

Hack it 'til You Crack It:

“An Introduction to Python in Dynamo for Revit Users”

John Pierson

Design Technology Specialist
Parallax Team



hack it 'til you crack it

“PYTHON IN DYNAMO FOR BEGINNERS”

WWW.PARALLAXTEAM.COM

Summary

With Python, Dynamo users can access the Revit API (application programming interface) in a whole new way, and create additional functionality to make the making of things easier. This lab will not be an overly complex look at Python from a computer programmer's point of view. Instead, it will provide an introduction to the Python environment within Dynamo from a Revit user's perspective. As a Dynamo user, the speaker has struggled with the same scenarios and has used Python in ways never anticipated. **This lab will cover key concepts for Python, including syntax, indentation, object types, and iteration, while relating it all back to Revit terms.**

Learning Objectives

- **Gain** an introductory understanding of Python in Dynamo.
- **Learn** how to translate and understand Python terms in a Revit way.
- **Understand** how to implement standard Python methods in Dynamo.
- **Learn** how to deconstruct other's Python code for additional learning and modification.

About the Speaker



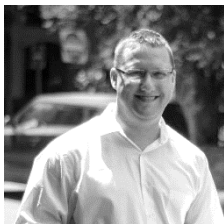
John Pierson

- **Design Technology Specialist** at Parallax Team
- **Reviteer** since 2012
- **Dynamo User** since 2014
- **Dynamo Package Developer** since 2015, (Rhythm, Bang!, DuctTape, Monocle).
- **BILTNA** Top Rated Speaker
- **Revit Certified Professional** for Architectural, Structural, MEP

The Core Team

Parallax Team is a Practice Implementation Consultancy specializing in implementation, computational design, and pretty much anything AEC related.

www.parallaxteam.com | @prlxteam

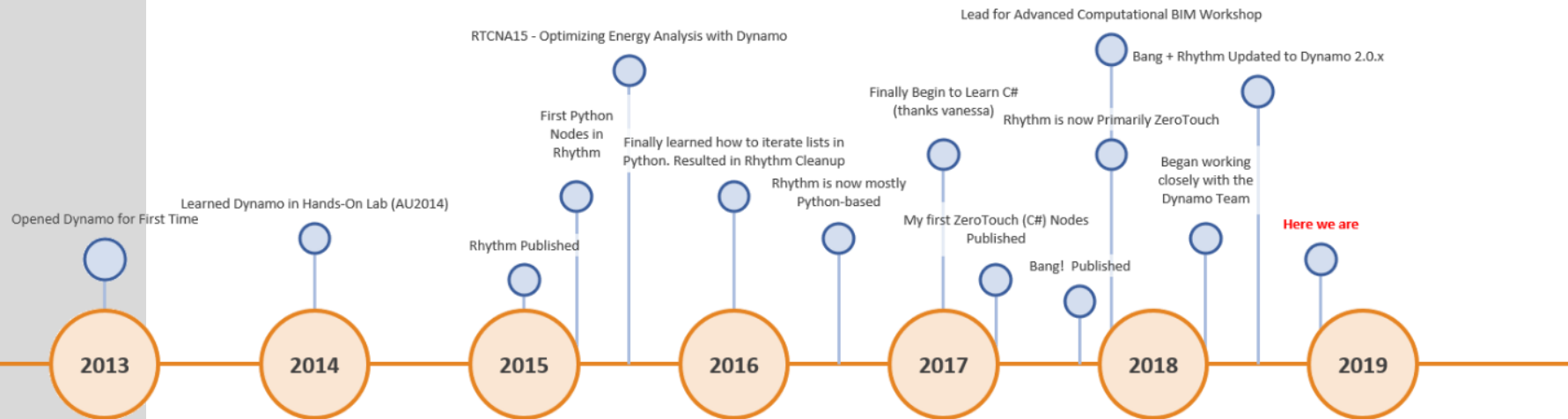


Aaron Maller
Director
@twiceroadsfool



John Pierson
Design Technology Specialist
@60secondrevit

My Dynamo Journey



Disclaimer

This class aims to demonstrate how I (as a non-formally trained programmer) learned how to utilize Python in Dynamo in a fairly efficient manner.

There is a **slight** chance that I:

- Will not get terms perfect
- Will need to reference other Python resources for answers
- Will demonstrate workflows that “*real*” programmers would consider “*questionable*”

There is a **100%** chance that we:

- Will have fun
- Will make some awesome stuff
- Leave this class with at least 1 piece of knowledge that we did not have walking in
- Leave this class with resources to be successful in our Dynamo journeys

That being said...Resources

(from some smart folks!)

