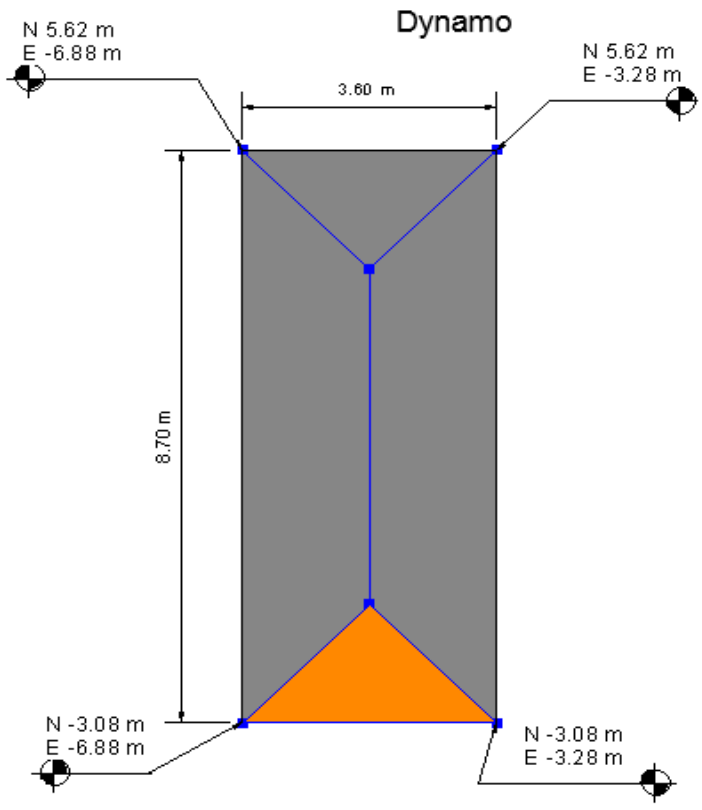
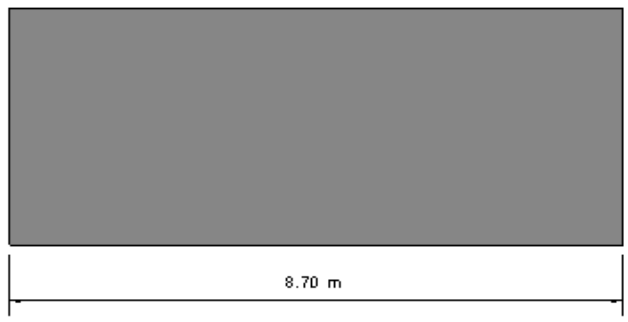
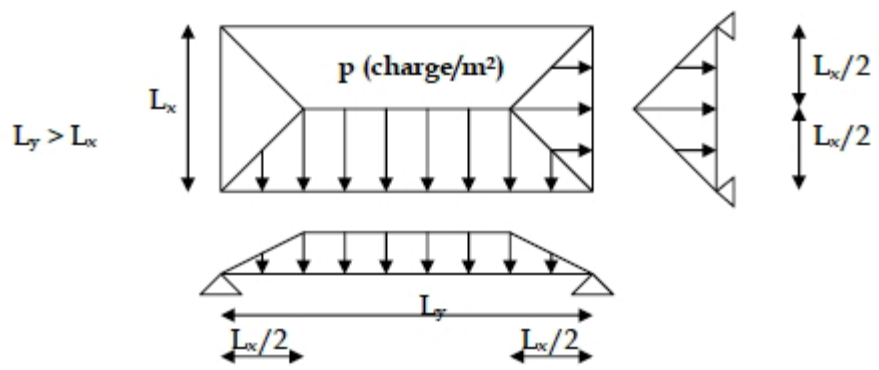
Aide à l'introduction PYTHON pour un Programme Dynamo pour REVIT



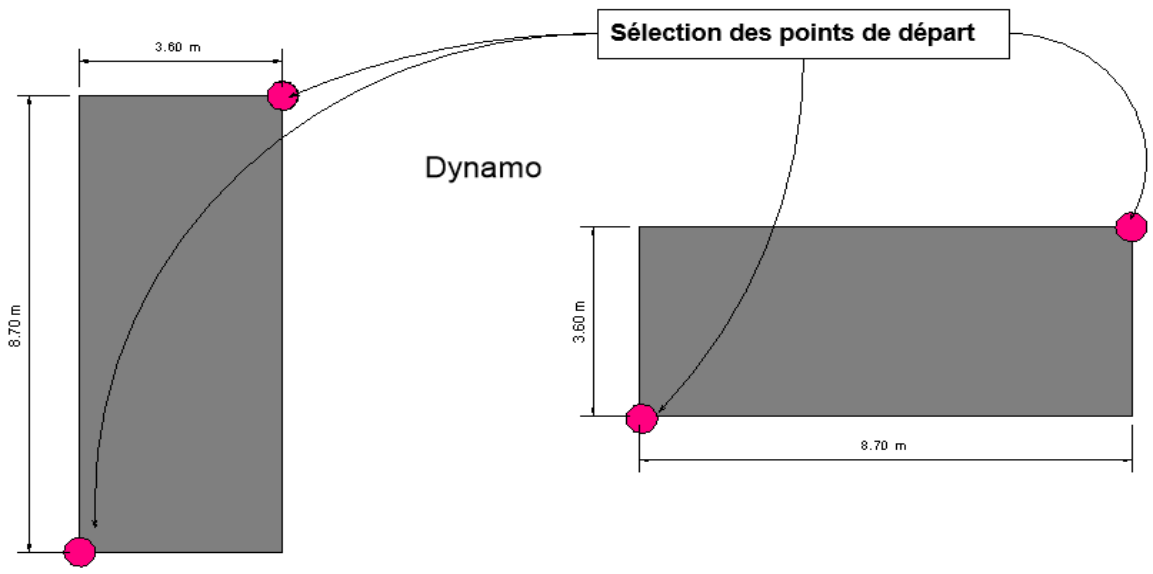




OBJECTIF: PROGRAMMATION & INITIATION au Langage PYTHON DANS DYNAMO

But de l'exercice :

Créer un découpage d'une surface rectangulaire par des diagonales à 45° pour calculer automatiquement les surfaces d'influences sur les cotés de cette dalle.



PROGRAMMATION DYNAMO REVIT

Qu'est-ce que la programmation visuelle ?

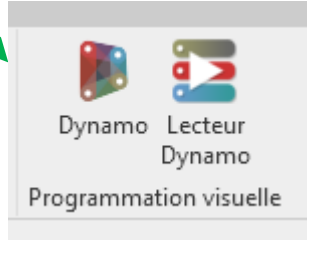
Dynamo est une plate-forme de programmation visuelle Open Source pour les concepteurs. Elle fait partie de l'installation de Revit.

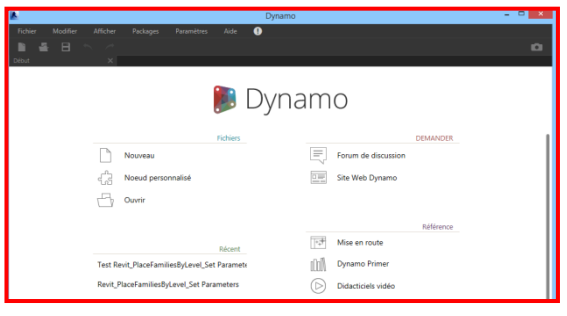
La conception implique souvent l'établissement de relations visuelles, systémiques ou géométriques entre les parties d'une conception.
Plus souvent qu'autrement, ces relations sont développées par des flux de travail qui nous amènent du concept au résultat au moyen de règles.
Peut-être sans le savoir, nous travaillons algorithmiquement, en définissant un ensemble d'actions qui suivent une logique de base d'entrée, de traitement et de sortie.
La programmation nous permet de continuer à travailler de cette façon mais en formalisant nos algorithmes.
L'application fournie à REVIT s'appelle DYNAMO. (Elle possède aussi un langage source qui s'appelle Python)

C'est un logiciel intégré de programmation visuelle. (Fourni dans le Pack 2019)
Extrêmement puissant et assez facile d'utilisation, il paraît très important d'explorer ses possibilités.

MODE OPERATOIRE :

Ouvrir un projet REVIT. Pour bien comprendre comment Dynamo fonctionne, penser à ouvrir un fichier avec des familles. (Exemples : structures, ossature ...)
Pour accéder à Dynamo, cliquez sur l'onglet Gérer ➤ le groupe de fonctions Programmation visuelle ➤ Dynamo.
Ouvrir ensuite DYNAMO en cliquant sur son icône dans l'onglet « Gérer » de REVIT-2019. Une fenêtre s'ouvre avec les options de Dynamo.





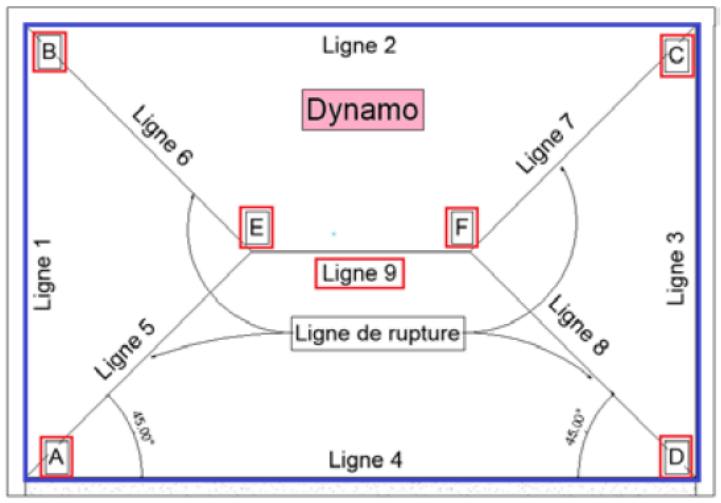
Organigramme

ORGANIGRAMME
Tracé et partage
D'une zone Rectangulaire

Saisie des Points A et C

$Point\ A$ $Point\ C$

Transformation
Des coordonnées (X,Y,Z)
Pour créer des points fixes (A et C)



Calcul base = b
Calcul hauteur = h

b = Cx - Ax
h = Cy - Ay

si b < 0
ou
si h < 0

NON

OUI

Envoie message d'erreur de saisie

Calcul des coordonnées (X,Y,Z) pour Créer les points (B et D)

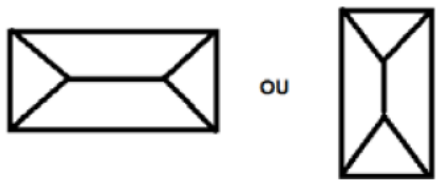
B = A (Translation.Y + h)
D = A (Translation.X + b)

Tracé du rectangle (A,B,C,D) → Lignes 1 à 4

h > b

NON

OUI



Rectangle Horizontale

Calcul des coordonnées (X,Y,Z) des points (E et F)

Rectangle Verticale

E = A (Trans.X + h/2) & A (Trans.Y + h/2)
F = C (Trans.X - h/2) & C (Trans.Y - h/2)

E = A (Trans.X + b/2) & A (Trans.Y + b/2)
F = C (Trans.X - b/2) & C (Trans.Y - b/2)

Tracé des Arrêtes → Lignes 5 à 9

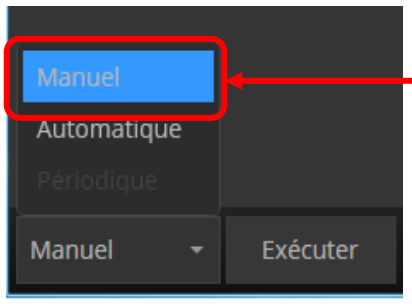
Calcul des surfaces → Surf 1 = Surf 3
Surf 2 = Surf 4

Tracé des Arrêtes → Lignes 5 à 9

Calcul des surfaces → Surf 1 = Surf 3
Surf 2 = Surf 4

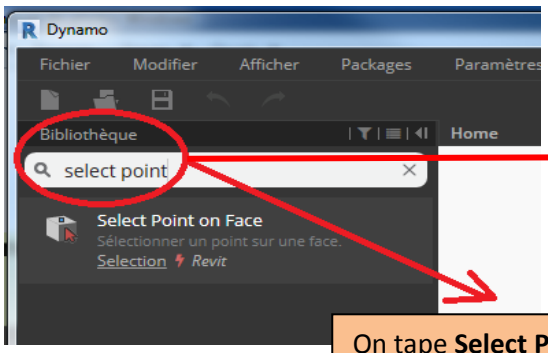
Affichage des résultats

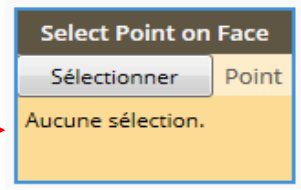
Vous gérez vos programmes Dynamo comme n'importe quel logiciel mais le format est un « .dyn ».
Vous pouvez en ouvrir un, ou en créer un nouveau, etc.



