

El Poder de la Automatización: Python en Dynamo para Revit y Civil 3D

Raquel Bascones Recio

Senior Implementation Consultant, Autodesk

David Licona

Implementation Consultant, Autodesk

Ponentes



Raquel Bascones Recio
Senior Implementation Consultant

2010 – 2017: Architect & Landscape Architect

2017 – 2019: Autodesk Global Product Support

Desde 2019: Autodesk Consulting

raquel.bascones.recio@autodesk.com

[LinkedIn](#)



David Licona
Implementation Consultant

2014 – 2017: Project Engineer – Construction

2017 – 2020: Project Manager – Rail infrastructure

Desde 2020: Autodesk Consulting

david.licona@autodesk.com

[LinkedIn](#)

Objetivos de aprendizaje

- Descubrir la programación orientada a objetos
- Aprender las bases y elementos de lenguaje de Python
- Diferenciar entre Python, IronPython y Cpython
- Comprender las bases de las API de Revit, AutoCAD y Civil 3D



Python en Dynamo

¿Por qué Python?

Python en Dynamo

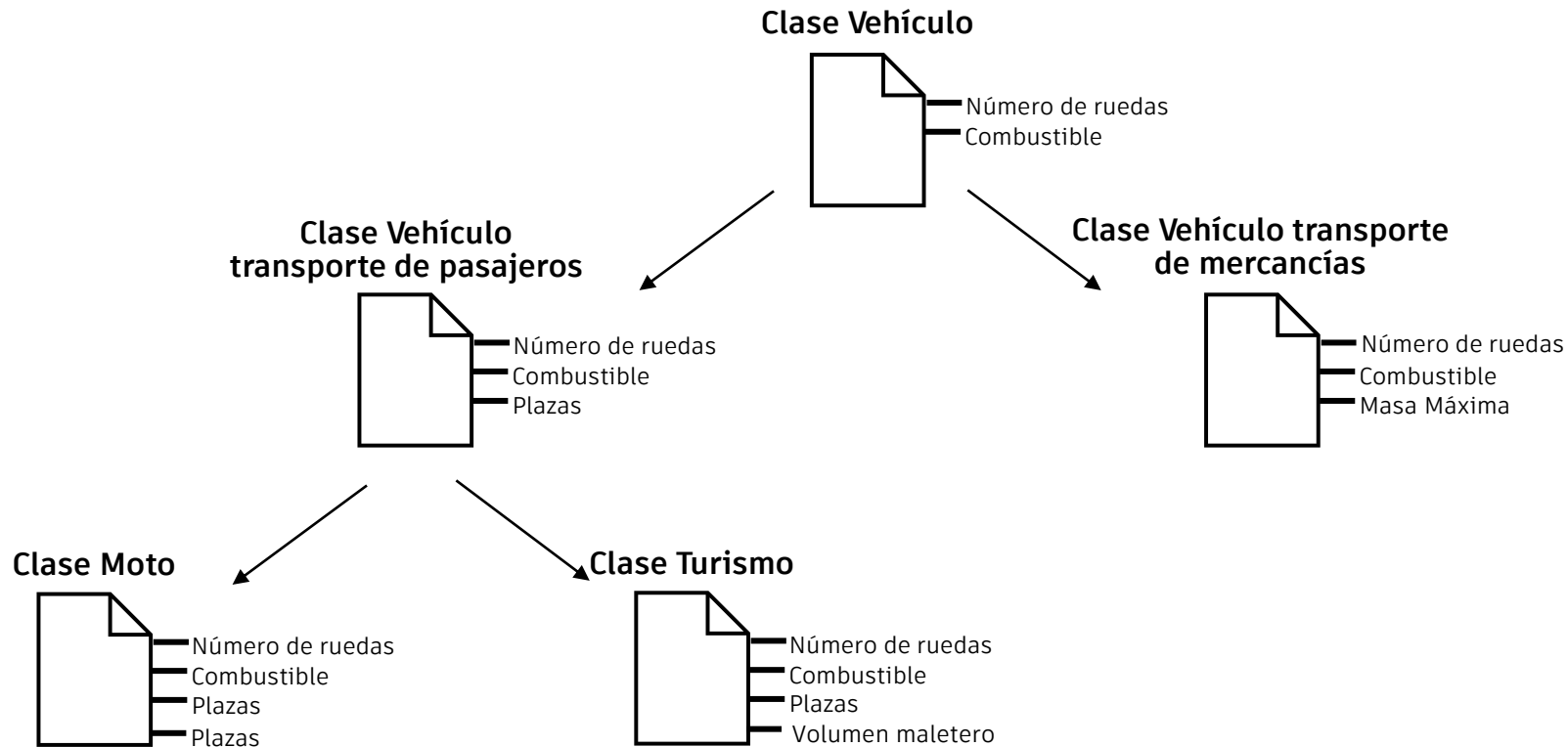
Conceptos clave

- Uno de los lenguajes de programación más usados en la actualidad
- Creado en 1980s por Guido van Rossum

✓ Ventajas	✗ Inconvenientes
Fácil de entender	No diseñado para ser <i>compiled</i>
Gran comunidad de usuarios y módulos	Más lento que otros lenguajes
Flexible	<i>Debugging</i> es lento y tedioso

Python en Dynamo

Programación orientada a objetos



Python en Dynamo

Implementaciones



IronPython

- Focalizado en .NET
- Originalmente creado por Microsoft, actualmente mantenido por voluntarios
- Utilizado por Dynamo desde sus inicios
- Usa Python 2.7, actualmente sin soporte

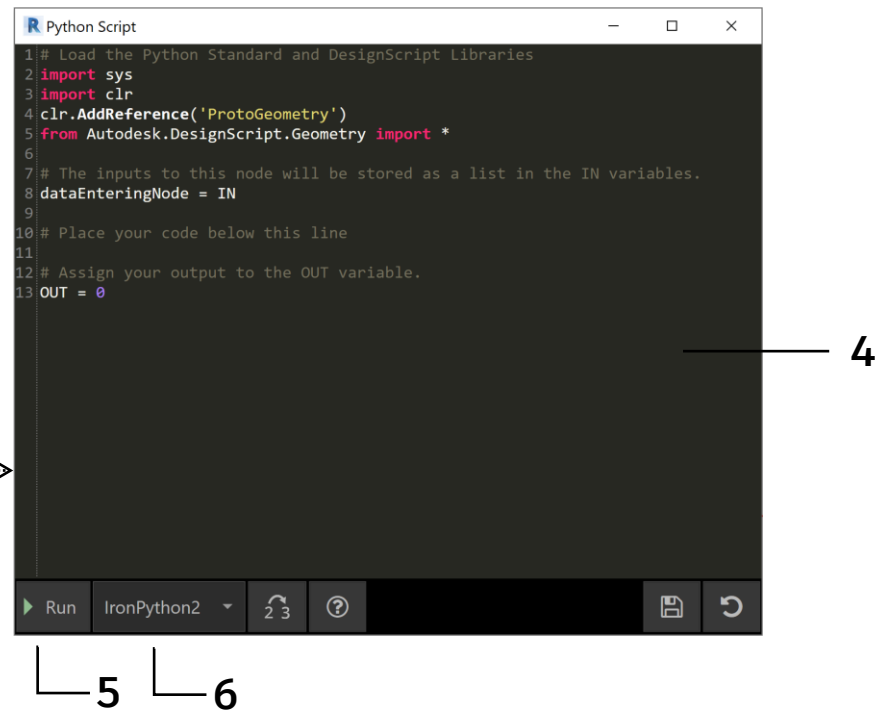
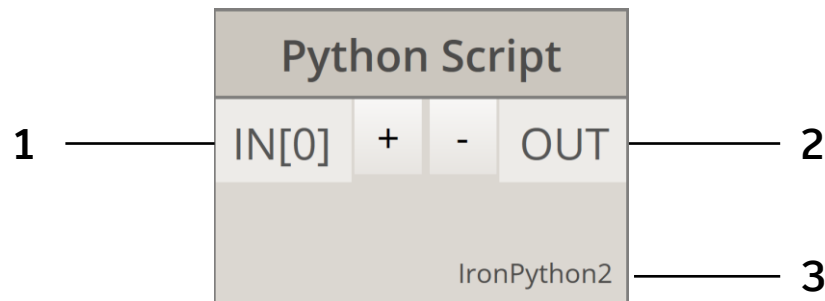


CPython

- Implementación oficial de Python
- Presente en Dynamo desde la version 2.7
- Actualizado a las últimas versiones de Python
- Permite importar módulos y librerías como NumPy, Panda, etc.

