

Creating Automated Documentation with Dynamo

Drew Jarvis

Applications Specialist





About the speaker

Drew Jarvis

An innovative and motivated professional with almost 20 years' experience in CAD/BIM Management, design, and training. Extensive experience in the theoretical and practical implementation of BIM both from a strategic and tactical level. With a strong passion for the AEC industry and a solid business sense and entrepreneurial background I am always looking to see how emerging technologies can help companies remain competitive while advancing their practices.

I live in Port Moody BC with my wife, daughter and dog and enjoy outdoor pursuits and following whatever local sports team is playing.

Learning Objectives – lots of text...

LEARN HOW TO AUTOMATE THE CREATION OF VIEW AND SHEETS

A big time saver for any project setup

LEARN HOW TO CREATE 2D SCHEMATICS FROM 3D MODEL GEOMETRY AND ASSOCIATED DATA

Revit doesn't seem to want to do this out of the box, so lets see if we can automate some creation

LEARN HOW TO TAKE YOUR IDEAS FOR AUTOMATION AND PUT THEM INTO EFFECT USING DYNAMO

Stay awake and you should pick up something from this class, download the graphs and example videos too

GET MORE COMFORTABLE USING PYTHON IN DYNAMO

You DON'T need python to use Dyanmo, but it helps and someone else out there has already written someone cool you can copy and paste

Method

I am going to show you some graphs/scripts/dynamo things

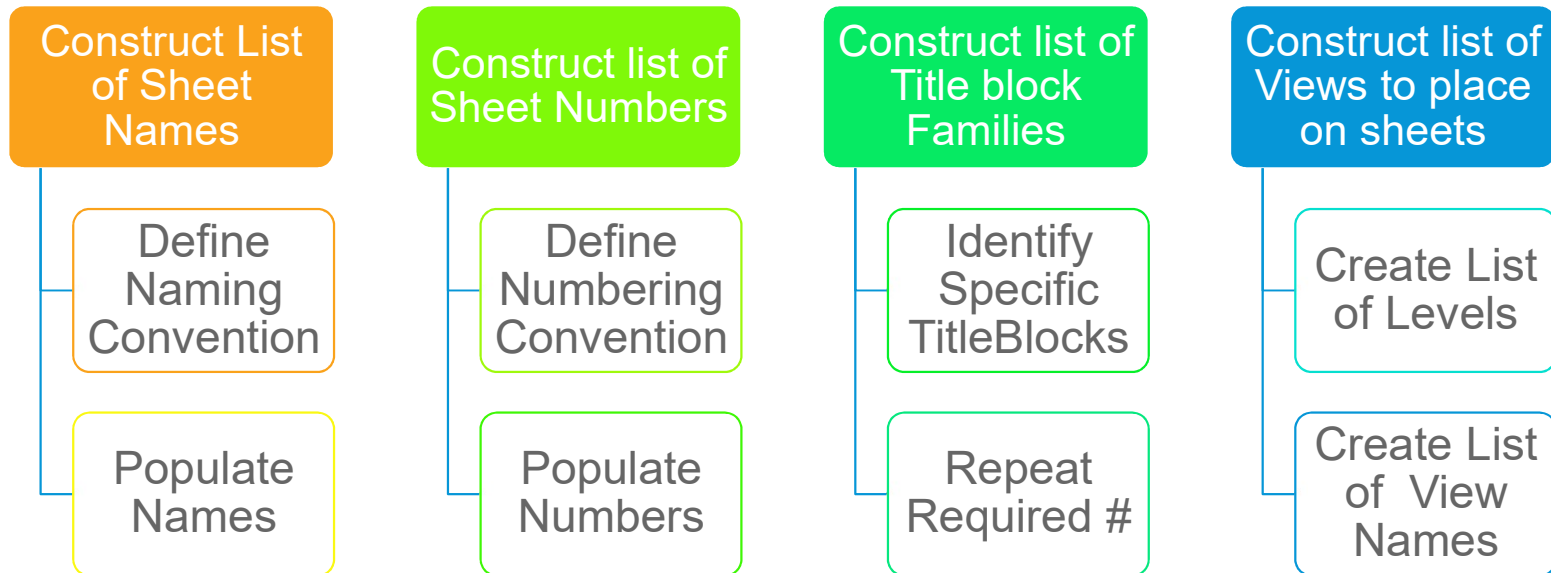
I have 4 examples for you, you can download them after the class

