

Walk-in Slide: AU 2014 Social Media Feed

1. Click on the link below, this will open your web browser

<http://aucache.autodesk.com/social/visualization.html>

2. Use “Extended Display” to project the website on screen if you plan to work on your computer. Use “Duplicate” to display same image on screen and computer.

Fusion 360 Extensibility

Robert Angus – Solution Architect

Patrick Rainsberry – Business Development Manager, Fusion Platform

Class summary

This class will look at the Fusion 360 application's APIs (application programming interfaces) and how you can use these APIs to extend the Fusion 360 app functionality. We will also discuss how you can use the APIs as part of your automation or integration processes

Key learning objectives

At the end of this class, you will be able to:

- Understand the different development environments in the Fusion 360 app
- Discover key usages of the different environments and the strengths and limitations of each
- Learn how to use the new JavaScript API
- Discover the types of operations available via the Fusion 360 API

Introduction to the Fusion API

Fusion API History

- Fusion is a new product and the API is evolving.
 - API like behavior was included with Fusion for QA purposes.
 - Text scripts
- Initial Python based API was implemented.
 - More of a prototype
 - Not officially supported
 - No IDE or debugging
- An official API development project was undertaken which evolved to our current API.

Current Fusion API

- Needs to support be able to support OSX, Windows, and possibly UNIX.
 - C++
 - JavaScript
 - Python



More about the new API

- Autodesk has hundreds of man years experience in API development.
 - New API leverages this experience to create a clean API.
- The new API has a object model which is designed to be program language independent.
 - Different language implementations are thin wrappers that make calls to the underlying C++ implementation.
 - Leads to the possibility to adding new languages (VB.net, C#)

Languages: C++

- C++ is a general purpose programming language. It has imperative, object-oriented and generic programming features, while also providing the facilities for low level memory manipulation.
 - Statically typed
 - Supported on most platforms
 - High performance
 - More complex to learn and debug



Languages: C++

- The C++ API is not yet released.
- Probably best used for computationally or behaviorally intensive Add-ins.
 - Add-in framework still in development
- C++ is more difficult to develop and test.
- Need libraries that are platform independent.
 - Network communications (SSL)
 - UI
 - File system



Languages: Python

- Python is a widely used general-purpose, high-level programming language that is designed to be concise and human readable.
 - Supports object-oriented, imperative and functional programming
 - Dynamically typed (interpreted)
 - Automatic memory management
 - A large set of standard and third party libraries



Languages: JavaScript

- JavaScript is a dynamic computer programming commonly used as part of web browsers. JavaScript executes asynchronously in the browser.
 - JavaScript is prototype-based scripting language
 - Supports object-oriented,[6] imperative, and functional programming styles
 - Dynamically typed (interpreted)
 - Automatic memory management
 - Sandbox and cross site scripting issues



Execution Differences

Python = Faster

Fusion Process

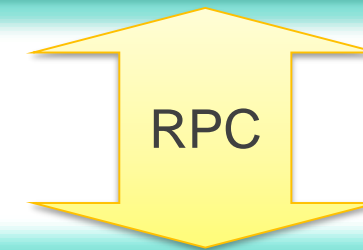
Python Engine
(in process)



OS Recourses

JavaScript = Slower

Fusion Process



Browser
Process



OS Resources

Languages: Summary

	C++	Python	JavaScript
Performance	Best	Good	Poor (runs out of process)
Typing	Static	Dynamic	Dynamic
Object Oriented Programming	Yes	Yes	Yes (but prototypical)
Procedural	Yes	Yes	Yes
Libraries	Many but few standard	Many standard libraries	Many but not standard
Memory Management	Manual	Automatic	Automatic
Debugging	IDE (not included)	IDE (included)	Web browser
Ease of cross platform development	Poor (often need different libraries)	Easy	Easy