Python en Dynamo

Nodo



Python en Dynamo

Uso de un editor externo

- Ventajas
 - Autocompletado de funciones nativas de Python
 - Revisión de sintaxis
 - Colapsado de bloques
 - Reconocimiento de variables
- Se puede copiar y pegar al editor de Dynamo o ejecutar directamente el archivo py en Dynamo
- Multitud de editores gratuitos y de código abierto

```
19 c = True
20 d = [2, 'happy', 4.562, False]
21 e = {'key 1': 1, 'key 2': 'this is the value 2', 'key 3': [1,2,3]}
22 f = set([1,2,3,3])
23 g = (1,2,3)
24 MessageBox.Show('Variable Types\nstr: {0}\nint: {1}\nbool: {2}\nlist: {3}\ndictionary: {4}\nset: {5}\ntuple: {6}'.format(a,b,c,d,e,f,g))
25
26 # Variable conversions
27 h = str(234)
28 i = int('15')
29 j = round(345.65321, 2)
30 k = "hello,hello,hello".split(',')
31 l = " ".join(['today', 'is', 'a', 'good', 'day'])
32 MessageBox.Show('Variable conversions:\nto string str(234): {0}\nto integer int("15"): {1}\nRound float round(345.65321, 2): {2}\nSplit string "hello,hello,hello": {3}\nJoin ["today", "is", "a", "good", "day"] {4}')
33
34 # Working with Lists
35 m = [10, 20, 30, 40, 50]
36 n = m[0]
37 o = m[-1]
38 p = m[2: 4]
39 q = m[:-1]
40 r = m[::2]
41 MessageBox.Show('Working with Lists\nOriginal list: {0}\nIndexing m[0]: {1}\nIndexing negative values m[-1]: {2}\nSlicing m[2 : 4]: {3}\nm[::2]: {4}')
42
43 # Unmutable objects
44 x = 'Hello'
45 y = x
46 MessageBox.Show('Unmutable Objects - Step 1\nx: {0}\ny: {1}'.format(x, y))
47
48 y += ' World!'
49 # What is the value of x?
50 MessageBox.Show('Unmutable Objects - Step 2\nx: {0}\ny: {1}'.format(x, y))
```

Python en Dynamo

Cheat Sheet

Indexing lists

Working with values from lists

Value at index	<code>x = ["a", "b", "c"] x[0] = "a"</code>
Index counting from end	<code>x = ["a", "b", "c"] x[-1] = "c"</code>
Remove value	<code>del x[2]</code>
Modify value	<code>x[1] = 25</code>
Length of a list	<code>x = ["a", "b", "c"] len(x) = 3</code>
Slicing lists	<code>list[start slice : end slice : step] x = [1, 2, 3, 4, 5] x[2:] = [3, 4, 5] x[1:3] = [2, 3, 4] x[::1] = [5, 4, 3, 2, 1] x[::2] = [1, 3, 5]</code>

Assignment of variables

- Binding of a name with a value

Unique assignment	<code>x = 1</code>
Multiple assignments	<code>x, y, z = 1, 2, 3</code>
Assignment to the same value	<code>x = y = z = 0</code>
Increment	<code>x += 1</code> equals <code>x = x + 1</code>
Decrement	<code>x -= 1</code> equals <code>x = x - 1</code>
Remove variable	<code>del x</code>
Empty variables	<code>x = None x = "" x = [] x = {} x = set()</code>

Variable conversions

- Change variable type

To integer	<code>int(x)</code>
To float	<code>float(x)</code>
Round float	<code>round(number, #digits)</code>
To str	<code>str(x)</code>
List from string	<code>list("abc") = ["a", "b", "c"] x.split(separator)</code>
String from list	<code>separator.join(list)</code>
Check type	<code>isinstance(x, type)</code>

Main operations on lists

Length of the list	<code>len(x)</code>	Append a value at the end	<code>x.append(a)</code>
Minimum / maximum value	<code>min(x) max(x)</code>	Add a sequence of values at the end	<code>x.extend([a, b, c])</code>
Iterator returning (index, value)	<code>enumerate(x)</code>	Insert item at index	<code>x.insert(index, a)</code>
Iterator returning values at same index of all collections	<code>zip(x, y, z)</code>	Remove first item of value a	<code>x.remove(a)</code>
Evaluate existence/not existence in list, returns bool	<code>a in x a not in x</code>	Remove item at index and return item	<code>x.pop(index)</code>
Index of an item in a list	<code>x.index(a)</code>	Sort the list in place	<code>x.sort()</code>
Count occurrences of an item in a list	<code>x.count(a)</code>	Make a sorted copy of the list	<code>sorted(x)</code>

Lambda Forms

Anonymous functions

- Examples of use

<code>names = ["Jo", "Mia", "Peter", "Aria", "Julian"]</code>	
<code>sorted(names, key = lambda x: len(x))</code>	<code>["Julian", "Peter", "Aria", "Mia", "Jo"]</code>
<code>filter(lambda x: len(x)>3, names)</code>	<code>["Peter", "Aria", "Julian"]</code>
<code>map(lambda x: x + " Parker", names)</code>	<code>["Jo Parker", "Mia Parker", "Peter Parker", "Aria Parker", "Julian Parker"]</code>

Math

- Import math module

Operators	<code>+ - * / % (remainder of division) // (integer value of division) **(pow, a^b)</code>
Absolute value	<code>abs(x)</code>
Round up to integer	<code>math.ceil(x)</code>
Round down to integer	<code>math.floor(x)</code>

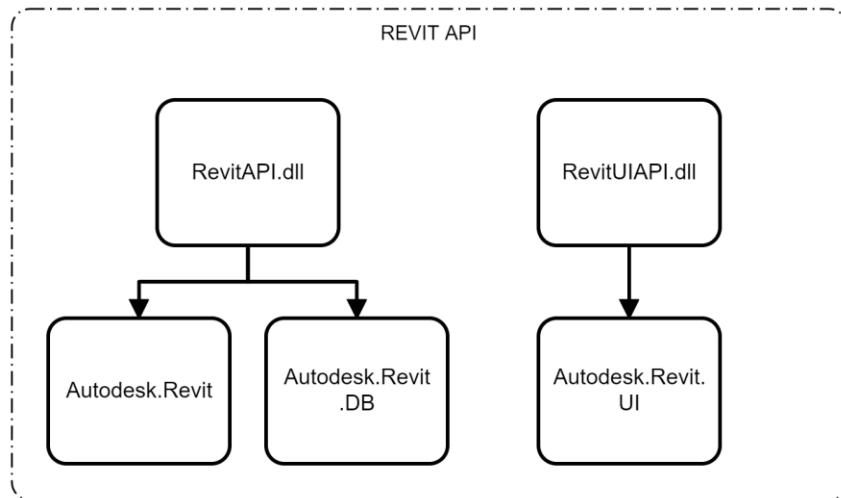
Python y Revit API

¿Qué es Revit API?