Each example will introduce some main workflows that I will delve into

We will take questions at the end

Lets get started with Views and Sheets Automation

View and Sheets

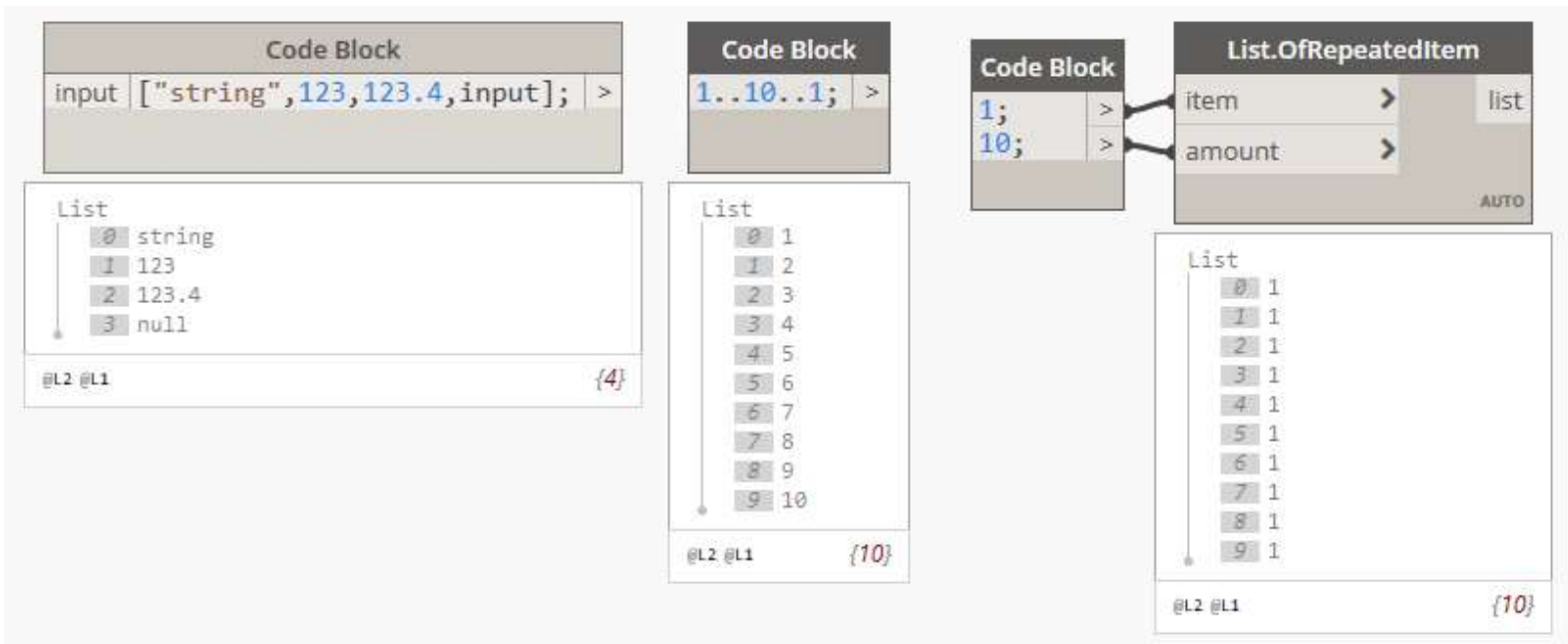


FloorPlanView.ByLevel	
level	FloorPlanView
AUTO	

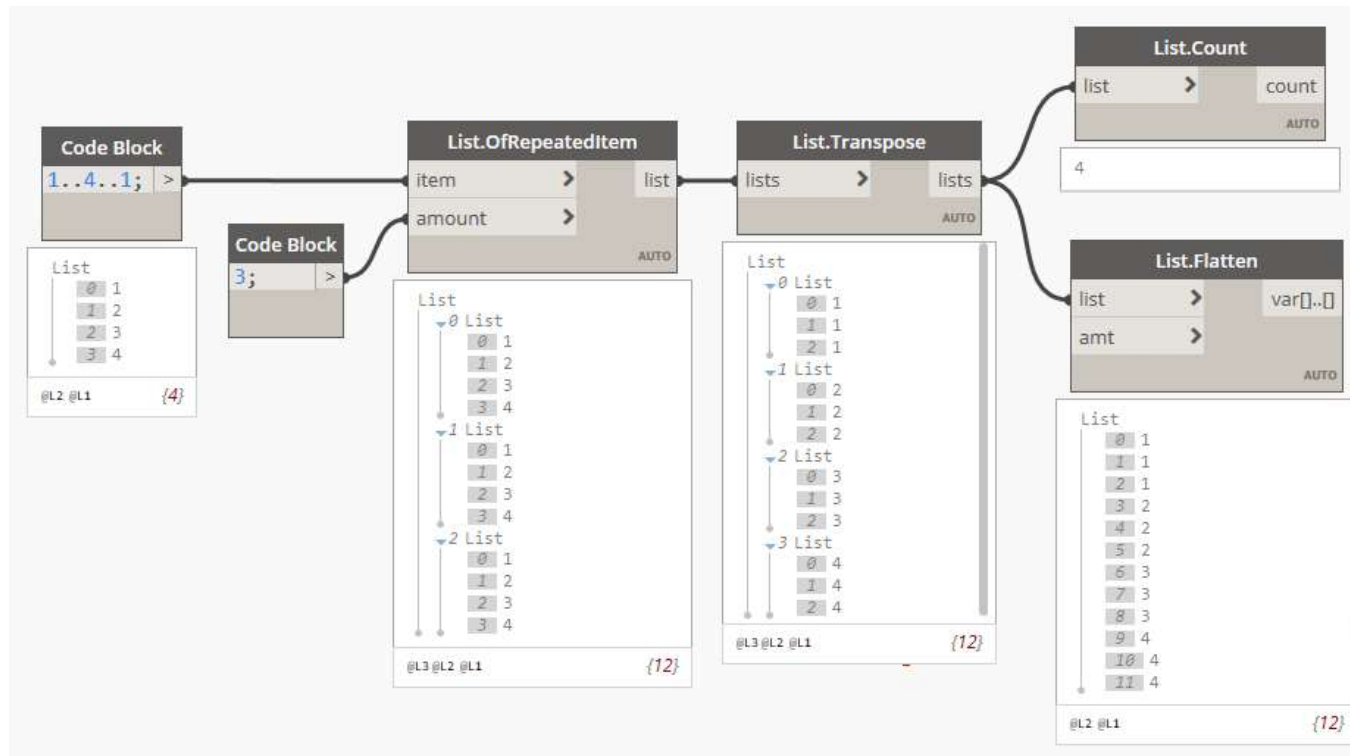
Family Types	
A1 metric:A1 metric	Family Type

Sheet.ByNameNumberTitleBlockAndView	
sheetName	Sheet
sheetNumber	
titleBlockFamilyType	
view	
AUTO	

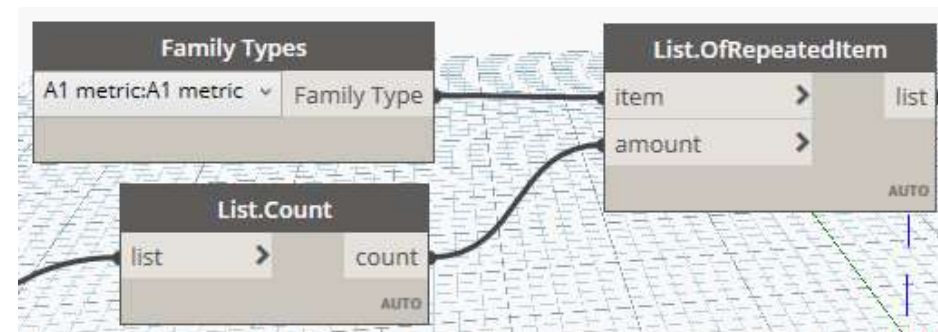
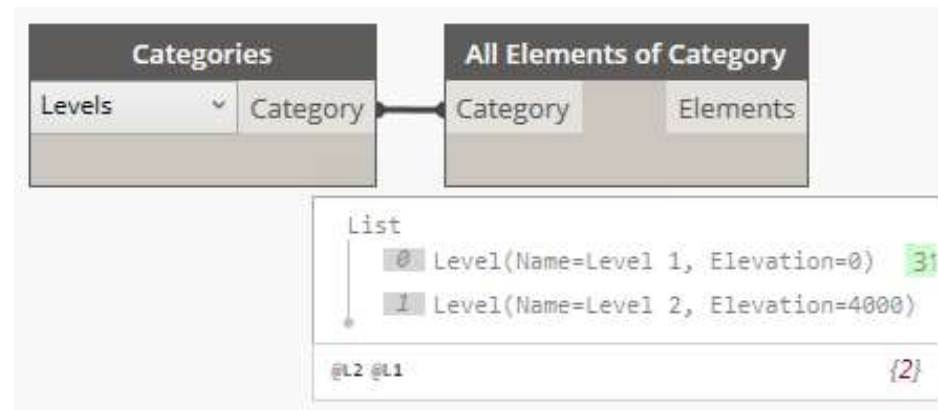
Creating Lists



