REVIT PURE PRESENTS

PAMPHLETS

ISSUE #15 / WINTER 2020

pyREVIT

revitpure.com

by: Nicolas Catellier, Architect

PAMPHLETS COLLECTION

ISSUE #1
WORKSETS

ISSUE #2
SCHEDULES

ISSUE #3
PHASES

ISSUE #4
LINK CAD

ISSUE #5
FILTERS

ISSUE #6
VIRTUAL REALITY

ISSUE #7
3D VIEWS

ISSUE #8
PLAN NOTES

ISSUE #9
COORDINATES

ISSUE #10
SCOPE BOXES

ISSUE #11
FINISHES

ISSUE #12
LINE WEIGHTS

PAMPHLETS COLLECTION

ISSUE #13
DOORS

ISSUE #14
SHARED SITES

ISSUE #15
pyREVIT

PAMPHLETS

ISSUE #15 / WINTER 2020 / pyREVIT

WHAT IS THIS "PAMPHLET" ?

Revit Pure Pamphlets are published 4 times a year by email. Each edition covers a very specific Revit theme. We like to pick themes that are complex and confusing. Our job is to make these topics simple for you.

WHY pyREVIT ?

pyRevit is the best Revit plugin you can install, and it is absolutely free. It is amazing for two reasons:

1- The basic tools included are amazing.
2- It allows you to create your own tools using the Python language.

In this pamphlet, we will explore both aspects. First, you will get the list of our favorite add-in tools and how to use them. Then, you will learn how to create your own basic extension using the Python language.

INSTALLING pyREVIT

First, make sure pyRevit is installed on your computer. Check out this link for the latest version. Click on Asset and download the .exe file.

https://github.com/eirannejad/pyRevit/releases

pyRevit is developped by Ehsan Iran-Nejad, who is based in Portland, Oregon. You can support the project on Patreon here:

https://www.patreon.com/pyrevit

PAMPHLETS

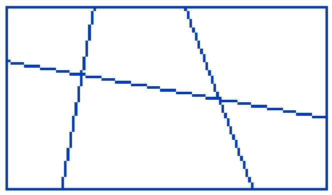
ISSUE #15 / WINTER 2020 / pyREVIT

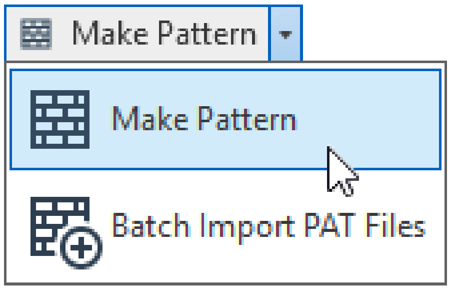
TOP 10 MOST USEFUL pyREVIT TOOLS

If you have tried to find the right Revit plugin in the past, you know it can be quite a pain in the ass.

1- MAKE PATTERN

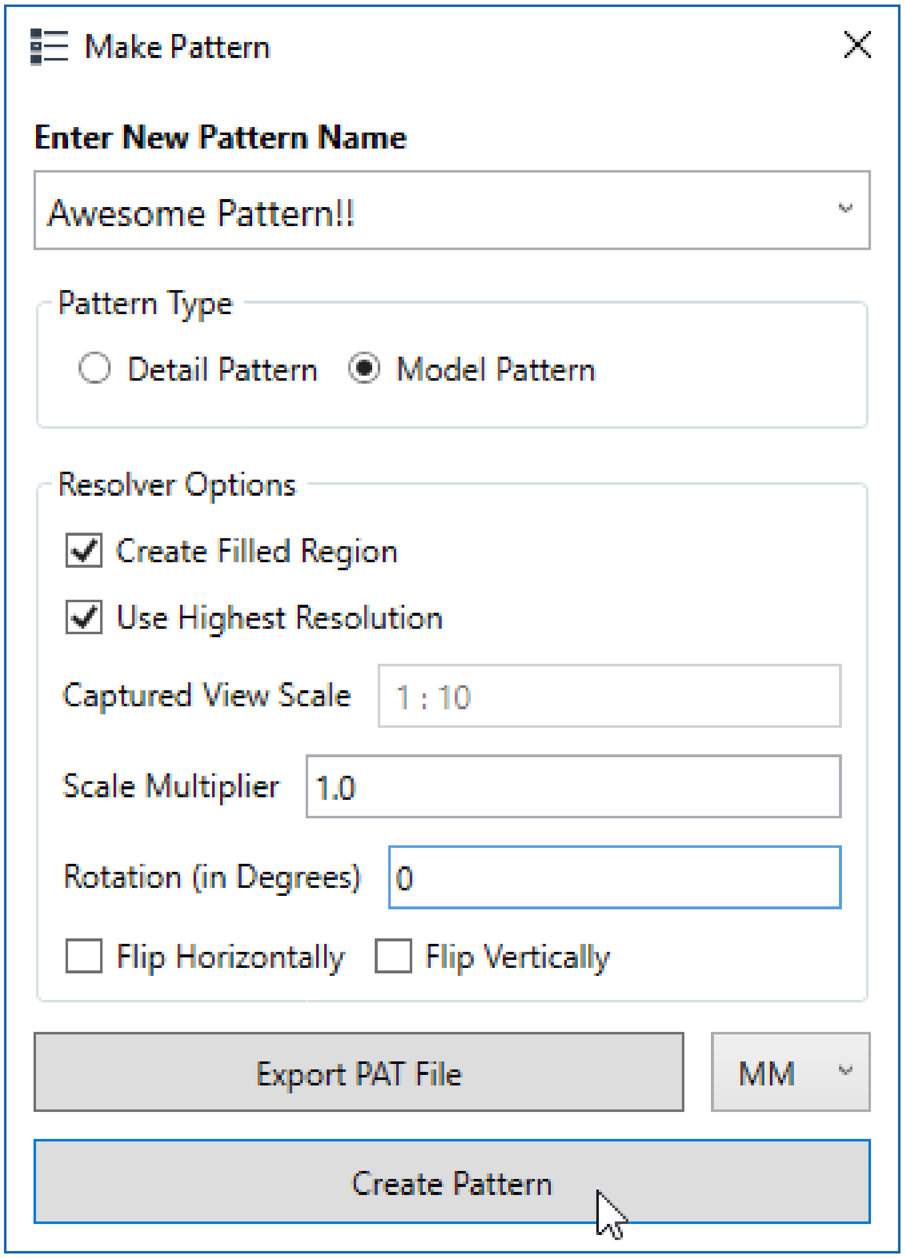
Oh my god. I don't even want to know how much time I've lost making patterns manually by using a .pat text file. This tool works even better than in my wildest dreams. To get started, you should create a **Drafting View**. Draft your pattern using detail lines. Make sure the whole pattern fits inside a rectangle, although you don't have to draw an actual rectangle. Select the lines and use the **Make Pattern** tool in the pyRevit tab.





SELECT ALL DETAIL LINES, CLICK ON "MAKE PATTERN"

You can activate some options like flip, scale, rotation and Create Filled Region. Enter a pattern name and select either Detail or Model pattern. When you are ready, click on Create Pattern.



Make Pattern

Enter New Pattern Name

Awesome Pattern!!

