Automating Tasks with the Fusion 360 API

Patrick Rainsberry, Autodesk

Business Strategy, Fusion 360

AUTODESK UNIVERSITY

Outline

- API Overview
- Key concepts of the API
- Building an Add-In
- Resources

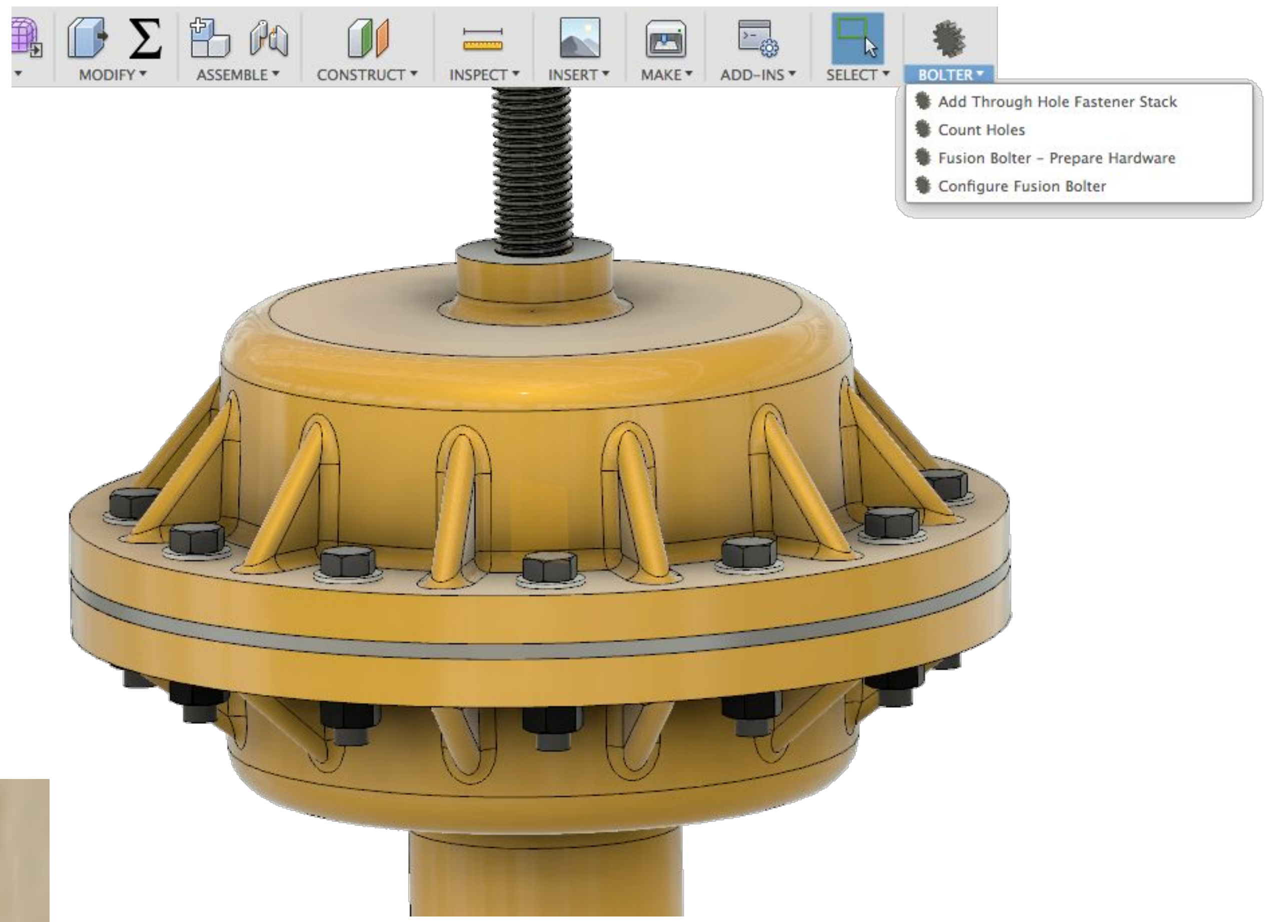
Fusion 360 API Overview

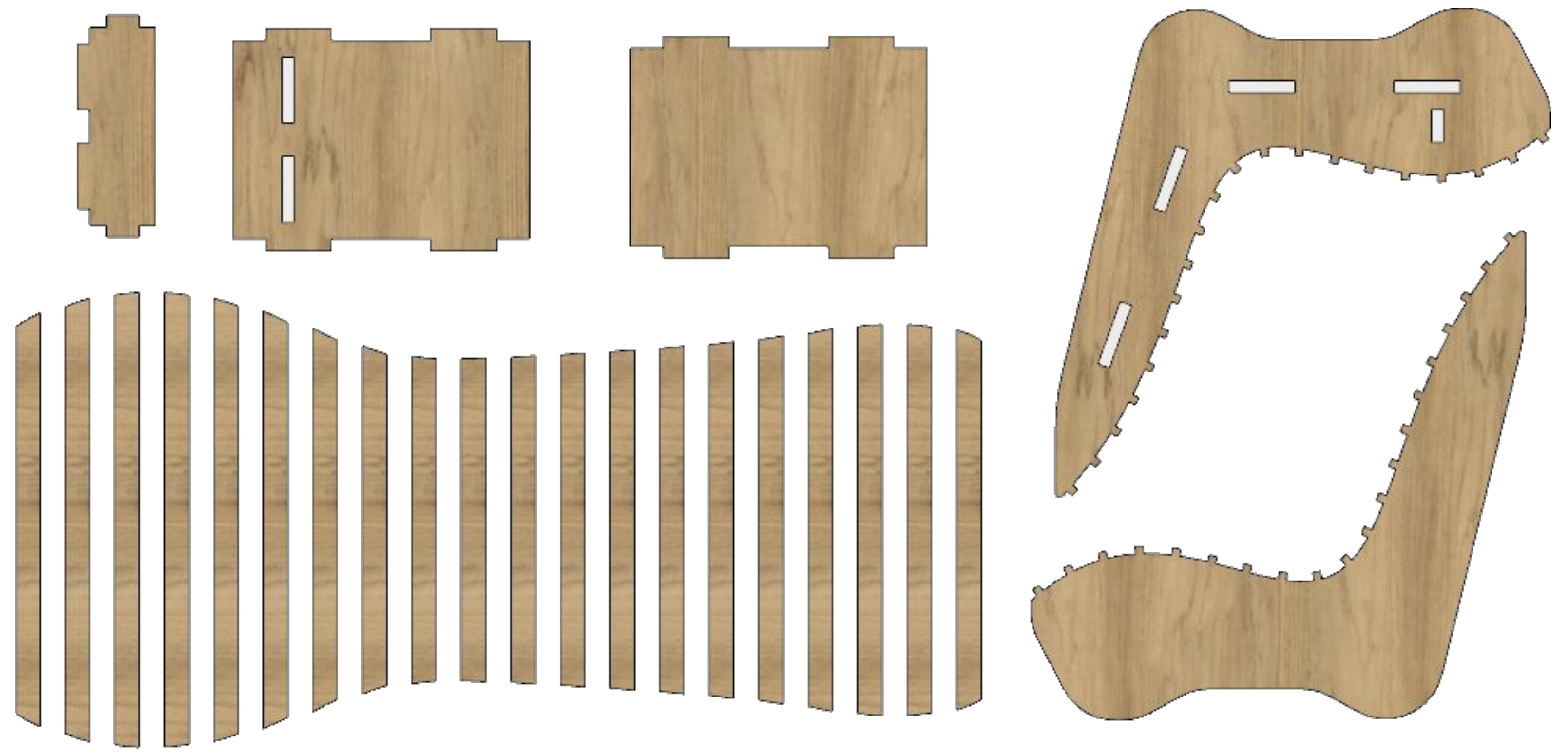
Things to Automate

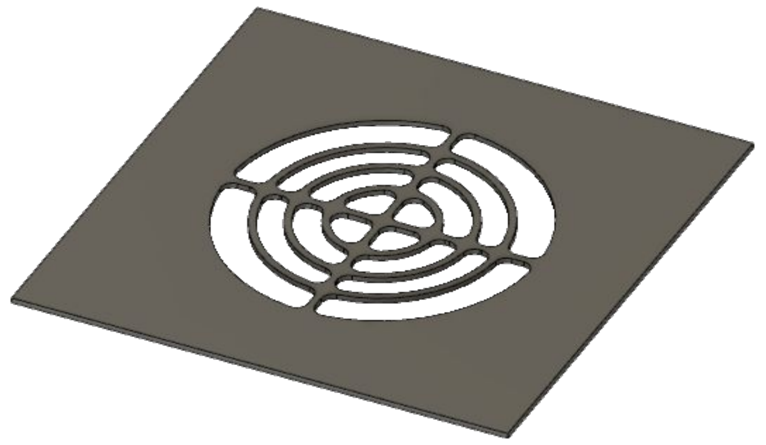
Repetitive tasks

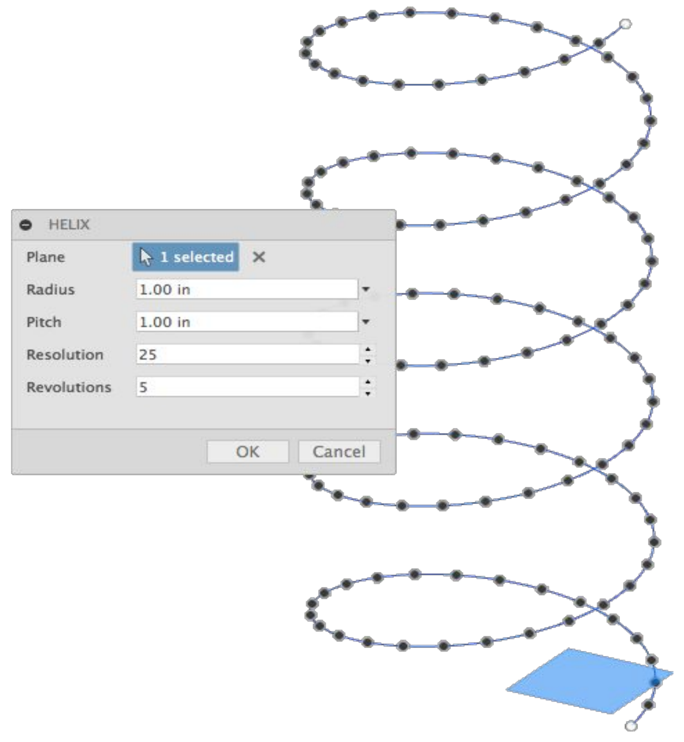
Data import / export

Complex operations








HELIX

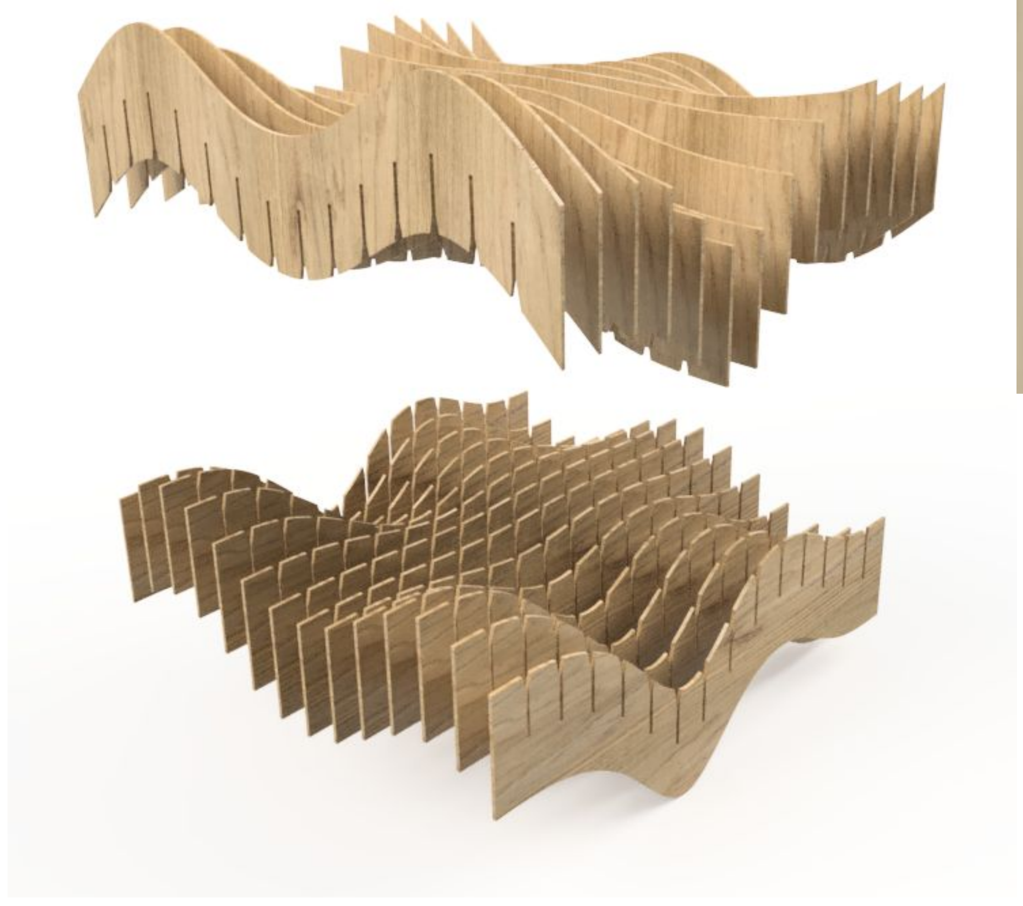
Plane: 1 selected

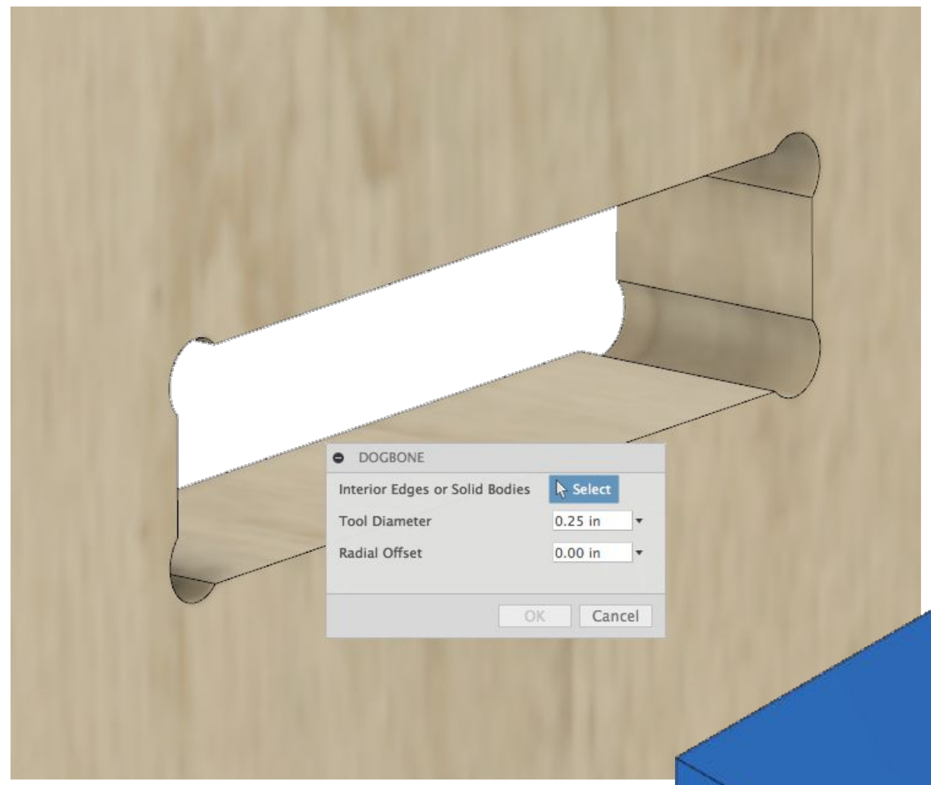
Radius: 1.00 in

Pitch: 1.00 in

Resolution: 25

Revolutions: 5

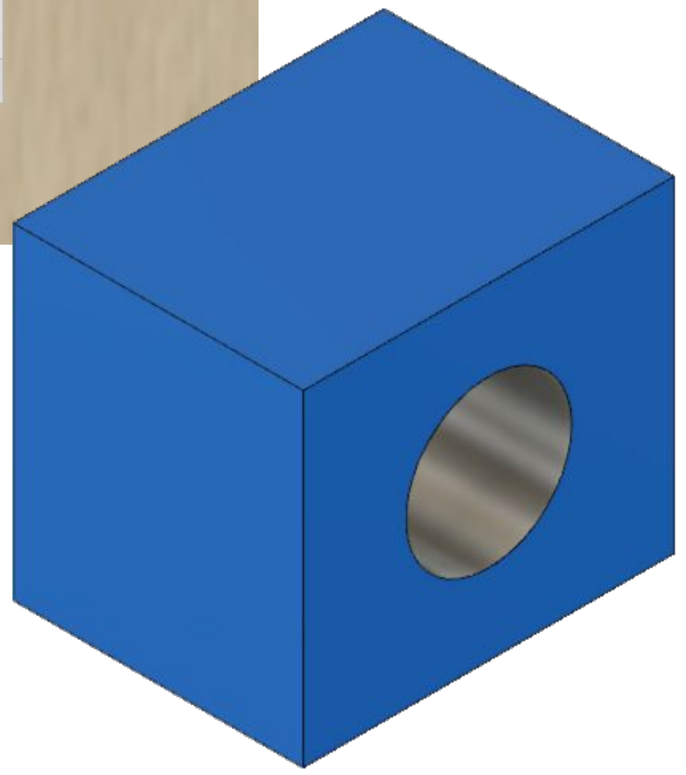


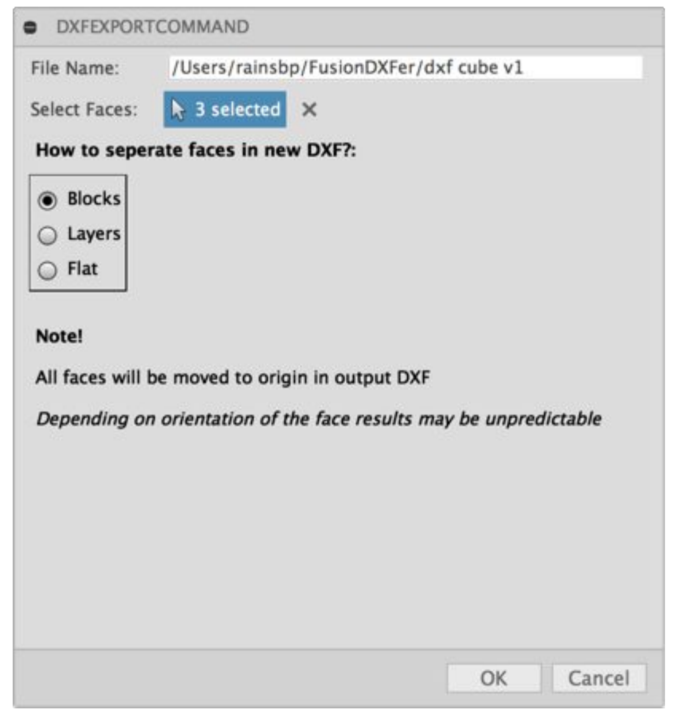

DOGBONE

Interior Edges or Solid Bodies: Select

Tool Diameter: 0.25 in

Radial Offset: 0.00 in




DXFEXPORTCOMMAND

File Name: /Users/rainsbp/FusionDXFer/dxf cube v1

Select Faces: 3 selected

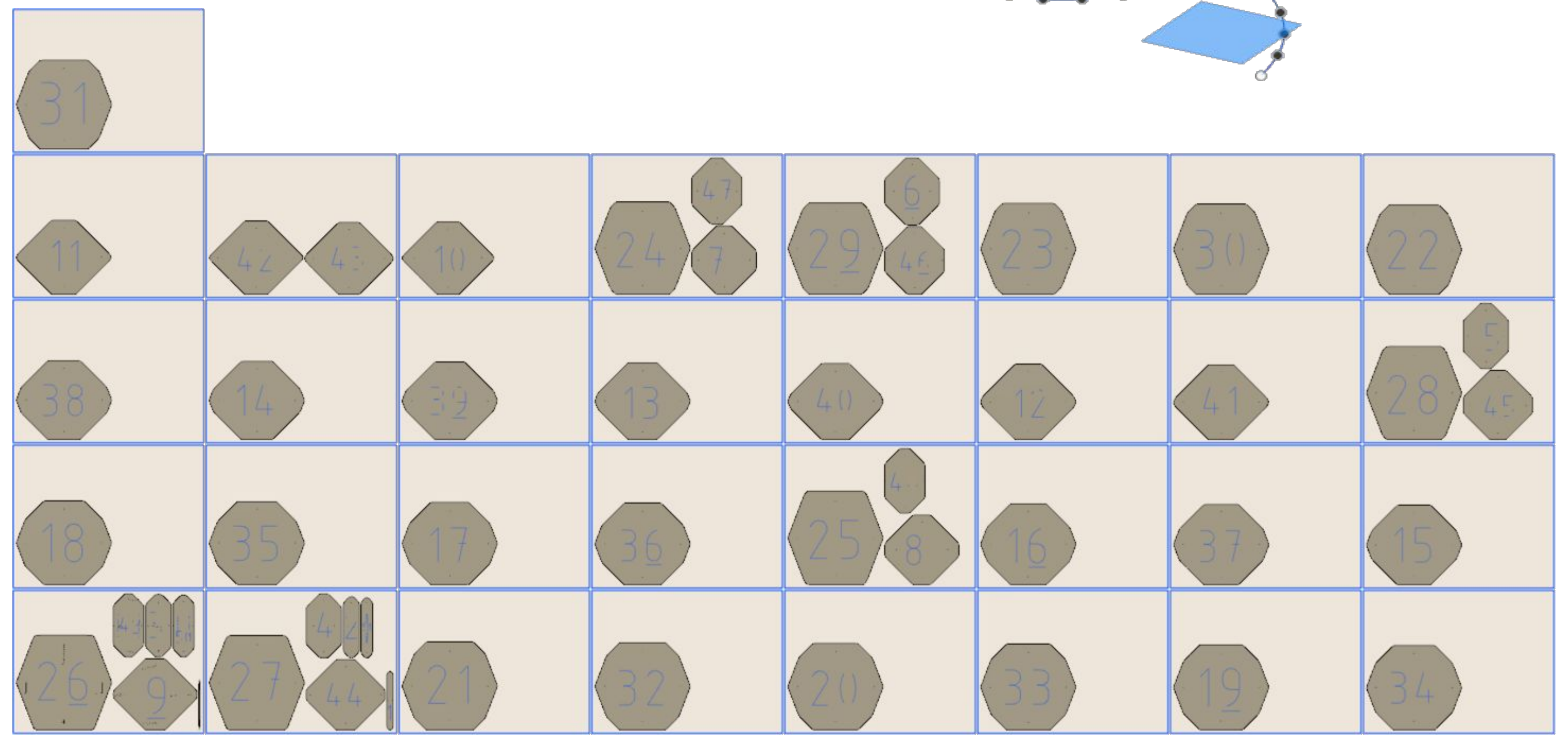
How to seperate faces in new DXF?:

[✓ Blocks] Layers Flat

Note!

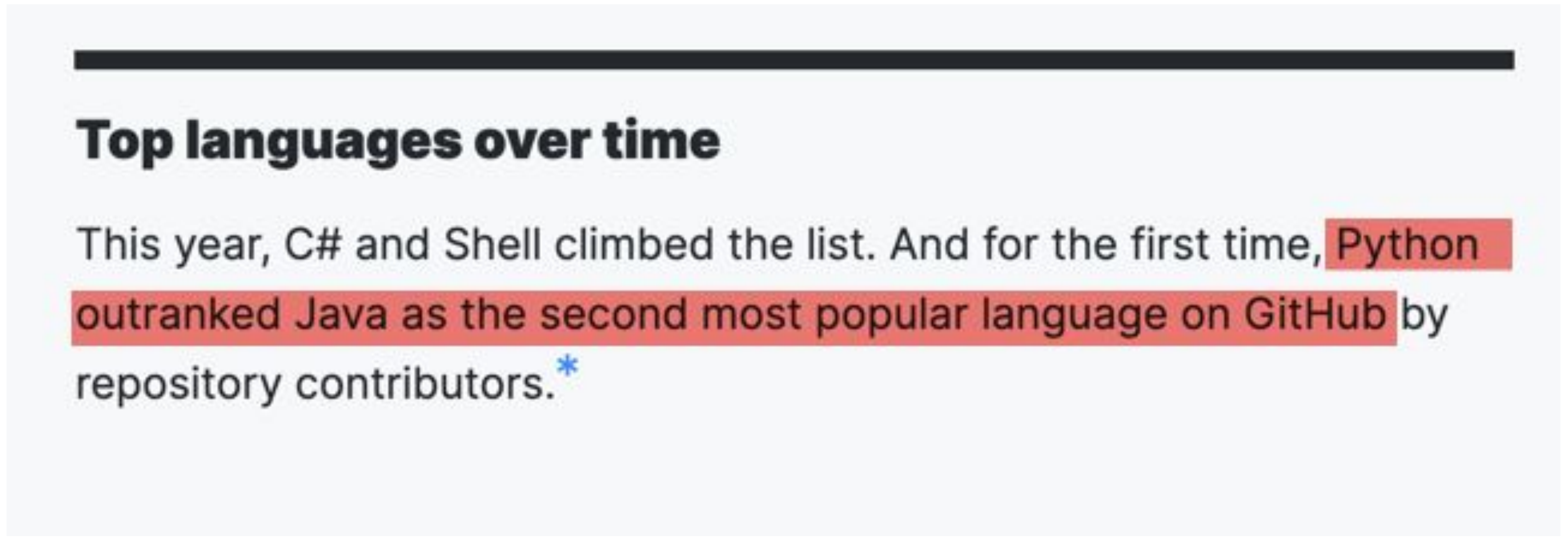
All faces will be moved to origin in output DXF

Depending on orientation of the face results may be unpredictable



Fusion 360 API

- Platform independent API supports OSX and Windows
- Designed to be program language independent, currently supports:
  - Python
  - C++
- Python is a widely used general-purpose, high-level programming language that is designed to be concise and human readable.



