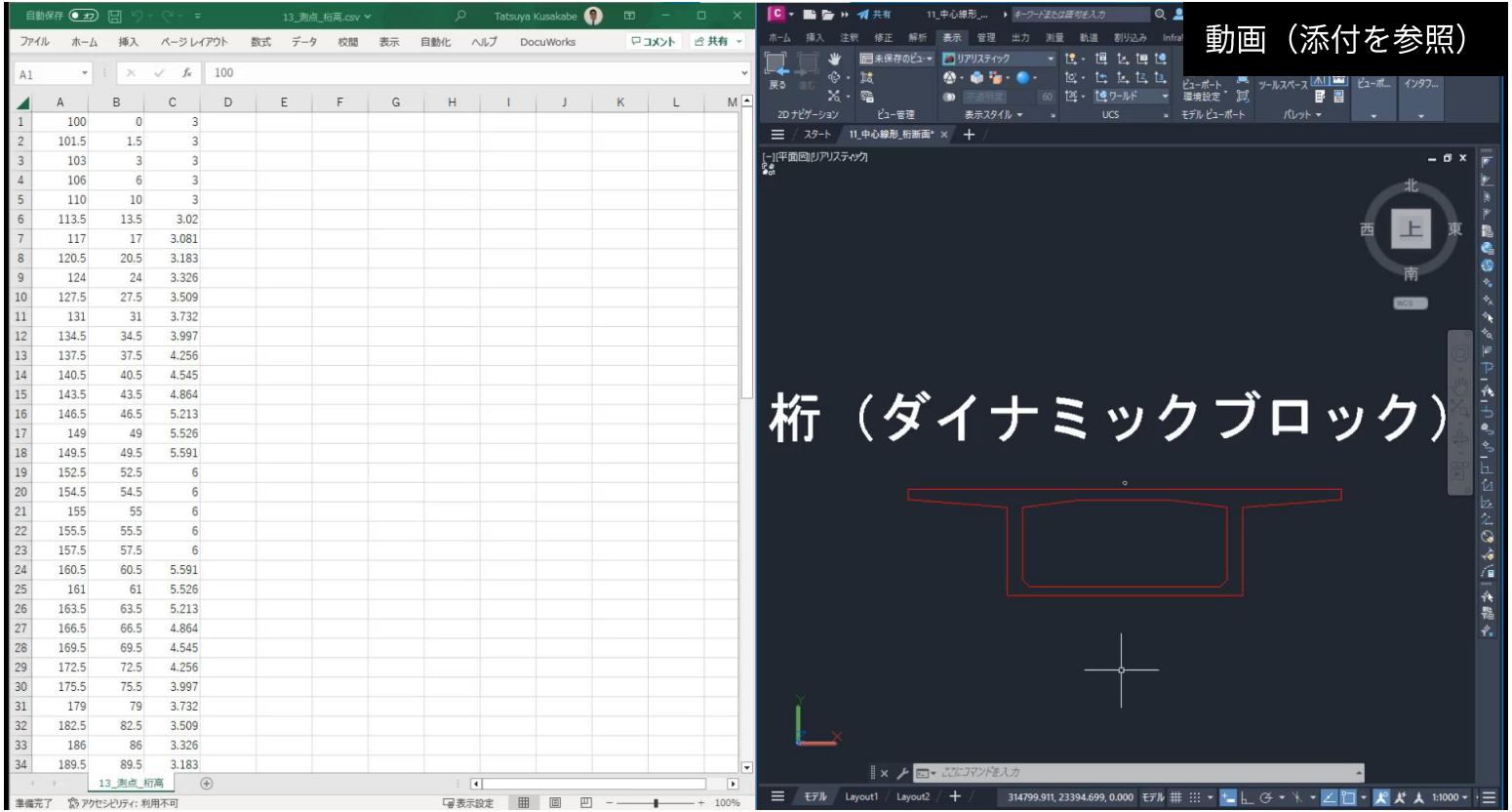
Dynamo サンプル解説

橋梁モデルの自動作成

Dynamo サンプル

測点と桁高の csv と、桁断面のブロックから、橋梁モデルを自動作成

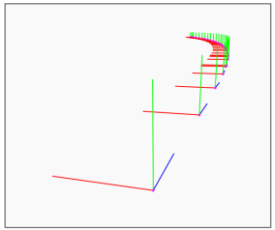


先ほどの Dynamo サンプルの中身を、細かく解説

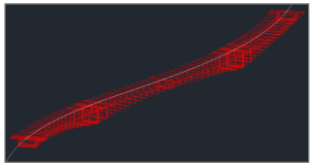
- 含まれる内容
  - csv との連携
  - Civil 3D との連携
    - 平面線形、縦断線形、ブロック
  - リストの編集
  - 図形（ジオメトリ）の作成
  - 座標系の作成・編集
  - パッケージの活用
  - Python スクリプトの活用

- データセット・デモ動画
  - https://autode.sk/3NerorC

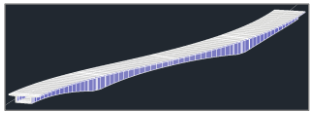
- データセットは、下記に分かれています
  - 練習１）Civil 3D の縦断線形に沿って、座標系を作成
    - 解説１）Civil 3D 平面線形＆縦断線形 – 取得
    - 解説２）Civil 3D 平面線形＆縦断線形 – 計算
    - 解説３）Excel / csv との入出力、リストの編集
    - 解説４）座標系の編集



  - 練習２）Civil 3D の縦断線形に沿って、ブロックを配置
    - 解説５）AutoCAD ブロック定義
    - 解説６）AutoCAD ブロック参照



  - 練習３）配置したブロックからソリッドを作成
    - 解説７）Civil3DToolkit パッケージの活用：
      ブロック参照を Dynamo ポリラインに変換
    - 解説８）Python スクリプトの活用：
      Dynamo ポリラインをソリッドに変換



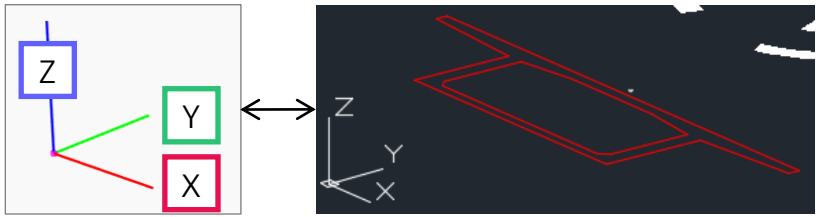
練習１）Civil 3D の縦断線形に沿って、座標系を作成

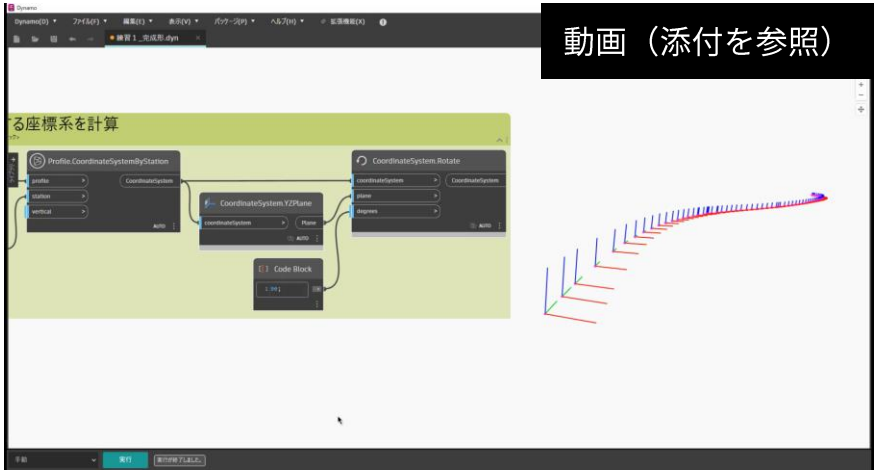
完成形を先にお見せします

- 手順
  - Civil 3D から、縦断線形を取得
  - csv から、ブロックを配置する測点を取得

	測点		桁高
	A	B	C
1	100	0	3
2	101.5	1.5	3
3	103	3	3

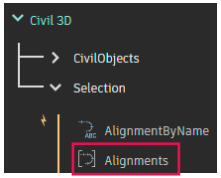
  - 縦断線形と測点に合わせ、座標系を配置
  - 座標系を回転
    - 座標系の XY 平面 ＝ ブロックの XY 平面

