



**The Boiler Plate**  
Python in Dynamo

# Related videos



MAKING A CUSTOM  
NODE



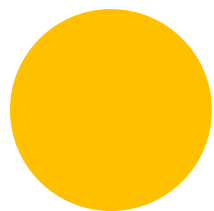
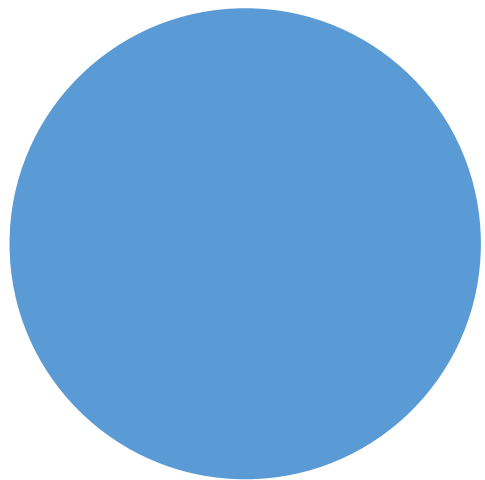
MAKING A CUSTOM  
PACKAGE



PYTHON  
FUNDAMENTALS



PYTHON IN  
DYNAMO (SERIES)



# Dynamo and Python

## The boiler plate

# This video

- Explain the boiler plate
- Explain each common piece of the boiler plate
- Show how to build your Python 'starter' template

```
Python Script
1 # Made by Gavin Crump
2 # Free for use
3 # BIM Guru, www.bimguru.com.au
4
5 #Reference for script name: https://github.com/Amoursol/dynamoPython/blob/master/dynamoAPI/dynamoAPICurrentGraphName.py
6 #Thanks to John Pierson for sharing the code for Dynamo file data
7
8 #Add CLR and Dynamo services for its API
9 import clr
10 clr.AddReference('DynamoRevitDS')
11 import Dynamo
12 from System import Environment
13
14 #Add document manager from Revit services
15 clr.AddReference('RevitServices')
16 import RevitServices
17 from RevitServices.Persistence import DocumentManager
18
19 # access to the current Dynamo instance and workspace
20 DynamoCurrent = Dynamo.Applications.DynamoRevit()
21 ActiveScript = DynamoCurrent.RevitDynamoModel.CurrentWorkspace
22 DynamoVersion = DynamoCurrent.RevitDynamoModel.Version
23
24 # ensure version 2.X, get name
25 if DynamoVersion.StartsWith("2."):
26     ScriptName = ActiveScript.Name + ".dyn"
27 else:
28     ScriptName = "Version not supported"
29
30 # get system user name
31 UserName = Environment.UserName
32
33 # get active document name and path
```

What is a  
boiler plate?



# Syntax

## import

# comments with a hashtag

`import Namespace`

`from Namespace import *`

`from Namespace import thing`

Woah...  
Let's break  
this down!

A screenshot of a 'Python Script' window in a software application, likely Revit. The window has a title bar with a blue 'R' icon and standard minimize, maximize, and close buttons. The script is a Python file with line numbers 1 through 53. It includes comments at the top about being free for use and attributing to BIM Guru. The script imports the clr module and sys, then appends a path to the IronPython 2.7 Lib directory. It then imports System, Array, and generic types from System.Collections.Generic. Next, it adds references to 'ProtoGeometry', 'RevitNodes', 'RevitServices', 'RevitAPI', and 'RevitAPIUI'. It then imports various classes from Autodesk.Revit.DB and Autodesk.Revit.UI, including DocumentManager, TransactionManager, Application, and ActiveUIDocument. The script sets up variables for the current document, application, and active document. It then prepares input from Dynamo to Revit, either by unwrapping an element or using IN[0]. Finally, it enters a transaction to perform actions and marks the transaction as done. The script ends with comments about outputting and changing the element to Dynamo for export, with options to mark it as Revit-owned or non-Revit-owned. At the bottom of the window, there are three buttons: 'Run' (with a green play icon), 'Save Changes', and 'Revert'.

See it a bit like  
borrowing books  
from a library for  
an assignment

clr = the library





Important  
Try not to  
borrow a  
whole book!