Top languages over time

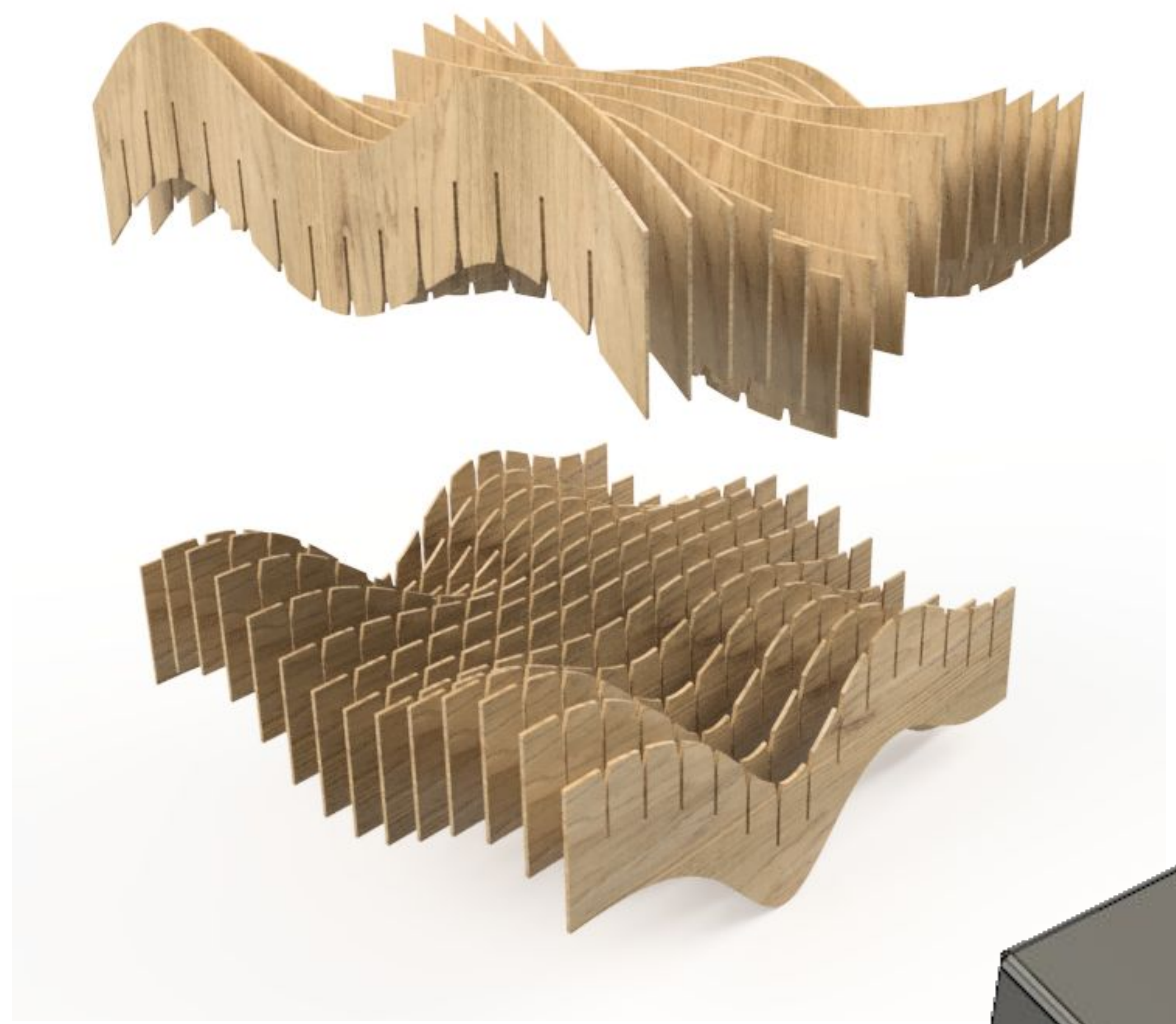
This year, C# and Shell climbed the list. And for the first time, Python outranked Java as the second most popular language on GitHub by repository contributors.*

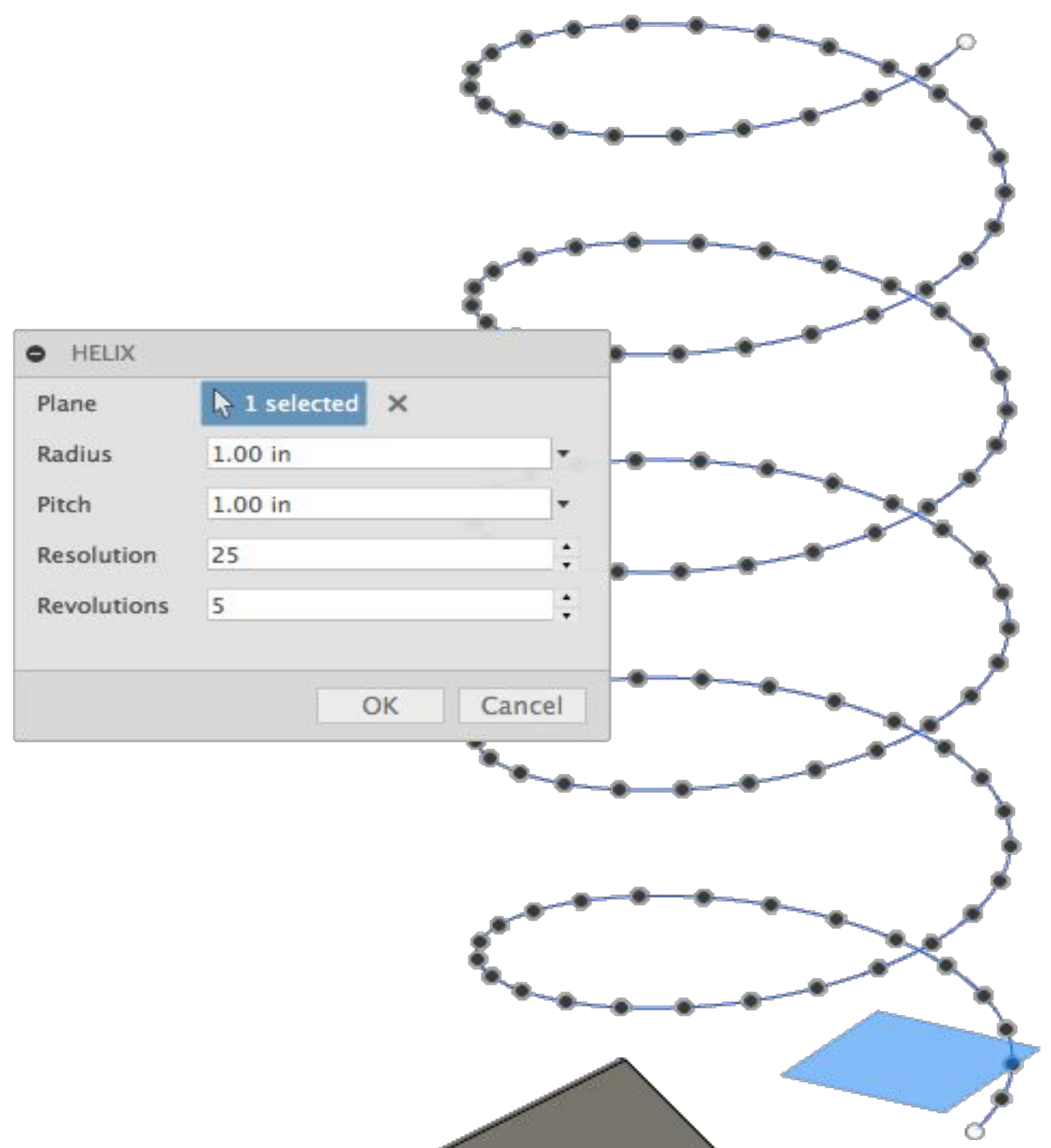
Primary Areas of the API

- **Design**
  - Automate creation and editing of solid and surface geometry
  - Interrogate and analyze geometry
- **CAM**
  - Interrogate basic CAM information
  - Automate post processing
- **Data**
  - Import/Export Data
  - Interrogate and manipulate Fusion 360 Data
- **Other Useful Concepts**
  - Custom Graphics
  - Palettes
  - Application Events
  - Attributes
  - Temporary BREP Manager

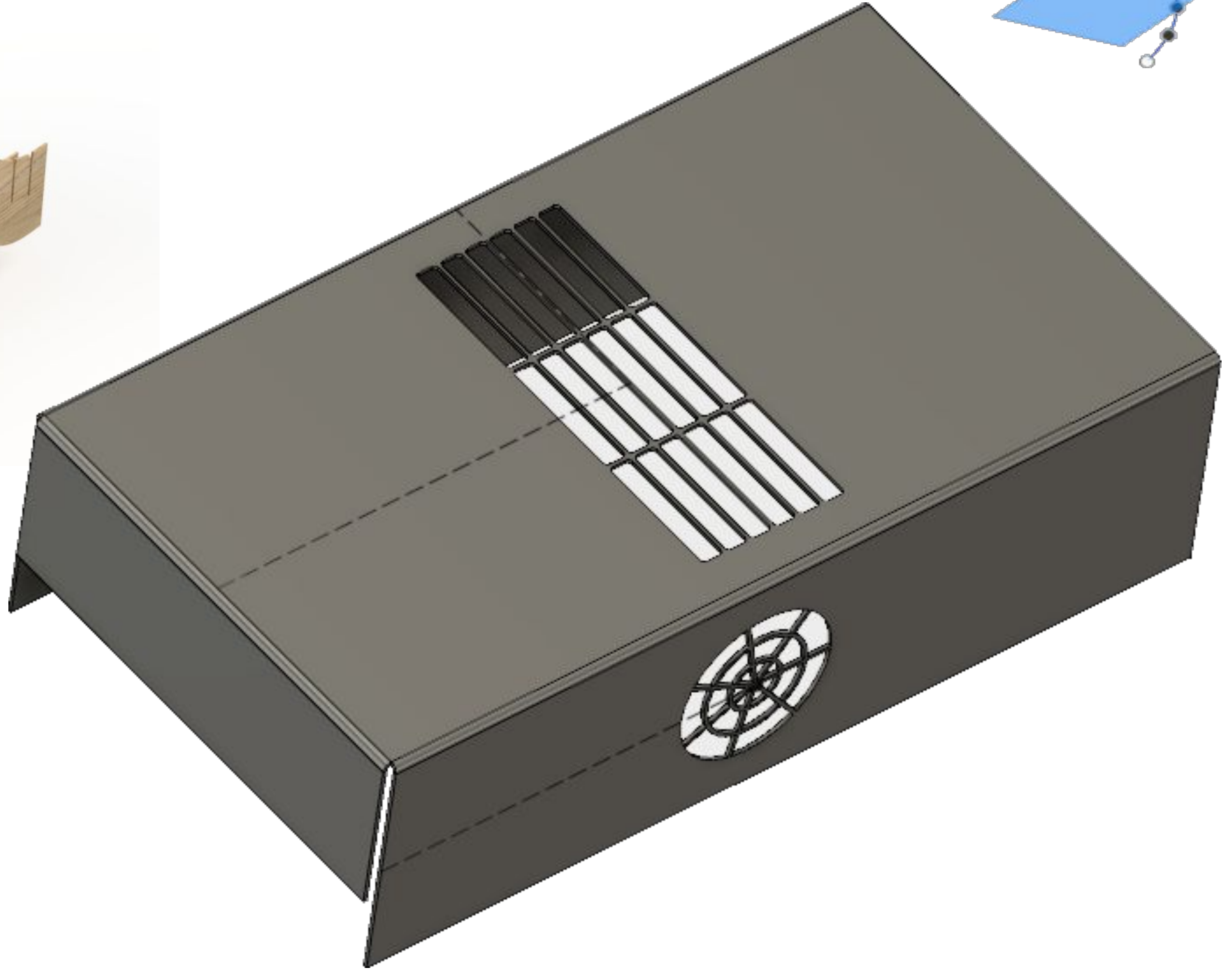
Design API

Automate Geometry Creation

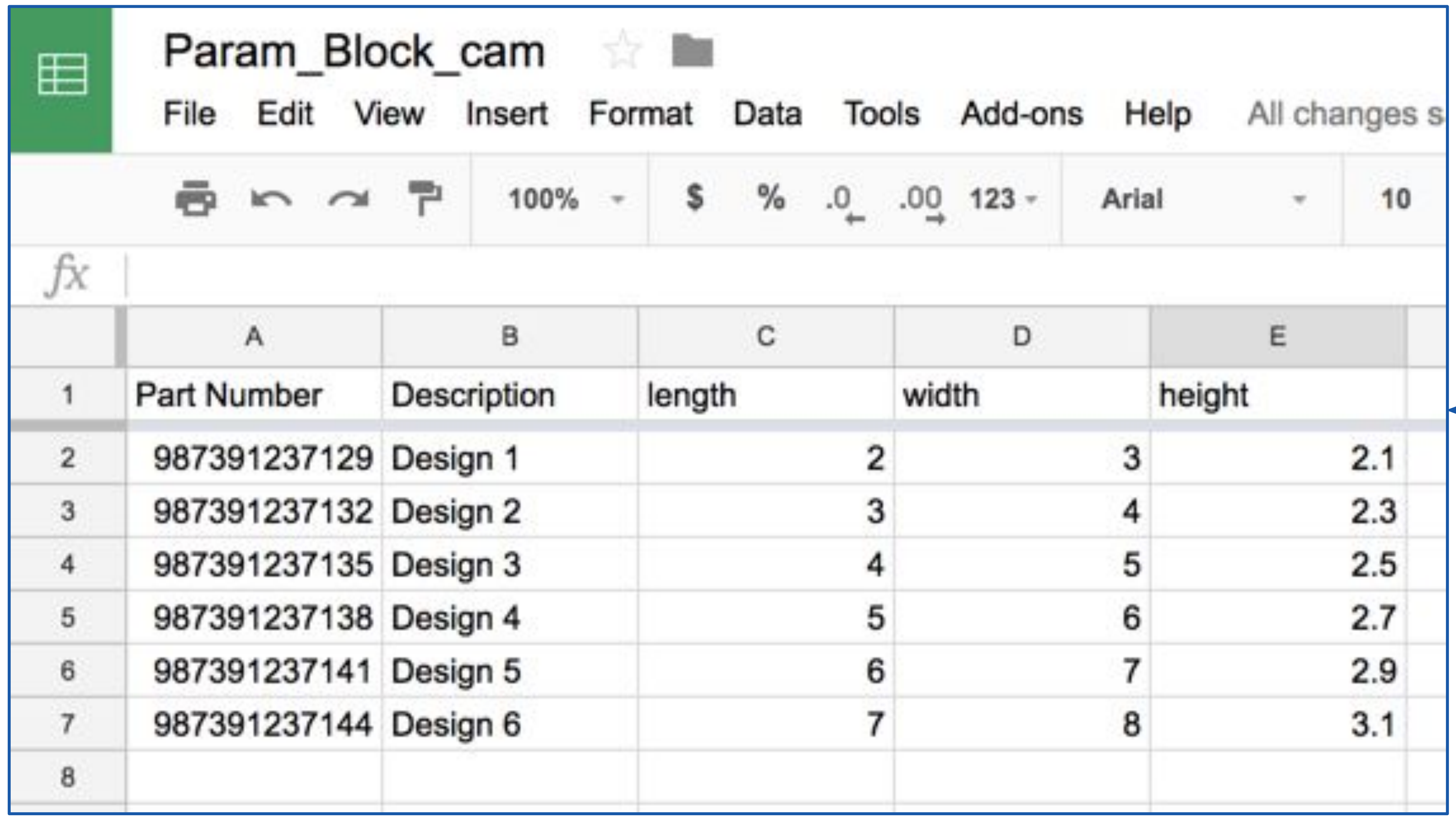




Plane: 1 selected
Radius: 1.00 in
Pitch: 1.00 in
Resolution: 25
Revolutions: 5



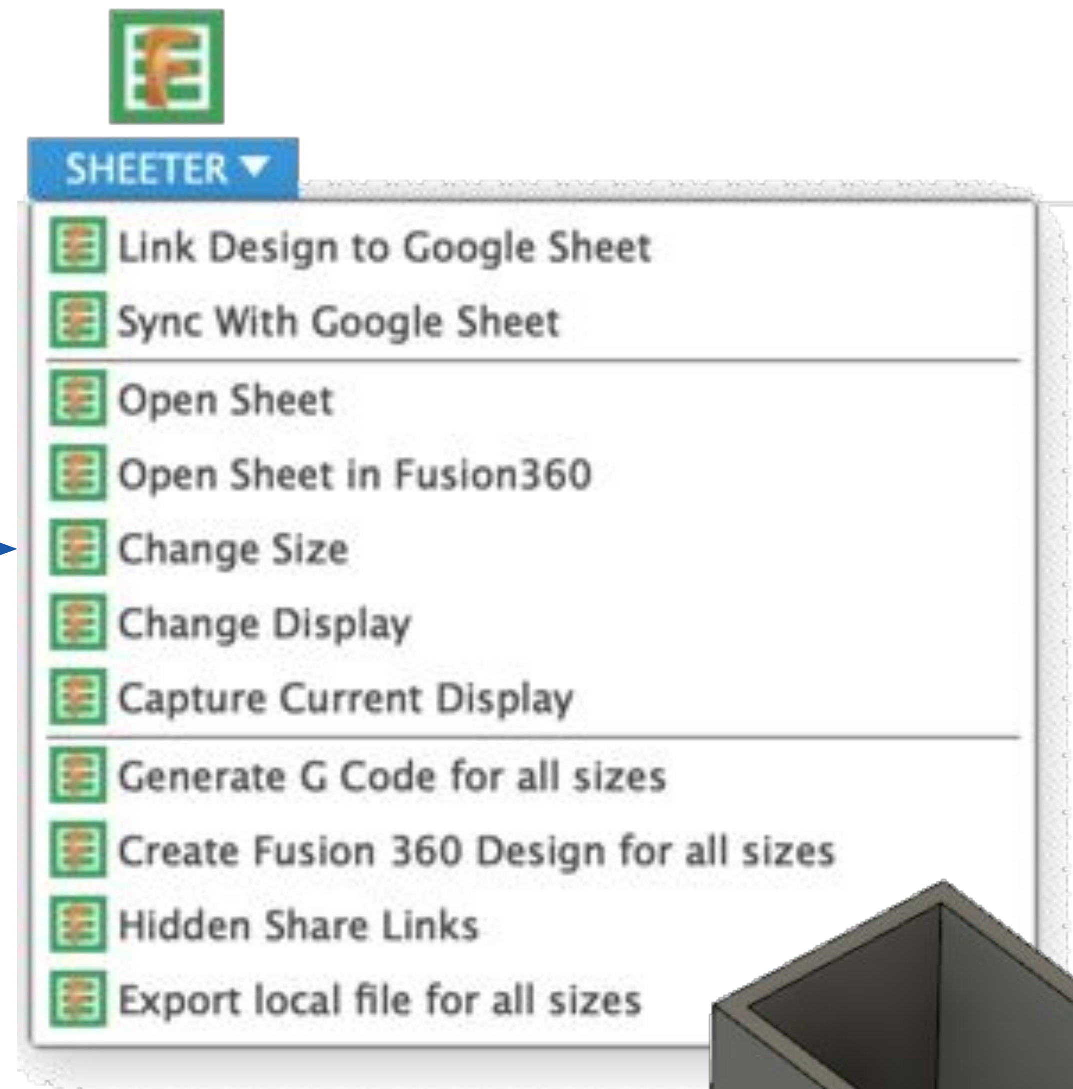
Automate Geometry Modification

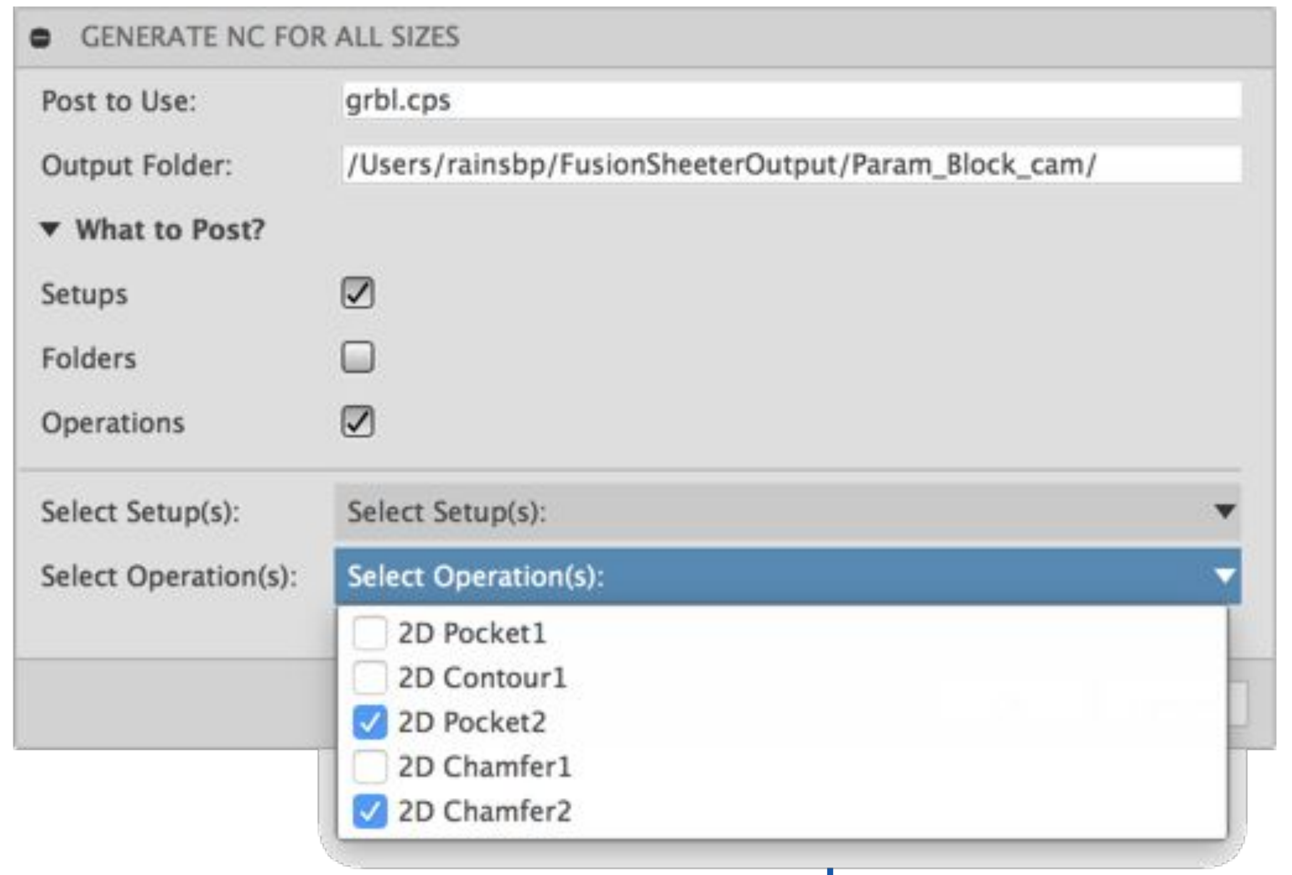


Param_Block_cam

File Edit View Insert Format Data Tools Add-ons Help All changes s

	A	B	C	D	E
1	Part Number	Description	length	width	height
2	987391237129	Design 1	2	3	2.1
3	987391237132	Design 2	3	4	2.3
4	987391237135	Design 3	4	5	2.5
5	987391237138	Design 4	5	6	2.7
6	987391237141	Design 5	6	7	2.9
7	987391237144	Design 6	7	8	3.1
8					





GENERATE NC FOR ALL SIZES

Post to Use: grbl.cps

Output Folder: /Users/rainsbp/FusionSheeterOutput/Param_Block_cam/

What to Post?

Setups ✓

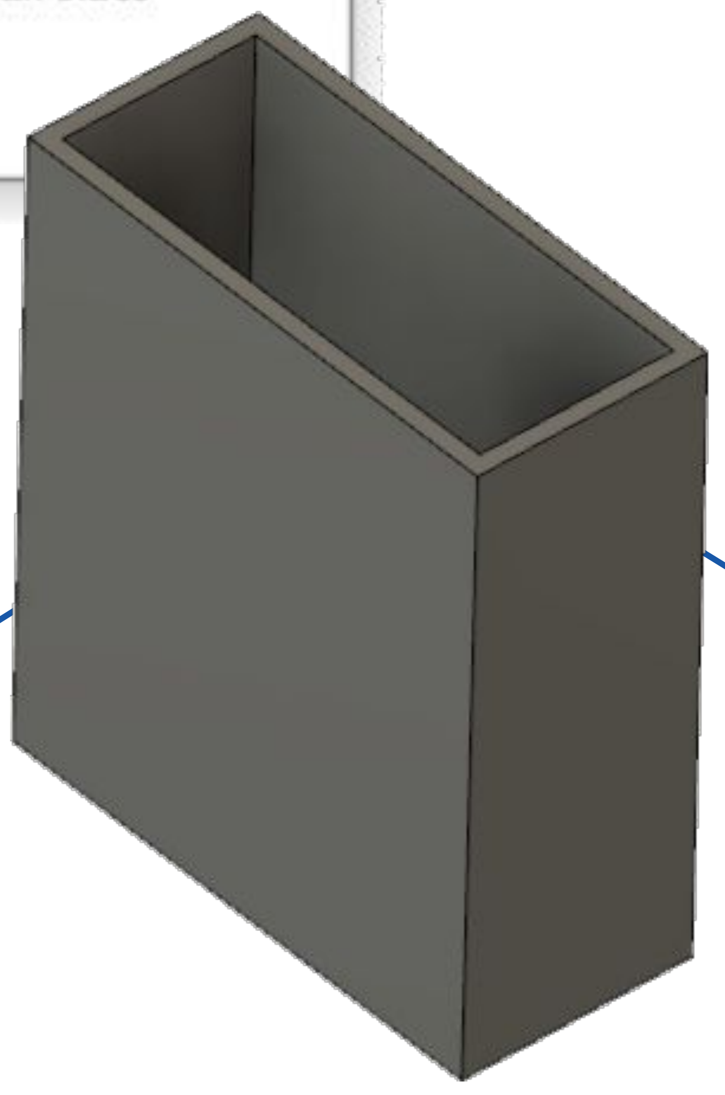
Folders ☐

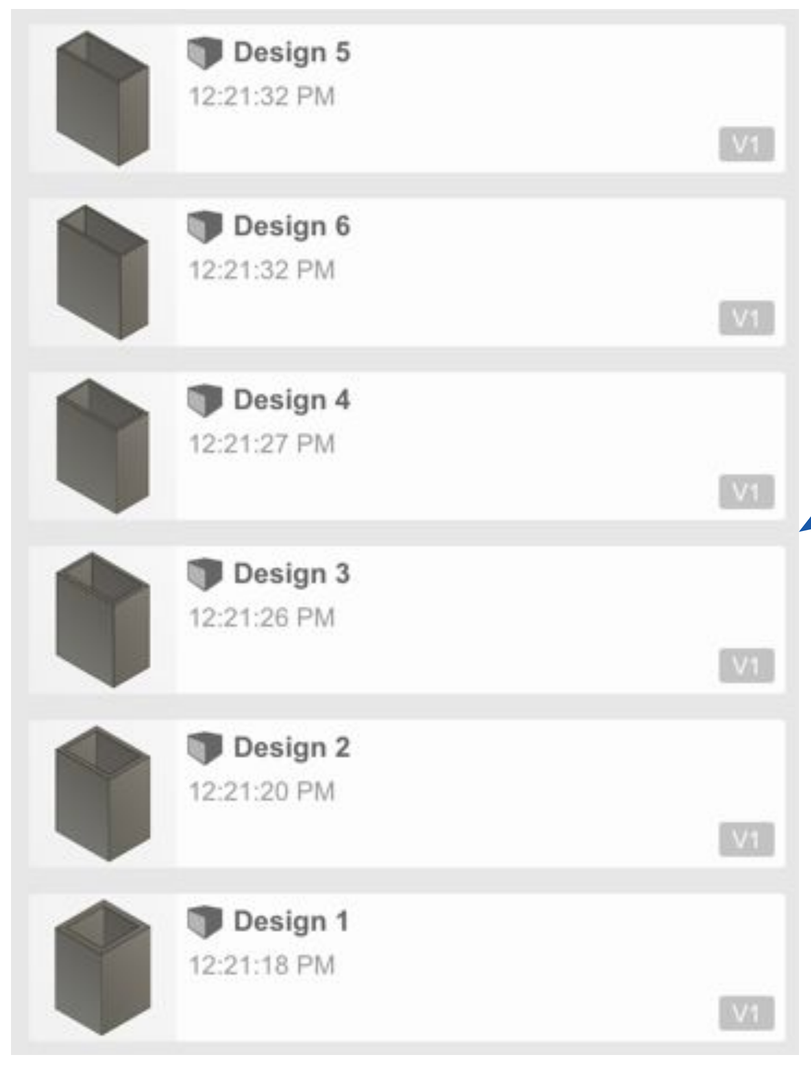
Operations ✓

Select Setup(s): Select Setup(s):