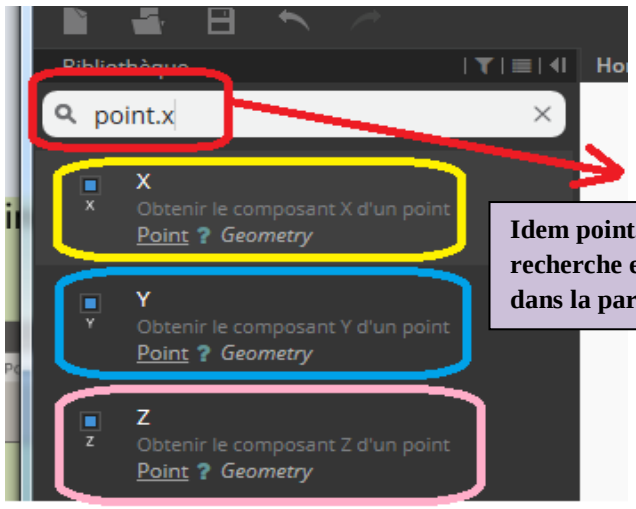
Pour des raisons de simplicité il faut placer le mode Manuel pour éviter une surcharge en mémoire

- **1er Etape on va chercher dans le menu les nœuds (*ou script de programme*) que l'on va utiliser dans notre programmation visuelle.**

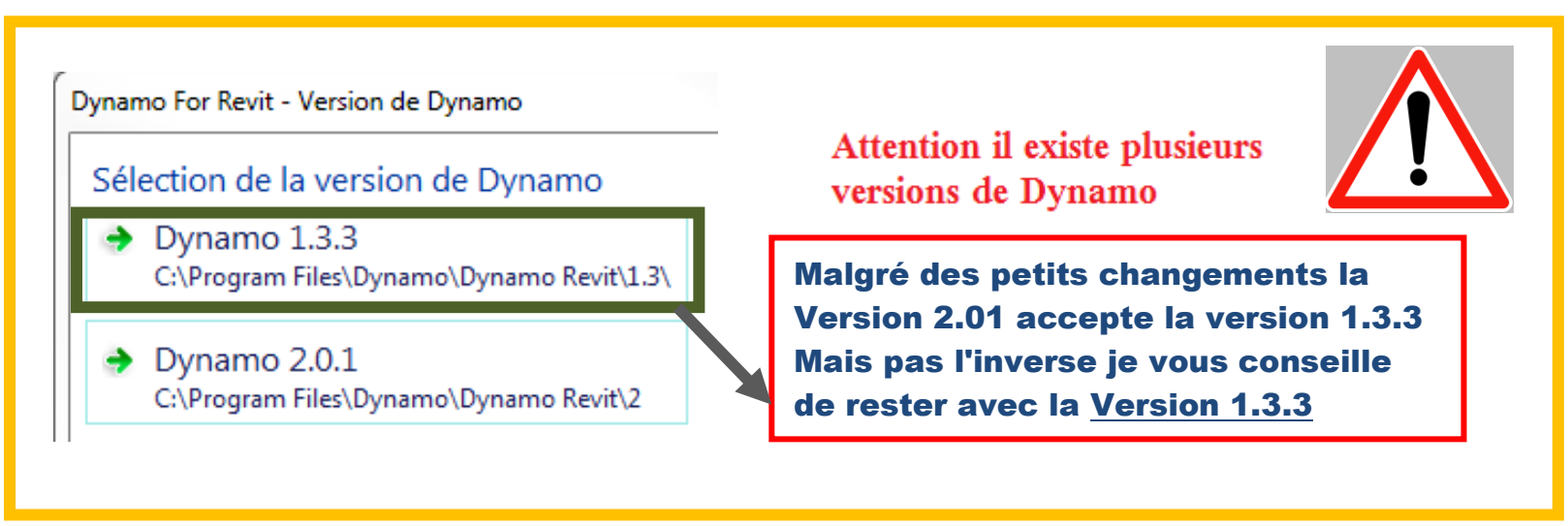




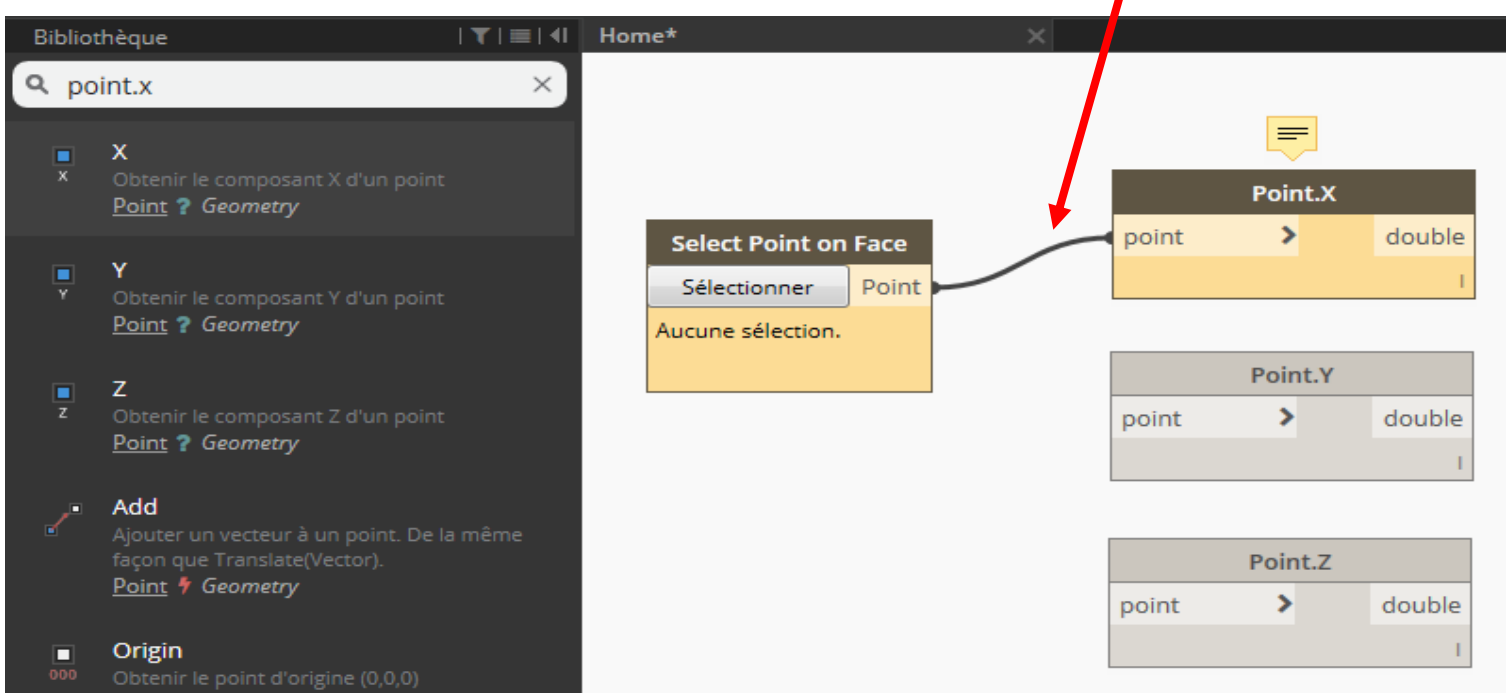
On tape **Select Point on face** dans la section recherche et on sélectionne l'objet dans la partie programmation

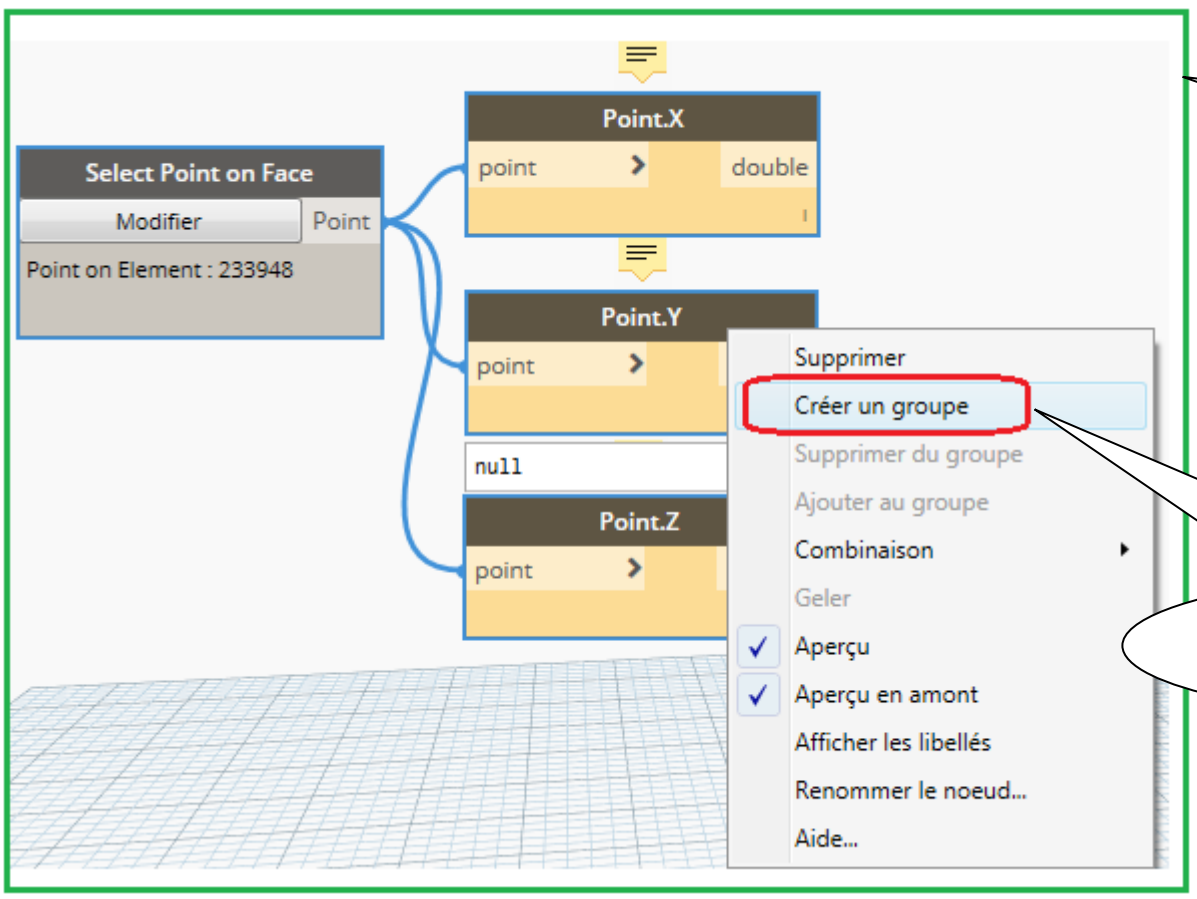


Idem point. X , Y et Z dans la section recherche et on sélectionne l'objet dans la partie programmation