Python & Revit API

Namespaces

- Revit API se compone de dos *assemblies* (archivos .dll)
 - RevitUIAPI.dll
 - RevitAPI.dll
- Agrupan objetos e identificadores según funcionalidades
- Evita colisiones de nombres



Python & Revit API

Namespaces

- Añadir el *assembly* usando la libreria *Common Language Runtime (clr)*
 - `clr.AddReference("RevitAPI")`
 - `clr.AddReference("RevitUIAPI")`
- Importar los objetos requeridos
 - Importar todo el *namespace*
 - `from Autodesk.Revit.DB import *`
 - `import Autodesk.Revit.DB`
 - Importar objetos determinados
 - `from Autodesk.Revit.DB import Wall`

```
32 # Import statements
33 import clr
34 import sys
35 # In IronPython it is possible to import references to .NET assemblies like those used to access the Revit API
36 clr.AddReference('RevitAPI')
37 clr.AddReference('RevitAPIUI')
38 clr.AddReference('RevitServices') # this is a Dynamo for Revit assembly
```

Python & Revit API

Documentos



Document

- Archivo Revit (RVT, RFA)
- Tiene propiedades que definen el archive (Title, PathName, etc)
- Métodos para modificar y crear elementos

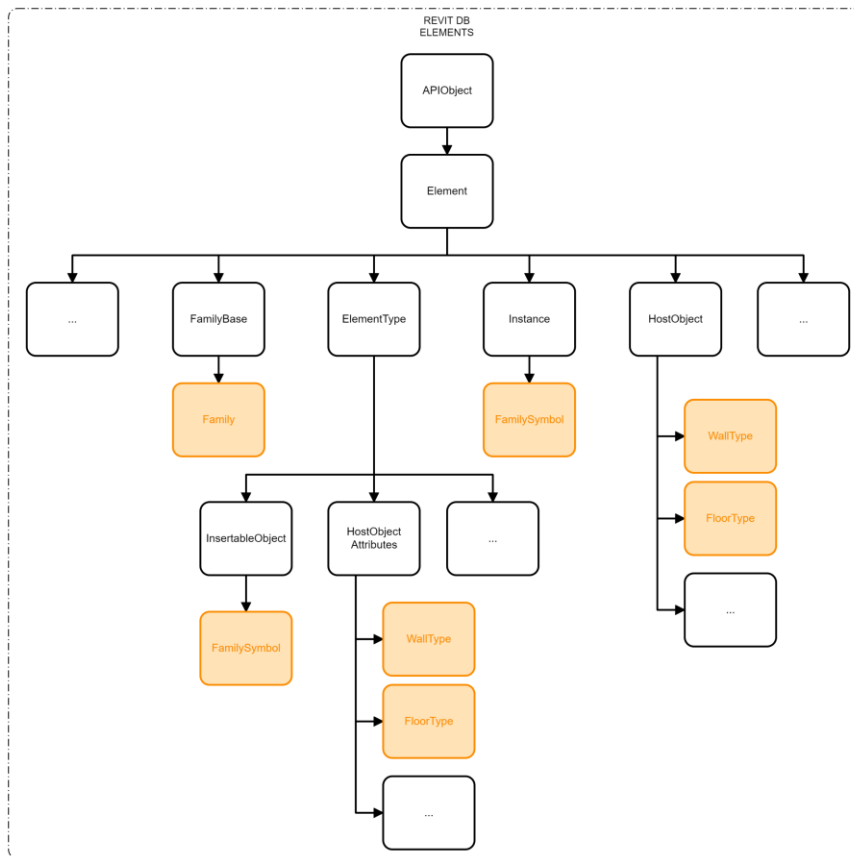
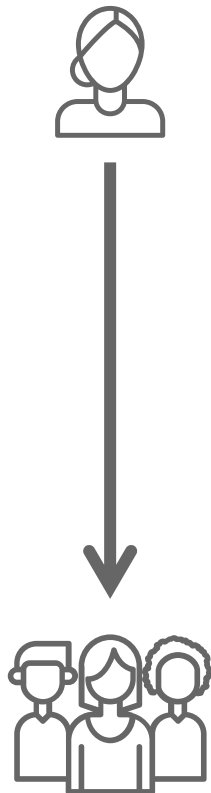


UIDocument

- Proyecto activo presentado al usuario
- Métodos para seleccionar elementos interactivamente
- Permite enviar mensajes al usuario mediante TaskDialog

Python & Revit API

DB Elements



Python & Revit API

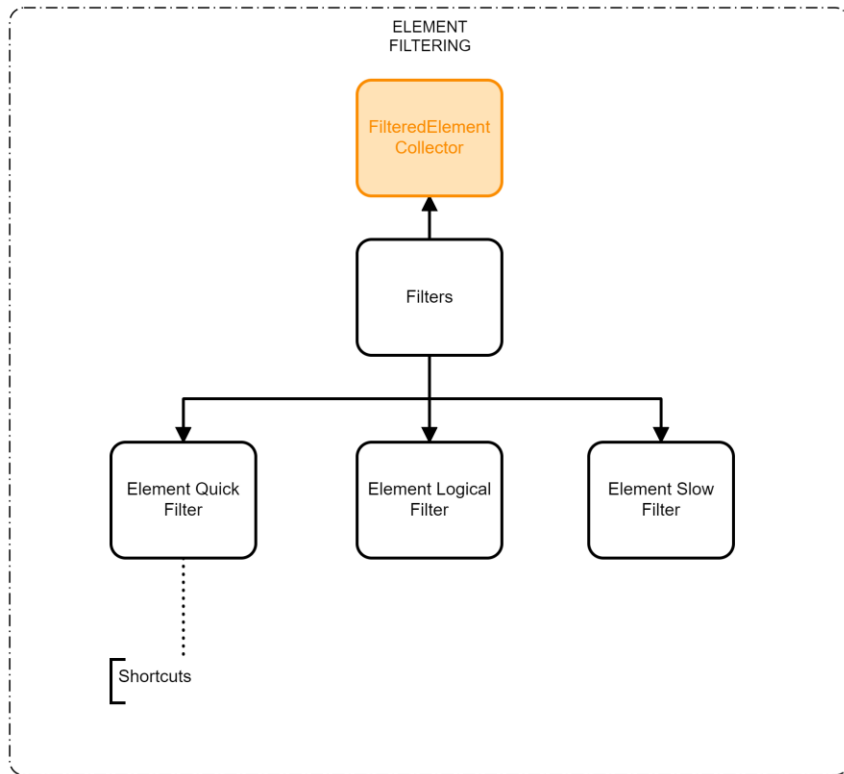
DB Elements

	Familia Sistema	Familia Cargable
Element Type	<i>WallType</i> <i>FloorType</i> ...	<i>FamilySymbol</i> & <i>Category - Doors,</i> <i>Windows...</i>
Instance	<i>Wall</i> <i>Floor</i> ...	<i>FamilyInstance</i> & <i>Category - Doors,</i> <i>Windows...</i>