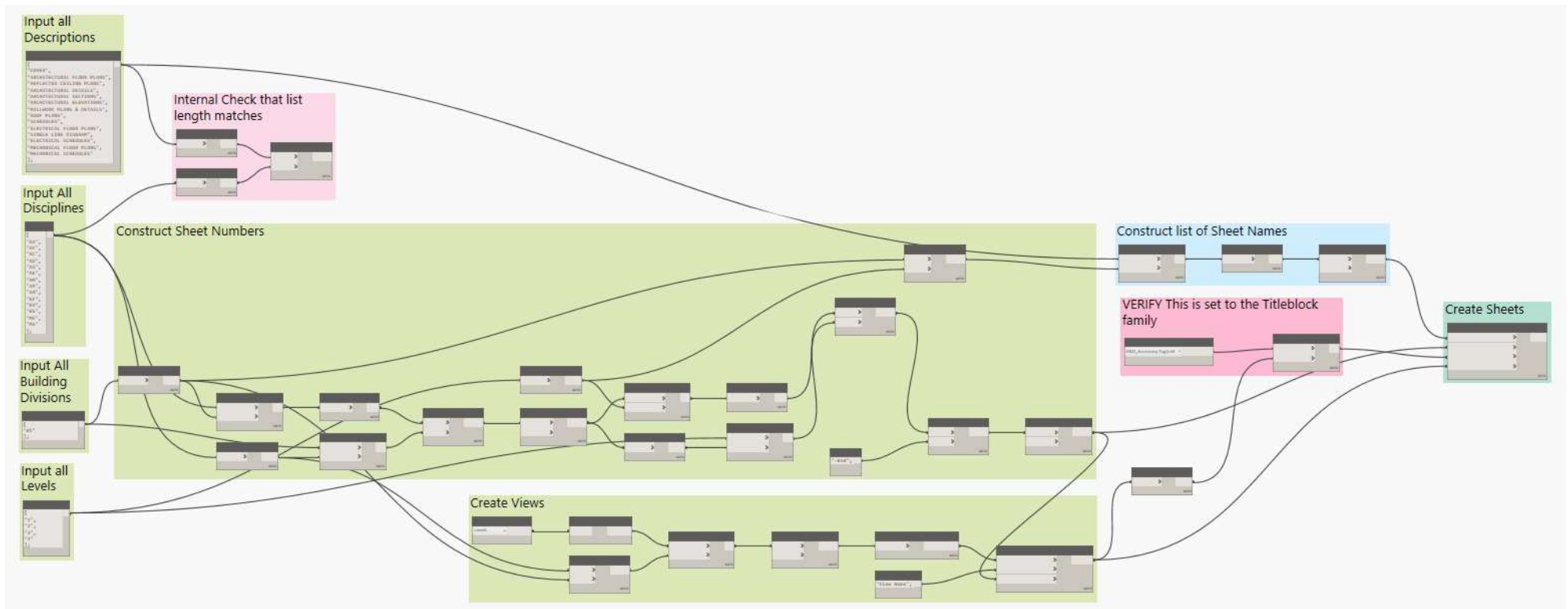
Manipulating Lists



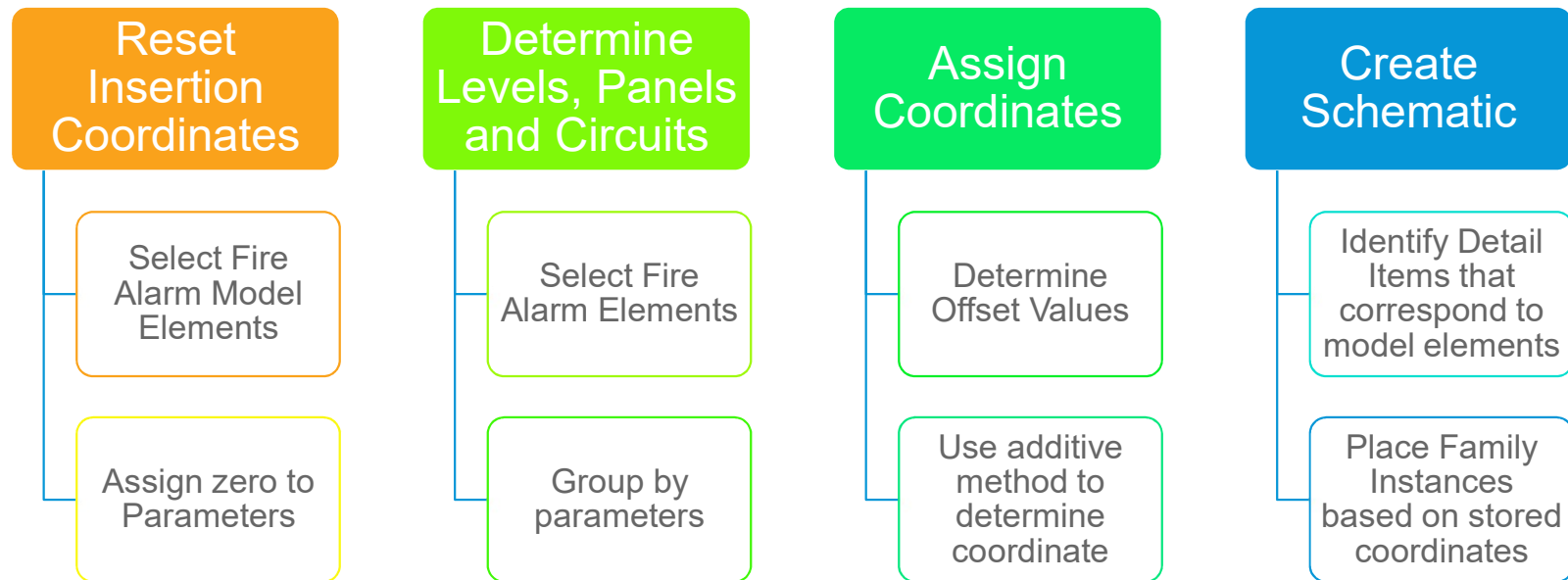
Collecting Elements



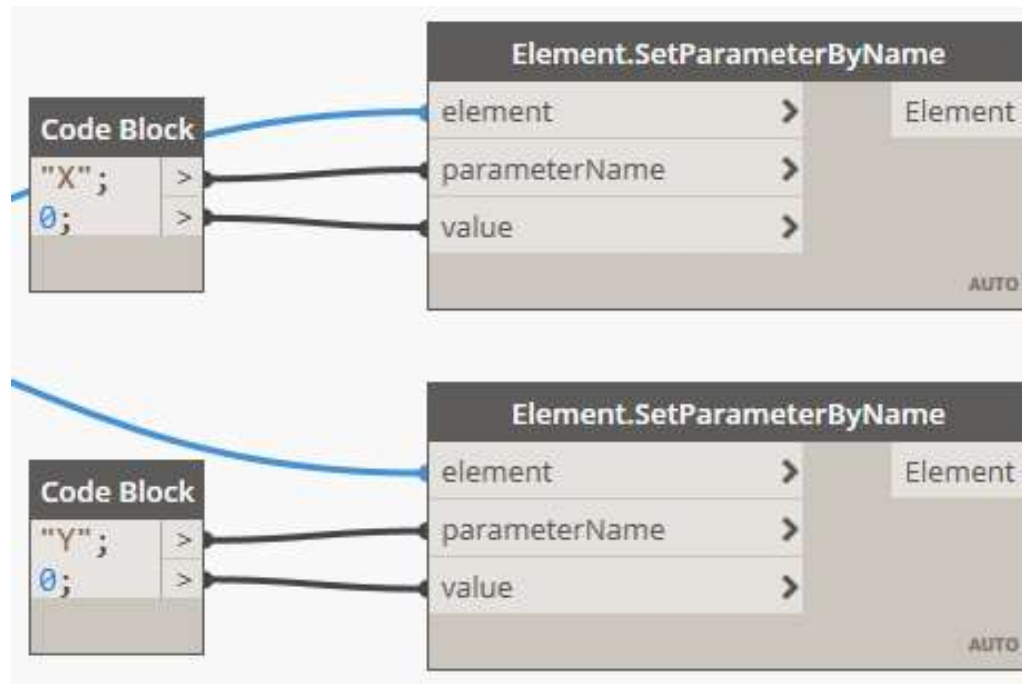
Lets Take a Look



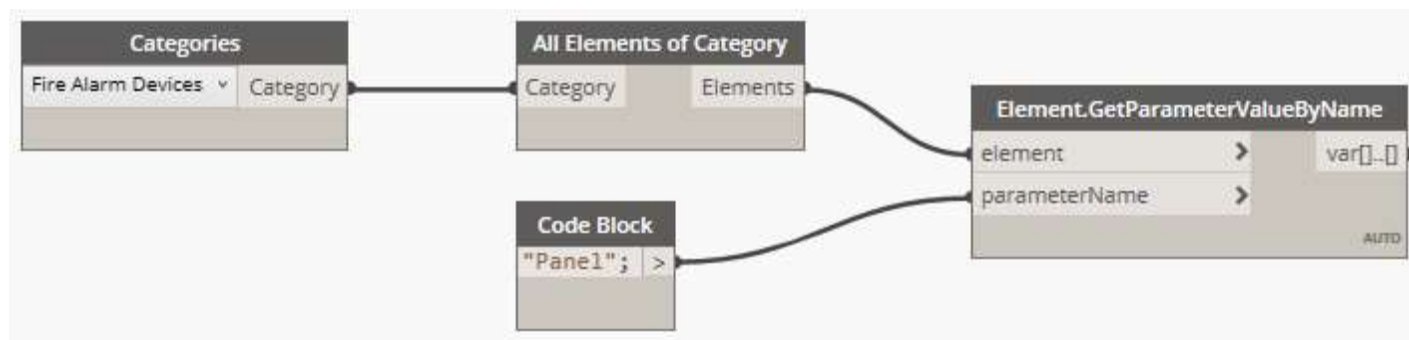
Fire Alarm Schematic



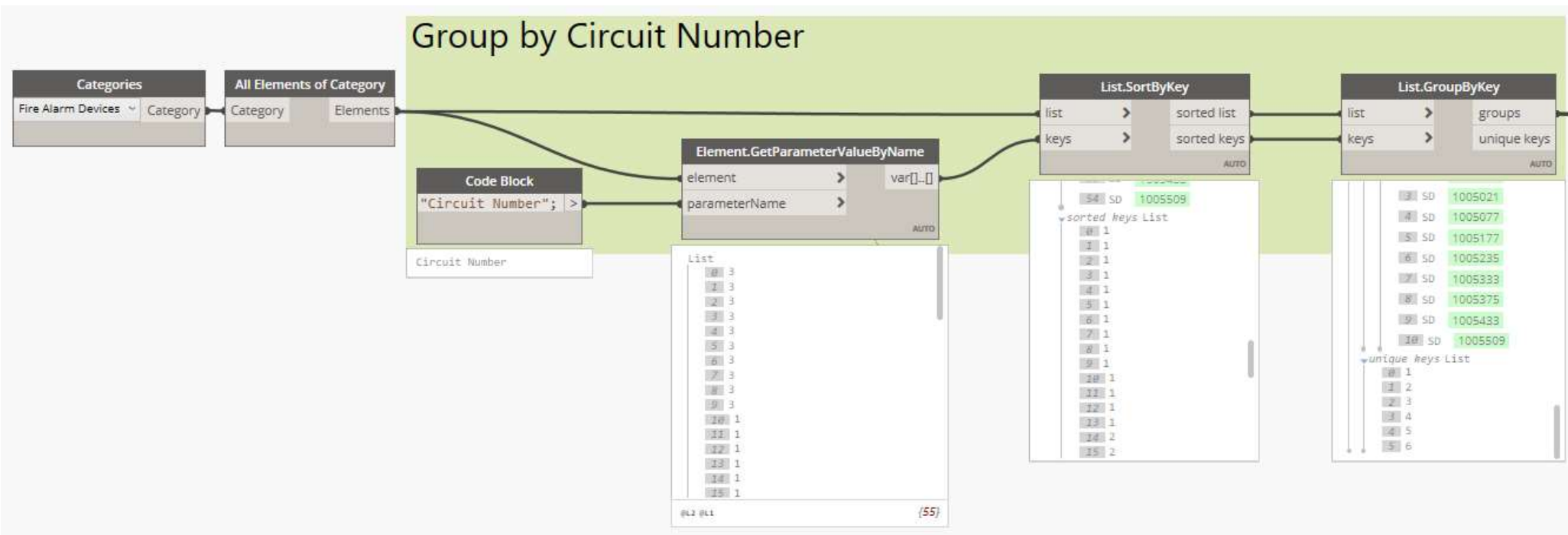
Setting Data on Elements



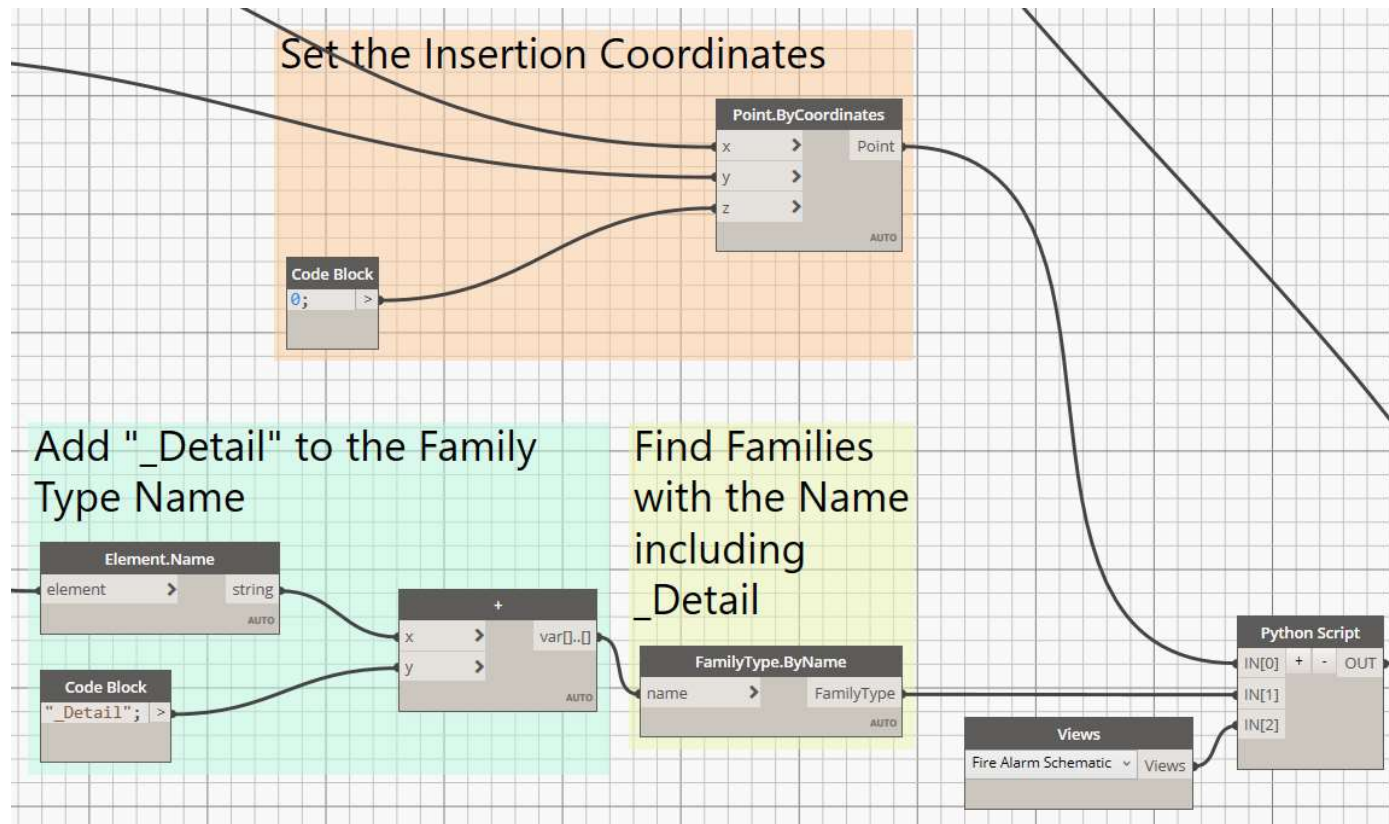
Getting Data from Elements



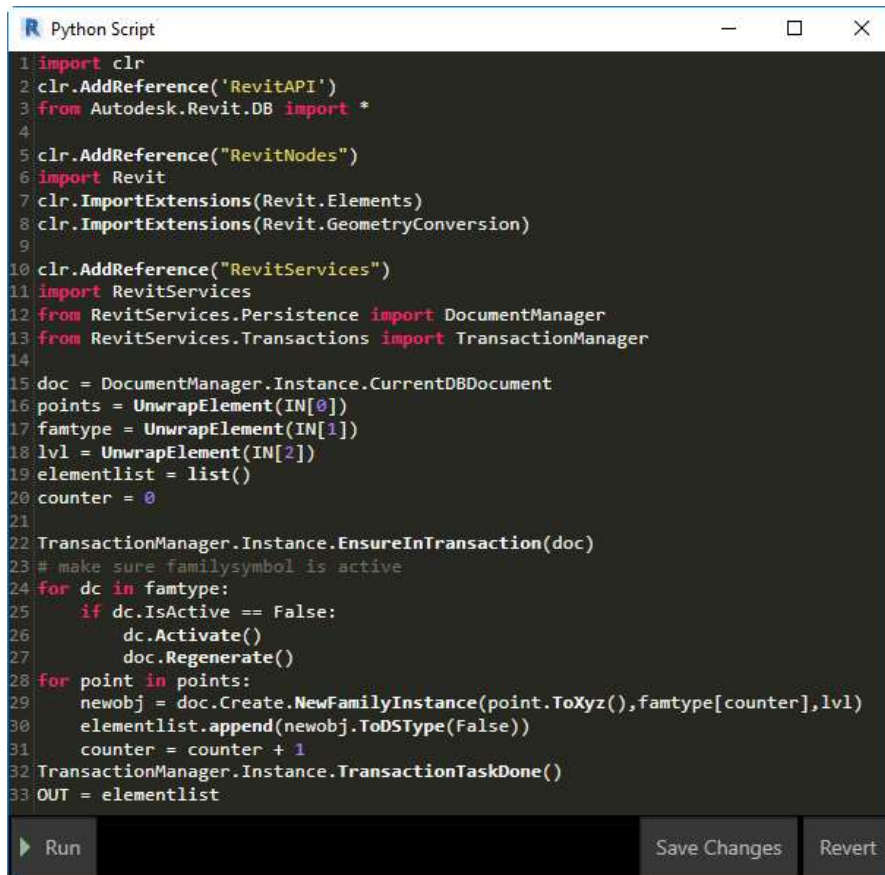
Sorting / Grouping Elements



String Manipulation and Select Specific Family Instances



Python Script



```
1 import clr
2 clr.AddReference('RevitAPI')
3 from Autodesk.Revit.DB import *
4
5 clr.AddReference("RevitNodes")
6 import Revit
7 clr.ImportExtensions(Revit.Elements)
8 clr.ImportExtensions(Revit.GeometryConversion)
9
10 clr.AddReference("RevitServices")
11 import RevitServices
12 from RevitServices.Persistence import DocumentManager
13 from RevitServices.Transactions import TransactionManager
14
15 doc = DocumentManager.Instance.CurrentDBDocument
16 points = UnwrapElement(IN[0])
17 famtype = UnwrapElement(IN[1])
18 lvl = UnwrapElement(IN[2])
19 elementlist = list()
20 counter = 0
21
22 TransactionManager.Instance.EnsureInTransaction(doc)
23 # make sure familysymbol is active
24 for dc in famtype:
25     if dc.IsActive == False:
26         dc.Activate()
27         doc.Regenerate()
28 for point in points:
29     newobj = doc.Create.NewFamilyInstance(point.ToXYZ(), famtype[counter], lvl)
30     elementlist.append(newobj.ToDSType(False))
31     counter = counter + 1
32 TransactionManager.Instance.TransactionTaskDone()
33 OUT = elementlist
```

Run Save Changes Revert

Nodes in Dynamo are written in c# and compiled into reference dll files

You can use Specific Nodes “Python Script” or “Python Script from String” to run Python code that references the Revit API (or just Python).

Most of the power of the Revit API in Dynamo, however, it requires knowledge of a programming language

Python Boilerplate

CLR

Common Language Runtime – basically manages the execution of .Net and gives access to the references to Revit.

REVITAPI

A comprehensive library for interacting with Revit

```
import clr
clr.AddReference('RevitAPI')
from Autodesk.Revit.DB import *
```

REVITSERVICES

Access to the DocumentManager & TransactionManager

```
clr.AddReference("RevitServices")
import RevitServices
from RevitServices.Persistence import DocumentManager
from RevitServices.Transactions import TransactionManager
```

REVITNODES

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Cras lacinia interdum odio, at cursus elit sagittis lobortis. Proin eu nisl molestie, dignissim ante ut, dictum ex.

```
clr.AddReference("RevitNodes")
import Revit