Python Script

```
1 # Boilerplate text
2 import clr
3
4 clr.AddReference("RevitAPI")
5 clr.AddReference("RevitAPIUI")
6 import Autodesk
7 from Autodesk.Revit.DB import *
```



Python Script

```
1 # Boilerplate text
2 import clr
3
4 clr.AddReference("RevitAPI")
5 clr.AddReference("RevitAPIUI")
6 import Autodesk |
7 from Autodesk.Revit.DB import Wall
```



Important  
Don't check  
out the  
whole library

You might  
not even  
need it!

```
Python Script
2 # Free for use
3 # BIM Guru, www.bimguru.com.au
4
5 # Boilerplate text
6 import clr
7
8 import sys
9 sys.path.append('C:\Program Files (x86)\IronPython 2.7\Lib')
10
11 import System
12 from System import Array
13 from System.Collections.Generic import *
14
15 clr.AddReference('ProtoGeometry')
16 from Autodesk.DesignScript.Geometry import *
17
18 clr.AddReference("RevitNodes")
19 import Revit
20 clr.ImportExtensions(Revit.Elements)
21 clr.ImportExtensions(Revit.GeometryConversion)
22
23 clr.AddReference("RevitServices")
24 import RevitServices
25 from RevitServices.Persistence import DocumentManager
26 from RevitServices.Transactions import TransactionManager
27
28 clr.AddReference("RevitAPI")
29 clr.AddReference("RevitAPIUI")
30
31 import Autodesk
32 from Autodesk.Revit.DB import *
33 from Autodesk.Revit.UI import *
34
35 # Current doc/app/ui
36 doc = DocumentManager.Instance.CurrentDBDocument
37 uiapp = DocumentManager.Instance.CurrentUIApplication
38 app = uiapp.Application
39 uidoc = uiapp.ActiveUIDocument
40
41 # Preparing input from dynamo to revit
42 # element = UnwrapElement(IN[0])
43 # or
44 element = IN[0]
45
46 # Do some action in a Transaction
47 TransactionManager.Instance.EnsureInTransaction(doc)
48 # your actions
49 TransactionManager.Instance.TransactionTaskDone()
50
51 # Output and Changing element to Dynamo for export
52 # <element>.ToDSType(True), #Not created in script, mark as Revit-
   owned
53 # <element>.ToDSType(False) #Created in script, mark as non-Revit-
   owned
```

Run Save Changes Revert



clr

```
import clr
```

## Common language runtime

- Gets it all started
- Connects python and .net
- Allows you to call on .dlls

dll's

## 'Dynamic link libraries'

- Called on by code
- Like pre-built 'instructions'

C:\Program Files\Autodesk  
 \ \*Revit xxxx\* \AddIns  
 \DynamoForRevit

- Root folder
- Revit folder

# RevitNodes

```
clr.AddReference("RevitNodes")
```

```
import Revit
```

```
clr.ImportExtensions(Revit.Elements)
```

```
clr.ImportExtensions(Revit.GeometryConversion)
```

- Reference the dll
- Import Revit namespace
- Add more element nodes
- Add more geometry nodes

# ProtoGeometry

```
clr.AddReference('ProtoGeometry')  
from Autodesk.DesignScript.Geometry import *
```

- Reference the dll
- Dynamo geometry classes

# Revit API

## RevitAPIUI

```
clr.AddReference("RevitAPI")  
clr.AddReference("RevitAPIUI")
```

```
import Autodesk  
from Autodesk.Revit.DB import *  
from Autodesk.Revit.UI import *
```

- Reference API dll's
- Import Autodesk namespace
- Load Revit API Classes
- Load Revit API UI Classes

# RevitServices

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

- Dynamo classes
- Dynamo's attentive document
- Dynamo making changes to the Revit model (transactions)



# Current Document

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

*And maybe*

```
uiapp = DocumentManager.Instance.CurrentUIApplication
```

```
app = uiapp.Application
```

```
uidoc = uiapp.ActiveUIDocument
```

- Current document
- 'doc' is now a variable

# Transaction Start/End

```
TransactionManager.Instance.EnsureInTransaction(doc)
```

```
# your actions
```

```
TransactionManager.Instance.TransactionTaskDone()
```

- If changing Revit model
- 'Checks out' a spot in the db transaction queue
- Usually near script end



# Syntax Help!

`dir(object)`

Shows all available methods.

`object.__doc__`

Shows the documentation.

# Templating

```
C:\Users\ *user name*  
\AppData\Roaming\Dynamo  
\Dynamo Revit\ *version*
```

# Templating

Dynamo Python Primer

<https://dynamopythonprimer.gitbook.io/>

By Oliver Green

Amoursol (github)

<https://github.com/Amoursol/dynamoPython/tree/master/pythonTemplates>

By Sol Amour

# Future Videos

Basic  
samples

Revit API

Advanced  
samples

And  
more!



Files are on  
github

<https://github.com/aussieBIMguru>



**The Boiler Plate**  
Python in Dynamo