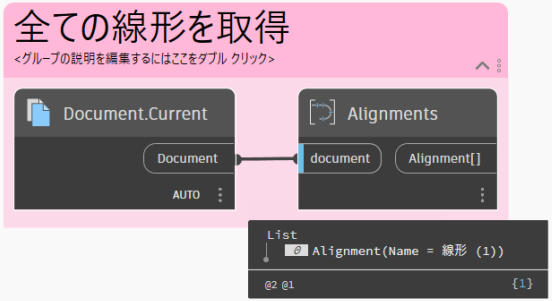


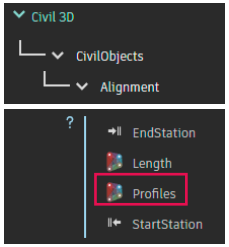


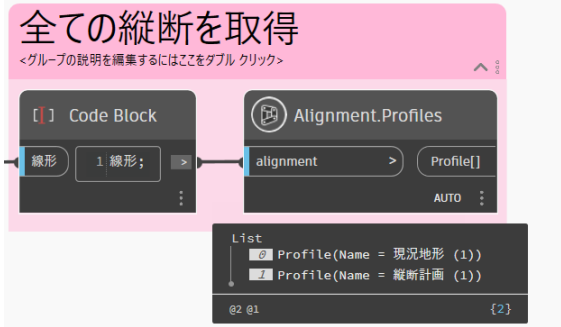
解説１）Civil 3D 平面線形＆縦断線形 – 取得

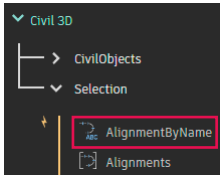
平面線形 や 縦断線形 を取得するには…

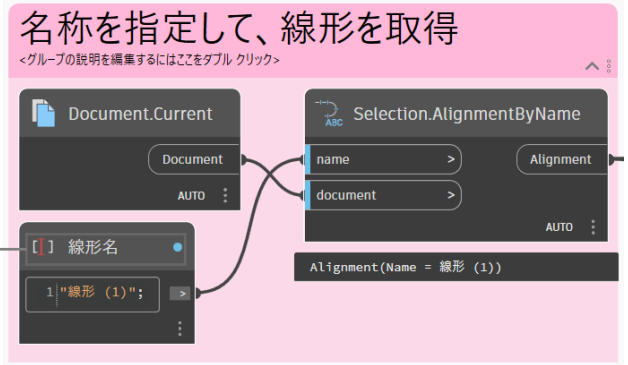


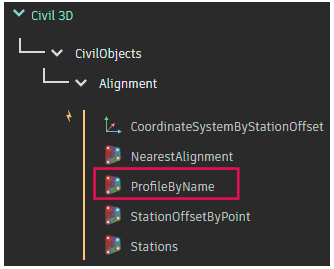


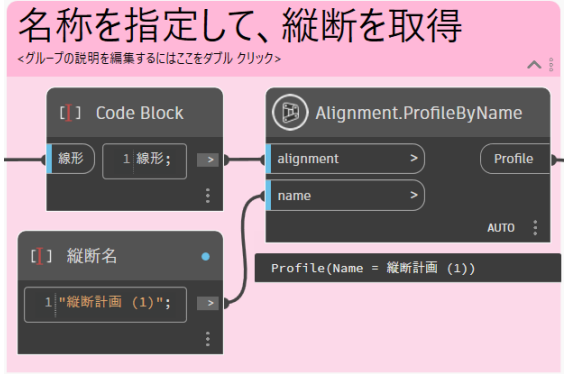








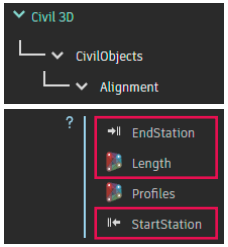


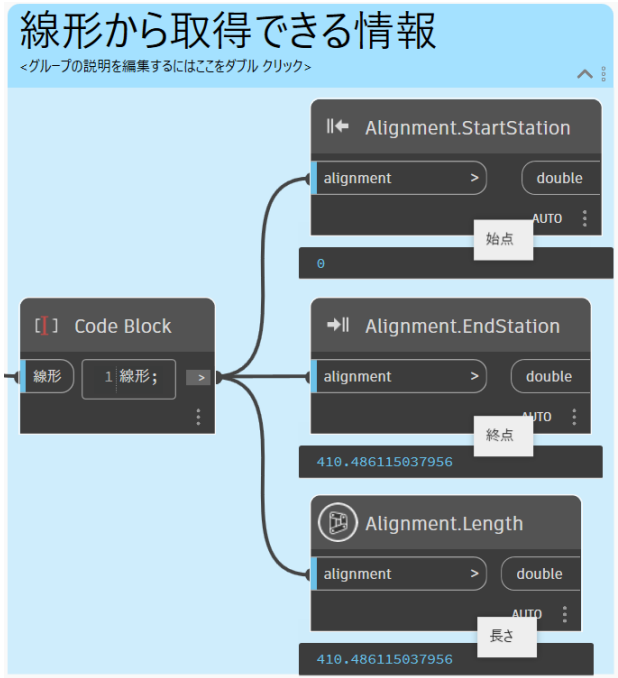


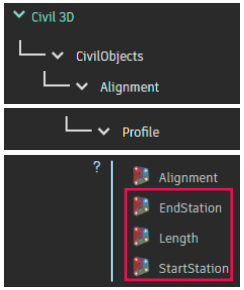
Code Block から名称を変更
→ ● マークがつく

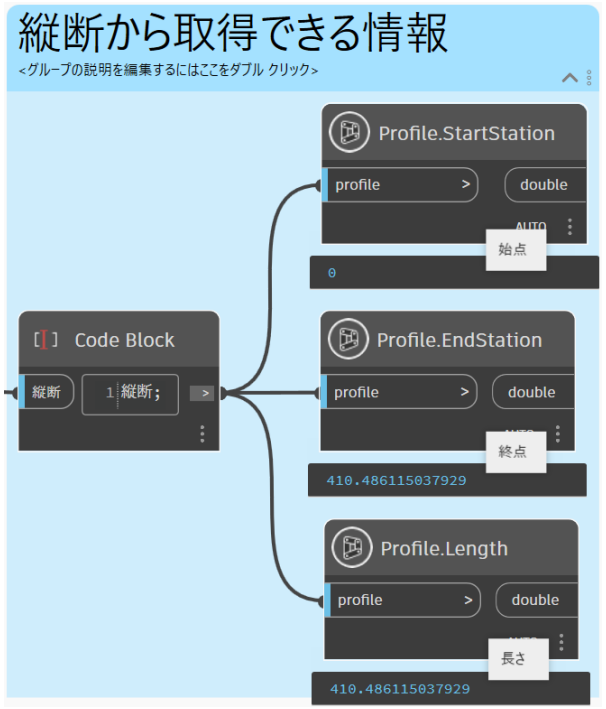
解説１）Civil 3D 平面線形＆縦断線形 – 取得

平面線形 や 縦断線形 から、そのまま抜き出せる情報は…



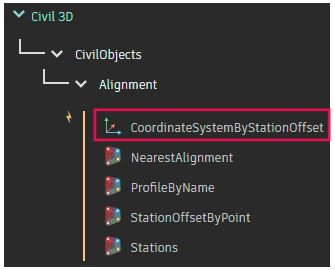


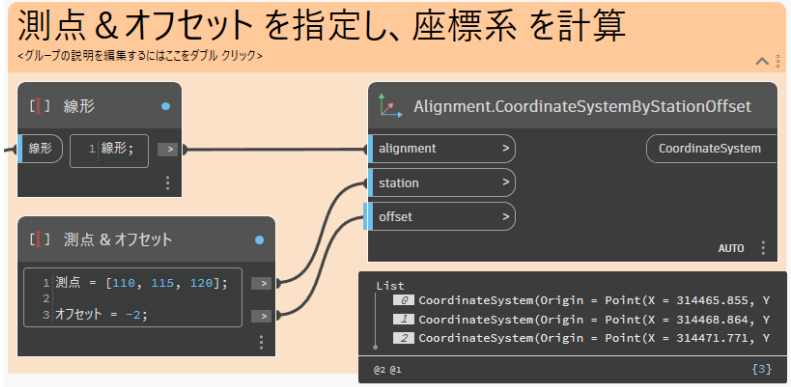


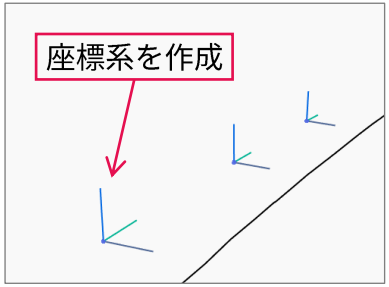


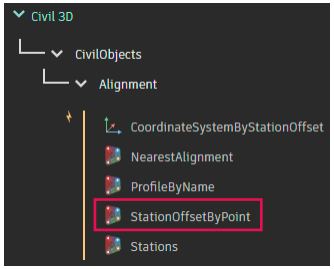
解説２） Civil 3D 平面線形＆縦断線形 – 計算

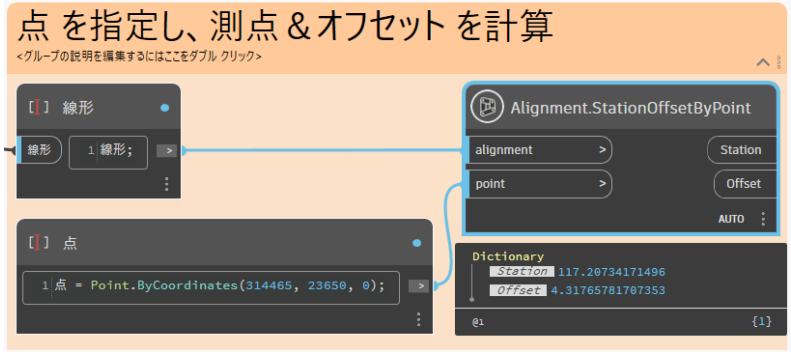
平面線形 から、計算して求められる情報は...