Python & Revit API

Selección mediante filtros

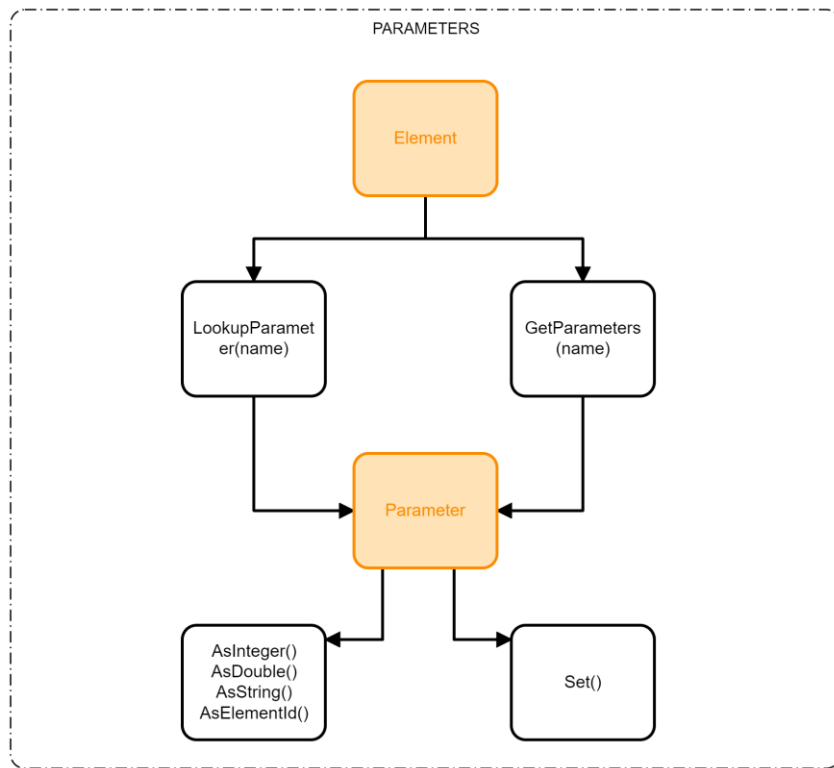
- *FilteredElementCollector(document):*
colección de elementos
- Necesita al menos un filtro
- Shortcuts
 - OfCategory()
 - OfClass()
 - WhereElementIsNotElementType()
 - Excluding()
 - ...
- No hay selección en la interfaz de usuario



Python & Revit API

Parámetros

- Todos los elementos de Revit tienen un set de propiedades
- Parámetros son **objetos** con sus propias propiedades y métodos
 - Unidades
 - Tipo de almacenamiento
 - ...
- Parámetros de tipo y de ocurrencia son el **mismo tipo de objeto**



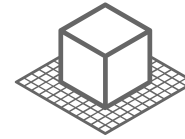
Python & Revit API

Modificación de elementos



Documento

- Métodos a nivel documento
 - Move
 - Rotate
 - Mirror
 - Array



Elemento

- Parámetros
- Location
- Element Type

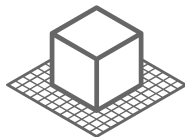
Python & Revit API

Creación de elementos



Proyecto

- *Document.Create*
- Para la creación de
 - Group
 - FamilyInstance
 - DetailLine
 - Floor
 -



Familia

- *Document.FamilyCreate*
- Para la creación de
 - Sweep
 - Extrusion
 - Lofts
 -



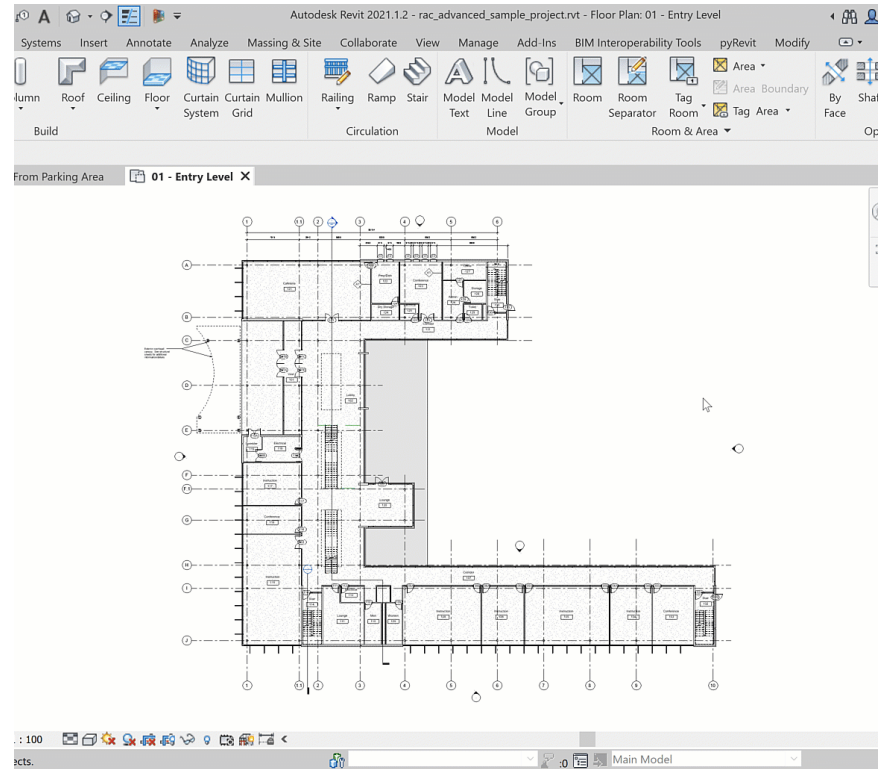
Métodos específicos

- Métodos de creación en la misma clase del objeto
- Para la creación de
 - Wall
 - AdaptiveComponent
 - ...

Python & Revit API

Herramientas de ayuda

- [Revit SDK](#)
- [RevitApiDocs.com](#)
- [LookUp Tool](#)



Python & Revit API

Ejemplos

- Creación de elementos
- Filtrado de elementos por nombre
- Modificación de parámetros

Python y Civil 3D API

Python & AutoCAD/Civil3D API

Plantilla en Dynamo

1

3

```
Python Script
1 # Load the Python Standard and DesignScript Libraries
2 import sys
3 import clr
4
5 # Add Assemblies for AutoCAD and Civil3D
6 clr.AddReference('AcMgd')
7 clr.AddReference('AcCoreMgd')
8 clr.AddReference('AcDbMgd')
9 clr.AddReference('AecBaseMgd')
10 clr.AddReference('AecPropDataMgd')
11 clr.AddReference('AeccDbMgd')
12
13 # Import references from AutoCAD
14 from Autodesk.AutoCAD.Runtime import *
15 from Autodesk.AutoCAD.ApplicationServices import *
16 from Autodesk.AutoCAD.EditorInput import *
17 from Autodesk.AutoCAD.DatabaseServices import *
18 from Autodesk.AutoCAD.Geometry import *
19
20 # Import references from Civil3D
21 from Autodesk.Civil.ApplicationServices import *
22 from Autodesk.Civil.DatabaseServices import *
23
24 # The inputs to this node will be stored as a list in the IN variables.
25 dataEnteringNode = IN
26
27 adoc = Application.DocumentManager.MdiActiveDocument
28 editor = adoc.Editor
29
30 with adoc.LockDocument():
31     with adoc.Database as db:
32
33         with db.TransactionManager.StartTransaction() as t:
34             # Place your code below
35             #
36             #
37
38             # Commit before end transaction
39             #t.Commit()
40             pass
41
42 # Assign your output to the OUT variable.
43 OUT = 0
44
45
```