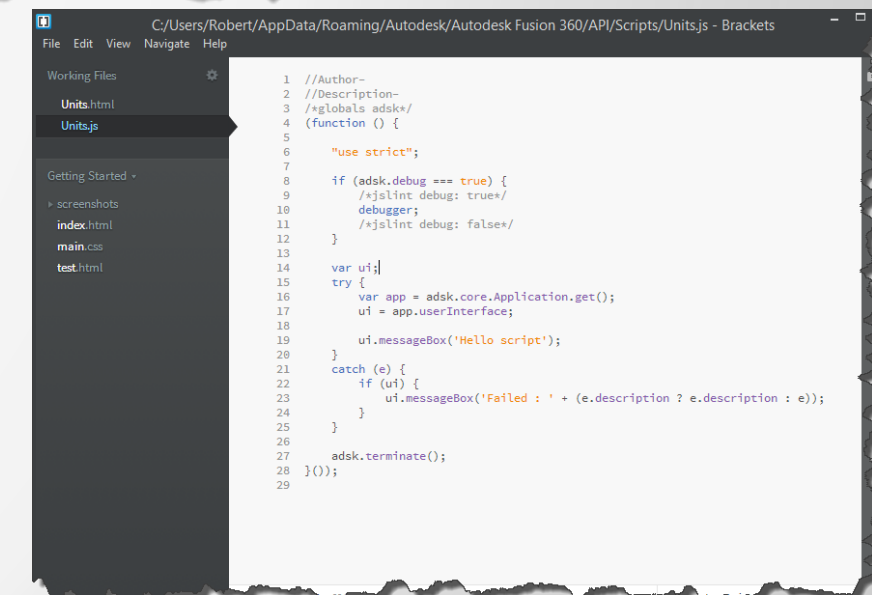
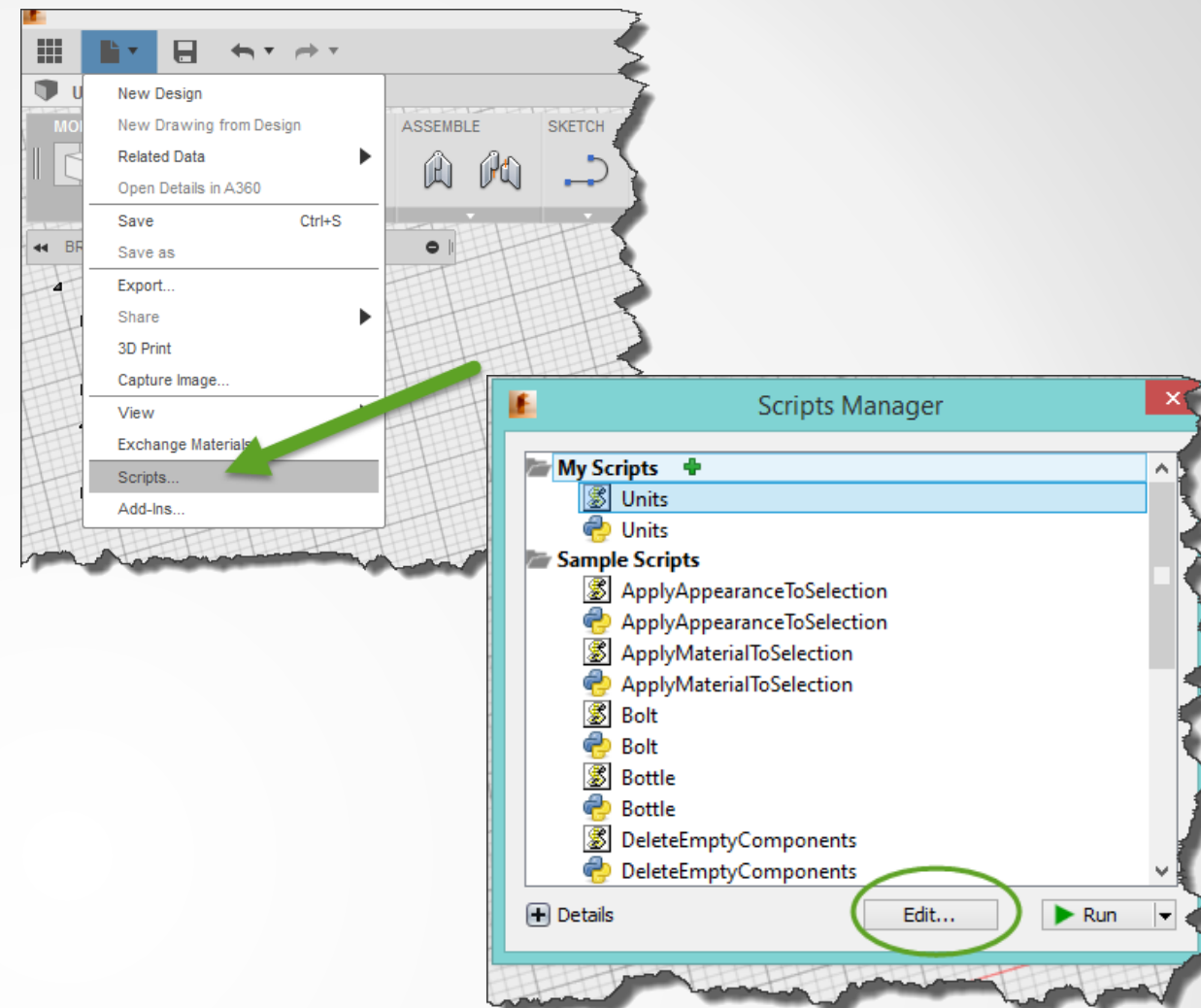
Working with JavaScript