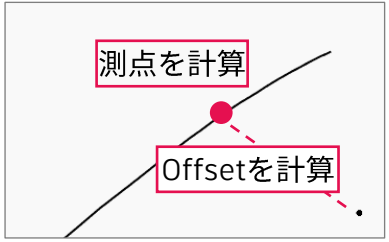






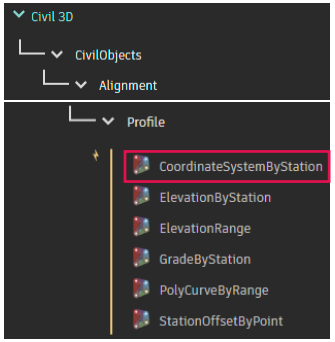


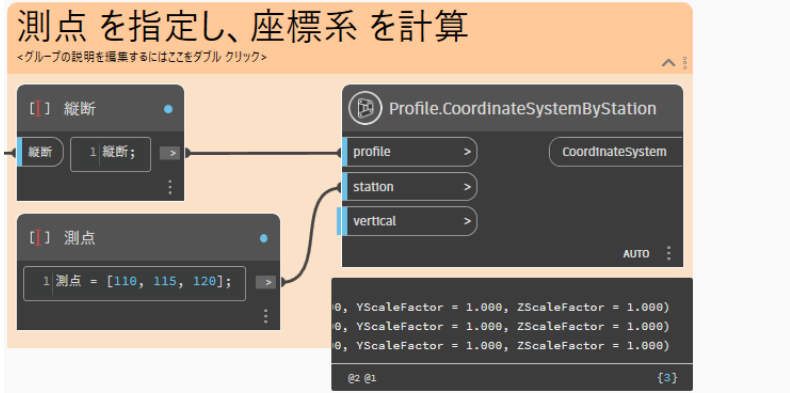


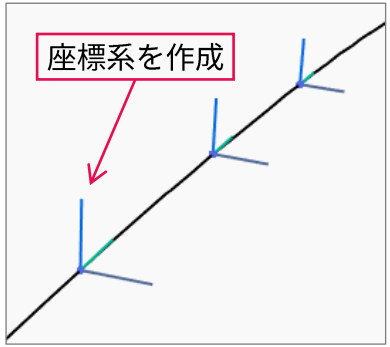


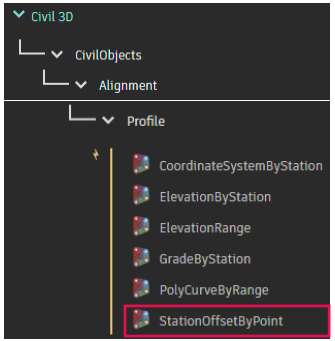
解説２） Civil 3D 平面線形＆縦断線形 – 計算

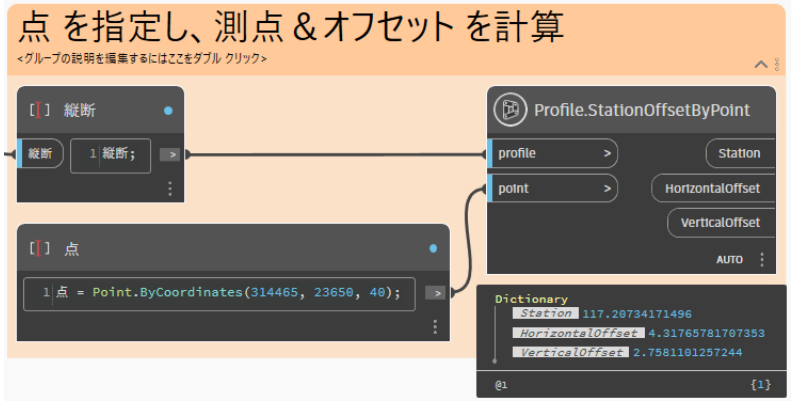
縦断線形 から、計算して求められる情報は...