- [Diving Deeper – A Beginners Look at Python in Dynamo – Sol Amour](#)
 - Sol is a Python artist. His class from AU London really breaks down the concepts and introduces them in a very deep way. In addition to his AU handout, Sol has tons of great stuff that is shared with the community on this repo.
 - <https://github.com/Amoursol/dynamoPython>
- [Untangling Python: A Crash Course on Dynamo's Python Node - Gui Talarico](#)
 - Gui's class is an amazing resource on using Python for Revit to its fullest. He covers really deep Python concepts and provides info on how to utilize outside development environments.
- Additional Resources:
 - Danny Bentley - [YouTube](#)
 - Jeremy Graham - <http://learndynamo.com/>

Words of Wisdom

(from some very smart dudes)



Anthony Hauck

@anthonyhauck

Following



Replying to @didonenov @60secondrevit @arch_laboratory

We should be open about these experiences.

@ikeough helped when he said to me,

"Almost every bug fix ends with the coder thinking he or she is an idiot for missing the obvious." It's discouraging for people to think everyone else is a genius creating prize-worthy code.

8:35 AM - 4 Aug 2018

<https://twitter.com/anthonyhauck/status/1025736916791189504>

HYPAR

Who are you?



Image of an actual dynamo user trying to figure out if they are beginner, intermediate, or advanced.

What is Dynamo?

(review)

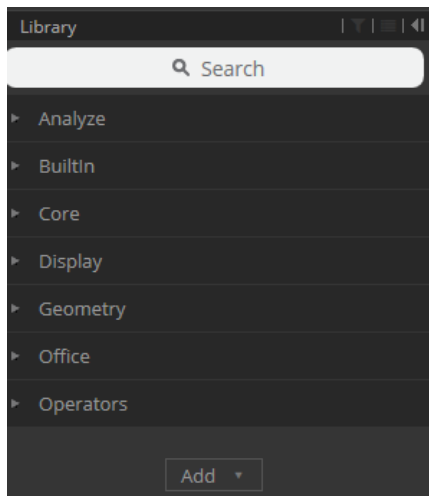
Dynamo is not only a software, but also a community of awesome people doing awesome things.

Current Versions:

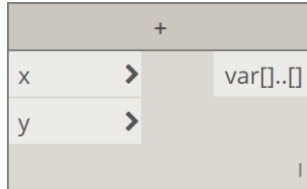
1.3.3 & 2.01 – Revit 2017 – 2019

For supported versions for older Revit versions refer to
http://dynamoprimer.com/en/08_Dynamo-for-Revit/8-1_The-Revit-Connection.html

Version Differences - 1.3.3



- Same library for several years.
- Packages bundled within the library categories

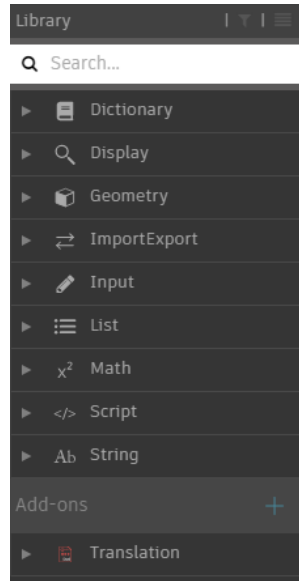


- Node in Dynamo 1.3.3

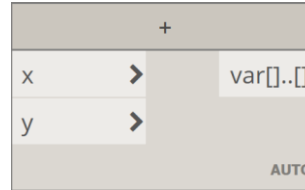
```
1 <Workspace Version="1.3.2.2480" X="-419.949656255947" Y="-
2 <NamespaceResolutionMap />
3 <Elements>
4   <Dynamo.Graph.Nodes.ZeroTouch.DSFunction guid="75531b0
5     <PortInfo index="0" default="False" />
6     <PortInfo index="1" default="False" />
7     <PortInfo index="2" default="False" />
8     <PortInfo index="3" default="False" />
9   </Dynamo.Graph.Nodes.ZeroTouch.DSFunction>
10  <Dynamo.Graph.Nodes.CustomNodes.Function guid="9dfd34a
11    <PortInfo index="0" default="True" />
12    <ID value="e620b7b6-640d-435e-982a-15fc1eeb8bdb" />
13    <Name value="Collector.PlacedRooms" />
14    <Description value="This will collect placed rooms i
15    <Inputs>
16      <Input value="toggle" />
17    </Inputs>
18    <Outputs>
19      <Output value="rooms" />
20    </Outputs>
21  </Dynamo.Graph.Nodes.CustomNodes.Function>
22  <CoreNodeModels.Input.BoolSelector guid="e2c2c8d3-50a2
23    <System.Boolean>False</System.Boolean>
```

- XML-Based file format
- Note the "<>"