- Edit source in Brackets
- Each script contains

1. .js file
2. .html file

.js file contains script code

.html file is loaded by the browser,
contains the references to scripts,
and starts script execution.



Working with JavaScript: HTML File

```
1 <!DOCTYPE html>
2 <html>
3   <script type="text/javascript" charset="UTF-8" src="../../JavaScript/src/adsk/core/application.js"></script>
4   <script type="text/javascript" charset="UTF-8" src="../../JavaScript/src/adsk/core/geometry.js"></script>
5   <script type="text/javascript" charset="UTF-8" src="../../JavaScript/src/adsk/core/materials.js"></script>
6   <script type="text/javascript" charset="UTF-8" src="../../JavaScript/src/adsk/core/userInterface.js"></script>
7   <script type="text/javascript" charset="UTF-8" src="../../JavaScript/src/adsk/utilities.js"></script>
8   <script type="text/javascript" charset="UTF-8" src="../../JavaScript/src/adsk/Fusion/bRep.js"></script>
9   <script type="text/javascript" charset="UTF-8" src="../../JavaScript/src/adsk/Fusion/components.js"></script>
10  <script type="text/javascript" charset="UTF-8" src="../../JavaScript/src/adsk/Fusion/construction.js"></script>
11  <script type="text/javascript" charset="UTF-8" src="../../JavaScript/src/adsk/Fusion/features.js"></script>
12  <script type="text/javascript" charset="UTF-8" src="../../JavaScript/src/adsk/Fusion/fusion.js"></script>
13  <script type="text/javascript" charset="UTF-8" src="../../JavaScript/src/adsk/Fusion/meshBody.js"></script>
14  <script type="text/javascript" charset="UTF-8" src="../../JavaScript/src/adsk/Fusion/meshData.js"></script>
15  <script type="text/javascript" charset="UTF-8" src="../../JavaScript/src/adsk/Fusion/sketch.js"></script>
16  <script type="text/javascript" charset="UTF-8" src="../../JavaScript/src/adsk/Fusion/tSpline.js"></script>
17
18  <script type="text/javascript" charset="UTF-8" src="Units.js"></script>
19 </body>
20 </html>
```

Paths to all of the Fusion libraries.

Note that paths are relative to the html file location.

Path to the script source. References to other JavaScript libraries need to be added here

Working with JavaScript: JavaScript File

```
1 //Author-
2 //Description-
3 /*globals adsk*/
4 (function () {
5
6     "use strict";
7
8     if (adsk.debug === true) {
9         /*jslint debug: true*/
10        debugger;
11        /*jslint debug: false*/
12    }
13
14    var ui;
15    try {
16        var app = adsk.core.Application.get();
17        ui = app.userInterface;
18        ui.messageBox('Hello script');
19    }
20    catch (e) {
21        if (ui) {
22            ui.messageBox('Failed : ' + (e.description ? e.description : e));
23        }
24    }
25
26    adsk.terminate();
27 }());
28
```

