

1. Старт

Статьи посвященные быстрому старту работы с pyRevit

1.1. Давайте знакомиться, pyRevit



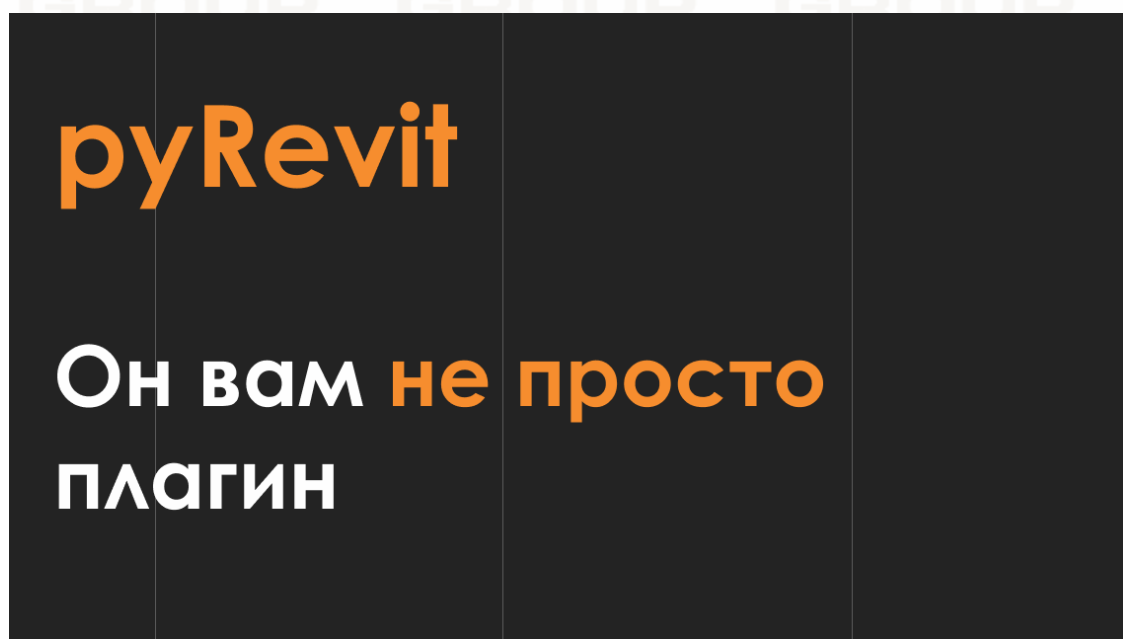
1.1.1. Введение

В мире архитектурного и инженерного проектирования программное обеспечение Autodesk Revit занимает особое место. Однако, несмотря на его мощь и возможности, у многих пользователей возникает желание расширить функционал

программы, сделать ее более гибкой и адаптированной под свои задачи. Здесь на помощь приходит **pyRevit** — инструмент, который способен существенно повысить эффективность работы в Revit.

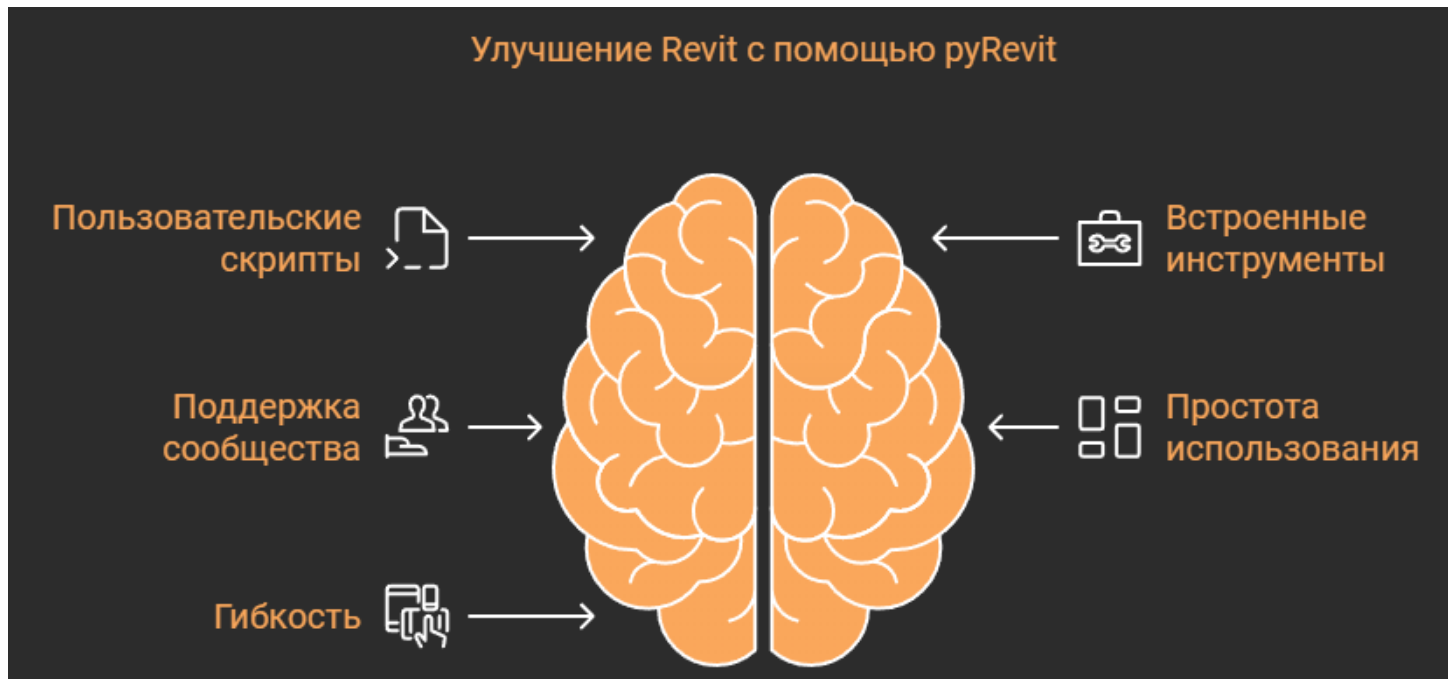
1.1.2. Что такое pyRevit?

pyRevit — это бесплатный и открытый фреймворк, разработанный для расширения возможностей Autodesk Revit с помощью языка программирования Python. Он позволяет создавать пользовательские скрипты, инструменты и даже целые панели инструментов, интегрированные непосредственно в интерфейс Revit.



Основная идея pyRevit заключается в том, чтобы предоставить пользователям простой и доступный способ автоматизировать рутинные задачи, создавать собственные инструменты и делиться ими с сообществом. Благодаря использованию Python, одного из самых популярных и простых в освоении языков программирования, pyRevit открывает двери в мир кастомизации Revit даже для тех, кто не является профессиональным программистом.

Улучшение Revit с помощью pyRevit



1.1.3. Встроенные кнопки и инструменты pyRevit

После установки pyRevit в Revit появляется новая вкладка с множеством полезных инструментов. Эти инструменты разработаны для решения различных задач, с которыми сталкиваются пользователи в повседневной работе:

- **Анализ и проверка моделей:** инструменты для поиска ошибок, дубликатов, несоответствий в модели.
- **Управление видами и листами:** быстрые способы создания и настройки видов, генерации листов и управления ими.
- **Работа с параметрами:** массовое изменение параметров, экспорт и импорт данных.
- **Геометрические операции:** дополнительные функции для работы с геометрией, которые отсутствуют в стандартном наборе Revit.

Эти инструменты созданы сообществом и постоянно обновляются, что позволяет решать самые актуальные задачи и быть в курсе последних тенденций в отрасли.

1.1.4. Создание собственных инструментов

Одним из главных преимуществ pyRevit является возможность создавать собственные скрипты и инструменты. Это открывает безграничные возможности для персонализации Revit под конкретные потребности:

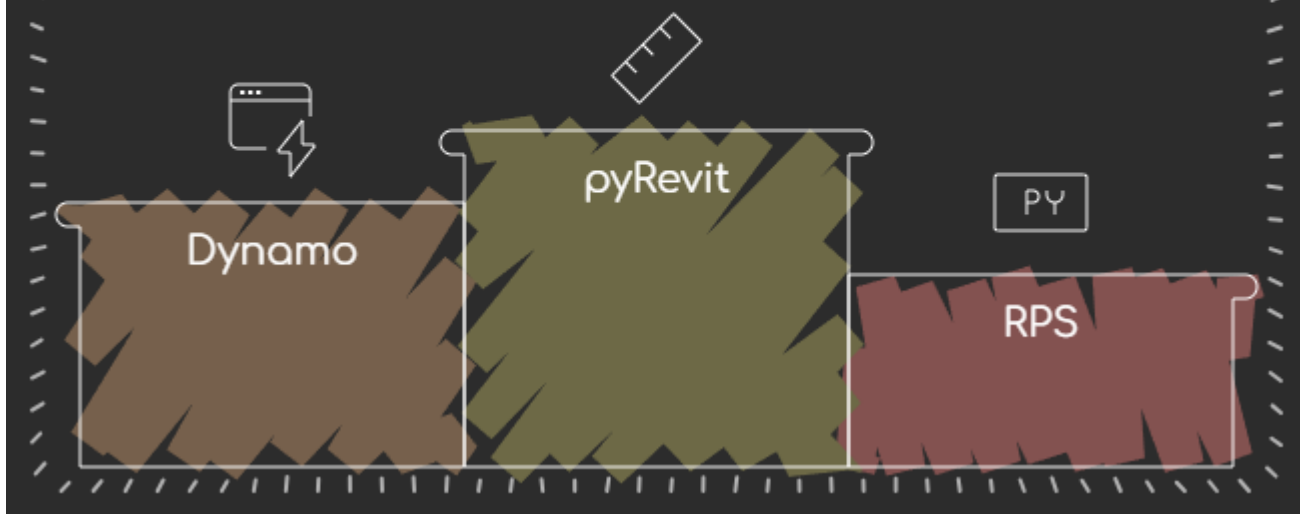
- **Автоматизация рутинных задач:** если вы регулярно выполняете одни и те же действия, вы можете написать скрипт, который сделает это за вас в несколько кликов.
- **Расширение функционала:** добавляйте новые функции, которых нет в стандартном наборе Revit.
- **Интеграция с внешними данными:** связывайте вашу модель с внешними источниками данных, такими как базы данных или веб-сервисы.