Run

2

4

Python & AutoCAD/Civil3D API

Assemblies



AutoCAD

Acmgd.dll
Acdbmgd.dll
Accoremgd.dll



AutoCAD Architecture

AecBaseMgd.dll
AecPropDataMgd.dll



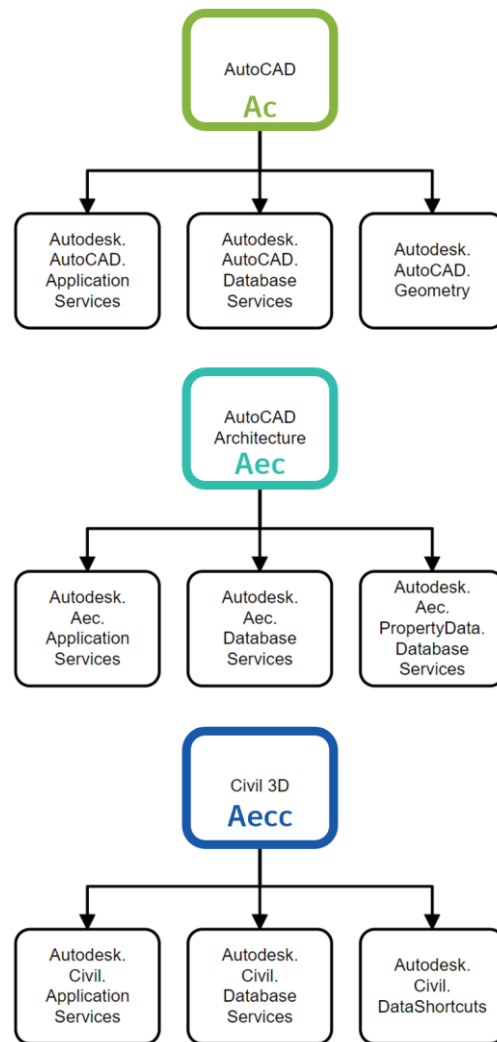
Civil 3D

AeccDbMgd.dll
AeccPressurePipes.dll
AeccDataShortcutMgd

Python & AutoCAD/Civil3D API

Namespaces

- Añadir el *assembly* :
 - `clr.AddReference("AecBaseMgd")`
 - `clr.AddReference("AeccDbMgd")`
- Importar los objetos en el namespace:
 - `from Autodesk.AutoCAD.Geometry import *`
 - `import Autodesk.AutoCAD.ApplicationServices.Application as acapp`



Python & AutoCAD/Civil 3D API

Documentos



AutoCAD Document

- Documento Activo
- Base de datos
- Transaction Manager
- Editor

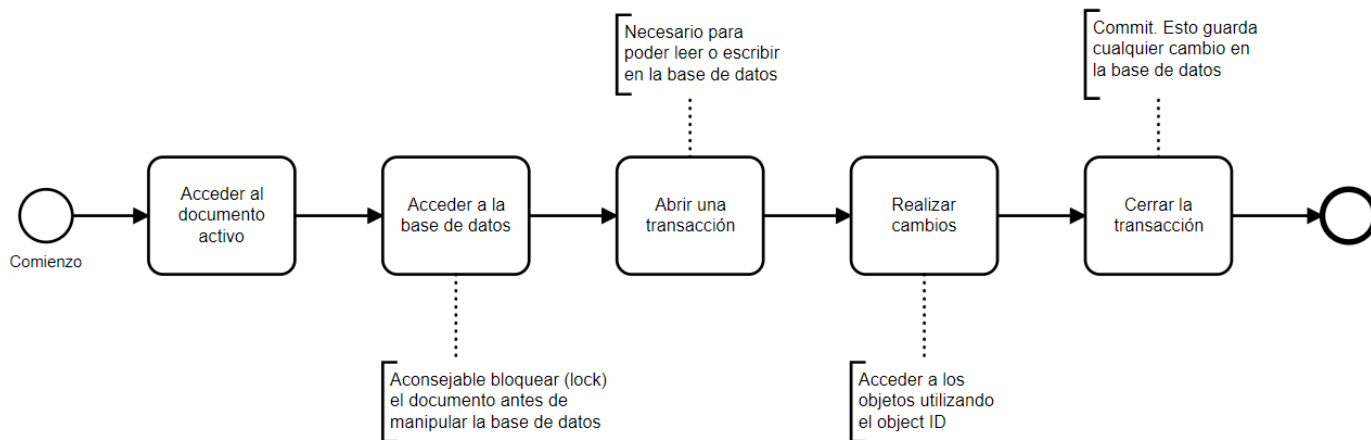


Civil Document

- Entidades Civil 3D:
 - Superficies
 - Corredores...
- Parámetros:
 - Estilos
 - Etiquetas...

Python & AutoCAD API

Transacciones



Python & AutoCAD/Civil 3D API

Objetos



Object ID

- Referencia un **objeto** en la base de datos
- Entero valido solamente para la **sesión activa**
- Utilizado para **interactuar con un objeto**
- Una de sus propiedades es el Object Handle



Object ID Collection

- Referencia a una **colección (grupo) de objetos** en la base de datos
- Las entidades Civil 3D normalmente son colecciones de objetos
- Acceder al objeto dentro de la colección para interactuar con él



Handle

- Referencia a un **objeto** en la base de datos
- Hexadecimal valido para **todas las sesiones** de AutoCAD
- Utilizado para **serializar informacion**

