BTS MEC – OUTILS DE PROGRAMMATION

Fiche n°4 – Importer un fichier CSV et affecter des valeurs à des objets composant une maquette

*Cette fiche fait partie d’un ensemble de documents destinés à faciliter l’appropriation des outils de programmation par les enseignants du BTS MEC suite à la réforme mise en œuvre depuis septembre 2021.*

*La modélisation sous forme d’objets est un élément essentiel du BIM. La maquette 3D n’est donc que la partie visible de la grande quantité d’informations organisée dans le modèle. Les outils de programmation sont donc souvent utilisés pour interroger ou de renseigner les données contenues dans le modèle.*

*Cette fiche présente la mise en œuvre d’un processus destiné à importer des données stockées dans un fichier CSV puis à affecter les valeurs importées à des objets composant la maquette. Pour cela l’utilisation de scripts en langage Python est également abordée.*



*Auteur :*

*Vincent Lefort - Lycée Cantau Anglet*

*vincent.lefort@ac-bordeaux.fr*

*Fichiers fournis :*

* *Dossier «1 - Import CSV »*
  + *ImportDonnees.dyn*
  + *data\_wall.txt*
  + *data.txt*
* *Dossier « 2 - Affectation automatique CSV »*
  + *Import et affectation et prix.dyn*

***SOMMAIRE***

[1 Importer un fichier CSV 3](#_Toc106255676)

[2 Affecter une valeur à un paramètre avec Dynamo 4](#_Toc106255677)

[3 Application 6](#_Toc106255678)

[4 Affecter des paramètres issus d’un fichier CSV à des objets 7](#_Toc106255679)

[4.1 Démarche « classique » permettant d’éviter l’utilisation de Dynamo 7](#_Toc106255680)

[4.2 Mise en œuvre d’un programme Dynamo pour renseigner des prix unitaires dans une maquette 8](#_Toc106255681)

[4.2.1 Préparation de la maquette 8](#_Toc106255682)

[4.2.2 Premiers nœuds personnalisés programmés à l’aide d’un Script Pyhton 9](#_Toc106255683)

[4.2.3 Application guidée 12](#_Toc106255684)

[4.2.4 Application (pouvant être réalisée en autonomie) : 15](#_Toc106255685)

[4.2.5 Application (pouvant être réalisée en travaux dirigés) : 16](#_Toc106255686)

# Importer un fichier CSV

Les valeurs numériques contenues dans un fichier CSV peuvent être importées à l’aide du nœud « ImportCSV »

« ImportCSV »

* Ce nœud se trouve dans la catégorie IMPORTEXPORT => DATA
* Il est nécessaire de préciser en paramètre :
  + L’adresse du fichier avec le nœud « File Path » (catégorie IMPORTEXPORT => FILE SYSTEM)
  + L’éventuelle transposition ligne/colonne des données (par défaut les données sont lues en colonnes), à l’aide d’un paramètre booléen (vrai/faux ; 1/0 ; oui/non) qui peut être généré par le nœud « Boolean » (catégorie INPUT => BASIC).

Quelques remarques contenant le fichier CSV :

* Le fichier contenant les données doit être au format texte (pas de .doc ou de .rtf : un simple encodage Unicode de type UTF-8, obtenu par exemple avec le petit éditeur de texte Windows « Bloc note »).
* Les valeurs doivent être séparées par virgules.
* Si vous modifiez le fichier CSV et que vous voulez lire les données modifiées, il est nécessaire de sélectionner un autre fichier dans le nœud « FilePath », puis de re-sélectionner le fichier initial afin de forcer sa relecture par Dynamo.

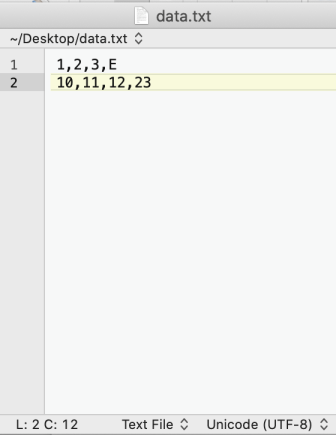
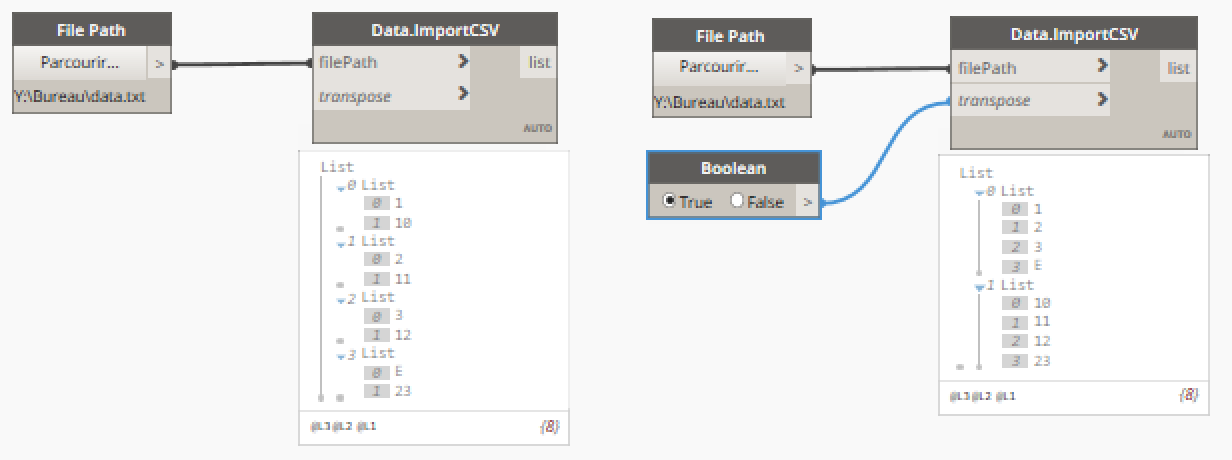
 

Figure - Fichier au format CSV (data.txt), nœud Dynamo permettant l'importation des données (avec ou sans transposition des données)

# Affecter une valeur à un paramètre avec Dynamo

Pour lire la valeur d’un paramètre, on utilise le nœud « Get Parameter Value By Name » présenté précédemment.

Pour modifier la valeur d’un paramètre, on utilise le nœud « Set Parameter By Name »

« Set Parameter By Name »

* Ce nœud se trouve dans la catégorie REVIT => ELEMENTS => ELEMENT
* Il permet d’affecter une valeur au paramètre d’un élément (ou d’une liste d’éléments)
* Il est nécessaire de préciser en paramètre :
  + Le ou les éléments concernés (par exemple avec le nœud « All Elements of Category »
  + Une chaine de caractères (String) égale au nom du paramètre à modifier
  + La valeur du paramètre à affecter

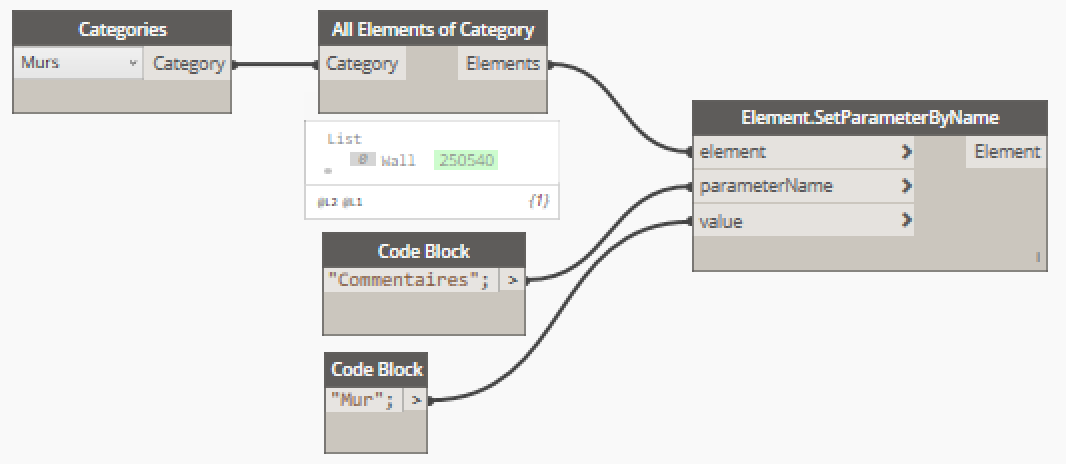
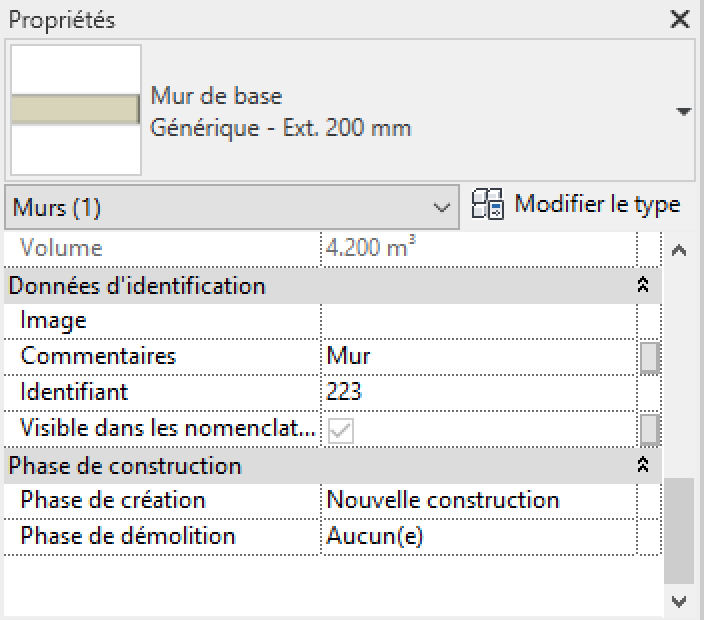
 

Figure - Affectation d'une valeur à un paramètre d'objet à l'aide du nœud "Set Parameter By Name"

**Point de vigilance sur les types de données manipulées**

Lors de l’importation, REVIT attribue automatique les types de variables adéquats aux données importées.