Version Differences - 2.0.x



- New Appearance
- Add-ons category for custom packages

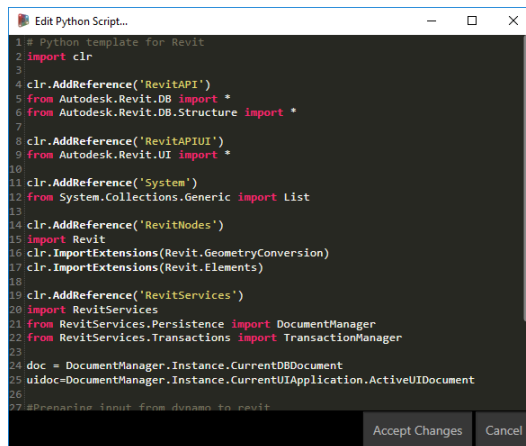


- Node in Dynamo 2.0.x
- Auto-Lacing by default

```
1 {
2   "Uuid": "652a7978-e658-4a67-91c3-fcb9a9875e9f",
3   "IsCustomNode": false,
4   "Description": null,
5   "Name": "dynamo2",
6   "ElementResolver": {
7     "ResolutionMap": {}
8   },
9   "Inputs": [],
10  "Outputs": [],
11  "Nodes": [
12    {
13      "ConcreteType": "Dynamo.Graph.Nodes.ZeroTouch.DSFunction, DynamoCore",
14      "NodeType": "FunctionNode",
15      "FunctionSignature": "**$var[]..[],var[]..[]",
16      "Id": "cbe31123ed7e4372838028bdfbdd2016",
17      "Inputs": [
18        {
19          "Id": "44b444ee5baa40160c9cfb71044f69f7",
20          "Name": "x",
21          "Description": "x value.\n\nvar[]..[]",
22          "UsingDefaultValue": false,
23          "Level": 2,
24          "UseLevels": false,
25          "KeepListStructure": false
26        },
27        {
28          "Id": "a333e56ff8e74e108c01c83a26072ab5",
29          "Name": "y",
```

- JSON-Based file format
- Note the "{ }

But, most **important** (for this class)

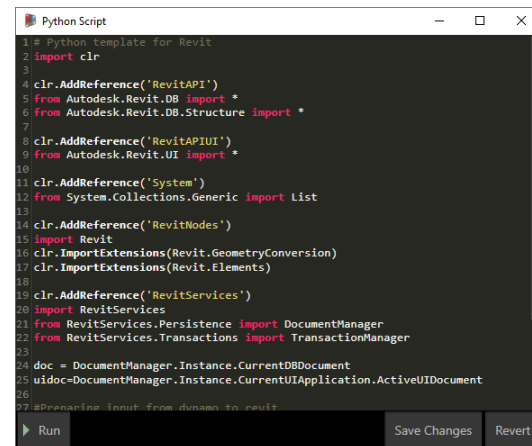


A screenshot of the 'Edit Python Script...' window in Dynamo 1.3.3. The window is titled 'Edit Python Script...' and contains a Python script template for Revit. The script includes imports for clr, Autodesk.Revit.DB, Autodesk.Revit.DB.Structure, Autodesk.Revit.UI, System.Collections.Generic, and RevitNodes. It also includes imports for Revit, RevitServices, and DocumentManager. The script is displayed in a dark-themed editor with line numbers. At the bottom of the window, there are two buttons: 'Accept Changes' and 'Cancel'.

```
1 # Python template for Revit
2 import clr
3
4 clr.AddReference('RevitAPI')
5 from Autodesk.Revit.DB import *
6 from Autodesk.Revit.DB.Structure import *
7
8 clr.AddReference('RevitAPIUI')
9 from Autodesk.Revit.UI import *
10
11 clr.AddReference('System')
12 from System.Collections.Generic import List
13
14 clr.AddReference('RevitNodes')
15 import Revit
16 clr.ImportExtensions(Revit.GeometryConversion)
17 clr.ImportExtensions(Revit.Elements)
18
19 clr.AddReference('RevitServices')
20 import RevitServices
21 from RevitServices.Persistence import DocumentManager
22 from RevitServices.Transactions import TransactionManager
23
24 doc = DocumentManager.Instance.CurrentDBDocument
25 uidoc = DocumentManager.Instance.CurrentUIApplication.ActiveUIDocument
26
27 #Preparsing input from dynamo to revit
28
```

Python Editor in 1.3.3 and prior

- No support for templates
- Modal Window – Meaning you cannot work outside of it