Select Operation(s): Select Operation(s):

- ☐ 2D Pocket1
- ☐ 2D Contour1
- ✓ 2D Pocket2
- ☐ 2D Chamfer1
- ✓ 2D Chamfer2





Design 5 12:21:32 PM V1

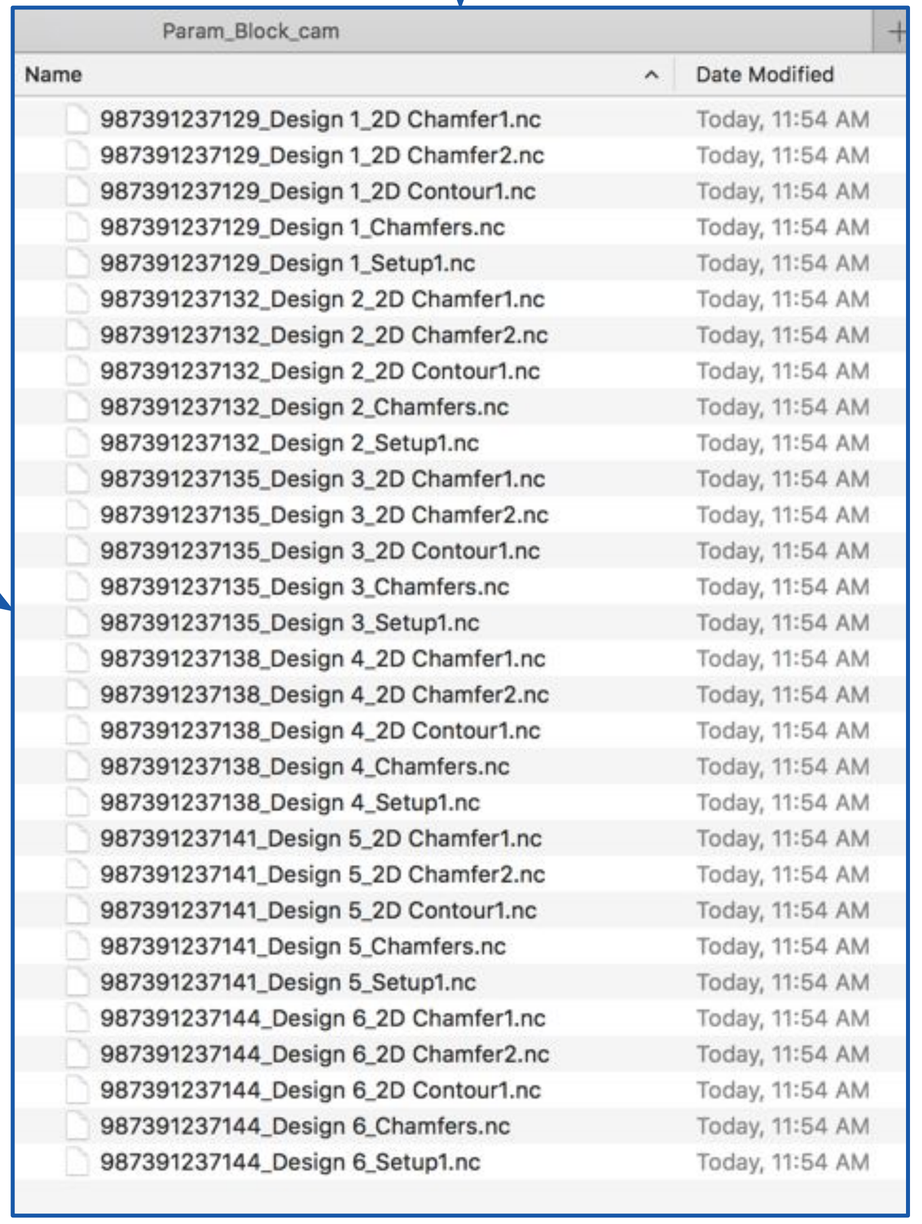
Design 6 12:21:32 PM V1

Design 4 12:21:27 PM V1

Design 3 12:21:26 PM V1

Design 2 12:21:20 PM V1

Design 1 12:21:18 PM V1



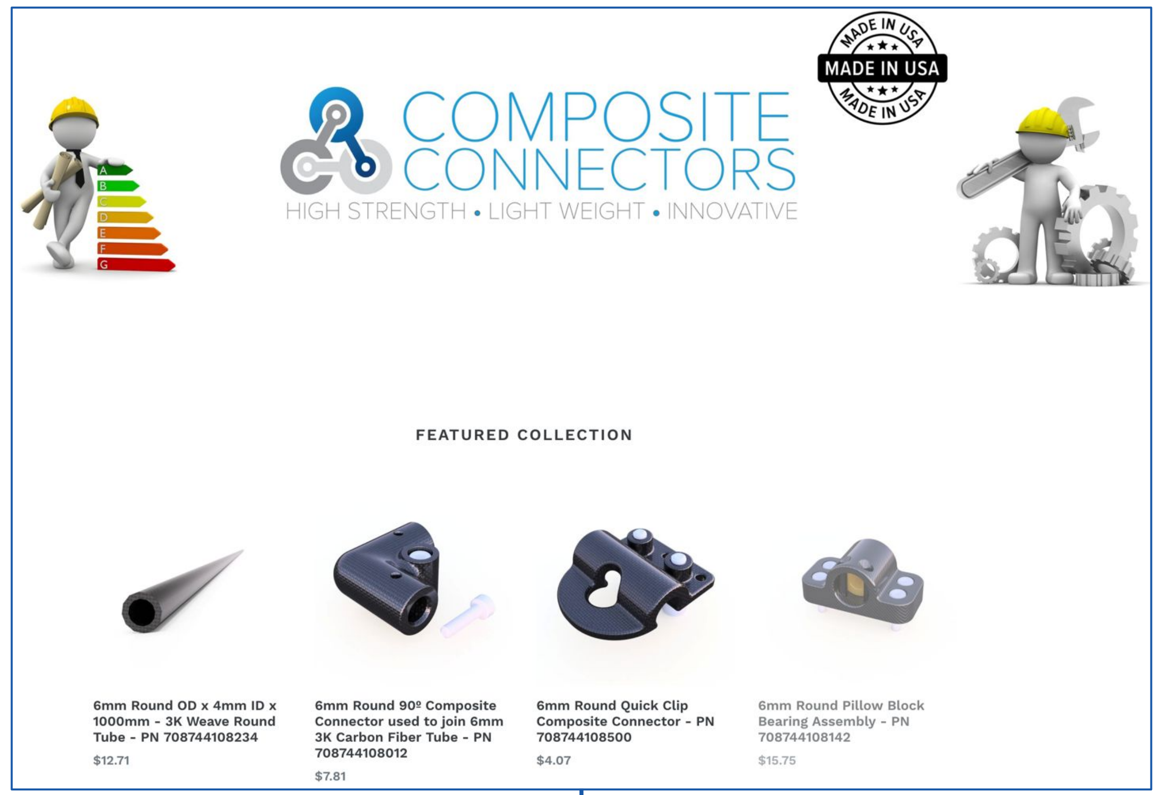
Param_Block_cam

Name	Date Modified
987391237129_Design 1_2D Chamfer1.nc	Today, 11:54 AM
987391237129_Design 1_2D Chamfer2.nc	Today, 11:54 AM
987391237129_Design 1_2D Contour1.nc	Today, 11:54 AM
987391237129_Design 1_Chamfers.nc	Today, 11:54 AM
987391237129_Design 1_Setup1.nc	Today, 11:54 AM
987391237132_Design 2_2D Chamfer1.nc	Today, 11:54 AM
987391237132_Design 2_2D Chamfer2.nc	Today, 11:54 AM
987391237132_Design 2_2D Contour1.nc	Today, 11:54 AM
987391237132_Design 2_Chamfers.nc	Today, 11:54 AM
987391237132_Design 2_Setup1.nc	Today, 11:54 AM
987391237135_Design 3_2D Chamfer1.nc	Today, 11:54 AM
987391237135_Design 3_2D Chamfer2.nc	Today, 11:54 AM
987391237135_Design 3_2D Contour1.nc	Today, 11:54 AM
987391237135_Design 3_Chamfers.nc	Today, 11:54 AM
987391237135_Design 3_Setup1.nc	Today, 11:54 AM
987391237138_Design 4_2D Chamfer1.nc	Today, 11:54 AM
987391237138_Design 4_2D Chamfer2.nc	Today, 11:54 AM
987391237138_Design 4_2D Contour1.nc	Today, 11:54 AM
987391237138_Design 4_Chamfers.nc	Today, 11:54 AM
987391237138_Design 4_Setup1.nc	Today, 11:54 AM
987391237141_Design 5_2D Chamfer1.nc	Today, 11:54 AM
987391237141_Design 5_2D Chamfer2.nc	Today, 11:54 AM
987391237141_Design 5_2D Contour1.nc	Today, 11:54 AM
987391237141_Design 5_Chamfers.nc	Today, 11:54 AM
987391237141_Design 5_Setup1.nc	Today, 11:54 AM
987391237144_Design 6_2D Chamfer1.nc	Today, 11:54 AM
987391237144_Design 6_2D Chamfer2.nc	Today, 11:54 AM
987391237144_Design 6_2D Contour1.nc	Today, 11:54 AM
987391237144_Design 6_Chamfers.nc	Today, 11:54 AM
987391237144_Design 6_Setup1.nc	Today, 11:54 AM

Google Sheets Integration

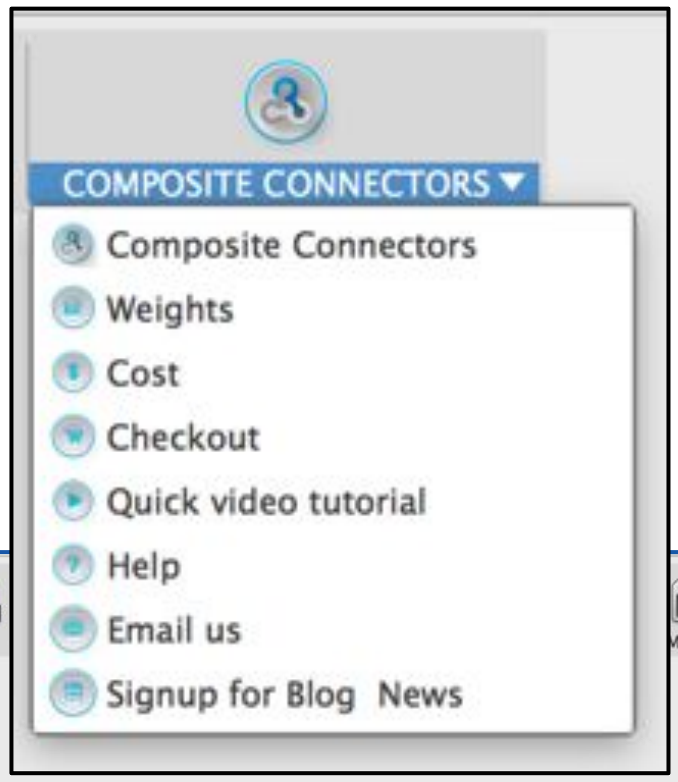
- Synchronize Parameters
- Export multiple sizes
- Post process multiple sizes
- Save display states

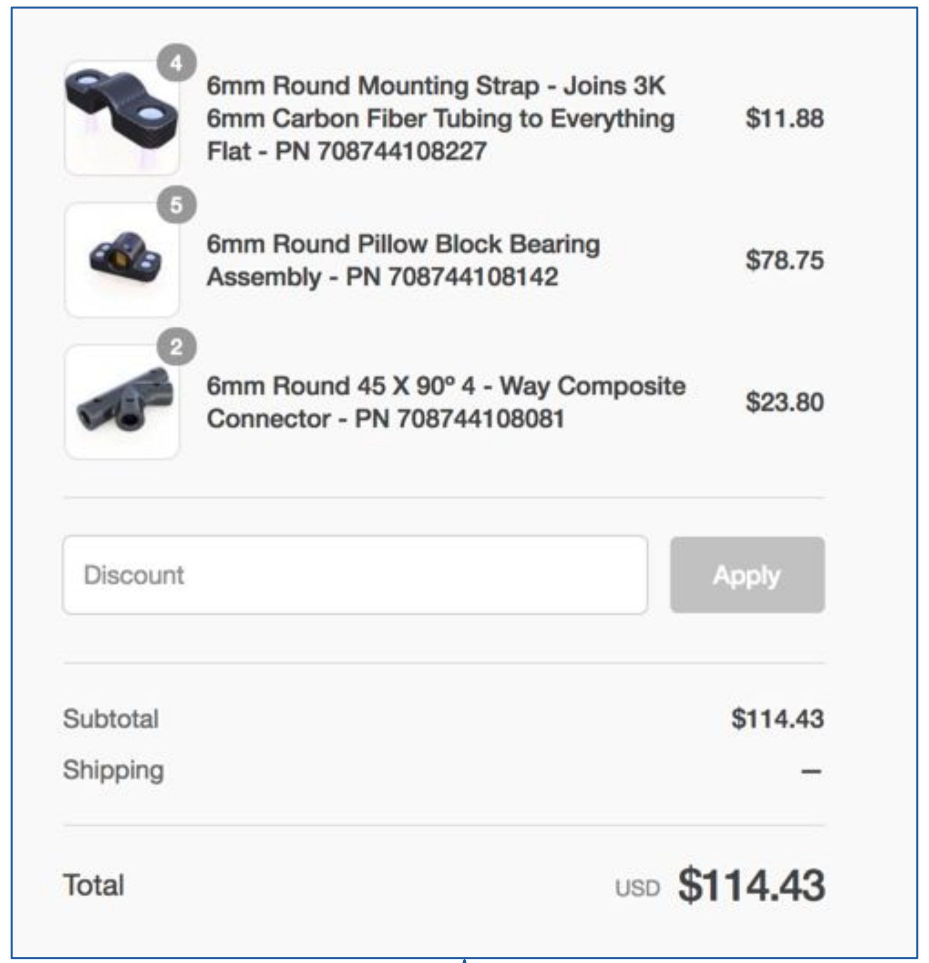
Interrogate and Analyze Geometry



FEATURED COLLECTION

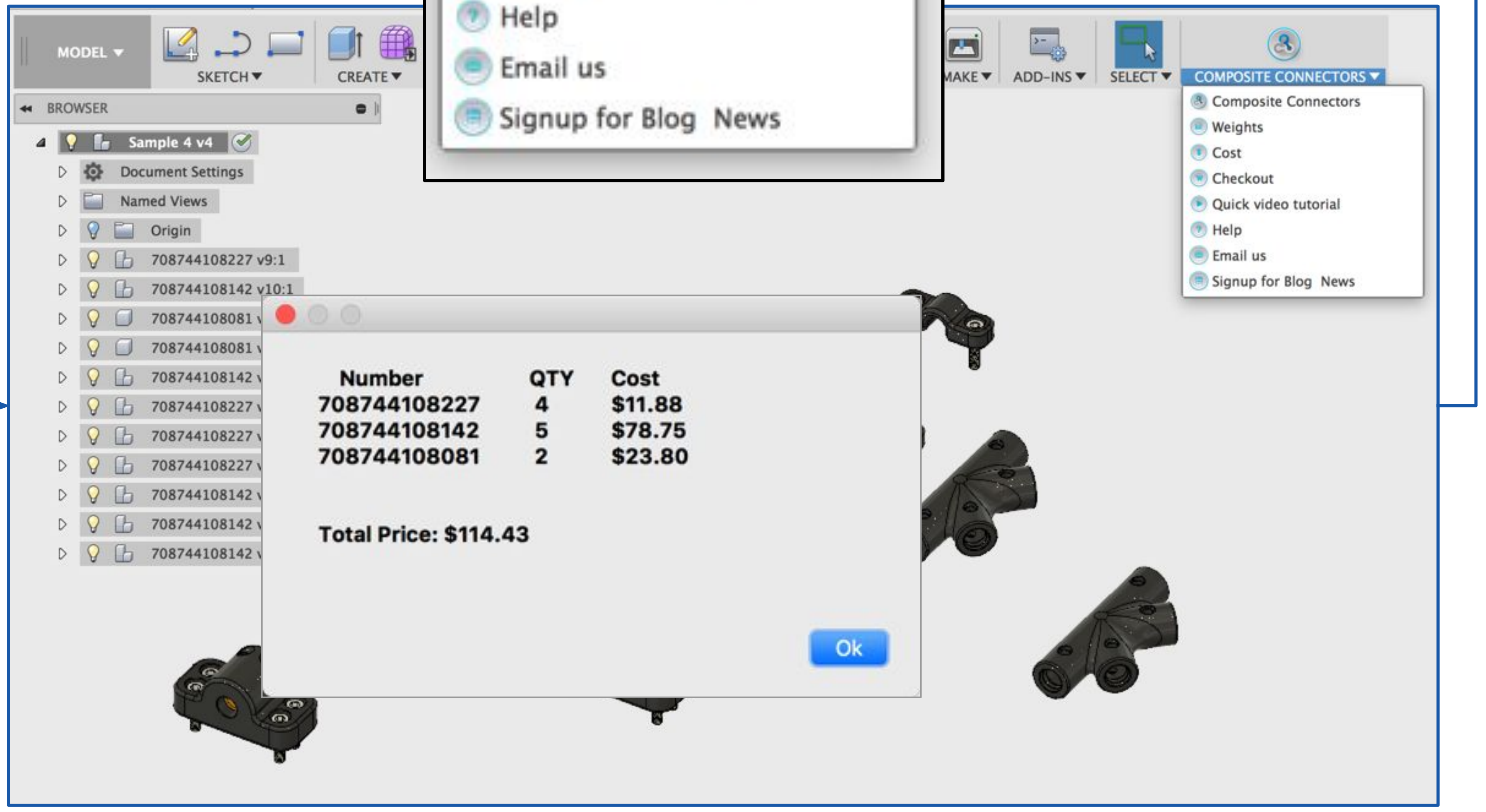
6mm Round OD x 4mm ID x 1000mm - 3K Weave Round Tube - PN 708744108234	6mm Round 90º Composite Connector used to join 6mm 3K Carbon Fiber Tube - PN 708744108012	6mm Round Quick Clip Composite Connector - PN 708744108500	6mm Round Pillow Block Bearing Assembly - PN 708744108142
$12.71	$7.81	$4.07	$15.75



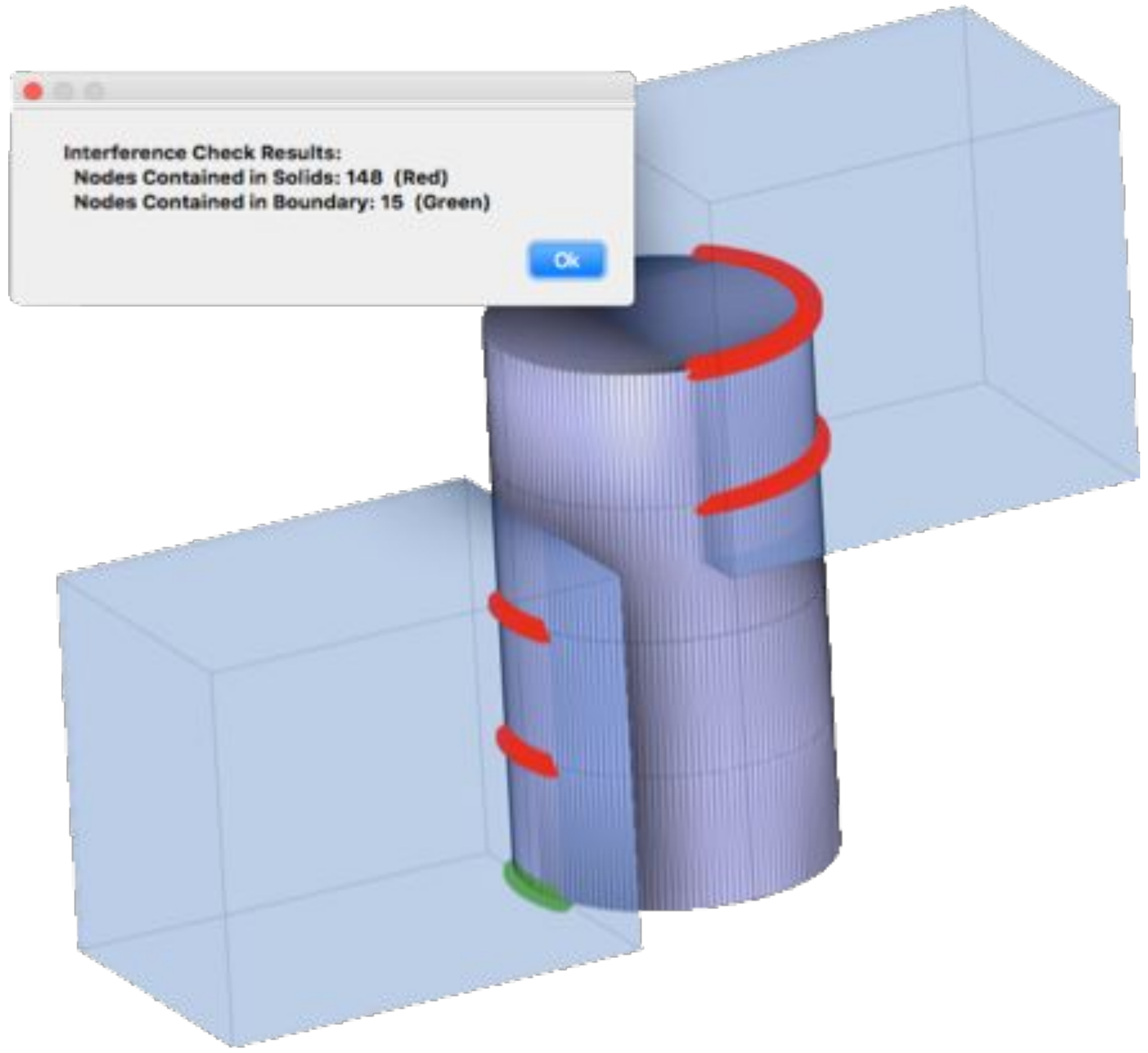


4	6mm Round Mounting Strap - Joins 3K 6mm Carbon Fiber Tubing to Everything Flat - PN 708744108227	$11.88
5	6mm Round Pillow Block Bearing Assembly - PN 708744108142	$78.75
2	6mm Round 45 X 90º 4 - Way Composite Connector - PN 708744108081	$23.80

Subtotal $114.43
Shipping —
Total USD $114.43

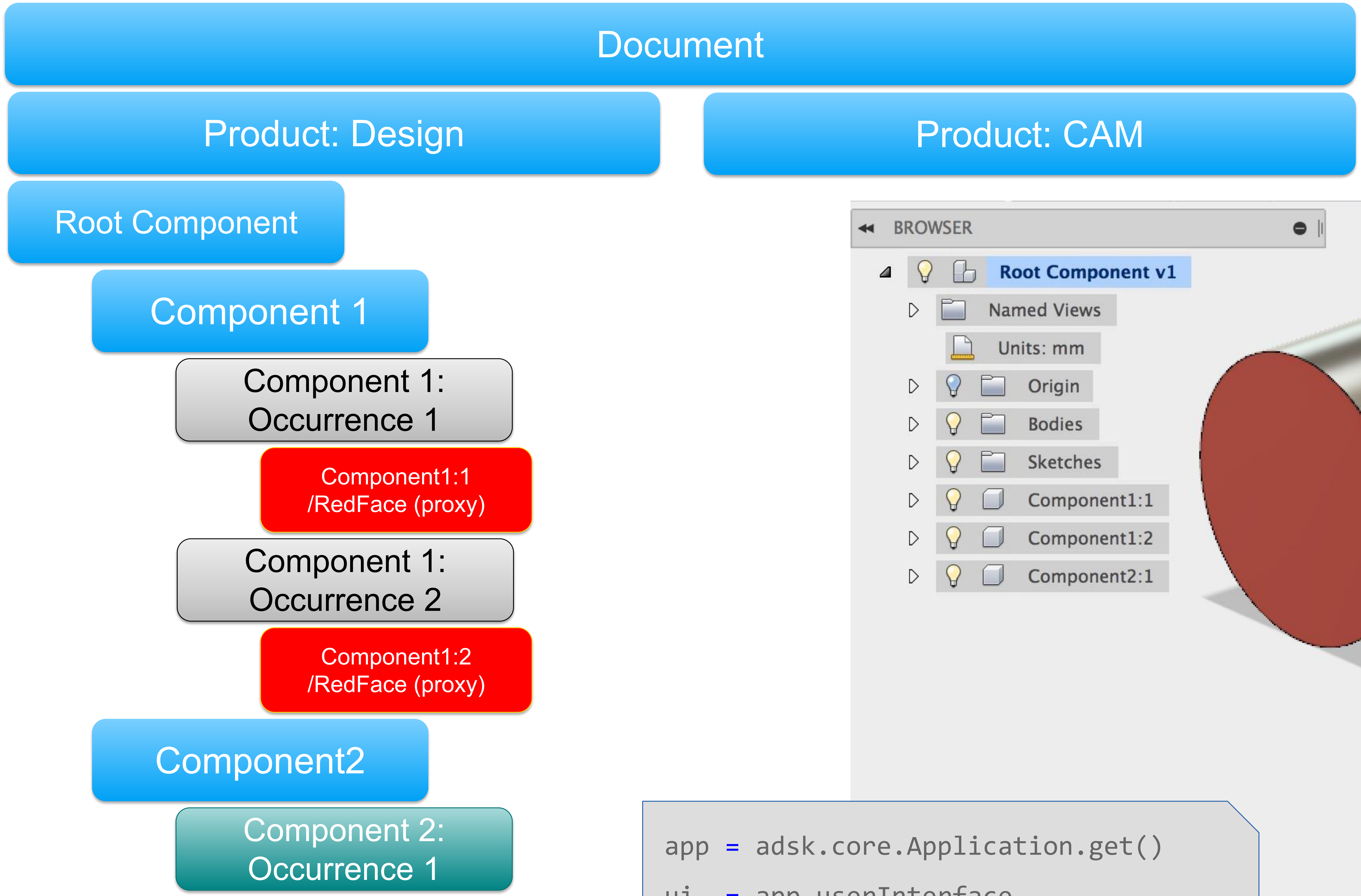


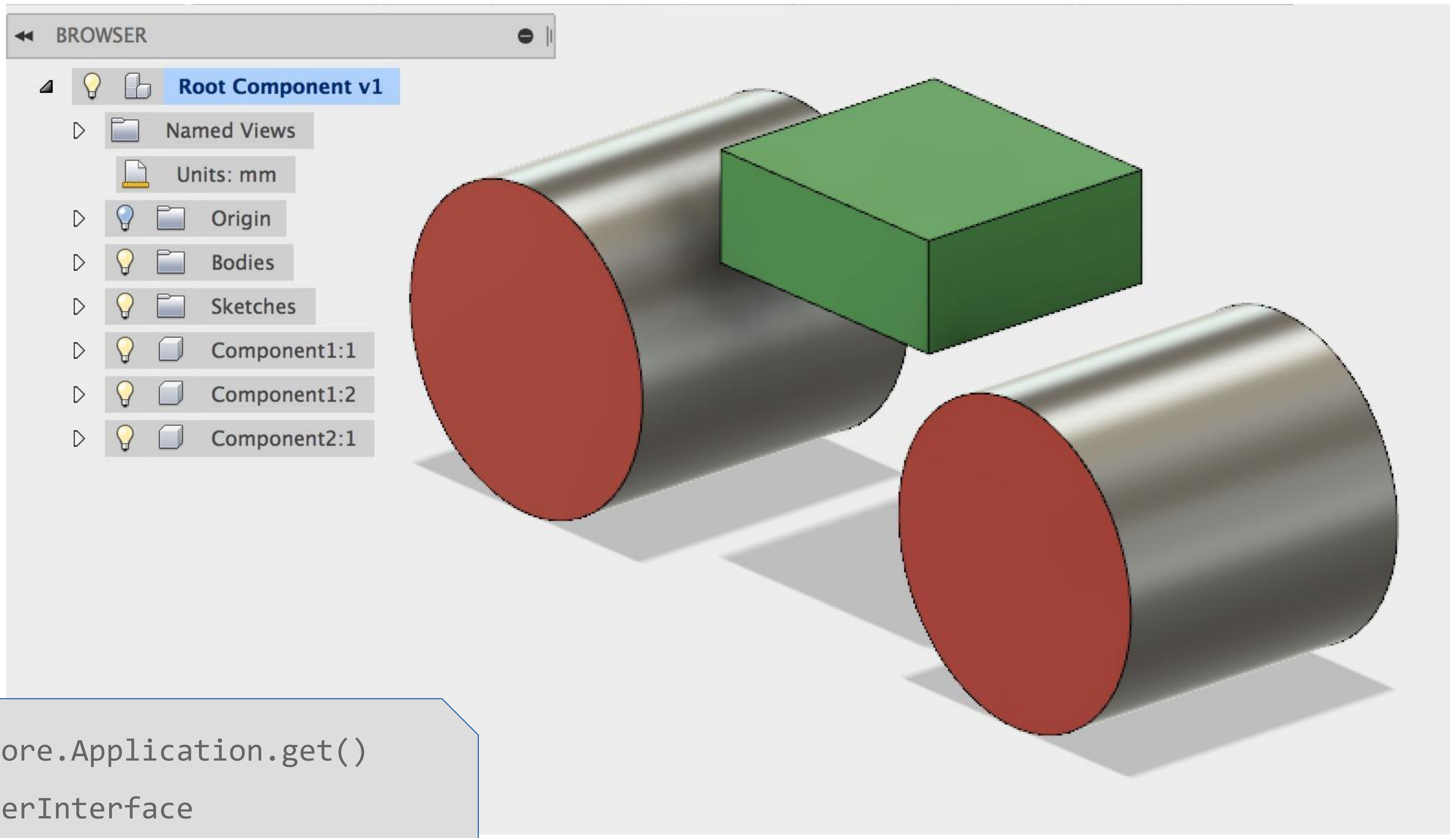
```
   Number        QTY    Cost
708744108227      4     $11.88
708744108142      5     $78.75
708744108081      2     $23.80

Total Price: $114.43
```



```
Interference Check Results:
  Nodes Contained in Solids: 148  (Red)
  Nodes Contained in Boundary: 15  (Green)
```