On peut contrôler le type d’un objet à l’aide du nœud « Type »

« Type »

* Ce nœud se trouve dans la catégorie INPUT => OBJECT
* Il permet de visualiser le type de la variable manipulée

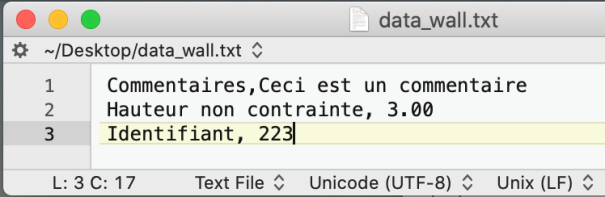
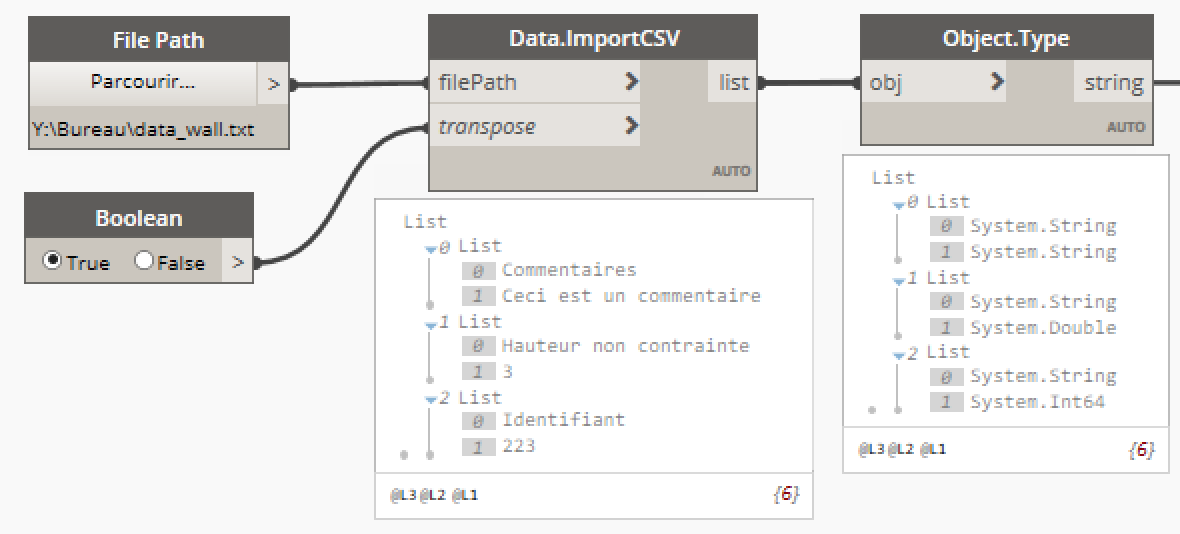
 

Figure - Identification des types de données manipulées à l'aide du nœud "Type" (INPUT => OBJECT)

Les types usuels sont :

* Les chaines de caractères : STRING
  + Création possible avec le nœud « String », ou bien avec un nœud « Code block » indiquant la valeur entre guillemets
* Les nombres réels : DOUBLE (car codés sur 8 octets et non 4 octets pour gagner en précision)
* Les nombres entiers : INTEGER (ou INT)
* Les booléens (TRUE/FALSE) : BOOL

Il faut être vigilant aux types des variables manipulées lorsque l’on souhaite affecter une valeur à un paramètre.

Par exemple, le champ « Commentaires » doit être rempli avec une variable de type STRING (ici 3.00 est un nombre réel donc un DOUBLE: le code va générer une erreur).

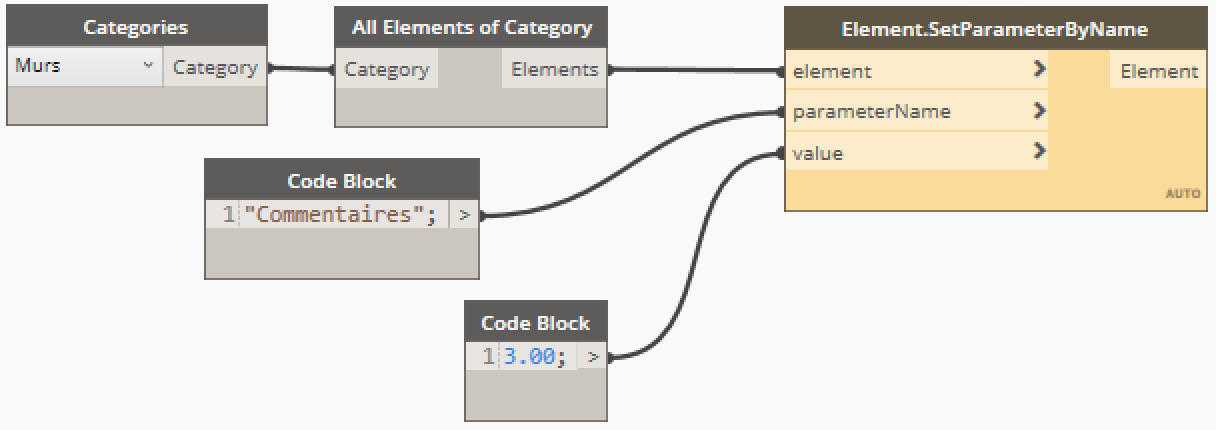


Figure - Erreur lors de l'affectation d'un DOUBLE au champ "Commentaires" => Le nœud apparait en jaune

Par exemple, le champ « Hauteur non contrainte » doit être rempli avec un DOUBLE (ici « 3 » est une chaine de caractères donc un STRING : le code va générer une erreur).

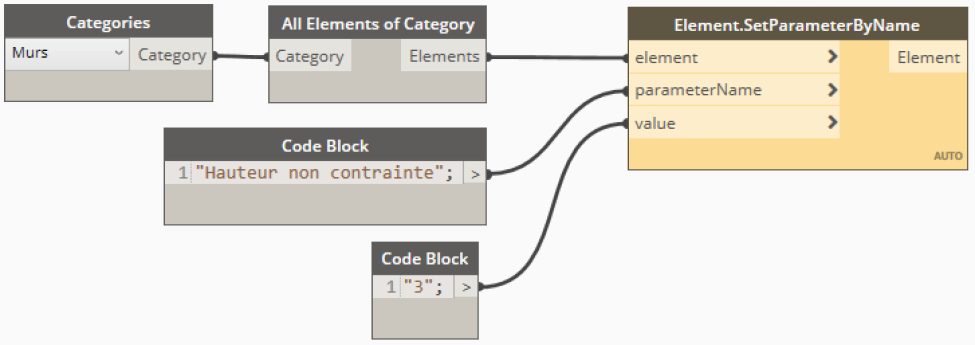


Figure - Erreur lors de l'affectation d'un STRING au champ "Hauteur non contrainte" => Le nœud apparait en jaune

# Application

**Objectif**

Un fichier CSV vous est fourni afin d’alimenter une maquette REVIT à l’aide de Dynamo. Ce fichier comporte pour chaque ligne le nom du paramètre à renseigner et la valeur associée.

A l’aide des nœuds vus précédemment, affectez à l’ensemble des murs composant la maquette la valeur correspondante dans les paramètres.