Создание собственных инструментов на Python не только повышает эффективность работы, но и способствует профессиональному росту, открывая новые горизонты в программировании и автоматизации.



1.1.5. Преимущества pyRevit по сравнению с RPS, Dynamo и C#

Инструменты для расширения функциональности Revit



Многие могут задаться вопросом: "Зачем использовать pyRevit, если есть другие инструменты для расширения Revit, такие как RevitPythonShell (RPS), Dynamo или возможности программирования на C#?"

Вот несколько причин, почему pyRevit выделяется на их фоне:

- **Удобство использования:** pyRevit интегрируется непосредственно в интерфейс Revit, предоставляя интуитивно понятный доступ к инструментам и скриптам.
- **Простота разработки:** Python считается одним из самых простых языков программирования для изучения. Его синтаксис понятен и логичен, что облегчает процесс создания скриптов даже для новичков.
- **Быстрая интеграция:** в отличие от RPS, pyRevit позволяет создавать полноценные кнопки и панели инструментов, что делает использование скриптов более удобным и профессиональным.
- **Гибкость:** хотя Dynamo предоставляет визуальное программирование, оно может быть менее эффективным для некоторых задач. Python в pyRevit позволяет писать более сложные и производительные скрипты.

- **Сообщество и поддержка:** pyRevit имеет активное сообщество пользователей и разработчиков, которые постоянно обмениваются идеями, скриптами и оказывают поддержку новичкам.
- **Отсутствие необходимости компиляции:** в отличие от C#, где для создания плагинов требуется компиляция кода, в pyRevit скрипты пишутся и запускаются напрямую, что ускоряет процесс разработки и отладки.

1.1.6. Заключение

pyRevit — это мощный инструмент, который может значительно повысить эффективность вашей работы в Autodesk Revit. Он объединяет в себе простоту использования, гибкость и широкие возможности для кастомизации.

Если вы ищете способ упростить рутинные задачи, добавить новые функции в Revit или просто хотите расширить свои навыки и возможности, pyRevit — это то, что вам нужно.

В следующих статьях мы более подробно рассмотрим, как установить pyRevit, как пользоваться встроенными инструментами и как начать создавать собственные скрипты. Присоединяйтесь к нам в этом увлекательном путешествии и откройте для себя новые возможности в мире BIM-проектирования с pyRevit!

1.2. Установка

pyRevit №0

Как установить самый лучший плагин



"...самый лучший плагин" - это не только кликабельный заголовок, но и моё официальное заявление.

Причины описаны [ТУТ](#)

1.2.1. Установка



Переходим на github: <https://github.com/pyrevitlabs/pyRevit/releases>

Выбираем версию для установки:


Downloads

- ◆ See **Assets** section below for all download options

pyRevit

-  [pyRevit 4.8.16.24121 Installer](#)
-  [pyRevit 4.8.16.24121 Installer](#) - Admin / All Users / %PROGRAMDATA%

pyRevit CLI (Command line utility)

-  [pyRevit CLI 4.8.16.24121 Installer](#) - Admin / System %PATH%

- 1ый installer ставит в \AppData\Roaming\pyRevit-Master
- 2ой installer ставит на диск C и имеет возможность настроить классические расширения.

Зачем их 2?

На случай если у вас нет прав администратора на компьютере.

Далее:

- ждем завершения скачивания
- закрываем ревиты
- запускаем установку.

Вот собственно и все.

Теперь у вас есть самый крутой плагин. Я же говорил, что будет быстро.

1.2.2. Расширения

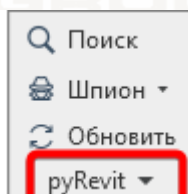
Пока мы не начали писать свои кнопки, плагины и скрипты, давайте поговорим о уже готовых пользовательских расширениях.

Их я условно разделю на "Одобрённые" и "Прочие", отличаются они лишь способ установки.

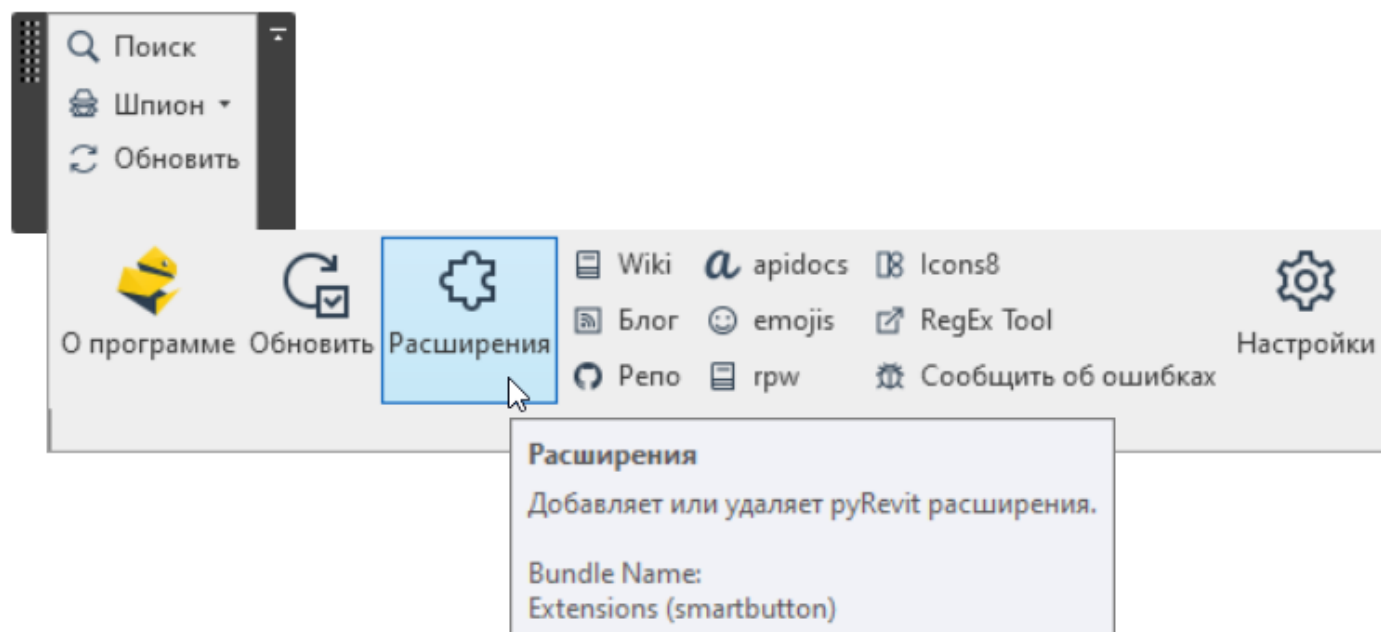
Кстати, если прям не терпится написать свое вот [статья](#).

1.2.2.1. Одобрённые

Переходим на вкладку pyRevit и ищем вот эту стрелку:



Далее "Расширения":



Откроется меню, где представлены одобренные самим pyRevit расширения:

Менеджер расширений pyRevit

Список зарегистрированных расширений:

A-Z

Имя	Тип	Автор	Built-in	Rocket-Mode	Установлен	Статус	Последний коммит
pyRevitCore	Инструменты Revit UI	Ehsan Iran-Nejad	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	Да	Вкл	
pyRevitTools	Инструменты Revit UI	Ehsan Iran-Nejad	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	Да	Выкл	
PyRevitPlus	Инструменты Revit UI	Gui Talarico	<input type="checkbox"/>	<input type="checkbox"/>	Нет	--	--
PyRevitMEP	Инструменты Revit UI	Cyril Waechter	<input type="checkbox"/>	<input checked="" type="checkbox"/>	Нет	--	--
pyApex	Инструменты Revit UI	Aleksey Melnikov	<input type="checkbox"/>	<input checked="" type="checkbox"/>	Нет	--	--
Revitron	Библиотека IronPython	Marc Anton Dahmen	<input type="checkbox"/>	<input checked="" type="checkbox"/>	Нет	--	--
Revitron UI	Инструменты Revit UI	Marc Anton Dahmen	<input type="checkbox"/>	<input checked="" type="checkbox"/>	Нет	--	--
pyStructure	Инструменты Revit UI	Shahabaz Sha	<input type="checkbox"/>	<input checked="" type="checkbox"/>	Нет	--	--
MEPDesign	Инструменты Revit UI	André Rodrigues da Silva	<input type="checkbox"/>	<input checked="" type="checkbox"/>	Нет	--	--
pyTiBa	Инструменты Revit UI	Tillmann Baumeister	<input type="checkbox"/>	<input checked="" type="checkbox"/>	Нет	--	--
EF-Tools	Инструменты Revit UI	Erik Frits	<input type="checkbox"/>	<input checked="" type="checkbox"/>	Нет	--	--
pyChilizer	Инструменты Revit UI	Archilizer	<input type="checkbox"/>	<input checked="" type="checkbox"/>	Нет	--	--
pySSG	Инструменты Revit UI	Kyle Bruxvoort	<input type="checkbox"/>	<input checked="" type="checkbox"/>	Нет	--	--
pyArchitect	Инструменты Revit UI	Roman Golev	<input type="checkbox"/>	<input checked="" type="checkbox"/>	Нет	--	--

pyRevitCore

Core Scripts for Autodesk Revit (<http://pyrevitlabs.github.io/pyRevit/>)

Разработчик: [Ehsan Iran-Nejad](#)

Репозиторий: <https://github.com/pyrevitlabs/pyRevit.git>

Папка установки:

C:\Program Files\pyRevit-Master\extensions\pyRevitCore.extension

Выключить

Любой из них можно выбрать и установить.