Function that is executed

When debugging this causes the browsers debugger to break

Debugging JavaScript

- Debugging is out of process in chrome

Step through code

Can view variable and global values

Set and Edit Breakpoints

Object Inspection

Tabs for viewing HTML and JavaScript

JavaScript Equality

- The == operator check for values are equal ("5" == 5)
- The === operator checks that both values and types are equal ("5" != 5)
- For objects instances the operators == and === return true if they are pointing to the same object (same memory location).
- From help:

```
var plane1 = component.constructionPlanes.item(0);  
var plane2 = component.constructionPlanes.item(0);  
plane1 === plane2; // false  
plane1.equals(plane2); // true
```
- Plane1 and Plane2 are both the same model entity, but are two different JavaScript object instances and when compared using === return false.
- The equals() method compares to see if two objects are the save model entity.

JavaScript Reference Arguments

- Java script does not support reference or out parameters.

```
boolean Point2D.getData( out double x, out double y )
```

- To handle this the API expects these parameters to be objects

```
var xObject = {};  
var yObject = {};  
point.getData(xObject, yObject );
```


JavaScript Object Casting

- In JavaScript object are dynamically typed.
 - IDE does not know the structure of a referenced object
- Using the following construct will enable intellisense and insure that you are working with the correct object type

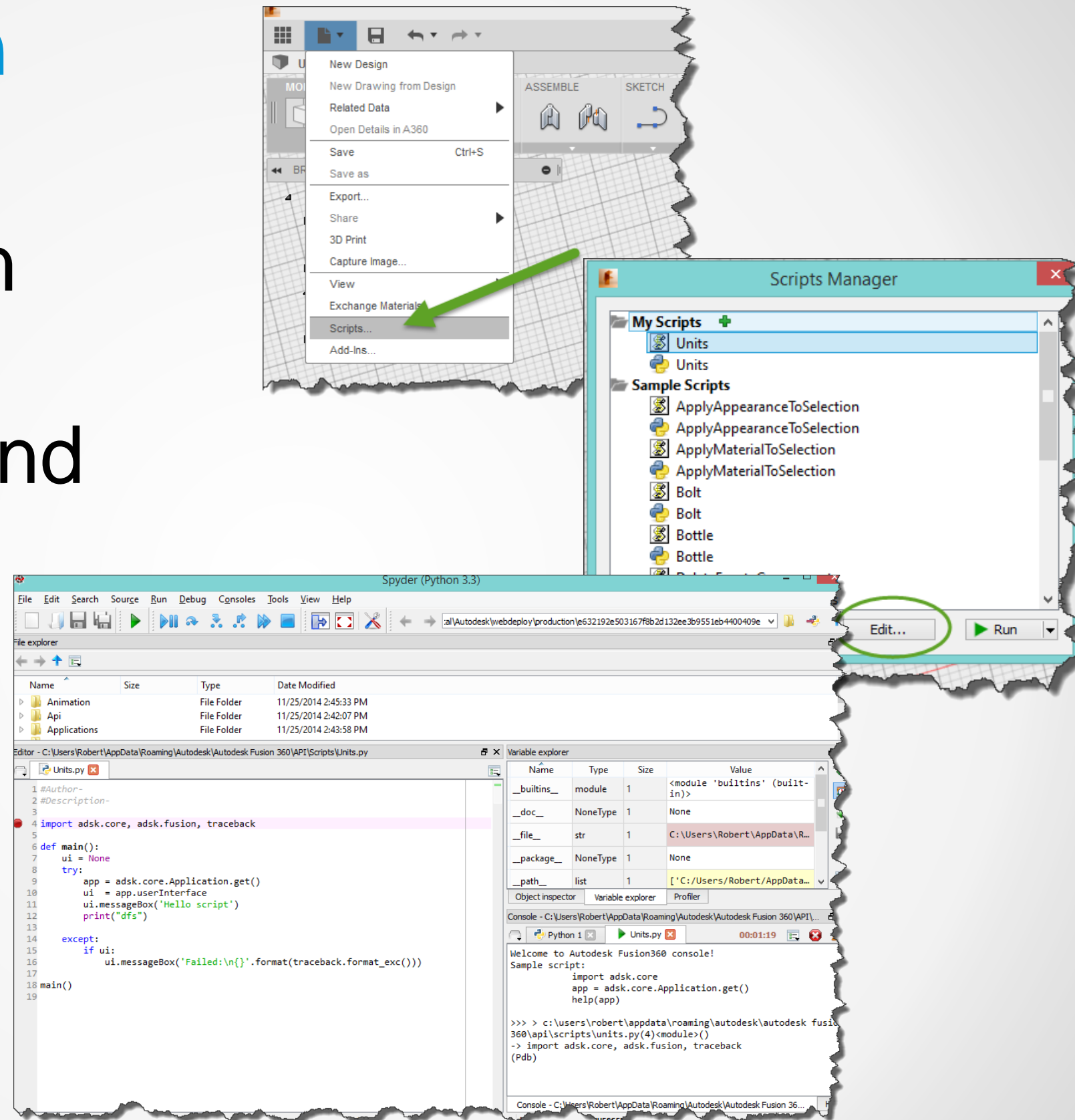
```
var face = adsk.fusion.BRepFace(selection.object);  
if (face) {  
    // Call BRepFace members here.  
    var surface = face.geometry;  
}
```