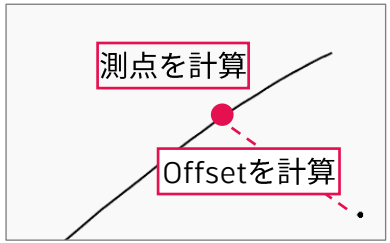






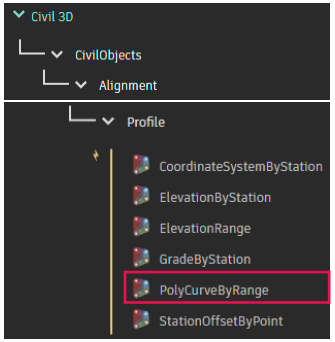


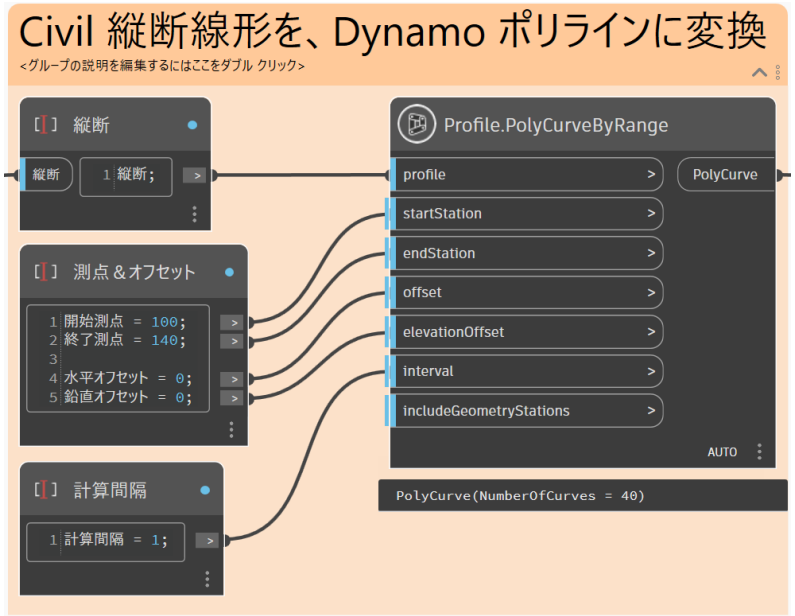


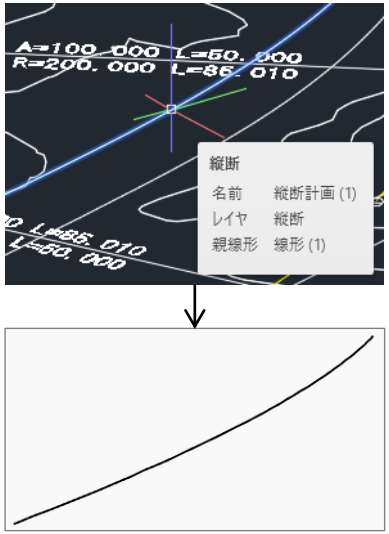


解説２）Civil 3D 平面線形＆縦断線形 – 計算

縦断線形 は、Dynamo ジオメトリに変換できる

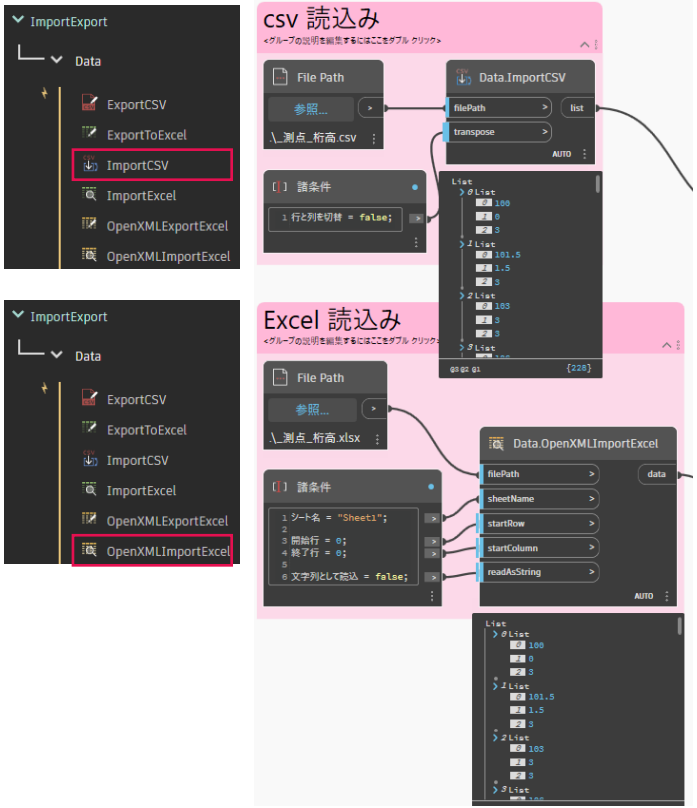


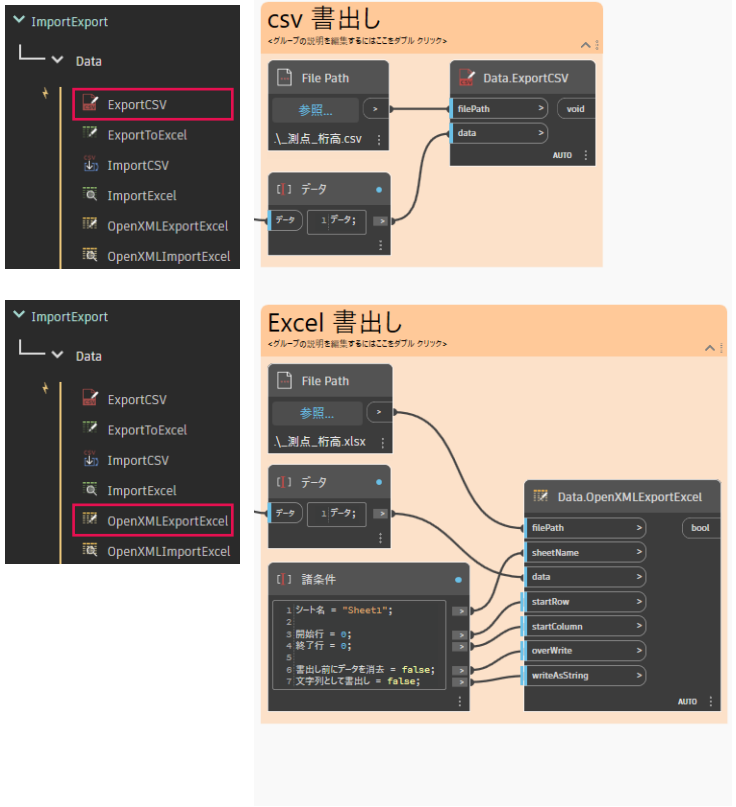




解説３）Excel / csv との連携、リストの編集

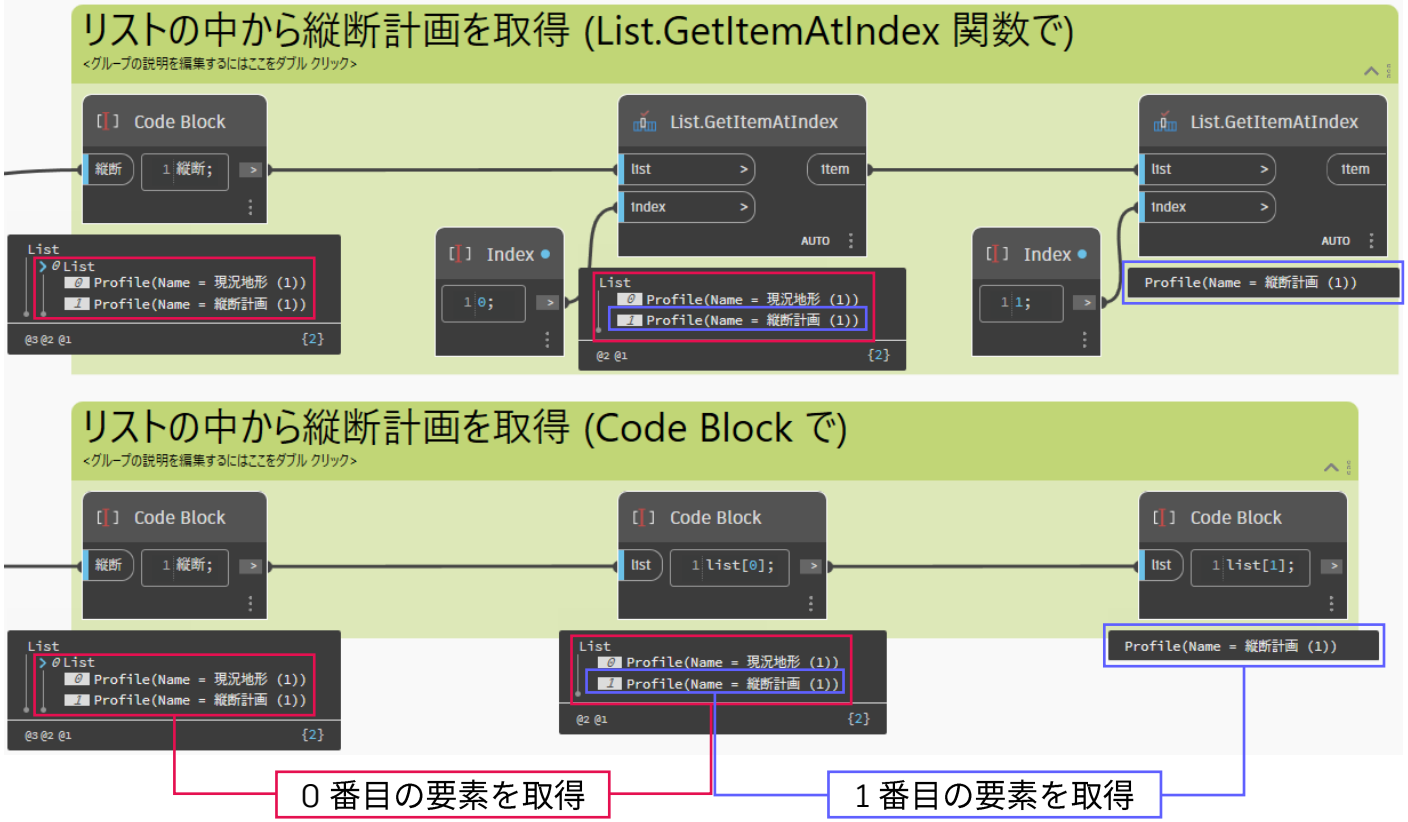
Excel や csv から、データを入出力するには…（復習も兼ねて）

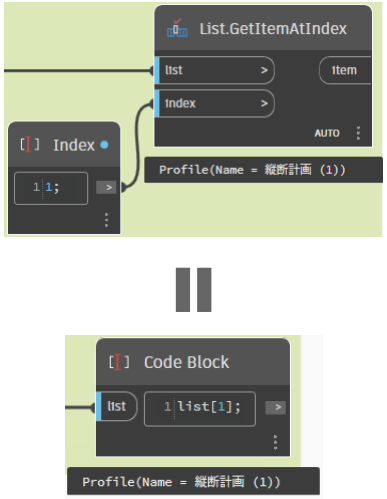




解説３）Excel / csv との連携、リストの編集

リストから、特定の要素を抽出するには…（Code Block が便利！）





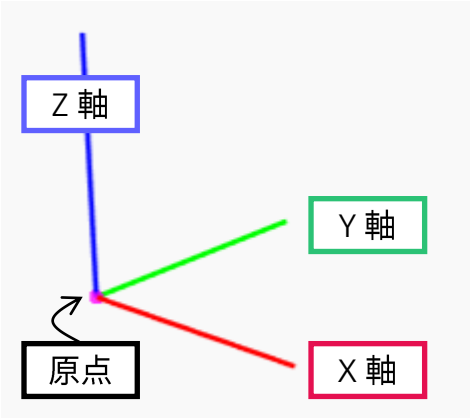
解説３）Excel / csv との連携、リストの編集

リストから、特定の要素を抽出するには…（Code Block が便利！）

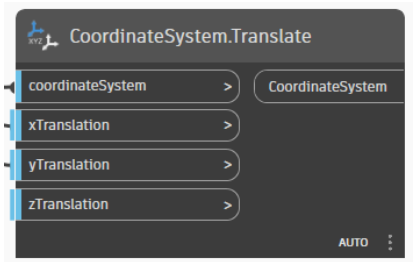
解説４）座標系の編集

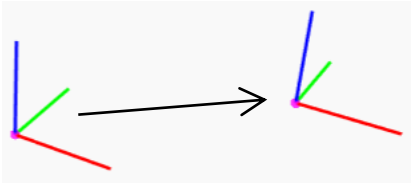
“座標系” とは？（復習も兼ねて）

座標系：位置（原点）と方向（X, Y, Z 軸）の組合せ

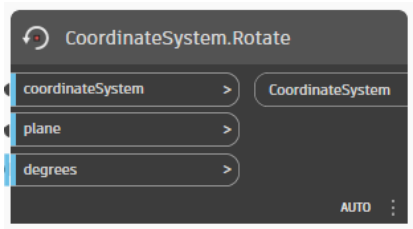


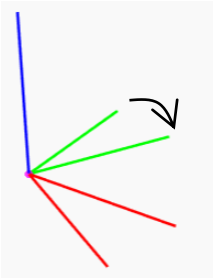
移動したり



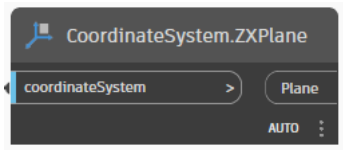


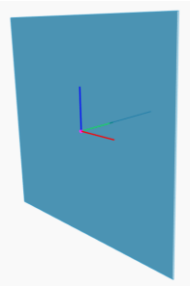
回転したり





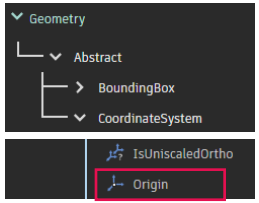
断面を求めたり

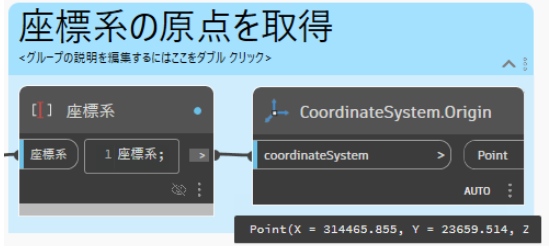