clr.ImportExtensions(Revit.Elements)
clr.ImportExtensions(Revit.GeometryConversion)
```


Python Boilerplate

DOCUMENTMANAGER

Gives access to document items like the active view

```
doc = DocumentManager.Instance.CurrentDBDocument
```

INPUTS

Values passed into the Python Node can be access and assigned to variables

```
variable = IN[0]
```

TRANSACTIONMANAGER

Enables you to modify the document, transactions are built into the Nodes in Dynamo, however when using Python you need to enclose your document modifications in the TransactionManager

```
TransactionManager.Instance.EnsureInTransaction(doc)  
# Do something in here that modifies the document  
TransactionManager.Instance.TransactionTaskDone()
```

OUTPUTS

You can output one item from the Python Node, however this item can contain multiple variables, these are then accessible as a zero index list

```
OUT = elementlist
```

Python Simple For Loop and List Assignment

The code to the right:

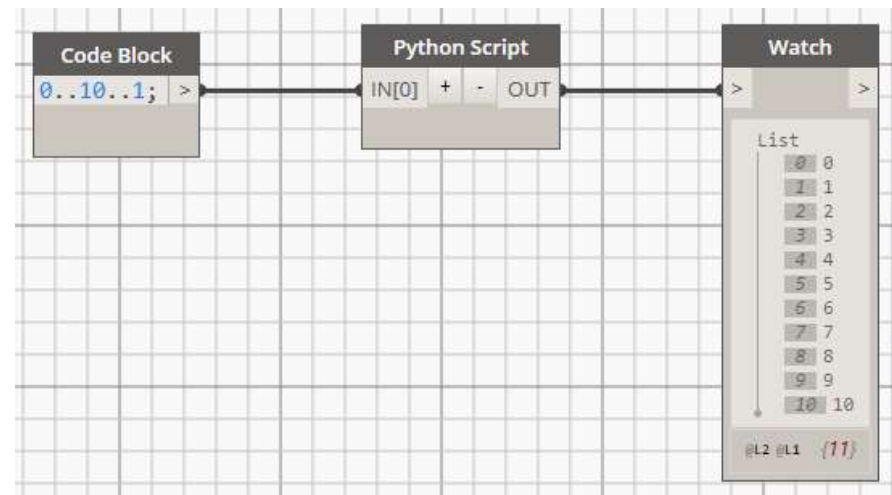
- Assigns the input IN[0] to the variable input
- Declares a variable listABC as a new list with no content
- Runs through each element in input and assigns the value to the listABC
- Outputs the new variable listABC

```
input = IN[0]
listABC = []

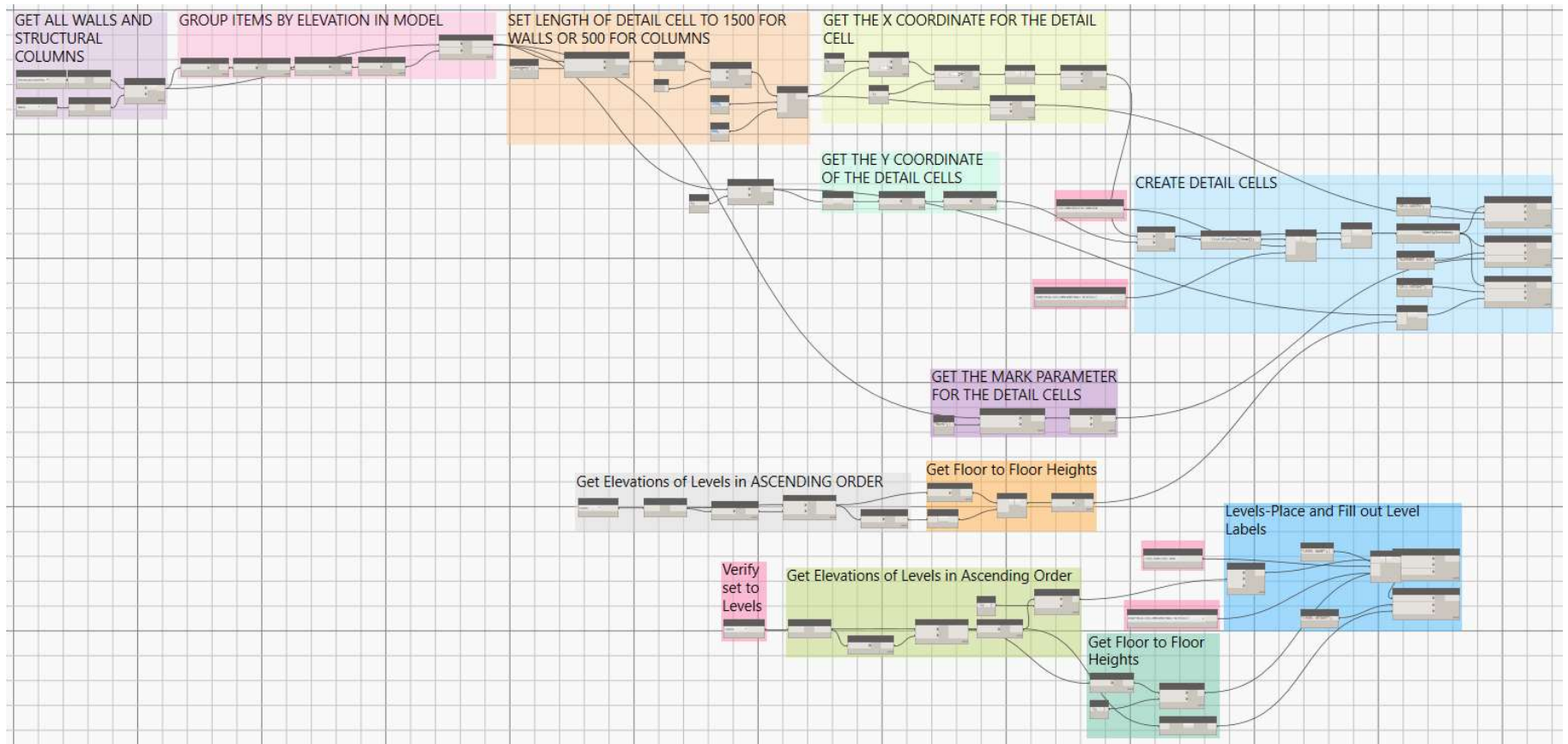
for i in input:
    listABC.append(i)

OUT = listABC
```