Pattern Type

○ Detail Pattern ● Model Pattern

Resolver Options

☑ Create Filled Region

☑ Use Highest Resolution

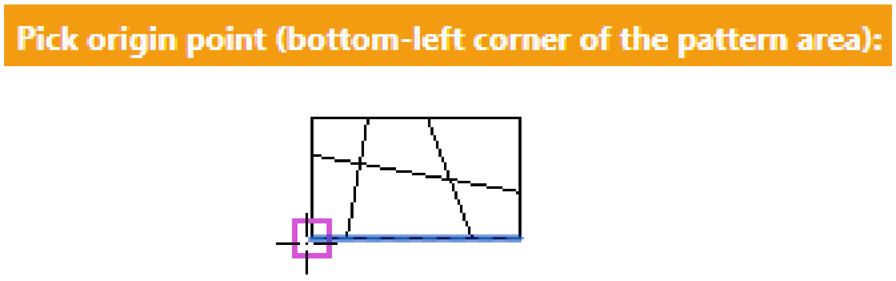
Captured View Scale: 1 : 10

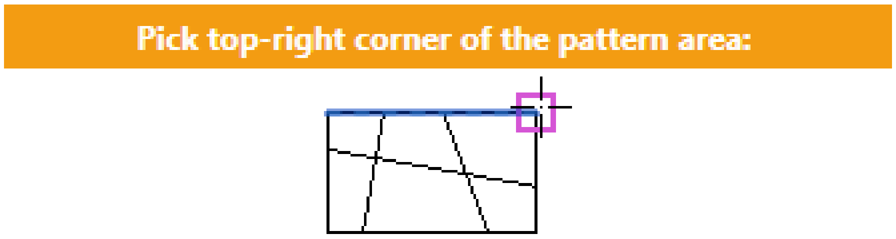
Scale Multiplier: 1.0

Rotation (in Degrees): 0

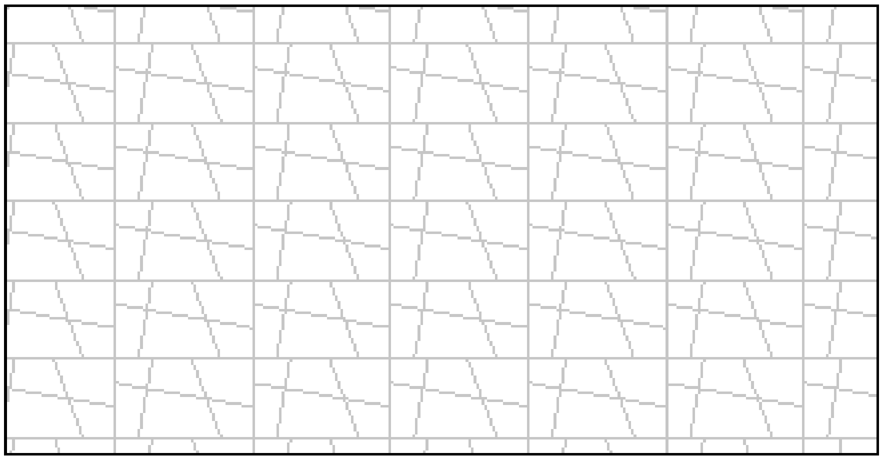
☐ Flip Horizontally ☐ Flip Vertically

Now, you have to click on two points: the origin (bottom-left corner) and then the top-right corner. Use snaps to guide you.





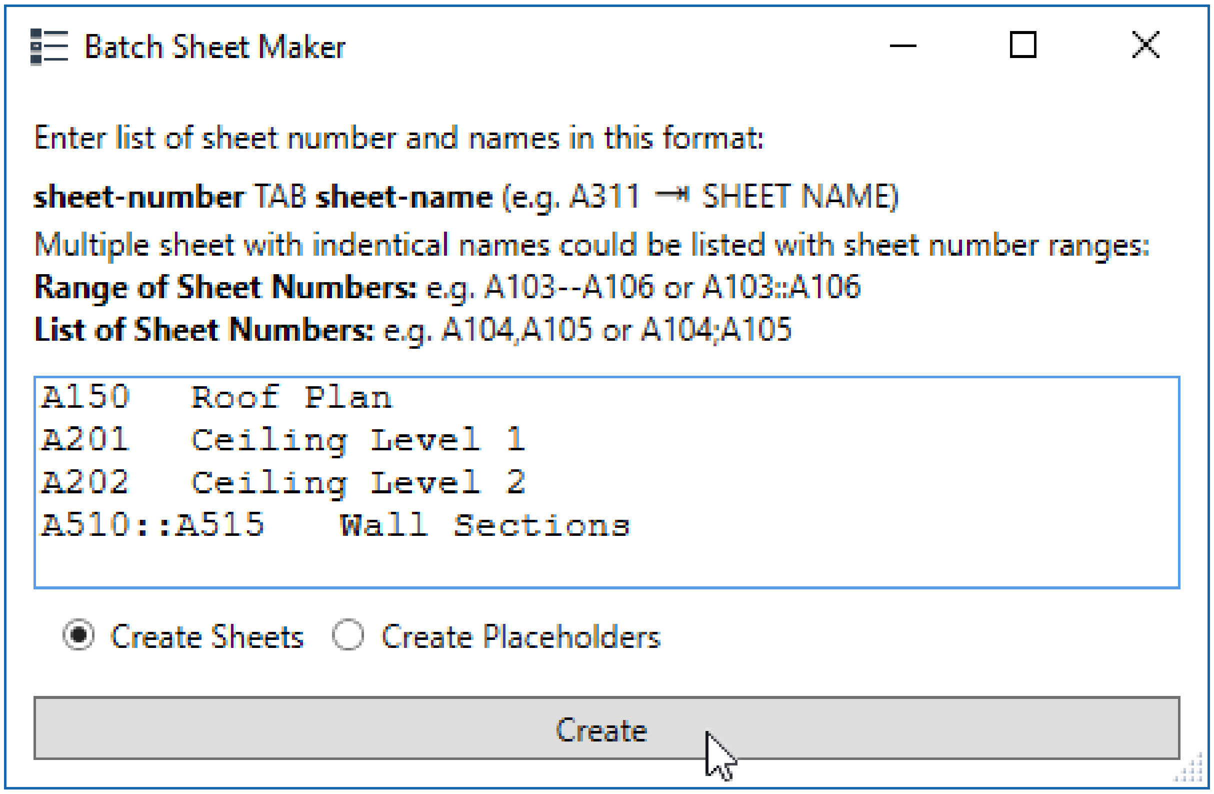
The pattern creation is now complete! It should appear in the list of available patterns. It can be used in a filled region.



2- BATCH SHEET MAKER

Creating multiple sheets in Revit has always been slow and boring. Some Dynamo scripts can help, but they can't compete with the efficiency of pyRevit **Batch Sheet Maker** tool. Click on the **Sheets** icon in the pyRevit tab, then select Batch Sheet Maker. You will get the following dialog box.

Enter the sheet number, then press tab and enter the sheet name. If multiple sheets have the same name, you can create a range of sheets by using the **::** symbol as below for the wall sections (A510::A515).


Batch Sheet Maker

Enter list of sheet number and names in this format:

sheet-number TAB **sheet-name** (e.g. A311 ⇥ SHEET NAME)

Multiple sheet with indentical names could be listed with sheet number ranges:

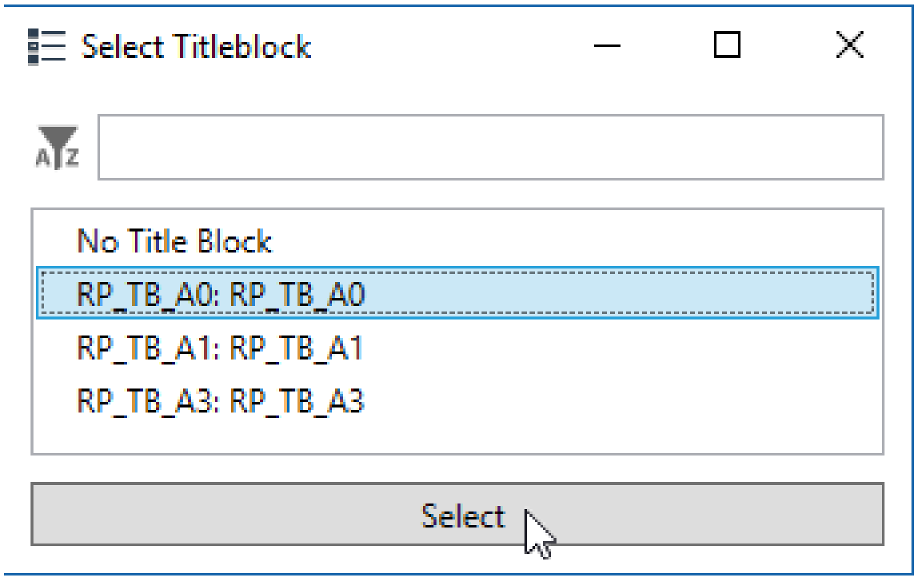
Range of Sheet Numbers: e.g. A103--A106 or A103::A106

List of Sheet Numbers: e.g. A104,A105 or A104;A105

```
A150    Roof Plan
A201    Ceiling Level 1
A202    Ceiling Level 2
A510::A515    Wall Sections
```

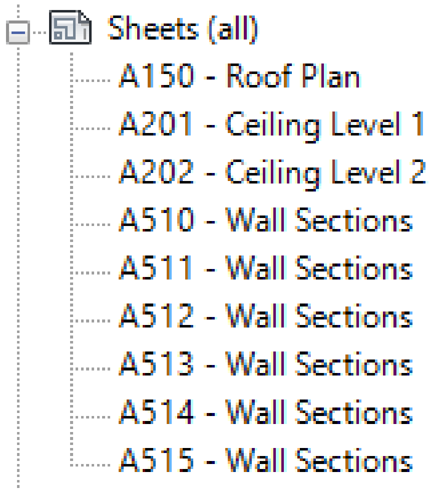
[✓ Create Sheets] Create Placeholders

You will then be asked to pick a Title Block. Pick one and click on OK.