A screenshot of the 'Python Script' window in Dynamo 2.0.x. The window is titled 'Python Script' and contains the same Python script template for Revit as the previous screenshot. The script is displayed in a dark-themed editor with line numbers. At the bottom of the window, there are three buttons: 'Run', 'Save Changes', and 'Revert'.

```
1 # Python template for Revit
2 import clr
3
4 clr.AddReference('RevitAPI')
5 from Autodesk.Revit.DB import *
6 from Autodesk.Revit.DB.Structure import *
7
8 clr.AddReference('RevitAPIUI')
9 from Autodesk.Revit.UI import *
10
11 clr.AddReference('System')
12 from System.Collections.Generic import List
13
14 clr.AddReference('RevitNodes')
15 import Revit
16 clr.ImportExtensions(Revit.GeometryConversion)
17 clr.ImportExtensions(Revit.Elements)
18
19 clr.AddReference('RevitServices')
20 import RevitServices
21 from RevitServices.Persistence import DocumentManager
22 from RevitServices.Transactions import TransactionManager
23
24 doc = DocumentManager.Instance.CurrentDBDocument
25 uidoc = DocumentManager.Instance.CurrentUIApplication.ActiveUIDocument
26
27 #Preparsing input from dynamo to revit
28
```

Python Editor in Dynamo 2.0.x

- Support for templates! (thanks Radu)
http://primer.dynamobim.org/en/10_Custom-Nodes/10-6_Python-Templates.html
- Allows “on-the-fly” execution of Python Scripts.
- Allows for multiple Python windows
- Edit and run graph with window open

What is Python?

- Python is an open-source, general-purpose programming language.
- Does not require compiling to be tested.
- Provides quick “try it out, see what it does” gratification.
- Revit Comparison: Similar to editing families, loading into project for testing, repeat.

What is Iron Python?

- A version of Python that is created in C#.
- Allows for use of the Common Language Runtime to talk to other .NET applications/libraries. (<http://ironpython.net/>)
- Version 2.7 Installed alongside Dynamo.

Let's do this



Walk

Opening Python
Key Terms + Syntax
Variables
Changing single item



Jog

Variables
Working with Lists
Output Methods
Templates



Run

Reference Revit API
API Methods
Wrap/Unwrap
Templates

Anatomy of Python Window

```
# 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
```

Notes are designated with pound “#” sign
These are the OOTB included imports
(more on this later)

General information regarding how to get data in. *The existence of the variable is to let you know that the inputs are all a list.

Return something

https://twitter.com/arch_laboratory/status/954398424039854080

Example

Uppercase Strings

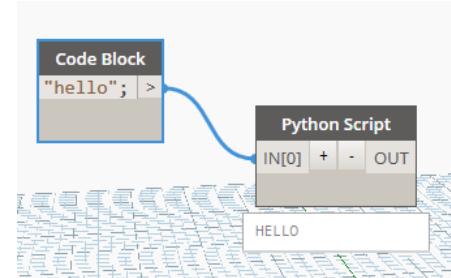
- Access Python methods for our use.
(https://www.tutorialspoint.com/python/string_upper.htm)
- Use Python inputs via IN[0] or whatever it is.

```
# 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

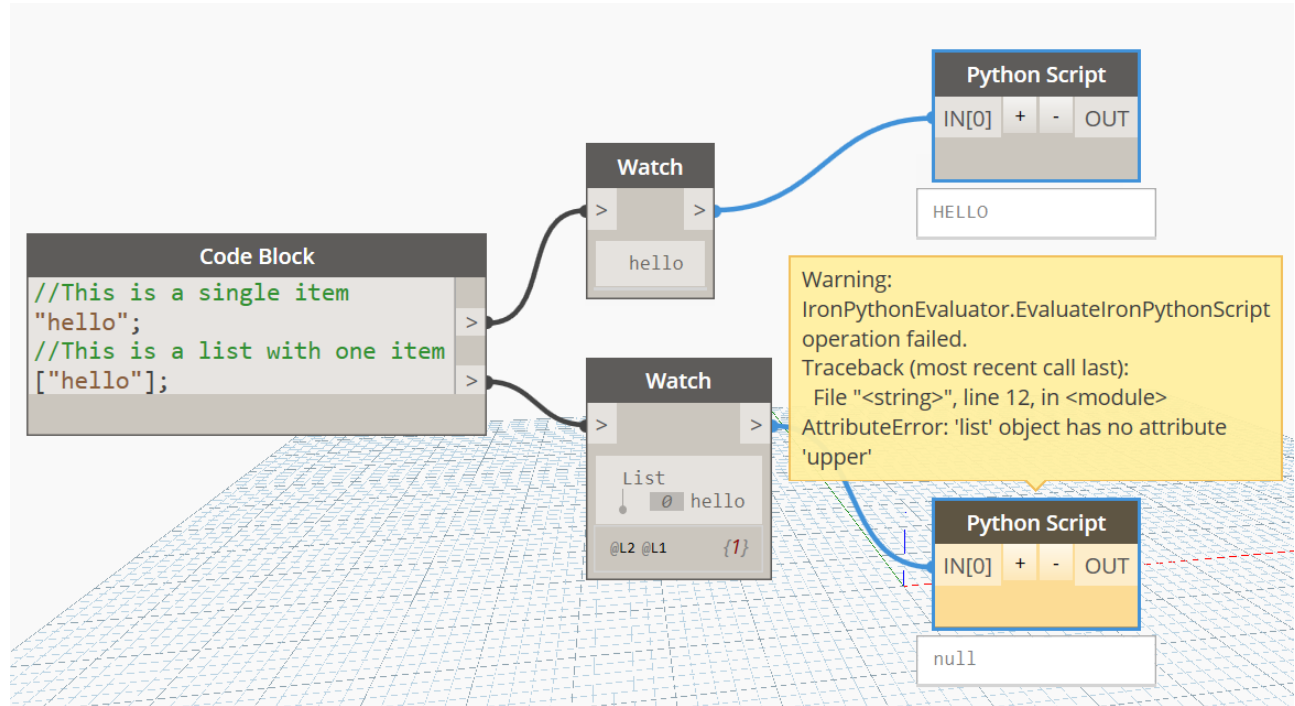
# Assign your output to the OUT variable.
OUT = IN[0].upper()
```