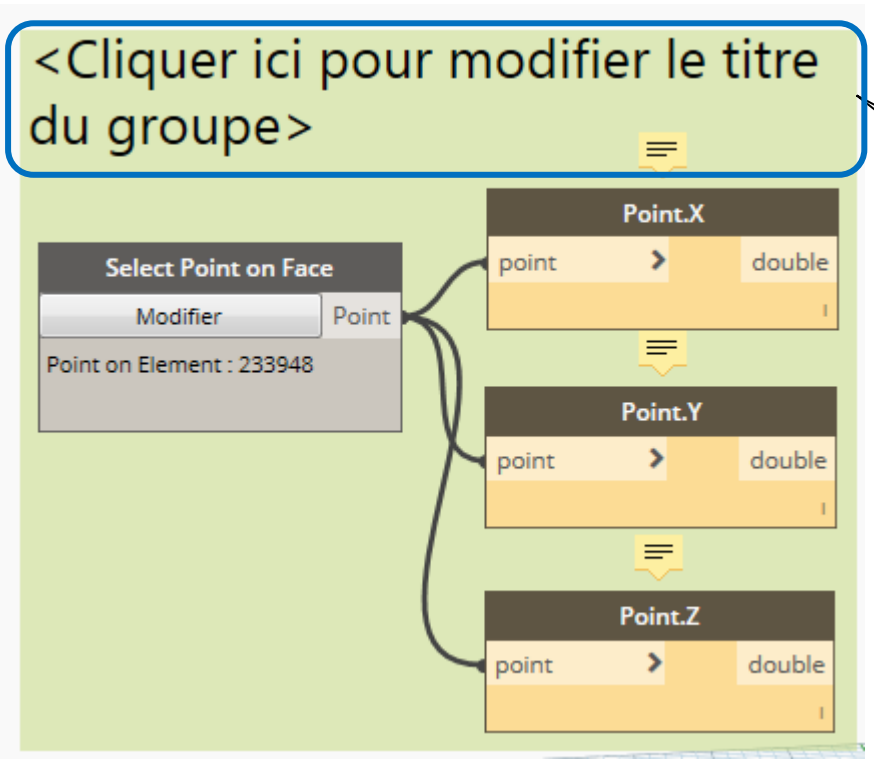
Ensuite on relie les points entre eux



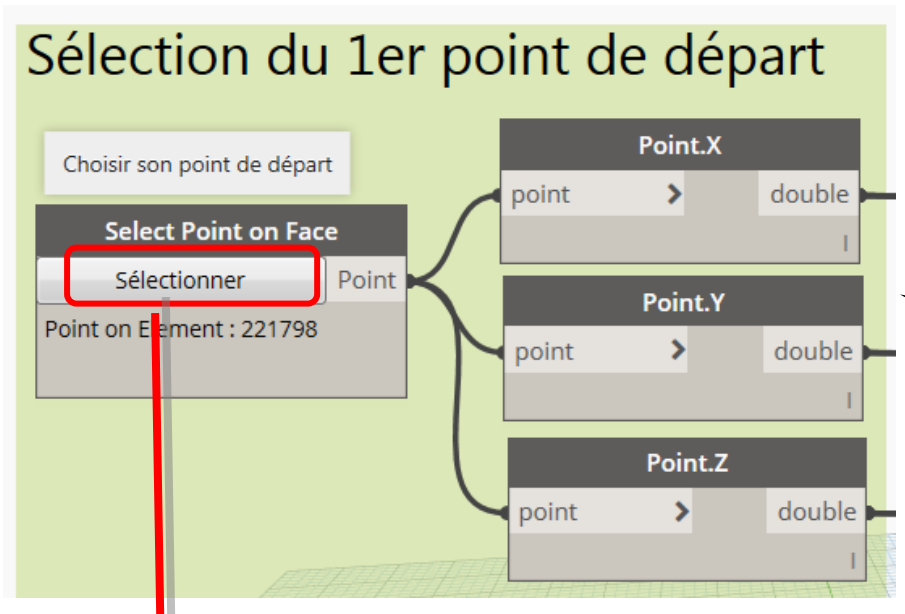


Lorsque l'on sélectionne par une fenêtre à l'aide de la souris

On peut créer un groupe qui facilite les déplacements et la présentation



On peut aussi modifier le titre du groupe



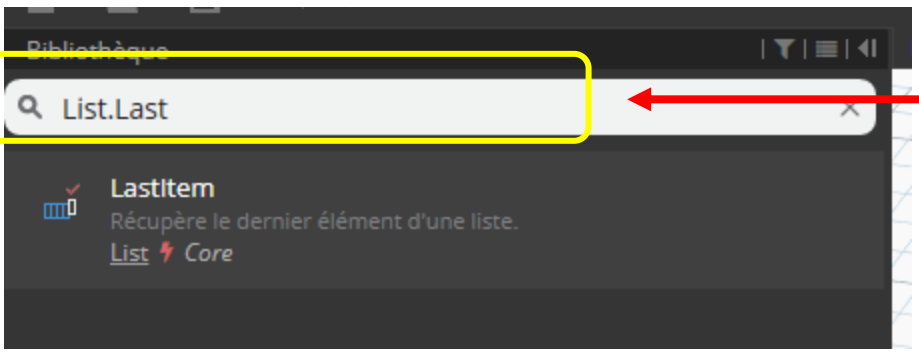
Les points sur REVIT peuvent être choisis à l'aide de la commande Select Point on Face

Il faudra cliquer ici lors de l'exécution
C'est une exécution que l'on verra plus tard

❖ 2e Etape suivante On désire sélectionner les entrées du programme :

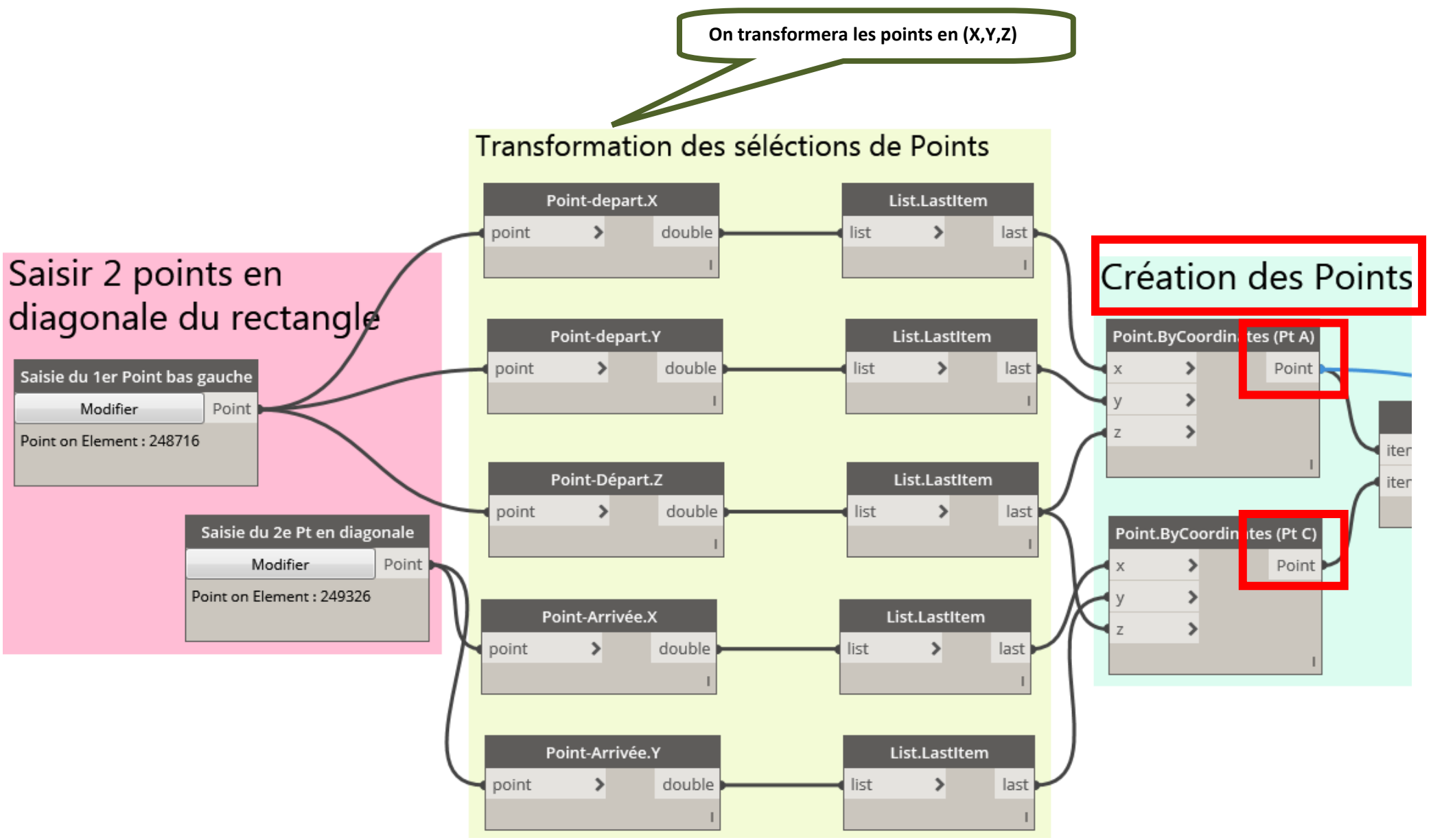
Nous allons créer 2 points d'insertion de données (c'est à dire le Point "A " Bas à Gauche et "C" Haut Droit).

Il faut utiliser un moyen pour trier les informations des points d'entrées

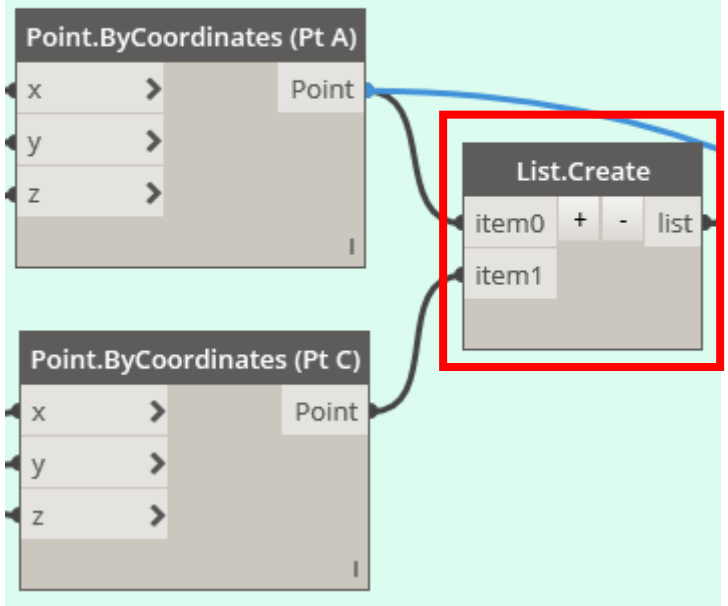


On tape dans la recherche le terme List.Last
C'est à dire les dernières valeurs de la liste.

On transformera les points en (X,Y,Z)



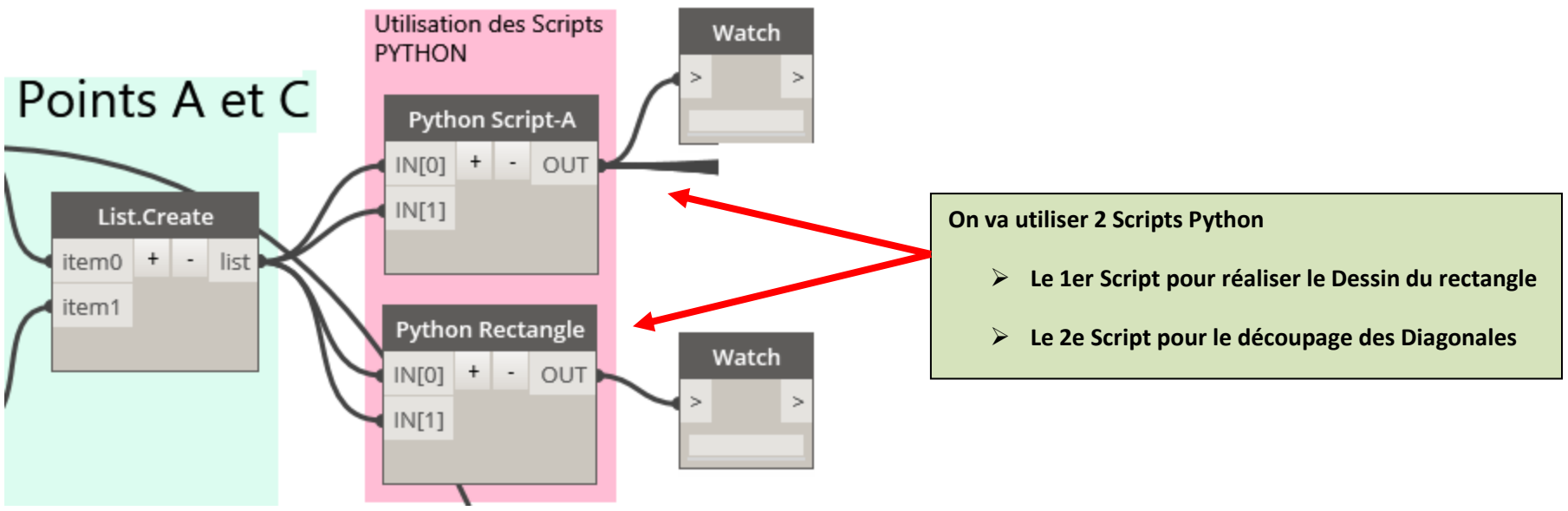
Création des Points A et C

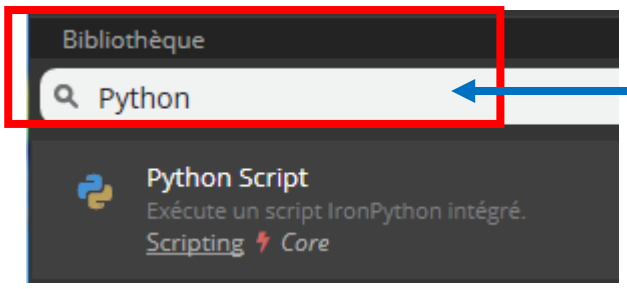


De la même manière on va rechercher le Node "List.Create"
Pour réaliser une liste de points

❖ 3e Etape Utilisation des formules PYTHON pour les entrées du programme :

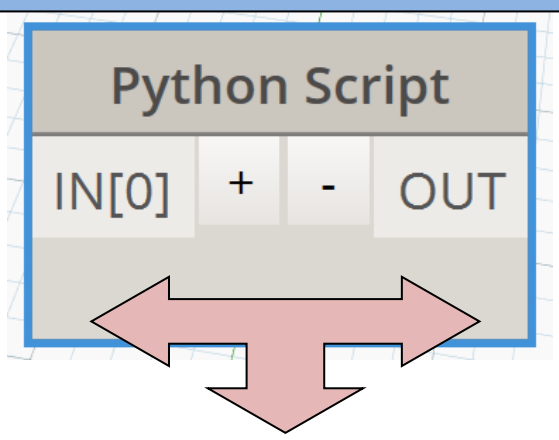
❖ **Objectif réduire le nombre de liens en forme de spaghettis qui encombrent le programme, en transformant quelques commandes par des Scripts Python.**