Select Titleblock

No Title Block
RP_TB_A0: RP_TB_A0
RP_TB_A1: RP_TB_A1
RP_TB_A3: RP_TB_A3

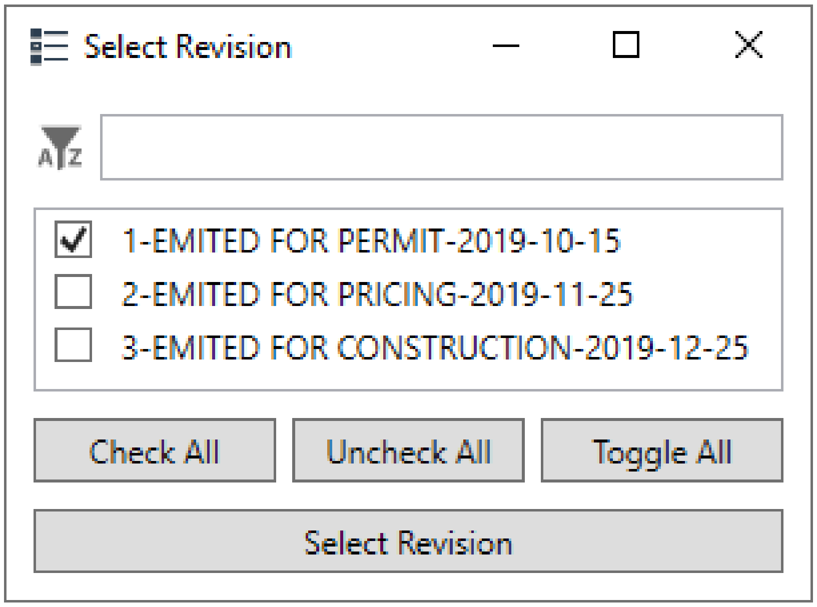
Have a look at the sheets of the projects. They have all been created! Another boring task automated.


Sheets (all)
A150 - Roof Plan
A201 - Ceiling Level 1
A202 - Ceiling Level 2
A510 - Wall Sections
A511 - Wall Sections
A512 - Wall Sections
A513 - Wall Sections
A514 - Wall Sections
A515 - Wall Sections

3- SET REVISIONS ON SHEETS

In Revit, revisions are indicated on a sheet once a revision cloud is added to a view or when you manually activate the revision on a sheet. The problem is that you can't activate a revision on multiple sheets at once. If you have a huge set of documents, it can be really long and boring to manually set the revision to all sheets.

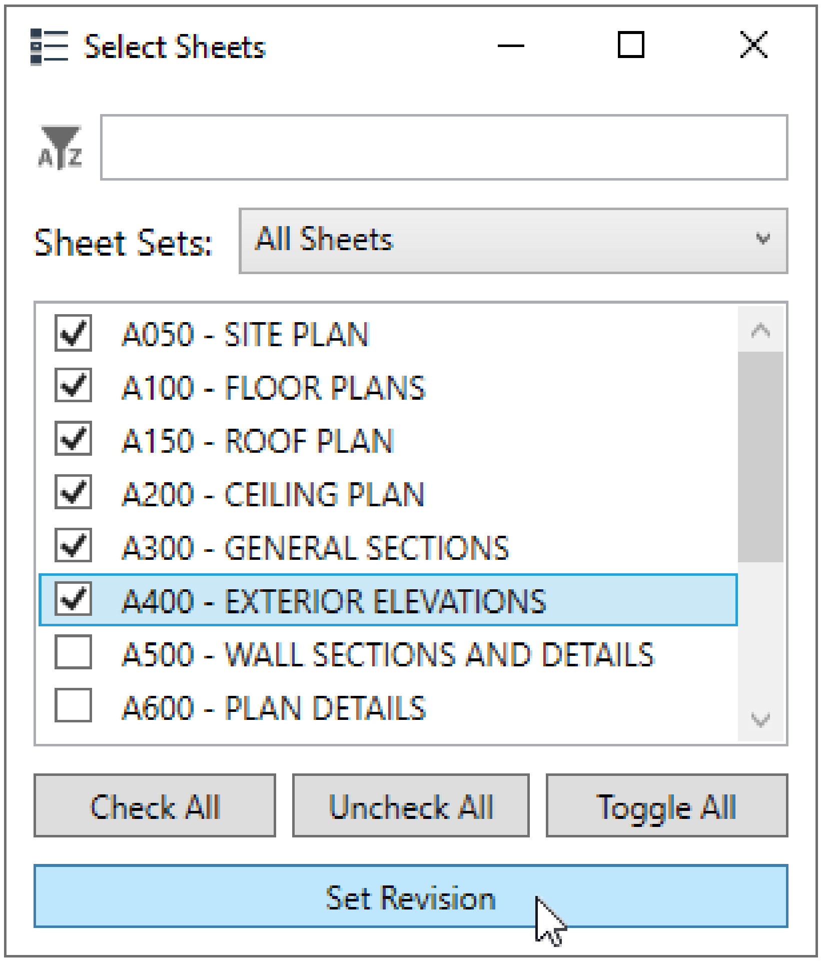
pyRevit has a fix for that. In the Revision menu, you will find the **Set Revisions On Sheet** menu. Select the revision you want to add.



[✓] 1-EMITED FOR PERMIT-2019-10-15
[ ] 2-EMITED FOR PRICING-2019-11-25
[ ] 3-EMITED FOR CONSTRUCTION-2019-12-25

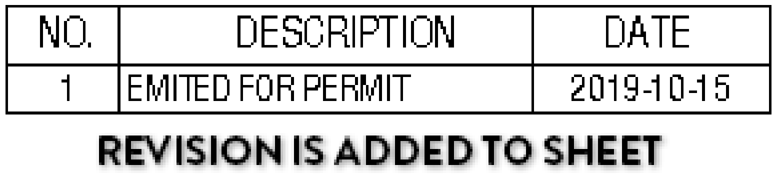
SELECT REVISION

Then, you will have to select the sheets to which you want to add the specific revision.



- [x] A050 - SITE PLAN
- [x] A100 - FLOOR PLANS
- [x] A150 - ROOF PLAN
- [x] A200 - CEILING PLAN
- [x] A300 - GENERAL SECTIONS
- [x] A400 - EXTERIOR ELEVATIONS
- [ ] A500 - WALL SECTIONS AND DETAILS
- [ ] A600 - PLAN DETAILS

If you look at the revision schedule inside one of the selected sheets, you will see the selected revision has been added.



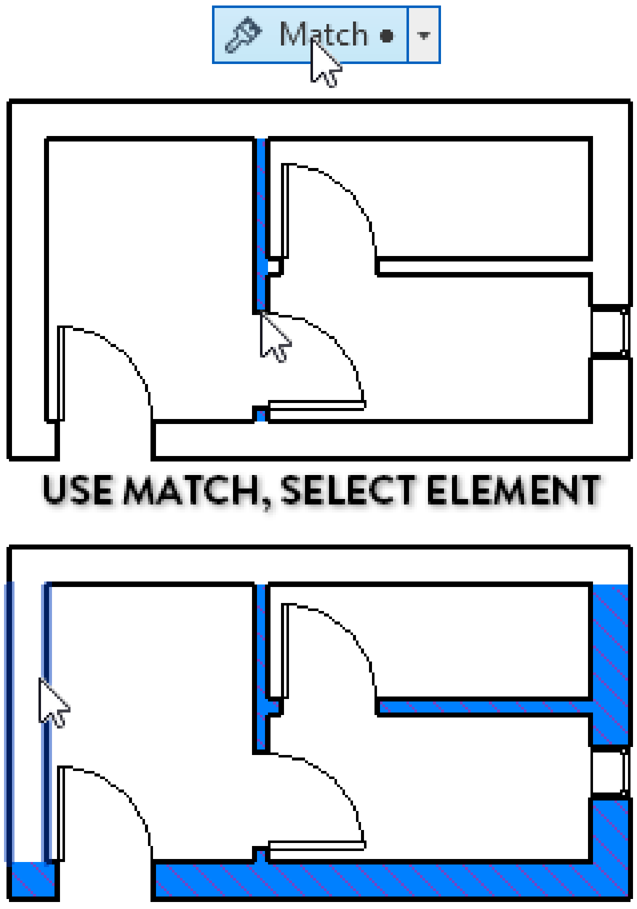
NO.	DESCRIPTION	DATE
1	EMITED FOR PERMIT	2019-10-15

REVISION IS ADDED TO SHEET

4- MATCH•

You might be familiar with Revit **Match Type Properties** tool. It takes the type properties from an element an spreads it to another element.

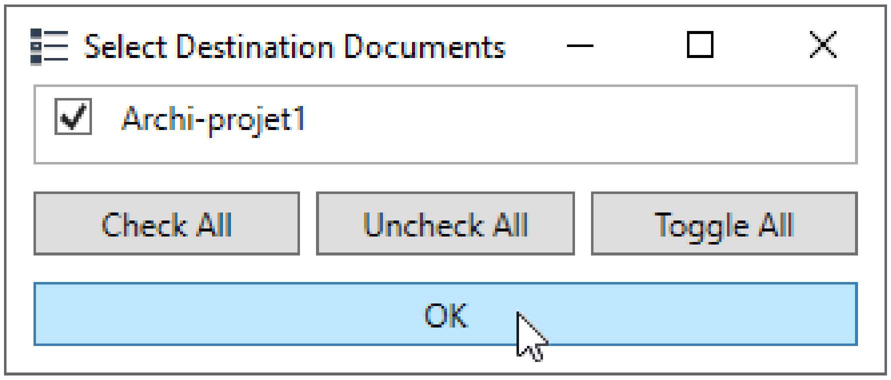
pyRevit has a seemingly similar tool simply called **Match•**. However, instead of changing the type properties, it will share the override graphics from one element to another. In the example below, we use the Match tool to spread the graphic override of a wall to other walls in the view.