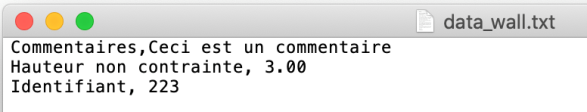
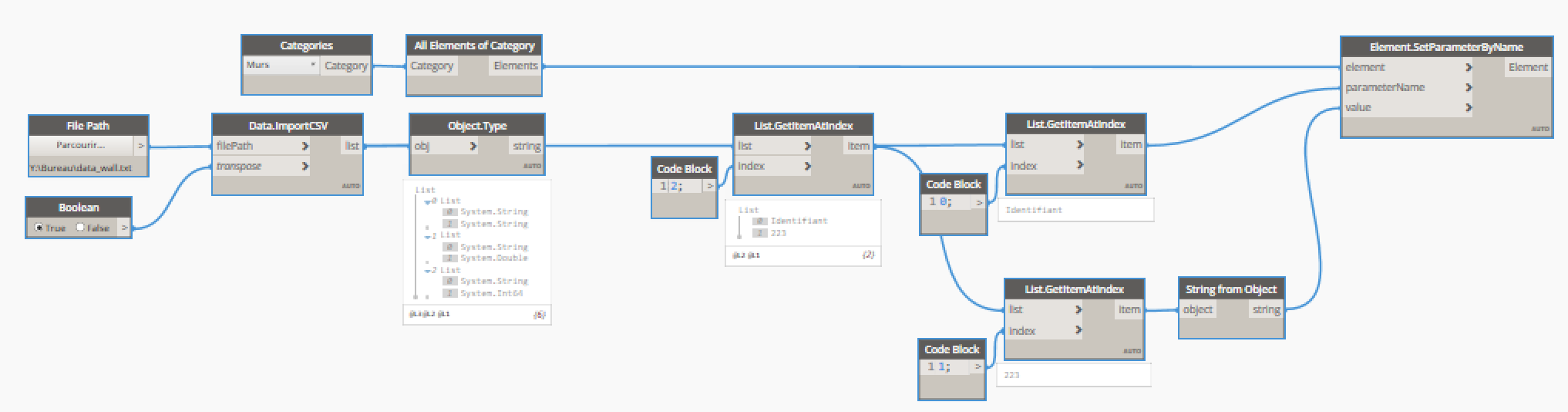


Figure - Fichier CSV (data\_wall.txt) fourni pour l’application

**Solution proposée**



# Affecter des paramètres issus d’un fichier CSV à des objets

## Démarche « classique » permettant d’éviter l’utilisation de Dynamo

L’outil « nomenclatures » permet d’afficher sous forme d’un tableau les valeurs de certains paramètres pour l’ensemble des objets composant une maquette.

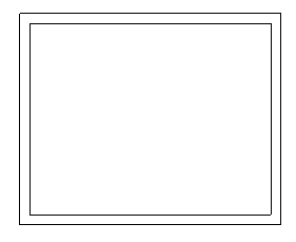
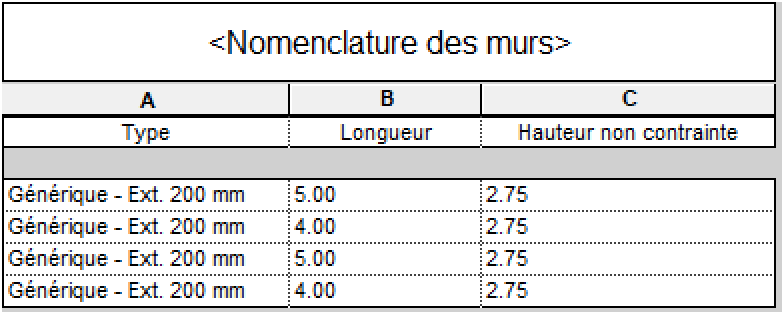
 

Figure – Vue de dessus d’une maquette REVIT comportant 4 murs, et nomenclature des murs associée à la maquette générée par REVIT

La gestion des données à l’aide des nomenclatures crée un lien naturel vers les tableurs que nous utilisons souvent dans le management des projets (métrés, caractéristiques, etc.). Il est donc rapidement utile de pouvoir relier les données de la maquette aux données stockées en dehors de la maquette sous forme de tableaux (DPGF, nomenclatures d’objets, etc.).

Pour cela des plugins gratuits existent. Citons par exemple :

* SheetLink de Diroots (https://diroots.com/revit-plugins/)
* Excel Import/Export de BIMOne (https://bimone.com/fr/Apps)

Mais il est aussi possible de développer ses propres outils, par exemple à l’aide de Dynamo…

## Mise en œuvre d’un programme Dynamo pour renseigner des prix unitaires dans une maquette

### Préparation de la maquette

L’établissement de sous-détails de prix a permis de définir les prix unitaires des différents ouvrages élémentaires composant le projet. Nous souhaitons renseigner dans la maquette ces prix unitaires afin d’établir par la suite un DPGF.

Nous allons enrichir les objets composant la maquette de plusieurs paramètres :

* **« Code\_CCTP » :** permet de faire le lien entre la maquette et les documents contractuels (description des ouvrages élémentaires issue du CCTP, prix unitaire issu des sous-détails, etc.).

Donnée à renseigner manuellement dans la maquette pour créer le lien.

* **« unite\_prix » :** permet de préciser l’unité du prix unitaire (au mètre cube, mètre carré, mètre linéaire, forfait, etc.).

Donnée alimentée de façon automatique à partir du Code\_CCTP

* **« prix\_unitaire » :** valeur du prix unitaire

Donnée alimentée de façon automatique à partir du Code\_CCTP

Ces paramètres sont créés sous forme de paramètres partagés dans REVIT

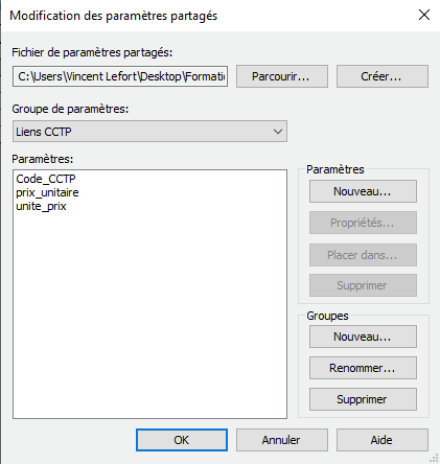
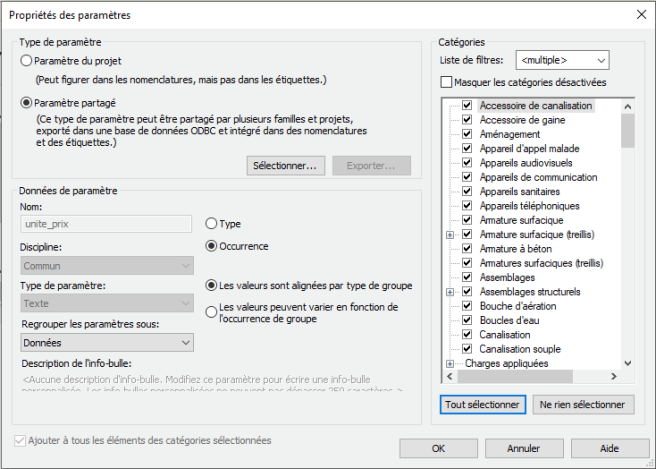
 

Figure - Création des paramètres utilisés : Création de paramètres partagés, puis affectation aux objets en tant que paramètre d'occurrence

On peut alors faire apparaître ces nouveaux paramètres dans les nomenclatures.

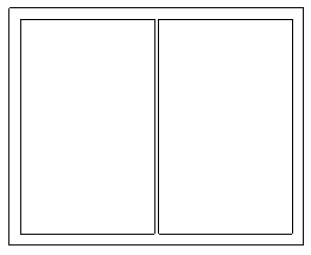
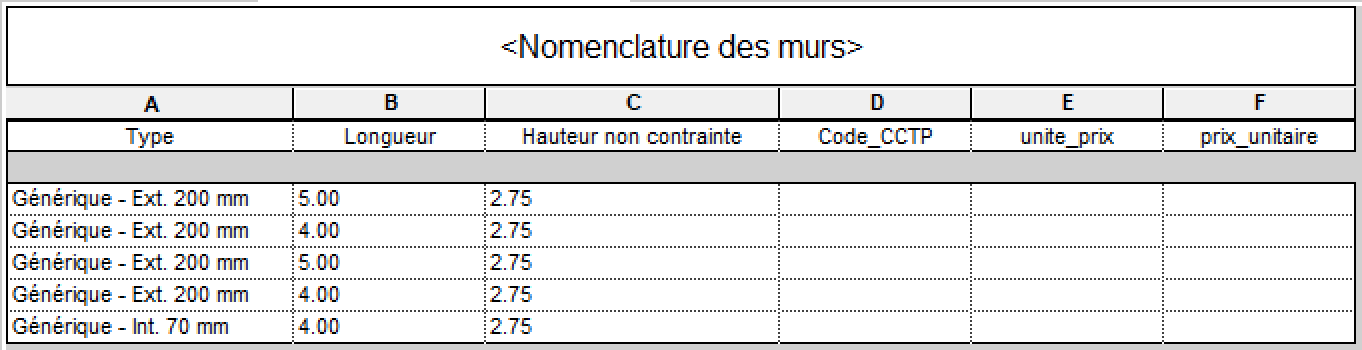
 

Figure – Vue de dessus d’une maquette REVIT comportant 4 murs et 1 cloison, et nomenclature des murs associée

Après avoir complété manuellement la colonne « Code\_CCTP », l’objectif est donc de faire en sorte que les colonnes « unite\_prix » et « prix\_unitaire » soit complétées automatiquement à partir des données issues des sous-détails de prix.

Ce travail nécessite d’aller un peu plus loin dans le domaine de la programmation, notamment en utilisant le langage Python, qui est un langage plus traditionnel de programmation.