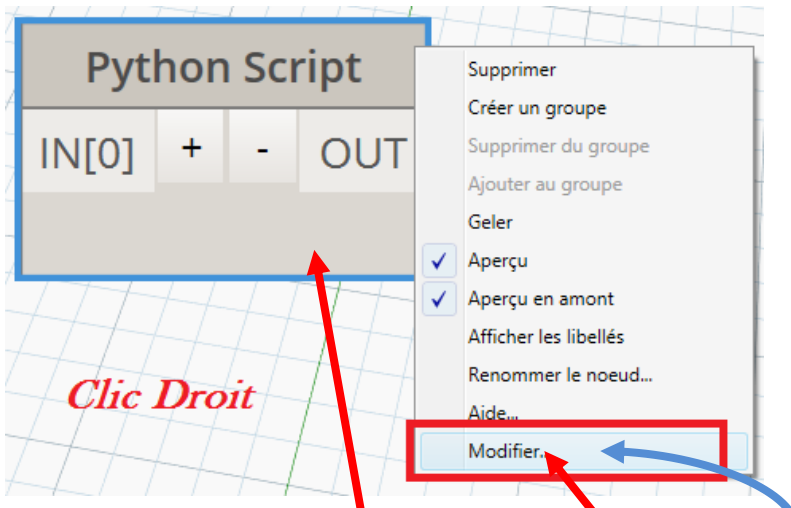


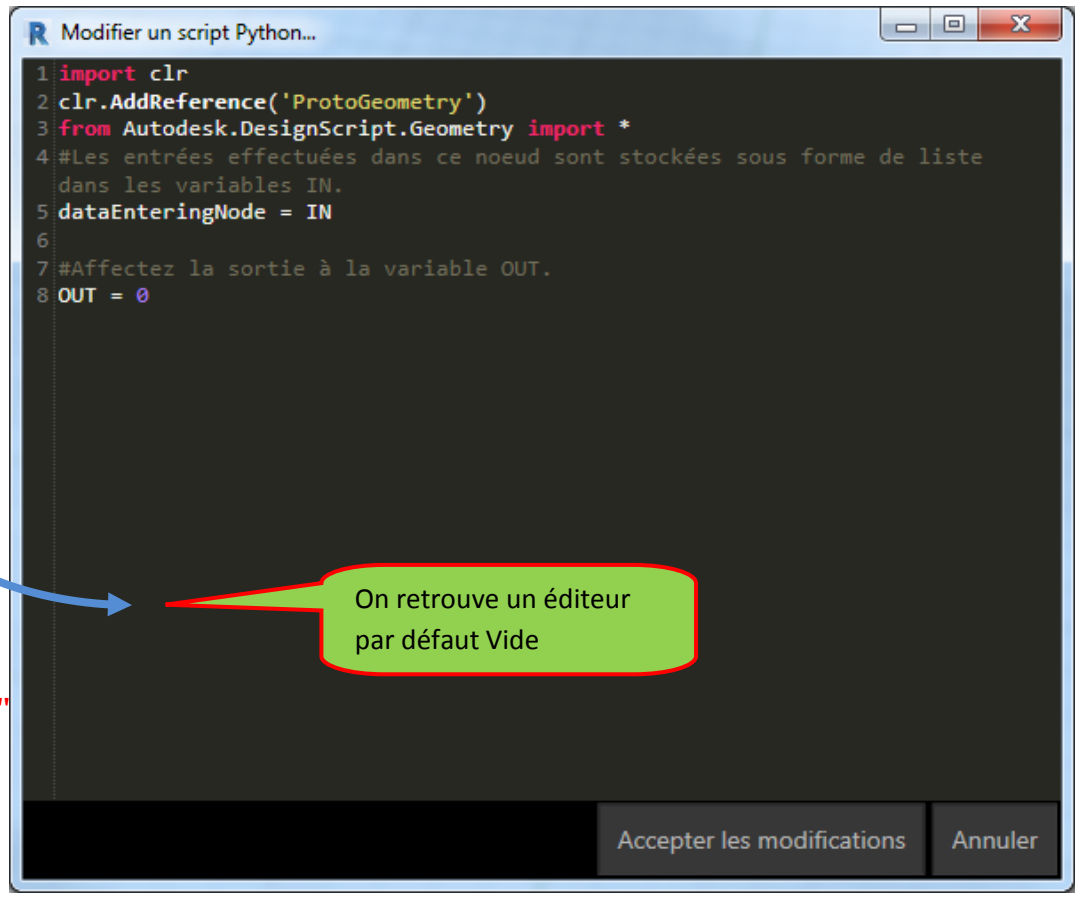


On tape dans la recherche le terme Python
Pour insérer le NOde du Script.

On retrouve les 2 types: 1 entrée multiple + 1 Sortie







Pour modifier le script on fait un clic Droit et on sélectionne "Modifier"

Préparation du Codage en Python

Nous allons rentrer les valeurs de départ.

Modifier un script Python...

```
import clr
clr.AddReference('ProtoGeometry')
from Autodesk.DesignScript.Geometry import *
#Les entrées effectuées dans ce noeud sont stockées sous forme de liste
dans les variables IN
dataEnteringNode = IN
Nod1 = IN[0]
Nod2 = IN[1]
# Création des points A et C
A = Nod1[0]
C = Nod2[1]
# On va faire une déclaration des nouveaux paramétres
nouveauPointB =[]
nouveauPointD =[]
# Calcul de la la base et la Hauteur du rectangle
# Sélection des valeurs en X & Y de chaque points
base = (C.X - A.X);
hauteur = (C.Y - A.Y);

B = Geometry.Translate(A,0,hauteur,0);
nouveauPointB.append(B)
D = Geometry.Translate(A,base,0,0);
nouveauPointD.append(D)
#Affectez la sortie à la variable OUT
Line1 = Line.ByStartPointEndPoint(A,B);
Line2 = Line.ByStartPointEndPoint(B,C);
Line3 = Line.ByStartPointEndPoint(C,D);
Line4 = Line.ByStartPointEndPoint(A,D);
OUT = base,hauteur,Line1,Line2,Line3,Line4,nouveauPointB,nouveauPointD

Accepter les modifications   Annuler
```

Affectation des entrées aux variables
Nod1 = IN[0] sera donc la 1e Entrée
Nod2 = IN[1] sera donc la 2e Entrée

Ne pas oublier de faire les déclarations des valeurs de sorties

On calcule la base et la hauteur du rectangle

On passe à la création des nouveaux Points B et D

Création des lignes 1 à 4

OUT = Sorties des informations et des Valeurs

Modèle de construction du Rectangle en partant du Point A jusqu'au point D

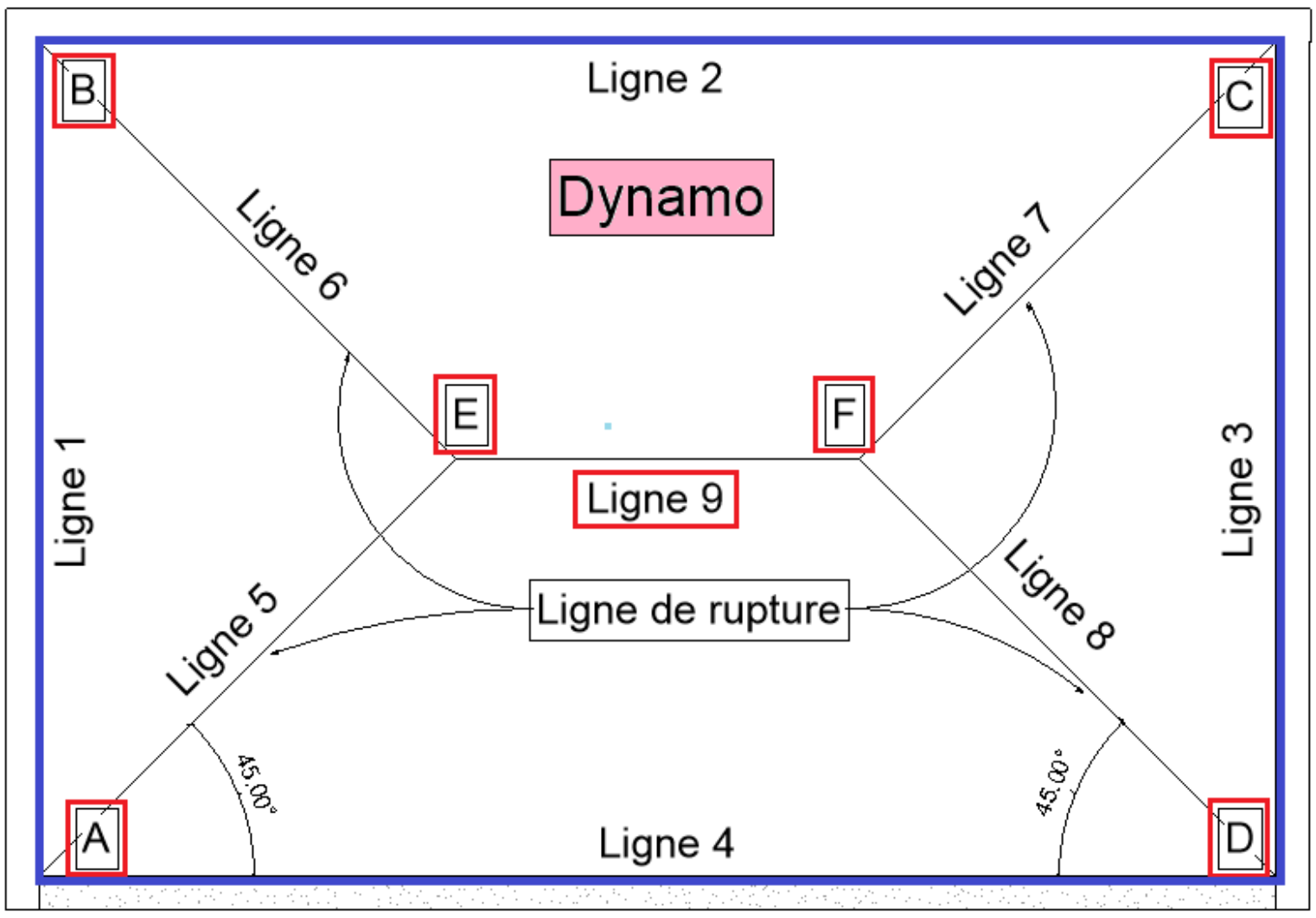
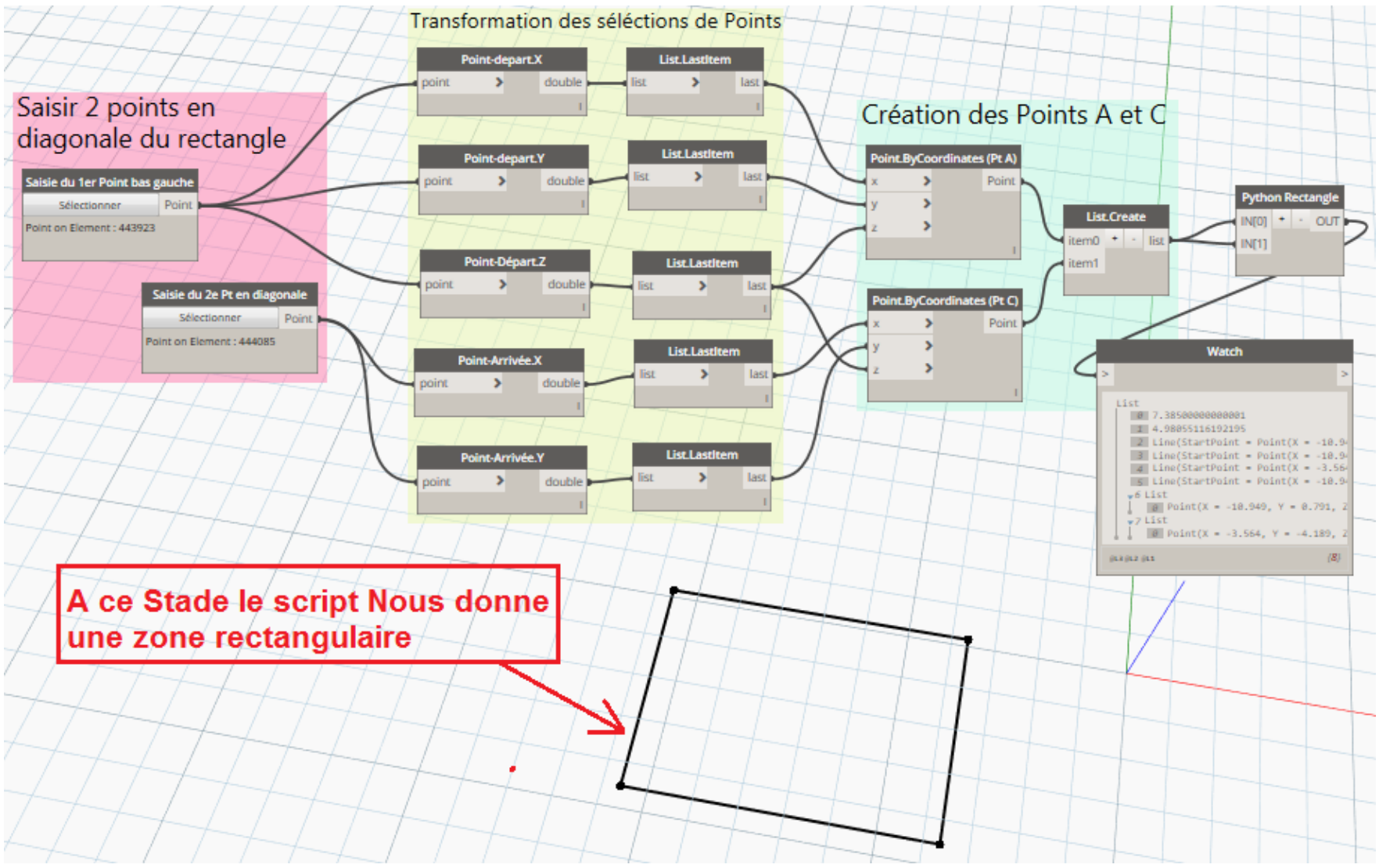