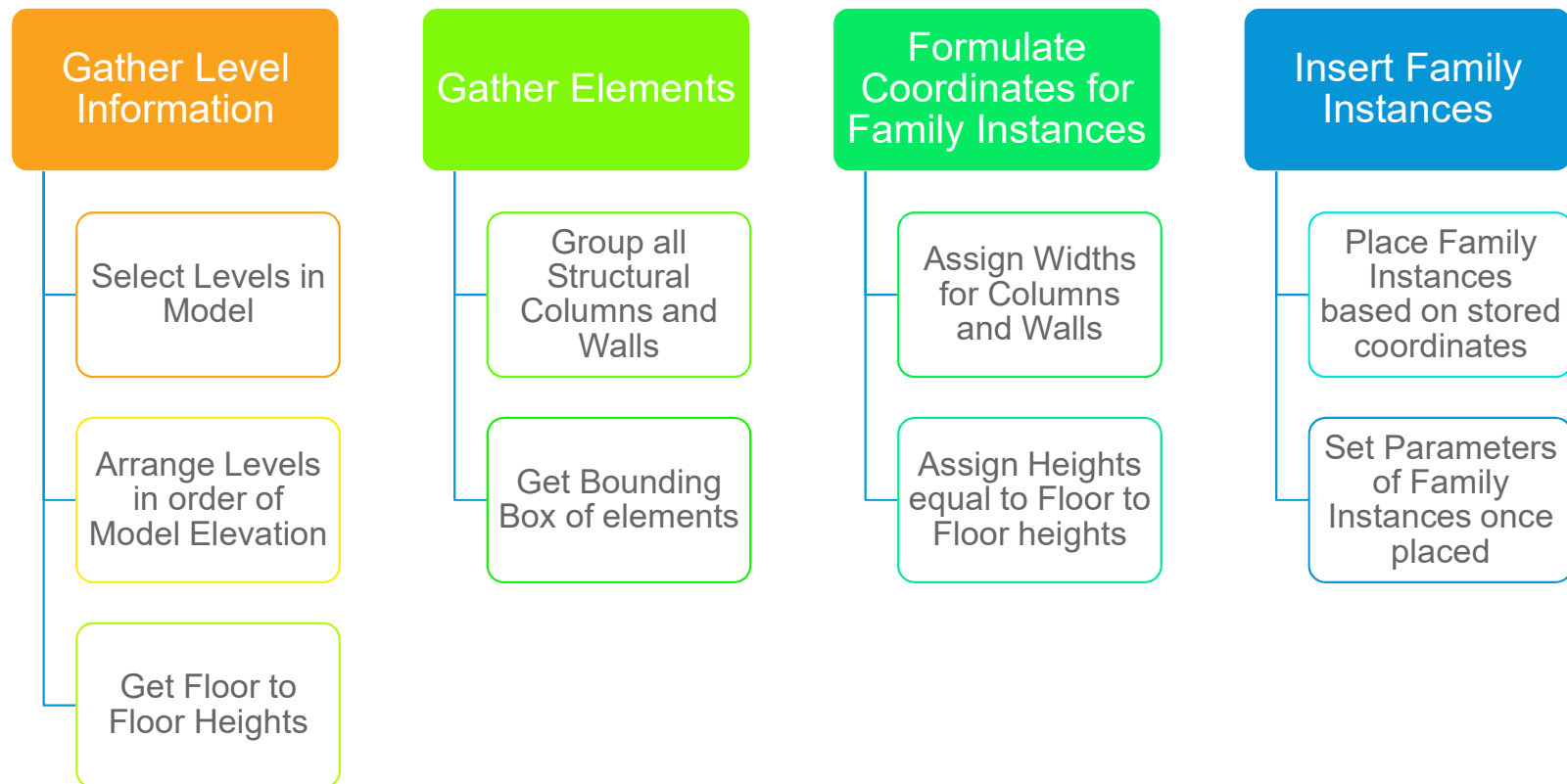
This is not a very useful piece of code in and of itself (the input and the output are the same), but the process is valuable to understand