### Premiers nœuds personnalisés programmés à l’aide d’un Script Pyhton



*Partie largement inspirée du paragraphe 10.4. Nœud Python (*[*https://primer.dynamobim.org*](https://primer.dynamobim.org)*)*

Python est un langage de programmation très répandu dont la popularité est liée à sa syntaxe très lisible, ce qui en facilite l'apprentissage. Python propose des méthodes beaucoup plus efficaces pour écrire des instructions conditionnelles (if/then) et des boucles. D’autre part le chargement de bibliothèques permet de profiter de fonctionnalités développées par d’autres utilisateurs.

La compétence visée ici n’est pas de rendre les élèves autonomes dans la programmation en langage python, mais qu’ils soient capables de comprendre la structure d’un script donné, puis de leur demander de le modifier légèrement pour atteindre un objectif fixé.

Comme les nœuds « Code Block » introduit précédemment, les nœuds « Python Script » sont une interface de script au sein d'un environnement de programmation visuelle. Le nœud Python se trouve sous SCRIPT => EDITOR.

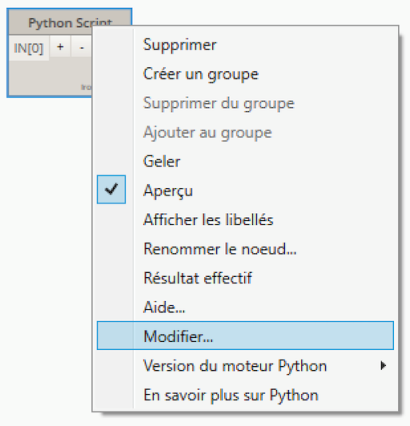
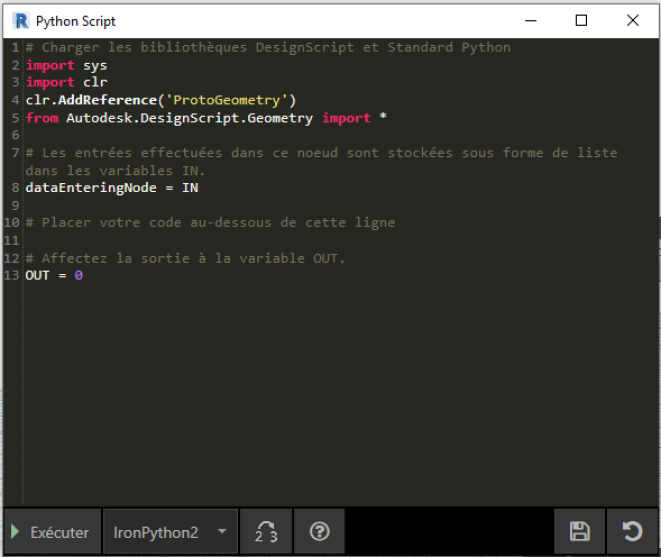
 

Figure - Nœud "Python Script" et éditeur de script associé

Les scripts sont composés de 2 parties :

1. Les premières lignes permettent de charger les bibliothèques permettant de disposer des fonctionnalités nécessaires (sys, clr, etc.).

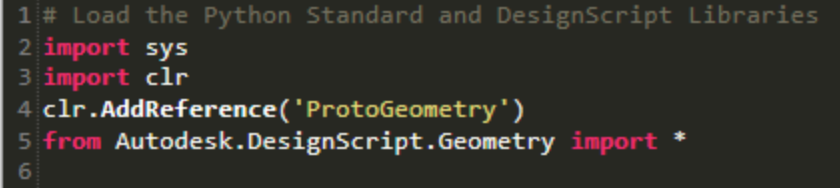


Figure - Script Python : chargement des bibliothèques

1. La suite du script permet de collecter les données d’entrée (stockées dans IN), de les traiter, puis d’affecter le résultat à la variable OUT.

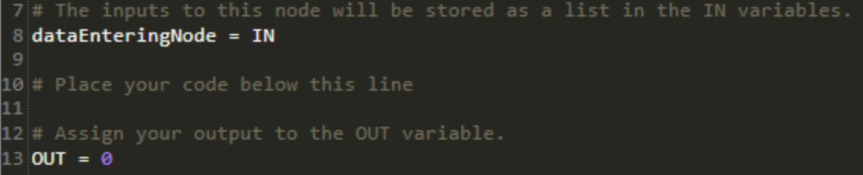


Figure - Script Python : trame pour la collecte des données, leur traitement et l’affectation du résultat

1. Les lignes qui débutent par un « # » sont des commentaires et ne correspondent pas à des instructions.

*Exemple 1 :*

Ci-dessous nous présentons un exemple tout simple consistant à définir un script multipliant un nombre par 3. Les lignes essentielles sont :

*triple = IN[0] \* 3.0*

*OUT = triple*

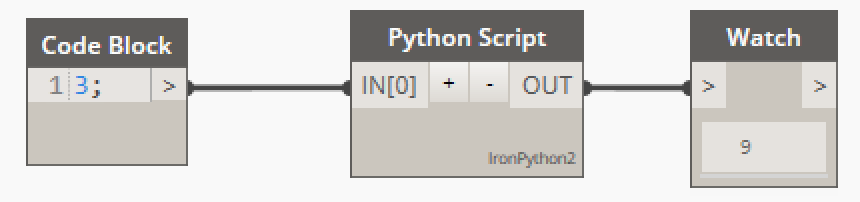
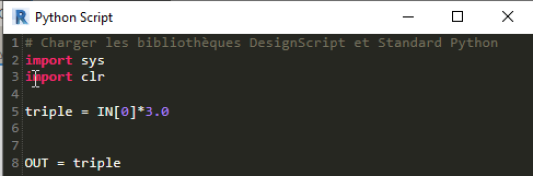
 

Figure - Nœud "Python Script" permettant de calculer le triple d'un nombre

*Exemple 2 :*

Autodesk propose une bibliothèque permettant d’interagir directement sur la maquette REVIT à partir de Python. Cette interaction directe se traduit par la mise en place d’un « TransactionManager » qui nécessite de charger des bibliothèques complémentaires. Les actions sur la maquette telles que la modification d’un paramètre ou la création d’un objet se font donc entre les lignes :

*TransactionManager.Instance.EnsureInTransaction(doc)*

*… instructions pour la modification de la maquette …*

*TransactionManager.Instance.TransactionTaskDone()*

En guise d’exemple, nous proposons un Script Python permettant d’affecter une valeur à un paramètre donné pour une liste d’objets. Les données d’entrée sont les suivantes :

* un objet, donné en IN[0],
* un nom du paramètre, donné en IN[1],
* une valeur du paramètre, donnée en IN[2].

Ce script comporte 2 commandes essentielles :

1. *ref\_param = elem.LookupParameter(nom\_param)*

* permet de récupérer l’adresse du paramètre « nom\_param » au sein des objets contenus dans la liste « elem »

1. *ref\_param.Set(val)*

* permet d’affecter la valeur « val » au paramètre stocké à l’adresse « ref\_param »

Enfin la commande *val\_param = ref\_param****.AsString****()* permet de lire la valeur du paramètre stocké à l’adresse « ref\_param » **sous forme d’une chaine de caractères**.

Si la valeur du paramètre est un **nombre**, il faut modifier cette ligne en *val\_param = ref\_param.****AsDouble****()*

Cette commande est utilisée ici pour vérifier que la valeur a bien été affectée.

|  |  |
| --- | --- |
|  |  |

Figure – à gauche : nœuds et Script Python permettant d'affecter la valeur "34" au paramètre "Identifiant" au premier mur d’une liste de murs  
à droite : palette des propriétés modifiée à l’aide du script (Identifiant : 34)

*Exemple 3 :*

Sur le tutoriel DynamoBIM, vous trouverez un exemple largement détaillé présentant la génération d’un motif de façade à l’aide de Script Python.