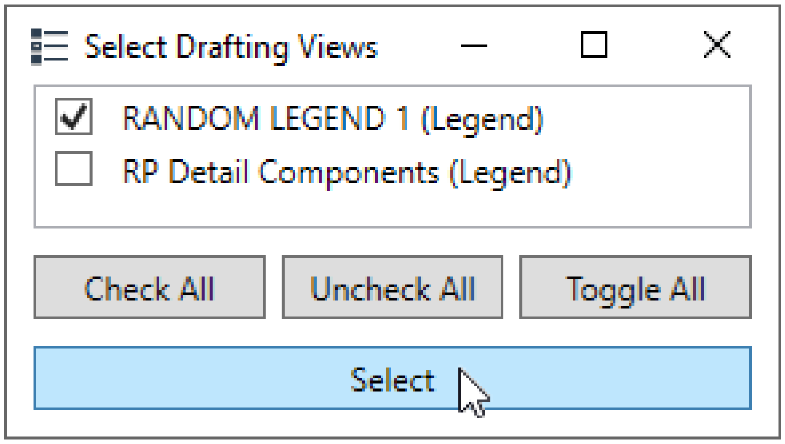
5- COPY LEGENDS TO OTHER DOCUMENTS

For some strange reason, you can't copy a legend from one Revit project to another. You have to create a new legend and manually copy/paste the content inside the schedule.

pyRevit has a tool called **Copy Legends To Other Documents.** It can be found under the **Legend** menu. Select the project you want to copy the legends to, then select the legends you want to copy.



SELECT MODEL



SELECT LEGENDS YOU WANT TO TRANSFER

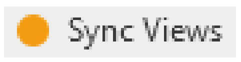
6- ● SYNC VIEWS

This is the kind of tool you didn't realize you needed so badly until you were able to try it. It is amazingly helpful. It will keep all views in the same zoomed area so you can keep working on the same sector without having to zoom and pan. For example, let's say you have a big 5 story building and you are working on the top left corner of it. When Sync Views is toggled, the same area will display on your screen when you switch to another level or to a RCP plan.

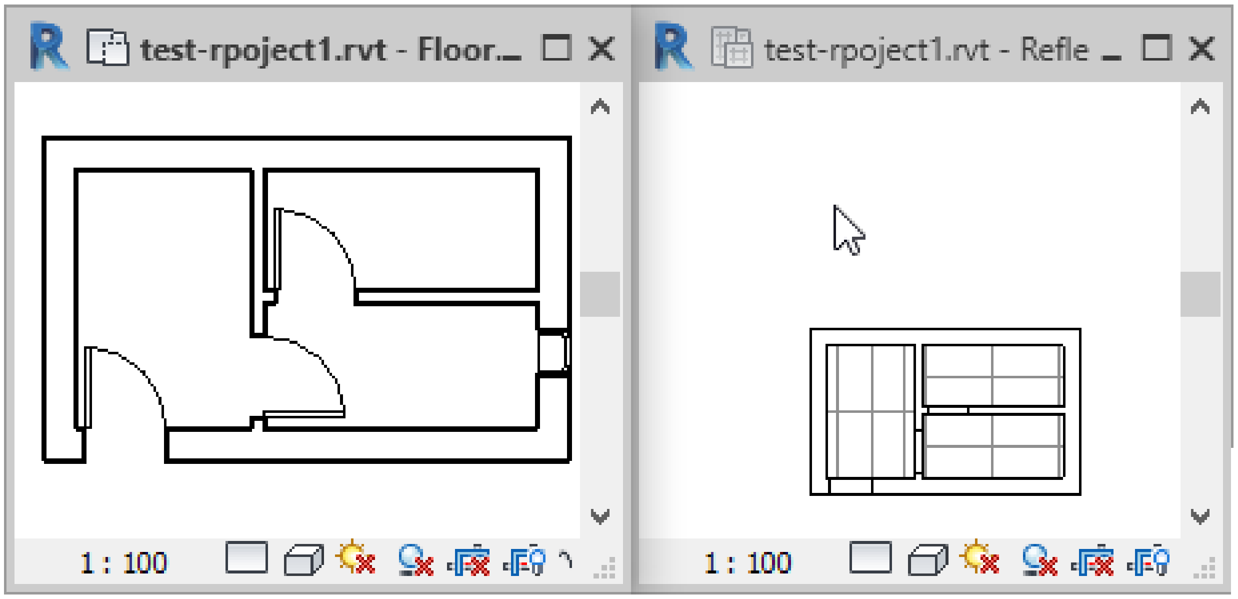
Check out the next page to see a demo.

PAMPHLETS

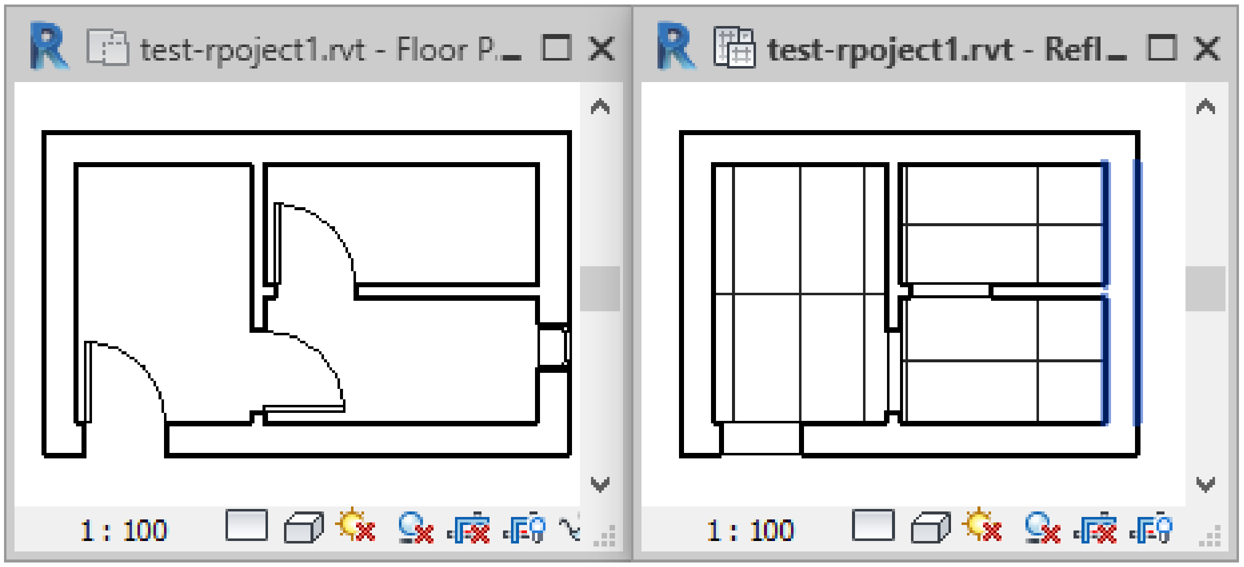
ISSUE #15 / WINTER 2020 / pyREVIT



ACTIVATE SYNC VIEWS



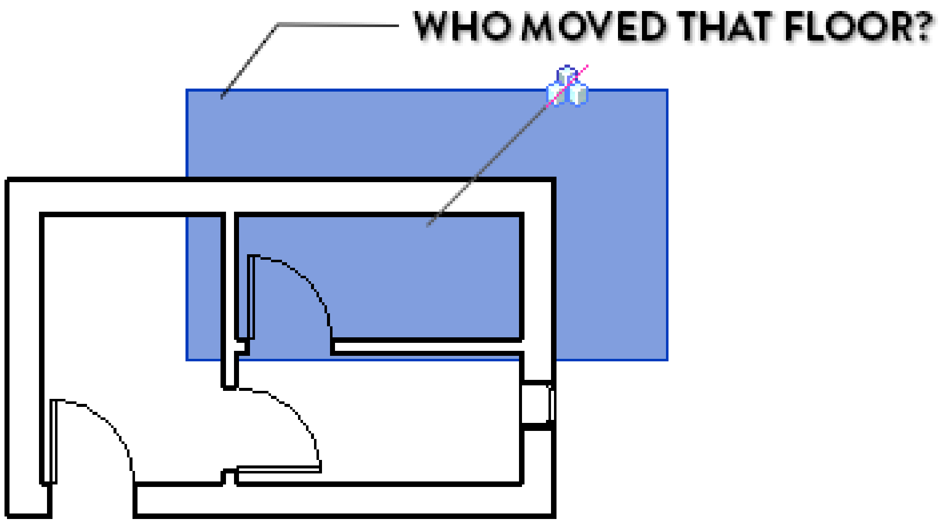
CLICK ON VIEW TO MATCH ZOOM EXTENT

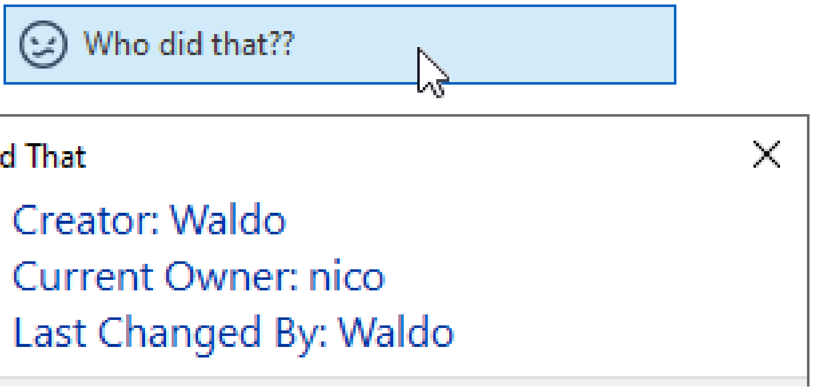


ALL PLAN AND RCP VIEWS WILL SHARE SAME EXTENTS

7- WHO DID THAT??

