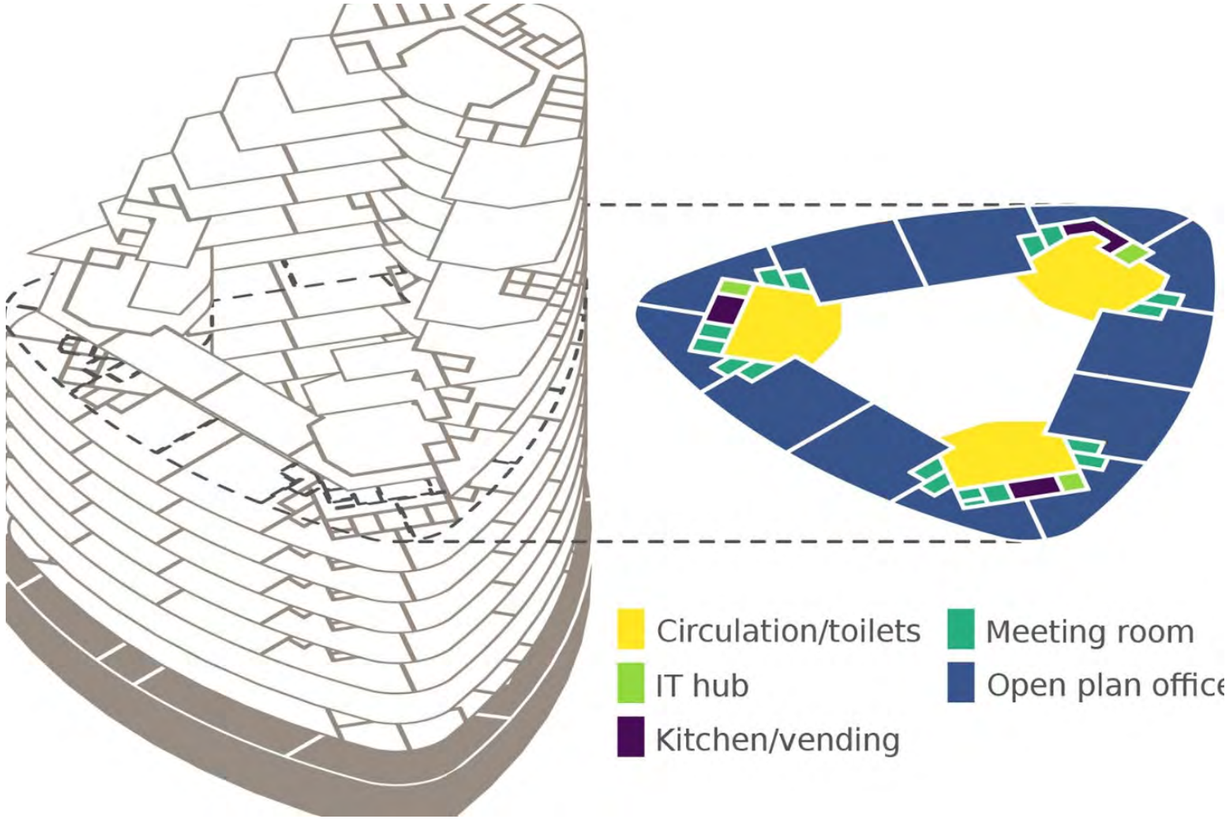
DAYLIGHT SIMULATION

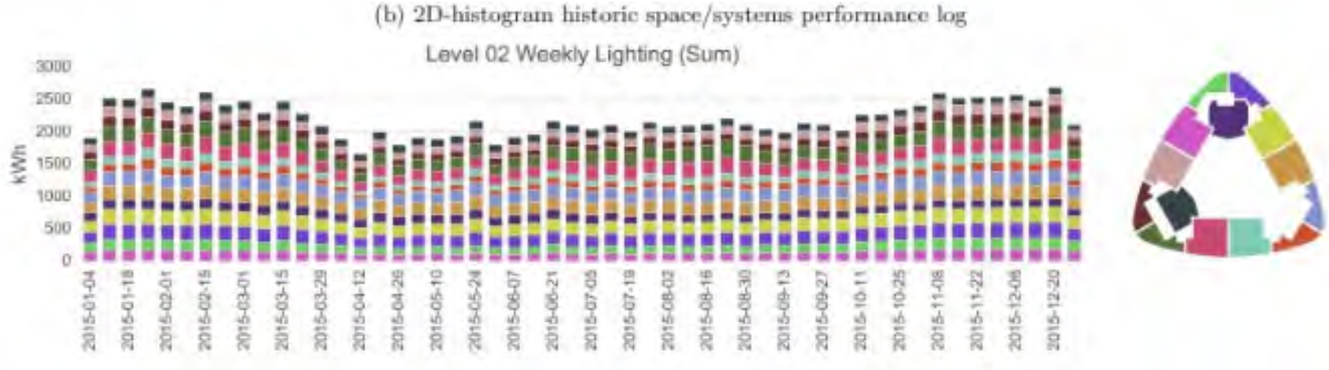
- Honey bee is a Python library to create run and visualize the results of daylight and energy analysis



FUTURE OF PYTHON IN ARCHITECTURE

- On top of all that Python is a great integrator of tools.
- All data is fluid meaning you can take the data you want out of a given software and move it into another.
- XML, CSV, HTML, JSON and other document formats can easily let you transfer data around.





FUTURE OF PYTHON IN ARCHITECTURE

- On top of all that Python is a great **integrator** of tools.
- XML, CSV, HTML, JSON and other document formats can easily let you transfer data around.
- Helps designers escape from enterprise software into **open source ecosystem**

TAKEAWAY

- Python Makes it possible!

CONTRIBUTORS AND OPEN SOURCE PROJECTS

- **Special Thanks to:**
- Gui Talarico (Revit API docs, Revit Python Wrapper)
- Ehsan Iran-Nejad (pyRevit)
- Dimon Moult (Blender BIM)
- Mostapha Sadeghipour Roudsari (Ladybug Tools)
- Dynamo Team
- Grasshopper Team
- Python contributors everywhere

- Pattern Language - https://www.patternlanguage.com/
- Dynamo - https://primer.dynamobim.org/10_Custom-Nodes/10-4_Python.html
- Grasshopper - https://developer.rhino3d.com/guides/rhinopython/your-first-python-script-in-grasshopper/
- Revit Python Shell - https://github.com/architecture-building-systems/revitpythonshell
- Revit Python Wrapper - https://revitpythonwrapper.readthedocs.io/en/latest/
- Revit API docs - https://www.revitapidocs.com/
- PyRevit - https://www.notion.so/pyRevit-bd907d6292ed4ce997c46e84b6ef67a0
- BlenderBIM - https://blenderbim.org/
- ifcOpenShell - http://ifcopenshell.org/
- Ladybug tools - https://www.ladybug.tools/

Thanks everyone!

Tadeh Hakopian Contact Information:

Twitter: https://twitter.com/tadeh_hakopian

Linkedin: https://www.linkedin.com/in/thakopian/

Github: https://github.com/thakopian

Email: thakopian@gmail.com