<https://primer.dynamobim.org/fr/10_Custom-Nodes/10-4_Python.html>

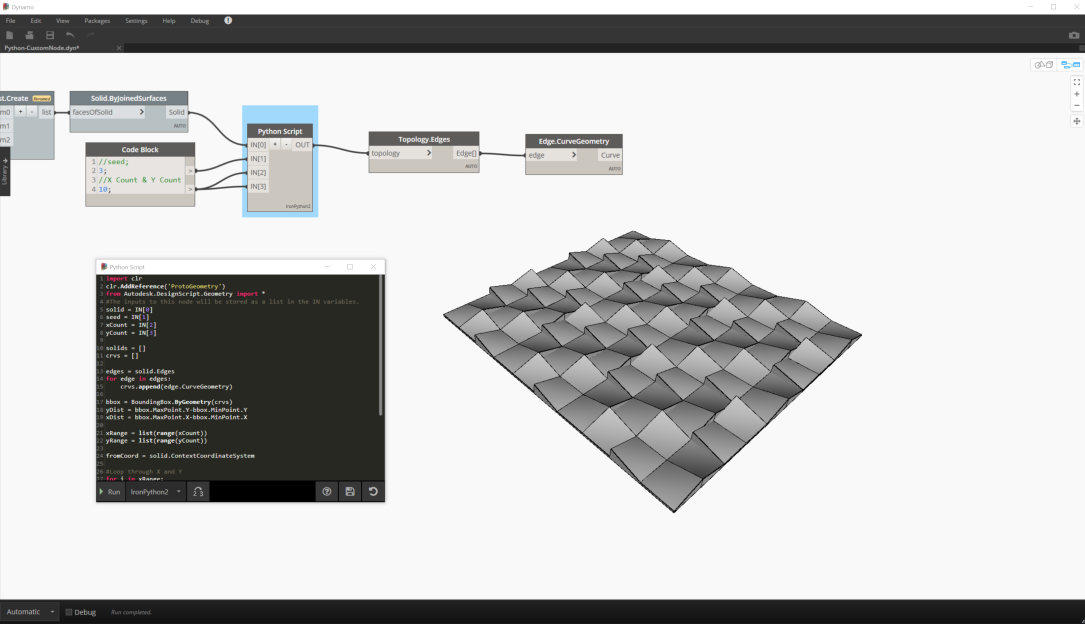


Figure - Copie d'écran issue du site Dynamo BIM présentant la création d'un script Dynamo générant des éléments de surface

### Application guidée

Nous disposons d’un tableau au format CSV comportant les informations issues par exemple de l’établissement des sous-détails de prix (Désignation, Code CCTP, unité, prix unitaire).

|  |  |  |  |
| --- | --- | --- | --- |
| **Désignation** | **Code\_CCTP** | **unite\_prix** | **prix\_unitaire** |
| Mur 200mm | A.2 | Surface | 48 |
| Cloison 70mm | A.5 | Surface | 35 |

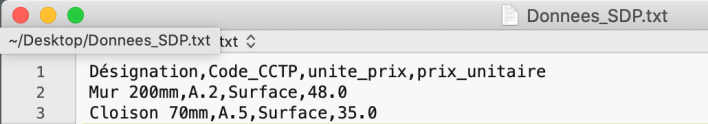


Figure – Données au format CSV à injecter dans la maquette (fichier non fourni)

#### Partie 1 : Préparation des données

A partir de la maquette REVIT et du fichier CSV nous allons générer grâce à Dynamo :

1. Une liste des objets présents dans la maquette (ici nous nous limitons aux catégories « Mur » et « Porte ». Nous mettons les objets à la suite dans une même liste à l’aide du nœud « List.Join »

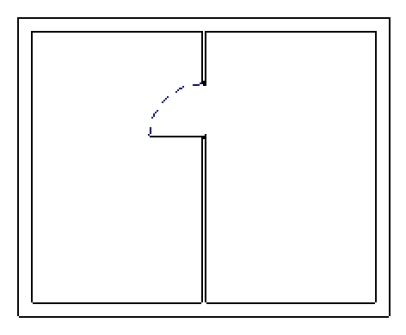
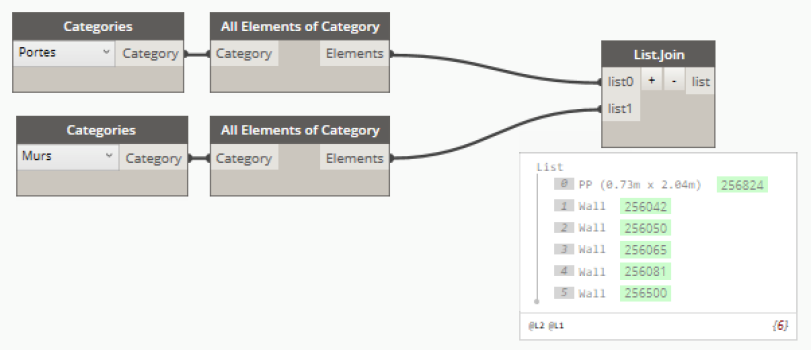
 

Figure - Génération de la liste des objets contenus dans la maquette sur lesquels le script va agir

1. L’intitulé du champ considéré (ici *Code\_CCTP*) et la liste des codes (ici *A.2* et *A.5*) utilisés pour identifier les objets sont importés à partir des données au format CSV. Nous utilisons les nœuds « List.FirstItem » et « List.RestOfItems » pour isoler respectivement le premier élément d’une liste et tous les éléments d’une liste autres que le premier élément.

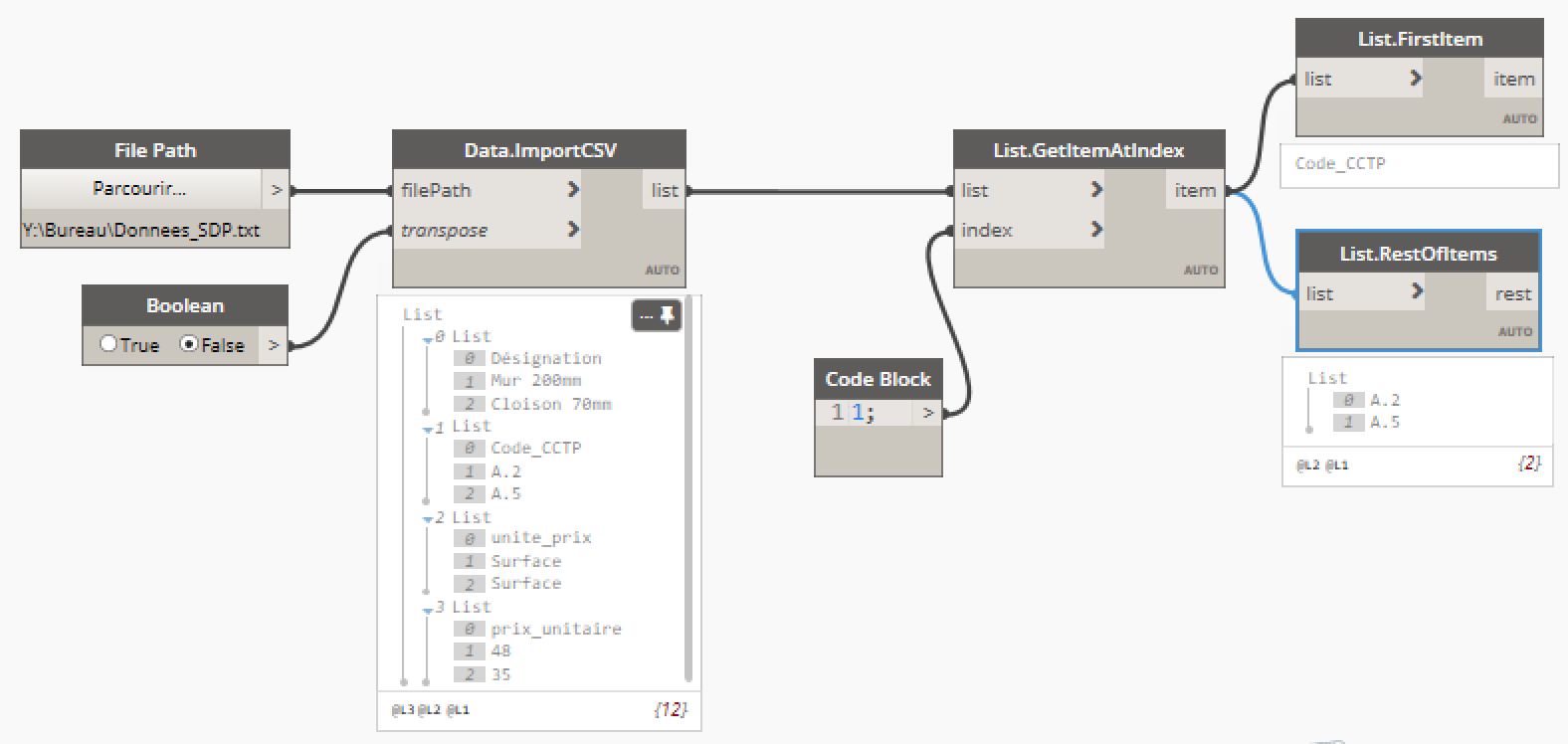


Figure - Importation des codes CCTP à partir du fichier CSV

1. L’intitulé du champ à affecter aux objets (ici *prix\_unitaire*) et la liste des codes (ici *48* et *35*) utilisés pour caractériser les objets sont importés à partir des données au format CSV de la même façon.