Why the “()”? – this is a function(action) in python, and well, it showed it in the link above, so do it.

Example

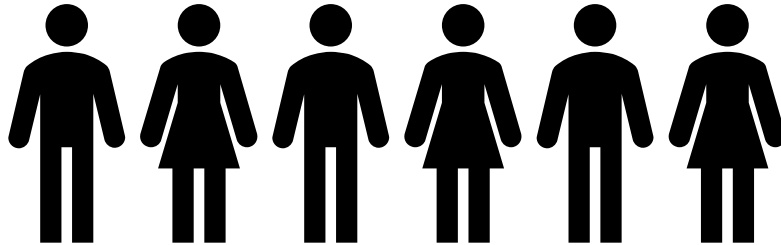
Working with Lists



Example

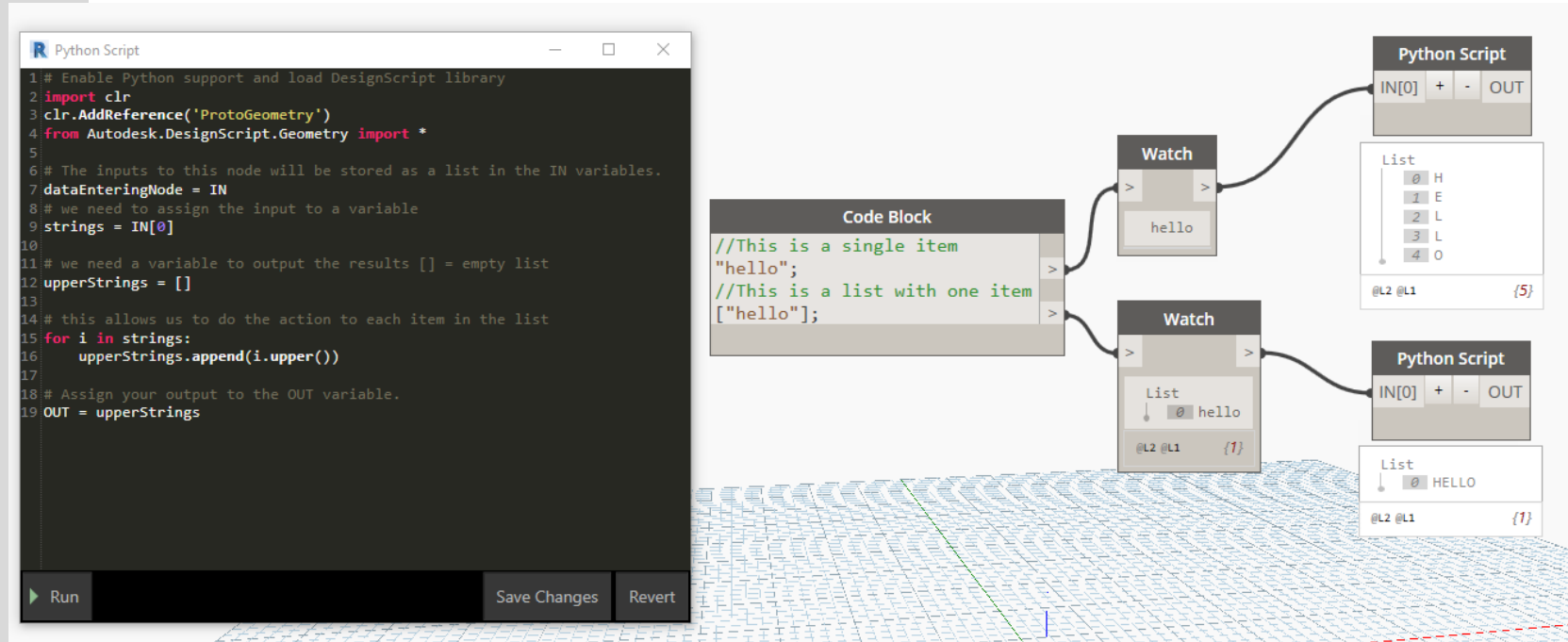
Uppercase Strings

- What is your name?