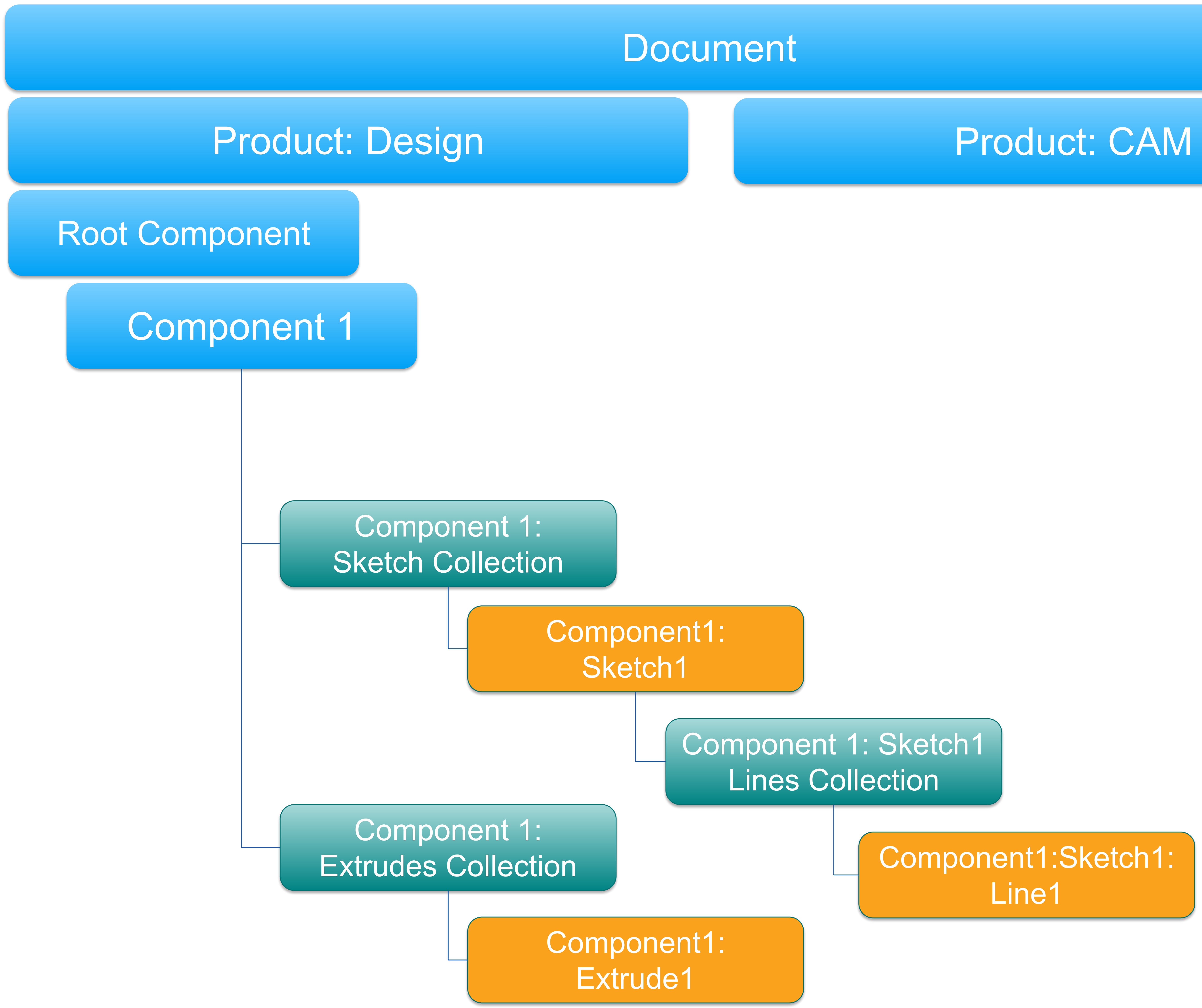
Fusion 360 Document Structure


Document
Product: Design
Product: CAM
Root Component
Component 1
Component 1: Occurrence 1
Component1:1 /RedFace (proxy)
Component 1: Occurrence 2
Component1:2 /RedFace (proxy)
Component2
Component 2: Occurrence 1



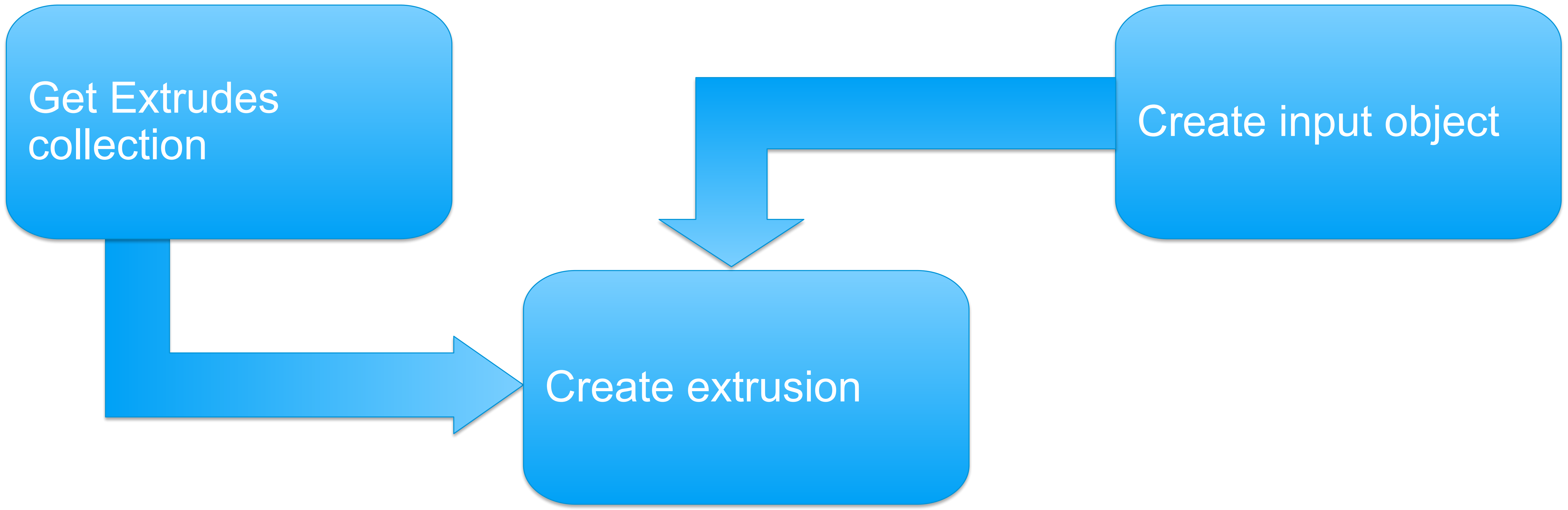
```
app = adsk.core.Application.get()
ui  = app.userInterface
design = app.activeProduct
```

Features and Collections

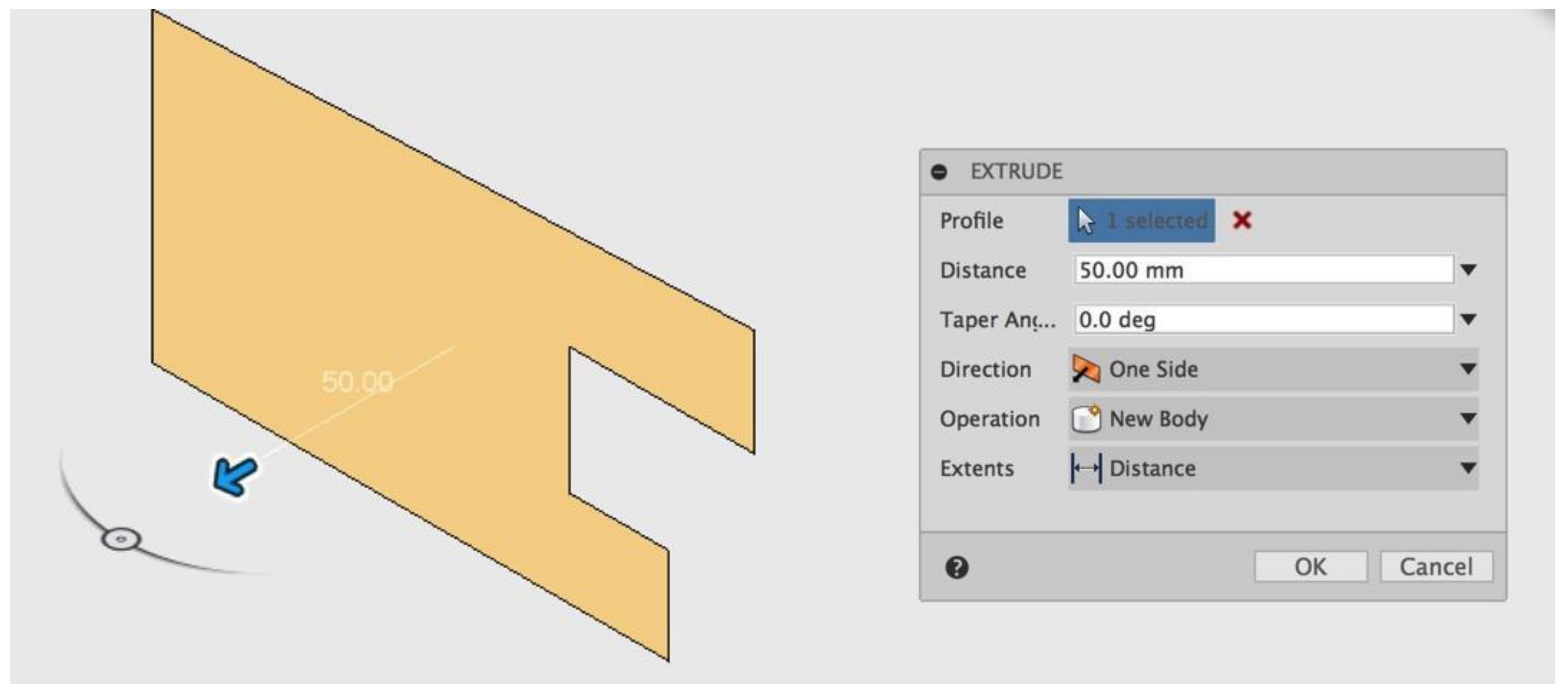

Document
Product: Design
Product: CAM
Root Component
Component 1
Component 1: Sketch Collection
Component1: Sketch1
Component 1: Sketch1 Lines Collection
Component1:Sketch1: Line1
Component 1: Extrudes Collection
Component1: Extrude1

```
# Get reference to the root component
rootComp = design.rootComponent

#Get reference to the sketches and plane
sketches = rootComp.sketches

#Create a new sketch and get lines reference
sketch = sketches.add(rootComp.xYConstructionPlane)
lines = sketch.sketchCurves.sketchLines
lines.addByTwoPoints(point0, point1)
```

Creating Features (Extrude)



- Define parameters for feature
- Define profile(s) for feature
- Define distance



```
EXTRUDE
Profile      1 selected
Distance     50.00 mm
Taper Ang... 0.0 deg
Direction    One Side
Operation    New Body
Extents      Distance
                         OK  Cancel
```

```python
# Get the profile defined by the sketch
profile = sketch.profiles.item(0)

# Create an extrusion input
extrudes = rootComp.features.extrudeFeatures
operation_type = adsk.fusion.FeatureOperations.NewBodyFeatureOperation
ext_input = extrudes.createInput(profile, operation_type)
```

Units in Fusion 360

Fusion Default Model Units

cm (*areas and volumes are cm^2 and cm^3*)

radians

kg

```
# Define that the extent is a distance extent of 1 cm
distance = adsk.core.ValueInput.createByReal(1)

# Set the distance extent to be single direction
ext_input.setDistanceExtent(False, distance)

# Set the extrude to be a solid one
ext_input.isSolid = True

# Create the extrusion
extrudes.add(ext_input)
```

Active units and feature definitions

Scripts must adapt to user changing units

Most features look for “Value Inputs” not raw values

var x = adsk.core.ValueInput.createByReal(23)

var x = adsk.core.ValueInput.createByString(“23 in”);

UnitsManager is a utility for values and units.

convert(1.5, "in", "ft") -> 0.125

evaluateExpression(“3 in * 5 in”, “in”) -> 38.1

formatInternalValue(2000, “ft*ft*ft“, true) -> "0.070629 ft^3"

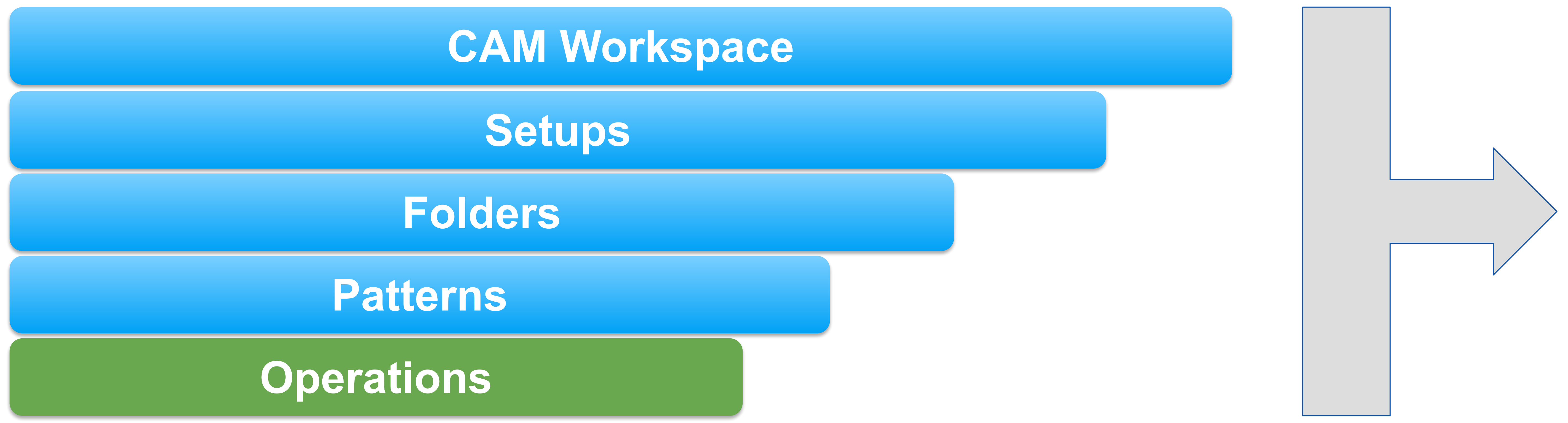
standardizeExpression("1.5", "in") -> "1.5 in"

Full Script

```python
# Author-Patrick Rainsberry
# Description-Basic Script to create a block

import adsk.core, adsk.fusion, adsk.cam, traceback

def run(context):
    ui = None
    try:
        app = adsk.core.Application.get()
        ui  = app.userInterface
        design = app.activeProduct

        # Get reference to the root component
        rootComp = design.rootComponent

        #Get reference to the sketchs and plane
        sketches = rootComp.sketches
        xyPlane = rootComp.xYConstructionPlane

        #Create a new sketch and get lines reference
        sketch = sketches.add(xyPlane)
        lines = sketch.sketchCurves.sketchLines

        # Use autodesk methods to create input geometry
        point0 = adsk.core.Point3D.create(0, 0, 0)
        point1 = adsk.core.Point3D.create(0, 1, 0)
        point2 = adsk.core.Point3D.create(1, 1, 0)
        point3 = adsk.core.Point3D.create(1, 0, 0)

        # Create lines
        lines.addByTwoPoints(point0, point1)
        lines.addByTwoPoints(point1, point2)
        lines.addByTwoPoints(point2, point3)
        lines.addByTwoPoints(point3, point0)

       # Get the profile defined by the square
        profile = sketch.profiles.item(0)

        # Create an extrusion input
        extrudes = rootComp.features.extrudeFeatures
        operation_type = adsk.fusion.FeatureOperations.NewBodyFeatureOperation
        ext_input = extrudes.createInput(profile, operation_type)

        # Define that the extent is a distance extent of 1 cm
        distance = adsk.core.ValueInput.createByReal(1)

        # Set the distance extent to be single direction
        ext_input.setDistanceExtent(False, distance)

        # Set the extrude to be a solid one
        ext_input.isSolid = True

        # Create the extrusion
        extrudes.add(ext_input)

    except:
        if ui:
            ui.messageBox('Failed:\n{}'.format(traceback.format_exc()))
```

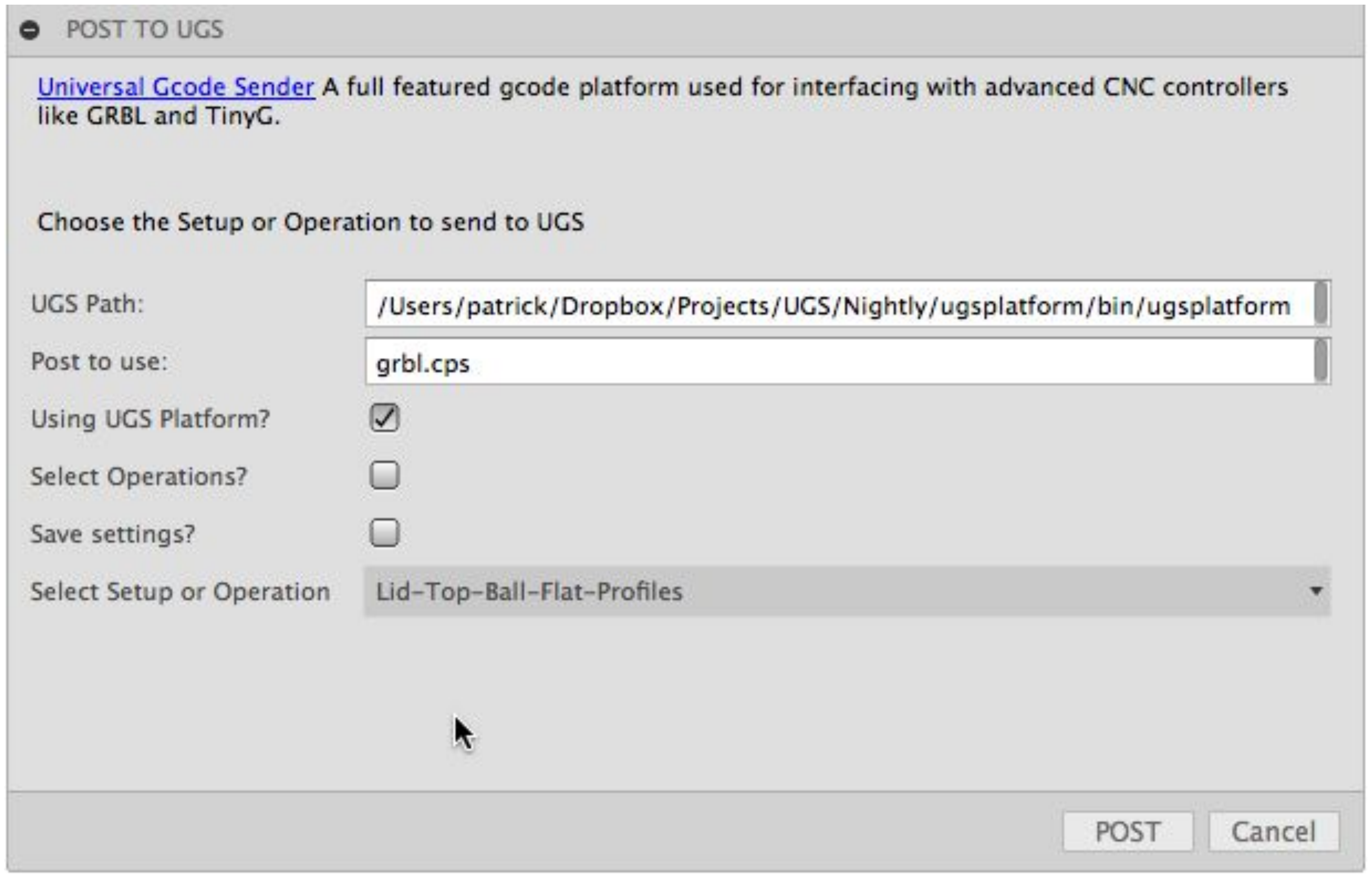
CAM API

Interrogate Basic CAM Information



Regenerate Toolpaths
Create Setup Sheets
Post Process one or more items
Get machining time for 1 or more items

Automate Post Processing

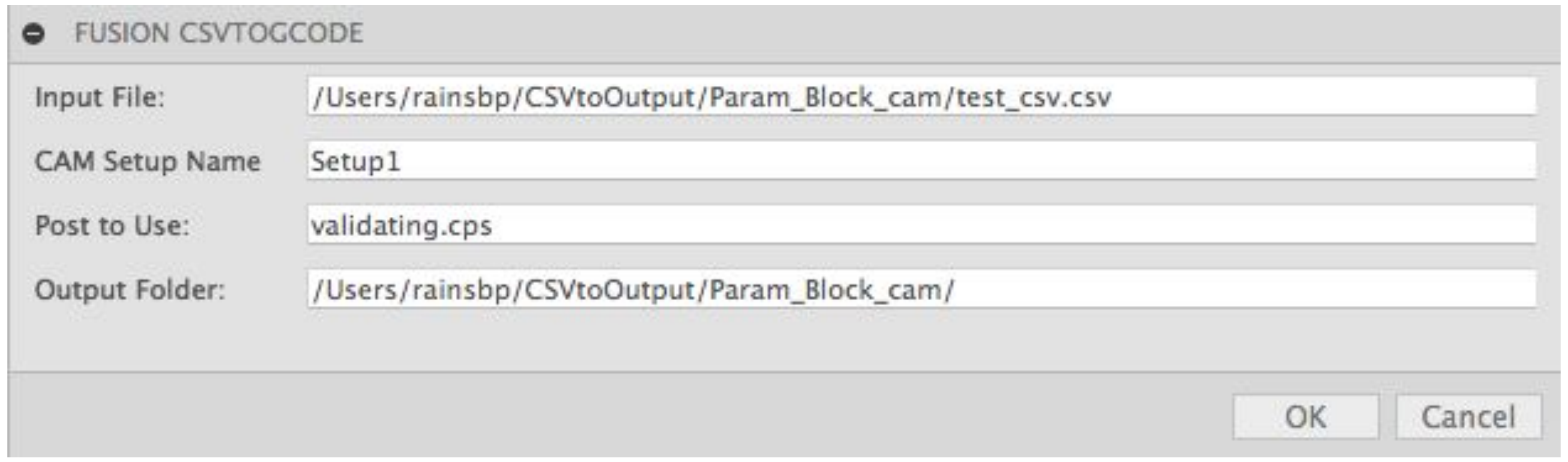


Universal Gcode Sender A full featured gcode platform used for interfacing with advanced CNC controllers like GRBL and TinyG.