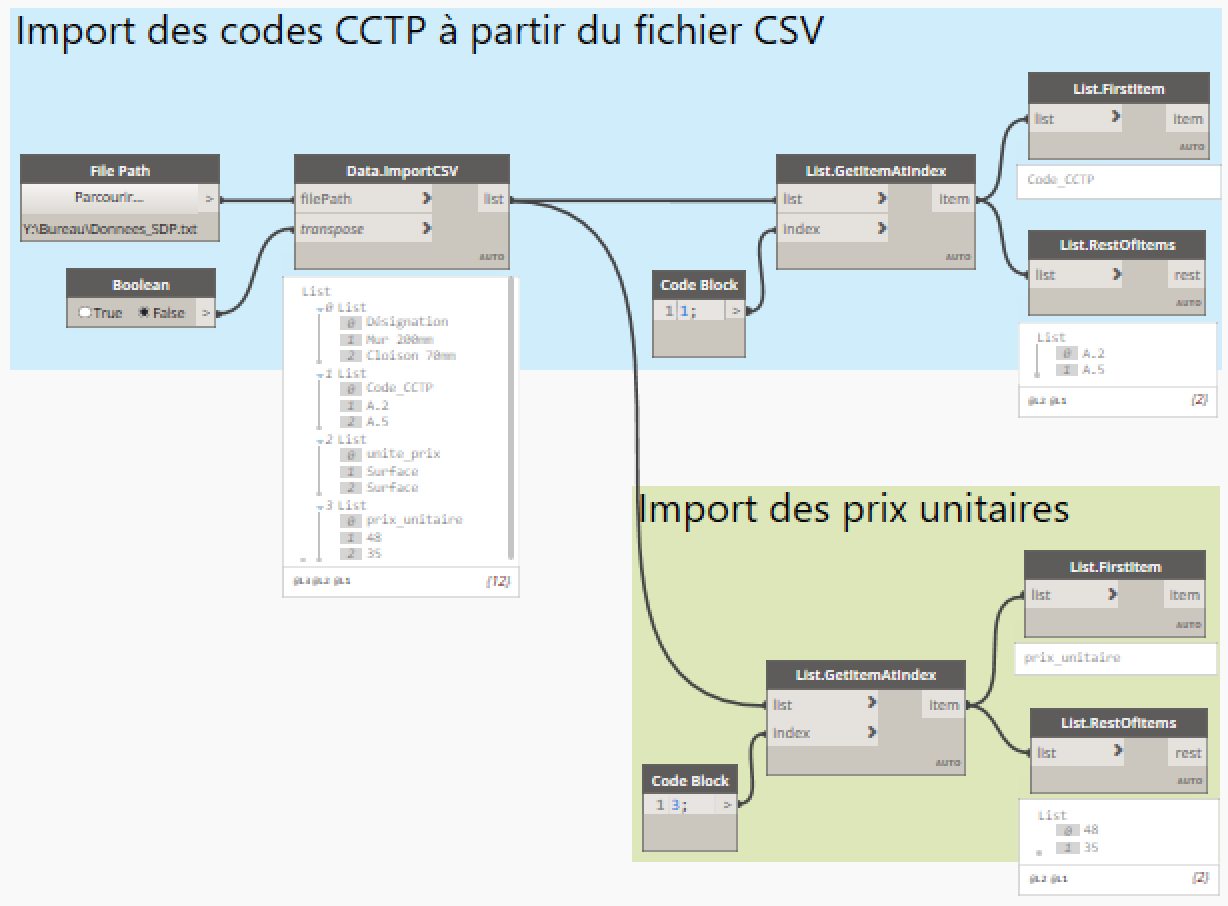


Figure - Importation des codes CCTP et des prix unitaires à partir du fichier CSV

#### Partie 2 : Traitement des données et affectation des valeurs aux objets à l’aide de Python

La suite du script est réalisée à l’aide d’un Script Python car nous avons besoin de structures de programmation de type boucles et conditionnelles.

L’algorithme proposé est le suivant :

Données :

1. **liste\_elem** : Liste des objets composant la maquette
2. **nom\_param\_id** : Intitulé du champ d’identification (ici *Code\_CCTP*)
3. **liste\_param\_id** : Liste des valeurs possibles pour le champ d’identification (ici *A.2* et *A.5*)
4. **nom\_param\_aff** : Intitulé du champ auquel des valeurs doivent être affectées (ici *prix\_unitaire*)
5. **liste\_param\_aff** : Liste des valeurs à affecter pour chaque valeur du champ identification (ici 48 et 35)

Procédure :

* Pour i variant de 0 à la taille de **liste\_elem**
  + Lecture de l’objet situé à la ligne « i » de la liste des objets **=> elem**
  + Lecture de la valeur du paramètre intitulé « **nom\_param\_id** » dans l’objet **elem => val\_param\_id**
  + Pour j variant de 0 à la taille de **liste\_param\_id**
    - Si la valeur du paramètre situé à la ligne « j » de la liste **liste\_param\_id** est égale à **val\_param\_id** alors
* On copie la valeur située à la ligne « j » de la liste **liste\_param\_aff**  dans le champ **nom\_param\_aff** de l’objet **elem**

Programmation de la procédure en langage Python :

for i in range(len(liste\_elem)):

elem = UnwrapElement(liste\_elem[i])

ref\_param\_id = elem.LookupParameter(nom\_param\_id)

val\_param\_id = ref\_param\_id.AsString()

ref\_param\_aff = elem.LookupParameter(nom\_param\_aff)

for j in range(len(liste\_param\_id)):

if str(liste\_param\_id[j]) == val\_param\_id:

ref\_param\_aff.Set(liste\_param\_aff[j])