JavaScript OS Utilities

- The Fusion API provides utility functions for JavaScript scripts to access OS functionality missing due to sandbox restrictions
 - `adsk.readFile`
 - `adsk.writeFile`
 - `adsk.copyFile`
 - `adsk.copyFile(fromFilename, toFilename)`
 - `adsk.renameFile`
 - `adsk.removeFile`
 - `adsk.createDirectory`
 - `adsk.createDirectory(filename)`
 - `adsk.listDirectoryFiles`
 - `adsk.listDirectoryFiles(filename, regularFilesOnly, recursive)`
 - `adsk.fileExists`
 - `adsk.fileIsDirectory`
 - `adsk.fileSize`
 - `adsk.tempDirectory`
 - `adsk.toBase64`
 - `adsk.fromBase64`
 - `adsk.utf8ToString`

Working with Python

- Source is stored in .py files
- Source is edited and debugged in the Spyder IDE



Sypder IDE

Spyder (Python 3.3)

File Edit Search Source Run Debug Consoles Tools View Help

File explorer

Name	Size	Type	Date Modified
Animation		File Folder	11/25/2014 2:45:33 PM
Api		File Folder	11/25/2014 2:42:07 PM
Applications		File Folder	11/25/2014 2:43:58 PM

Editor - C:\Users\Robert\AppData\Roaming\Autodesk\Autodesk Fusion 360\API\Scripts\Units.py

Units.py

```
1 # Author-
2 # Description-
3
4 import adsk.core, adsk.fusion, traceback
5
6 def main():
7     ui = None
8     try:
9         app = adsk.core.Application.get()
10         ui = app.userInterface
11         ui.messageBox('Hello script')
12         print("dfs")
13     except:
14         if ui:
15             ui.messageBox('Failed:\n{}'.format(traceback.format_exc()))
16
17 main()
18
19
```

Variable Explorer

Name	Type	Size	Value
__builtins__	module	1	<module 'builtins' (built-in)>
__doc__	NoneType	1	None
__file__	str	1	C:\Users\Robert\AppData\Roaming\Autodesk\Autodesk Fusion 360\API\Scripts\Units.py
__package__	NoneType	1	None
__path__	list	1	['C:/Users/Robert/AppData\Roaming\Autodesk\Autodesk Fusion 360\API\Scripts']

Object inspector Variable explorer Profiler

Console - C:\Users\Robert\AppData\Roaming\Autodesk\Autodesk Fusion 360\API\Scripts\Units.py

Python 1 Units.py 00:01:19

Welcome to Autodesk Fusion360 console!
Sample script:
import adsk.core
app = adsk.core.Application.get()
help(app)

>>> > c:\users\robert\appdata\roaming\autodesk\autodesk fusion360\api\scripts\units.py(4)<module>()
-> import adsk.core, adsk.fusion, traceback (Pdb)

Breakpoints

Can use print() function to query state

Spyder (Python 3.3)

File Edit Search Source Run Debug Consoles Tools View Help

File explorer

Name	Size	Type	Date Modified
Animation		File Folder	11/25/2014 2:45:33 PM
Api		File Folder	11/25/2014 2:42:07 PM
Applications		File Folder	11/25/2014 2:43:58 PM