Choose the Setup or Operation to send to UGS

UGS Path:	/Users/patrick/Dropbox/Projects/UGS/Nightly/ugsplatform/bin/ugsplatform
Post to use:	grbl.cps
Using UGS Platform?	☑
Select Operations?	☐
Save settings?	☐
Select Setup or Operation	Lid-Top-Ball-Flat-Profiles

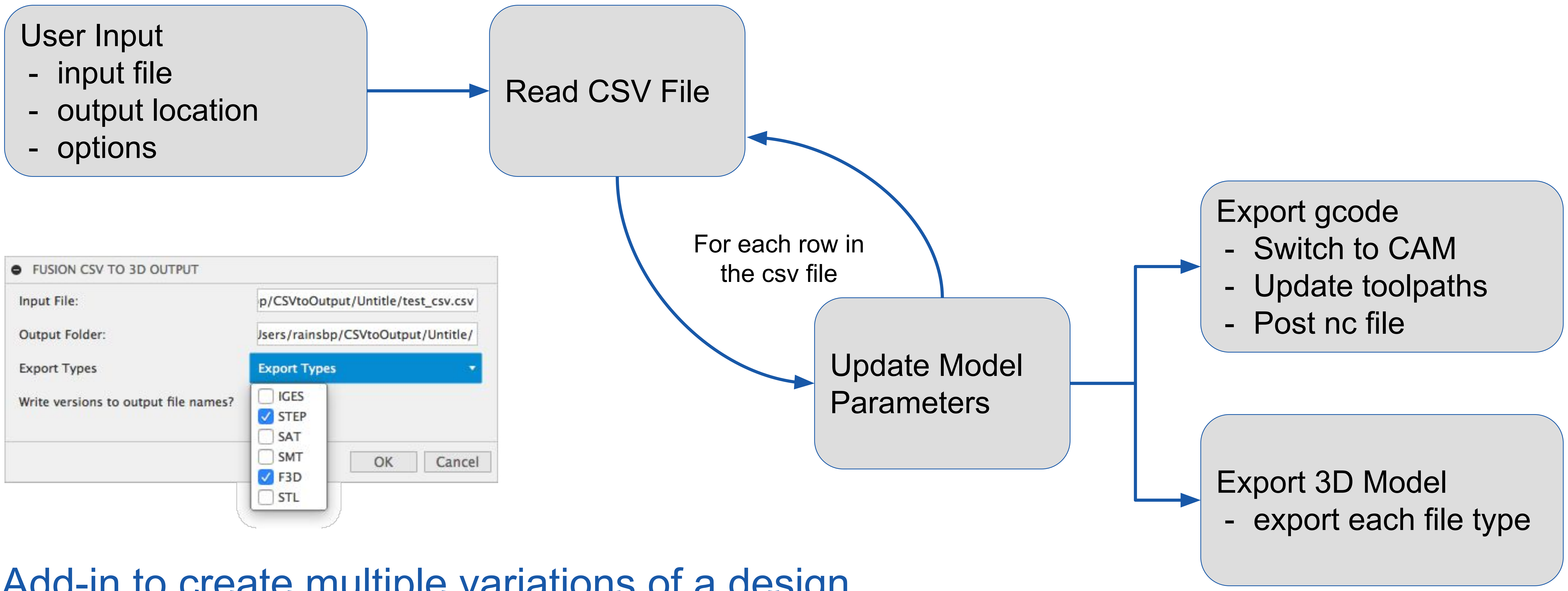
Post process and automatically send to controller software

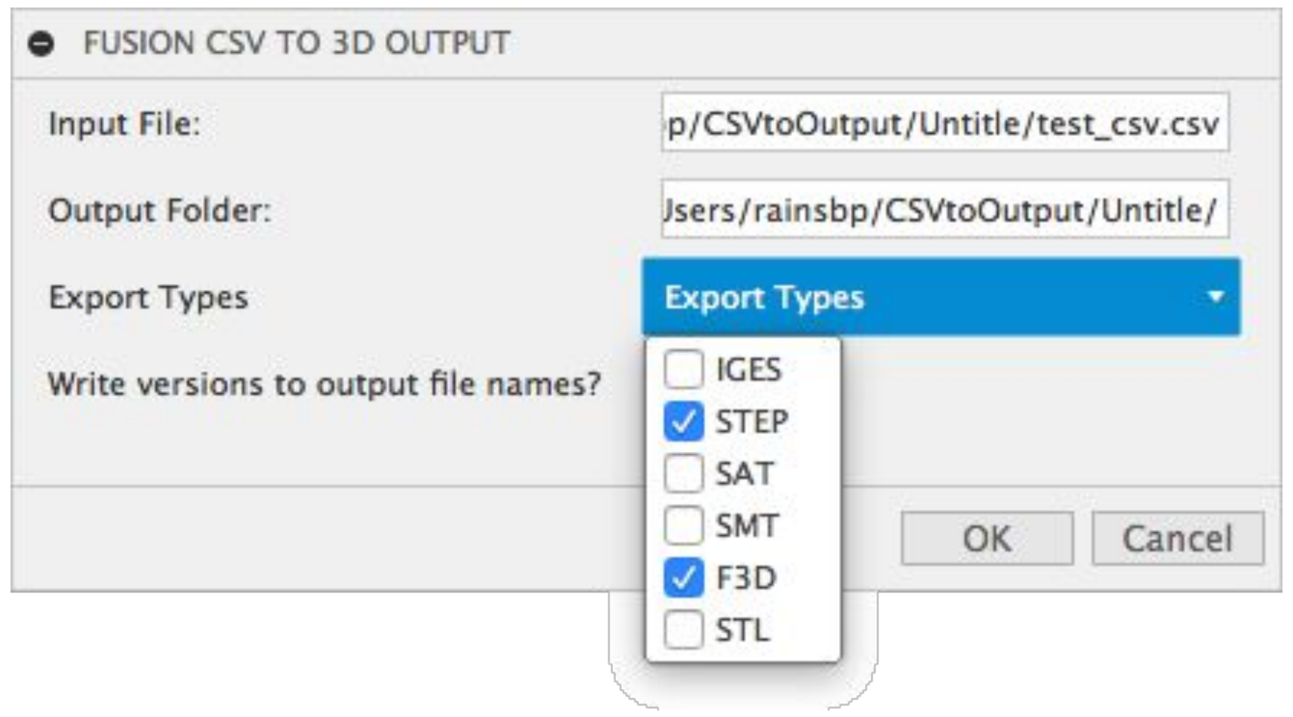


Input File:	/Users/rainsbp/CSVtoOutput/Param_Block_cam/test_csv.csv
CAM Setup Name	Setup1
Post to Use:	validating.cps
Output Folder:	/Users/rainsbp/CSVtoOutput/Param_Block_cam/

Read parameters, post process, for every row in a csv file

Automating Geometry Changes and Outputs




FUSION CSV TO 3D OUTPUT

Input File: p/CSVtoOutput/Untitle/test_csv.csv

Output Folder: Jsers/rainsbp/CSVtoOutput/Untitle/

Export Types: Export Types

- [ ] IGES
- [x] STEP
- [ ] SAT
- [ ] SMT
- [x] F3D
- [ ] STL

Write versions to output file names?

Add-in to create multiple variations of a design

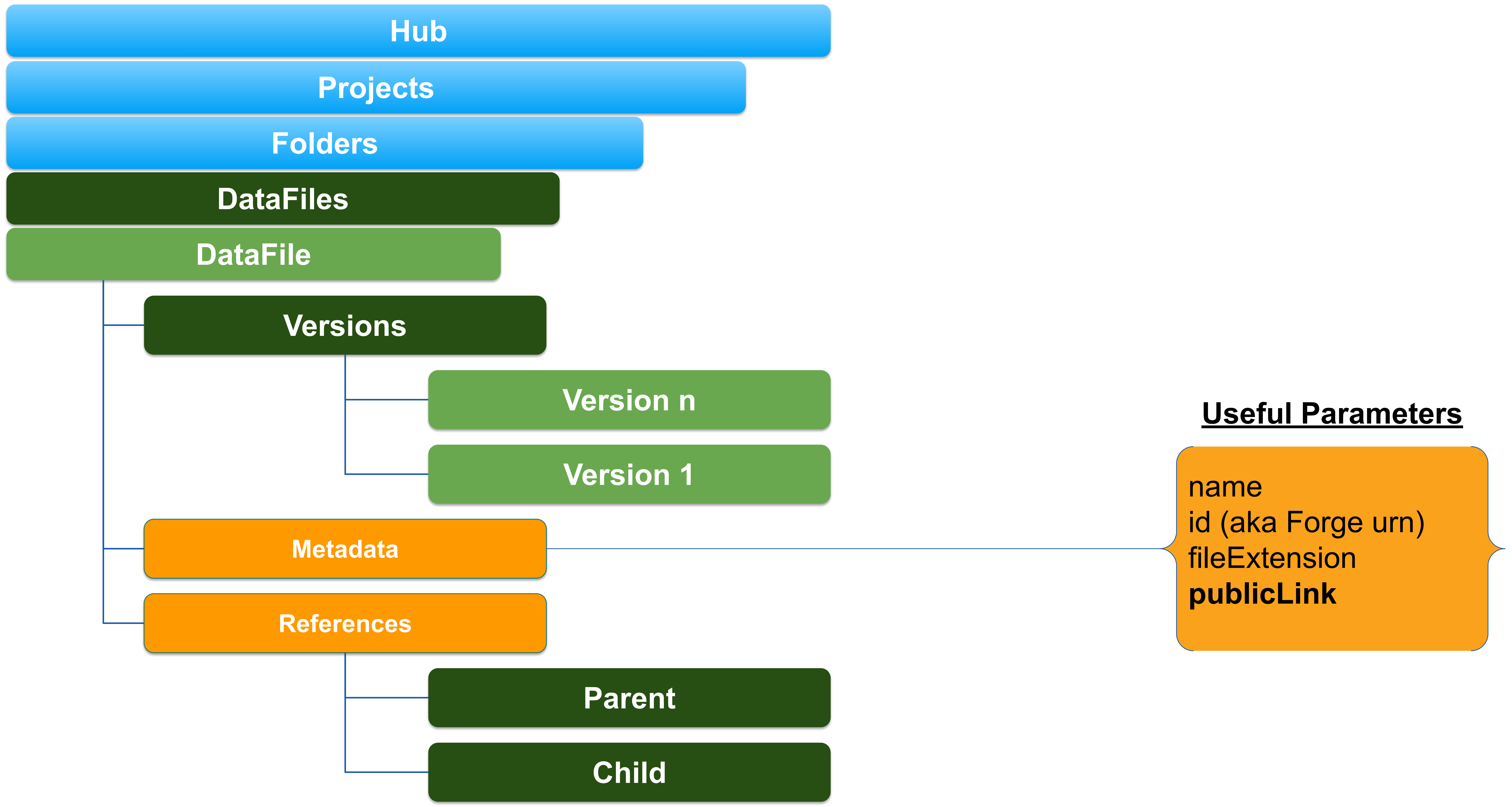
- Export 3D files (Step, IGES, SAT, f3d)
- Output g-code for an existing setup or operation

CAM API Sample

```
import time
TIMEOUT = 10

# Find setup
for setup in cam.setups:
    if setup.name == setup_name:
        to_post = setup

        # Update tool path
        future = cam.generateToolpath(to_post)

        check = 0
        while not future.isGenerationCompleted:
            adsk.doEvents()
            time.sleep(1)
            check += 1
            if check > TIMEOUT:
                ao['ui'].messageBox('Timeout')
                break

        # Set the post options
        post_config = os.path.join(cam.genericPostFolder, post_name)
        units = adsk.cam.PostOutputUnitOptions.DocumentUnitsOutput

        # create the postInput object
        post_input = adsk.cam.PostProcessInput.create(setup_name, post_config, output_folder, units)
        post_input.isOpenInEditor = False

        cam.postProcess(to_post, post_input)
```

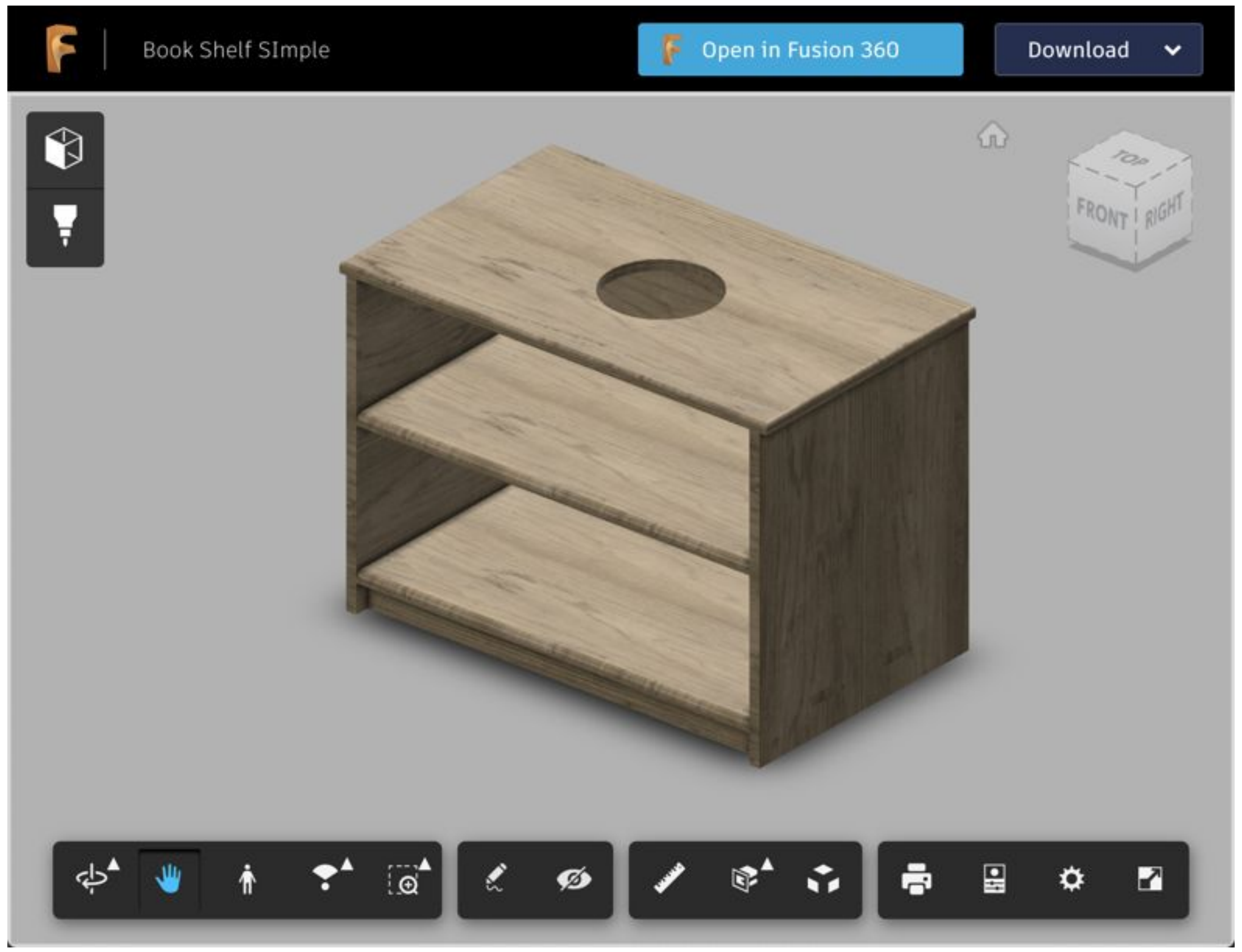
Data API

Interrogate and Manipulate Fusion 360 Data



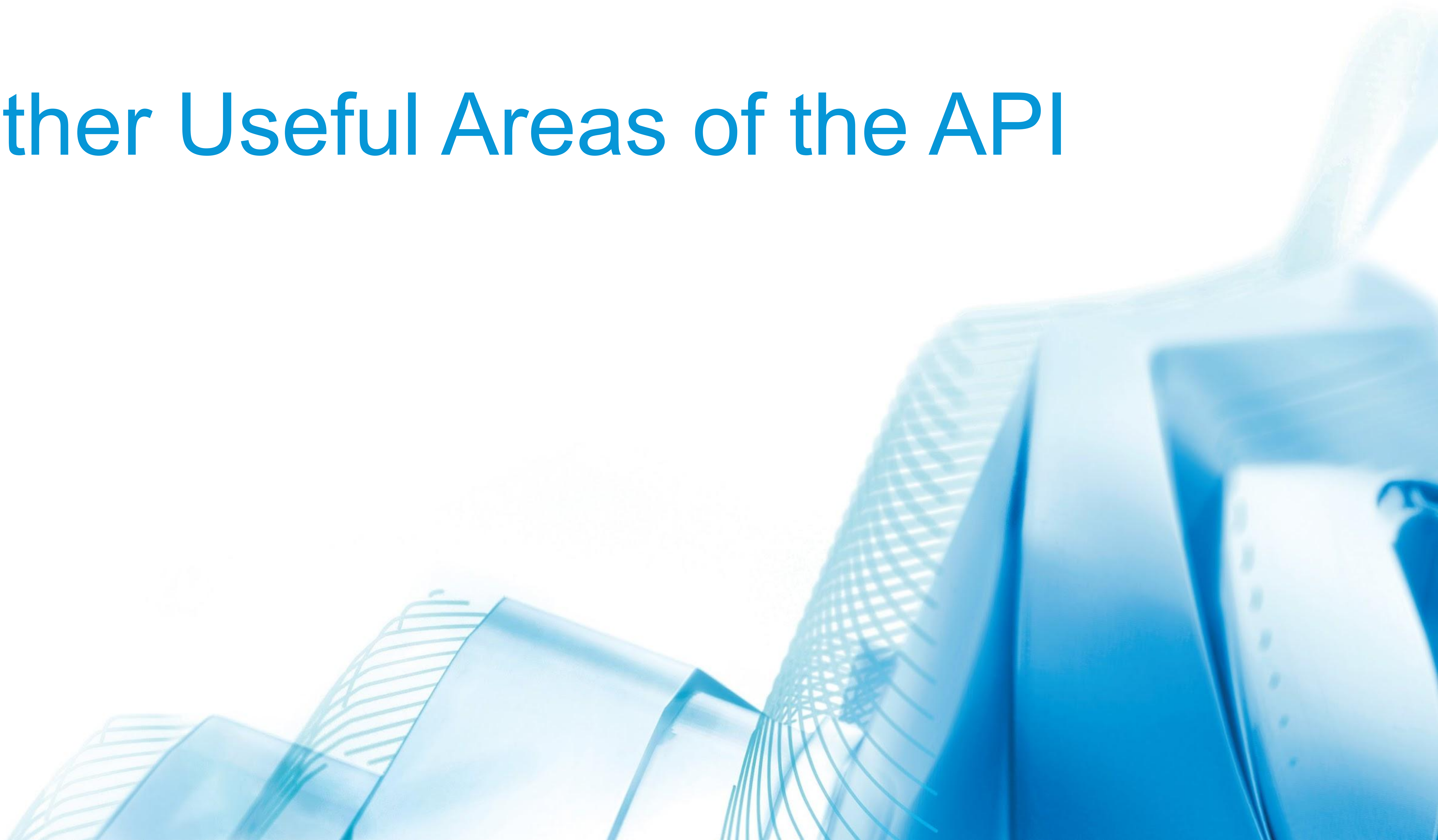
Data API Example

```
for data_file in app.activeDocument.dataFile.parentFolder.dataFiles:
  if data_file.fileExtension == "f3d":
    if test_name == data_file.name:
      short_public_link = data_file.publicLink
      public_link = un_shorten_url(short_public_link)
      custom_properties = {
        "short_public_link": short_public_link,
        "public_link": public_link,
        "public_link_id": public_link.split("/")[-1],
        "forge_urn": data_file.id,
        "forge_id": data_file.id.split(":")[-1]
      }
```



https://a360.co/2I3ANIa
https://autodesk3008.autodesk360.com/g/shares/SH56a43QTfd62c1cd9683a5210ff3637dbfa
SH56a43QTfd62c1cd9683a5210ff3637dbfa
urn:adsk.wipprod:dm.lineage:L-WnvzX-QiindaIZkCIelss
L-WnvzX-QiindaIZkCIelss

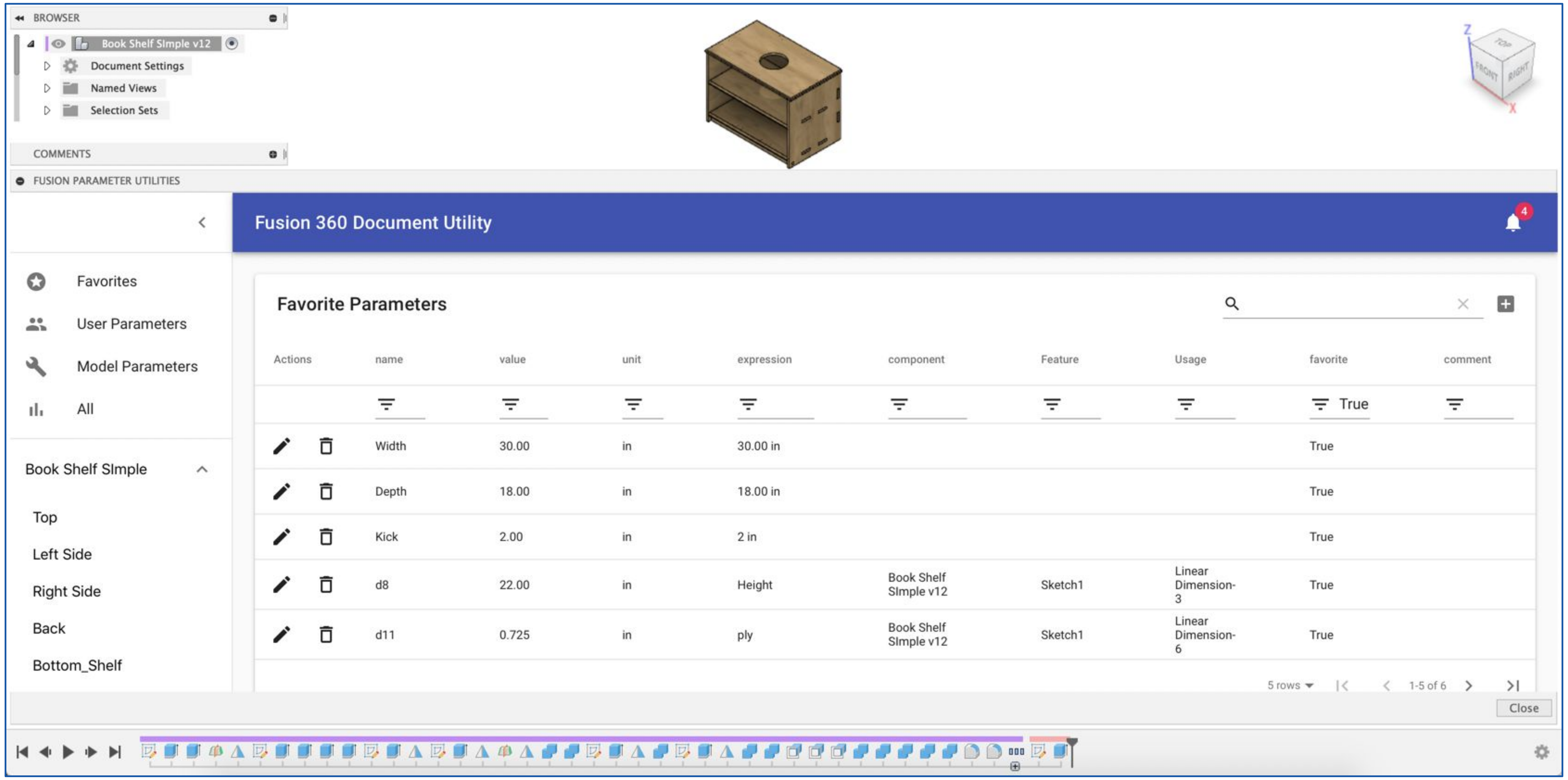
Other Useful Areas of the API



Palettes

Loads an html page in a frame in Fusion 360

Can send and receive information from the page.