This is the sneakiest tool of the bunch. Did you ever work on a project with multiple users only to find an important architecture element floating in the middle of nowhere? And no one admits their mistake? The **Who Did That** tool allows you to see the last person that performed a specific task. In the example below, we want to know who moved a floor. Select the floor and pick the **Who did that??** tool on the **Teams** dropdown menu. A dialog box will display the **Creator**, **Owner** and **Last Change By** user names for the element. Then, you can have an intervention with Waldo.



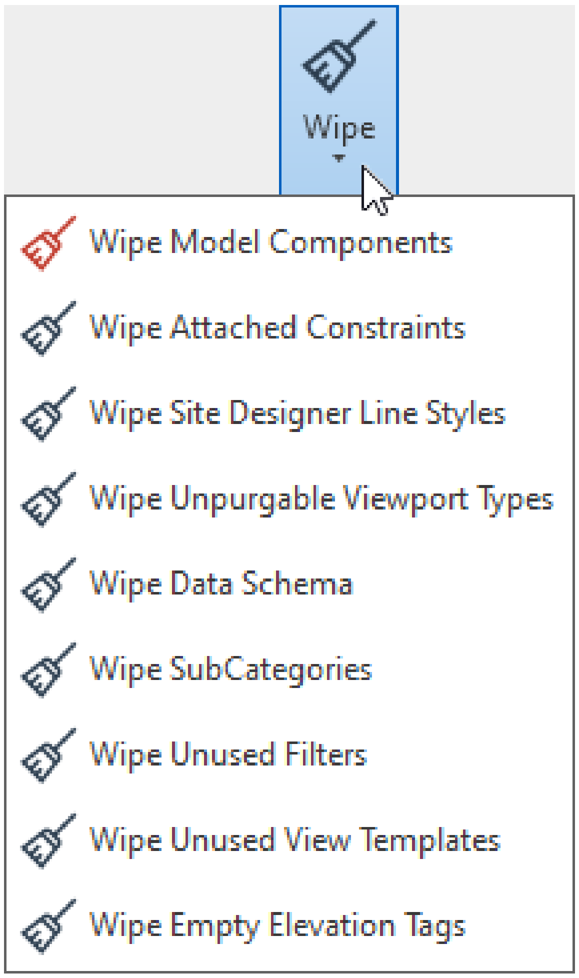


Who Did That

Creator: Waldo
Current Owner: nico
Last Changed By: Waldo

8- WIPE

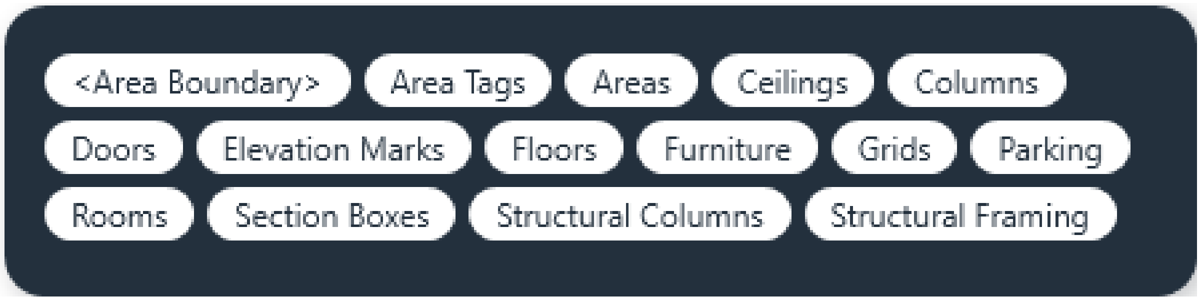
Revit has a tool called **Purge Unused** that can help you remove unused items in your models. However, this tool is often not enough if you want a deep clean. There are always a few remaining elements that can't be removed. pyRevit **Wipe** tool allows you to go much deeper. You can see below all the categories you can clean. The "Wipe Model Components" allows you to mass delete elements like views, sheets, walls, etc. That might be useful for a cleanup before sending a Revit model externally.


Wipe
Wipe Model Components
Wipe Attached Constraints
Wipe Site Designer Line Styles
Wipe Unpurgable Viewport Types
Wipe Data Schema
Wipe SubCategories
Wipe Unused Filters
Wipe Unused View Templates
Wipe Empty Elevation Tags

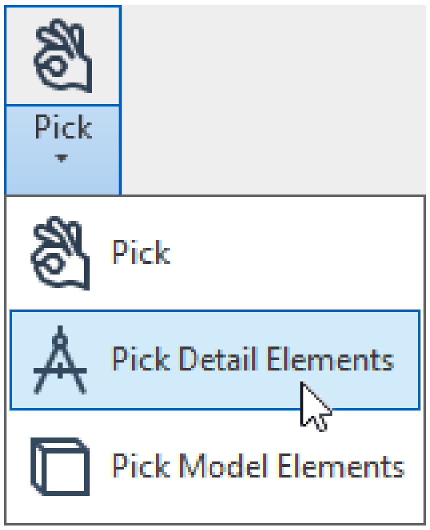
9- PICK

The most common way to select a category of elements is to use the default **Filter** tool. The problem is that it can be a little annoying to constantly check and uncheck categories.

pyRevit's **Pick** tool allows you to pick a specific category **before** selecting.



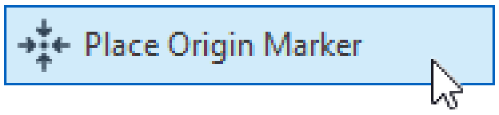
By expanding the **Pick** dropdown menu, you will also find tools to either pick **Detail** elements or **Model** elements.

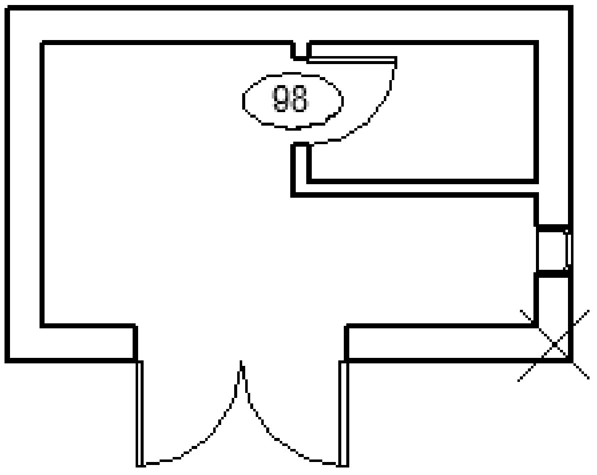


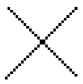
10- PLACE ORIGIN MARKER

If you have read previous pamphlets, you know how messed up the coordinate system in Revit can be. In addition to the Project Base Point and Survey Point, there is also an invisible point called the Internal Origin. This point is used by default when importing and exporting files to other software. ***New:** if you are using Revit 2020.2 or more recent, the internal origin now has a default marker. This tool isn't helpful in that case.*

If you don't know where the location of the internal origin is, try to use the **Place Origin Marke**r tool in pyRevit. It is located under the **Edit** dropdown menu. It will create a diagonal cross of detail lines at the location of the internal origin. You should probably indicate the spot with reference planes and delete the lines after using the tool.

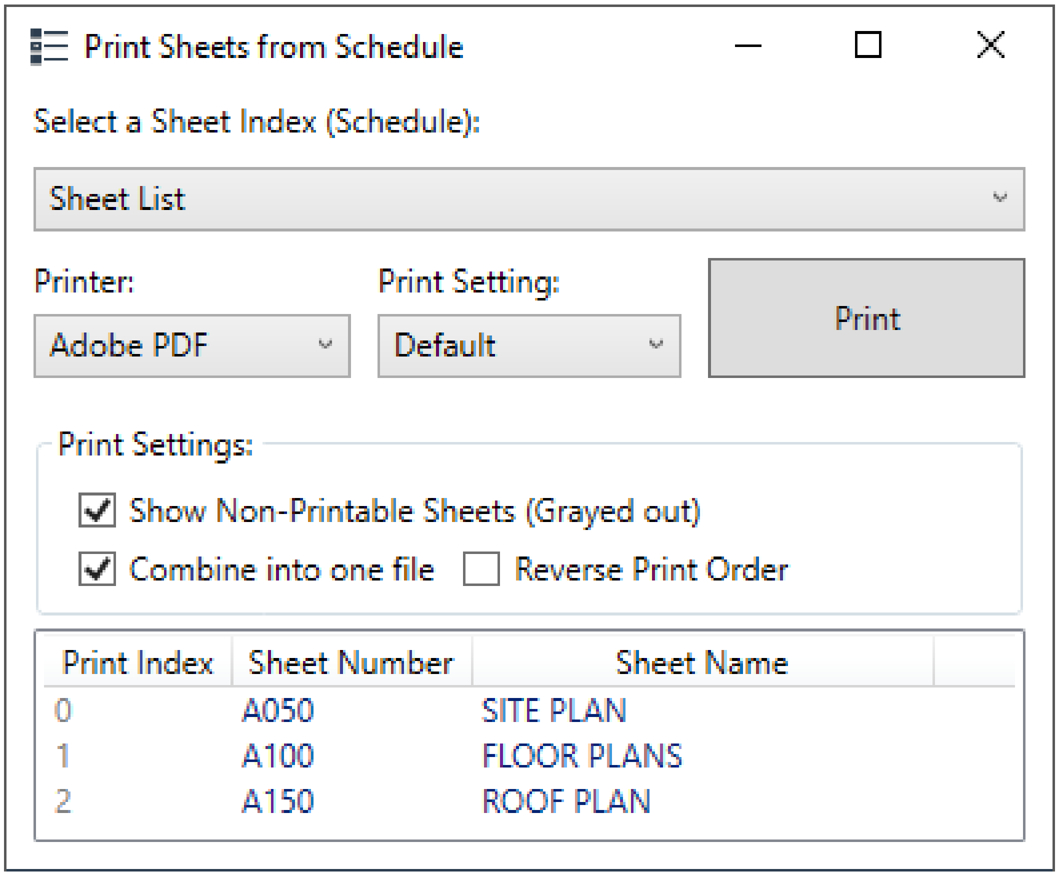





CROSS SYMBOL DETAIL LINES CREATED AT LOCATION OF INTERNAL ORIGIN

11- PRINT ORDERED SHEET INDEX

Revit has a special type of schedule called Sheet List. It allows you to create a list of sheets, usually to be placed on the front page on a set of sheets. This list is usually filtered to only show sheets from a specific revision. pyRevit has a tool called **Print Ordered Sheet Index** that allows you to print sheets from a specific sheet list. This is useful to make sure you are printing the correct sheets from a specific list or revision.


Print Sheets from Schedule

Select a Sheet Index (Schedule): Sheet List

Printer: Adobe PDF

Print Setting: Default

Print Settings:

- [✓] Show Non-Printable Sheets (Grayed out)
- [✓] Combine into one file
- [ ] Reverse Print Order

Print Index	Sheet Number	Sheet Name
0	A050	SITE PLAN
1	A100	FLOOR PLANS
2	A150	ROOF PLAN

CREATE YOUR OWN pyREVIT TOOL

This part of the pamphlet offers a brief overview of the tools included in pyRevit that allows you to create your own Revit tool using the Python programming language.

Python is known as one of the easiest programming languages to learn, especially when compared to language such as C# and VB.net. Using this feature is a little more complex than using Dynamo, but a little simpler than using the default API and other programming language.

Here is a great free online course to learn Python:

Programming for Everybody (Getting Started with Python)
University of Michigan via Coursera

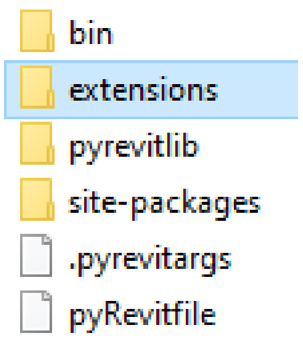
This course aims to teach everyone the basics of programming computers using Python. We cover the basics of how one constructs a program from a series of simple instructions in Python.

Link: https://www.classcentral.com/course/python-4319

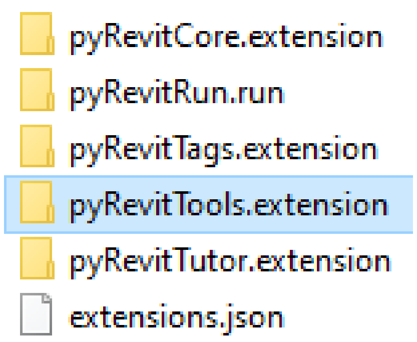
Another quick word: Thanks to Dan Boghean from OZ Architecture for his great course about pyRevit at the BILT 2019 conference in Seattle. A lot of the content in the next pagesis inspired by the documents he provided.

pyREVIT FOLDER STRUCTURE

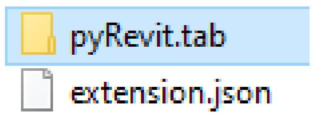
What is great about pyRevit's tool creating capabilities is how easy it is to create your own tabs and dropdown menus. Have a look at the pyRevit folder, usually located in AppData/Roaming:


bin
extensions
pyrevitlib
site-packages
.pyrevitargs
pyRevitfile

pyRevit master folder

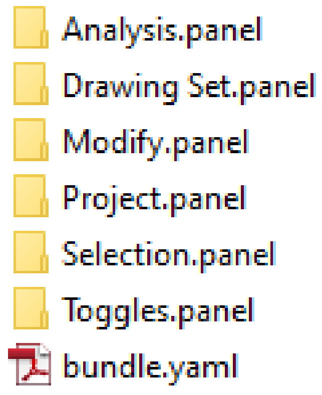

pyRevitCore.extension
pyRevitRun.run
pyRevitTags.extension
pyRevitTools.extension
pyRevitTutor.extension
extensions.json

extensions folder

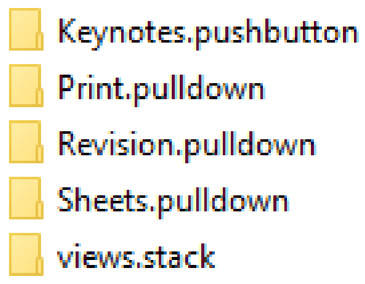

pyRevit.tab
extension.json

pyRevitTools.extension folder

The pyRevit.tab folder then contains folders for all the panels. In the image below, you can see all the subfolders contained in the path of the **pyRevit.tab** folder, then inside the Drawing Set.panel folder. See next page to understand what all these terms mean.

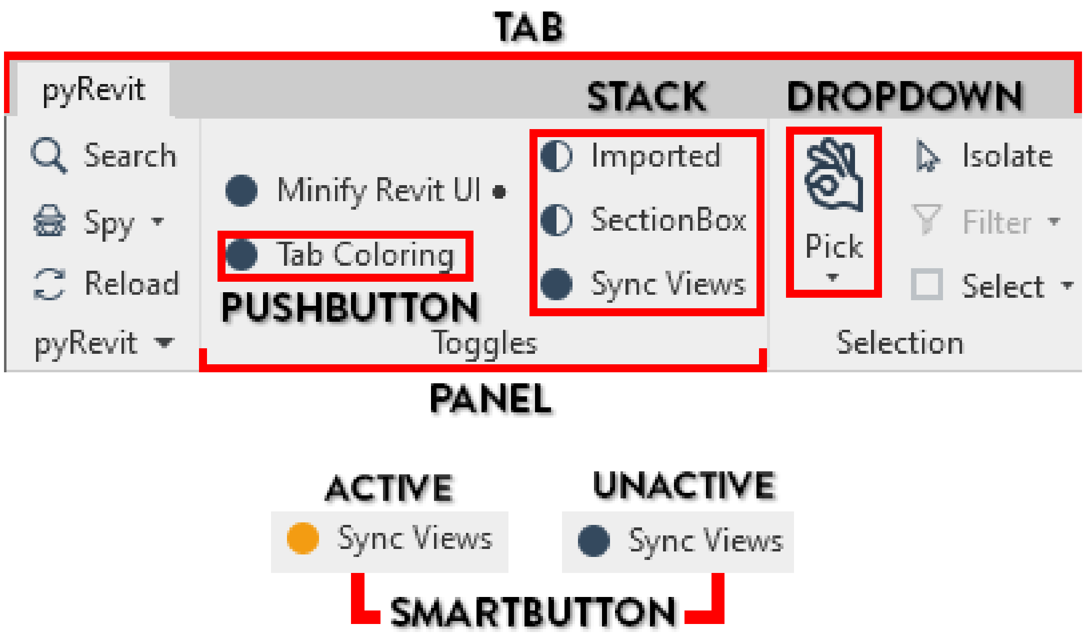

Analysis.panel
Drawing Set.panel
Modify.panel
Project.panel
Selection.panel
Toggles.panel
bundle.yaml

pyRevit.tab folder

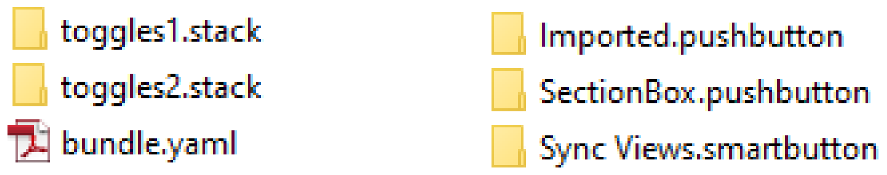

Keynotes.pushbutton
Print.pulldown
Revision.pulldown
Sheets.pulldown
views.stack

Drawing Set.panel folder

Once you are inside a panel folder, your subfolders should have .pushbutton, .stack or .dropdown at the end of their name. The image below explains all the terminology.