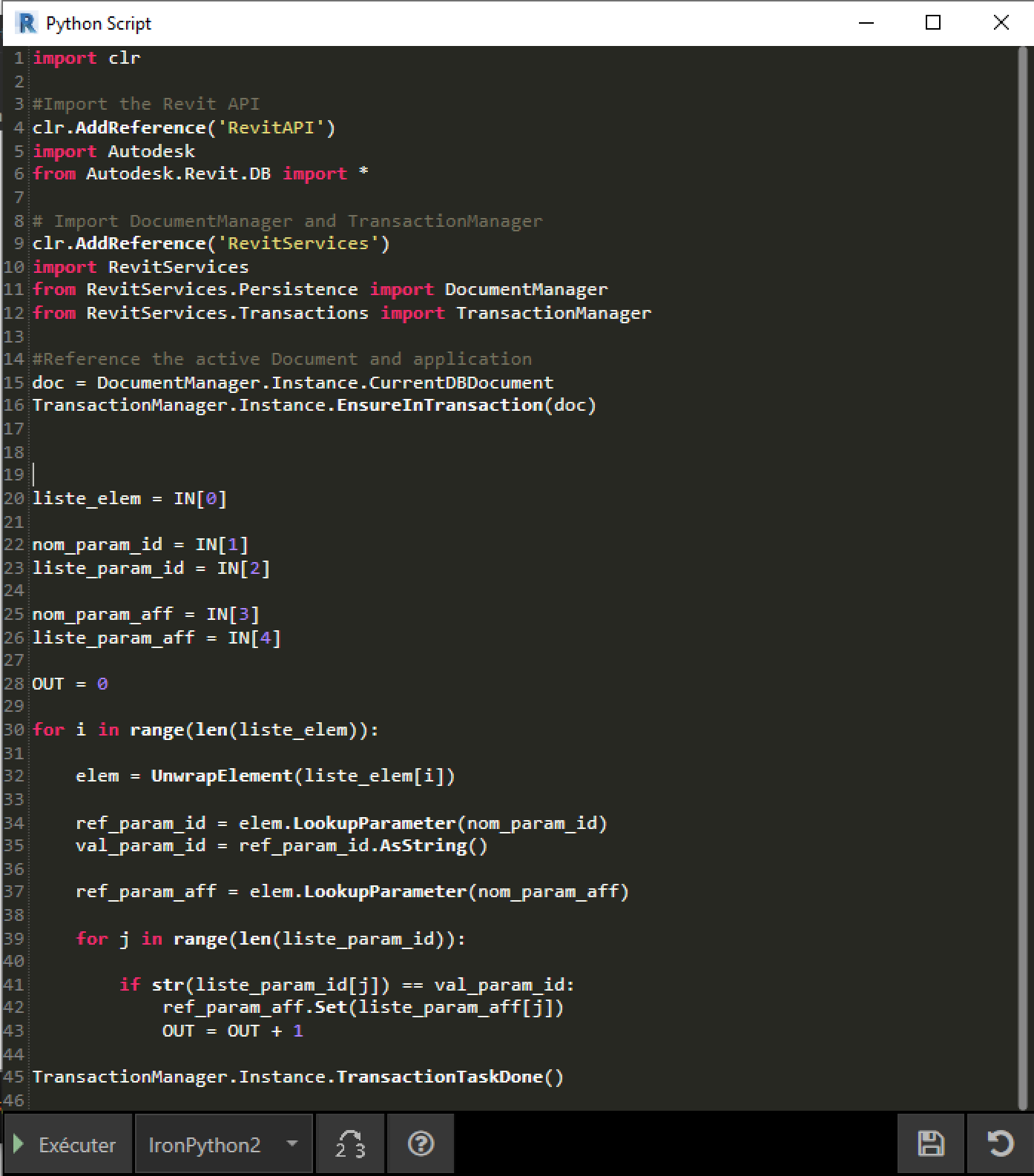


Figure - Script Python permettant l'affectation automatique de valeurs

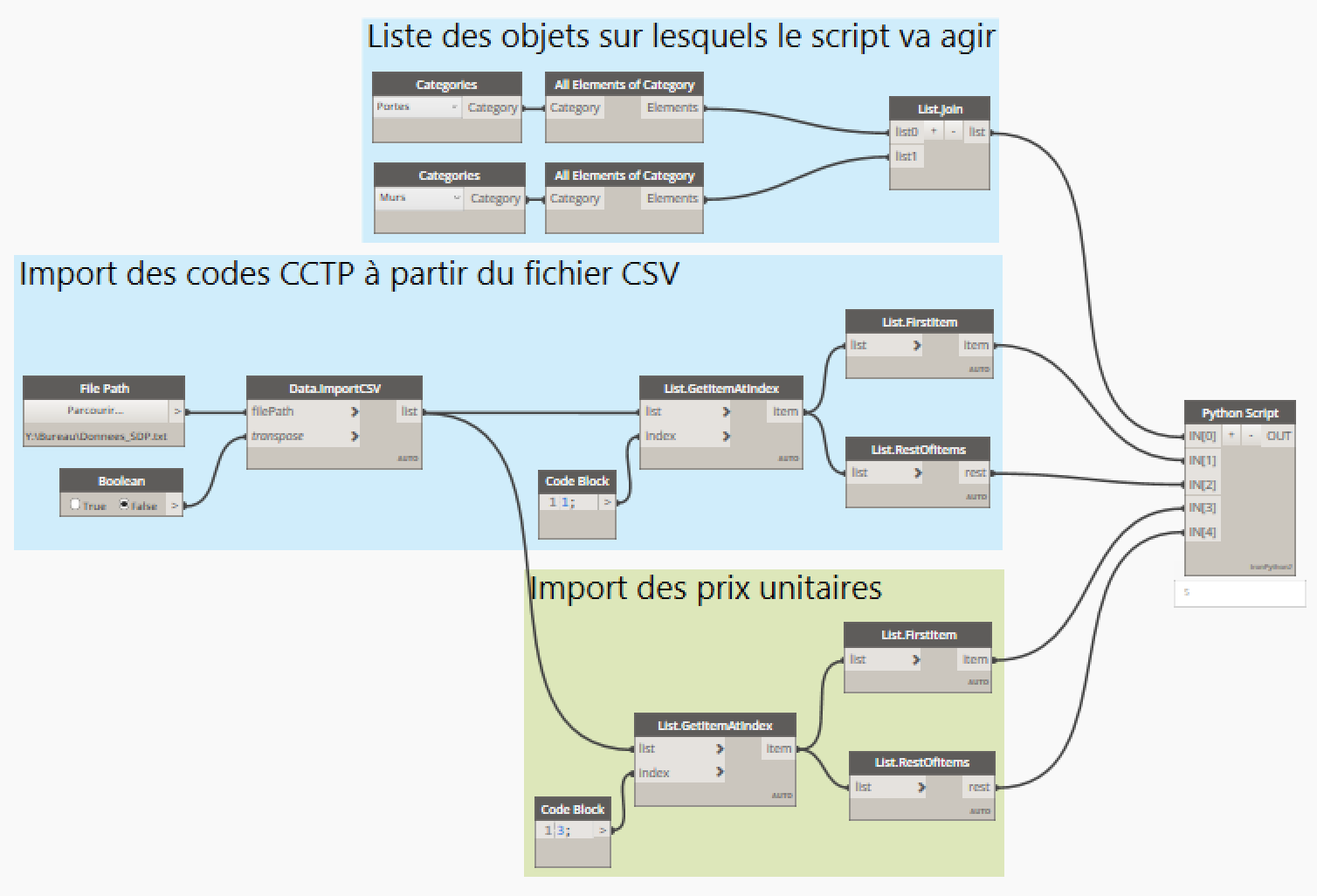


Figure - Alimentation du script Python à partir des données collectées dans Dynamo

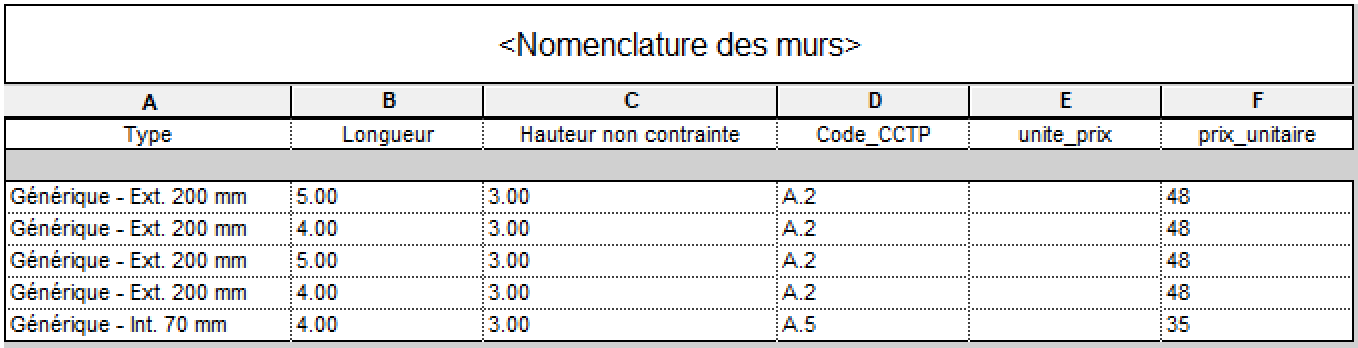


Figure - Nomenclature modifiée par l'outil Dynamo+Python à partir du fichier CSV

### Application (pouvant être réalisée en autonomie) :

A partir des éléments déjà vus, établissez un outil permettant d’affecter également les bonnes valeurs du champ unite\_prix aux objets de la maquette.

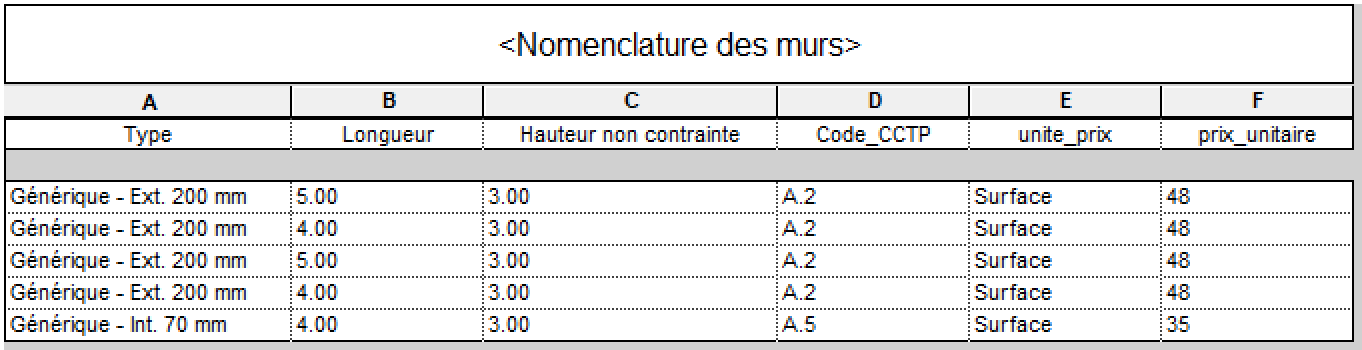
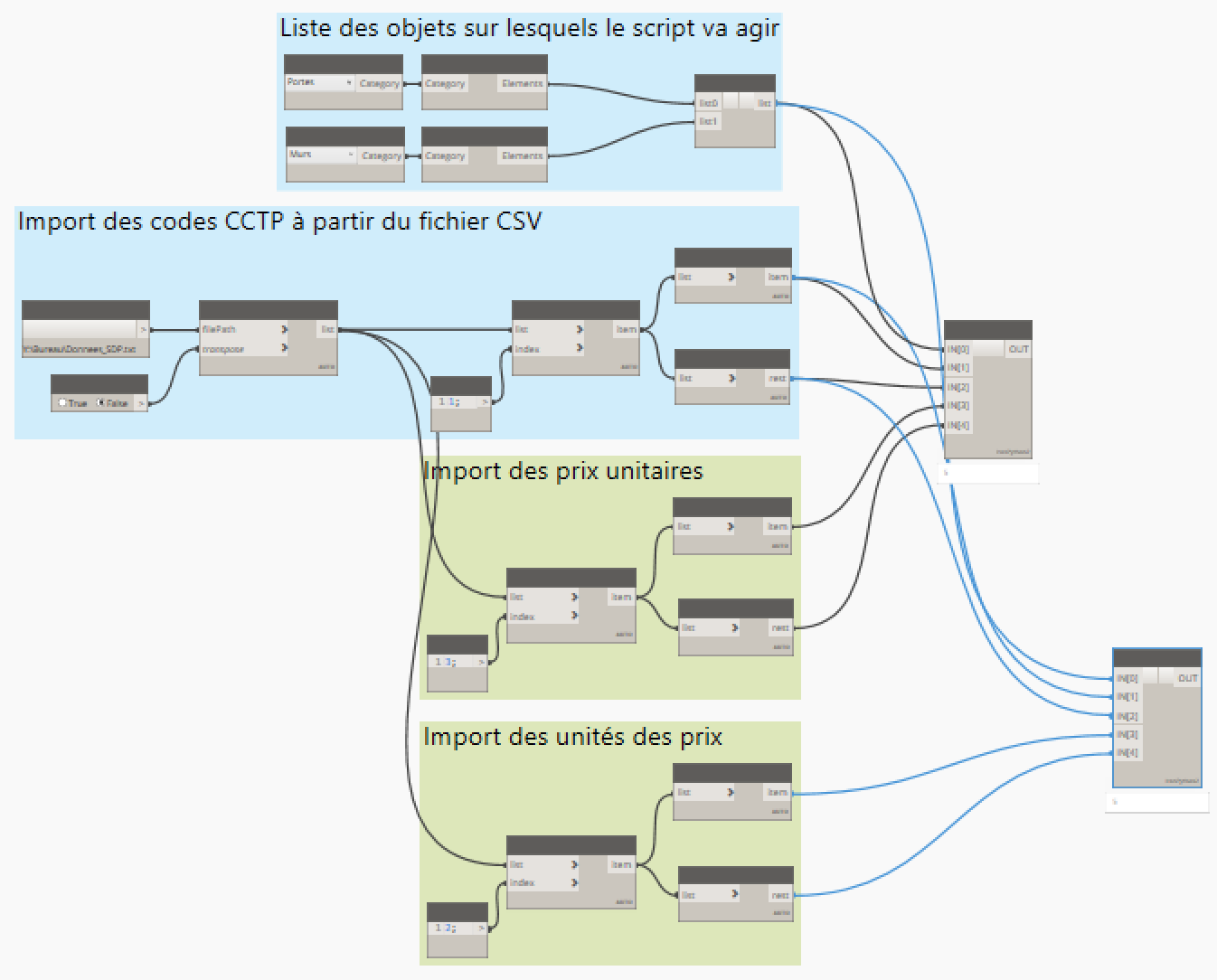