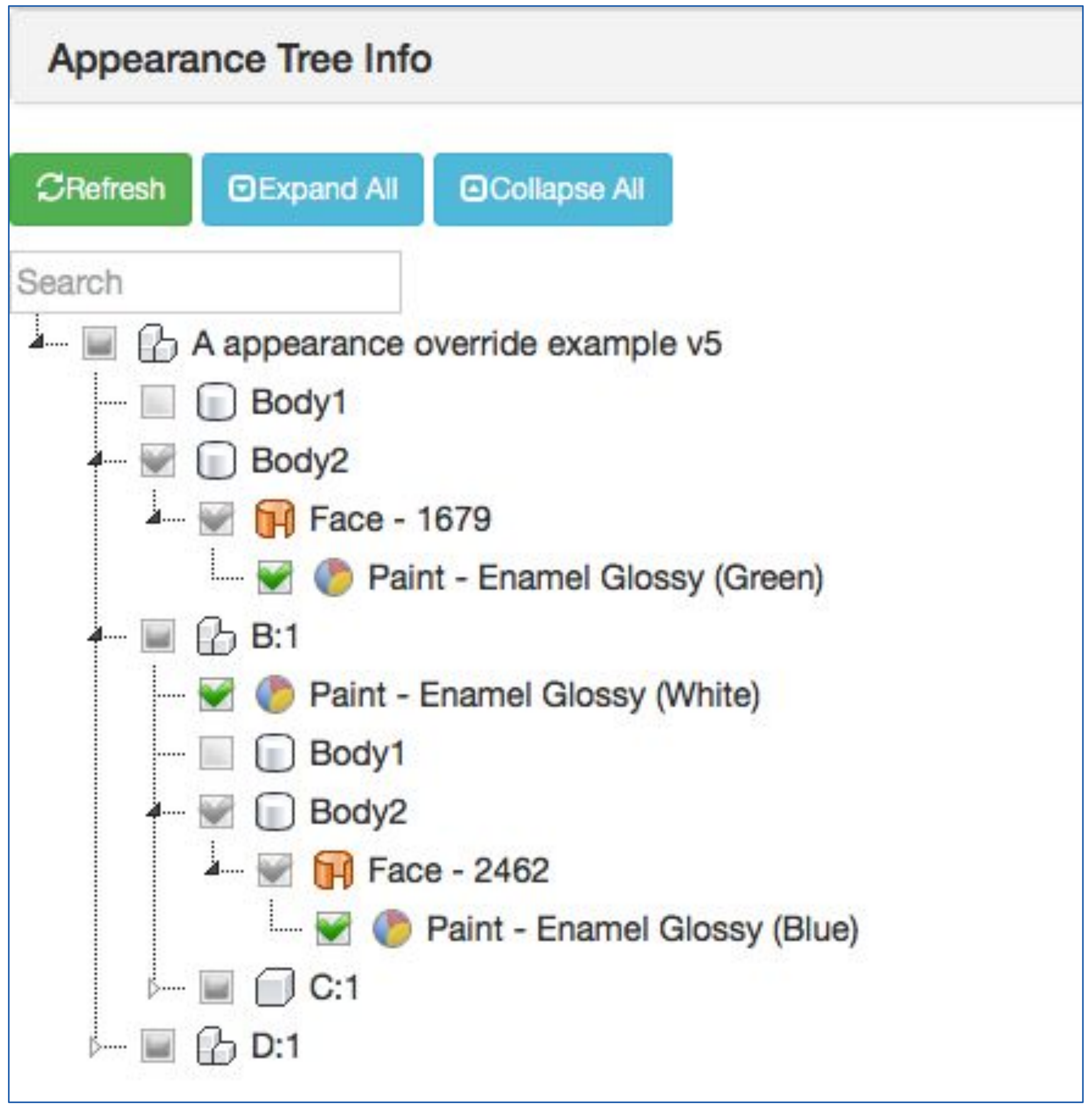


Fusion 360 Document Utility

Favorite Parameters

Actions	name	value	unit	expression	component	Feature	Usage	favorite	comment
								True	
	Width	30.00	in	30.00 in				True	
	Depth	18.00	in	18.00 in				True	
	Kick	2.00	in	2 in				True	
	d8	22.00	in	Height	Book Shelf Simple v12	Sketch1	Linear Dimension-3	True	
	d11	0.725	in	ply	Book Shelf Simple v12	Sketch1	Linear Dimension-6	True	

5 rows 1-5 of 6



- A appearance override example v5
  - Body1
  - Body2
    - Face - 1679
      - Paint - Enamel Glossy (Green)
  - B:1
    - Paint - Enamel Glossy (White)
    - Body1
    - Body2
      - Face - 2462
        - Paint - Enamel Glossy (Blue)
    - C:1
  - D:1

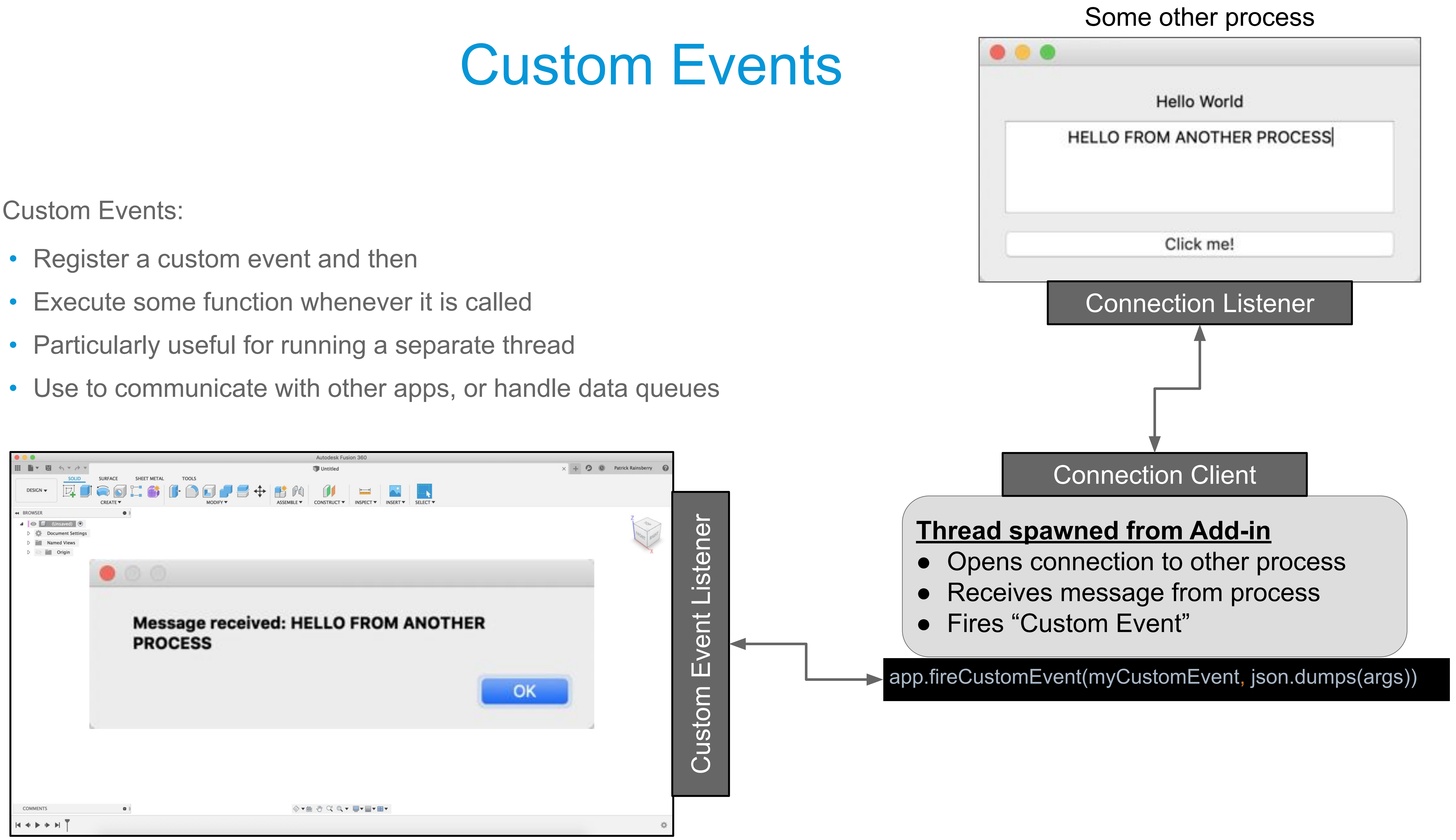
Leverage client side libraries:
- jquery + jstree (above)
- react + material-ui + material-table (left)

Connect Directly to a web server:
- Insert components from a catalog
- Synchronize data

Custom Events

Custom Events:

- Register a custom event and then
- Execute some function whenever it is called
- Particularly useful for running a separate thread
- Use to communicate with other apps, or handle data queues



Hello World

HELLO FROM ANOTHER PROCESS

Click me!

Thread spawned from Add-in

- Opens connection to other process
- Receives message from process
- Fires “Custom Event”

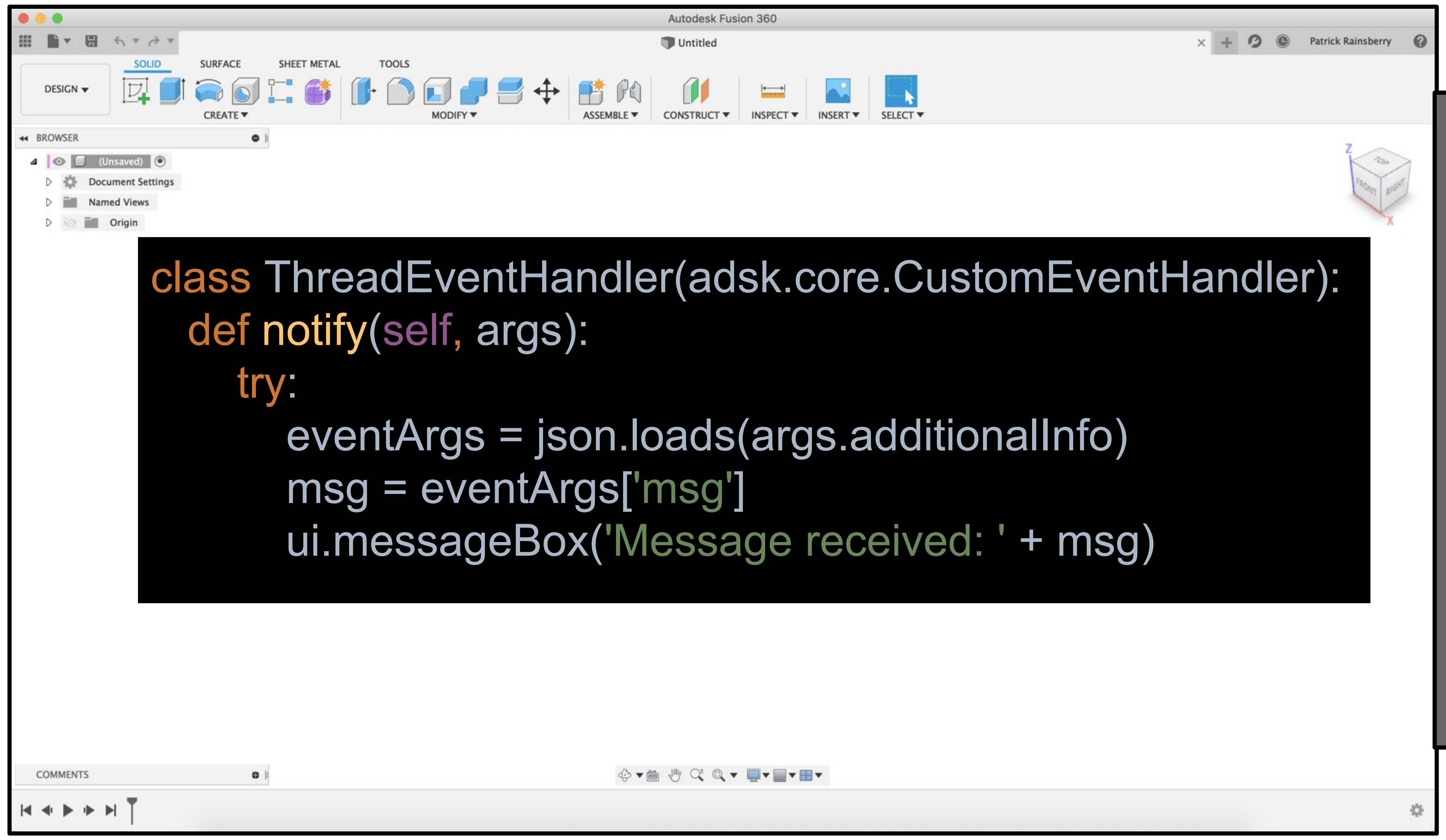
```
app.fireCustomEvent(myCustomEvent, json.dumps(args))
```

Message received: HELLO FROM ANOTHER PROCESS

Custom Events

Custom Events:

- Register a custom event and then
- Execute some function whenever it is called
- Particularly useful for running a separate thread
- Use to communicate with other apps, or handle data queues



```
class ThreadEventHandler(adsk.core.CustomEventHandler):
   def notify(self, args):
      try:
         eventArgs = json.loads(args.additionalInfo)
         msg = eventArgs['msg']
         ui.messageBox('Message received: ' + msg)
```

Custom Event Listener

Some other process

```
from multiprocessing.connection import Listener
address = ('localhost', 6000)
with Listener(address, authkey=b'secret password') as listener:
   with listener.accept() as conn:
      widget.conn = conn


# Elsewhere in application:
self.button.clicked.connect(conn.send([test, False])
```

Connection Listener

Connection Client

```
from multiprocessing.connection import Client
address = ('localhost', 6000)
with Client(address, authkey=b'secret password') as conn:
   while not self.stopped.wait(5):
      msg = conn.recv()[0]
      args = {'msg': msg}
      app.fireCustomEvent(myCustomEvent, json.dumps(args))
```

Attributes

- Attributes can be assigned to nearly any object in a Fusion 360 document (including the document itself)
- Attributes are a string value, but a VERY useful practice is to store a json string as the attribute:
    - *json.dumps(some_python_dictionary)*
- Objects can retrieved by searching for a particular attribute value or group in a document
- Another VERY useful technique is to assign a unique id to any object (geometry, etc.) that you want to maintain a reference to.

`Fusion360Utilities.item_id`*(item, group_name)*

```
items_to_remember = []
for item in object_collection_of_interesting_fusion_objects:
  items_to_remember.append(item_id(item, "MyAppName"))
document_settings = {"items_to_remember": items_to_remember}
document.attributes.add("MyAppName", "settings", json.dumps(document_settings))
```

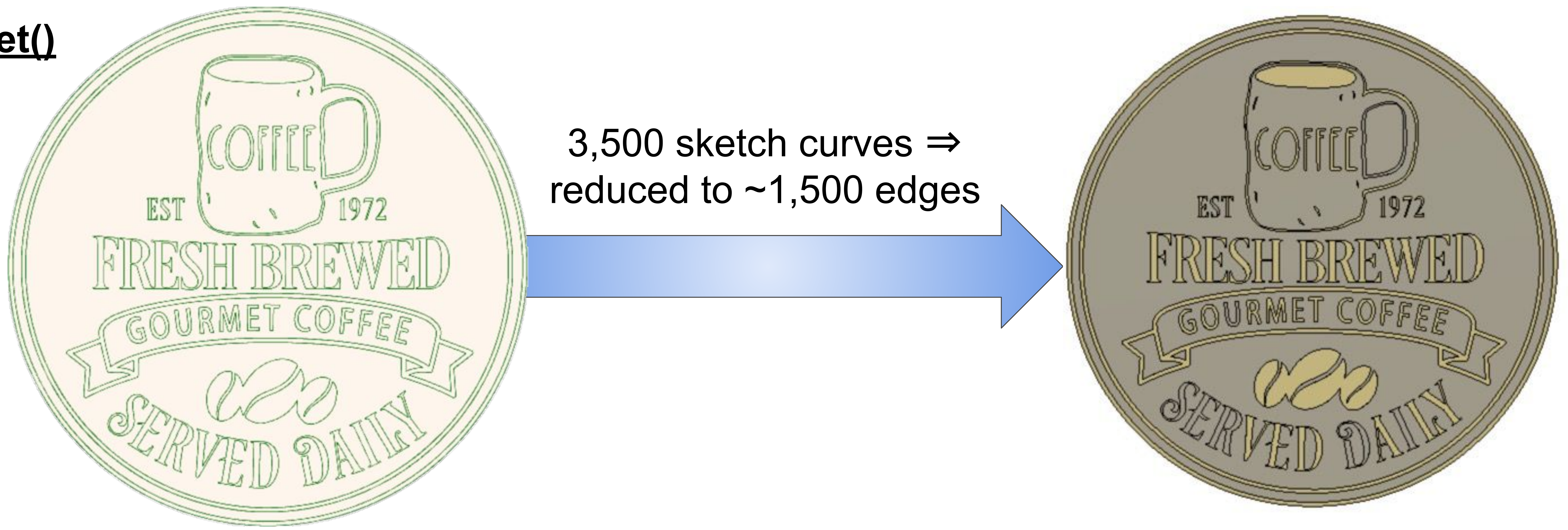
`Fusion360Utilities.get_item_by_id`*(this_item_id, app_name)*

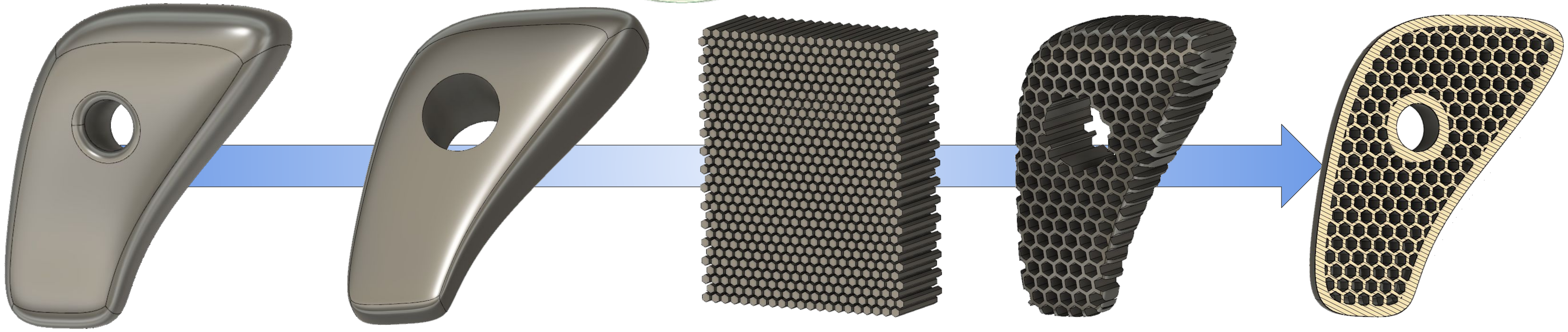
```
settings_attribute = document.attributes.itemByName("MyAppName", "settings")
settings = json.loads(settings_attribute.value)
remembered_items = []
for object_id in settings["items_to_remember"]:
  remembered_items.append(get_item_by_id(object_id, "MyAppName"))
```

Temporary BREP Manager

<u>adsk.fusion.TemporaryBRepManager.get()</u>

- Call functions directly into asm
- Geometry is “transient”
- Finally add geometry to a base feature
- EXTREMELY FAST

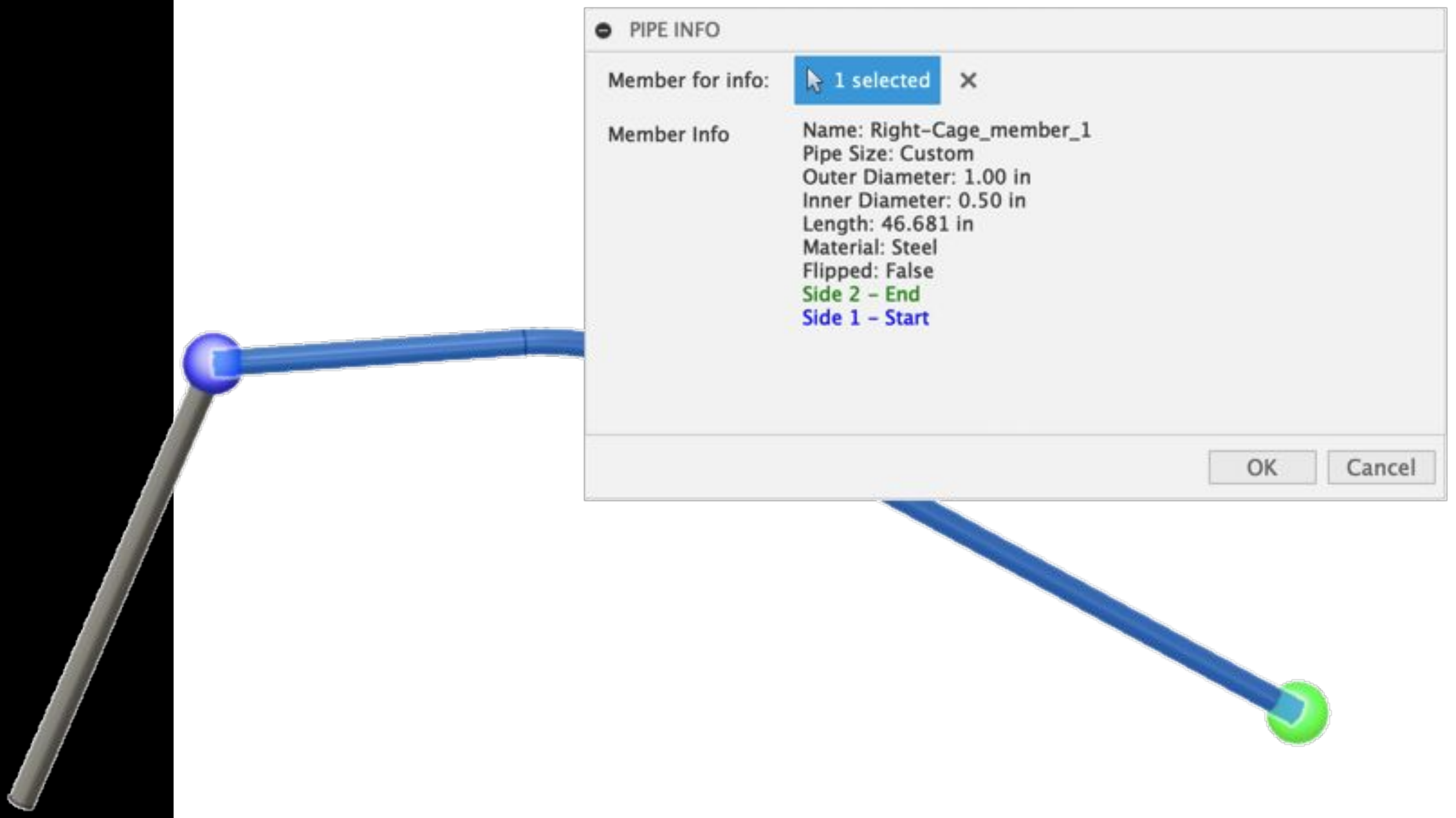




Custom Graphics

Works very well in conjunction with temporaryBrepManager

```
COLORS = {
  "blue": adsk.fusion.CustomGraphicsBasicMaterialColorEffect.create(
     adsk.core.Color.create(10, 10, 245, 255)
  ),
  "green": adsk.fusion.CustomGraphicsBasicMaterialColorEffect.create(
     adsk.core.Color.create(10, 245, 10, 255)
  )}

def create_graphics(center_point, color):
  ao = AppObjects()
  graphics_group = ao.root_comp.customGraphicsGroups.add()

  temp_brep_mgr = adsk.fusion.TemporaryBRepManager.get()
  sphere = temp_brep_mgr.createSphere(center_point, 3.0)
  sphere_graphic = graphics_group.addBRepBody(sphere)

  transform = adsk.core.Matrix3D.create()
  transform.translation = center_point.asVector()

  sphere_graphic.transform = transform
  sphere_graphic.color = COLORS[color]
```

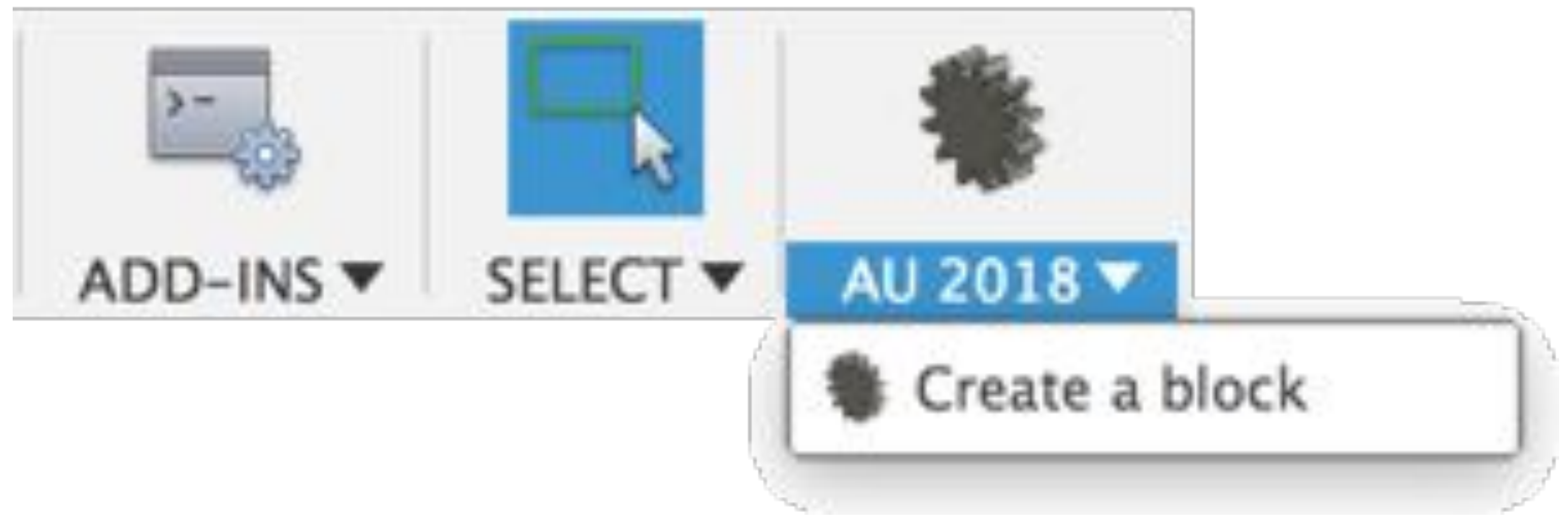


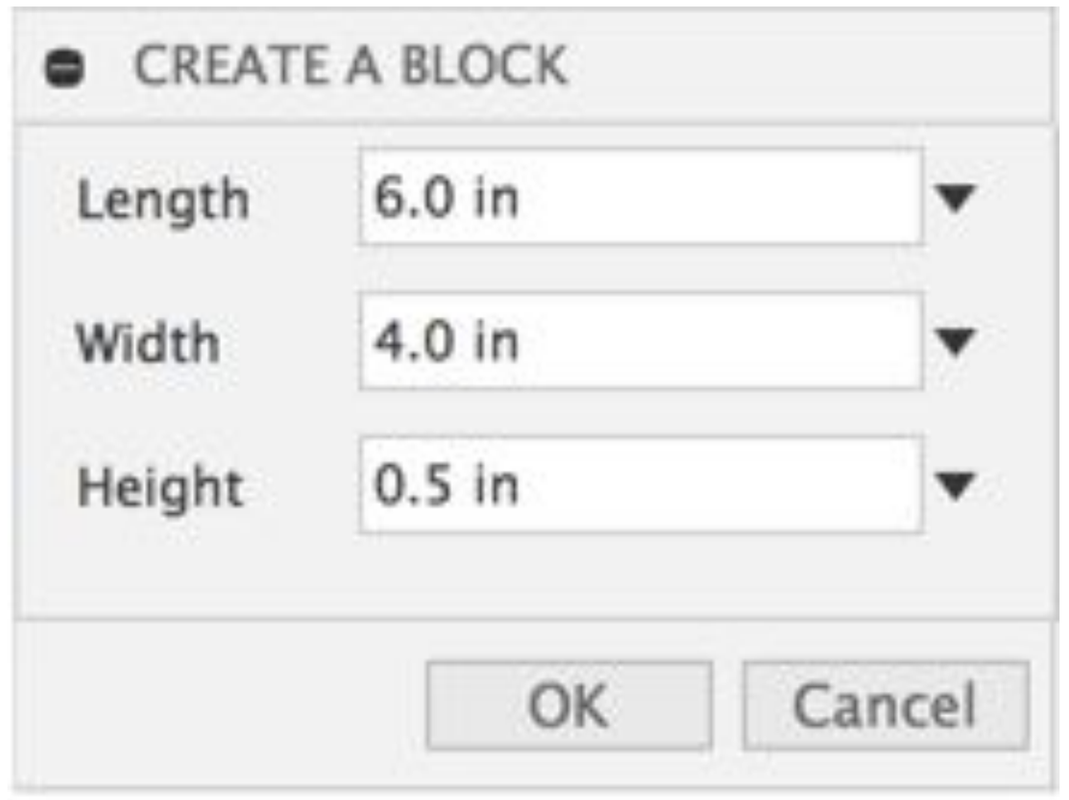
Name: Right-Cage_member_1
Pipe Size: Custom
Outer Diameter: 1.00 in
Inner Diameter: 0.50 in
Length: 46.681 in
Material: Steel
Flipped: False
Side 2 – End
Side 1 – Start

Creating an Addin

Add-ins vs. Scripts

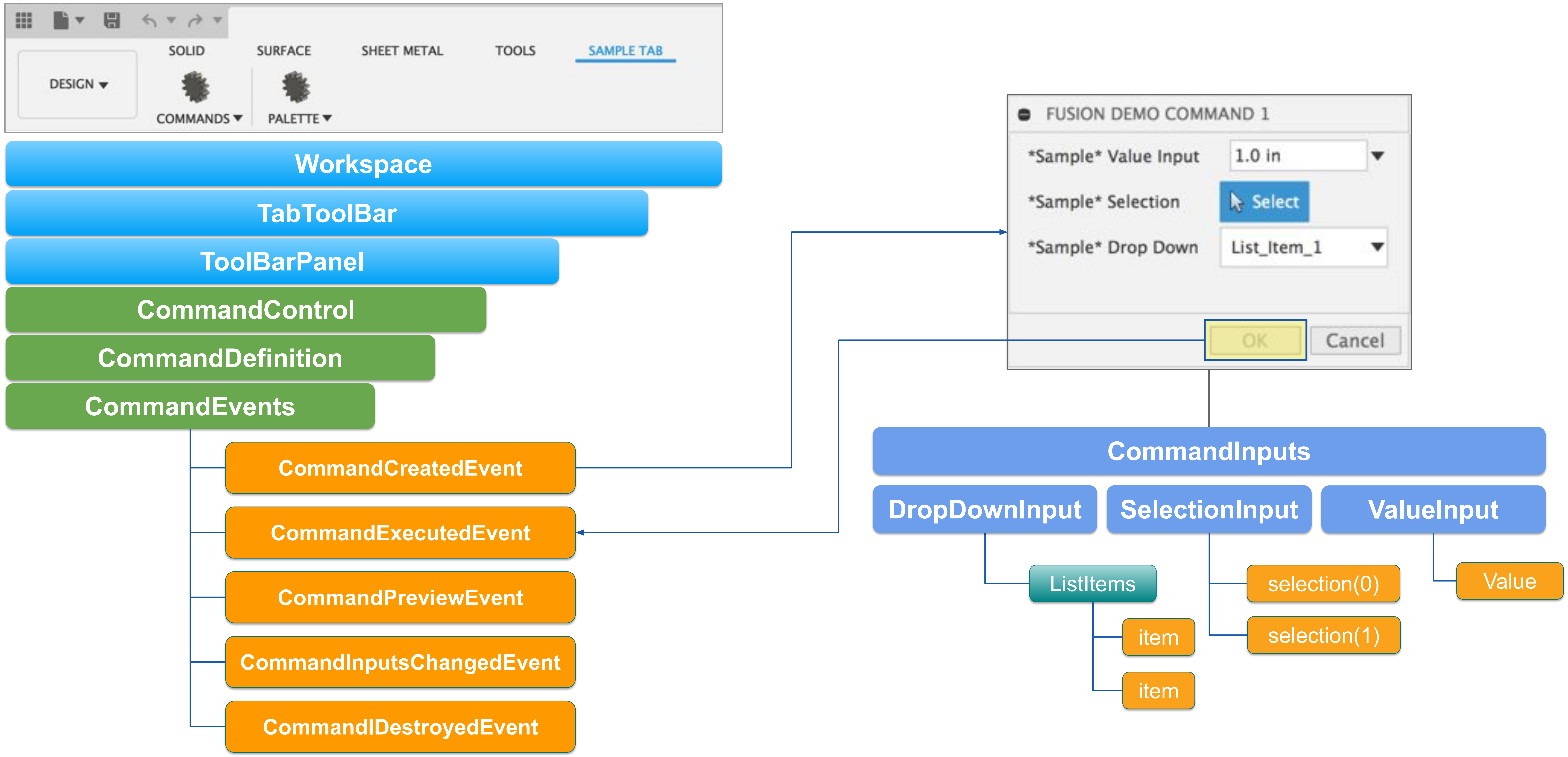
- Add-ins are always running (once started)
- They create a command in the UI (typically)
- When a user clicks the command it reacts:
    - Typically would show a dialog box
    - User inputs values / makes selections
    - Add-in processes values and creates result
- All actions of command result in single “undo” step
    - *They may create many features in the timeline*




CREATE A BLOCK

Length	6.0 in
Width	4.0 in
Height	0.5 in

Commands



Sample Value Input 1.0 in

Sample Selection

Sample Drop Down List_Item_1

Apper

Using Apper

- Apper is a wrapper to create Fusion 360 Add-ins
- The idea is to simplify the creation of add-ins for users
- Note this is a bit of a “pet project” and not really endorsed or maintained by anybody that actually knows what they are doing…

- Two main elements:
  - Add-In Definition: **Fusion360App.py**
  - Commands: **Fusion360CommandBase.py**

https://github.com/tapnair/apper

Using Apper

- Apper is a wrapper to create Fusion 360 Add-ins
- The idea is to simplify the creation of add-ins for users
- Note this is a bit of a “pet project” and not really endorsed or maintained by anybody that actually knows what they are doing…

Use at your own risk

This sample is provided "As-is" with no guarantee of performance, reliability or warranty.

