**BILT:** Session 1.2

LAB: TAKE CONTROL OVER REVIT BY CREATING TOOLS WITH PYREVIT

Jean-Marc Couffin, BIM One Inc.

Une image contenant texte

Description générée automatiquement

Class Description

This presentation will run you through the process and the tools for you to create a custom toolbar with a set of custom tools thanks to pyRevit, some python coding or a dynamo script. The idea is to empower advanced users of Revit and help them structure a toolbox to improve their productivity and get rid of the repetitive or tedious tasks.

* First part will cover the structure of pyRevit Framework, its capabilities, and its set of tools
* Second part will be about creating a simple toolbar
* Third part about the creation of a few scripts to populate our toolbar
* Finally, to show a way to distribute the toolbar across an organization

About the Speaker

**Jean-Marc Couffin**

Architect DPLG, Senior BIM specialist

Une image contenant personne, posant, sombre

Description générée automatiquementTrained architect both in France and in the USA, Jean-Marc Couffin worked in Singapore, Vietnam, France, Czech Republic before working in Canada for Provencher Roy Architects as a BIM manager supporting teams in their efforts to create above standards building. Moving back in Europe in Czech Republic, he recently joined BIM One as a consultant to pursue his interests in BIM problem solving and automatization and explore all dimensions of BIM in relation to the construction world. His main area of focus for the past years has been developing and implementing company BIM standards and methods. He dedicates himself to build co-workers efficiency and is always on the look for innovative technologies that can improve design practice. Jean-Marc’s experience includes creating and managing complex BIM models and projects for the Agence Métropolitaine de Transport, the Place des Arts, the Canadian Space Agency, TPSGC and many major clients.

Table des matières

[00\_Prerequisites 3](#_Toc105667866)

[Yourself 🧠 3](#_Toc105667867)

[Basics 3](#_Toc105667868)

[Not Necessarly Basic 3](#_Toc105667869)

[Account 4](#_Toc105667870)

[Tool to investigate the Revit Document 4](#_Toc105667871)

[01\_Getting to know the pyRevit Framework 5](#_Toc105667872)

[pyRevit is a framework 5](#_Toc105667873)

[pyRevit is a set of (growing) tools 5](#_Toc105667874)

[pyRevit has a daddy 6](#_Toc105667875)

[pyRevit is a community 6](#_Toc105667876)

[pyRevit relies on the RevitAPI 7](#_Toc105667877)

[pyRevit provides you with tools 8](#_Toc105667878)

[Set of tools 8](#_Toc105667879)

[CLI 8](#_Toc105667880)

[Documentation to give you the basics and more 8](#_Toc105667881)

[Python Modules to interact with Revit 8](#_Toc105667882)

[02\_Setting up a pyRevit Extension the easy way 9](#_Toc105667883)

[-01\_*NOOB\_*Install the finished toolbar 9](#_Toc105667884)

[00\_*Advanced\_*Create a github repository (gitignore, licence, readme, ...) 9](#_Toc105667885)

[01\_ *Advanced\_*A minimal set of files and folders: 10](#_Toc105667886)

[02\_ *Advanced\_*And a file: 12](#_Toc105667887)

[03\_ *Advanced\_*Installation of the toolbar 14](#_Toc105667888)

[04\_*Everyone\_*Update Tools 15](#_Toc105667889)

[Update 15](#_Toc105667890)

[Reload 16](#_Toc105667891)

[05\_*Everyone\_*startup.py 17](#_Toc105667892)

[03\_Building some more tools 18](#_Toc105667893)

[-02\_ *Everyone\_*pyRevit modules 18](#_Toc105667894)

[-01\_ *Everyone\_*Staples 19](#_Toc105667895)

[00\_*Everyone\_.urlbutton* 21](#_Toc105667896)

[01\_ *Everyone\_.pushbutton* 22](#_Toc105667897)

[02\_ *Everyone\_.pushbutton* Dynamo flavored 23](#_Toc105667898)

[03\_ *Advanced\_.content* 24](#_Toc105667899)

[04\_ *Advanced\_*.nobutton 25](#_Toc105667900)

[05\_ *Advanced\_*.pushbutton with configuration 27](#_Toc105667901)

[04\_ *Advanced\_*Distribute to the team 29](#_Toc105667902)

[Install pyRevit, pyRevit CLI and the toolbar in one go 29](#_Toc105667903)

[outro 30](#_Toc105667904)

# 00\_Prerequisites

## Yourself 🧠

* Good mood
* 1 bit of knowledge of python or another programming language, basics like date types, …
* 1 bit of knowledge of Dynamo for Revit or the [Revit API](https://apidocs.co/apps/revit/2022/d4648875-d41a-783b-d5f4-638df39ee413.htm) (or just understanding of what is an API 🙂) going through this 👇 would be an excellent start

<https://dynamopythonprimer.gitbook.io/dynamo-python-primer/4-revit-specific-topics/introduction-to-revits-api>

## Basics

* Une image contenant texte, signe

  Description générée automatiquementat least one Revit version installed (2020-2022 will be fine)   
  <https://www.autodesk.com/products/revit/free-trial>
* pyRevit installed  
  <https://github.com/eirannejad/pyRevit/releases>