Fig: 1-A

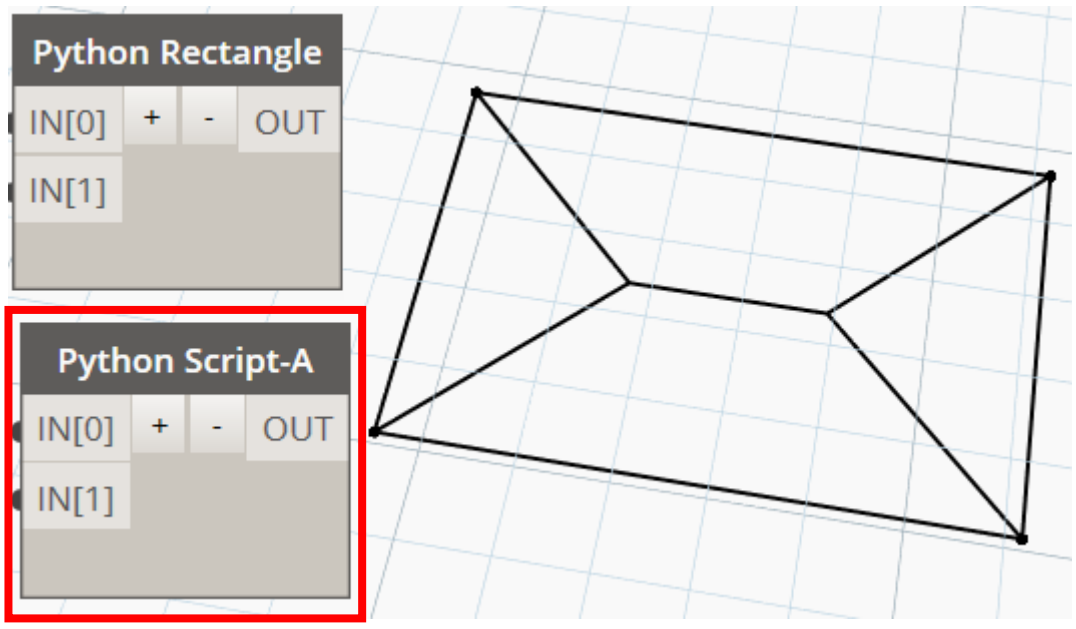
Résumer du code source PYTHON: (Rectangle)

```
import clr
clr.AddReference('ProtoGeometry')
from Autodesk.DesignScript.Geometry import *
#Les entrées effectuées dans ce noeud sont stockées sous forme de liste dans les variables IN.
dataEnteringNode = IN
Nod1 = IN[0]
Nod2 = IN[1]
# Création des points A et C
A = Nod1[0]
C = Nod2[1]
# On va faire une déclaration des nouveaux paramètres
nouveauPointB =[ ]
nouveauPointD =[ ]
# Calcul de la base et la Hauteur du rectangle
# Sélection des valeurs en X & Y de chaque point
base = (C.X - A.X);
hauteur = (C.Y - A.Y);
B = Geometry.Translate(A,0,hauteur,0);
nouveauPointB.append(B)
D = Geometry.Translate(A,base,0,0);
nouveauPointD.append(D)
#Affectez la sortie à la variable OUT.
Line1 = Line.ByStartPointEndPoint(A,B);
Line2 = Line.ByStartPointEndPoint(B,C);
Line3 = Line.ByStartPointEndPoint(C,D);
Line4 = Line.ByStartPointEndPoint(A,D);
OUT = base,hauteur,Line1,Line2,Line3,Line4,nouveauPointB,nouveauPointD
```

Si on sélectionne les 2 points A et C on doit avoir ce schéma ci-dessous.



❖ 4e Etape Utilisation création d'un autre script PYTHON pour créer les arrêtes :



<u>SCRIPT A:</u> **Calcul de la base et de la hauteur + création des arrêtes.**

Modifier un script Python...

```
import clr
clr.AddReference('ProtoGeometry')
from Autodesk.DesignScript.Geometry import *
#Les entrées effectuées dans ce noeud sont stockées sous forme de liste
dans les variables IN.
dataEnteringNode = IN
Nod1 = IN[0]
Nod2 = IN[1]
A = Nod1[0]
C = Nod2[1]
nouveauPointB =[]
nouveauPointD =[]
nouveauPointE =[]
nouveauPointF =[]
base = (C.X - A.X);
hauteur = (C.Y - A.Y);
B = Geometry.Translate(A,0,hauteur,0);
nouveauPointB.append(B)
D = Geometry.Translate(A,base,0,0);
nouveauPointD.append(D)
```

Accepter les modifications | Annuler

<u>Affectation des entrées aux variables</u>
Nod1 = IN[0] sera donc la 1e Entrée (A)
Nod2 = IN[1] sera donc la 2e Entrée (C)

On passe à la création des nouveaux Points B,D,E,F

On calcule la base et la hauteur du rectangle
On utilisera les coordonnées:
de C(x,y) et de A(x,y):

Il est clair que l'on peut facilement utiliser un seul script pour ces opérations mais le but est de décomposer en plusieurs scripts afin de réaliser un apprentissage par étapes qui sera plus productif vis à vis de la vérification.

Modifier un script Python...

```
B = Geometry.Translate(A,0,hauteur,0);
nouveauPointB.append(B)
D = Geometry.Translate(A,base,0,0);
nouveauPointD.append(D)
if hauteur > base:
        E = Geometry.Translate(A,base/2,base/2,0);
        nouveauPointE.append(E)
        F = Geometry.Translate(C,-base/2,-base/2,0);
        nouveauPointE.append(F)
        Line5 = Line.ByStartPointEndPoint(A,E);
        Line6 = Line.ByStartPointEndPoint(B,F);
        Line7 = Line.ByStartPointEndPoint(C,F);
        Line8 = Line.ByStartPointEndPoint(D,E);
        Line9 = Line.ByStartPointEndPoint(E,F);

else:
        E = Geometry.Translate(A,hauteur/2,hauteur/2,0);
        nouveauPointE.append(E)
        F = Geometry.Translate(C,-hauteur/2,-hauteur/2,0);
        nouveauPointE.append(F)
        Line5 = Line.ByStartPointEndPoint(A,E);
        Line6 = Line.ByStartPointEndPoint(B,E);
        Line7 = Line.ByStartPointEndPoint(C,F);
        Line8 = Line.ByStartPointEndPoint(D,F);
        Line9 = Line.ByStartPointEndPoint(E,F);

OUT = Line5,Line6,Line7,Line8,Line9
```

La position des points E et F
En position Verticale

La position des points E et F
En position Horizontale

Sortie des tracés des Arrêtes

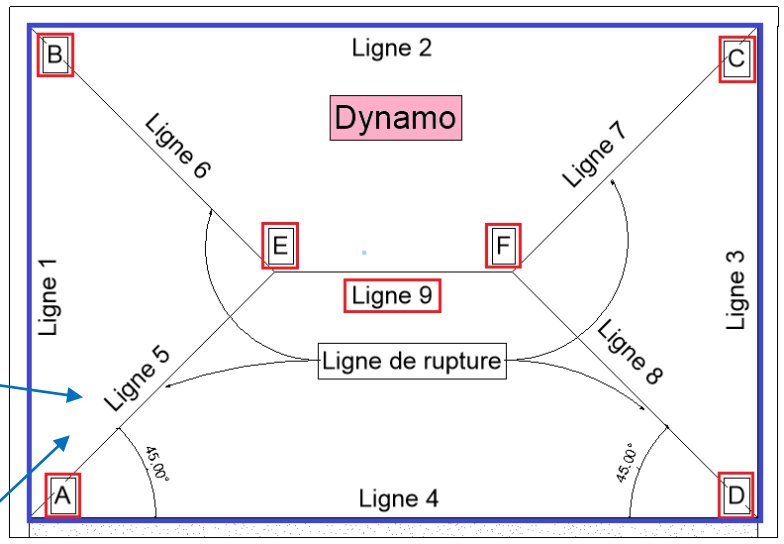
On va réaliser dans cette 2e Partie
Les tests de vérifications si le rectangle est en position verticale ou horizontale.

Attention de bien dégager des Espaces dans le Script Sinon vous aurez des erreurs !.

Résumer du code source PYTHON: (Tracé des arêtes)

```
import clr
clr.AddReference('ProtoGeometry')
from Autodesk.DesignScript.Geometry import *
#Les entrées effectuées dans ce noeud sont stockées sous forme de liste dans les variables IN.
dataEnteringNode = IN
Nod1 = IN[0]
Nod2 = IN[1]
A = Nod1[0]
C = Nod2[1]
nouveauPointB =[ ]
nouveauPointD =[ ]
nouveauPointE =[ ]
nouveauPointF =[ ]
base = (C.X - A.X);
hauteur = (C.Y - A.Y);
B = Geometry.Translate(A,0,hauteur,0);
nouveauPointB.append(B)
D = Geometry.Translate(A,base,0,0);
nouveauPointD.append(D)
if hauteur > base:
        E = Geometry.Translate(A,base/2,base/2,0);
        nouveauPointE.append(E)
        F = Geometry.Translate(C,-base/2,-base/2,0);
        nouveauPointE.append(F)
        Line5 = Line.ByStartPointEndPoint(A,E);
        Line6 = Line.ByStartPointEndPoint(B,F);
        Line7 = Line.ByStartPointEndPoint(C,F);
        Line8 = Line.ByStartPointEndPoint(D,E);
        Line9 = Line.ByStartPointEndPoint(E,F);
else:
        E = Geometry.Translate(A,hauteur/2,hauteur/2,0);
        nouveauPointE.append(E)
        F = Geometry.Translate(C,-hauteur/2,-hauteur/2,0);
        nouveauPointE.append(F)
        Line5 = Line.ByStartPointEndPoint(A,E);
        Line6 = Line.ByStartPointEndPoint(B,E);
        Line7 = Line.ByStartPointEndPoint(C,F);
        Line8 = Line.ByStartPointEndPoint(D,F);
        Line9 = Line.ByStartPointEndPoint(E,F);

OUT = Line5,Line6,Line7,Line8,Line9 ----------> Sortie des 5 Lignes
```



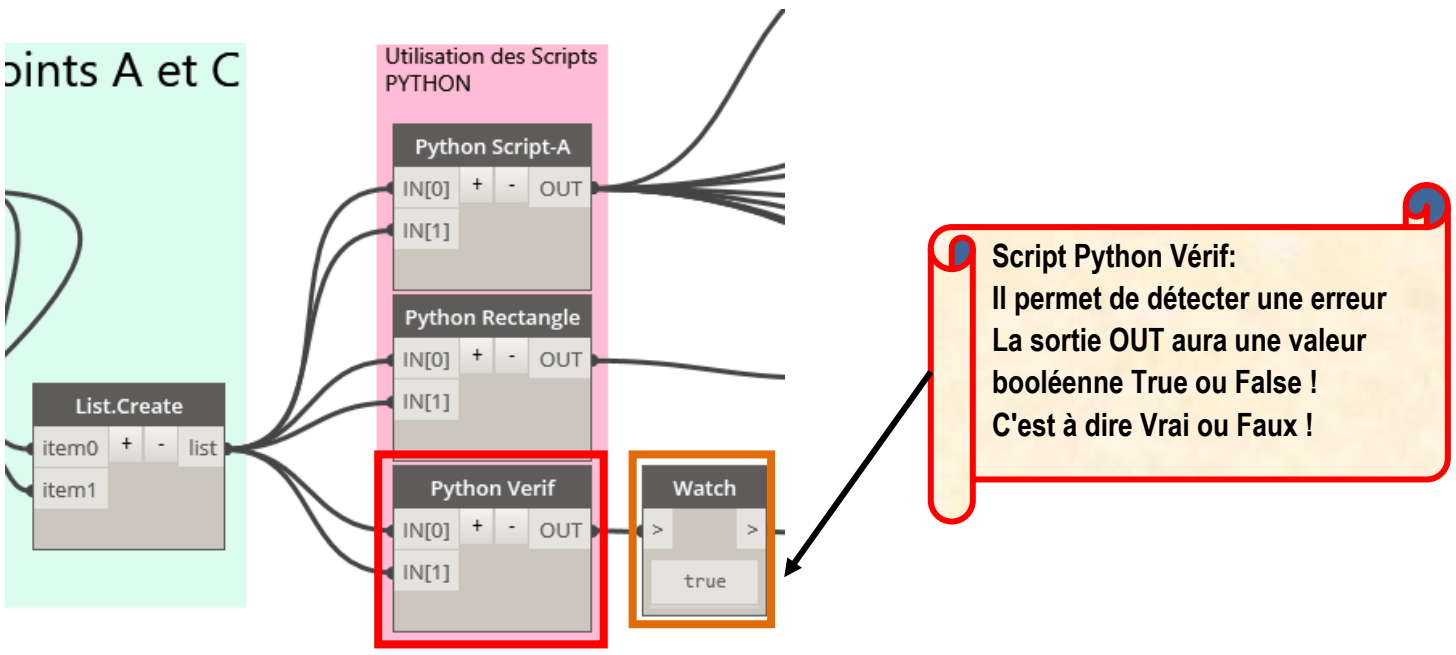
❖ 5e Etape Utilisation création d'un autre script PYTHON pour faire une vérification :

Comme tout bon programme qui se respecte il faut réaliser des vérifications afin que l'utilisateur distrait ne fasse pas d'erreurs de saisie.

Dans ce Script la vérification sera simple si une saisie de point transforme la base ou la hauteur avec une valeur négative.

Cela voudra dire que la saisie des points n'a pas été correct.

Je précise de nouveau, que ce Script aurait pu être intégré dans le même Module de création des dessins, mais pour débuter il est souhaitable de séparer ces blocs.