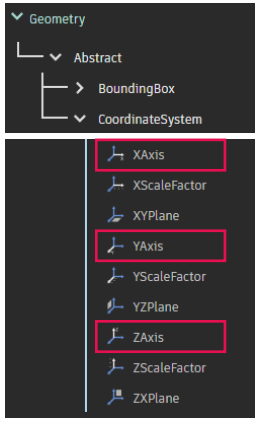


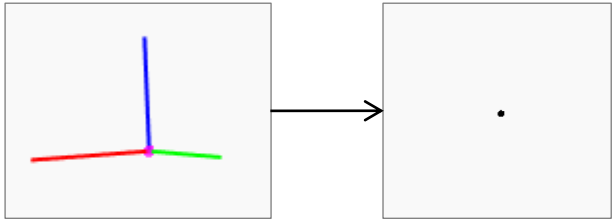
解説４）座標系の編集

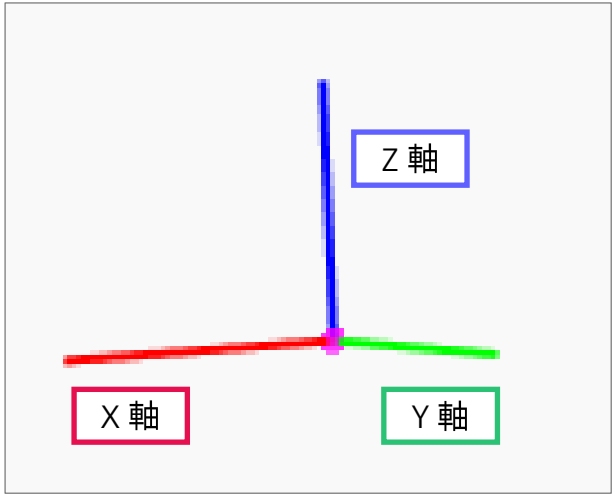
座標系 から、そのまま抜き出せる情報は…





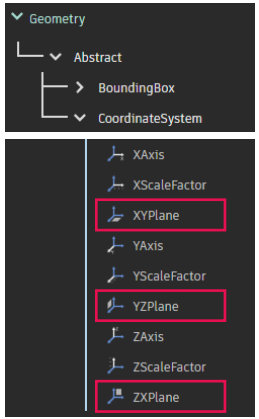


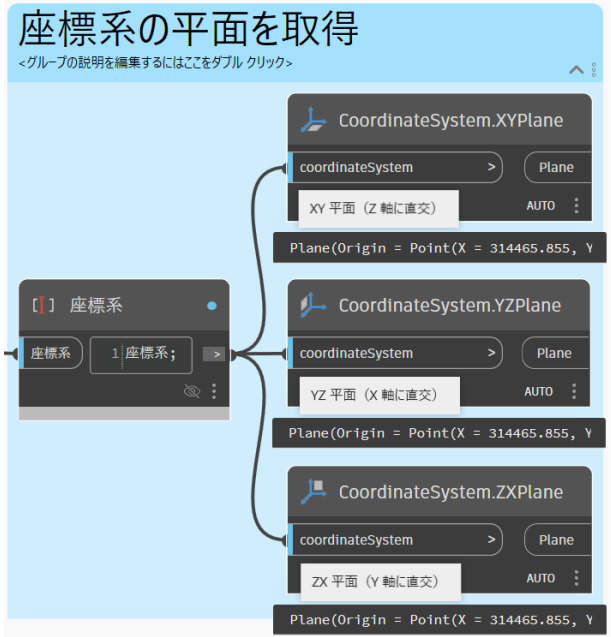




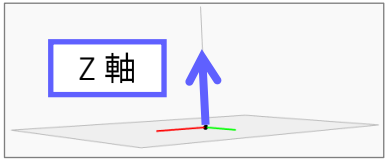
解説４）座標系の編集

座標系 から、そのまま抜き出せる情報は…

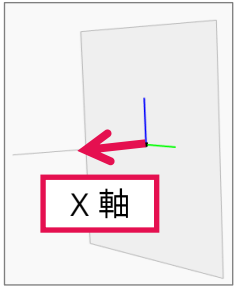




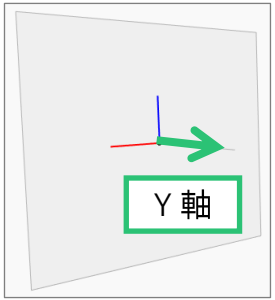
XYPlane：
Z 軸に直交
する平面



YZPlane：
X 軸に直交
する平面

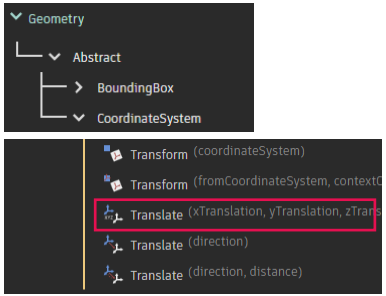


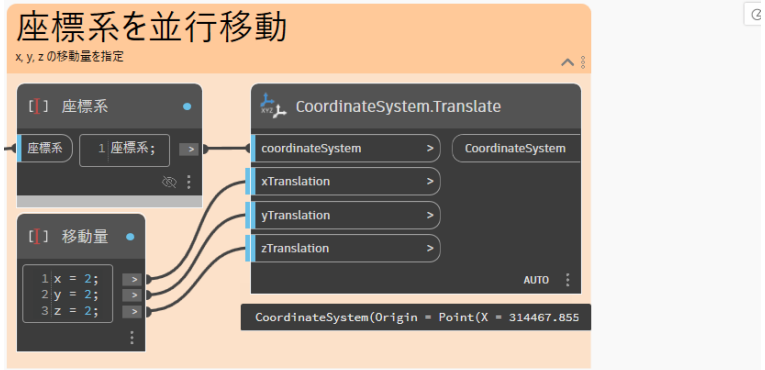
ZXPlane：
Y 軸に直交
する平面

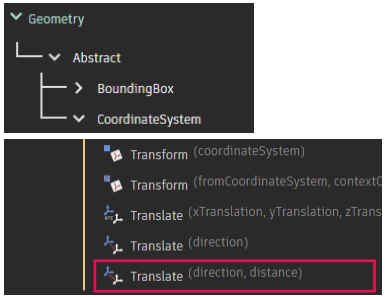


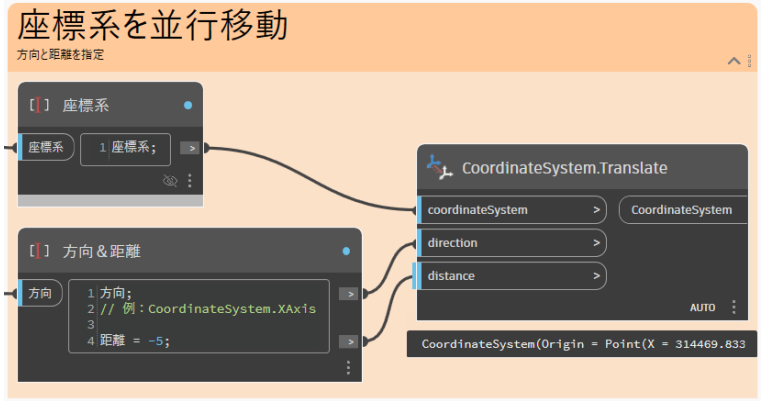
解説４）座標系の編集

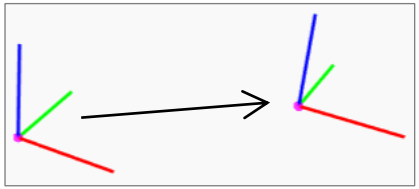
座標系 は、動かすことができる





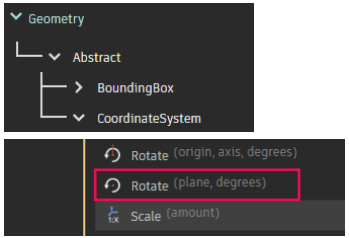


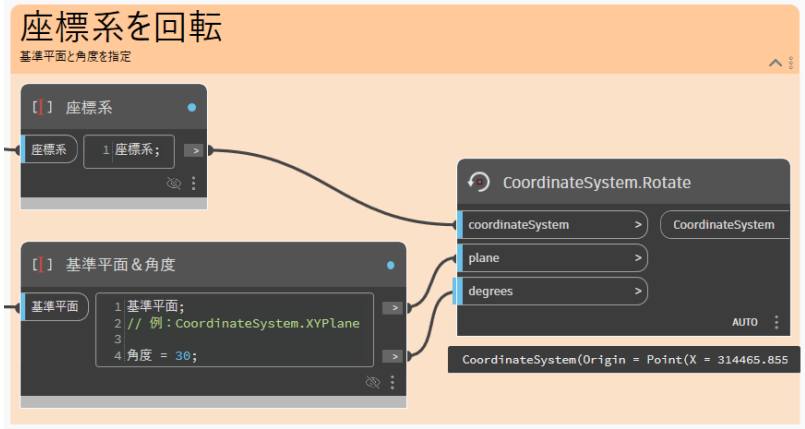


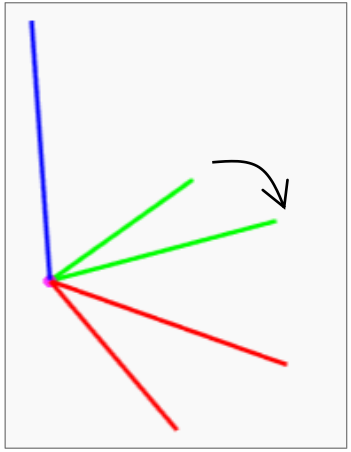


解説４）座標系の編集

座標系 は、動かすことができる



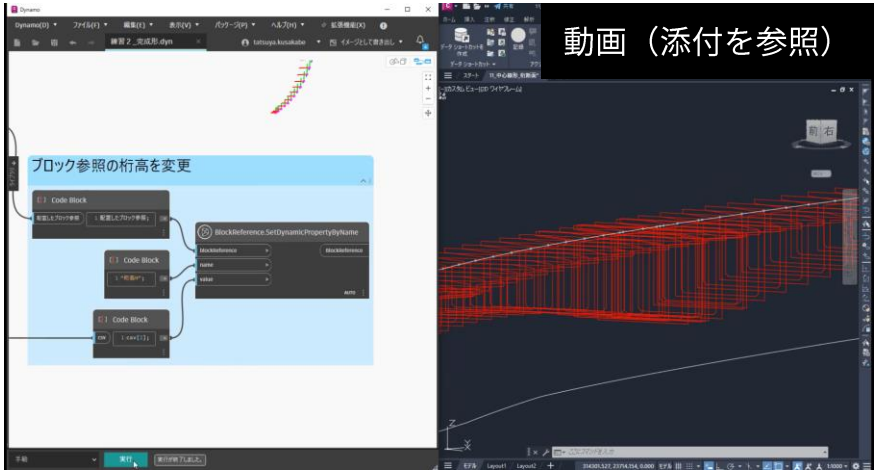




練習２）Civil 3D の縦断線形に沿って、ブロックを配置

完成形を先にお見せします

- 手順
  - Civil 3D から、ブロック定義を取得
  - Civil 3D に、ブロック参照を作成
  - ブロック参照のパラメータを変更
    - ダイナミックブロックの活用