Example

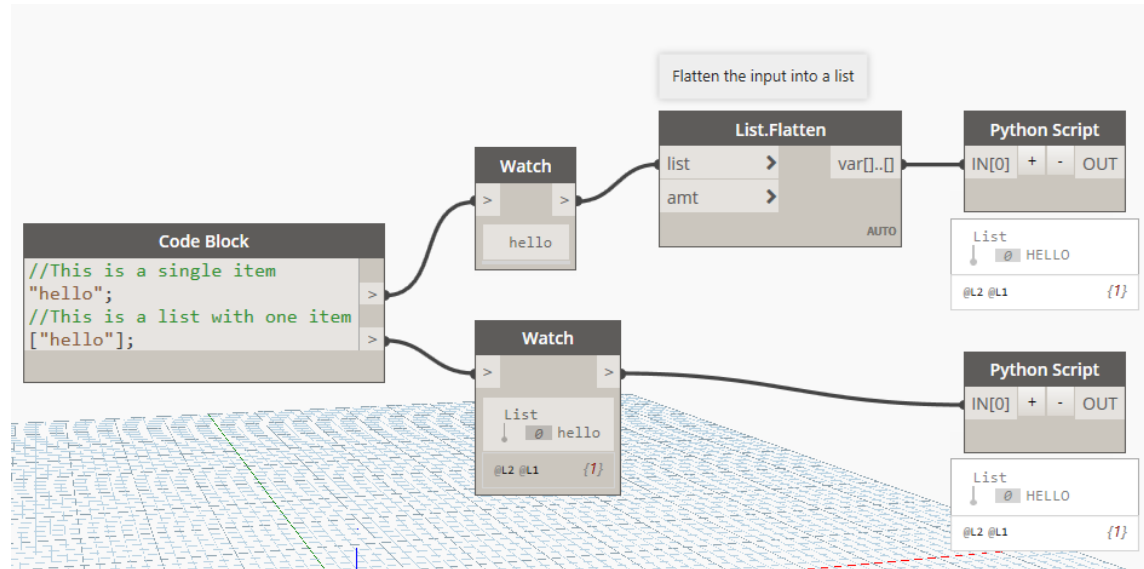
Uppercase List of Strings



Example

Uppercase List of Strings

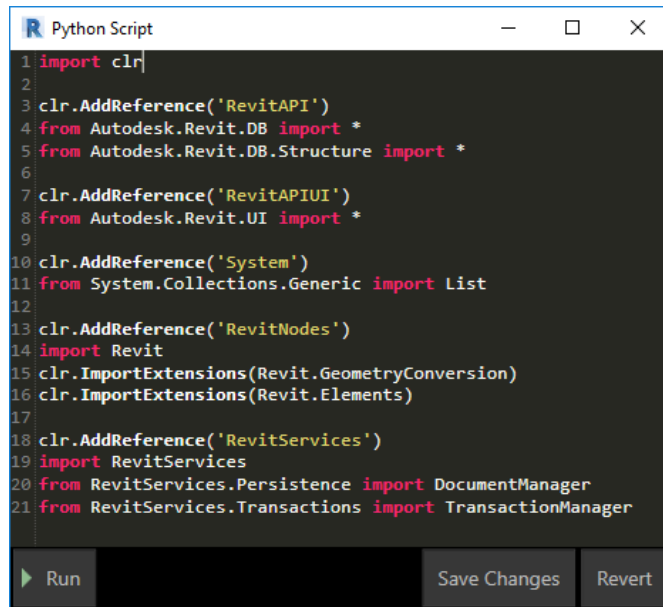
- Work with an assumption of receiving a list.
- Convert the input to a list yourself.



well.

THAT WASN'T SO BAD. RIGHT?

What in the heck are **Imports?**

A screenshot of a Python Script window with a dark background and light-colored text. The window title is "Python Script". The code is as follows:

```
1 import clr
2
3 clr.AddReference('RevitAPI')
4 from Autodesk.Revit.DB import *
5 from Autodesk.Revit.DB.Structure import *
6
7 clr.AddReference('RevitAPIUI')
8 from Autodesk.Revit.UI import *
9
10 clr.AddReference('System')
11 from System.Collections.Generic import List
12
13 clr.AddReference('RevitNodes')
14 import Revit
15 clr.ImportExtensions(Revit.GeometryConversion)
16 clr.ImportExtensions(Revit.Elements)
17
18 clr.AddReference('RevitServices')
19 import RevitServices
20 from RevitServices.Persistence import DocumentManager
21 from RevitServices.Transactions import TransactionManager
```

At the bottom of the window, there are three buttons: "Run" (with a green play icon), "Save Changes", and "Revert".