- Two main elements:
  - Add-In Definition: **Fusion360App.py**
  - Commands: **Fusion360CommandBase.py**

https://github.com/tapnair/apper

FusionApp.py

- The **Fusion360App** class wraps the common tasks used when creating a Fusion 360 Command.
- To create a new add-in a new instance of **Fusion360App(*Add_In_Name, Company_Name, Debug*)**
- Add commands with the function: **add_command(*Command_Name, Command_Class, Options*)**
  - The command class should be a subclass on **Fusion360CommandBase** (described on next slide)
  - In the **options** dictionary you define how and where the command will be placed in the user interface

```python
from .apper import apper

my_app = apper.FusionApp('SampleApp', "MyOrganization", False)

my_app.add_command(
  'Sample Command 1',
  SampleCommand1.SampleCommand1,
  {
     'cmd_description': 'Hello World!',
     'cmd_id': 'sample_cmd_1',
     'workspace': 'FusionSolidEnvironment',
     'toolbar_tab_id': 'Sample Tab',
     'toolbar_panel_id': 'Commands',
     'cmd_resources': 'demo_icons',
     'command_visible': True,
     'command_promoted': True,
  })
```

Fusion360CommandBase.py

- The **Fusion360CommandBase** class wraps the common tasks used when creating a Fusion 360 Command.
- To create a new command create a new subclass of **Fusion360CommandBase**
- Then override the methods and add functionality as required
    - **onCreate**: Build your UI components here
    - **onExecute**: Will be executed when user selects OK in command dialog.
    - **onPreview**: Executed when any inputs have changed, will updated the geometry in the graphics window
    - **onInputChanged**: Executed when any inputs have changed. Useful for updating command UI.
    - **onDestroy**: Executed when the command is done. Sometimes useful to check if a user hit cancel

```python
import adsk.core
from ..apper import apper
from ..apper.apper import AppObjects


class SampleCommand1(apper.Fusion360CommandBase):
   def on_execute(self, command: adsk.core.Command, inputs: adsk.core.CommandInputs, args, input_values):
       ao = AppObjects()
       ao.ui.messageBox("Hello World")
```

PaletteCommandBase.py

- Subclass of Fusion360CommandBase
- Defines a command that creates a Palette
- Used to display an existing web page (or simply a local HTML UI)

```
my_app.add_command(
  'Sample Palette Command - Show',
  SamplePaletteCommand.SamplePaletteShowCommand,
  {
     'cmd_description': 'Fusion Demo Palette Description',
     'cmd_id': 'sample_palette_show',
     'workspace': 'FusionSolidEnvironment',
     'toolbar_tab_id': 'Sample Tab',
     'toolbar_panel_id': 'Palette',
     'cmd_resources': 'demo_icons',
     'command_visible': True,
     'command_promoted': True,
     'palette_id': 'sample_palette',
     'palette_name': 'Sample Fusion 360 HTML Palette',
     'palette_html_file_url': 'palette_html/demo.html',
     'palette_is_visible': True,
     'palette_show_close_button': True,
     'palette_is_resizable': True,
     'palette_width': 500,
     'palette_height': 600,
  })
```

On Create

When the user clicks the command icon in the Fusion ui (command control) this function will be executed

By referencing the *inputs* object you can easily add dialog box elements to your command

Sometimes you may want to read some data or analyze the model BEFORE creating the dialog box

```python
def on_create(self, command: adsk.core.Command, inputs: adsk.core.CommandInputs):

    # Create a default value using a string
    ao = AppObjects()
    default_value = adsk.core.ValueInput.createByString('1.0 in')
    default_units = ao.units_manager.defaultLengthUnits
    inputs.addValueInput('value_input_id', '*Sample* Value Input', default_units, default_value)

    # Other Input types
    inputs.addBoolValueInput('bool_input_id', '*Sample* Check Box', True)
    inputs.addStringValueInput('string_input_id', '*Sample* String Value', 'Some Default Value')
    inputs.addSelectionInput('selection_input_id', '*Sample* Selection', 'Select Something')

    # Read Only Text Box
    inputs.addTextBoxCommandInput('text_box_input_id', 'Selection Type: ', 'Nothing Selected', 1, True)

    # Create a Drop Down
    drop_down_input = inputs.addDropDownCommandInput('drop_down_input_id', '*Sample* Drop Down',
                                                     adsk.core.DropDownStyles.TextListDropDownStyle)
    drop_down_items = drop_down_input.listItems
    drop_down_items.add('List_Item_1', True, '')
    drop_down_items.add('List_Item_2', False, '')
```

On Input Changed

When a user changes anything in the command dialog this method is executed.

Typically used for making changes to the command dialog itself.

For example if a user selects STL as an export type, you can then display an option to show a refinement option

```
def on_input_changed(self, command: adsk.core.Command, inputs: adsk.core.CommandInputs, changed_input, input_values):

    # Selections are returned as a list so lets get the first one
    all_selections = input_values.get('selection_input_id', None)

    if all_selections is not None:
        the_first_selection = all_selections[0]

        # Update the text of the string value input to show the type of object selected
        text_box_input = inputs.itemById('text_box_input_id')
        text_box_input.text = the_first_selection.objectType
```

On Preview / On Destroy

On preview will also execute on any changes to the command inputs

- Code in this function will cause the graphics to refresh.
- Note if your addin is complex it may be useful to only preview a subset of the full operations

On Destroy executes after the command has run

- You can use this to do any clean up that may otherwise be difficult until after the command has completed
- Like firing a second command for example

```
# Run whenever a user makes any change to a value or selection in the addin UI
# Commands in here will be run through the Fusion processor and changes will be reflected in  Fusion graphics area
def on_preview(self, command: adsk.core.Command, inputs: adsk.core.CommandInputs, args, input_values):
  pass

# Run after the command is finished.
# Can be used to launch another command automatically or do other clean up.
def on_destroy(self, command: adsk.core.Command, inputs: adsk.core.CommandInputs, reason, input_values):
  pass
```

On Execute

-

```
def on_execute(self, command: adsk.core.Command, inputs: adsk.core.CommandInputs, args, input_values):

    # Get the values from the user input
    the_value = input_values['value_input_id']
    the_boolean = input_values['bool_input_id']
    the_string = input_values['string_input_id']
    all_selections = input_values['selection_input_id']
    the_drop_down = input_values['drop_down_input_id']

    # Selections are returned as a list so lets get the first one and its name
    the_first_selection = all_selections[0]
    the_selection_type = the_first_selection.objectType

    # Get a reference to all relevant application objects in a dictionary
    ao = AppObjects()

    converted_value = ao.units_manager.formatInternalValue(the_value, 'in', True)

    ao.ui.messageBox('The value, in internal units, you entered was:  {} \n'.format(the_value) +
                     'The value, in inches, you entered was:  {} \n'.format(converted_value) +
                     'The boolean value checked was:  {} \n'.format(the_boolean) +
                     'The string you typed was:  {} \n'.format(the_string) +
                     'The type of the first object you selected is:  {} \n'.format(the_selection_type) +
                     'The drop down item you selected is:  {}'.format(the_drop_down)
                     )
```

Extra Capabilities: input_values

In the on_execute, on_preview, on_input_changed methods there is a parameter called "input_values"

This parameter is a dictionary containing the relevant values for all of the user inputs.

- The key is the name of the input.
- The value is dependant on the type input:
  - Value type inputs will have their actual value stored (string or number depending)
  - List type inputs (drop downs, etc) will have the name of the selected item as the value (string)
  - Selection inputs are returned as an array of the selected objects (even if 1 item is selected)

```
# Get the values from the user input
the_value = input_values['value_input_id']
the_boolean = input_values['bool_input_id']
the_string = input_values['string_input_id']
all_selections = input_values['selection_input_id']
the_drop_down = input_values['drop_down_input_id']
```

Note: you can still access the raw command inputs object with the "inputs" variable. This would behave similar to any of the examples in the API documentation. i.e. length = inputs.itemById('length').value

Extra Capabilities: AppObjects

This is a helper class that can be used to easily access of many useful fusion 360 objects.

It contains many properties:

- app - Application Object
- document - Active Document
- product - Active Product
- design - Design Product (if it exists)
- cam - CAM Product (if it exists)
- ui - User Interface
- import_manager - Application Import Manager
- export_manager - Export Manager (if the active product is Design)
- units_manager - Fusion Units Manager (if the active product is design) or Units Manager
- root_comp - Root Component (if the active product is design)
- time_line - (if the active product is design and the type os Parametric Design Type)

```
from ..apper.apper import AppObjects
ao = AppObjects()
ao.ui.messageBox("Hello Patrick Rainsberry")
```

Fusion 360 Custom Thread

```
my_addin.add_custom_event("sample_message_system", SampleCustomEvent1)
```

Allows you run a separate thread and fire events to Fusion 360

Communicate with other applications or services asynchronously

custom_event_received:

Override this function to react to the firing of:

```
ao.app.fireCustomEvent(some_json_string)
```

run_in_thread:

Override this function.

Will be executed in a separate thread.

Communicate to Fusion 360 window with:

```
ao.app.fireCustomEvent(some_json_string)
```

```
import json
import time

from ..apper import apper


class SampleCustomEvent1(apper.Fusion360CustomThread):

  def custom_event_received(self, event_dict):
     ao = apper.AppObjects()
     ao.ui.messageBox(str(event_dict))

  def run_in_thread(self, thread, event_id):
     ao = apper.AppObjects()

     # Every five seconds fire a custom event, passing a random number.
     for i in range(3):
        message = {
           "text": "Hello World!",
           "index": str(i),
           "event_id": event_id
        }
        time.sleep(3)
        ao.app.fireCustomEvent(event_id, json.dumps(message))
```

Fusion 360 Document Event

Wrapper for creating and reacting to document events

Specify the specific event to react to from:

Application.documentActivated,

Application.documentActivating,

Application.documentClosed,

Application.documentClosing,

Application.documentCreated,

Application.documentDeactivated,

Application.documentDeactivating,

Application.documentOpened,

Application.documentOpening,

Application.documentSaved,

Application.documentSaving

```
my_addin.add_document_event(
    "sample_open_event",
    app.documentActivated,
    SampleDocumentEvent1
)
```

```
class SampleDocumentEvent1(apper.Fusion360DocumentEvent):

  def document_event_received(self, event_args, document):
    app = adsk.core.Application.cast(adsk.core.Application.get())
    msg = "You just ACTIVATED a document called: {} ".format(document.name)
    app.userInterface.messageBox(msg)
```

Fusion 360 Workspace Event

Wrapper for creating and reacting to document events

Specify the specific event to react to from:

UserInterface.workspaceActivated,

UserInterface.workspaceDeactivated,

UserInterface.workspacePreActivate,

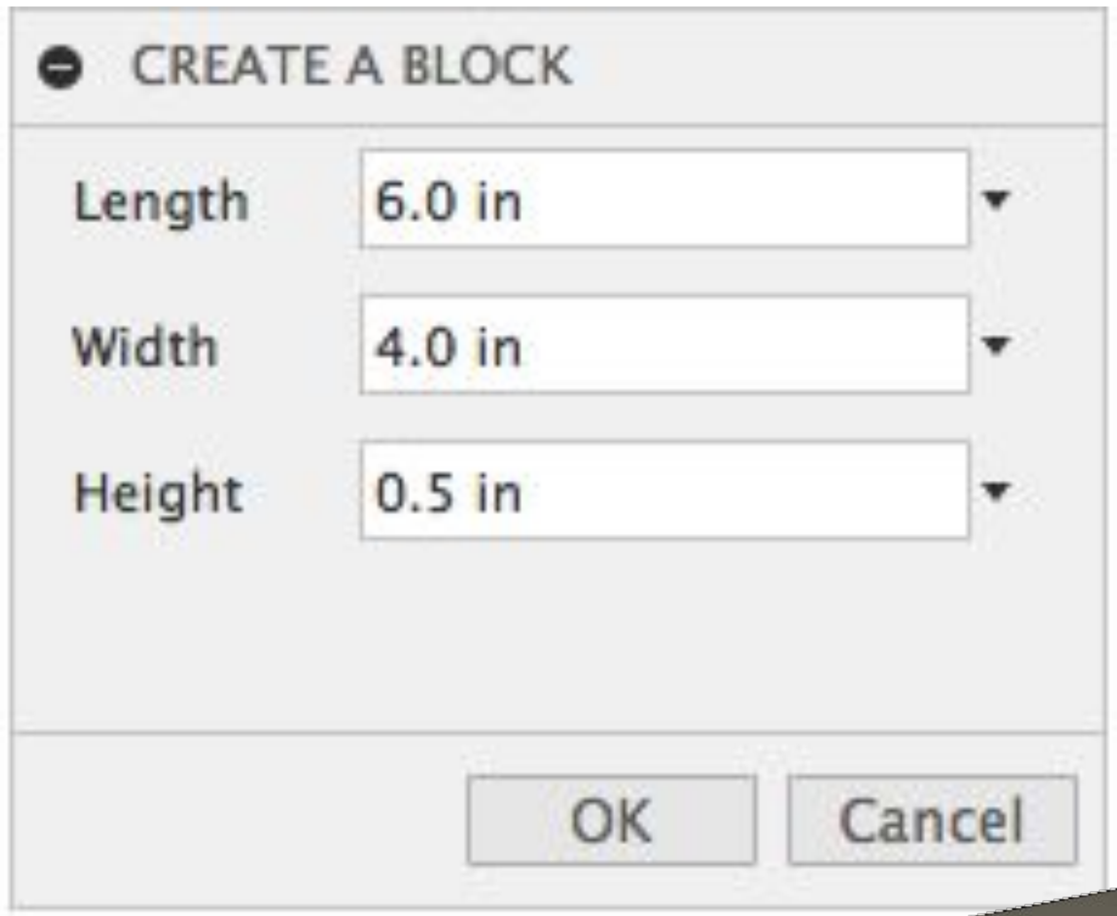
UserInterface.workspacePreDeactivate

```
my_addin.add_workspace_event
    ("sample_workspace_event",
    ui.workspaceActivated,
    SampleWorkspaceEvent1
)
```

```
class SampleWorkspaceEvent1(apper.Fusion360WorkspaceEvent):

  def workspace_event_received(self, event_args, workspace):
     app = adsk.core.Application.cast(adsk.core.Application.get())
     msg = "You just ACTIVATED a workspace called: {} ".format(workspace.name)
     app.userInterface.messageBox(msg)
```

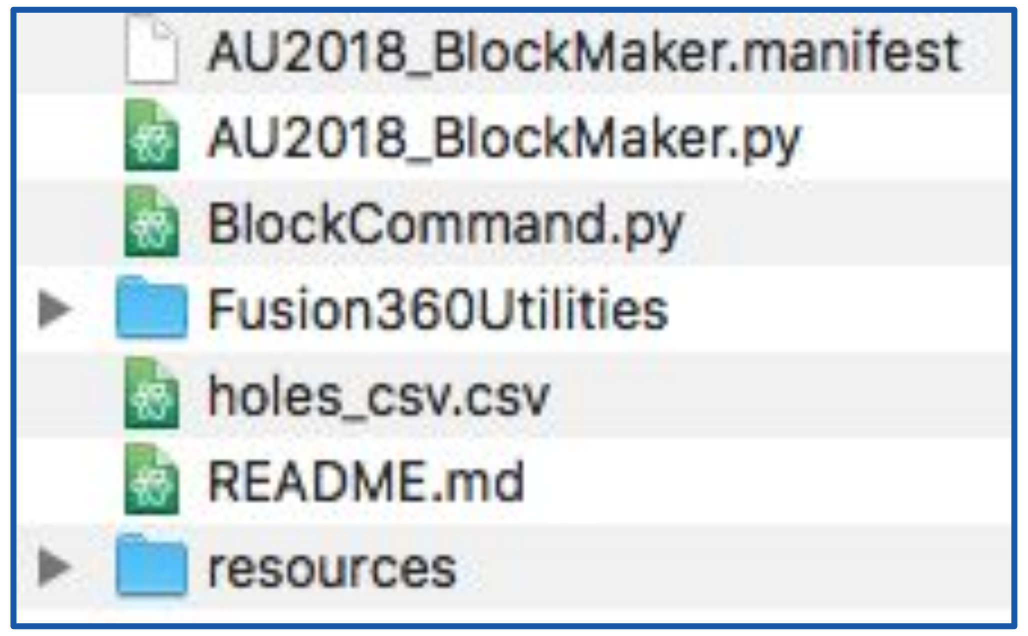
Refactoring the Block

- Follow previous steps to create a new add-in
- Take block code and move into a new function in BlockCommand.py
- Create UI elements to capture user input

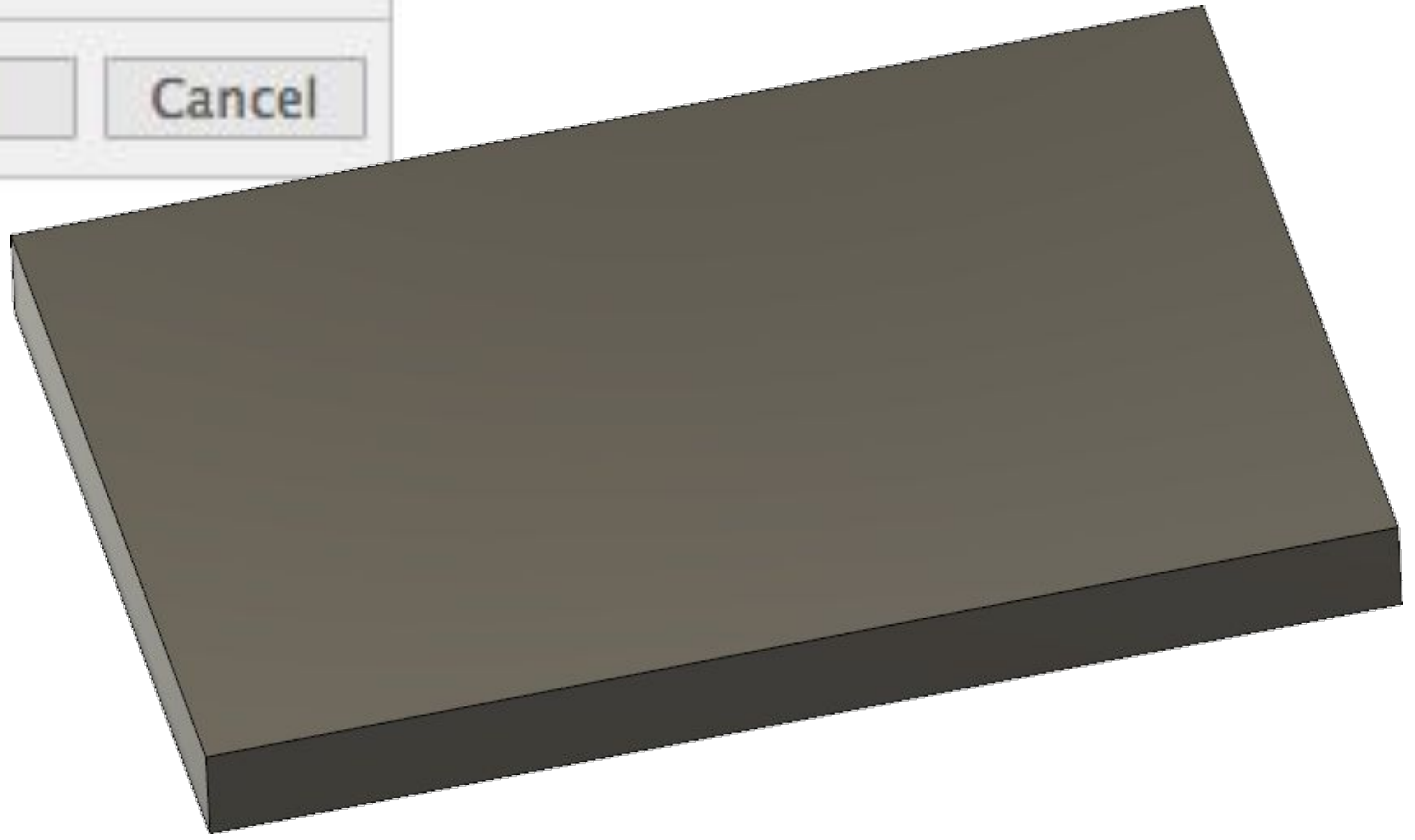


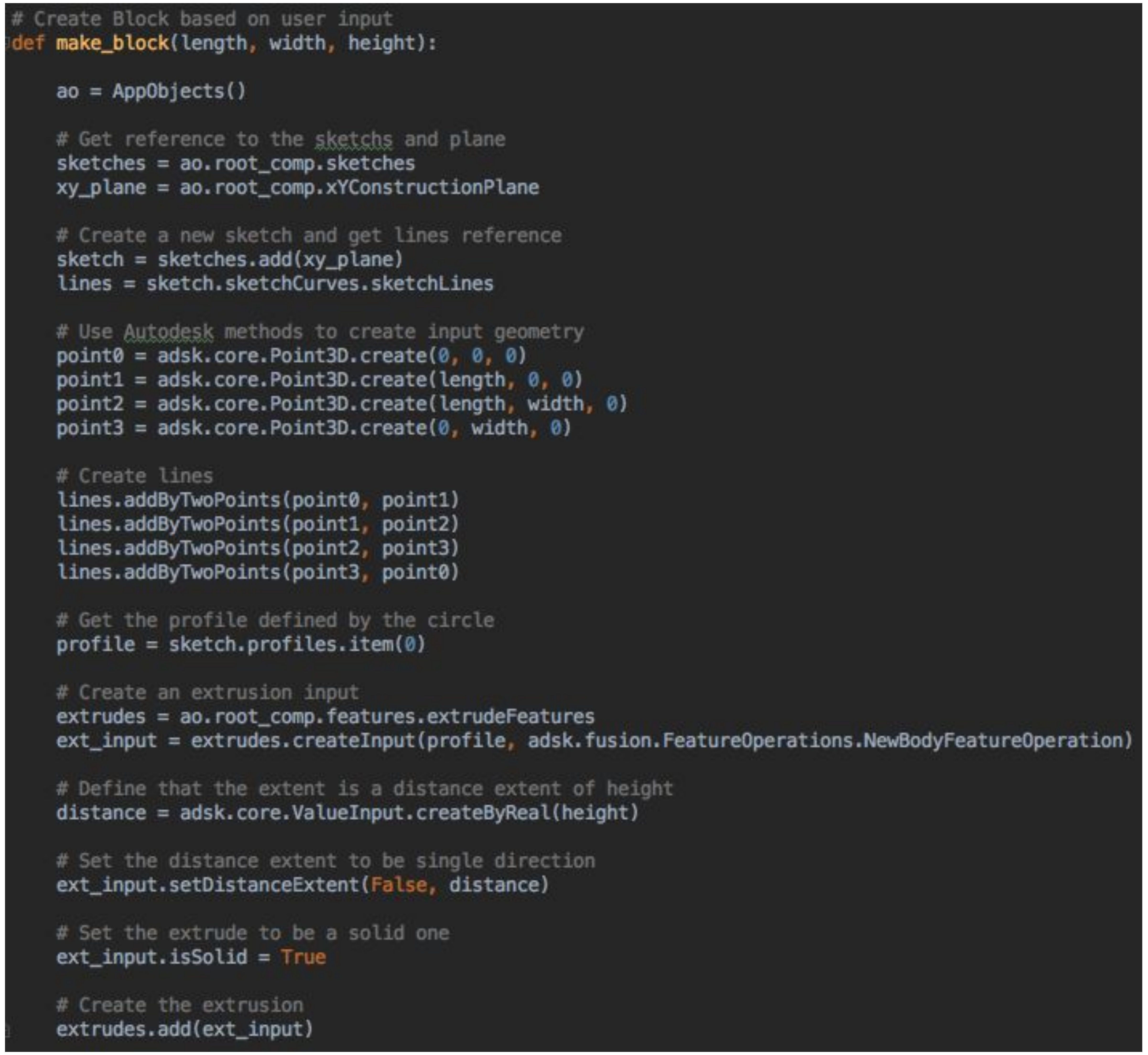
CREATE A BLOCK

Length	6.0 in
Width	4.0 in
Height	0.5 in



```
AU2018_BlockMaker.manifest
AU2018_BlockMaker.py
BlockCommand.py
▶ Fusion360Utilities
holes_csv.csv
README.md
▶ resources
```