解説５）AutoCAD ブロック定義

二つの “ブロック” … “ブロック定義” と “ブロック参照”

ブロック定義

図形を組み合わせて、一まとめに登録

例：BLOCKSPALETTE の一覧

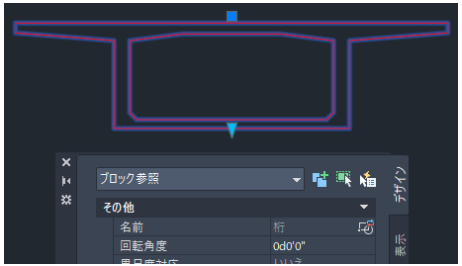
桁断面	路線記号	人孔記号	断面記号	…

要素	始点 X	始点 Y	終点 X	終点 Y	…
線分 1	xx	yy	xx	yy	
…					

ブロック参照

ブロック定義を参照し、CAD 上に配置

例：ブロック参照 オブジェクト

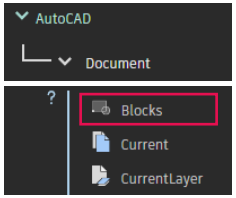


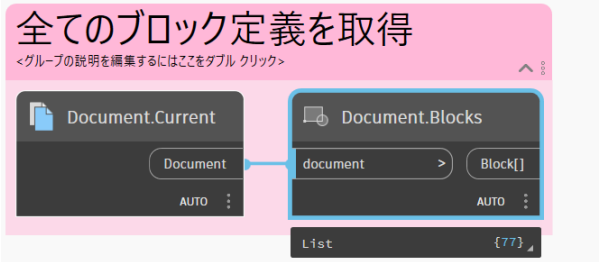
ブロック参照 1	ブロック参照 2	線分	…

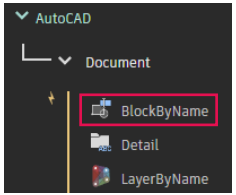
ブロック定義	配置座標 X	配置座標 Y	…
桁断面	xx	yy	

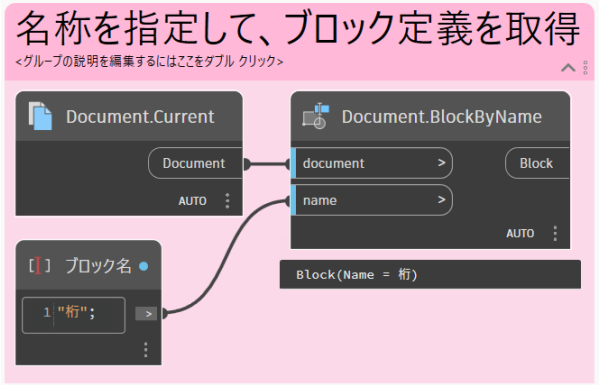
解説５）AutoCAD ブロック定義

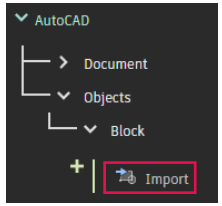
ブロック定義 を取得するには...





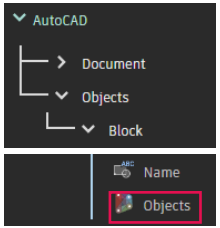


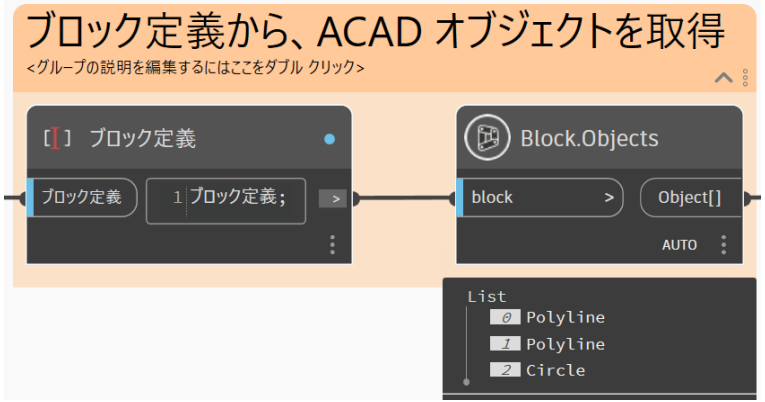


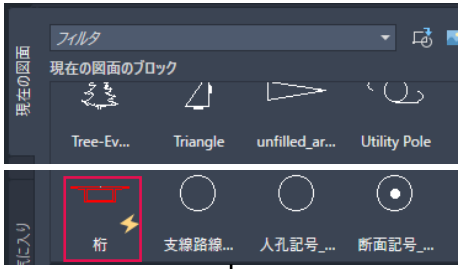


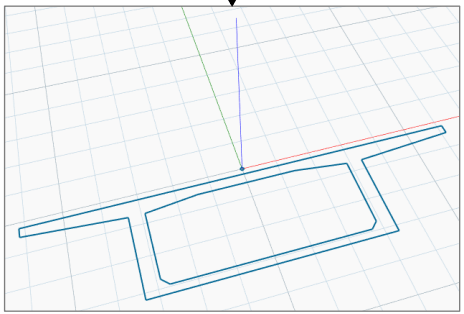
解説５）AutoCAD ブロック定義

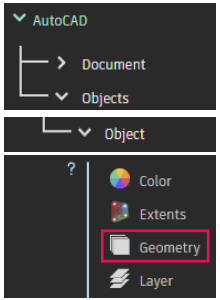
ブロック定義 は、Dynamo ジオメトリに変換できる

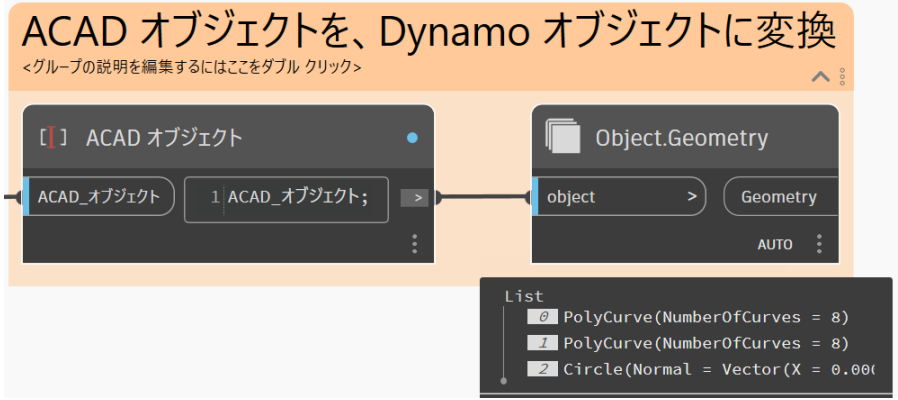












解説６）AutoCAD ブロック参照

二つの “ブロック” … “ブロック定義” と “ブロック参照”