- Imports are a way to load additional libraries into your code.
- For us Reviteers, we can think of this as the same as loading content into a file.
- Next we will break down what these mean.

Breaking Down Imports

```
# import common language runtime (allows us to load other .NET libraries)
import clr

# clr.AddReference loads and imports .net assembly(dll) as python module and import all of its classes
clr.AddReference('RevitAPI')
from Autodesk.Revit.DB import *

# import Revit UI classes
clr.AddReference('RevitAPIUI')
from Autodesk.Revit.UI import *

# to handle lists
clr.AddReference('System')
from System.Collections.Generic import List

# import Dynamo Revit nodes and converters
clr.AddReference('RevitNodes')
import Revit
clr.ImportExtensions(Revit.GeometryConversion)
clr.ImportExtensions(Revit.Elements)


# import Transaction methods and Document helpers
clr.AddReference('RevitServices')
import RevitServices
from RevitServices.Persistence import DocumentManager
from RevitServices.Transactions import TransactionManager
```

Transactions?



Charlie knows about transactions!

Do I **have** to memorize that?

- Heck no.
- Template provided that covers most of your Dynamo Revit workflows.  [PythonTemplate.py](#)
- Auto load on new Python script placement.

Python **Template** in Dynamo

- Adding templates to get going faster.
 - http://dynamoprimer.com/en/10_Custom-Nodes/10-6_Python-Templates.html
 - To get the path quickly use, AS224711-00-GetCurrentUserPythonTemplatePath_end.dyn in the dataset

Wrapped versus Unwrapped

Tortilla
Toasted
Warm
More Inside?



Rice
Tomatoes
Cheese
Steak
Guac
Etc.



Wrapped versus Unwrapped

Select Model Element

Change	Element
Element : 149293	

Wall 149293

The user readable Dynamo way of things

"Wrapped" Dynamo Revit Element

IN[0] + - OUT

Object.Type

obj > string

AUTO

Revit.Elements.Wall

What is potentially possible...

List

- 0 BoundingBox
- 1 ByCurveAndHeight
- 2 ByCurveAndLevels
- 3 ByFace
- 4 Curves
- 5 Dispose
- 6 ElementCurveReferences
- 7 ElementFaceReferences
- 8 ElementType
- 9 Equals
- 10 Faces
- 11 Geometry
- 12 GetCategory
- 13 GetHashcode
- 14 GetLocation
- 15 GetMaterials
- 16 GetParameterValueName

@L2 @L1 {54}

The internal Revit API way of things

"Unwrapped" Internal Revit Element

IN[0] + - OUT

"Unwrapped" Internal Revit Element

IN[0] + - OUT

Autodesk.Revit.DB.Wall

What is potentially possible...

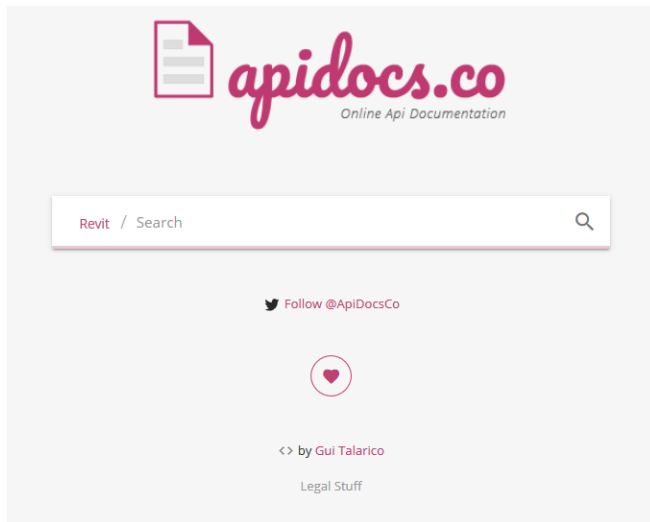
List

- 0 ArePhasesModifiable
- 1 AssemblyInstanceId
- 2 CanBeHidden
- 3 CanBeLocked
- 4 CanDeleteSubelement
- 5 CanHaveAnalyticalModel
- 6 CanHaveTypeAssigned
- 7 Category
- 8 ChangeTypeId
- 9 Create
- 10 CreatedPhaseId
- 11 CurtainGrid
- 12 DeleteEntity
- 13 DeleteSubelement
- 14 DeleteSubelements
- 15 DemolishedPhaseId
- 16 DesignOption

@L2 @L1 {103}

I wish Dynamo had a node for **[blank]**.

- First, check if it is possible in the Revit API with API Docs by Gui Talarico.