## Not Necessarly Basic

* Github desktop installed  
  <https://desktop.github.com>
* VSCode or pyCharm (or notepad++ if you like pain)  
  <https://code.visualstudio.com/download>
* Extensions to Vscode (CTRL+Shift+X in VSCode)
  + python
  + Github Pull requests
  + docs-yaml
  + yaml
  + prettier
* ⚠️ a bit trickier **VSCode settings**:
  + add it to python autocomplete extrapaths in vscode settings  
    Adding stubs file will allow you to use the auto-complete feature of VSCode with the Revit API and pyRevit references

1. Download stubs files from (and unzip):
   1. <https://github.com/gtalarico/ironpython-stubs/archive/master.zip>

or from

* 1. <https://github.com/BIMOpenGroup/RevitAPIStubs/releases/download/v1.0.0/stubs.rar>

1. File > Preferences > Settings > Python > auto-complete > ExtraPaths > Settings.json
2. Add stubs file path to the extraPaths section of the settings.json
   * Beware of the file path format with the double \\

Mine looks like this:  
{

"python.analysis.extraPaths": [

**"C:\\Gits\\RevitAPIStubs\\stubs\\common"**,

**"C:\\Gits\\RevitAPIStubs\\stubs\\revit\\2022",**

**"C:\\Users\\Jean-Marc\\AppData\\Roaming\\pyRevit-Master\\pyrevitlib"**

],

"python.autoComplete.extraPaths": [

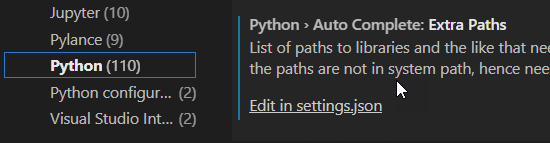
"**C:\\pyRevit\\stubs\\revit\\2023",**

**"C:\\pyRevit\\stubs\\common"**,

**"C:\\Users\\Jean-Marc\\AppData\\Roaming\\pyRevit-Master\\pyrevitlib\\pyrevit"**,

]

}



Complete explanation and links:

<https://discourse.pyrevitlabs.io/t/vscode-for-pyrevit-and-revit-api/413/5> or <https://forum.dynamobim.com/t/intellisense-step-by-step-configuration-on-visual-studio-code/27085/2?u=jean-marc> or <https://sumptuous-rhubarb-de0.notion.site/LAB-TAKE-CONTROL-OVER-REVIT-BY-CREATING-TOOLS-WITH-PYREVIT-705cd44ad90e46fa8011fe4047637ad8>

## Account

a Github account  
<https://github.com/signup>

## Tool to investigate the Revit Document

⚠️ Version specific

Revit Lookup Tool

[Releases · jeremytammik/RevitLookup](https://github.com/jeremytammik/RevitLookup/releases)

ALL THE CODE WILL BE AVAILABLE HERE <https://github.com/jmcouffin/pyRevit-BILT_NA_2022>

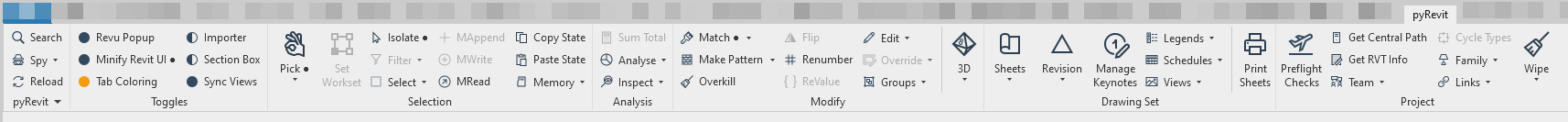
# 01\_Getting to know the pyRevit Framework

## pyRevit is a framework

pyRevit (with lowercase py) is a Rapid Application Prototyping (RAD) environment for Autodesk Revit. It helps you quickly sketch out your automation and add-on ideas, in whichever language that you are most comfortable with, inside the Revit environment and using its APIs. It also ships with an extensive set of powerful tools that showcase its capabilities as a development environment. Download and install pyRevit, launch Revit, and note the new **pyRevit** tab that includes these tools. pyRevit also ships with a handy CLI utility for customized configuration and deployment of your tools, and a telemetry server to monitor pyRevit usage across your teams.

**That means it is not only a set of tools to do stuffs in Revit but also to build your own tools**

## pyRevit is a set of (growing) tools



## pyRevit has a daddy



<https://ein.sh/>

and lots of geeky heirs…

## pyRevit is a community

*with lots of people trying to go further than Revit*

<https://discourse.pyrevitlabs.io/>Une image contenant texte

Description générée automatiquement

## pyRevit relies on the RevitAPI

The original content for the RevitAPI is here:

<https://www.autodesk.com/developer-network/platform-technologies/revit>

But a better way to navigate it is happening here:

<https://apidocs.co/>

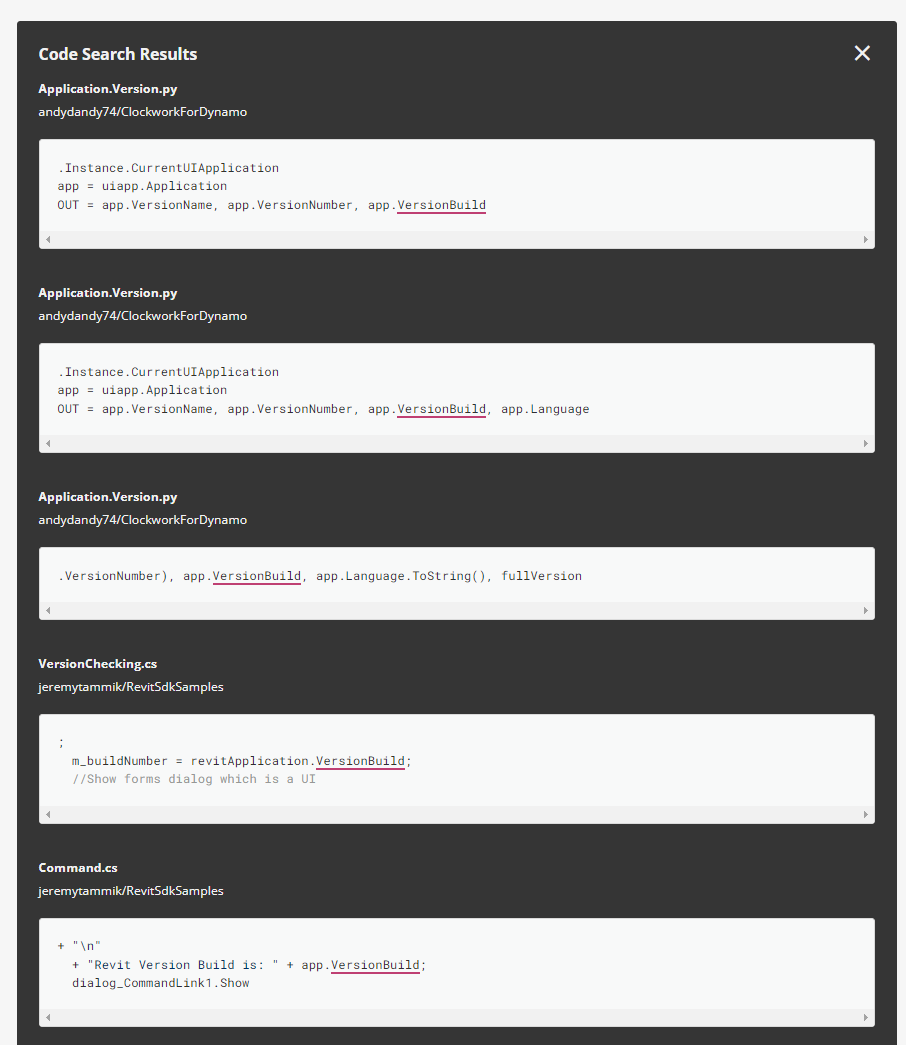
Thanks to <https://gtalarico.com/>

So if you want to know how to use the Revit API, this is the one stop with a special trick:

Une image contenant texte

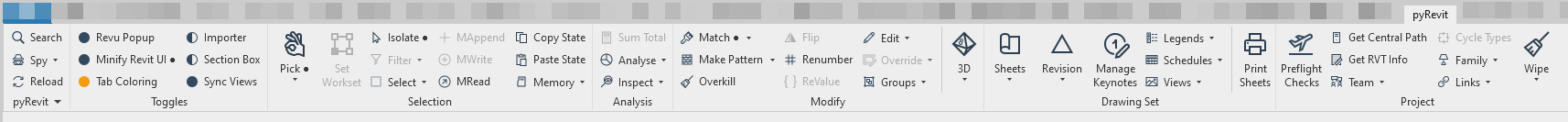
Description générée automatiquement

This button gives you access to samples of code, while not perfect as it is using a basic search method from the github API, it does provide you with samples in C# or Python to illustrate how to use a specific Revit API Method



## pyRevit provides you with tools

### Set of tools



**Check out the preflight checks -> model checker**

### CLI

Command Line Tool with the **pyrevit** handle

Une image contenant texte

Description générée automatiquement

### Documentation to give you the basics and more

Here: <https://www.notion.so/pyrevitlabs/pyRevit-bd907d6292ed4ce997c46e84b6ef67a0>

### Python Modules to interact with Revit

<https://pyrevit.readthedocs.io/en/latest/>

# 02\_Setting up a pyRevit Extension the easy way



Revit should be closed (not entirely true 😉 )

## -01\_*NOOB\_*Install the finished toolbar

* in command line (WIN + cmd), we will install our toolbar

**pyrevit extend ui pyBiltNA https**://github.com/jmcouffin/pyRevit-BILT\_NA\_2022.git --dest="C:\pyRevit"

## 00\_*Advanced\_*Create a github repository (gitignore, licence, readme, ...)

main → master in github

and **open it with github desktop**

Une image contenant texte

Description générée automatiquement

Then ‘**show in explorer’**

Note that we could also do that from the github website

## 01\_ *Advanced\_*A minimal set of files and folders:

* File ***extension.yaml***

**type**: extension

**rocket\_mode\_compatible**: **true**

**name**: pyBiltNA # a name for your pyrevit extension

**description**: pyRevit tools creation workshop # a description of what it is

**author**: Jean-Marc Couffin # your name

**author\_profile**: https://linkedin.com/in/jmcouffin # [optional] a link to your profile

**url**: https://github.com/jmcouffin/pyRevit-BILT\_NA\_2022.git # the link to your extension's repository on github

**website**: http://eirannejad.github.io/pyRevit/ # [optional] a link to your website

**image**: https://ein.sh/pyRevit/pyRevitLogo.svg # [optional] a picture

* a folder structure:

📁BILT.tab/📁BILTpanel.panel/🔘Hello.pushbutton

***.tab*** will identify the toolbar in Revit

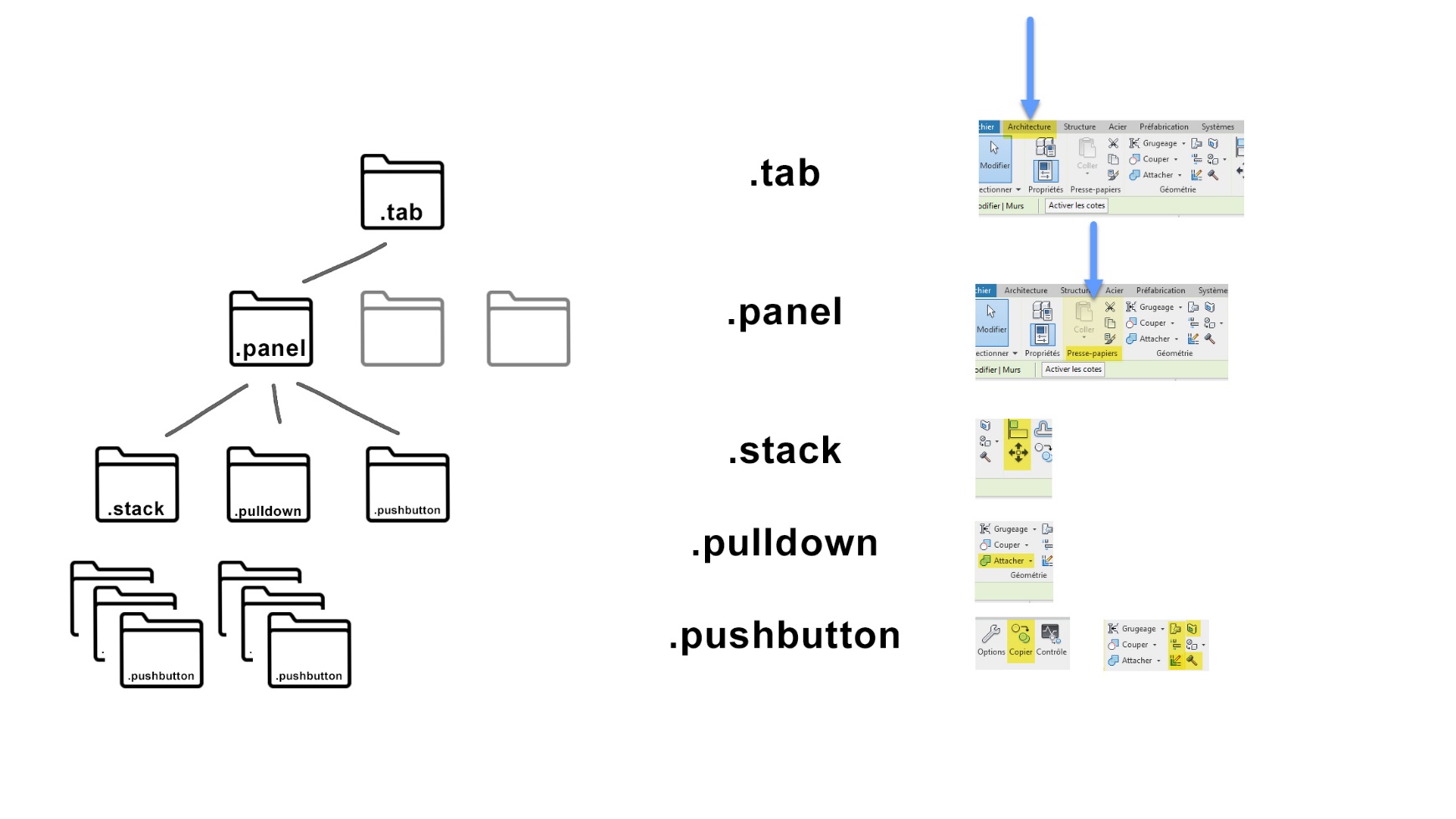
***.panel*** will help you group your buttons in sets

***.pushbutton*** is your first button in your toolbar

The structure of your toolbar is aligned with your folder structure.

Each folder extension in the form of .**variable** will be used to create: tabs, panels, pulldown, stacks, and so on.





## 02\_ *Advanced\_*And a file:

***script.py***

with the following content:

**print** **(**'Hello World hello le monde'**)**

Une image contenant texte

Description générée automatiquement

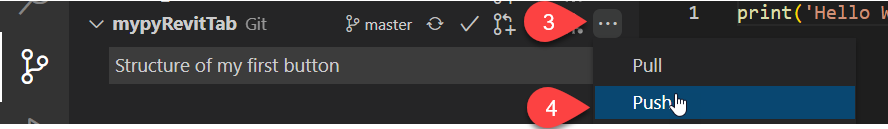
Une image contenant table

Description générée automatiquement

To commit and push all that to Github, either use **VSCode**:

Une image contenant texte

Description générée automatiquement

**CTRL+Enter** then Push to Github

Or **Github Desktop**:

Une image contenant texte

Description générée automatiquement

Then **CTRL+P** to push to github

*Okay we have potentially a toolbar setup and a first button on github, where do we go from there?*

## 03\_ *Advanced\_*Installation of the toolbar

*We want to link the github repository to pyRevit. The command line will help us do it*

* in command line (WIN + cmd), we will install our toolbar

**pyrevit extend ui pyBiltNA https**://github.com/jmcouffin/pyRevit-BILT\_NA\_2022.git --dest="C:\pyRevit"

we want to install it to a different folder than the one we worked on previously as the pyRevit Command Line (CLI) will create an .extension folder to make things happen between pyRevit and Revit

the pyrevit command is extremely powerful, try  
‘**pyrevit --help’** to see the possibilities