ブロック定義

図形を組み合わせて、一まとめに登録

例：BLOCKSPALETTE の一覧

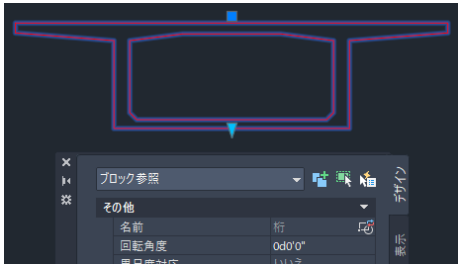
桁断面	路線記号	人孔記号	断面記号	…

要素	始点 X	始点 Y	終点 X	終点 Y	…
線分 1	xx	yy	xx	yy	
…					

ブロック参照

ブロック定義を参照し、CAD 上に配置

例：ブロック参照 オブジェクト

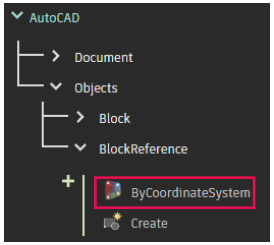


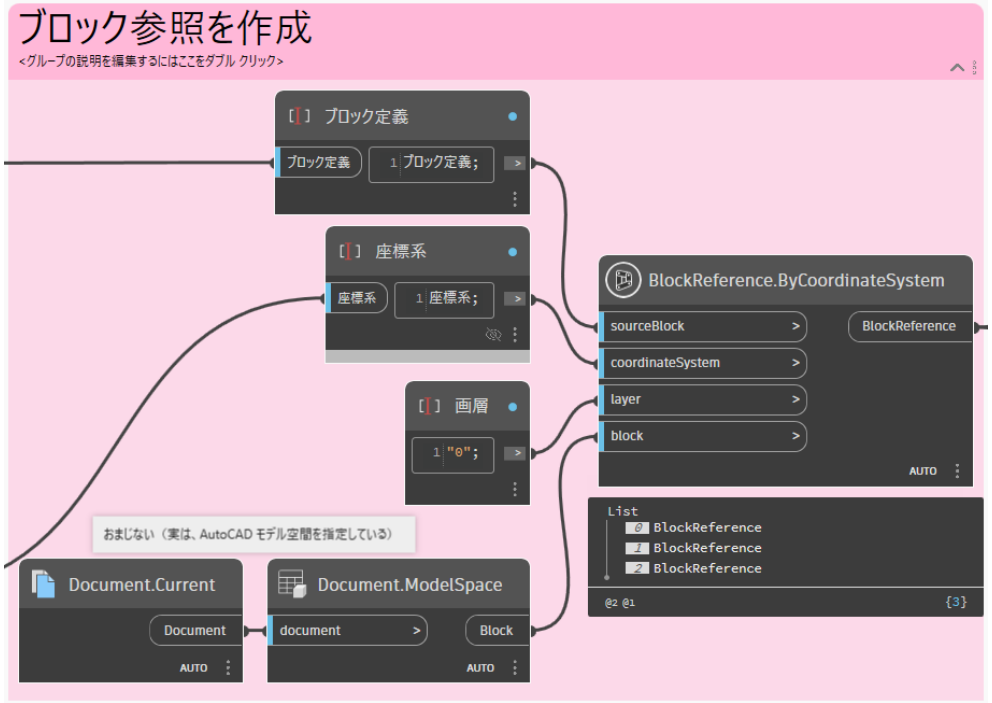
ブロック参照 1	ブロック参照 2	線分	…

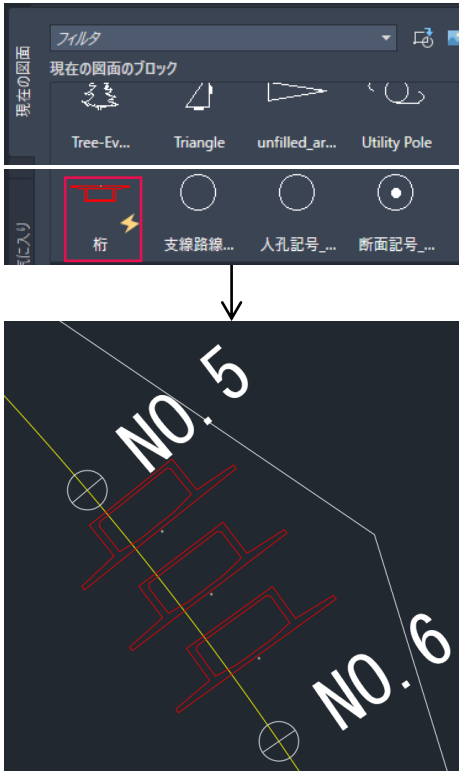
ブロック定義	配置座標 X	配置座標 Y	…
桁断面	xx	yy	

解説６）AutoCAD ブロック参照

ブロック参照 を作成するには...

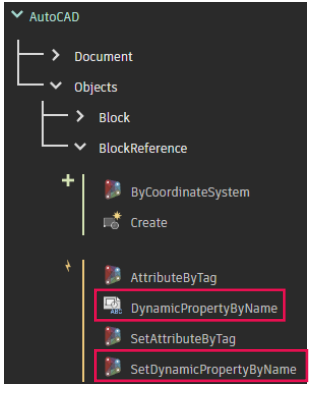


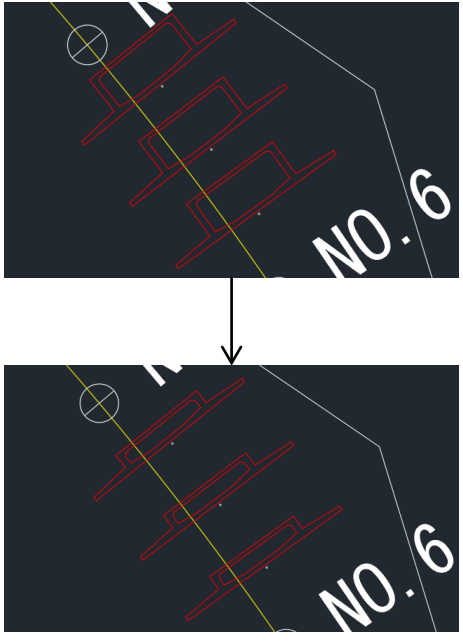




解説６）AutoCAD ブロック参照

ブロック参照 は、パラメトリックに形状を変えられる（ダイナミックブロックの場合）

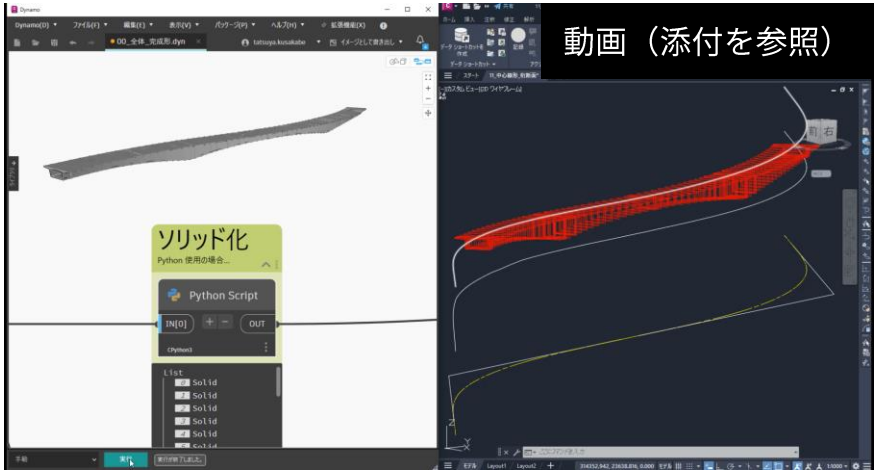




練習３）配置したブロックから、ソリッドを作成

完成形を先にお見せします

- 手順
  - ブロック参照を Dynamo ポリラインに変換
    - Civil3DToolkit パッケージの活用
  - Dynamo ポリラインをソリッドに変換
    - Python スクリプトの活用



解説７）パッケージの活用

パッケージの概要＆使用方法

- パッケージの概要
  - Autodesk 以外の第三者が開発した
    ノードをまとめて配布したもの
  - いわば “便利なノード集”
- 使用方法
  - パッケージを検索する
  - パッケージをダウンロードする
  - ライブラリ内に、パッケージが追加される
- Civil 3D Toolkit パッケージ
  - Civil 3D, AutoCAD ノードを
    大幅に拡張

解説７）パッケージの活用

Civil 3D Toolkit パッケージで出来るようになること（多すぎるので一部のみ抜粋）

- オブジェクトの作成
  - Arc.ByGeometry
    - Arc を Dynamo ジオメトリに変換
  - PolylineExtensions.ObjectByGeometry
    - Arc を含む Polyline を Dynamo ジオメトリに変換
  - BlockReferenceExtension.ByGeometry
    - Dynamo ジオメトリから、ブロック参照を作成
  - BlockReferenceExtension.GetGeometry
    - ブロック参照から、Dynamo ジオメトリを取得
  - AlignmentExtensions.CreateAlignmentByPolyline
    - ポリラインから平面線形を作成
  - AlignmentExtensions.AddProfileByName / ProfileExtensions.AddPVI
    - 空の縦断線形を作成 → PVI 追加
- オブジェクトの選択
  - Select Objects
    - 複数のオブジェクトを一括で選択
  - Object Types
    - Arc, CogoPoint, 計画線, ハッチング など、フィルタできるオブジェクトの種類が増加
  - Surface Names / Alignment Names / Corridor Names
    - 名前を指定して、サーフェス, 線形, コリドー を取得
- オブジェクトから情報を取得
  - ObjectExtensions.SetParameterValueByName
    - Civil 3D を含むオブジェクトに、パラメータを設定
  - ObjectExtensions.GetParameters
    - Civil 3D を含むオブジェクトの、パラメータを取得

解説７）パッケージの活用

Civil 3D Toolkit パッケージで出来るようになること（多すぎるので一部のみ抜粋）

- 主な拡張項目 (AutoCAD)
  - ハッチング (Hatch)
  - テキスト、引出線 (MText, MLeader)
  - 凡例 (Legend, LegendItem)
  - 画層 (Layer)
  - ブロック (Block, BlockReference)
  - 表 (Table)
  - レイアウト (Layout, Viewport)
  - プロットスタイル (PlotStyle)

- 主な拡張項目 (Civil 3D)
  - サーフェス (TinSurface)
  - 線形 (Alignment)
  - 縦断 (Profile, PVI)
  - 縦断ビュー (ProfileView)
  - アセンブリ (Assembly, AssemblyGroup)
  - サブアセンブリ (Subassembly, ...Link, ...Shape, ...Point, ...TargetInfo TkSubassemblyParameter)
  - コリドー (Corridor)
  - コリドー計画線 (TkCorridorFeatureLine)
  - 横断 (Section)
  - 横断ビュー (SectionView)

解説７）パッケージの活用

Civil 3D Toolkit パッケージで出来るようになること（多すぎるので一部のみ抜粋）

- Autodesk University セッションの中で、各ノードの機能を網羅的に解説
  - Supercharge Your Dynamo Graph with Civil 3D Toolkit | Autodesk University

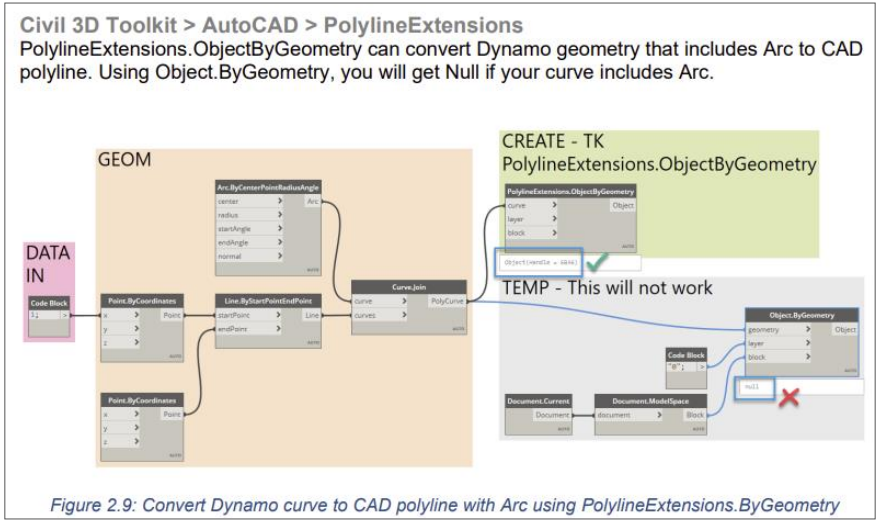


Figure 2.9: Convert Dynamo curve to CAD polyline with Arc using PolylineExtensions.ByGeometry