Revit Example

Element Pinned

- [API Docs](#) returns the following
- This is a property on an element of type, *Autodesk.Revit.DB.Element*
 - This means that we need to convert the selected element to an internal version.
- { [get](#); [set](#) } means that we can query the value or change it.

Pinned Property ☆

Revit 2018 API

[Element Class](#)

[See Also](#)

Identifies if the element has been pinned to prevent changes.

Namespace: [Autodesk.Revit.DB](#)

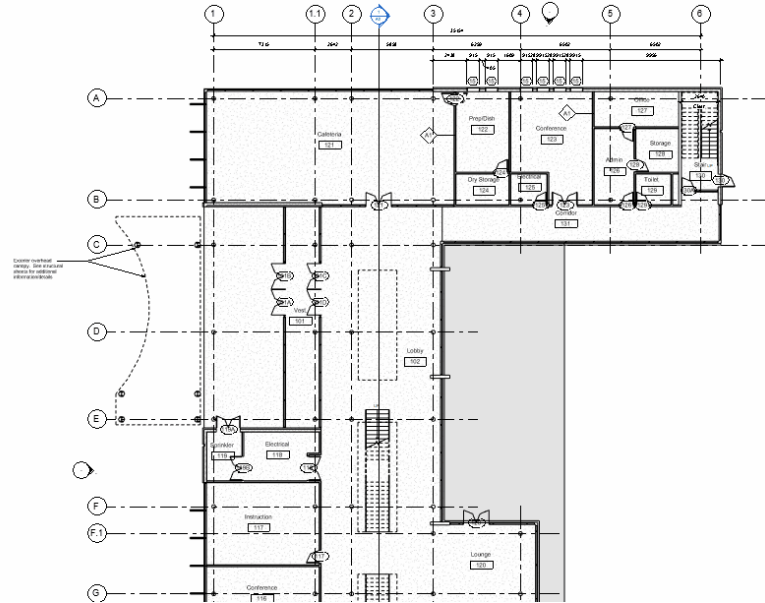
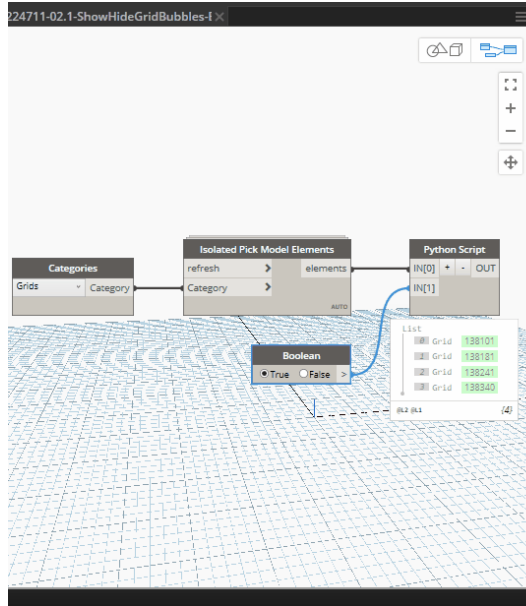
Assembly: RevitAPI (in RevitAPI.dll) Version: 18.0.0.0 (18.2.0.0)

Syntax

C#

```
public bool Pinned { get; set; }
```

Revit Example w/ Grid Bubbles



Revit Example

Find Doors in Walls

With Revit 2018.1, we were given the ability to find out what elements are dependent on another.

GetDependentElements Method ☆

Revit 2018 API

[Element Class](#) | [See Also](#)

Get all elements that, from a logical point of view, are the children of this Element.

Namespace: Autodesk.Revit.DB

Assembly: RevitAPI (in RevitAPI.dll) Version: 18.0.0.0 (18.2.0.0)

Since: 2018.1

Lets build a node to implement this.

Finding problems to solve.

Part of learning is practicing this stuff over and over. Often times an issue is not having enough workflows to experiment with. Here are some resources to get ideas.

Finding problems to solve. Forums.

- The [Dynamo](#) forum: When people have questions this is where they go very often.
- [Revit Forum](#): Revit forum is another great site to get help and help people.

Finding problems to solve. Twitter.



Purvi Irwin @BIMchiq · 1h

I have a building in #Revit with grids that are radial. How do I extend them all down to a couple levels lower? I can't see them in elevation unless I draw a section perpendicular to EACH ONE. I need them on lower levels. Scope box makes them all too long - bldg is an arc.



4



Tom Kunsman

@tkunsman

Following

Replying to @BIMchiq

@60secondrevit thoughts??



1:27 PM - 2 Nov 2018

thank you.

NOW, GO DO GREAT DYNAMO THINGS

WWW.PARALLAXTEAM.COM

@prlxteam | @twiceroadsfool | @60secondrevit



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.