Python & AutoCAD/Civil 3D API

Navegar el API



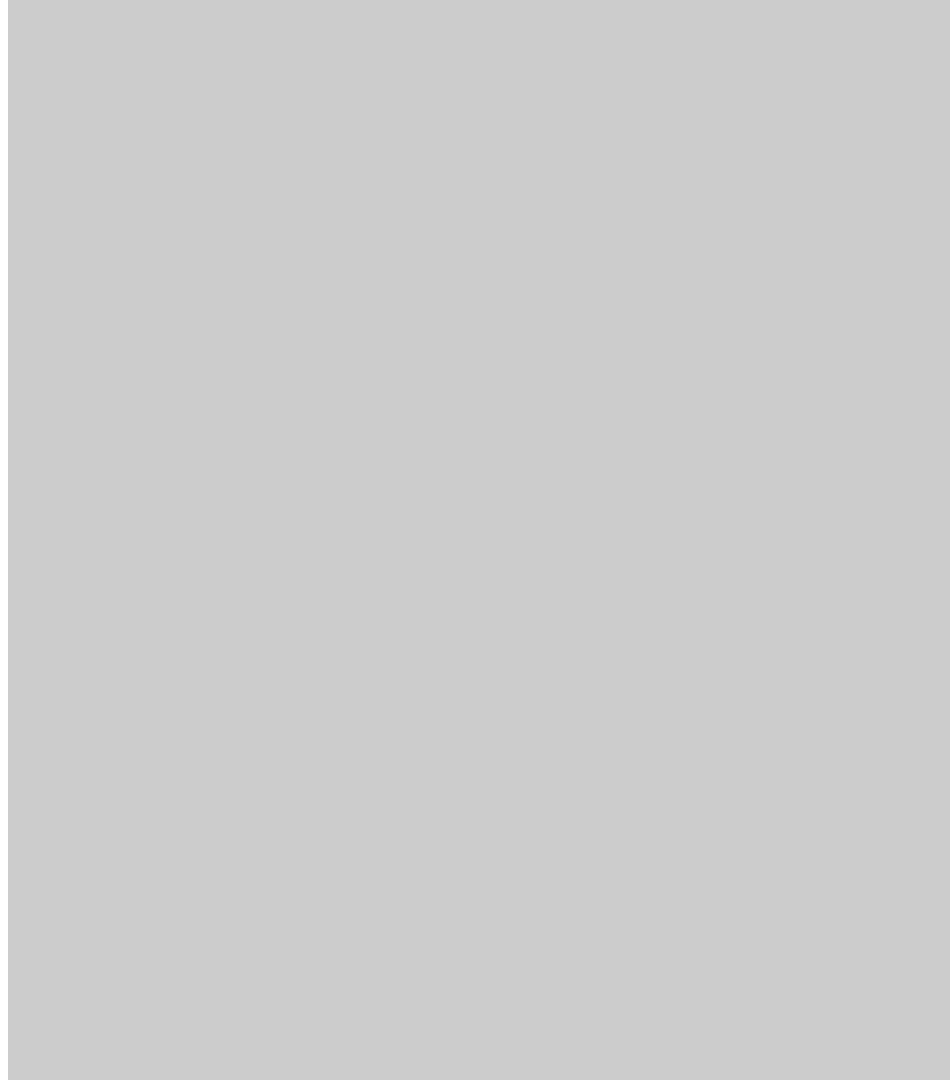
Developer's Guide

- Información sobre el API
- *Ejemplos (en C# generalmente)*



API Reference Guide

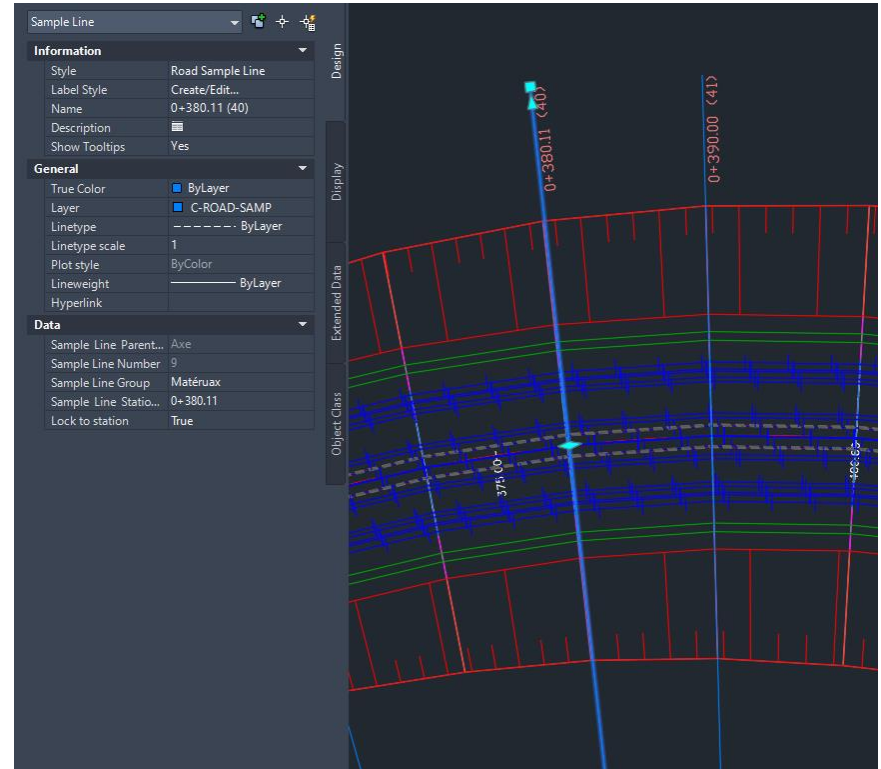
- “Diccionario” para navegar el API
- Clases y miembros de las clases :
 - Métodos
 - Propiedades



Python & AutoCAD/Civil 3D API

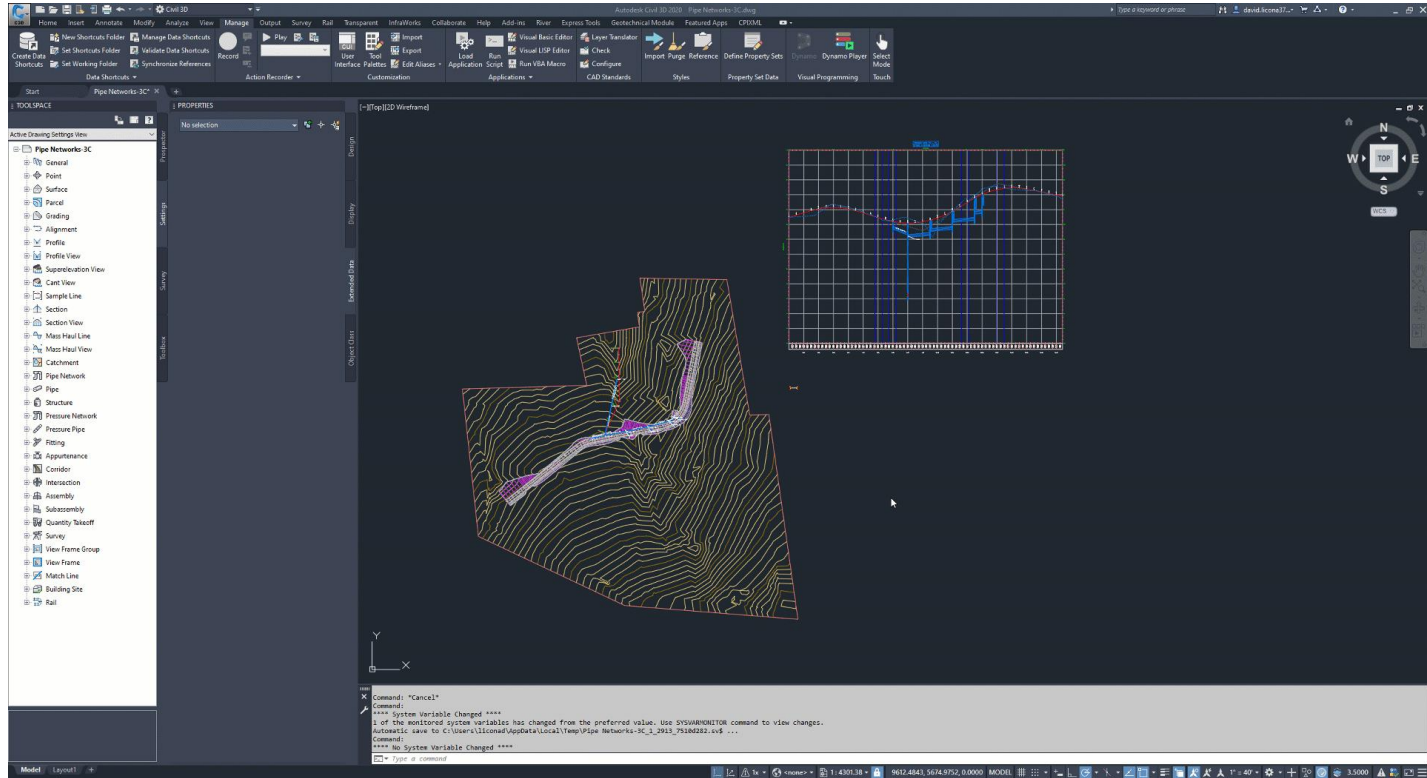
Necesario Python para interactuar con

- *Objetos :*
 - *Sample Lines*
 - *Datashortcuts*
 - *Volume surfaces*
 - *Section Views*
- *Comandos :*
 - *Audit*
 - *View*
 - *Wblock*



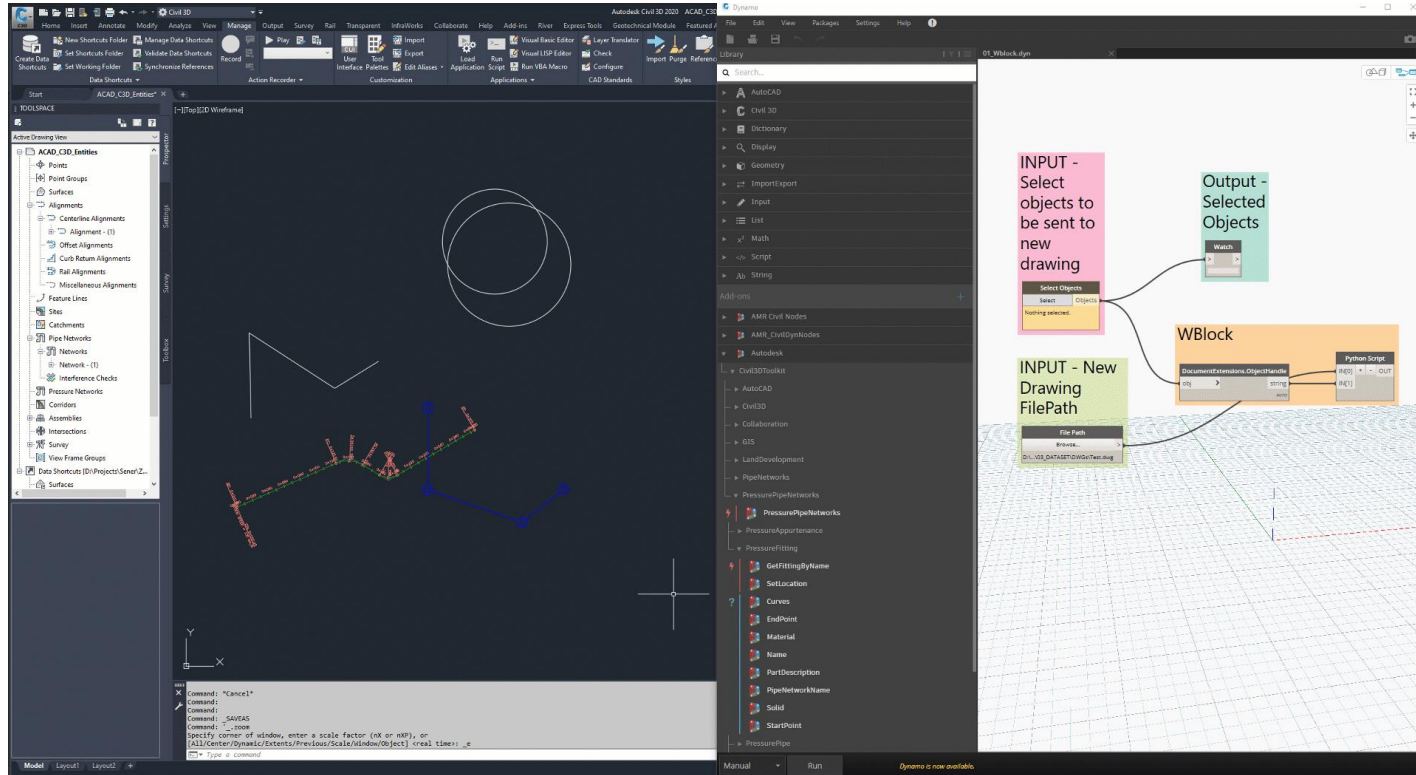
Ejemplo 1

Enviar comandos al command line de Civil 3D



Ejemplo 2

Exportar objetos a un nuevo dibujo



Recursos

Conclusiones

Objetivos de aprendizaje

- Descubrir la programación orientada a objetos
- Aprender las bases y elementos de lenguaje de Python
- Diferenciar entre Python, IronPython y Cpython
- Comprender las bases de las API de Revit, AutoCAD y Civil 3D

Primeros Pasos

Empezando a utilizar Python para resolver problemas



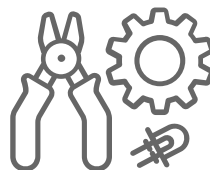
Definición

Identificar problema práctico que necesite el uso de Python



Descripción

Describir el problema (BPMN)



Solución

Implementar la solución



Evaluación

Test y debug

Foro Dynamo

Subtitle

all categories ▾	python ▾	Latest	New (8)	Top	Categories		+ New Topic	
Topic						Replies	Views	Activity
Create section and rotate section in same script ■ Revit dynamo, python						1	58	1h
Numbering every second paragraph ■ Developers dynamo, revit, python						2	54	16h
How can I create an Autodesk.DB.References from XYZ or DesignScript Point dynamo, revit, python						2	37	22h
How to convert FamilyInstances to Autodesk.revit.db.element type? dynamo, revit, python						2	52	1d
Retrieve dimension types in revit ■ Developers revit, python, api						5	112	1d
Get PipeInsulation center reference ■ Developers dynamo, revit, python, geometry, mep						1	37	1d
How can I draw a viewport from a polyline? ■ Civil3D dynamo, python, geometry						5	126	1d
Get Element from FamilyInstance ■ Revit dynamo, python						3	202	1d
☑ Lists: Different results for build-in list-node and regex-node? ■ Lists-Logic python, civil3d						2	41	2d
☑ Python condition in Dynamo ■ Revit python						2	110	2d

 **mzjensen** Regular

 lebinhx3c2012

Here's the Python script.
IN[0] = Profile View object
IN[1] = station (or list of stations)
IN[2] = elevation (or list of elevations)

The number of stations should match the number of elevations.

```
import sys
import clr

clr.AddReference('AcMgd')
clr.AddReference('AcDbMgd')
clr.AddReference('AeccDbMgd')
clr.AddReference('ProtoGeometry')

from Autodesk.AutoCAD.ApplicationServices import *
from Autodesk.AutoCAD.DatabaseServices import *

from Autodesk.Civil.DatabaseServices import *

adoc = Application.DocumentManager.MdiActiveDocument

from Autodesk.DesignScript.Geometry import *

def get_profile_view_points(profileView, station, elevation):

    global adoc

    output = []

    if not isinstance(station, list) and not isinstance(elevation, list):
```

 **Solution** 4     Reply

The background is black with four abstract, metallic-looking geometric shapes in the corners. These shapes are composed of flat planes and sharp edges, reflecting light to create highlights and shadows. They appear to be fragments of a larger, complex structure.

AUTODESK UNIVERSITY

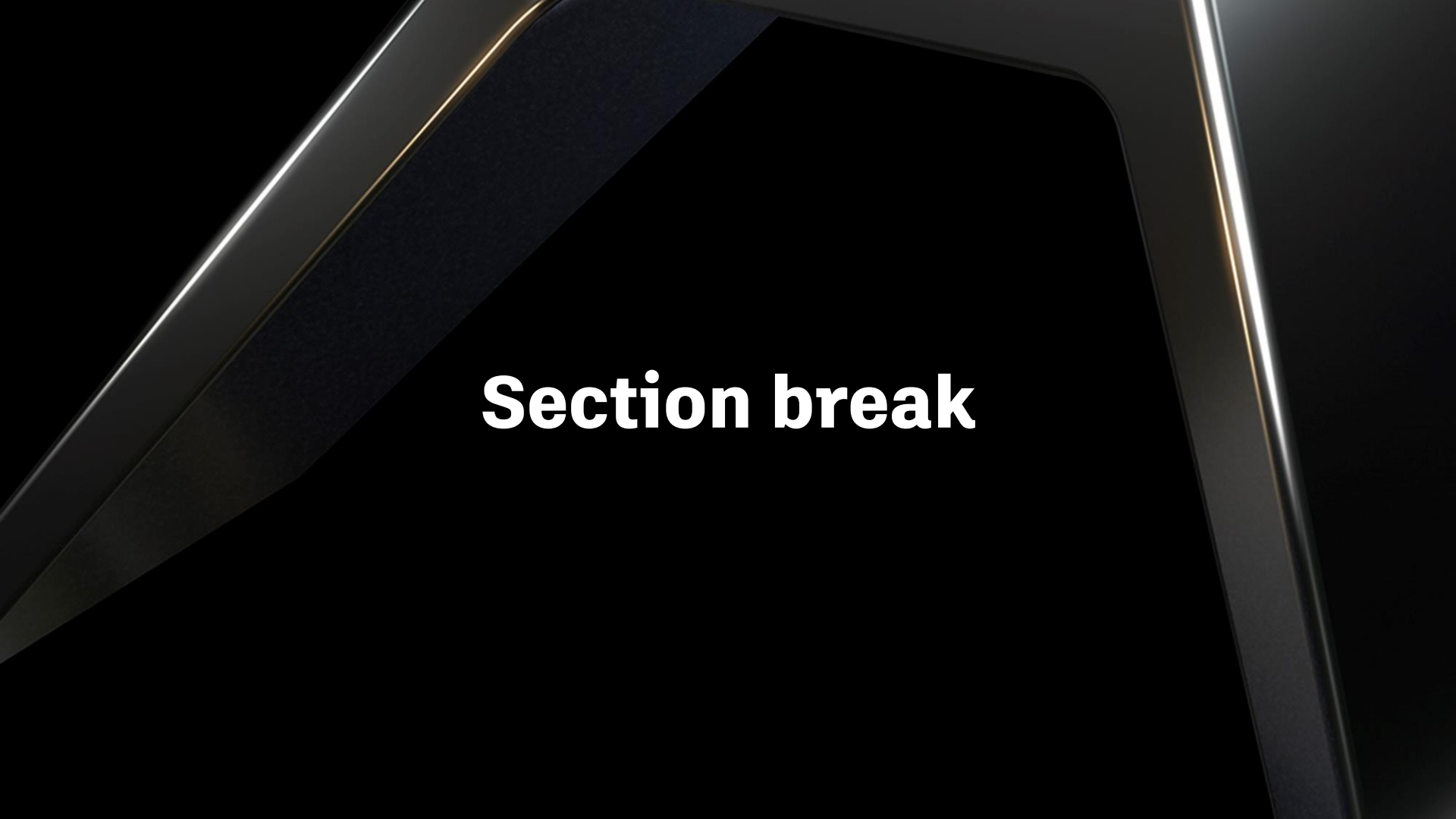
Autodesk and the Autodesk logo are registered trademarks or trademarks of Autodesk, Inc., and/or its subsidiaries and/or affiliates in the USA and/or other countries. All other brand names, product names, or trademarks belong to their respective holders. Autodesk reserves the right to alter product offerings, specifications and pricing at any time without notice, and is not responsible for typographical or graphical errors that may appear in this document.