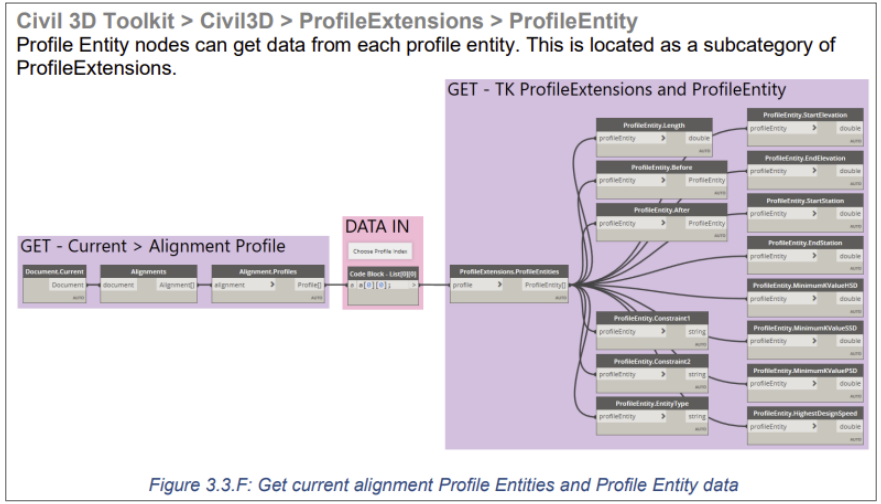
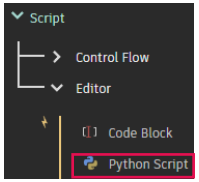


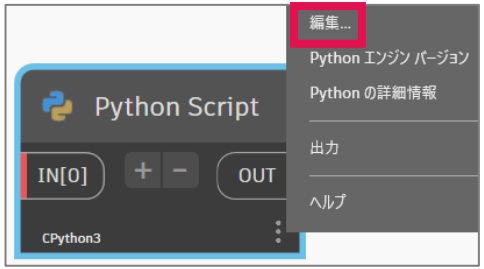
Figure 3.3.F: Get current alignment Profile Entities and Profile Entity data

解説８）Python スクリプトの活用

Python Script テンプレートに記載されている内容は…

Python で Civil 3D, AutoCAD のデータを直接操作する（上級者向け）






```
Python Script
# Load the Python Standard and DesignScript Libraries
import sys
import clr

# Add Assemblies for AutoCAD
clr.AddReference('AcMgd')
clr.AddReference('AcCoreMgd')
clr.AddReference('AcDbMgd')

# Import references from AutoCAD
from Autodesk.AutoCAD.ApplicationServices import *
from Autodesk.AutoCAD.DatabaseServices import *
from Autodesk.AutoCAD.Runtime import *
from Autodesk.AutoCAD.EditorInput import *
from Autodesk.AutoCAD.Geometry import *

# Add Assemblies for Civil3D
clr.AddReference('AecBaseMgd')
clr.AddReference('AecPropDataMgd')
clr.AddReference('AeccDbMgd')

# Import references from Civil3D
from Autodesk.Civil.ApplicationServices import *
from Autodesk.Civil.DatabaseServices import *

# The inputs to this node will be stored as a list in the IN variables.
adoc = Application.DocumentManager.MdiActiveDocument
editor = adoc.Editor

with adoc.LockDocument():
    with adoc.Database as db:
        with db.TransactionManager.StartTransaction() as t:
            ここにコードを書く…
            t.Commit()

# Assign your output to the OUT variable.
OUT = 0
```

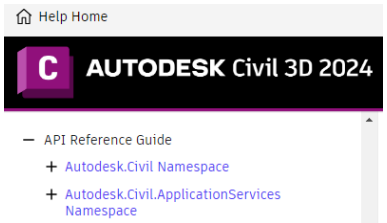
AutoCAD API にアクセスする
ための “おまじない”

Civil 3D API にアクセスする
ための “おまじない”

dwg 編集の開始＆終了を宣言する
ための “おまじない”

プログラムを実装

* API Reference を参照しながら



解説８）Python スクリプトの活用

今回のサンプルで作成した内容は…

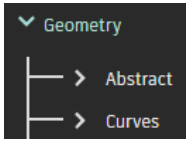
Python で繰り返し計算を行う、Python で Dynamo ジオメトリを操作する（中級者向け）

Python Script

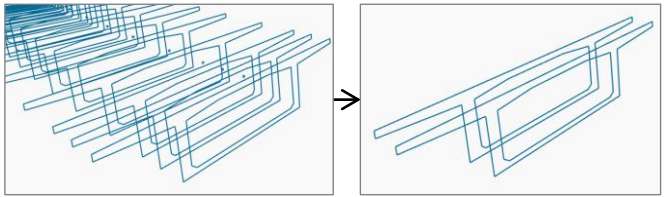
```
# おまじない
import sys
import clr

# Geometry フォルダのノードを扱う場合
clr.AddReference('ProtoGeometry')
from Autodesk.DesignScript.Geometry import *

input = IN[0]
diffSolidList = []

# Python の関数で繰り返し計算
for i in range(1, len(input)-1):

    # ソリッドの開始断面と終了断面で、ポリラインを取得
    outerPolyline = [input[i][0], input[i+1][0]]
    innerPolyline = [input[i][1], input[i+1][1]]

    # ポリラインから、ソリッドを作成
    outerSolid = Solid.ByLoft(outerPolyline)
    innerSolid = Solid.ByLoft(innerPolyline)
    diffSolid = Solid.Difference(outerSolid, innerSolid)
    diffSolidList.append(diffSolid)

    # 要らないソリッドを消去
    outerSolid.Dispose()
    innerSolid.Dispose()

OUT = diffSolidList
```

実行　CPython3

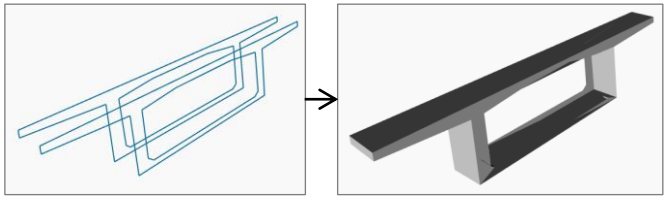
Dynamo の Geometry ライブラリに
アクセスするための “おまじない”



ソリッドの作成に必要な
ポリラインを抽出



ポリラインから
ソリッドを作成



使用しない図形を削除（処理の高速化のため）

解説８）Python スクリプトの活用

今回のサンプルで作成した内容は…

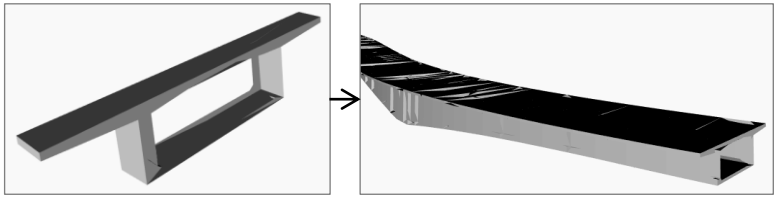
Python で繰り返し計算を行う、Python で Dynamo ジオメトリを操作する（中級者向け）

Python Script

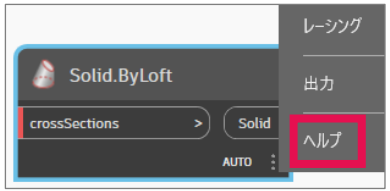
```python
# おまじない
import sys
import clr

# Geometry フォルダのノードを扱う場合
clr.AddReference('ProtoGeometry')
from Autodesk.DesignScript.Geometry import *

input = IN[0]
diffSolidList = []

# Python の関数で繰り返し計算
for i in range(1, len(input)-1):

    # ソリッドの開始断面と終了断面で、ポリラインを取得
    outerPolyline = [input[i][0], input[i+1][0]]
    innerPolyline = [input[i][1], input[i+1][1]]

    # ポリラインから、ソリッドを作成
    outerSolid = Solid.ByLoft(outerPolyline)
    innerSolid = Solid.ByLoft(innerPolyline)
    diffSolid = Solid.Difference(outerSolid, innerSolid)
    diffSolidList.append(diffSolid)

    # 要らないソリッドを消去
    outerSolid.Dispose()
    innerSolid.Dispose()

OUT = diffSolidList
```

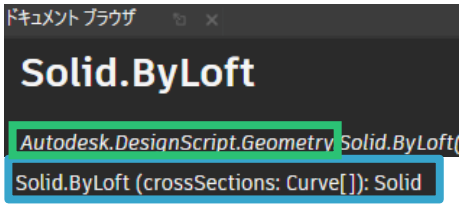
実行　CPython3

ソリッドを作成 の処理を
繰り返し実施



Dynamo ノードの
Python での記法は
ヘルプを見ながら






```python
# Geometry フォルダのノードを扱う場合
clr.AddReference('ProtoGeometry')
from Autodesk.DesignScript.Geometry import *
    # ポリラインから、ソリッドを作成
    outerSolid = Solid.ByLoft(outerPolyline)
    innerSolid = Solid.ByLoft(innerPolyline)
```