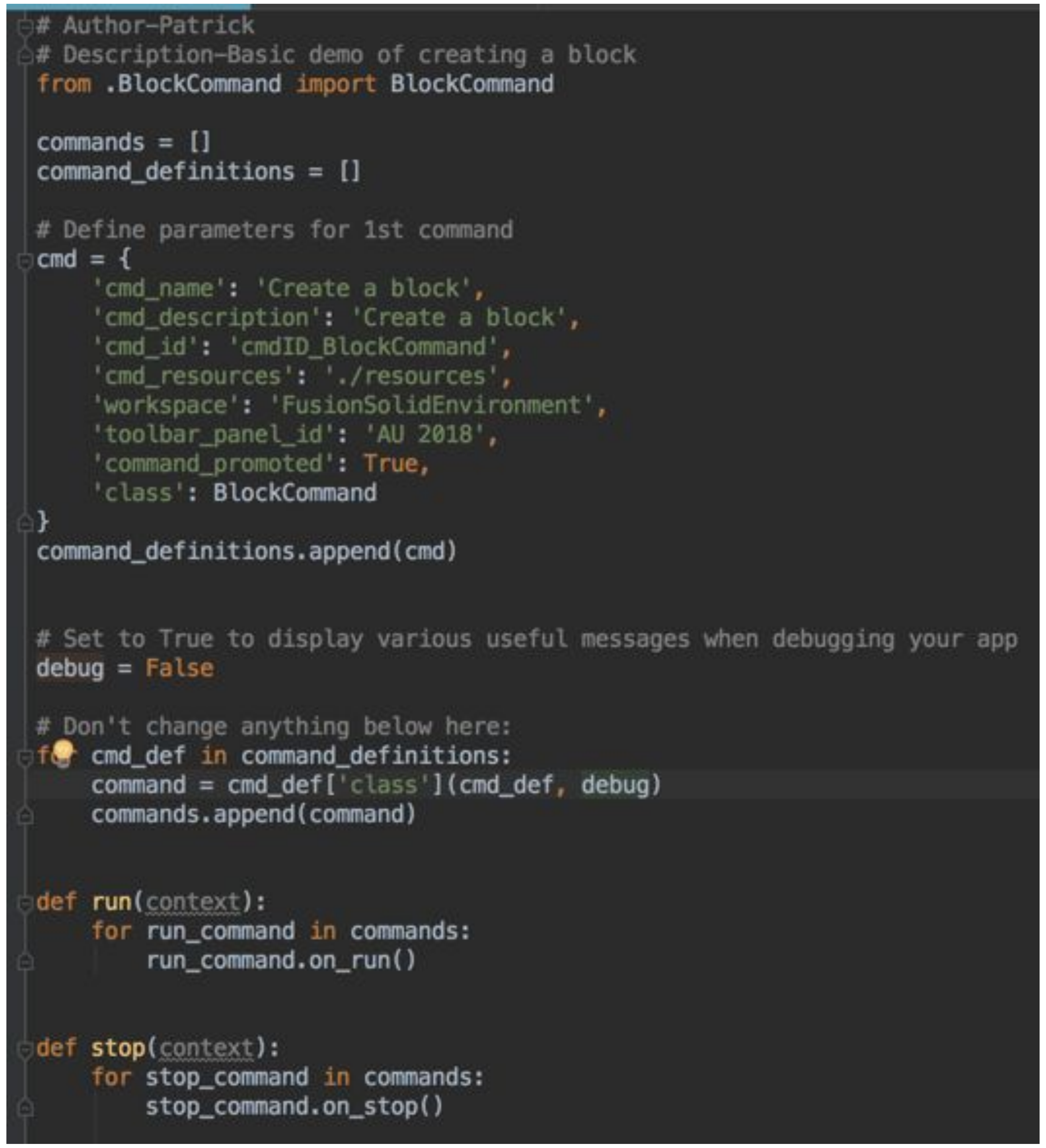




```python
# Create Block based on user input
def make_block(length, width, height):

    ao = AppObjects()

    # Get reference to the sketchs and plane
    sketches = ao.root_comp.sketches
    xy_plane = ao.root_comp.xYConstructionPlane

    # Create a new sketch and get lines reference
    sketch = sketches.add(xy_plane)
    lines = sketch.sketchCurves.sketchLines

    # Use Autodesk methods to create input geometry
    point0 = adsk.core.Point3D.create(0, 0, 0)
    point1 = adsk.core.Point3D.create(length, 0, 0)
    point2 = adsk.core.Point3D.create(length, width, 0)
    point3 = adsk.core.Point3D.create(0, width, 0)

    # Create lines
    lines.addByTwoPoints(point0, point1)
    lines.addByTwoPoints(point1, point2)
    lines.addByTwoPoints(point2, point3)
    lines.addByTwoPoints(point3, point0)

    # Get the profile defined by the circle
    profile = sketch.profiles.item(0)

    # Create an extrusion input
    extrudes = ao.root_comp.features.extrudeFeatures
    ext_input = extrudes.createInput(profile, adsk.fusion.FeatureOperations.NewBodyFeatureOperation)

    # Define that the extent is a distance extent of height
    distance = adsk.core.ValueInput.createByReal(height)

    # Set the distance extent to be single direction
    ext_input.setDistanceExtent(False, distance)

    # Set the extrude to be a solid one
    ext_input.isSolid = True

    # Create the extrusion
    extrudes.add(ext_input)
```

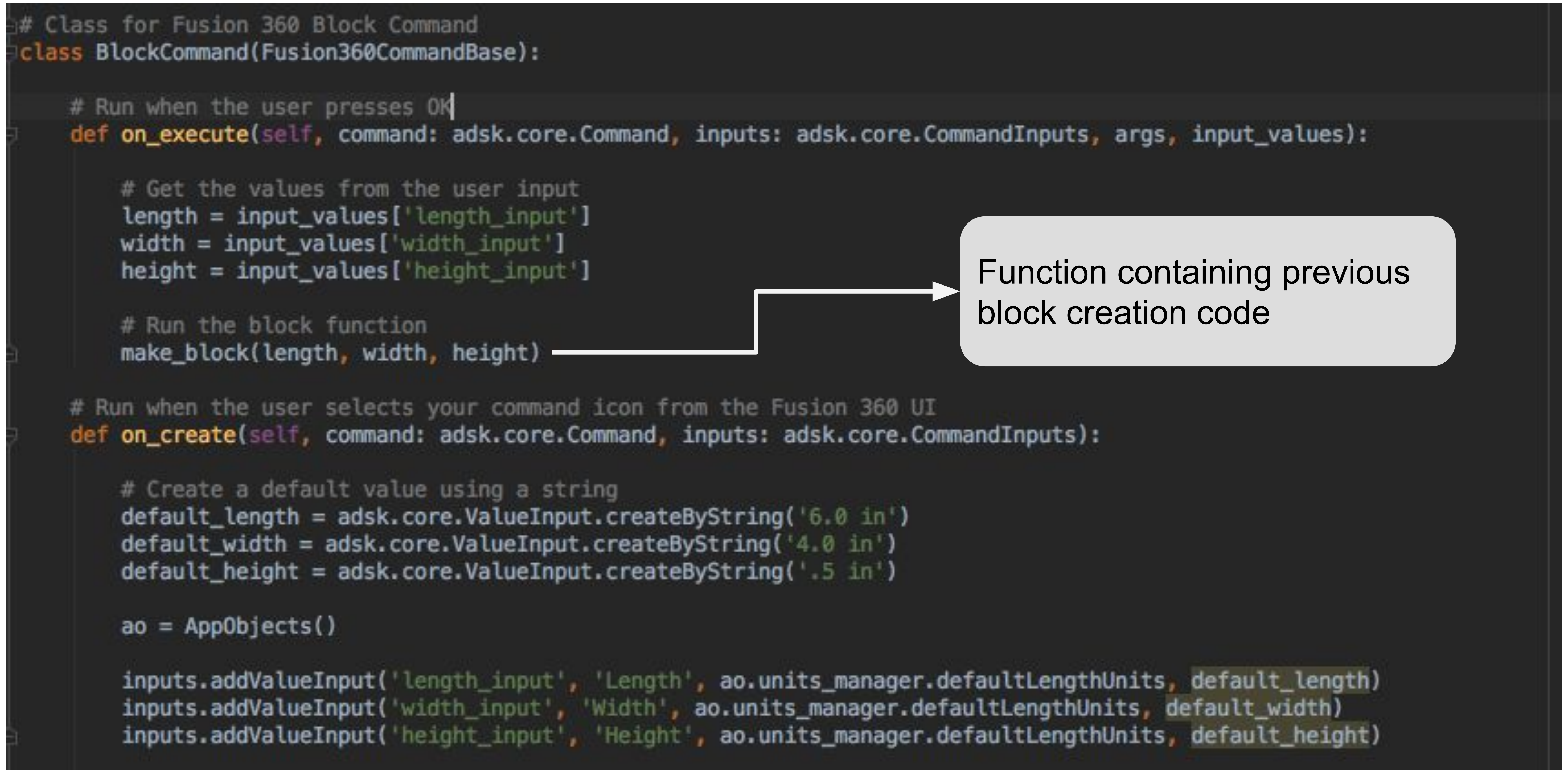
Refactoring the Block

AU2018_BlockMaker.py



```
# Author-Patrick
# Description-Basic demo of creating a block
from .BlockCommand import BlockCommand

commands = []
command_definitions = []

# Define parameters for 1st command
cmd = {
    'cmd_name': 'Create a block',
    'cmd_description': 'Create a block',
    'cmd_id': 'cmdID_BlockCommand',
    'cmd_resources': './resources',
    'workspace': 'FusionSolidEnvironment',
    'toolbar_panel_id': 'AU 2018',
    'command_promoted': True,
    'class': BlockCommand
}
command_definitions.append(cmd)


# Set to True to display various useful messages when debugging your app
debug = False

# Don't change anything below here:
f   cmd_def in command_definitions:
    command = cmd_def['class'](cmd_def, debug)
    commands.append(command)


def run(context):
    for run_command in commands:
        run_command.on_run()


def stop(context):
    for stop_command in commands:
        stop_command.on_stop()
```

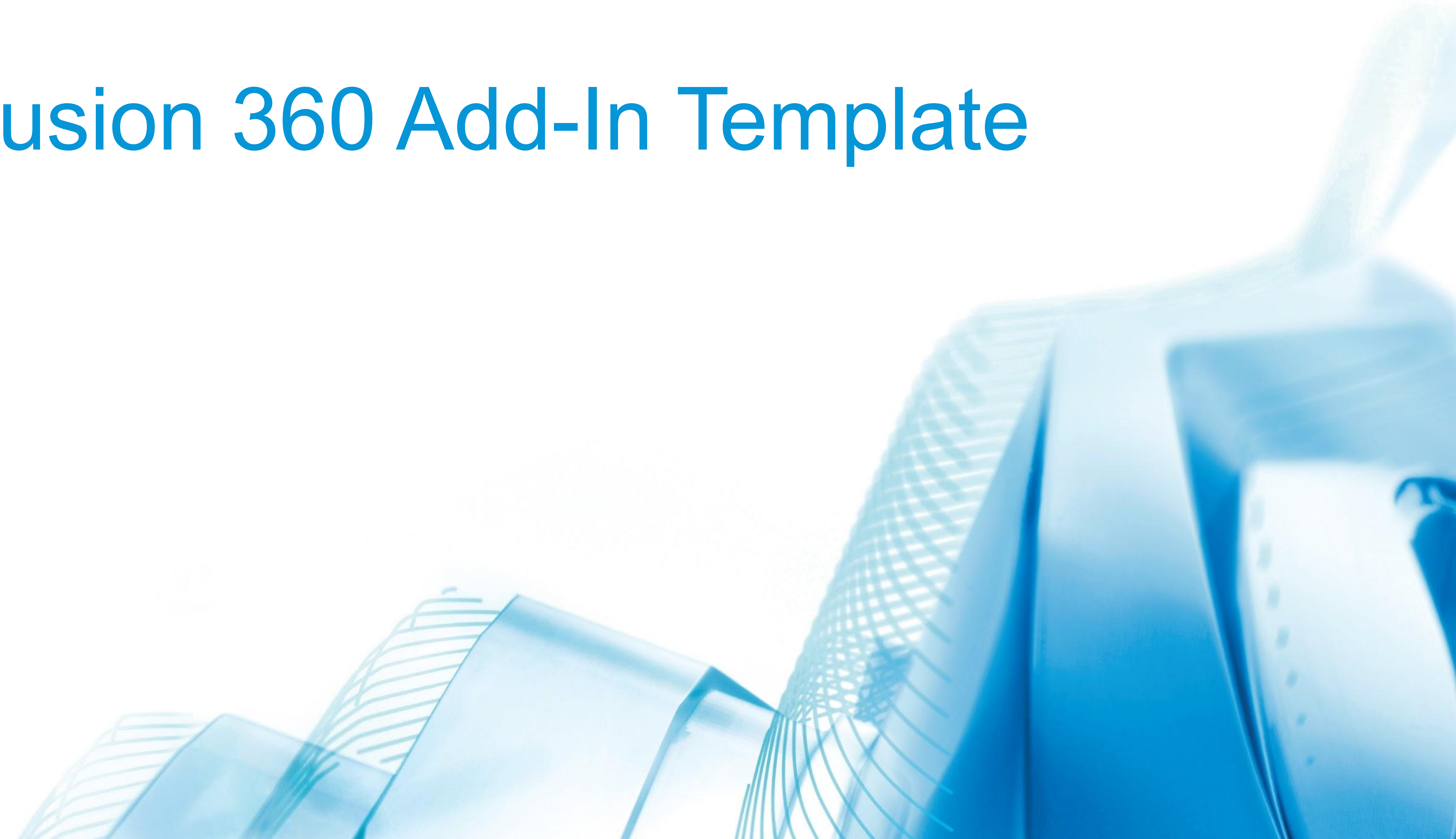
BlockMakerCommand.py



```
# Class for Fusion 360 Block Command
class BlockCommand(Fusion360CommandBase):

    # Run when the user presses OK
    def on_execute(self, command: adsk.core.Command, inputs: adsk.core.CommandInputs, args, input_values):

        # Get the values from the user input
        length = input_values['length_input']
        width = input_values['width_input']
        height = input_values['height_input']

        # Run the block function
        make_block(length, width, height)

    # Run when the user selects your command icon from the Fusion 360 UI
    def on_create(self, command: adsk.core.Command, inputs: adsk.core.CommandInputs):

        # Create a default value using a string
        default_length = adsk.core.ValueInput.createByString('6.0 in')
        default_width = adsk.core.ValueInput.createByString('4.0 in')
        default_height = adsk.core.ValueInput.createByString('.5 in')

        ao = AppObjects()

        inputs.addValueInput('length_input', 'Length', ao.units_manager.defaultLengthUnits, default_length)
        inputs.addValueInput('width_input', 'Width', ao.units_manager.defaultLengthUnits, default_width)
        inputs.addValueInput('height_input', 'Height', ao.units_manager.defaultLengthUnits, default_height)
```

Function containing previous block creation code

Fusion 360 Add-In Template



Fusion 360 Add-In Sample

Commands:

- **Sample Command 1:** Simple Hello World example
- **Sample Command 2:** Create a very basic UI and then accesses the input parameters.
- **Sample Palette Command - Show:** Shows a Palette that loads a simple local html file
- **Sample Palette Command - Send:** Sends a message to the Palette and updates the html displayed

Events:

- **SampleCustomEvent1:** Sends a message to the application window in 5 sec intervals (3 times)
- **SampleDocumentEvent1:** Executes when a document is activated
- **SampleDocumentEvent2:** Executes when a document is closed
- **SampleWorkspaceEvent1:** Executes when a workspace is activated

Cookiecutter

The easiest way to get started with apper is to start from a template project.

cookiecutter creates projects from project templates and is an amazing resource

For more detailed installation instructions see their documentation

Learn more here:

https://apper.readthedocs.io/en/latest/usage/setup.html

```
>>> pip3 install cookiecutter
>>> cd ~
>>> cd /Library/Application Support/Autodesk/Autodesk Fusion 360/API/AddIns/
>>> cookiecutter https://github.com/tapnair/cookiecutter-fusion360-addin.git
```

Developer Resources

Useful Information and troubleshooting an add-in

The best place to get help is the Fusion 360 forum. Otherwise I find an infinite resource in places like stack exchange. Most of the challenges I come across are really python questions more than anything.

Useful Links:

Forum to ask questions:
https://forums.autodesk.com/t5/api-and-scripts/bd-p/22

Offline API DOcumentation (chm):
https://ekinssolutions.com/sdm_downloads/fusion-360-api-chm-file/

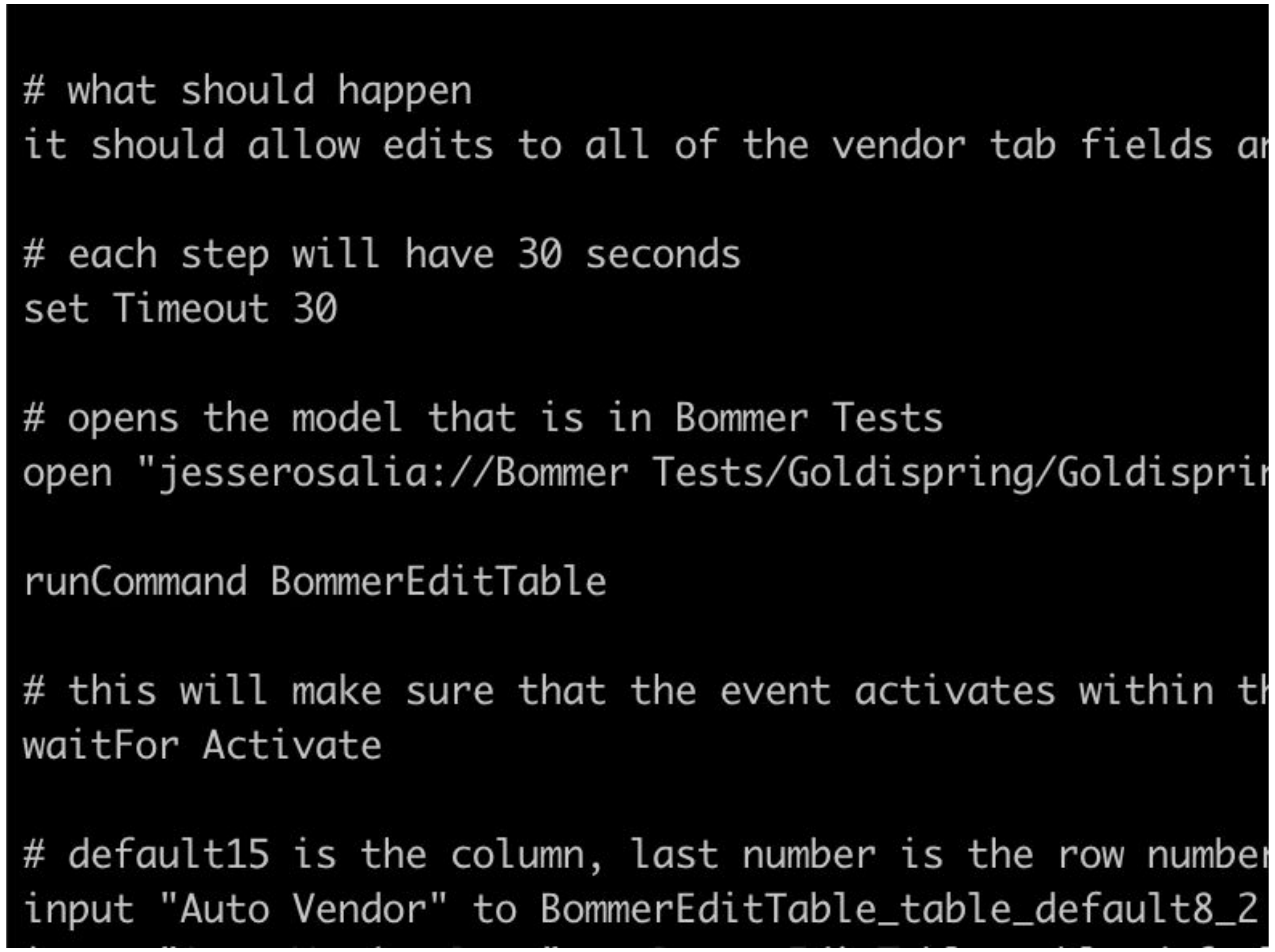
For more detailed information about editing and debugging your scripts and add-ins see the language specific topics (Python or C++) because the process is different depending on which programming language you're using:
Python Specific Issues
C++ Specific Issues

Samples:

My main page for these projects: https://tapnair.github.io/index.html

Sodium

- Test automation framework
- Script UI and integration tests for your Fusion 360 add-in.
- Tests are defined via a simple, declarative, language
- Current capabilities:
  - Run commands
  - Find data
  - Get/set attributes
  - Evaluate expressions

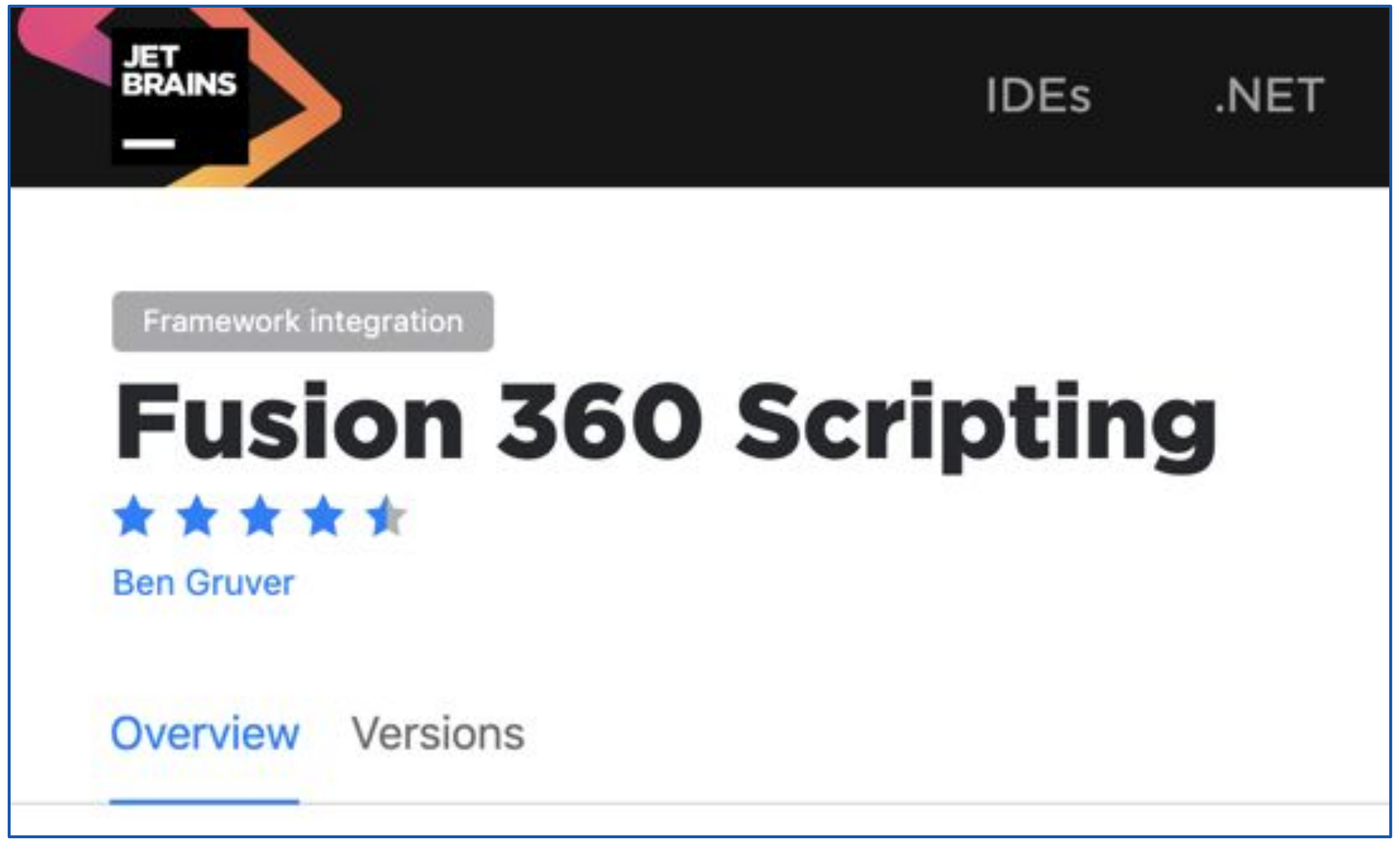


```
# what should happen
it should allow edits to all of the vendor tab fields ar

# each step will have 30 seconds
set Timeout 30

# opens the model that is in Bommer Tests
open "jesserosalia://Bommer Tests/Goldispring/Goldisprir

runCommand BommerEditTable

# this will make sure that the event activates within th
waitFor Activate

# default15 is the column, last number is the row numbe
input "Auto Vendor" to BommerEditTable_table_default8_2
```

Sodium is currently in closed alpha testing
If you're interested contact it's creator at jesse@bommer.io

PyCharm Plugin

- Run script in Fusion 360
  - Launches a script in Fusion 360
  - As if you had run it from the AddIn window
- Debug script in Fusion 360
  - Launches a script in fusion 360 and attached a debugger
  - Stop at breakpoints
  - All the usual debuggery goodness.
  - Redirects stdout and stderr to the debugging console
- Attach to Process
  - Attaches to a Fusion 360 process without running a script.
  - Any breakpoints will be hit
    - *Assuming Fusion happens to run the break pointed code.*
    - *e.g. if you start the script manually in Fusion 360 itself.*
- Automatically adds a dependency for the Fusion Python APIs
  - Used for autocomplete, contextual docs, etc.



https://plugins.jetbrains.com/plugin/11343-fusion-360-scripting

AUTODESK®

Make anything™

Appendix A - Samples

Samples

My main page for these projects: https://tapnair.github.io/index.html

ventMaker - Create custom vent features in Fusion 360. Circular, Slot and rectangular vents.

HelixGenerator - Generate Helical Curves in Fusion 360

ParamEdit - Quick editor to make changes to user parameters with real time update.

stateSaver - Save the current state of: hide/show, suppress/unsuppress, and user parameter values.

ShowHidden - Display utilities for Fusion 360. Show hidden or all: bodies, components and planes.

Project-Archiver - Automate the export of all designs in a project to a local archive directory.

copyPaste - Copy and paste bodies between documents in Fusion 360, explicitly breaking references

NESTER - Semi automated nesting of sheet/flat parts in Fusion 360.

OctoFusion - Automate the process of exporting a file and sending it to Octoprint.

UGS_Fusion - Automate the process of posting a file and opening it in Universal G-code Sender