* let’s control everything went as planned, in the command line:

**pyrevit env**

it should list you: pyRevit clones, installs, extensions, and Revit installs as well

Une image contenant texte

Description générée automatiquement

## 04\_*Everyone\_*Update Tools

### Update

A way to update an extension is to use the following command line:

**pyrevit extensions update** pyBiltNA

In the form of a python script

# -\*- coding: UTF-8 -\*-

**import** os

os**.**system**(**'cmd /c "pyrevit extensions update pyBiltNA"'**)**

### Reload

If you just changed the code, you don’t need to refresh the UI

But if you changed the UI, pyRevit has a definition for that:

sessioninfo.get\_session\_uuid()

Directly from pyRevit Core tools: <https://github.com/eirannejad/pyRevit/blob/master/extensions/pyRevitCore.extension/pyRevit.tab/pyRevit.panel/tools.stack/Reload.pushbutton/script.py>

"""Reload pyRevit into new session."""

# -\*- coding=utf-8 -\*-

#pylint: disable=import-error,invalid-name,broad-except

**from** pyrevit **import** EXEC\_PARAMS

**from** pyrevit **import** script

**from** pyrevit **import** forms

**from** pyrevit**.**loader **import** sessionmgr

**from** pyrevit**.**loader **import** sessioninfo

res **=** **True**

**if** EXEC\_PARAMS**.**executed\_from\_ui**:**

res **=** forms**.**alert**(**'Reloading increases the memory footprint and is '

'automatically called by pyRevit when necessary.\n\n'

'pyRevit developers can manually reload when:\n'

' - New buttons are added.\n'

' - Buttons have been removed.\n'

' - Button icons have changed.\n'

' - Base C# code has changed.\n'

' - Value of pyRevit parameters\n'

' (e.g. \_\_title\_\_, \_\_doc\_\_, ...) have changed.\n'

' - Cached engines need to be cleared.\n\n'

'Are you sure you want to reload?'**,**

ok**=False,** yes**=True,** no**=True)**

**if** res**:**

logger **=** script**.**get\_logger**()**

results **=** script**.**get\_results**()**

# re-load pyrevit session.

logger**.**info**(**'Reloading....'**)**

sessionmgr**.**reload\_pyrevit**()**

results**.**newsession **=** sessioninfo**.**get\_session\_uuid**()**

## 05\_*Everyone\_*startup.py

Do stuffs at Revit startup:

Une image contenant texte

Description générée automatiquement

* Update your toolbar with:

# -\*- coding: UTF-8 -\*-

# auto update at startup

**import** os

os**.**system**(**'cmd /c "pyrevit extensions update pyBiltNA"'**)**

* Check if Revit version is up to date by getting the build number
* Check warnings and display to users

# -\*- coding: UTF-8 -\*-

**from** pyrevit **import** script

**from** pyrevit **import** revit**,** DB

output **=** script**.**get\_output**()**

doc **=** revit**.**doc

**def** doc\_warnings**(**doc**):**

warnings **=** doc**.**GetWarnings**()**

descriptions **=** **[]**

**for** warning **in** warnings**:**

descriptions**.**append**(**DB**.**FailureMessage**.**GetDescriptionText**(**warning**))**

**if** len**(**descriptions**):**

**return** str**(**len**(**descriptions**))** **+** ' Warnings in the project'

# set minimal value to empty string

warnings **=** ""

warnings **=** doc\_warnings**(**doc**)**

output**.**print\_md**(**warnings**)**

* Display a company message, new tools information, or good habits, …

# 03\_Building some more tools

## -02\_ *Everyone\_*pyRevit modules

**from pyrevit import …**

pyRevit comes with a set of modules that help you deal with Revit stubbornness; the documentation can be found here: <https://pyrevit.readthedocs.io/en/latest/>

The code itself is heavily commented > easy to read and understand

<https://github.com/eirannejad/pyRevit/tree/master/pyrevitlib>

## -01\_ *Everyone\_*Staples

* **Icon** file should be .png with a size of 96 x 96 pixels  
  <https://icons8.com/icons/set/pyrevit> is a good source for icons and let’s you combine, recolor them at will
* [**bundle.yaml**](https://www.notion.so/Bundle-Metadata-9fa4911c14fa49c48e715421400f1427)file will help us personalize the user experience, it works for all types of button:

*# from the pyRevit documentation*  
# bundle title

**title**: "Make\nPattern"

# title can also be in various locales

# pyRevit pulls the correct name based on Revit language

**title**:

**en\_us**: Test Bundle (Custom Title)

**chinese\_s**: 测试包

# bundle tooltip

**tooltip**: Create new patterns in Revit

# tooltip can also be in various locales

# pyRevit pulls the correct tooltip based on Revit language

**tooltip**:

**en\_us**: Create new patterns in Revit

**chinese\_s**: 创建新模式

# bundle highlighting ('new' or 'updated')

# Revit UI will show a orange marker on the button and a border around the tooltip

**highlight**: new # highlight as new

**highlight**: updated # highlight as updated

# bundle help url

**help\_url**: "https://www.youtube.com/watch?v=H7b8hjHbauE&t=8s&list=PLc\_1PNcpnV57FWI6G8Cd09umHpSOzvamf"

# help url can also be in various locales

# pyRevit pulls the correct help url based on Revit language

**help\_url**:

**en\_us**: "https://www.youtube.com/watch?v=H7b8hjHbauE&t=8s&list=PLc\_1PNcpnV57FWI6G8Cd09umHpSOzvamf"

**chinese\_s**: "https://www.youtube.com/watch?v=H7b8hjHbauE&t=8s&list=PLc\_1PNcpnV57FWI6G8Cd09umHpSOzvamf"

# bundle author

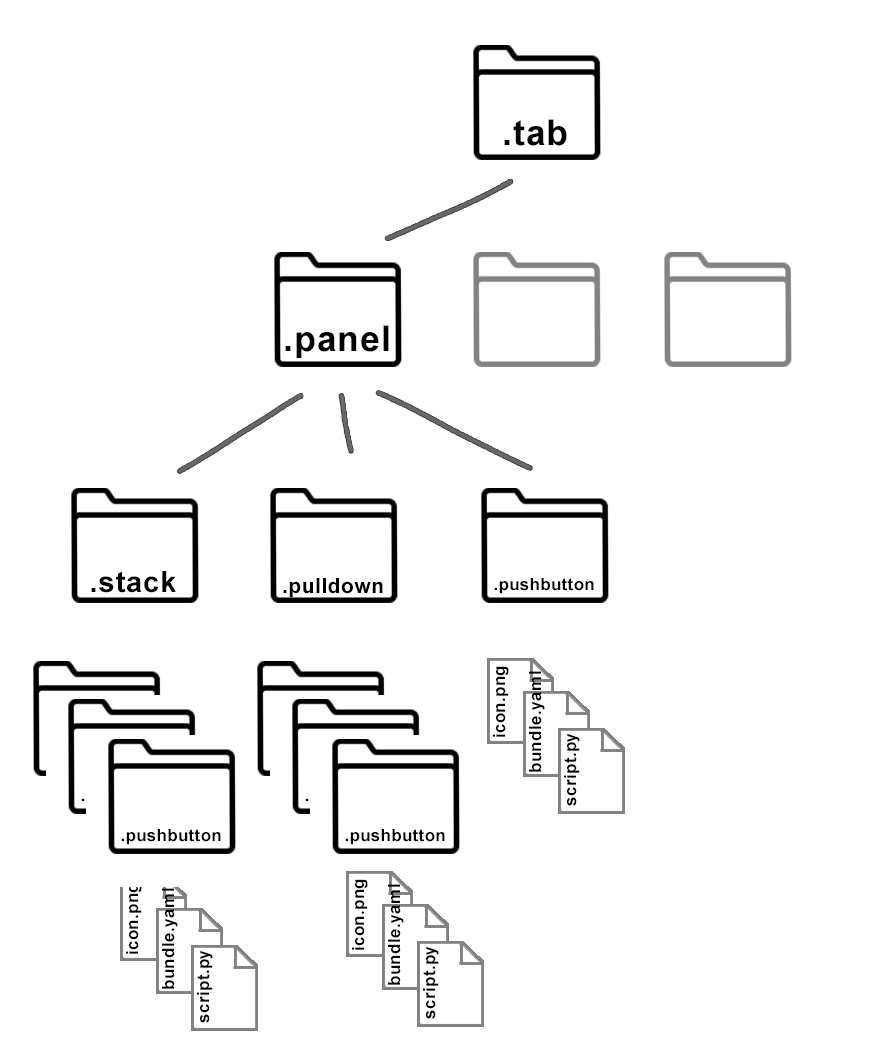
**author**: Ehsan Iran-Nejad

# bundle author can also be a list of authors

**authors**:

- John Doe

- Ehsan Iran-Nejad



## 00\_*Everyone\_.urlbutton*

**Recipe: folder ending with .urlbutton containing Icon.png + bundle.yaml > “hyperlink: ”**

*We will create a button that opens the internet browser to a specific URL*

* Bundle file with the key “hyperlink: ” is the necessary syntax required by pyRevit to let you click on the button to go to this specific address.

**hyperlink**: https://theoatmeal.com/comics/working\_home

## 01\_ *Everyone\_.pushbutton*

**Recipe: folder ending with .pushbutton containing Icon.png + bunde.yaml + \*script.py**

*We will create a button that Displays some text, a picture and also the revit build*

* \*script.py is a python file containing a command that can be directly related to Revit thanks to the Revit API (or) the pyrevit modules

**from** pyrevit **import** script

**from** pyrevit **import** revit

# grab existing windows

output **=** script**.**get\_output**()**

# close them

output**.**close\_others**()**

# center the new window

output**.**center**()**

# print a picture in a window

output**.**print\_md**(**'# No Revit was hurt in the process!'**)**

output**.**print\_image**(**"C:\pyRevit\pyBiltNA.extension\BILT.tab\Types of buttons.panel\\01\_Push Button.pushbutton\meme.jpg"**)**

# let's do something with revit at the Application Level

output**.**print\_md**(**'You are using the following revit build {}'**.**format**(**revit**.**HOST\_APP**.**build**))**

# automatically close the window

output**.**self\_destruct**(**15**)**

If you need the picture file: <https://github.com/jmcouffin/pyRevit-BILT_NA_2022/blob/master/BILT.tab/Types%20of%20buttons.panel/01_Push%20Button.pushbutton/meme.jpg>

For more information: <https://www.notion.so/pyRevit-Bundles-12323e3090904d9aa7cdc3d82095d3e3#32c67ca843c84d0684ea7f7e876b9737>

## 02\_ *Everyone\_.pushbutton* Dynamo flavored

**Recipe: folder ending with .pushbutton containing Icon.png + bundle.yaml (clean engine) + \*script.dyn**

The purpose of the lab is not to create a dynamo file, so let’s just grab this one <https://github.com/jmcouffin/pyRevit-BILT_NA_2022/blob/master/BILT.tab/Types%20of%20buttons.panel/02_Push%20Button%20DYN.pushbutton/script.dyn>

*This dynamo scripts let’s you untick specific sections of view templates*

Two items are of importance:

1. The .dyn file should be set to automatic:
   1. To do so, open it with a text editor
   2. CTRL+F search for “RunType”
   3. Type “Automatic” instead of “Manual”



1. The **bundle.yaml** file for this button should contain the key “engine” and subkey “clean” set to true if you want to restart the dynamo engine in the background for each run of the script. It will take longer to run but might be necessary depending on your dynamo script (the ones with a UI in particular).