For example, here is the folder for the Toggles panel. It contains 2 stacks. The second stack contains 2 pushbuttons and 1 smartbutton.


toggles1.stack
toggles2.stack
bundle.yaml

Toggles.panel folder

Imported.pushbutton
SectionBox.pushbutton
Sync Views.smartbutton

toggles2.stack folder

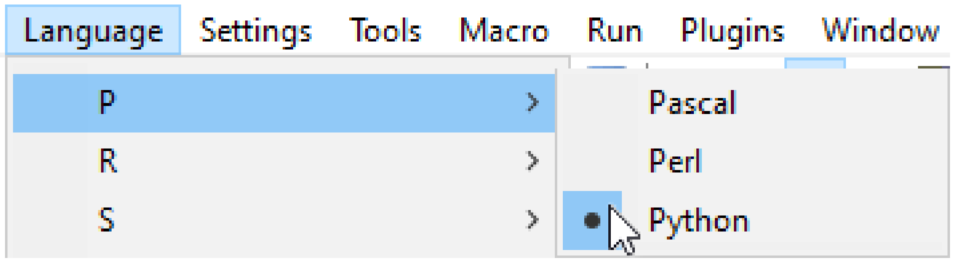
When creating your own extension set using pyRevit, you'll have to learn how to navigate this folder structure to organize your tools.

PREPARE TO CODE

Before you start coding, you should get prepared. First, you should download **Notepad++**. It is a glorified notepad designed specifically to write code. It is available for free at this link:

https://notepad-plus-plus.org/downloads/

In the language menu of the software, you can select Python. Colors will be added to your script, helping you visualize.



Then, you will need to use the **Revit API**. Basically, it contains the name of all the tools and commands used in Revit. In your script, you will have to import and refer to these commands to execute actions. Check out the API docs at this link:

https://apidocs.co/apps/revit/2020

You are locked and loaded! Let's get ready to code.

START CODING

We'll now create the most basic add-in. You will create a dialog box that says "Hello World!". The first thing you need to do is to import tools from the Revit interface. In this case, we will import the specific "TaskDialog" tool. We use the following line:

```
from Autodesk.Revit.UI import TaskDialog
```

You can then use the TaskDialog tool to display whatever you want. Use the following line:

```
TaskDialog.Show("Hello World", "Thanks a lot!")
```

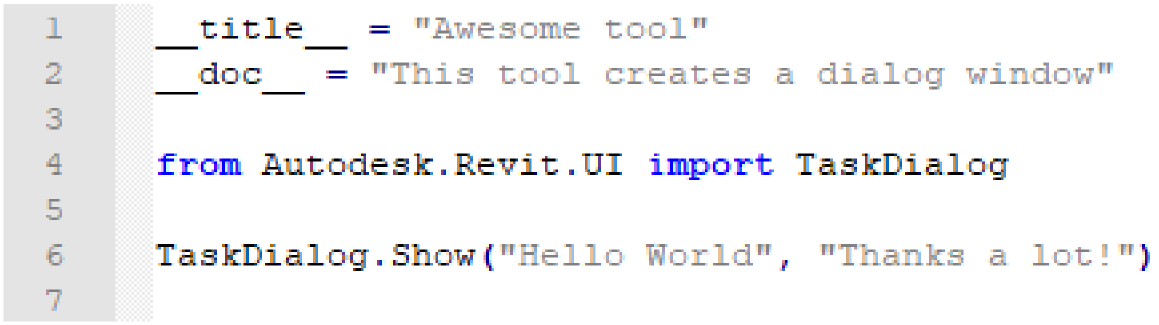
Now, you can add elements in your code that are specific to pyRevit. You need to fill in the **title** and **doc** information. **Title** is the name of the tool that will be displayed in the ribbon, **doc** is the description when your cursor hovers over the icon. Write it down at the top of your code just like this:

```
__title__ = "Enter Title Here"
__doc__ = "Enter description when cursor hovers on icon here"
```

In the tool we are currently building, it would look like this:

```
__title__ = "Awesome Tool"
__doc__ = "This tool creates a dialog window"
```

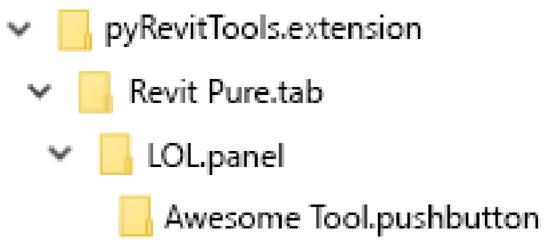
Here is what the entire text for your tool should look like:



```
__title__ = "Awesome tool"
__doc__ = "This tool creates a dialog window"

from Autodesk.Revit.UI import TaskDialog

TaskDialog.Show("Hello World", "Thanks a lot!")

```

CREATE FOLDERS

We have the script ready. We now need to create the folders so the script appears in Revit. We will create a tab called **Revit Pure**, with a panel called **LOL** and the smartbutton will be called **Awesome Tool**. This is what your folder structure should look like.



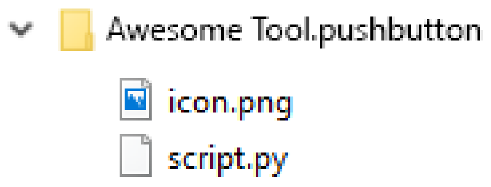
```
pyRevitTools.extension
  Revit Pure.tab
    LOL.panel
      Awesome Tool.pushbutton
```

You can also create an extension folder and place it on a local server. In pyRevit settings, you can then add a link to this folder.

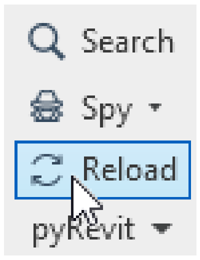
ADD ICON AND SCRIPT

Each button folder contains a script file and an icon file. The script file should always be called **script.py**. The icon should always be called **icon.png**. Try to use an icon image of 96px or less.

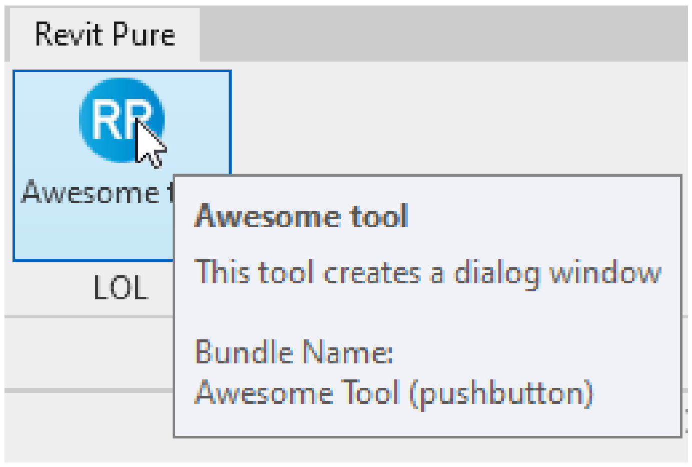
Here is what the **Awesome Tool.pushbutton** folder should contain:



Make sure to place the complete folder structure inside the pyRevitTools.extension folder, located in your AppData/Roaming/pyRevitMaster folder. Then, hit the refresh button in the pyRevit tab.



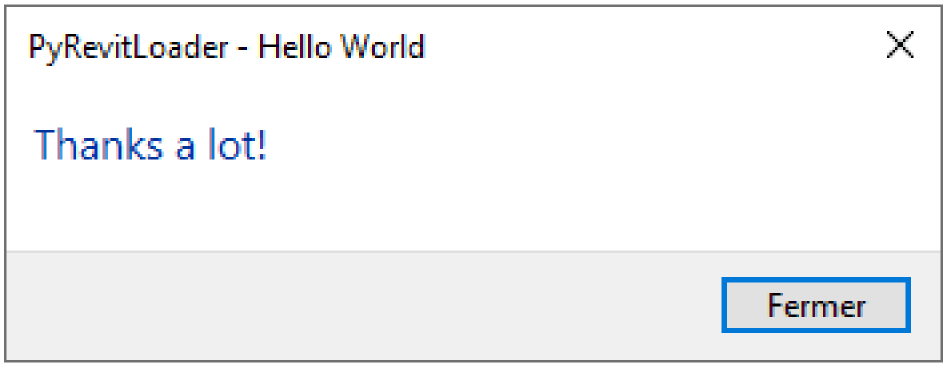
The Revit Pure tab, the LOL panel and the Awesome Tool pushbutton should appear as in the image below. You can also see the description when you hover your cursor above the icon.



Awesome tool
This tool creates a dialog window

Bundle Name:
Awesome Tool (pushbutton)

When you click on the tool, this dialog window should appear:



PyRevitLoader - Hello World

Thanks a lot!

Congratulations, you just created your first add-in in Revit! Of course this tool is completely useless, but at least you know the basics.

CREATE A REAL TOOL

Now that we've created a useless tool, let's create one that actually does something. We'll create a tool that displays the Type Name of selected elements.

First, let's pre-load the database (DB), the user interface (UI) and the selection option (UI Selection). The * symbol imports all the tools.

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

Now, you need to define the document and user interface of the document.

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

Now, you need to define how elements will be selected. Create a variable called **selectionIds**. Then, type in the User Interface defined earlier (**uidoc**). In this example, we use a selection tool called PickObjects. By researching the Revit API docs, we find out this is what you need to type:

```
selectionIds = uidoc.Selection.PickObjects(ObjectType.Element)
```

Then, you need to create a "For" loop to cycle through all selected elements. This is how you create the loop:

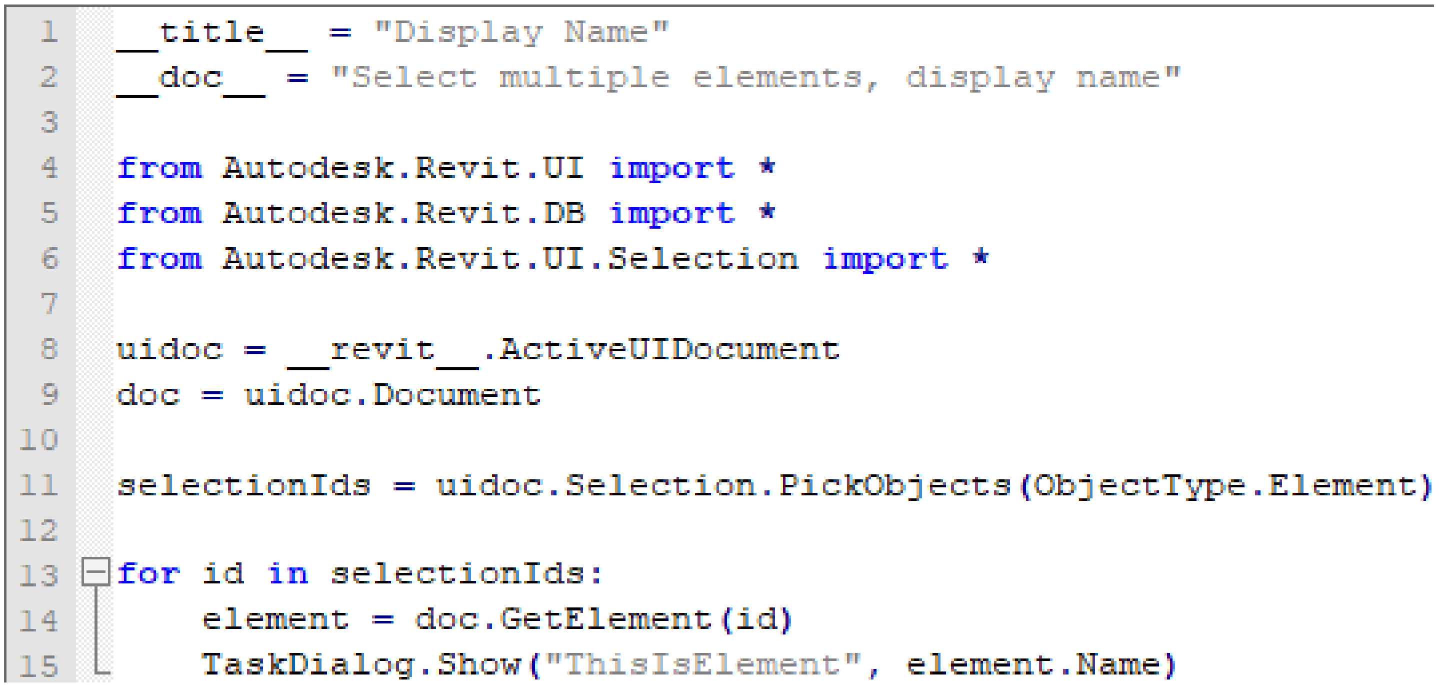
```
for id in selectionIds:
        element = doc.getElement(id)
```

The final step is to create a Dialog window that displays the type name of each element in the loop:

```
TaskDialog.Show("Element-Type-Name", element.Name)
```

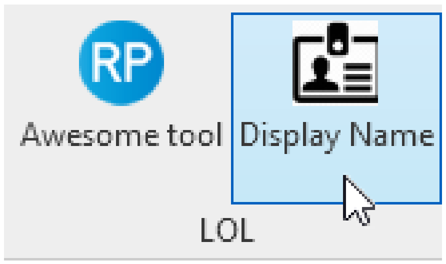
Be careful: this technique will display a different dialog window for each individual element you have selected.

This is the final code:

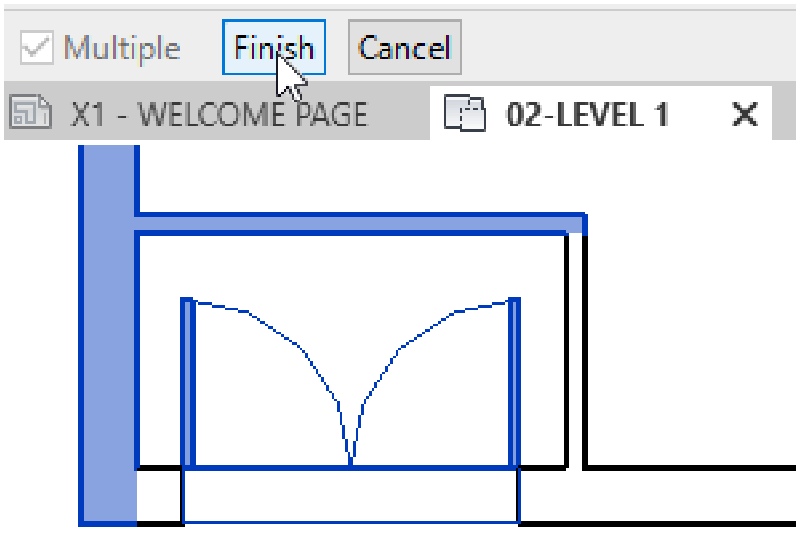


```python
__title__ = "Display Name"
__doc__ = "Select multiple elements, display name"

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

uidoc = __revit__.ActiveUIDocument
doc = uidoc.Document

selectionIds = uidoc.Selection.PickObjects(ObjectType.Element)

for id in selectionIds:
    element = doc.GetElement(id)
    TaskDialog.Show("ThisIsElement", element.Name)
```

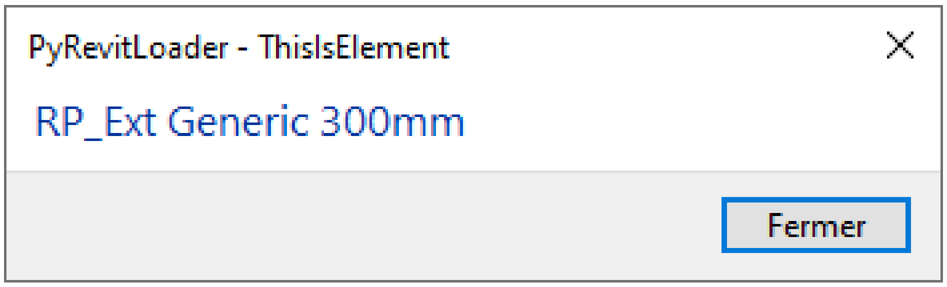
Create a new folder in the LOL panel. Place the script and icon files. Reload all the scripts inside Revit. The tool should appear in the tab. Let's try it out to make sure it works properly



Once you click on the tool, you will have to select elements. You can select as many elements as you want. Click on **Finish** to complete the selection process. As you can see, the dialog box will display the type name of the element.



SELECT ELEMENTS, CLICK ON FINISH


PyRevitLoader - ThisIsElement

RP_Ext Generic 300mm

DIALOG BOX DISPLAYS ELEMENT TYPE NAME

Of course, this tool is not really helpful. But by learning Python and by exploring the Revit API, you can easily expand this feature and build a helpful plugin. Good luck!

THANKS FOR READING!

As always, send your thoughts to nick@revitpure.com. I read and answer all emails. Let me know if you loved or hated this pamphlet.

REVIT PURE PACKAGES

Did you like the simple, efficient style of this pamphlet? That means you will love our learning packages! **BASICS** will help you learn all the essential tools of Revit. Check it out at: revitpure.com/basics. **DESIGN** will teach you how to create beautiful presentation documents using Revit. Check it out at revitpure.com/design.

Both packages contain an eBook and video tutorials, but also bonus content such as templates and Revit families. Use code **learn** to get 15% off on any package.