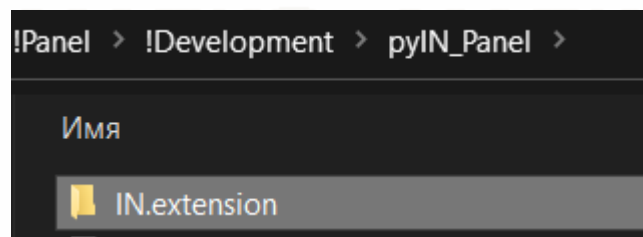
После установки панель перезапустится и станет доступна среди вкладок.

Для того чтобы попасть в список этих расширений необходимо следовать этой [инструкции](#).

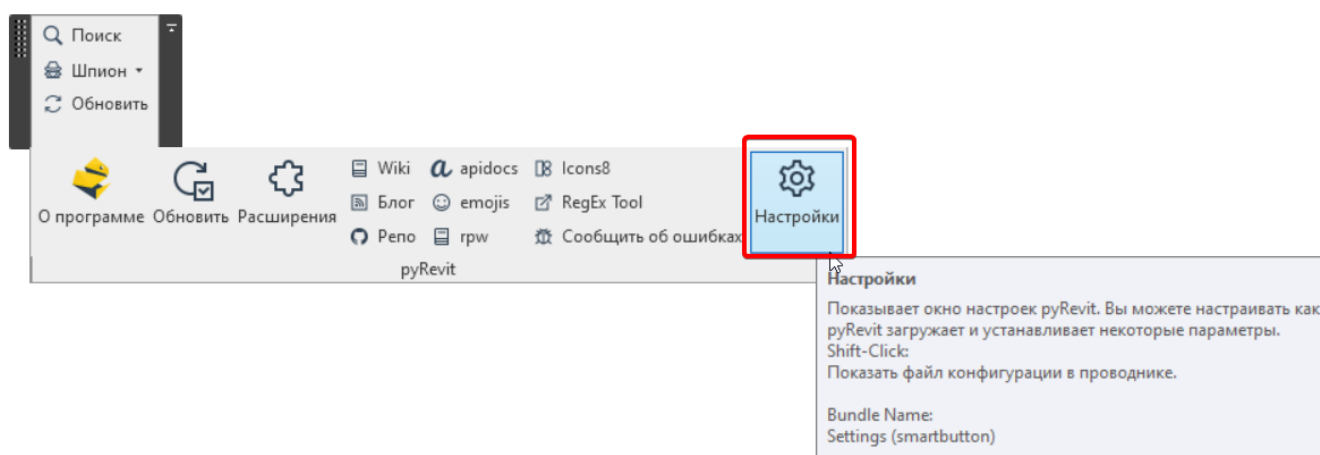
1.2.2.2. Прочие

Вот эти расширения надо поискать самостоятельно, но ниже в главе "Ресурсы" я закину пару расширений, которые смог найти.

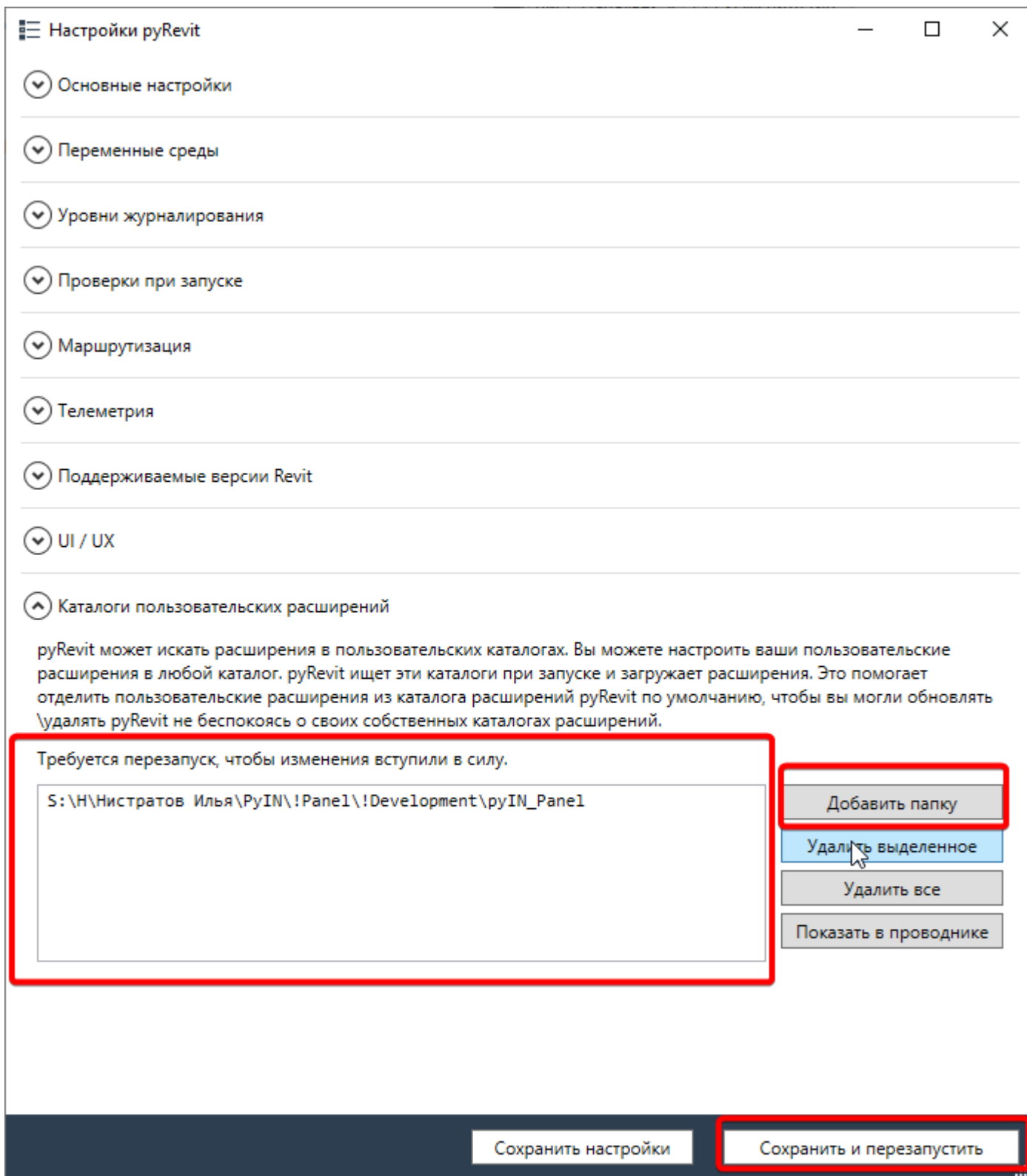
Теперь представим, что вы скачали чьё-то расширение или сделали его сами, и у вас будет папка с окончанием `.extension` в имени.



Надо указать паю путь до нее. Для этого переходим в "Настройки":



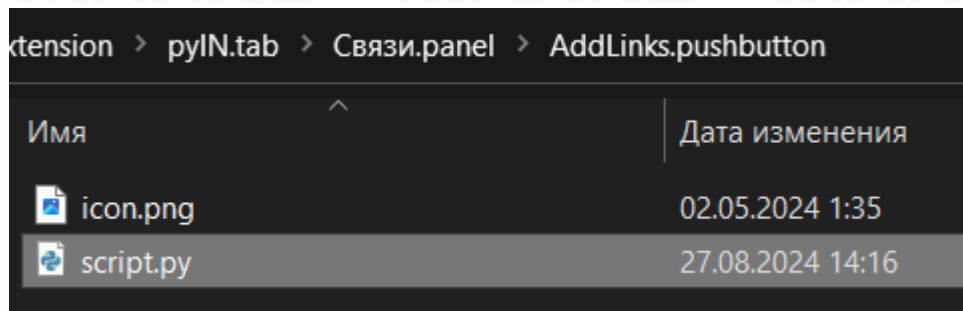
Откроем такое окно:



- Выбираем "Добавить папку"
- Указываем папку *В КОТОРОЙ* находится папка с расширением .extension
- Нажимаем "Сохранить и перезапустить"

После перезагрузки новые расширения отобразятся в Ревит.

И да, теперь любую из кнопок (при условии что она была написана на python) можно открыть и посмотреть ее исходный код. Для этого зажмите *Alt + ЛКМ* по кнопке - откроется папка с исполняющим файлом.