Figure - Nomenclature complétée en totalité à partir du fichier CSV

Proposition de solution



### Application (pouvant être réalisée en travaux dirigés) :

Nous disposons maintenant dans la maquette des prix unitaires (ainsi que leurs unités de calcul) et des quantités. Il nous reste à calculer le prix total.

Pour cela il faut :

* Créer un paramètre partagé « prix » à l’aide des fonctions de l’onglet « Gérer » de Revit
* Créer un script Python qui va calculer la valeur du paramètre « prix » à partir du prix unitaire et de la quantité issue de Revit (selon le cas : Longueur, Surface ou Volume).

**Attention : REVIT lorsqu’il est interrogé via Python retourne des valeurs en unités impériales…**

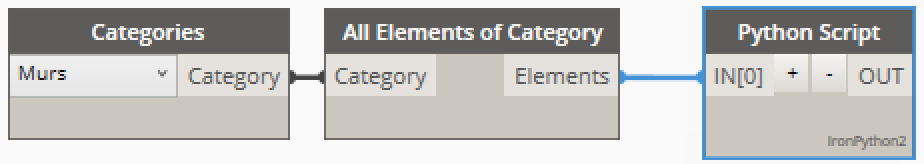
Pour récupérer les quantités de la maquette au sein du script Python, vous aller utiliser une commande du type :

***- reference\_parametre*** *=* ***element.****LookupParameter(****“nom du parametre”****)*

***- valeur\_parametre*** *=* ***reference\_parametre.****AsDouble()*

Cette commande, lorsqu’elle est appliquée aux paramètres Longueur, Surface, Volume, retourne la valeur en pieds (linéaire, carré ou cube selon le cas). Il faut donc penser à insérer le facteur de conversion adapté (1/3,28 ou 1/3,282 ou 1/3,283).

Proposition de solution



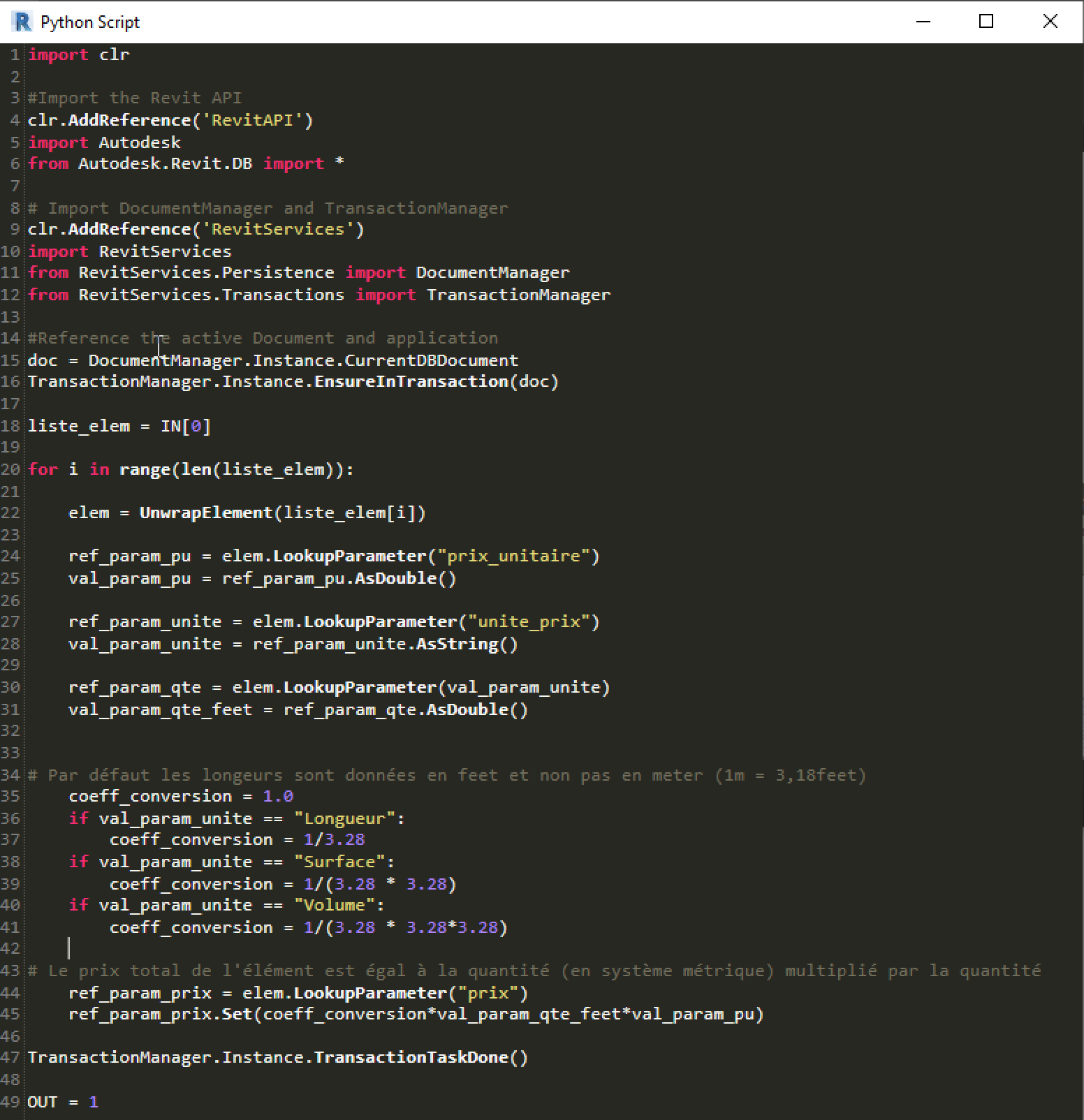


Figure - Proposition de script permettant de calculer le prix de l'objet à partir de son prix unitaire

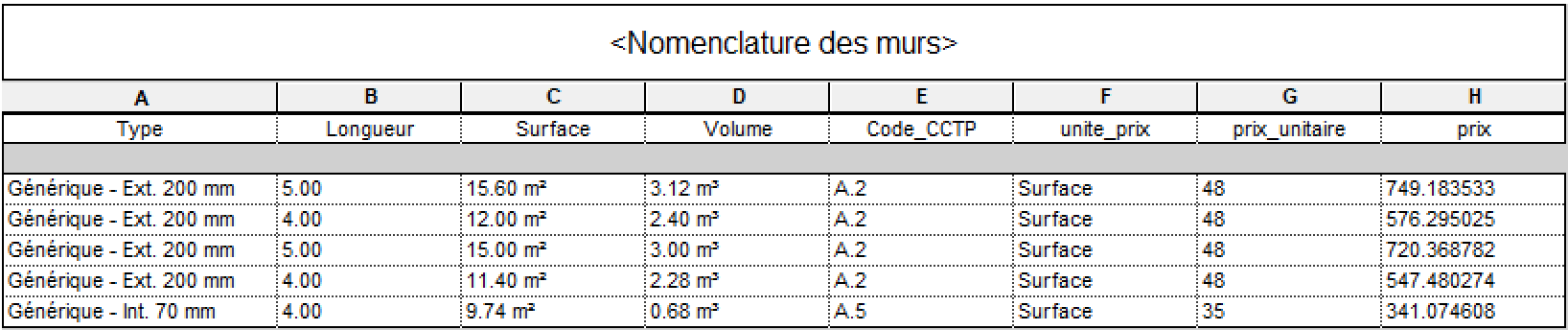


Figure - Nomenclature complétée à partir du script proposé

Pour aller au-delà, sauriez compléter le script pour gérer des objets dont le prix est donné par unité, par forfait, ou par masse ?