**engine**:

**clean**: **true**

For more info: <https://www.notion.so/pyRevit-Bundles-12323e3090904d9aa7cdc3d82095d3e3#193440cef00048e7a62f4c541e3c83e7>

## 03\_ *Advanced\_.content*

**Recipe: folder ending with .content containing Icon.png + \*content.rfa + \*other.rfa + bundle.yaml**

*We will create a button that lets us load two families*

* This one allows you to load two types of Revit family
  + The first one needs to be named **\*.content.rfa**, and will be accessible clicking the button, you could use this one <https://github.com/jmcouffin/pyRevit-BILT_NA_2022/blob/master/BILT.tab/Types%20of%20buttons.panel/03_Content%20Button.content/AT-AT_16488_content.rfa>
  + The second one **\*other.rfa**, and will be accessible by shift clicking the button, you could use this one <https://github.com/jmcouffin/pyRevit-BILT_NA_2022/blob/master/BILT.tab/Types%20of%20buttons.panel/03_Content%20Button.content/Star_Wars_R2D2_4123_other.rfa>

## 04\_ *Advanced\_*.nobutton

**Recipe: folder ending with .nobutton containing Icon.png + bundle.yaml + \*script.py**

*We will create a button to de-activate the analytical model for structural elements*

* The script uncommented version:

**from** pyrevit **import** script**,** revit**,** DB**,** forms

output **=** script**.**get\_output**()**

doc **=** revit**.**doc

param **=** DB**.**BuiltInParameter**.**STRUCTURAL\_ANALYTICAL\_MODEL

provider **=** DB**.**ParameterValueProvider**(** DB**.**ElementId**(** param **)** **)**

evaluator **=** DB**.**FilterNumericEquals**()**

rule **=** DB**.**FilterIntegerRule**(** provider**,** evaluator**,** 1 **)**

filter **=** DB**.**ElementParameterFilter**(** rule **)**

analyticalCollector **=** DB**.**FilteredElementCollector**(** doc **).**WherePasses**(** filter **).**ToElements**()**

processed\_list **=** 0

**with** revit**.**Transaction**(**'Set Analytical Model'**):**

**for** i **in** analyticalCollector**:**

object\_param\_AnalyticalModel **=** i**.**get\_Parameter**(**DB**.**BuiltInParameter**.**STRUCTURAL\_ANALYTICAL\_MODEL**)**

new\_value **=** **False**

**try:**

object\_param\_AnalyticalModel**.**Set**(**new\_value**)**

processed\_list **+=** 1

**except:**

**pass**

output**.**close\_others**(**all\_open\_outputs**=True)**

msg **=** str**(**processed\_list**)** **+** ' processed elements'

forms**.**alert**(**msg**,** title**=**'Turn of analytical model property'**,** ok**=True)**

The commented version can be found here: <https://github.com/jmcouffin/pyRevit-BILT_NA_2022/blob/master/BILT.tab/Types%20of%20buttons.panel/04_No%20Button.nobutton/script.py>

* The bundle file will have the following information:

**title**:

**fr\_fr**: Modèle Analytique OFF

**en\_us**: OFF Analytical model

**tooltip**:

**fr\_fr**: Permet de désactiver tous les éléments ayant le modèle analytique coché

**en\_us**: De-activates analytical model on structural elements

## 05\_ *Advanced\_*.pushbutton with configuration



The black dot is for SHIFT+Click

**Recipe: folder ending with .pushbutton containing Icon.png + bundle.yaml + \*script.py + config.py**

*We will create a button that grabs a series of information from the current Revit file based on a configuration specified separately*

* Config file named config.py will be run if the button is pressed with the SHIFT key

**from** pyrevit **import** script**,** forms

# -\*- coding: utf-8 -\*-

my\_config **=** script**.**get\_config**()**

**def** get\_control\_points**():**

# grab token

list\_checks **=** **[**"Project Name"**,** "Project Number"**,** "Warnings"**]**

form **=** forms**.**SelectFromList**.**show**(**list\_checks**,** "Checks"**,** 300**,**500**,** multiselect**=True,** infopanel**=True)**

**if** form**:**

setattr**(**my\_config**,** "BILT\_tests"**,** form**)**

script**.**save\_config**()**

**else:**

setattr**(**my\_config**,** "BILT\_tests"**,**list\_checks**)**

script**.**save\_config**()**

**if** \_\_name\_\_ **==** "\_\_main\_\_"**:**

get\_control\_points**()**

* The script file, taking advantage of the configuration:

# -\*- coding: UTF-8 -\*-

**import** datetime

**from** pyrevit **import** script

**from** pyrevit **import** revit**,** DB

output **=** script**.**get\_output**()**

output**.**close\_others**(True)**

output**.**center**()**

output**.**set\_title**(**'Models Checker'**)**

doc **=** revit**.**doc

# Grab data from config

my\_config **=** script**.**get\_config**()**

tests **=** getattr**(**my\_config**,** "BILT\_tests"**)**

# Series of queries

**def** project\_number**(**doc**):**

project\_number **=** doc**.**ProjectInformation**.**Number

**return** project\_number

**def** project\_name**(**doc**):**

project\_name **=** doc**.**ProjectInformation**.**Name

**return** project\_name

**def** doc\_warnings**(**doc**):**

warnings **=** doc**.**GetWarnings**()**

descriptions **=** **[]**

**for** warning **in** warnings**:**

descriptions**.**append**(**DB**.**FailureMessage**.**GetDescriptionText**(**warning**))**

**if** len**(**descriptions**):**

**return** str**(**len**(**descriptions**))** **+** ' Warnings in the project'

# set minimal value to empty string

pname**,** pnumber**,** warnings **=** ""**,** ""**,** ""

# check if queries requested in config file

**if** tests **==** **[]** **or** tests **==** **None:**

pname **=** project\_name**(**doc**)**

pnumber **=** project\_number**(**doc**)**

warnings **=** warnings**(**doc**)**

**if** "Project Name" **in** tests**:**

pname **=** project\_name**(**doc**)**

**if** "Project Number" **in** tests**:**

pnumber **=** project\_number**(**doc**)**

**if** "Warnings" **in** tests**:**

warnings **=** doc\_warnings**(**doc**)**

# print the whole thing

output**.**print\_md**(**pname **+** "\n\n" **+** pnumber **+** "\n\n" **+** warnings**)**

# 04\_ *Advanced\_*Distribute to the team

## Install pyRevit, pyRevit CLI and the toolbar in one go

Powershell file **install\_pyrevit.ps1**

#Declaring Path

#--------------------------------------------------

$basefilePath **=** "C:\pyRevit"

#Where pyRevit Installer are located

$pyRevit **=** $basefilePath **+** "\pyRevit\_4.8.10.22040\_signed.exe"

$pyRevitCLI **=** $basefilePath **+** "\pyRevit\_CLI\_4.8.10.22040\_signed.exe"

#Installing pyRevit or pyRevit CLI

#--------------------------------------------------

Start**-**Process **-**Wait **-**FilePath $pyRevit **-**arg "/qn" **-**PassThru

Write**-**Host "pyRevit Installed"

Start**-**Process **-**Wait **-**FilePath $pyRevitCLI **-**arg "/qn" **-**PassThru

Write**-**Host "pyRevit CLI Installed"

#Extend pyRevit

#--------------------------------------------------

pyrevit extend ui pyBiltNA https**://**github**.**com**/**jmcouffin**/**pyRevit**-**BILT\_NA\_2022**.**git **--**dest**=**"C:\pyRevit"

+ a **install\_pyrevit.bat** file

powershell -ExecutionPolicy Bypass -File "%~dp0\ install\_pyrevit.ps1"

a more complete approach <https://www.notion.so/pyRevit-For-Teams-ddc6c312d6f6488691eed2ec7704fd97>

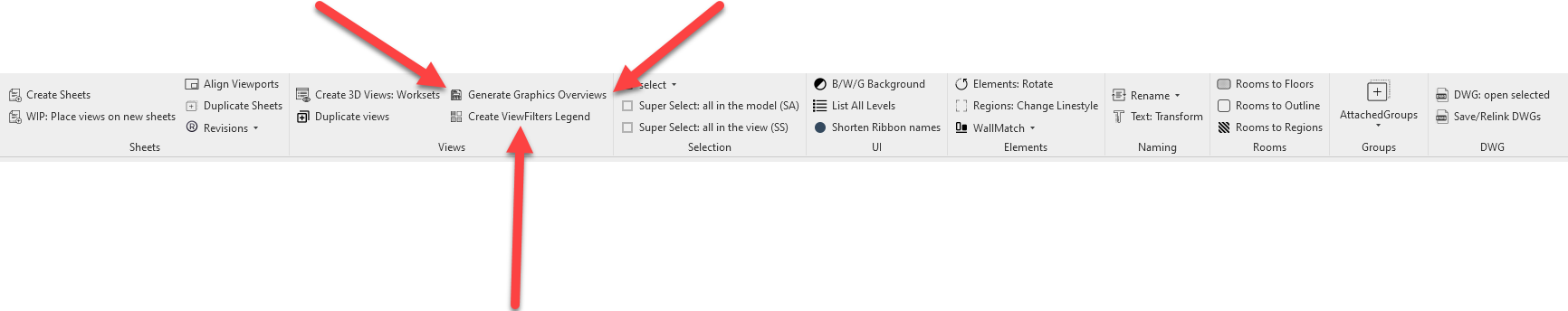
# outro

All the code of pyRevit is open source and many extensions exist, so be curious, explore and don’t forget:

**ALT+Click** will open the folder containing the code of the button clicked

Cool extensions to look at:

[EF-Tools](https://github.com/ErikFrits/EF-Tools) and Erik does badass tutorials. The Generate Graphics Overrides is Huge…



[pyChilizer](https://github.com/dnenov/pyChilizer) Deyan and Daria are doing a great job. Legend from Filters… Inplace to Loadable!!!

Une image contenant texte, périphérique, mètre, jauge

Description générée automatiquement

And these ones in the extensions menu of pyRevit:

