

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 строки - 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.

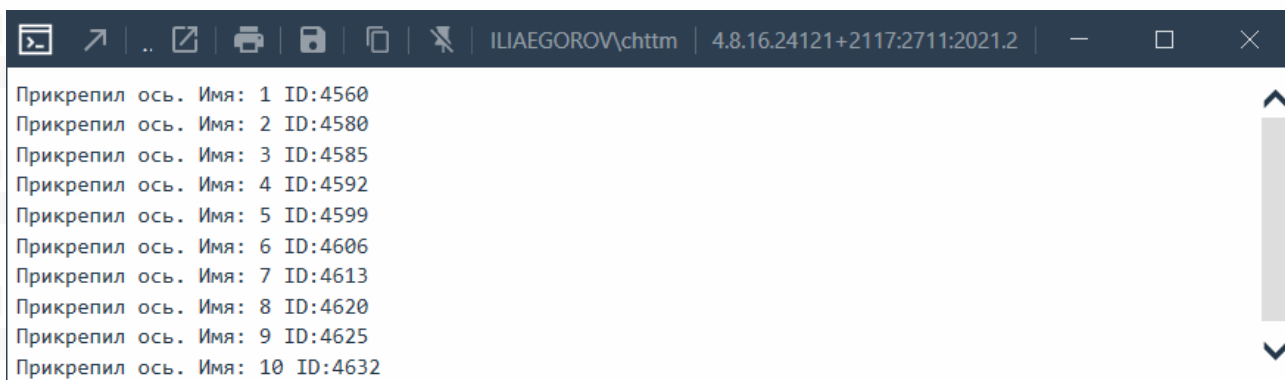
.ToElements(): Преобразует отфильтрованную коллекцию в список элементов.

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.

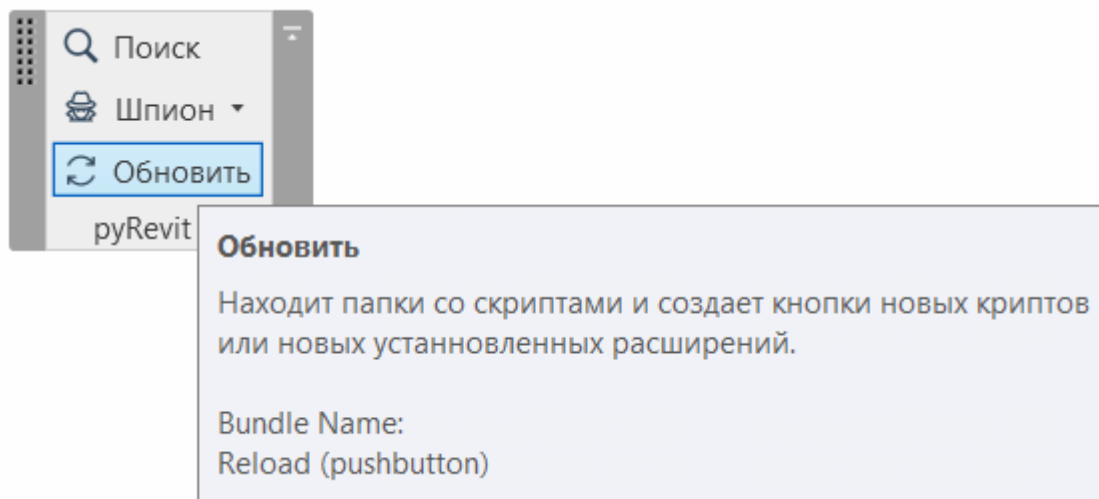
Сохраняем скрипт, возвращаемся к Revit'у и активируем кнопку. Появится окно:

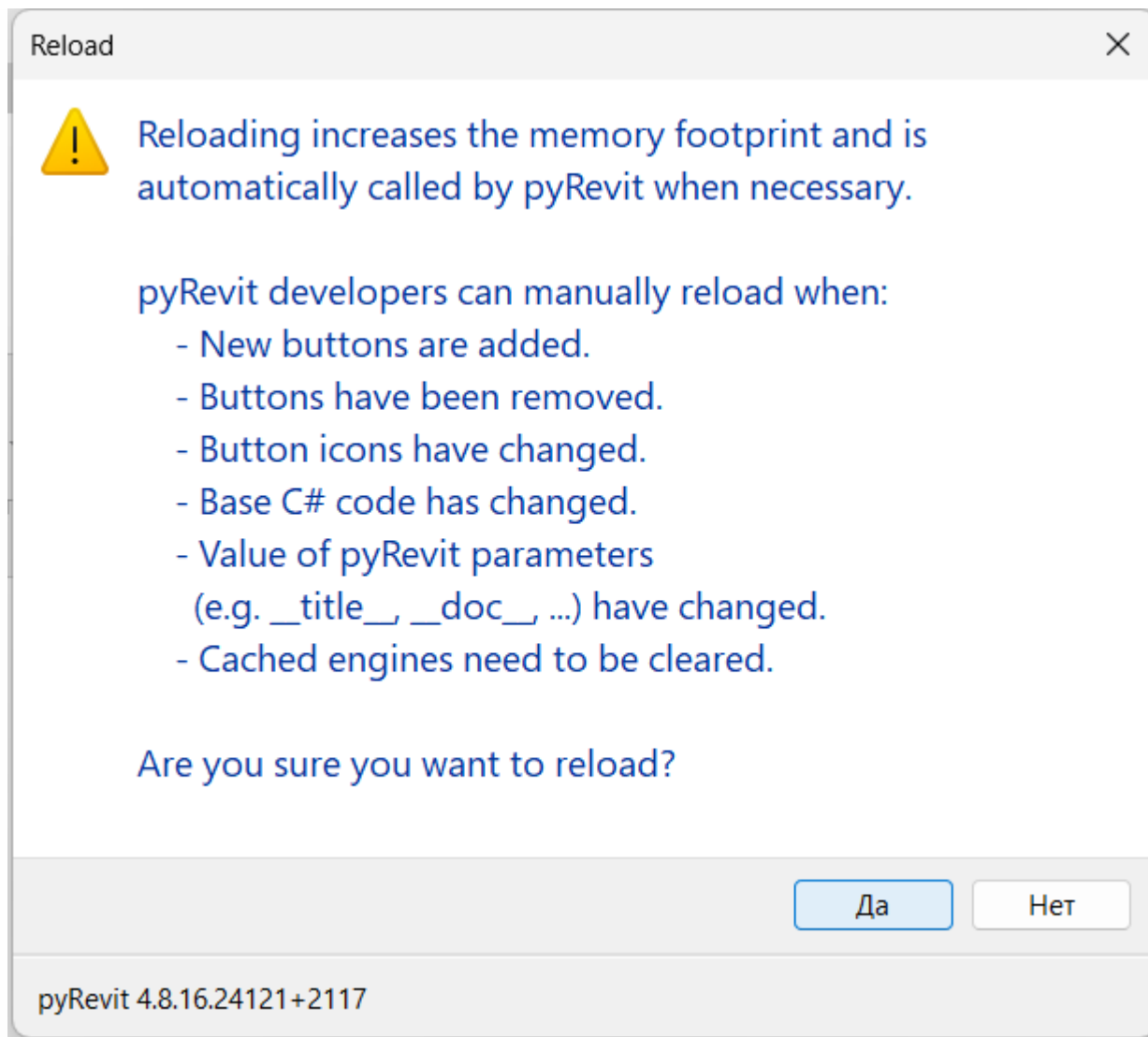


Как видно мы внесли изменения в файл скрипта и он отработал согласно последним изменениям в нем.

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.

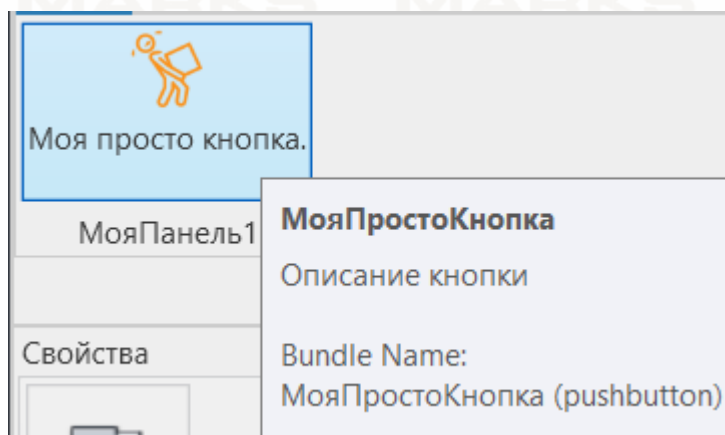
Для этого на вкладке pyRevit есть кнопка “Обновить”





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](#)

MARKS GROUP