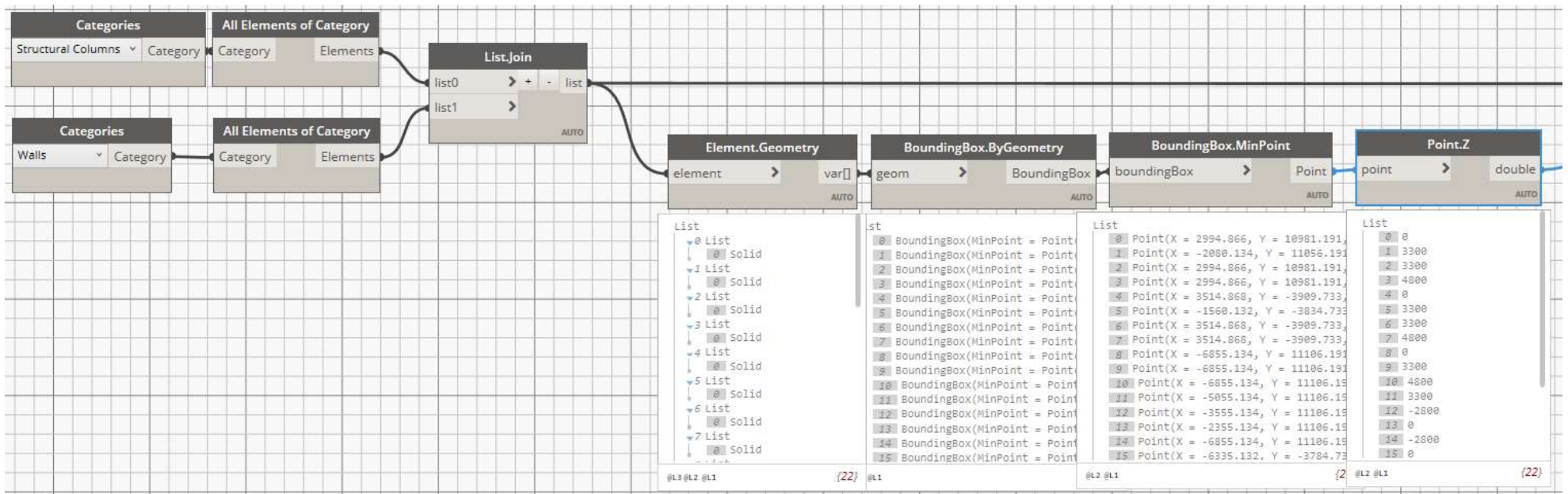
Lets Take a Look



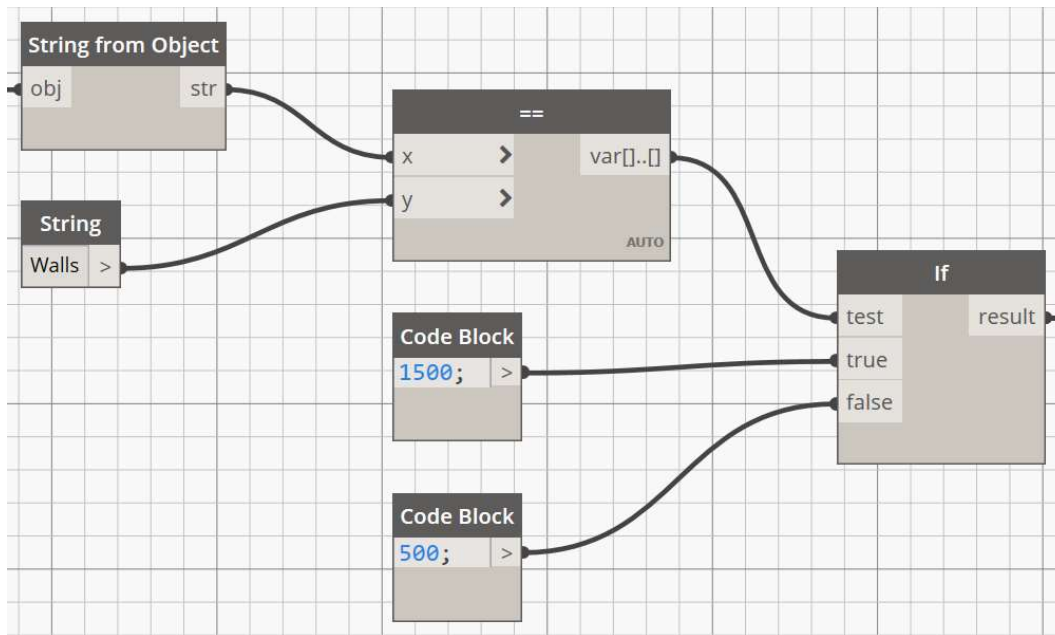
Graphical Column Schedule



Element.Geometry

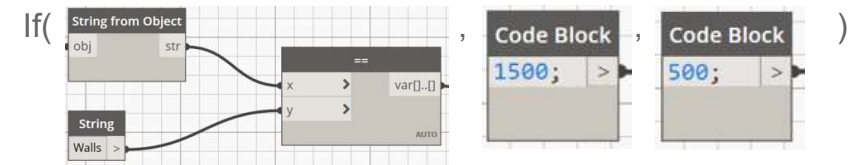


IF Statement

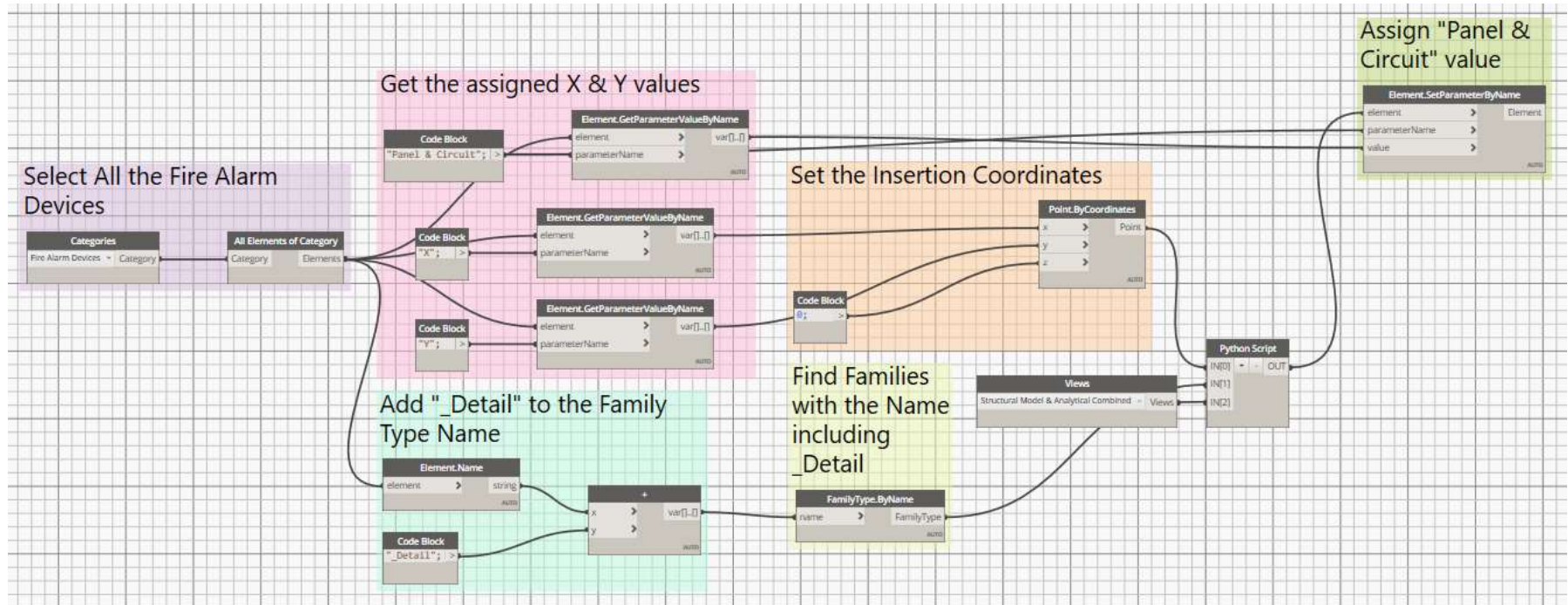


If(test,true,false)

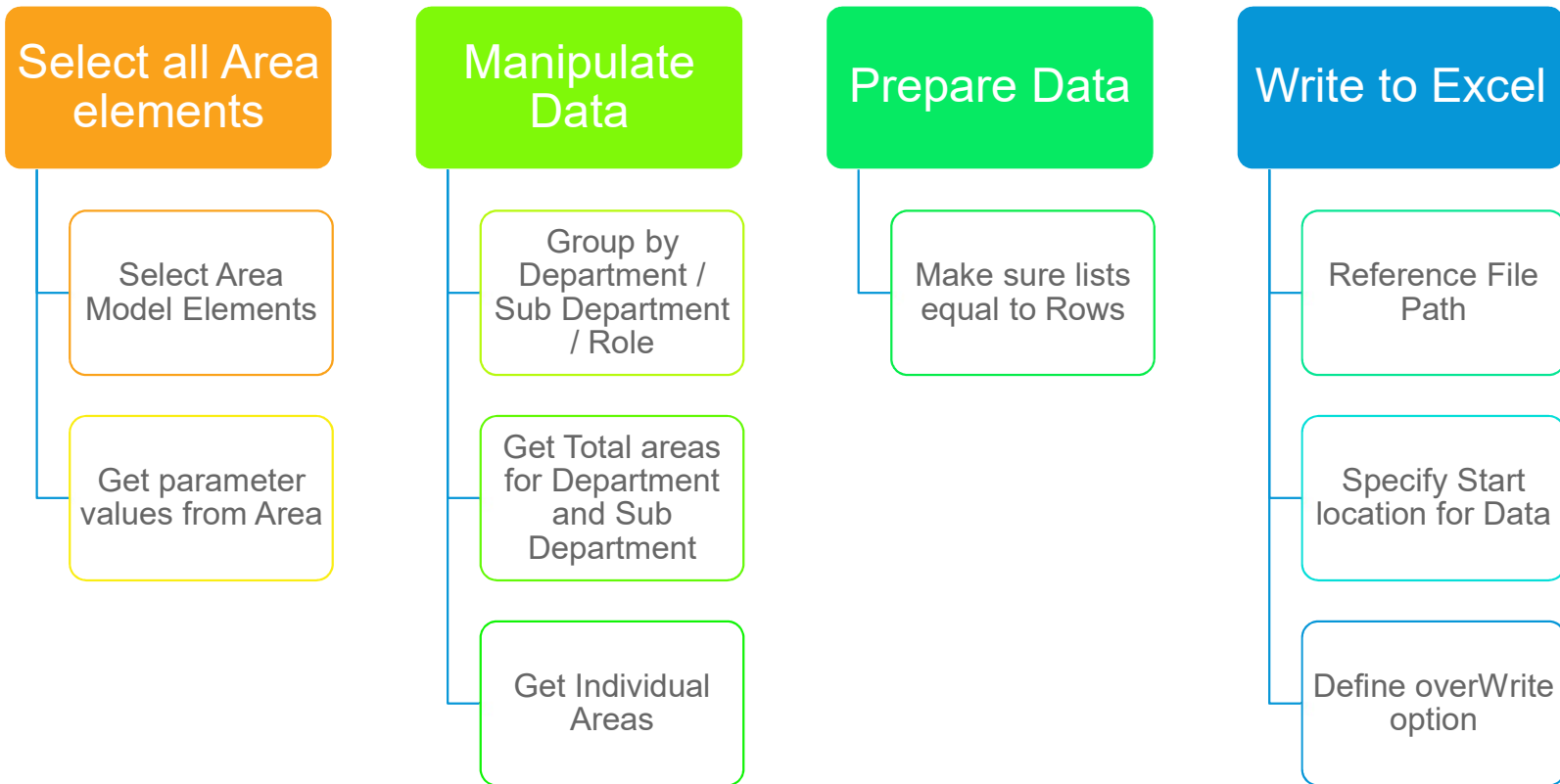
If(5>6,"Drew","Jen")