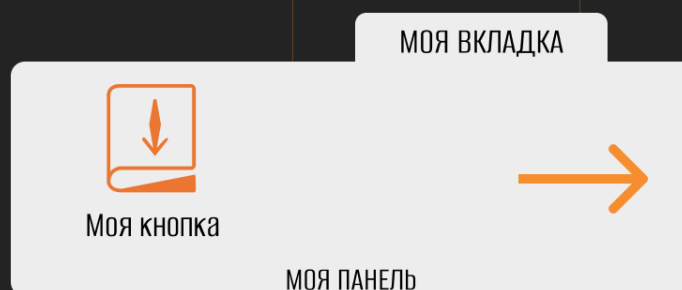
1.2.3. Ресурсы

- [Офф. документация по установке](#)
- [Офф. документация](#)
- [Офф. инструкция по установке расширений:](#)
- [Гайды от сообщества](#)

1.3. Первая кнопка

pyRevit.

Как создать свою кнопку в Revit.



Сегодня мы с вами создадим свою вкладку, свою панель и свою кнопку в пару кликов.

Все это благодаря pyRevit (далее "*пай*").

1.3.1. В начале была папка.

Добавить пользовательское расширение в пай легко.

Нужно лишь создать правильную структуру папок остальное на себя возьмет сам плагин.

Для начала предлагаю выбрать какую-то папку, куда мы будем в последствие добавлять все нами написанные кнопки.

Если мы говорим о разработке плагинов внутри компании, хорошей практикой будет расположить ее в общедоступном месте. После чего, все желающие могут подключиться к директории и использовать скрипты.

Для статьи я выберу папку просто на рабочем столе:

`C:\Users\chttm\OneDrive\Рабочий стол\TEST\`

Открываем папку TEST идем по пунктам:

1.3.2. ".extension"

Нам нужна папка с окончанием *.extension*.

Это сообщит pyRevit, что у нас внутри есть структура папок расширений.

Создаем папку с именем "*МоеРасширение.extension*".

.extension - специальное окончание в имени папки, так пай будет понимать, что это расширение.

МоеРасширение - произвольное имя, оно не будет фигурировать в самом ревите.

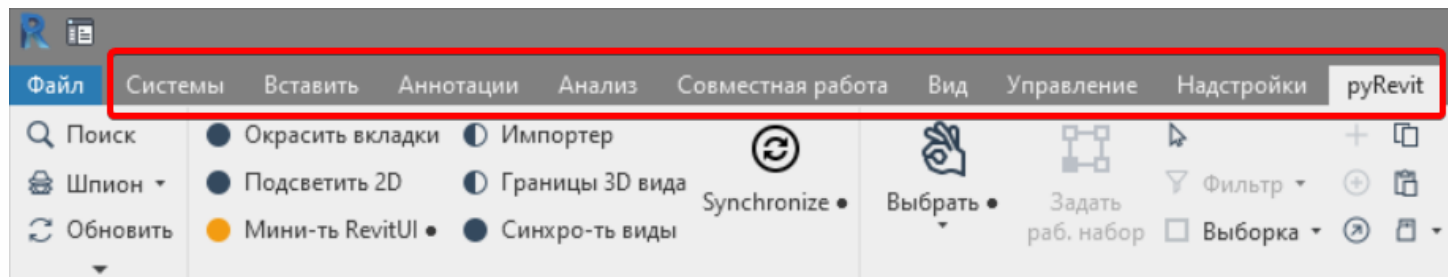
1.3.3. ".tab"

Внутри *МоеРасширение.extension* создадим папку "*МояВкладка.tab*".

.tab - специальное окончание, которое укажет паю, что эту папку надо рассматривать как вкладку ревита

"МояВкладка" - имя вкладки. Оно отобразится в ревите.

Это в Revit называется "**Вкладки**".



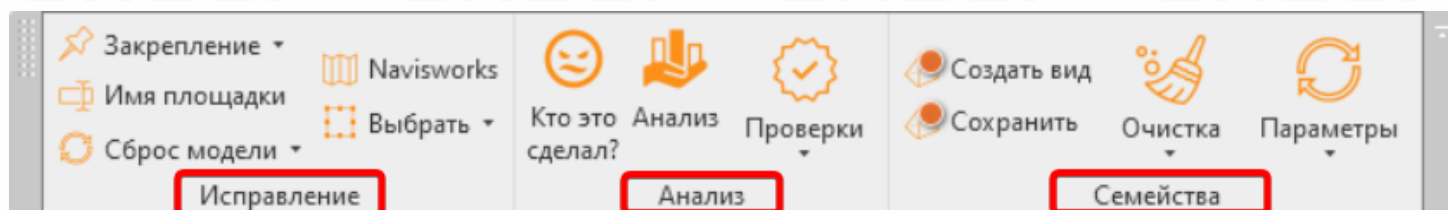
1.3.4 ".panel"

Перейдем в "*МояВкладка.tab*" и создадим папку *МояПанель1.panel*

.panel - специальное окончание, которое укажет паю, что содержимое этой папки надо поместить на отдельную панель.

МояПанель1 - имя панели.

Это "**Панели**".



1.3.5. ".pushbutton"

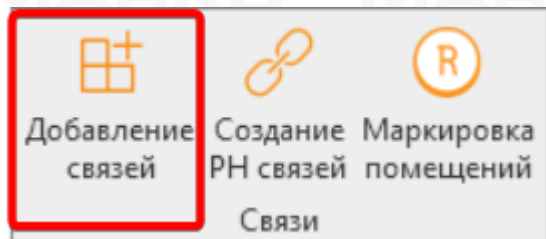
Наконец, нам нужно создать кнопки.

Перейдем в папку *МояПанель1.panel* и создадим папку "*МояПростоКнопка.pushbutton*".

.pushbutton - специальное окончание, которое укажет паю, что эта папка должна стать кнопкой.

МояПростоКнопка - имя кнопки, если нет иного указания имени.

Это "Кнопки".



Внутри этой папки должны находиться:

- `"script.py"` - скрипт.

□ Мы можем называть наши скрипты как угодно, лишь бы в конце стоял `script.py`.

□ Например, `supernatural_script.py`

- `icon.png` - иконка.

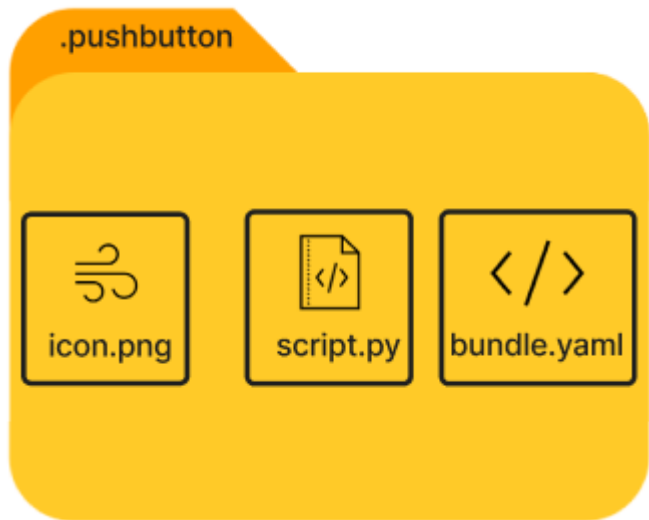
□ Не обязательный файл.

□ Максимальный размер `96x96` пикселей.

- `bundle.yaml` - файл метаданных. [Тут подробнее](#)

□ Не обязательный файл.

□ Удобный способ задать имя, контекст, описание, автора и многое другое для кнопки.



У нас получится такая структура папок:

- МоеРасширение.extension
 - - МояВкладка.tab
 - - - МояПанель1.panel
 - - - - МояПростоКнопка.pushbutton
 - - - - - script.py
 - - - - - icon.png



MARKS
GROUP

MARKS
GROUP

MARKS
GROUP

MARKS
GROUP

MARKS
GROUP

MARKS
GROUP

MARKS
GROUP

MARKS
GROUP

MARKS
GROUP

MARKS
GROUP

MARKS
GROUP

MARKS
GROUP

MARKS
GROUP

MARKS
GROUP

MARKS
GROUP

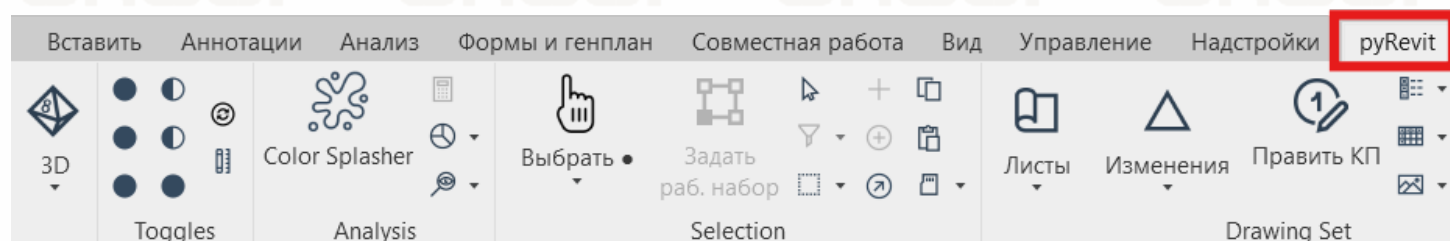
MARKS MARKS MARKS MARKS

Поздравляю вас, первая кнопка создана.