Suite Script Python de vérification

```python
import clr
clr.AddReference('ProtoGeometry')
clr.AddReference('RevitAPIUI')
from Autodesk.Revit.UI import TaskDialog

from Autodesk.DesignScript.Geometry import *
#Les entrées effectuées dans ce noeud sont stockées sous forme de liste
dans les variables IN.
dataEnteringNode = IN
Nod1 = IN[0]
Nod2 = IN[1]
# Saisie des points A et C
A = Nod1[0]
C = Nod2[1]
# On va faire une déclaration des nouveaux paramétres
windowTitle =  []
windowMessage =[]
Verif = []
Verif = True # valeur par défaut
# Calcul de la Hauteur et la Largeur !!

base = (C.X - A.X);
hauteur = (C.Y - A.Y);

if base < 0 or hauteur < 0:
        windowTitle =  "Message d'erreur !!"
        windowMessage ="Erreur de saisie recommencer vos points !!"
        messageDialog = TaskDialog
        Verif = False
        #Affectez la sortie à la variable OUT envoie le message d'erreur.
        OUT = messageDialog.Show(windowTitle, windowMessage), Verif
#Affectez la sortie à la variable OUT.
OUT = Verif
```

Saisie des 2 points d'entrée

Calcul de la base et la hauteur

Vérification si la valeur de la base ou de la hauteur est négative.

La fonctionTaskDialog permet la visualisation du message d'alerte dans REVIT

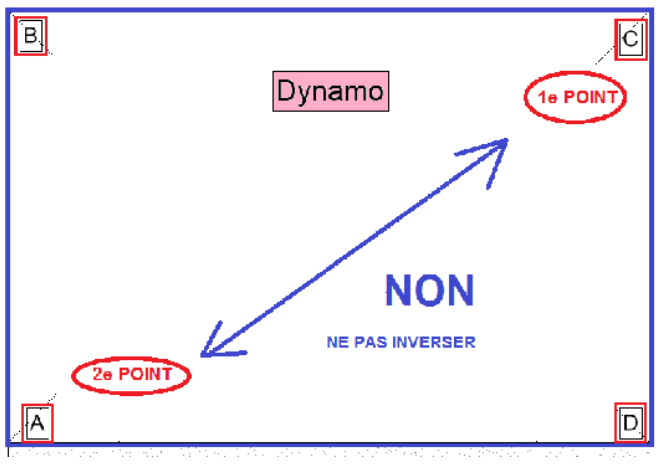
Résumer du code source PYTHON: (Vérifications)

import clr
clr.AddReference('ProtoGeometry')
clr.AddReference('RevitAPIUI')
from Autodesk.Revit.UI import TaskDialog

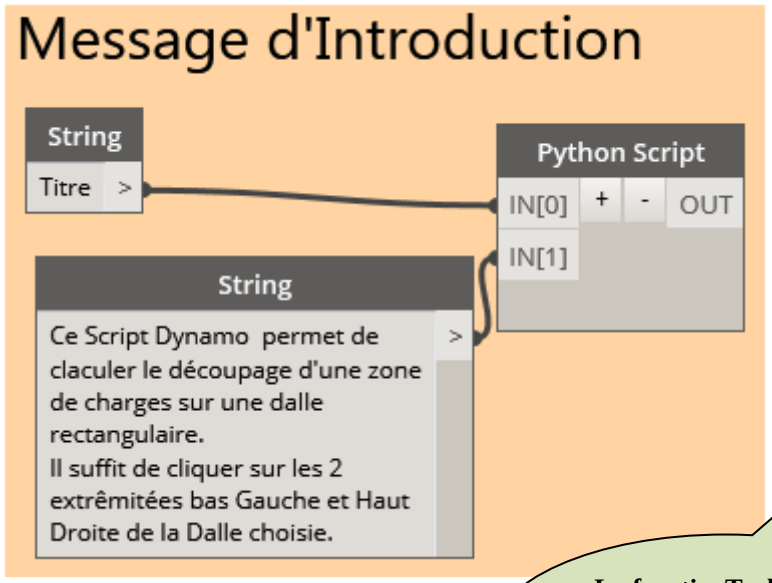
from Autodesk.DesignScript.Geometry import *
#Les entrées effectuées dans ce noeud sont stockées sous forme de liste dans les variables IN.
dataEnteringNode = IN
Nod1 = IN[0]
Nod2 = IN[1]
 # Saisie des points A et C
A = Nod1[0]
C = Nod2[1]
 # On va faire une déclaration des nouveaux paramétres
windowTitle = [ ]
windowMessage =[ ]
Verif = [ ]
Verif = True # valeur par défaut
 # Calcul de la Hauteur et la Largeur !!

base = (C.X - A.X);
hauteur = (C.Y - A.Y);

if base < 0 or hauteur < 0:
windowTitle = "Message d'erreur !!"
windowMessage ="Erreur de saisie recommencer vos points !!"
messageDialog = TaskDialog
Verif = False
#Affectez la sortie à la variable OUT envoie le message d'erreur.
OUT = messageDialog.Show(windowTitle, windowMessage), Verif
#Affectez la sortie à la variable OUT.
OUT = Verif ----------> Sortie de la valeur Booléenne "Verif" (True ou False suivant le cas)



- **Autre Script non obligatoire qui permet de donner une explication au démarrage à l'utilisateur du code**

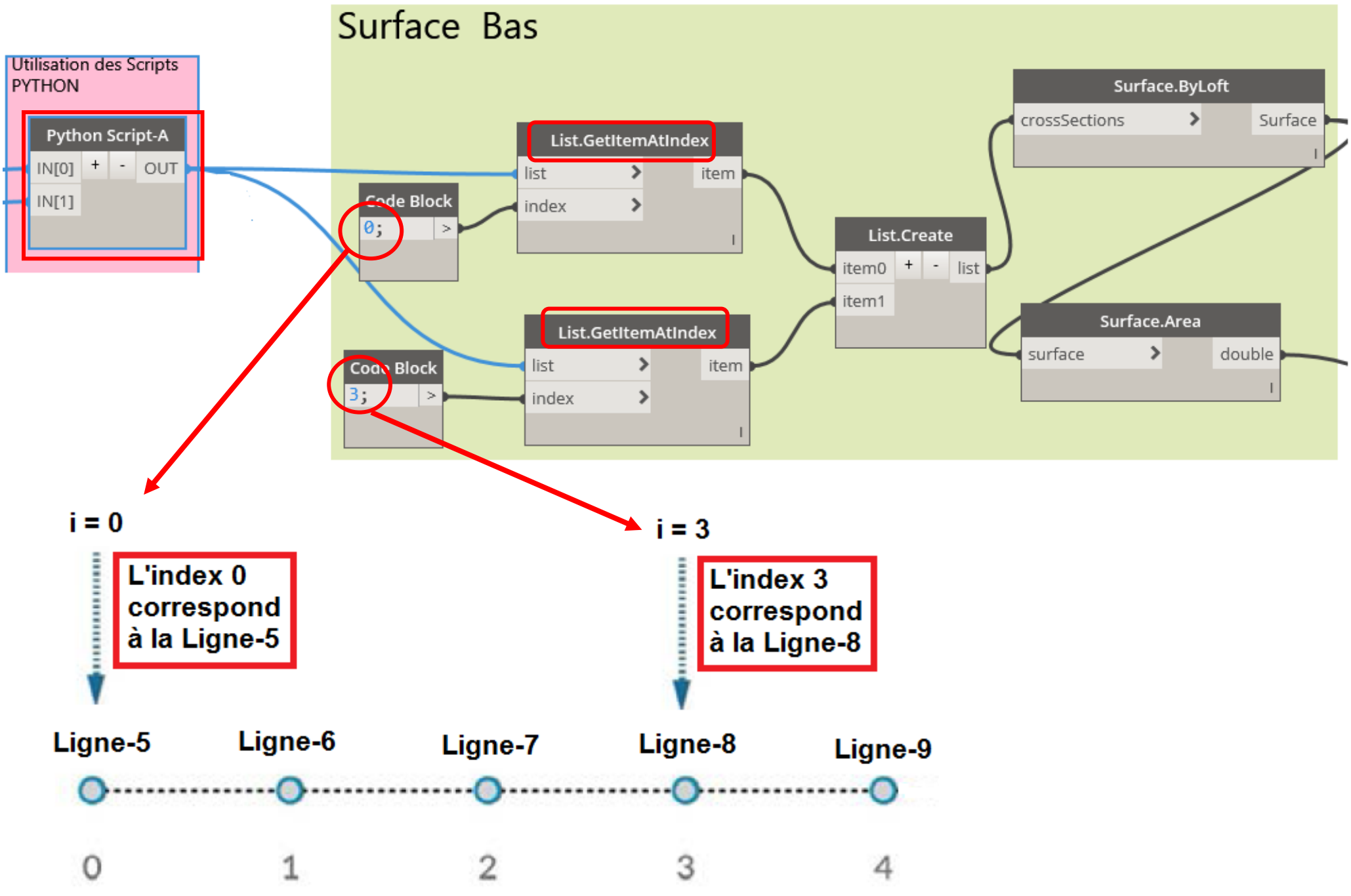



```python
import clr
clr.AddReference('ProtoGeometry')
from Autodesk.DesignScript.Geometry import *
clr.AddReference('RevitAPIUI')
from Autodesk.Revit.UI import TaskDialog
#Les entrées effectuées dans ce noeud sont stockées sous forme de liste dans les variables IN.
dataEnteringNode = IN
windowTitle = IN[0]
windowMessage = IN[1]

messageDialog = TaskDialog
#Affectez la sortie à la variable OUT.
OUT = messageDialog.Show(windowTitle, windowMessage)
```

La fonctionTaskDialog permet la visualisation du message d'ouverture dans REVIT

❖ 6e Etape Formules pour créer une surface avec Dynamo REVIT :

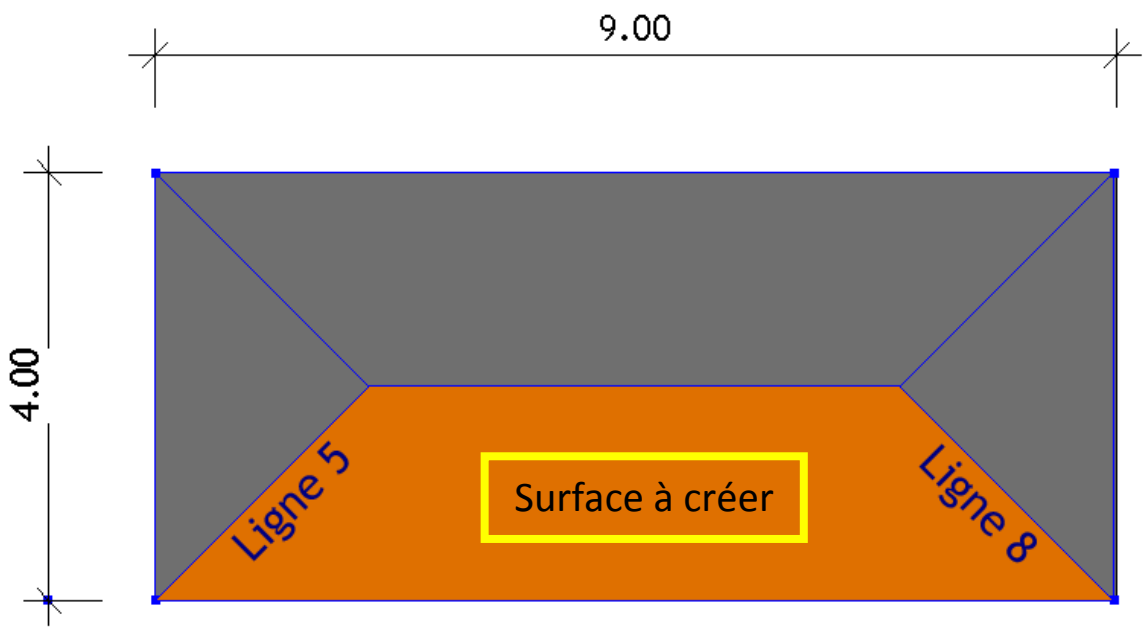


List.GetItemAtIndex est un moyen fondamental d'interroger un élément dans la liste.