Editor - C:\Users\Robert\AppData\Roaming\Autodesk\Autodesk Fusion 360\API\Scripts\Units.py

Units.py

```
re, adsk.fusion, traceback
adsk.core.Application.get()
app.userInterface
ui.messageBox('Hello script')
print("dfs")
ui.messageBox('Failed:\n{}'.format(traceback.format_exc()))
```

Source is edited here

Variable Explorer

Name	Type	Size	Value
__builtins__	module	1	<module 'builtins' (built-in)>
__doc__	NoneType	1	None
__file__	str	1	C:\Users\Robert\AppData\Roaming\Autodesk\Autodesk Fusion 360\API\Scripts\Units.py
__package__	NoneType	1	None
__path__	list	1	['C:/Users/Robert/AppData\Roaming\Autodesk\Autodesk Fusion 360\API\Scripts']

Object inspector Variable explorer Profiler

Console - C:\Users\Robert\AppData\Roaming\Autodesk\Autodesk Fusion 360\API\Scripts\Units.py

Python 1 Units.py 00:01:19

Welcome to Autodesk Fusion360 console!
Sample script:
import adsk.core
app = adsk.core.Application.get()
help(app)

>>> > c:\users\robert\appdata\roaming\autodesk\autodesk fusion360\api\scripts\units.py(4)<module>()
-> import adsk.core, adsk.fusion, traceback (Pdb)

View current code executing here

Python Reference Arguments

- In Python reference arguments are return as a tuple (immutable list)

```
boolean Point2D.getData( out double x, out double y )
```

- The first value of the tuple is the return value of the function. The out or reference parameters are the next values.

```
results = point.getData()  
if results[0]:  
    (retVal, x, y) = results  
    print( "x = {0}, y = {1}".format(x, y) )
```


Python Type Checking

- The “is” operator is often used to check type.
 - This operator will not check to see if you are a generalized type of something.

```
selObj = app.activeSelection.item(0).object  
itype(selObj) is adsk.fusion.SketchEntity # false
```

- Use the isinstanceOf function

```
selObj = app.activeSelection.item(0).object  
isinstance(selObj, adsk.fusion.SketchEntity)  
# will return true if it is a SketchEntity instance
```


Python Type Checking

- The “is” operator is often used to check type.
 - This operator will not check to see if you are a generalized type of something.

```
selObj = app.activeSelection.item(0).object  
itype(selObj) is adsk.fusion.SketchEntity # false
```

- Use the isinstanceOf function

```
selObj = app.activeSelection.item(0).object  
isinstance(selObj, adsk.fusion.SketchEntity)  
# will return true if it is a SketchEntity instance
```

Units

- Fusion Model Units
 - cm (so areas and volumes are cm^2 and cm^3)
 - radians
 - kg
- Documents have active units
- Users can input any unit
 - 3
 - 3 in + 5 in
 - 3 m^2

UnitsManager

- UnitsManager is a utility class to work with 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"

Parameters

- When you add features you define parameters by creating inputValues.
 - You can create them by value
 - `var x = adsk.core.ValueInput.createByReal(23)`
 - You can create them using an expression
 - `var x = adsk.core.ValueInput.createByString("23 in");`
- When changing the value of a parameter you can change the value or the expression.

Commands

- You add a command by creating a CommandDefinition
 - Button
 - Checkbox
 - Listbox
- A commands have a list of inputs where you can define UI input controls to get values from uses
- Commands have a events

Building a Script

Practical Application

Build up a file importer

- Read files
- Create components
- Position components



Document, Product, Component, Occurrence & Proxies

- Document: Data panel item. ~Fusion File
- Product: Data type. Design, Tool-path, etc.
- Component: Unique part geometry
- Occurrence: Instance of a component
- Proxy: Reference to occurrence geometry