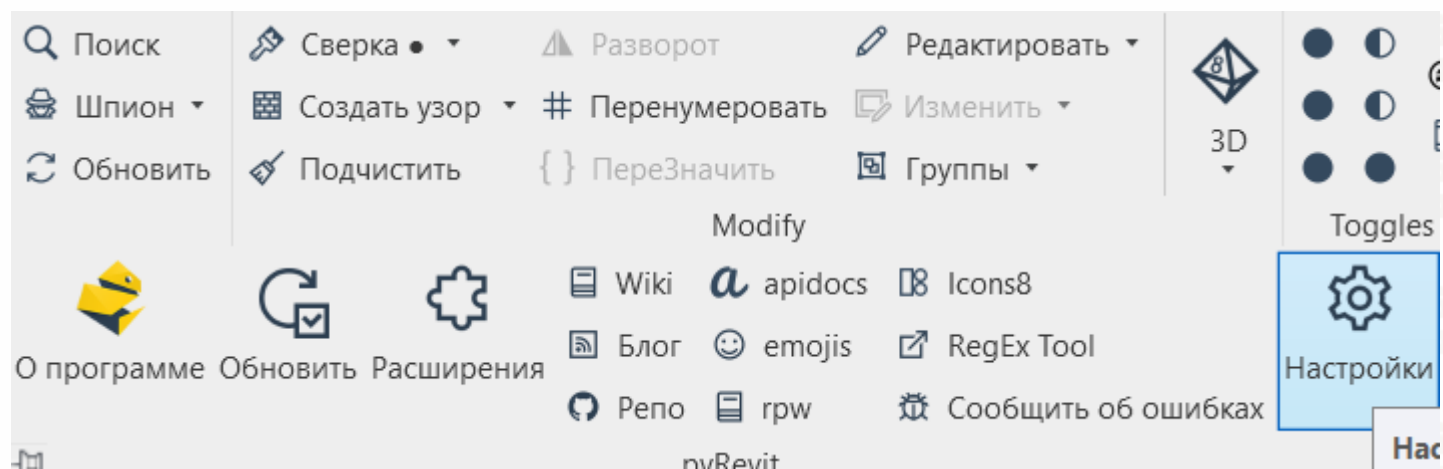
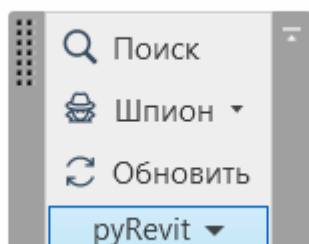
Теперь подключим наше расширение к Revit.

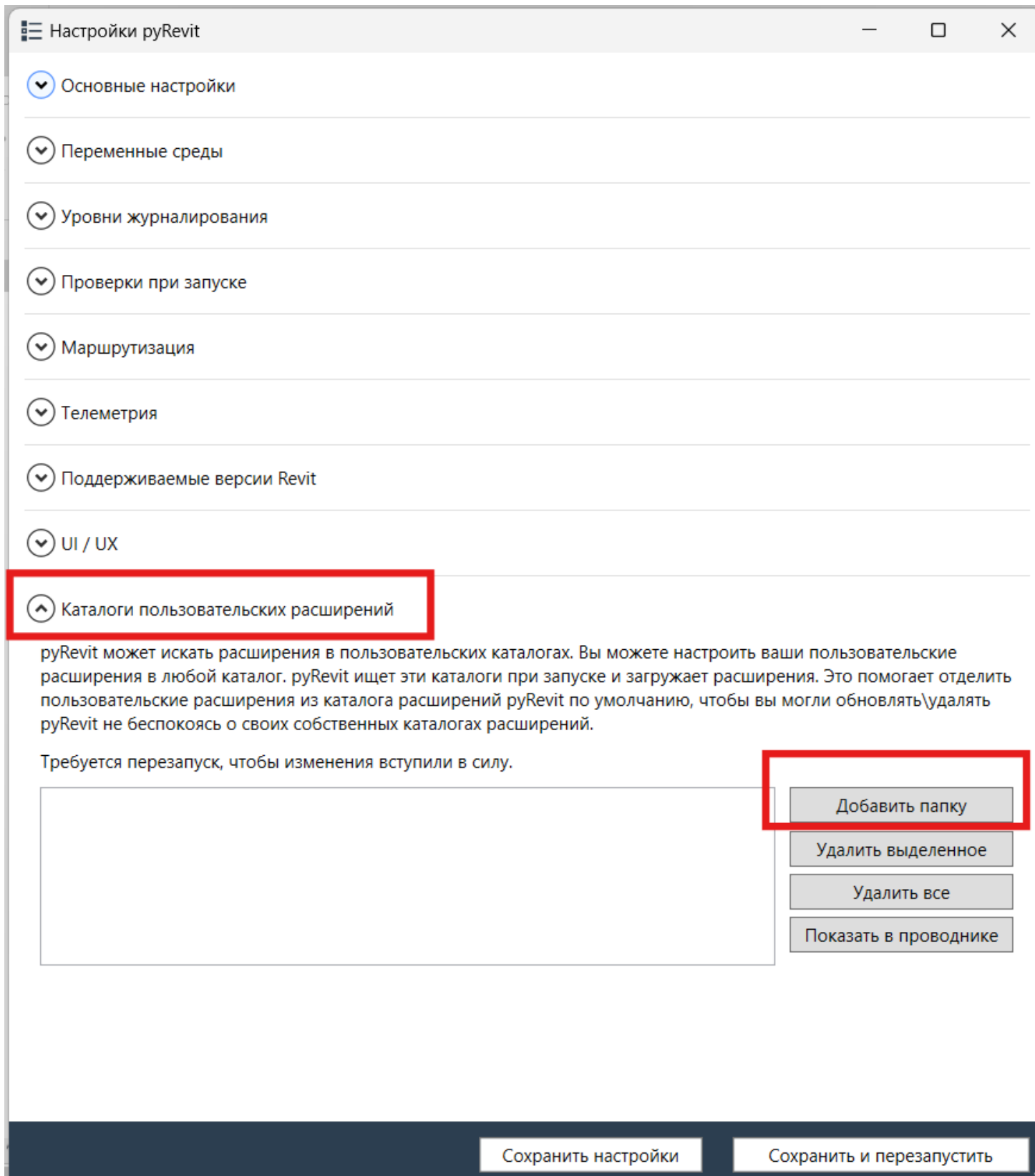
1.4. Подключение своего расширения

В этом нам поможет классическая вкладка pyRevit, которая появляется сразу после его установки.



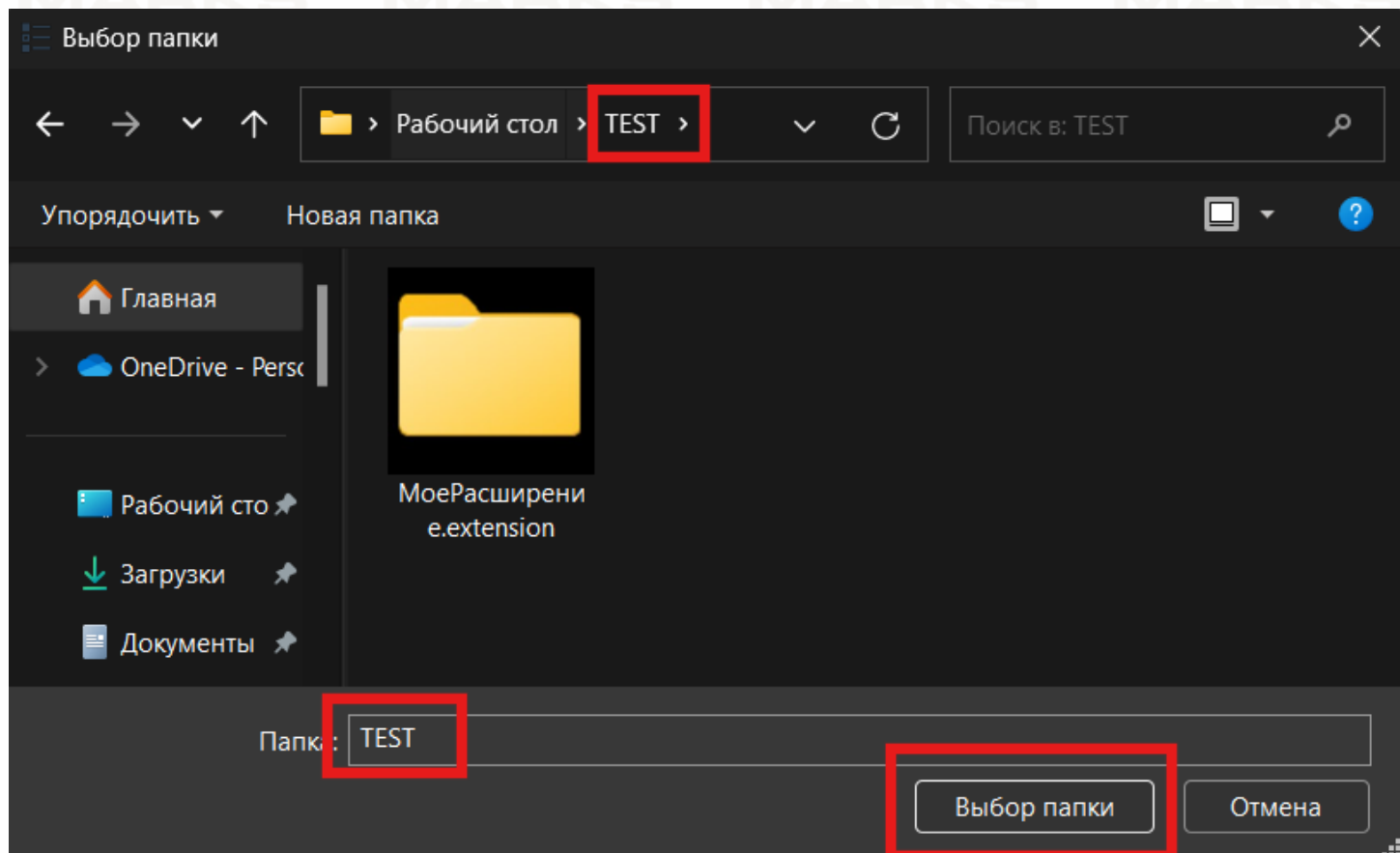
Переходим по пути: *pyRevit* - *Настройки* - *Каталоги пользовательских расширений*





Нажимаем “Добавить папку” и указываем папку TEST - то есть, ту папку, в которой находится папка с окончанием `.extension`

Еще раз - не саму папку с расширением `.extension`, а директорию в которой она находится.