© 2021 Autodesk. All rights reserved.

Section break



Section break



Section break

Sample blank slide – title (sentence case)

Sample blank slide – subtitle (sentence case)

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat.

- Consectetuer adipiscing elit
- Maecenas porttitor congue massa
- Fusce posuere, magna sed pulvinar ultricies

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat.

- Consectetuer adipiscing elit
- Maecenas porttitor congue massa
- Fusce posuere, magna sed pulvinar ultricies

Sample 2-column content slide

Subtitle

Lorem ipsum dolor sit amet

Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut

- Consectetuer adipiscing elit
- Maecenas porttitor congue massa
- Fusce posuere, magna sed pulvinar

Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut

Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut

Lorem ipsum dolor sit amet

Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut

- Consectetuer adipiscing elit
- Maecenas porttitor congue massa
- Fusce posuere, magna sed pulvinar

Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut

Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut

Sample 3-column slide

Subtitle

Lorem ipsum dolor sit amet



Ut enim ad minim
veniam, quis nostrud
exercitation ullamco
laboris nisi ut aliquip ex



Ut enim ad minim
veniam, quis nostrud
exercitation ullamco
laboris nisi ut aliquip ex



Ut enim ad minim
veniam, quis nostrud
exercitation ullamco
laboris nisi ut aliquip ex

Stats & figures sample slide

Subtitle

90

WRITE HERE

Ut enim ad
minim veniam,
quis nostrud

35

WRITE HERE

Ut enim ad
minim veniam,
quis nostrud

70

WRITE HERE

Ut enim ad
minim veniam,
quis nostrud

90

WRITE HERE

Ut enim ad
minim veniam,
quis nostrud

Text keyword

A man and a woman in business attire are looking at a tablet together in a modern office setting. The man is wearing a dark blue blazer over a light blue checkered shirt, and the woman is wearing a white shirt with black vertical stripes and dots. They are standing in front of a red metal railing. The background shows a large window and some office equipment.

Image keyword

Photo slide

Subtitle

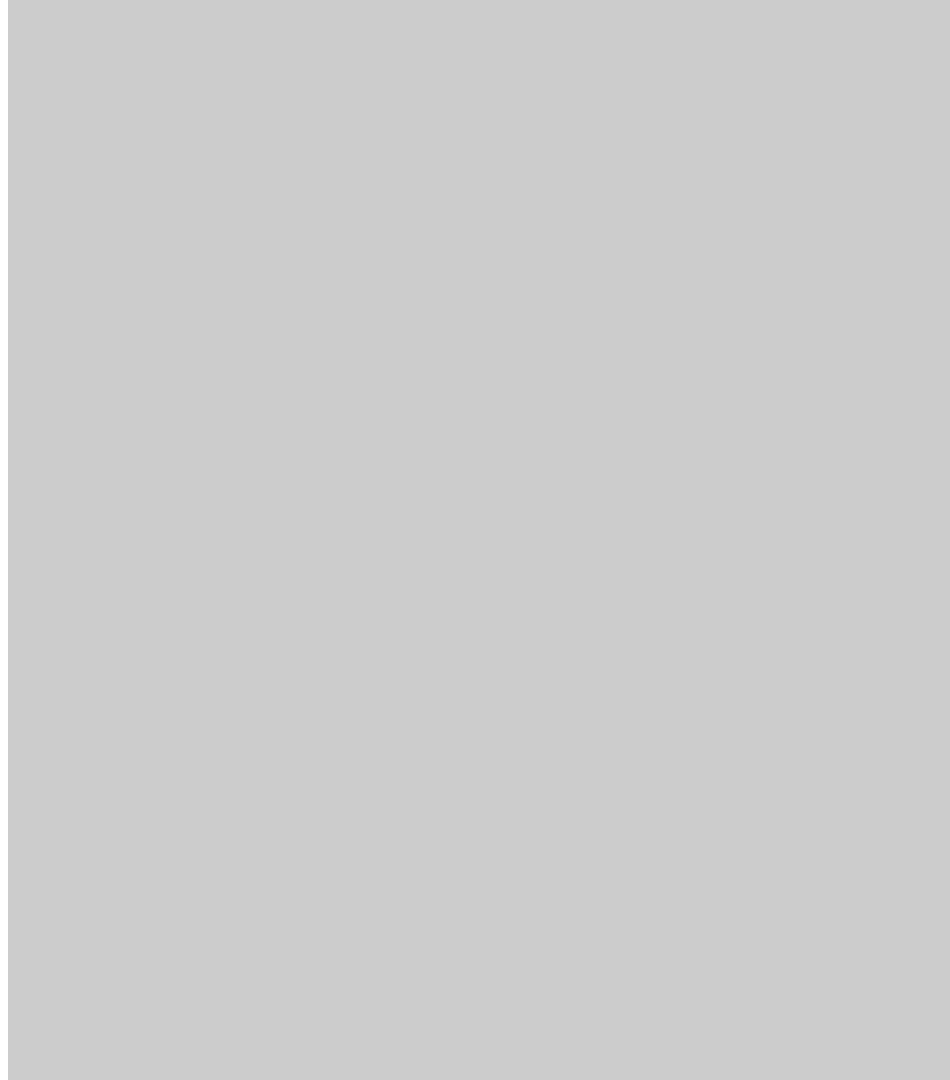
**Lorem ipsum dolor sit amet, consectetur
adipiscing elit, sed do eiusmod tempor
incididunt ut labore et dolore magna aliqua.**

Ut enim ad minim veniam, quis nostrud
exercitation ullamco laboris nisi ut aliquip ex ea
commodo consequat.

Two Photo slide

Subtitle





The background is black with four abstract, metallic-looking geometric shapes in the corners. These shapes are composed of flat planes and sharp edges, reflecting light to create highlights and shadows. They appear to be fragments of a larger, complex structure.

AUTODESK UNIVERSITY

Autodesk and the Autodesk logo are registered trademarks or trademarks of Autodesk, Inc., and/or its subsidiaries and/or affiliates in the USA and/or other countries. All other brand names, product names, or trademarks belong to their respective holders. Autodesk reserves the right to alter product offerings, specifications and pricing at any time without notice, and is not responsible for typographical or graphical errors that may appear in this document.

© 2021 Autodesk. All rights reserved.