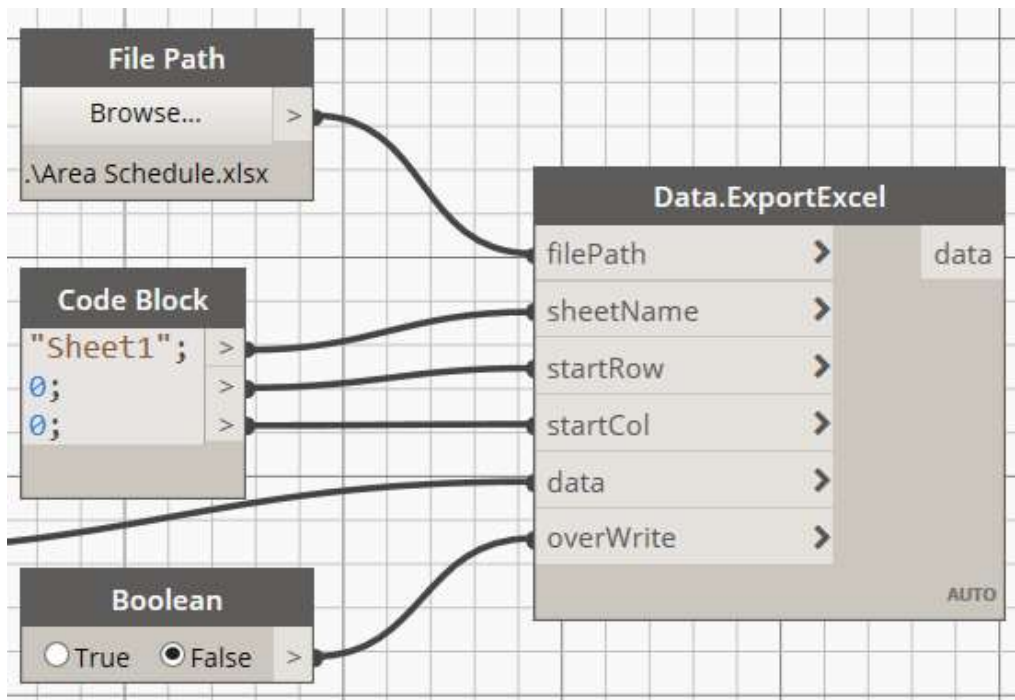
Lets Take a Look



Area Schedule

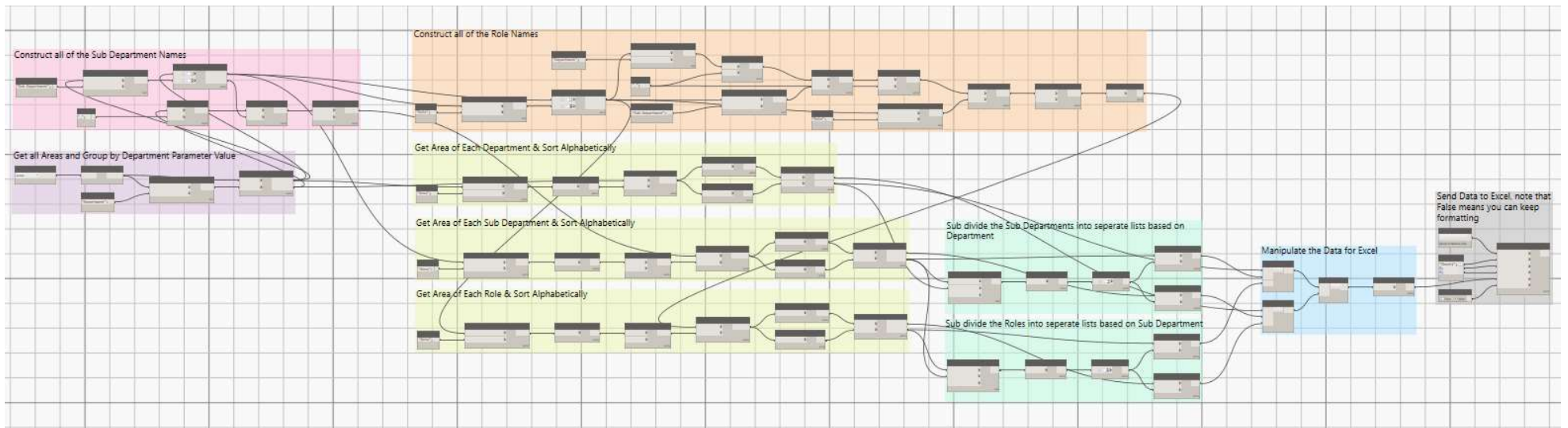


Excel Export



Note that the overWrite relates to whether you want a blank Worksheet or if you want to just put your data into an existing Worksheet

Lets Take a Look



Dynamo Workflow Recommendations

Always map out your requirements

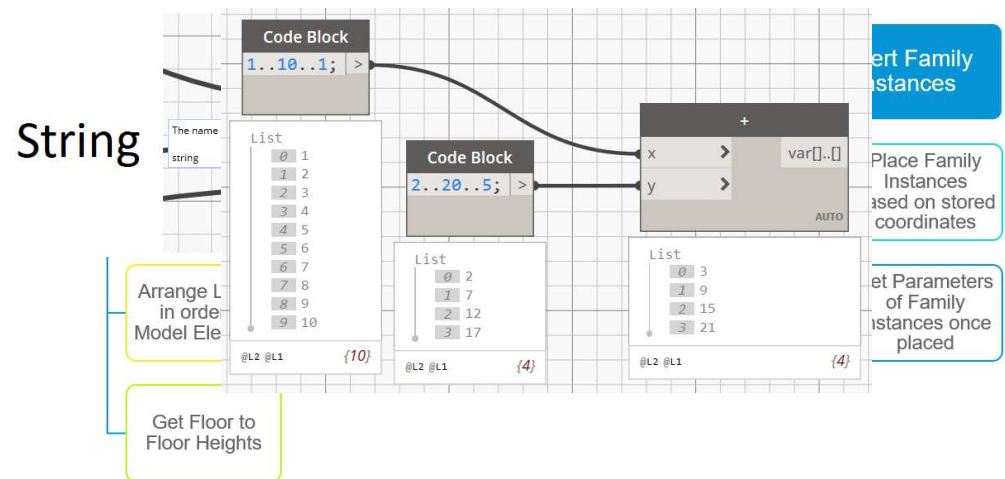
Include what inputs you need

Make sure you understand what format the inputs should be in (String, Int, FamilySymbol)

When you get into lists, make sure your list structure is matching for each input

Often good to start at the end and work backwards

Graphical Column Schedule



Include the name of the input to pass
effectively, which is breaking an
element

Q&A

Remind me to repeat the question if you don't hear it 😊



AUTODESK®

Make anything™

Autodesk and the Autodesk logo are registered trademarks or trademarks of Autodesk, Inc., and/or its subsidiaries and/or affiliates in the USA and/or other countries. All other brand names, product names, or trademarks belong to their respective holders. Autodesk reserves the right to alter product and services offerings, and specifications and pricing at any time without notice, and is not responsible for typographical or graphical errors that may appear in this document.

© 2018 Autodesk. All rights reserved.