И нажимаем “Выбор папки” и применяем настройки нажав “Сохранить и перезапустить”

Каталоги пользовательских расширений

pyRevit может искать расширения в пользовательских каталогах. Вы можете настроить ваши пользовательские расширения в любой каталог. pyRevit ищет эти каталоги при запуске и загружает расширения. Это помогает отделить пользовательские расширения из каталога расширений pyRevit по умолчанию, чтобы вы могли обновлять/удалять pyRevit не беспокоясь о своих собственных каталогах расширений.

Требуется перезапуск, чтобы изменения вступили в силу.

C:\Users\chttm\OneDrive\Рабочий стол\TEST	Добавить папку
C:\pyin	Удалить выделенное
	Удалить все
	Показать в проводнике

Сохранить настройки

Сохранить и перезапустить

После перезагрузки появляется новая вкладка “МояВкладка”

pyRevit

pyIN

MARKS-ПРОЧЕЕ

МояВкладка

Откроем ее и увидим нашу первую кнопку.



МояПростоКнопка

МояПанель1

1.5. Пишем первый скрипт

Раз уж есть кнопка, предлагаю познакомиться немного с api и написать какую-то базовую штуку, но только не “привет мир”.

Пусть кнопка закрепляет все наши оси в проекте.

Для этого откроем файл *script.py*. В качестве редактора я буду использовать VSCode.

Для того чтобы быстро открыть скрипт какой-либо кнопки, необходимо

активировать ее левой кнопкой мыши зажав *ALT*

ALT + Click - открывает папку с скриптом

Напишем первые строчки - метаданные.

```
1  # -*- coding: utf-8 -*-
2
3  __title__ = "Моя просто кнопка."
4  __author__ = 'NistratovIlia'
5  __doc__ = "Описание кнопки"
```

Пройдем по ним

1 - Задаем кодировку для отображения и использования русских символов:

Указание кодировки UTF-8 позволяет корректно отображать и использовать русские символы.

3-5 - Заголовок, автор и описание скрипта:

- `__title__`: Устанавливает заголовок скрипта.
- `__author__`: Указывает автора скрипта.
- `__doc__`: Предоставляет описание скрипта.

Это метаданные для кнопки. Их можно определять в качестве глобальных переменных в самом коде или в специальном файле *bundle.yaml*. О нем поговорим попозже.

```
6
7  from Autodesk.Revit import DB
8
9  doc = __revit__.ActiveUIDocument.Document
10
11  def pin(el,status):
12      el.get_Parameter(DB.BuiltInParameter.ELEMENT_LOCKED_PARAM).Set(status)
13      print("Прикрепил ось. Имя: {} ID:{}".format(el.Name,el.Id))
14
15  grids = DB.FilteredElementCollector(doc).\
16      OfCategory(DB.BuiltInCategory.OST_Grids).\
17      WhereElementIsNotElementType().\
18      ToElements()
19
20  with DB.Transaction(doc,"Автозакрепление") as t:
21      t.Start()
22      if grids:
23          for grid in grids:
24              pin(grid,1)
25      t.Commit()
```

7 - Импортируем библиотеку Autodesk Revit API: Импортируем пространство имен DB из библиотеки Autodesk Revit, которое содержит классы и методы для работы с элементами Revit.

9 - Получаем активный документ Revit: doc хранит ссылку на текущий активный документ Revit, который используется для выполнения операций с элементами модели.

11-13 - Определяем функцию pin для закрепления элемента: Функция которая будет закреплять элемент и выводить информацию об этом через print

15-18 - Создаем коллекцию всех осей в документе:

DB.FilteredElementCollector(doc): Создает коллекцию для фильтрации элементов в документе doc.

.OfCategory(DB.BuiltInCategory.OST_Grids): Фильтрует коллекцию по категории "Оси" (Grids).

.WhereElementIsNotElementType(): Исключает из коллекции типы элементов, оставляя только экземпляры.

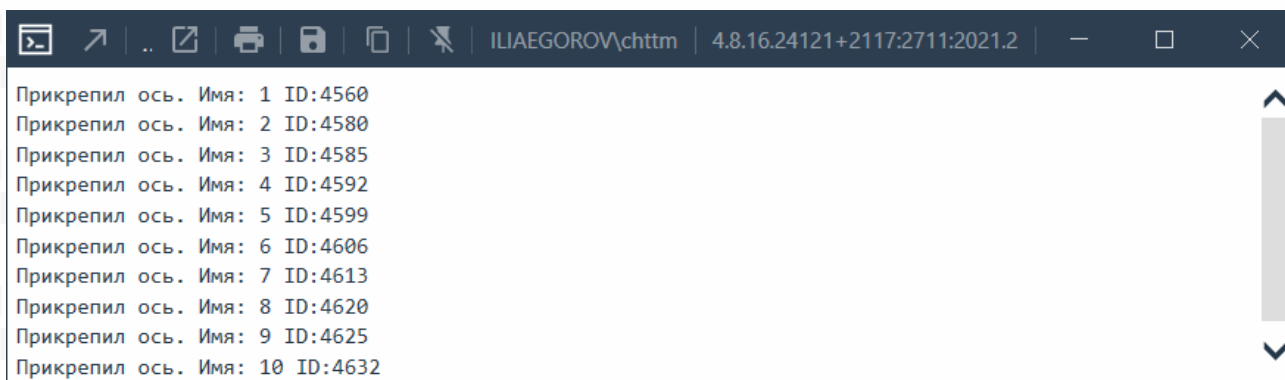
.ToElements(): Преобразует отфильтрованную коллекцию в список элементов.

20-25 - Открываем транзакцию для выполнения изменений в документе:

with DB.Transaction(doc, "Автозакрепление") as t: Создает транзакцию с именем "Автозакрепление" и открывает ее в контексте *with*.

- *t.Start()*: Запускает транзакцию.
- *if grids*: Проверяет, если список осей не пустой.
- *for grid in grids*: Перебирает каждую ось в списке grids.
- *pin(grid, 1)*: Вызывает функцию pin для закрепления текущей оси grid (устанавливает статус в 1, что означает "закреплено").
- *t.Commit()*: Фиксирует изменения в документе, завершает транзакцию.

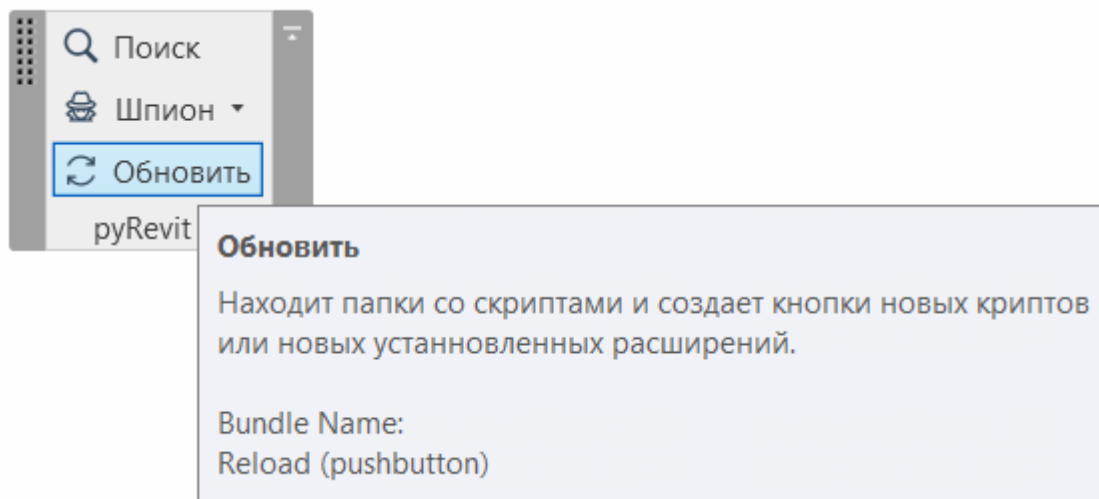
Сохраняем скрипт, возвращаемся к Revit'у и активируем кнопку. Появится окно:

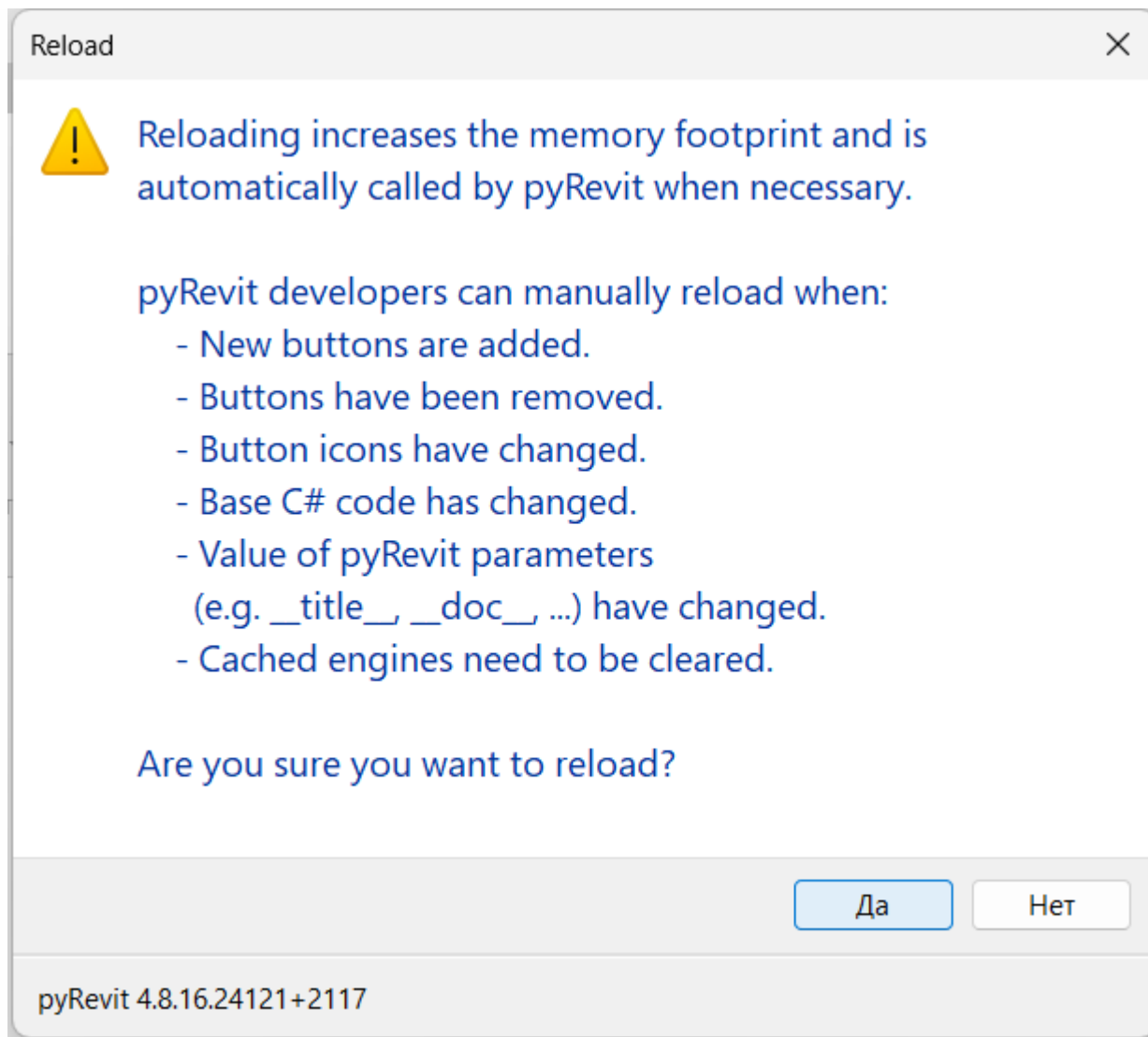


Как видно мы внесли изменения в файл скрипта и он отработал согласно последним изменениям в нем.

То есть, при изменении самого кода перезапускать панель не надо, но вот чтобы отобразить новые кнопки или изменения в оформлении(строчки 1-4) необходимо произвести перезагрузку панелей пая.

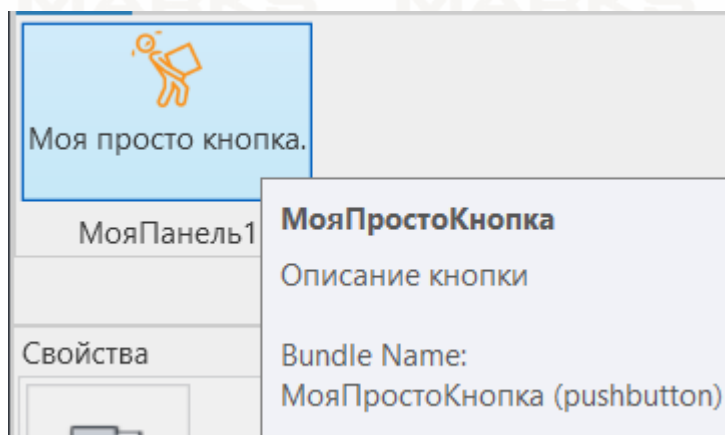
Для этого на вкладке pyRevit есть кнопка “Обновить”





Нажимаем “Да” и ждем обновления панелей.

Теперь мы видим, что имя кнопки сменилось на то, которое мы указали внутри самого скрипта и добавилось описание.



Наша первая полностью функционирующая кнопка готова.

Код тут: gist.github.com

1.6. Типы кнопок

pyRevit

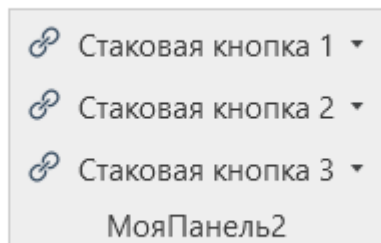
Какие еще бывают кнопки в Revit

В [предыдущей статье](#) мы разбирали как создать кнопку типа `.pushbutton`, но на этом функционал Revit и pyRevit не заканчивается.

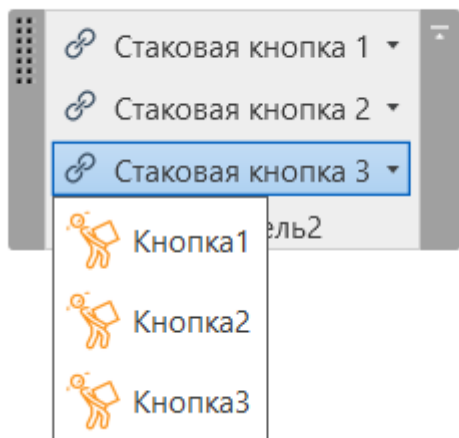
Давайте посмотрим, что они еще могут.

1.6.1. Выпадающий список и стек (`.pulldown` и `.stack`)

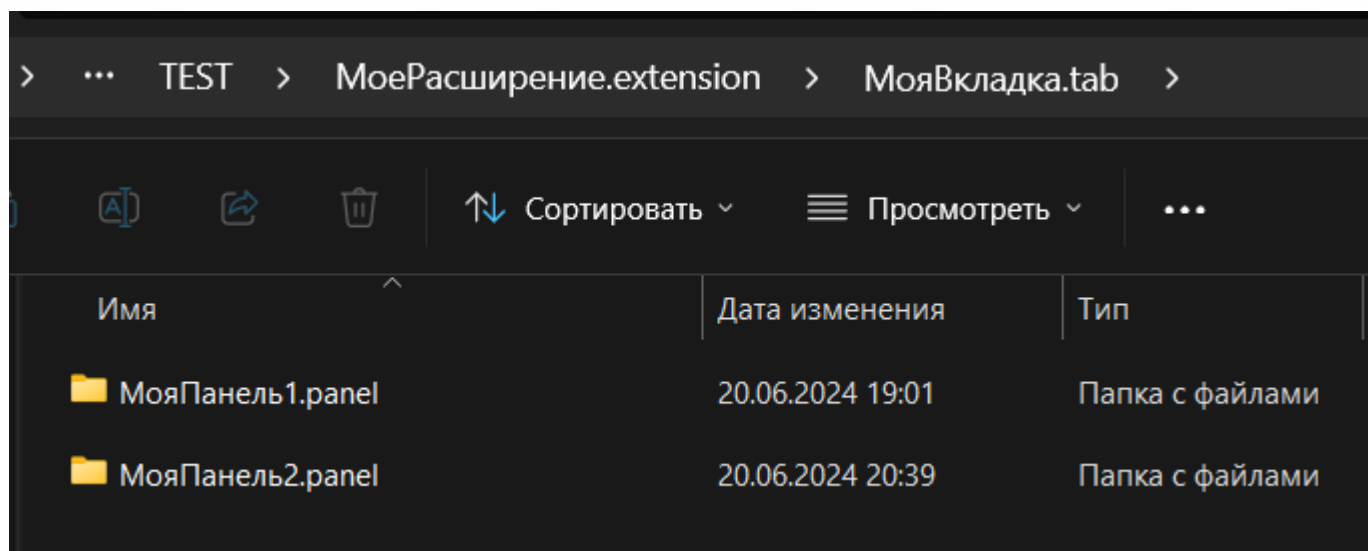
Стак выглядит так:



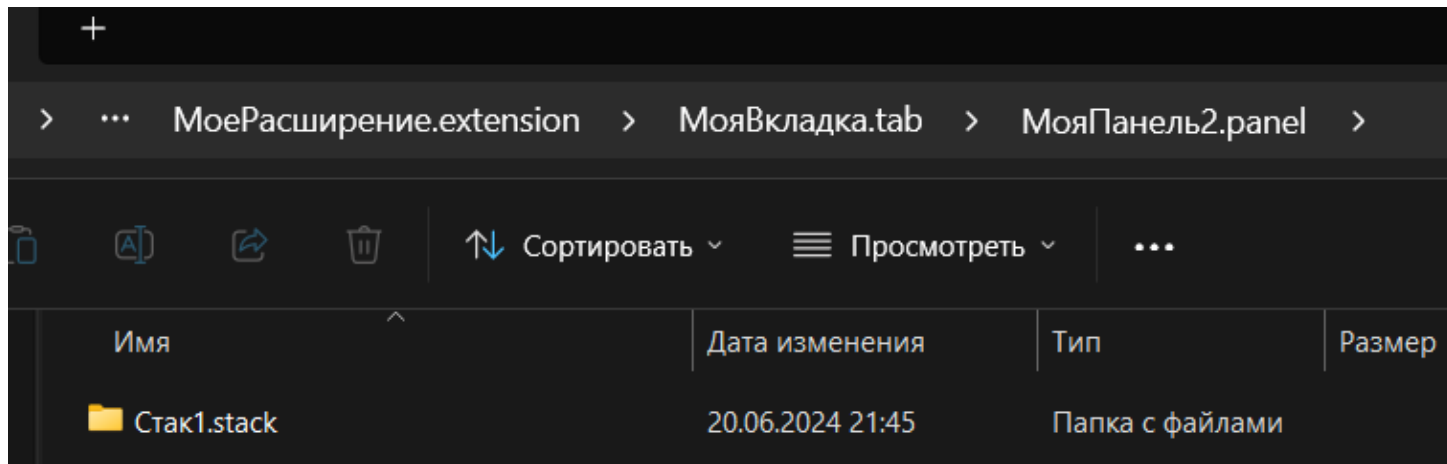
Выпадающий список выглядит так:



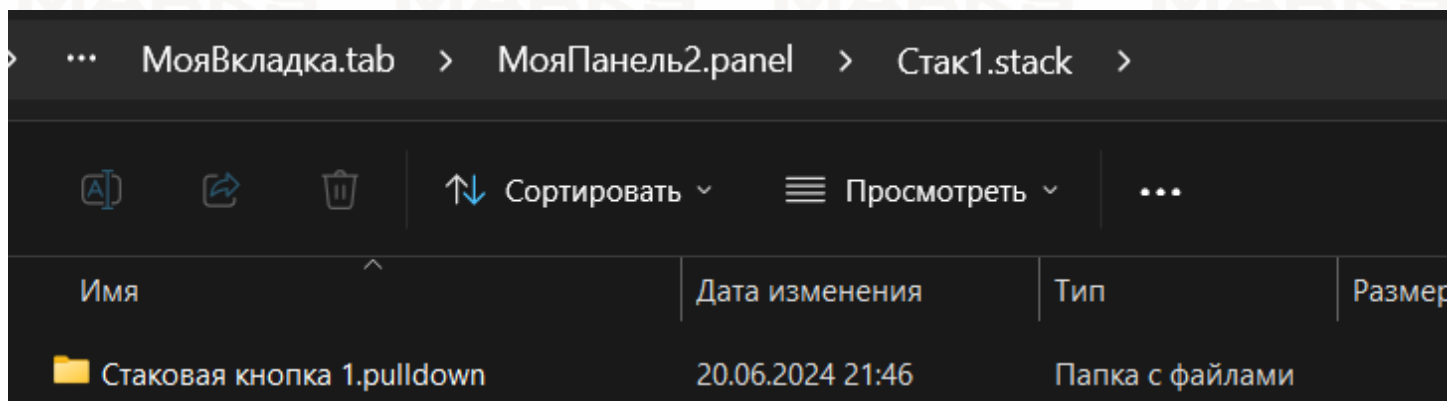
Для этого в папке "МояВкладка.tab" создадим папку "МояПанель2.panel"



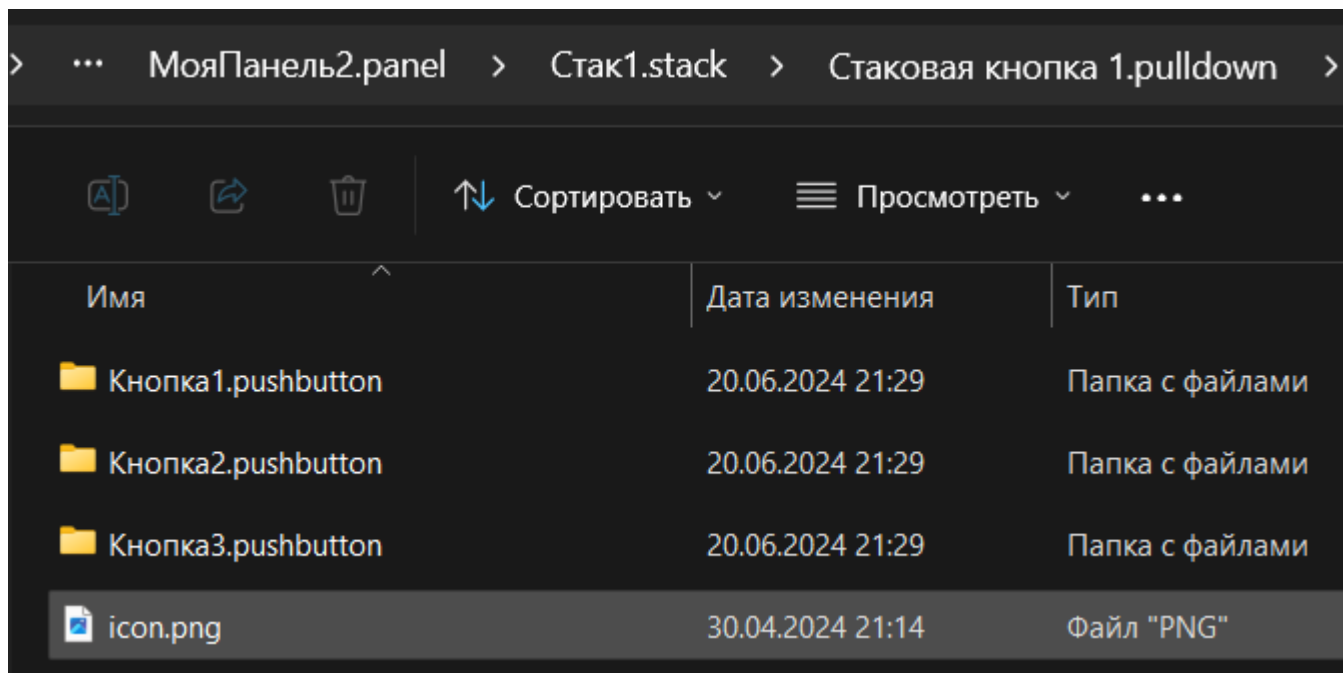
В нее поместим папку с именем "Стак1.stack"



А в нее поместим “Стаковая кнопка 1.pulldown”



Внутри папки .pulldown расположим 3 папки с расширением .pushbutton (в .pushbutton закидываем script.py и icon.png)



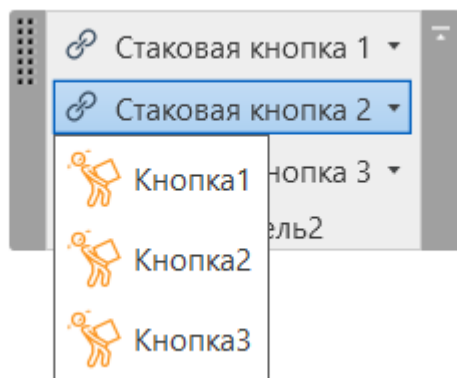
После чего скопируем .pulldown 2 раза поменяв названия.

МояВкладка.tab			МояПанель2.panel			Стак1.stack		
Имя			Дата изменения			Тип		
Стаковая кнопка 1.pulldown			20.06.2024 21:46			Папка с файлами		
Стаковая кнопка 2.pulldown			20.06.2024 21:46			Папка с файлами		
Стаковая кнопка 3.pulldown			20.06.2024 21:45			Папка с файлами		

Должна получится такая структура:

- МояВкладка.tab
 - МояПанель2.panel
 - Стак1.stack
 - Стаковая кнопка 1.pulldown
 - Стаковая кнопка 2.pulldown
 - Стаковая кнопка 3.pulldown
 - Кнопка1.pushbutton
 - script.py
 - icon.png
- Кнопка2.pushbutton
 - Кнопка3.pushbutton
 - icon.png

Перезапускаем панель нажав “Обновить” и вауля:



Попробую еще раз но другими слова для закрепления.

Сначала мы создали `.stack` сказав паю что надо расплагать все что будет внутри друг под другом - типа таблицей. Максимальная высота стака - 3.

А "таблица" состоин из выпадающих списков, и что бы пай понял, что это "выпадающий" мы задаем в имени папки на конце `.pulldown`

В основном я пользуюсь только 3 типа кнопок:

- `stack`
- `pulldown`
- `pushbutton`

и просто по разному комбинирую их.

1.6.2. А какие еще могут быть кнопки?

Вообще, вы могли заметить, что по сути тип кнопки не поменялся, как был `pushbutton` так и остался. Поменялась структура, в которой расположена кнопка.

Но в пай действительно можно создать другой тип кнопки. Описывать подробно в рамках статей я не буду, но дам заправку:

- .smartbutton
- .nobutton
- .splitpushbutton
- .splitbutton
- .urlbutton

подробнее [тут](#)