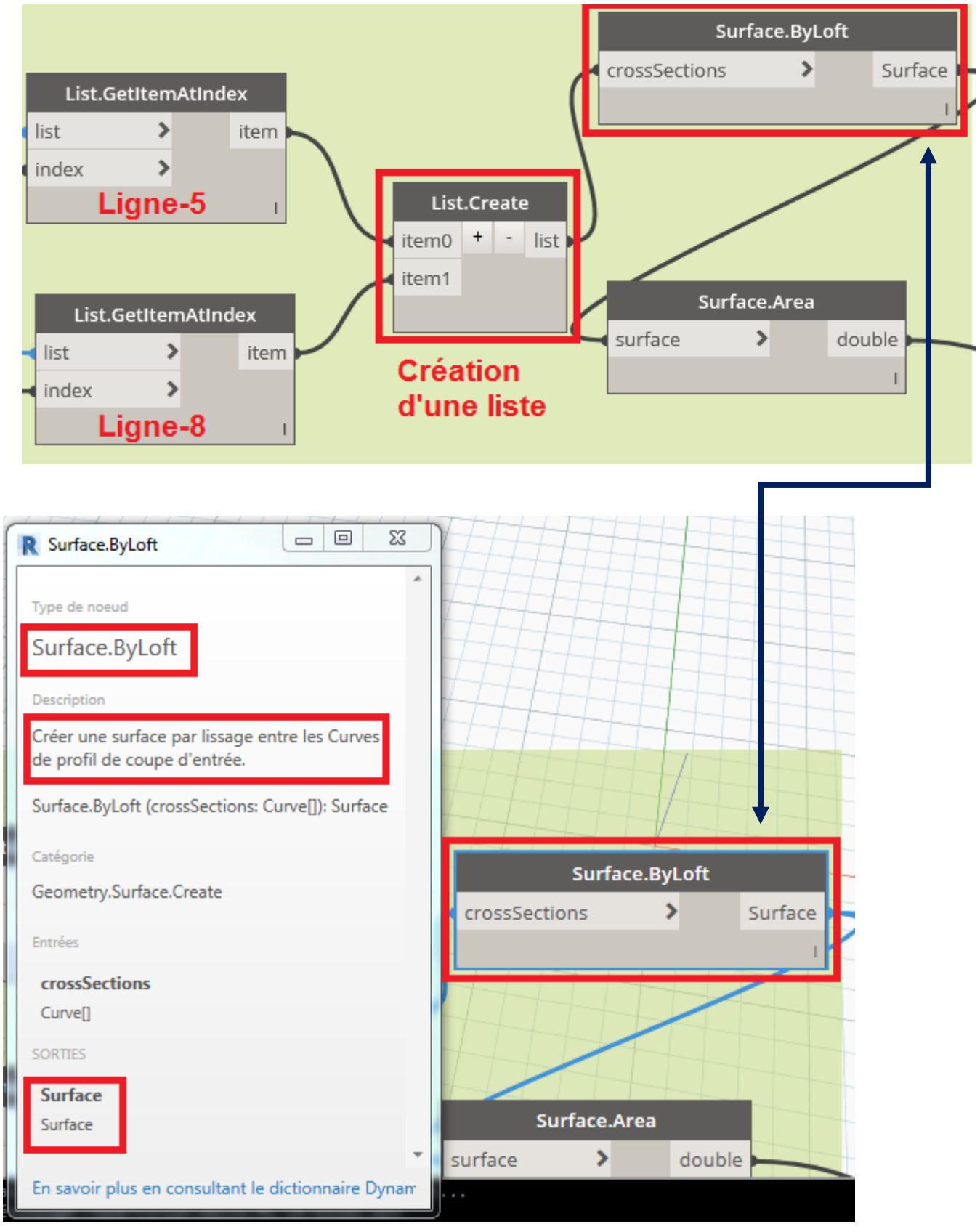
Dans l'image ci-dessus, nous utilisons un indice de "0 " pour interroger la ligne intitulé " Ligne-5 ».

Et pour un indice de "3 " cela permet d'interroger la ligne intitulé " Ligne-8 ».

Modèle de Surface à créer



En partant de la ligne 5 vers la ligne 8 l'objectif est de créer une surface à l'aide de la commande "Surface.ByLoft".

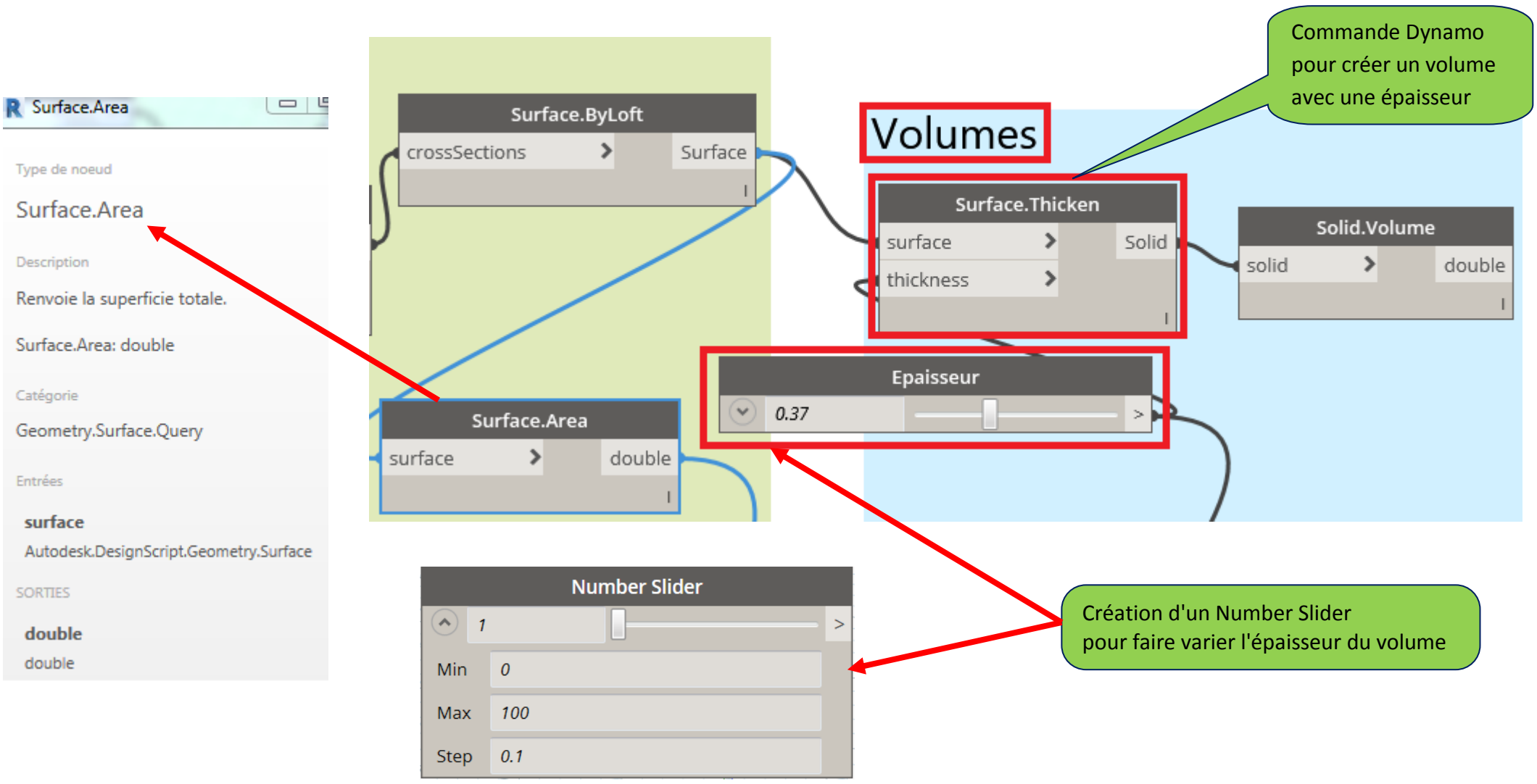


En résumer nous allons donc créer par découpage, le rectangle en 4 surfaces (dont 2 identiques).
C'est à dire: 2 parallélogrammes + 2 Triangles.

(Faire un copier - coller pour répéter l'opération des autres surfaces)

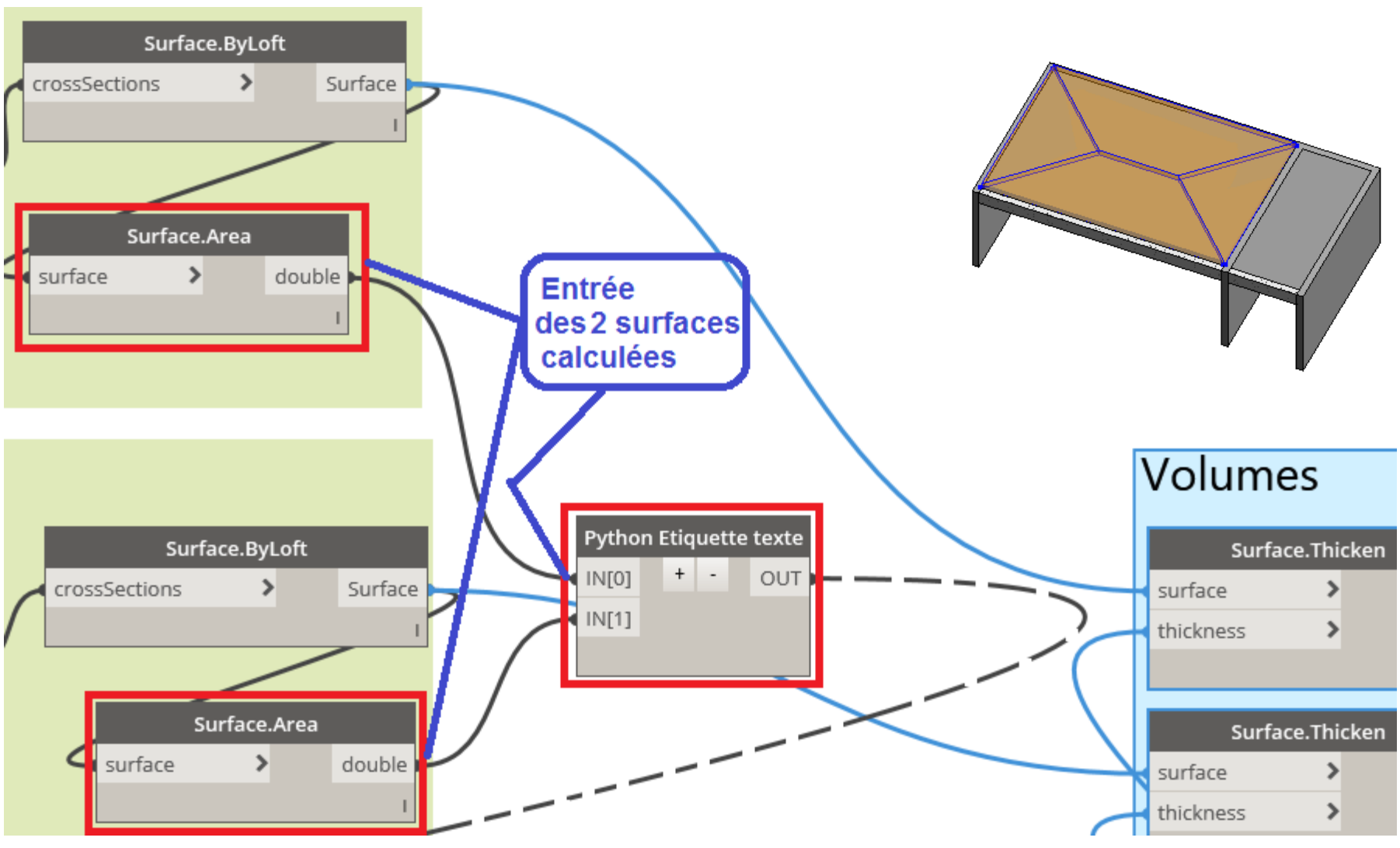
❖ 7e Etape Formules pour créer un volume avec Dynamo REVIT :

Pour une question uniquement esthétique : (car ce n'est pas utile pour notre exemple)
Nous allons créer par découpage, les volumes des 2 parallélogrammes + 2 Triangles



(Faire un copier - coller pour répéter l'opération des autres Volumes)

❖ 8e Etape Création d'un Script Python pour afficher les étiquettes surfaces avec Dynamo REVIT :



→ Suite Script Python création d'étiquette texte

```python
import clr
clr.AddReference('ProtoGeometry')
from Autodesk.DesignScript.Geometry import *
#Les entrées effectuées dans ce noeud sont stockées sous forme de liste
dans les variables IN.
dataEnteringNode = IN
Dep0 = IN[0]
Dep1 = IN[1]
DepA = []
DepB = []
DepT = []
Resultat = []
del Resultat

Total = (Dep0 + Dep1) * 2
#DepA =  str(Dep0);
DepA = "{0:.2f}".format(round(Dep0,2))
#DepB =  str(Dep1);
DepB = "{0:.2f}".format(round(Dep1,2))
#DepT =  str(Total)
DepT = "{0:.2f}".format(round(Total,2))

# "\n" donne le Retour à la ligne
if Dep1 > Dep0:
    DepA = "Triangle= "+ DepA + " m2" + "\n";
    DepB = "Parallélogramme= "+ DepB + " m2" + "\n";
    DepT = "Sur_Tot = " + DepT + " m2"
else:
    DepA = "Parallélogramme= "+ DepA + " m2" + "\n";
    DepB = "Triangle= "+ DepB + " m2" + "\n";
    DepT = "Sur_Tot = " + DepT + " m2"

Resultat = DepA + DepB + DepT
#Affectez la sortie à la variable OUT.
OUT = Resultat
```

Entrée des valeurs de surfaces

Calcul surface totale

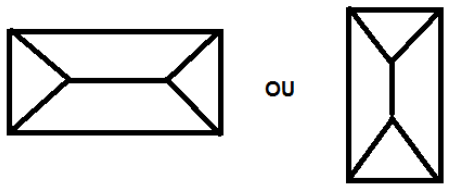
Commande Format Texte

Ne pas oublier de placer des espaces

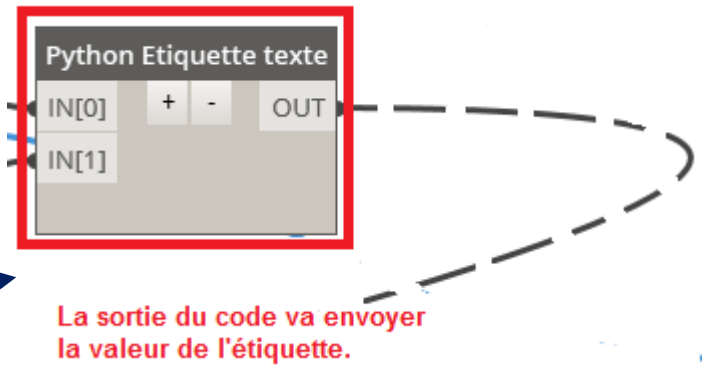
L'instruction Python :
"{0:.2f}".format(round(..)
permet de placer une valeur arrondie à 2 chiffres après la virgule

L'instruction Python :
"\n" permet un retour à la ligne

Test if (si la valeur Dep1 est supérieur (>) à Dep0 cela veut dire que la surface rectangulaire est Horizontale

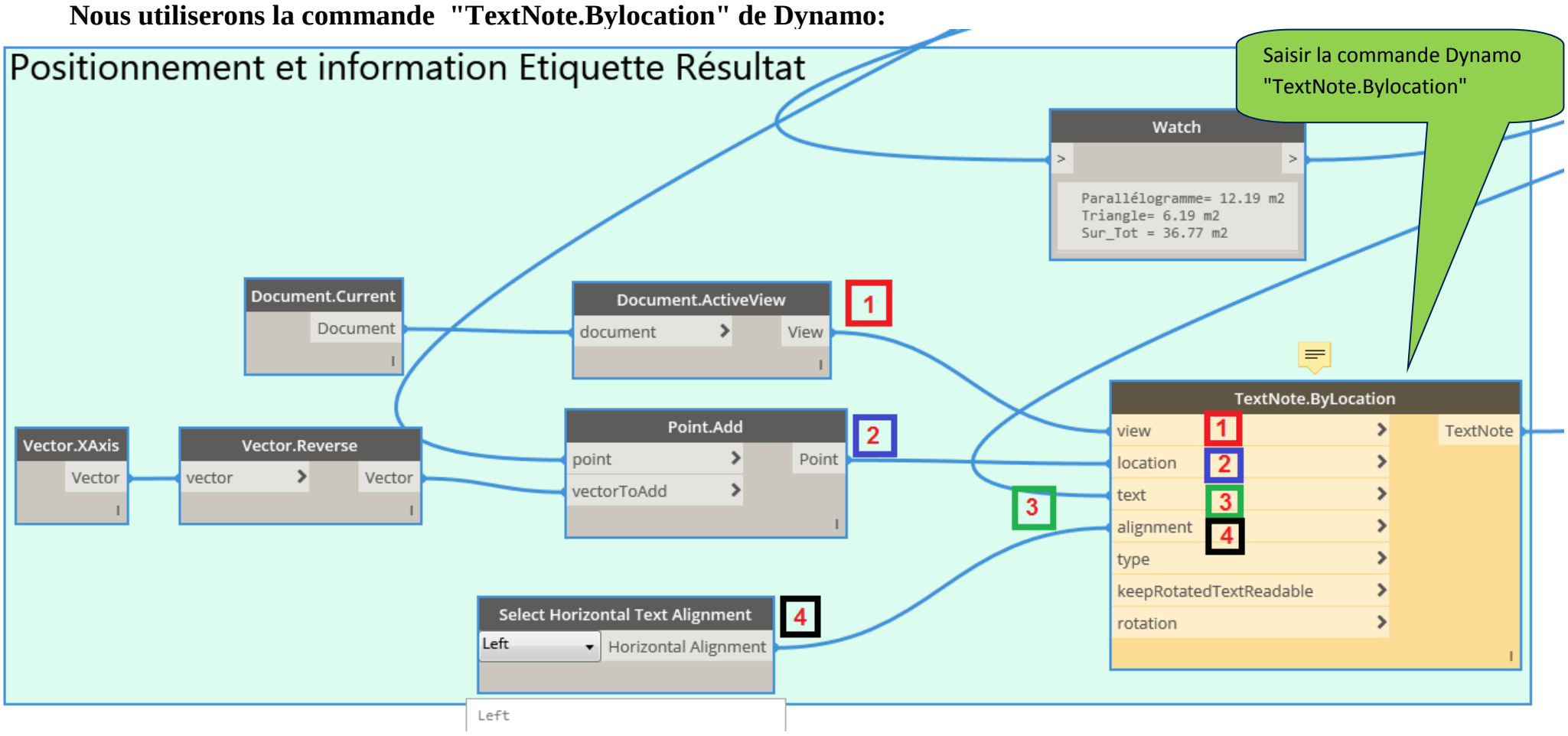


Résumer du code source PYTHON: (Etiquettes)



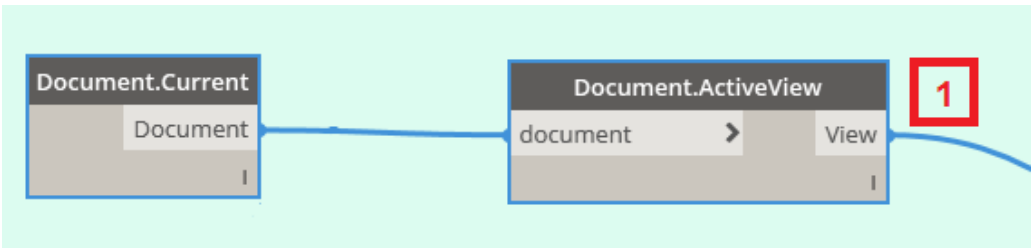
❖ 9ᵉ Etape POSITIONNEMENT pour afficher les Etiquettes surfaces avec Dynamo REVIT :

Nous utiliserons la commande "TextNote.Bylocation" de Dynamo:



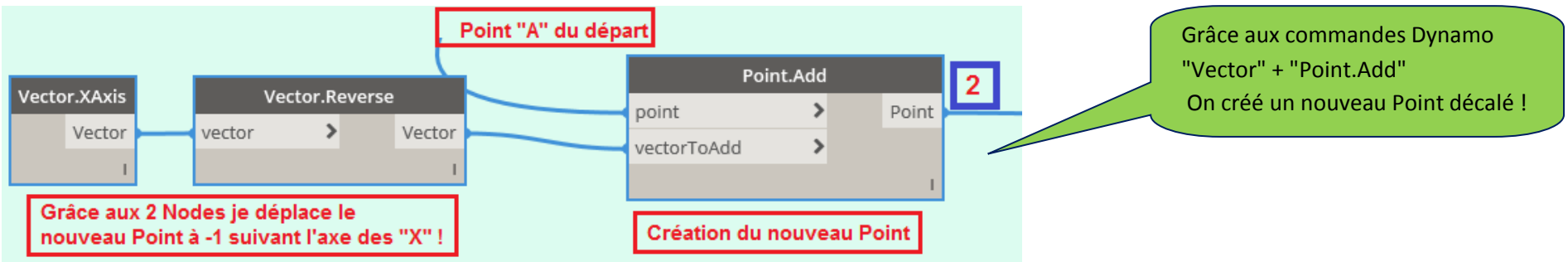
Entrée No 1: "View" (c'est à dire la vue dans laquelle sera affichée le Texte).

Ici nous aurons la vue active de REVIT.



Entrée No 2: "location" (c'est à dire la position du Texte).

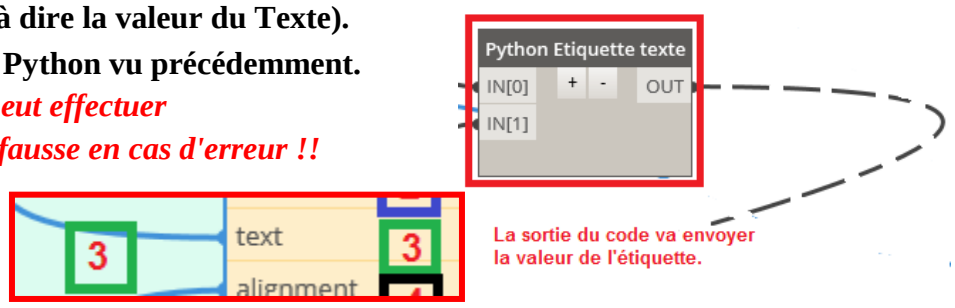
Ici nous aurons le Point d'origine "A" comme référence de décalage.



Entrée No 3: "text" (c'est à dire la valeur du Texte).

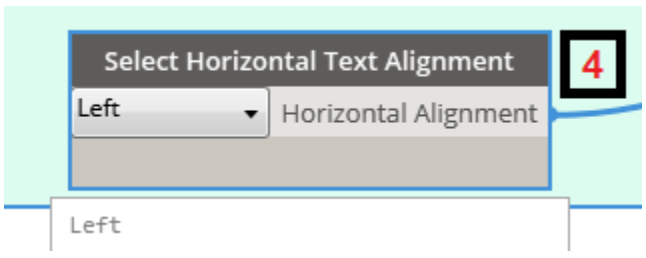
Valeur de sortie du Programme Python vu précédemment.

Nous verrons plus loin que l'on peut effectuer
un test pour indiquer une valeur fausse en cas d'erreur !!



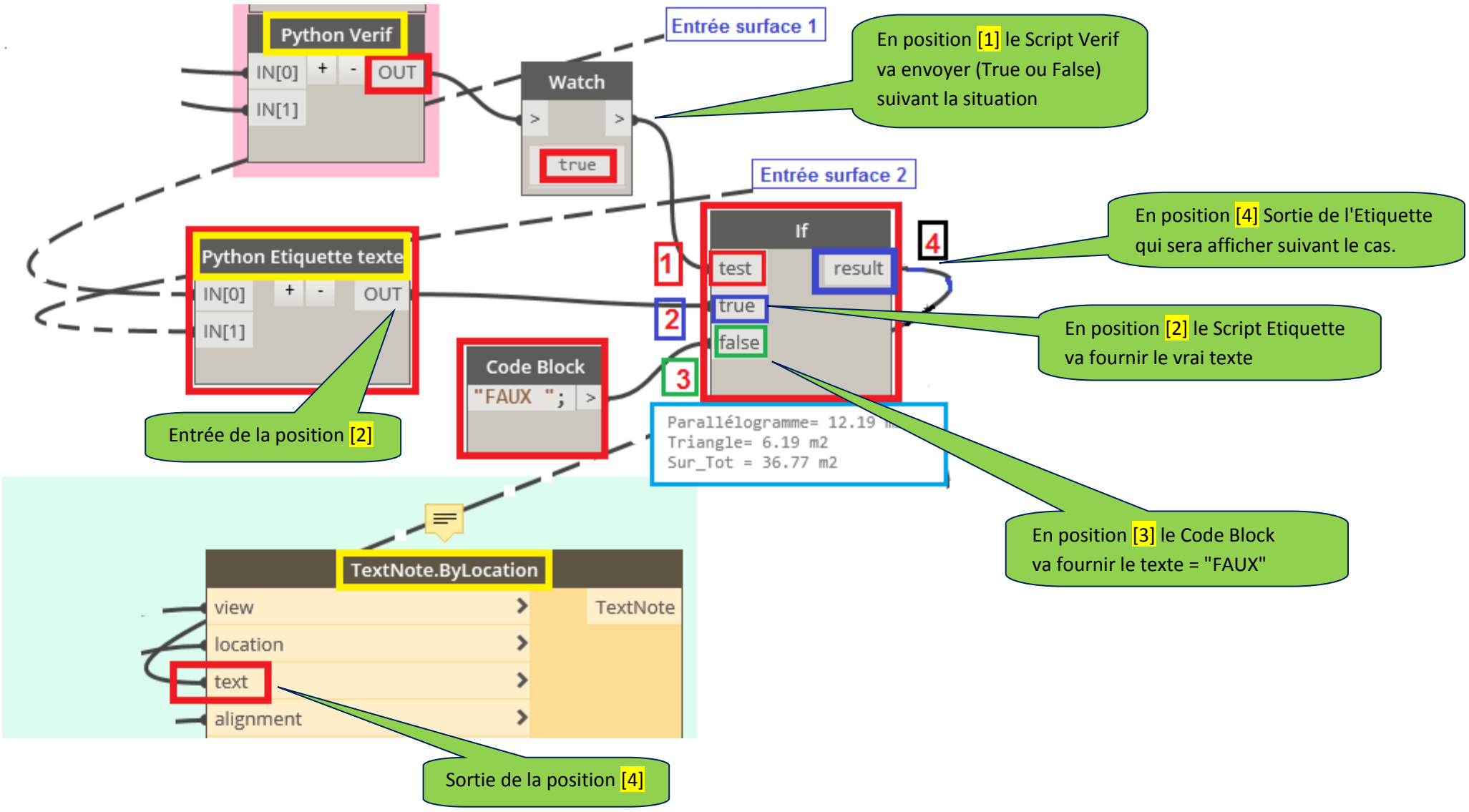
Entrée No 4: "alignment" (c'est à dire l'orientation du texte).

Valeur de sortie "Left". (*Qui peut être changée si on le souhaite*)



❖ 10e Etape Vérification d'erreur pour afficher un texte "FAUX" avec Dynamo REVIT :

Cette fois promis c'est la fin, j'utilise un code Dynamo ("If")
Quand on aime on compte plus .....
Petit rappel: Nous avons 2 codes Python (Vérif + Etiquette) et nous allons faire un affichage ("FAUX") en cas d'erreur.



❖ 11e Etape LANCEMENT du programme Dynamo pour REVIT

OUVRIR un Fichier REVIT 2019
Ici dans notre exemple
"Projet-Decoupe-Dalle-MCc.rvt"
Ouvrir le fichier Dynamo
Découpage-repartition-Dalle-MC-14.dyn

CONCLUSION:

Cet exemple peut certainement être amélioré et même simplifié.
Pour plus de simplicité vous pouvez utiliser un éditeur Python qui vous éviterai des erreurs et un débogage plus facile.
Le nombre de Scripts Python peut être diminué.
A vos claviers pour inventer un processus plus complexe par exemple un Script plus complet avec des surfaces quelconques,

Sources utiles :
https://www.python.org/
https://villagebim.typepad.com/
http://primer.dynamobim.org/index.html (En anglais)

SCHEMA GENERAL DU PROGRAMME