Document

Product: Design

Product: Toolpath

Component1

Component2

Root Component

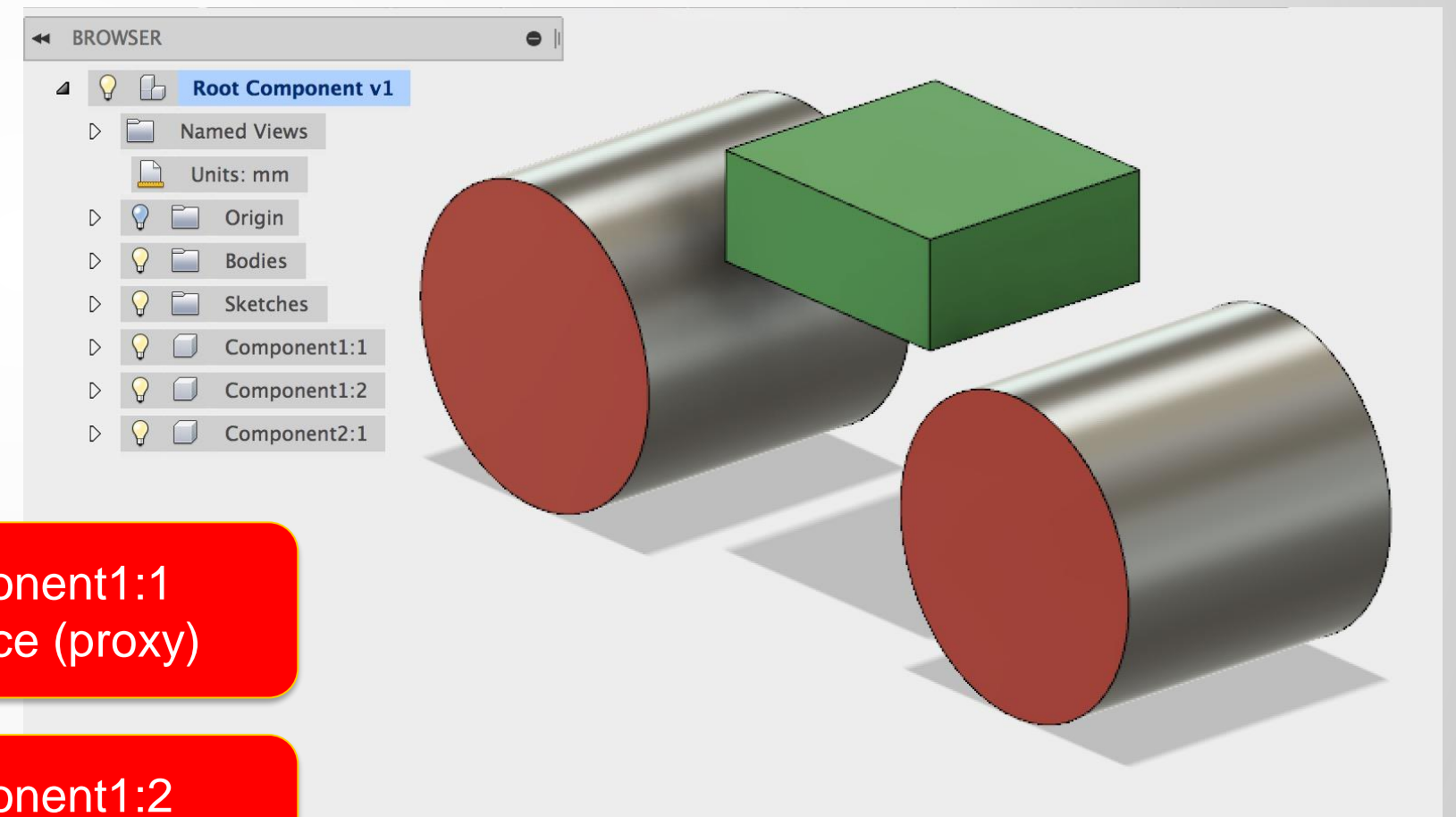
Component1:
Occurrence 1

Component1:
Occurrence 2

Component2:
Occurrence 1

Component1:1
/RedFace (proxy)

Component1:2
/RedFace (proxy)



“Component1:1/RedFace”

Collections

- Collections provide access to a common set of objects.
- Sketch collection contains all sketches in a component
- Create a new sketch by adding an item to the collection
- Lines collection contains all lines within a sketch
- Circles collection contains all circles in the sketch, etc...
- To create a new line add to the Lines collection

Sketching

- Get Sketches collection from component
- Create a new sketch on a plane
- Get Lines collection
- Create 2 points
- Add a line to the collection from points



Sketching

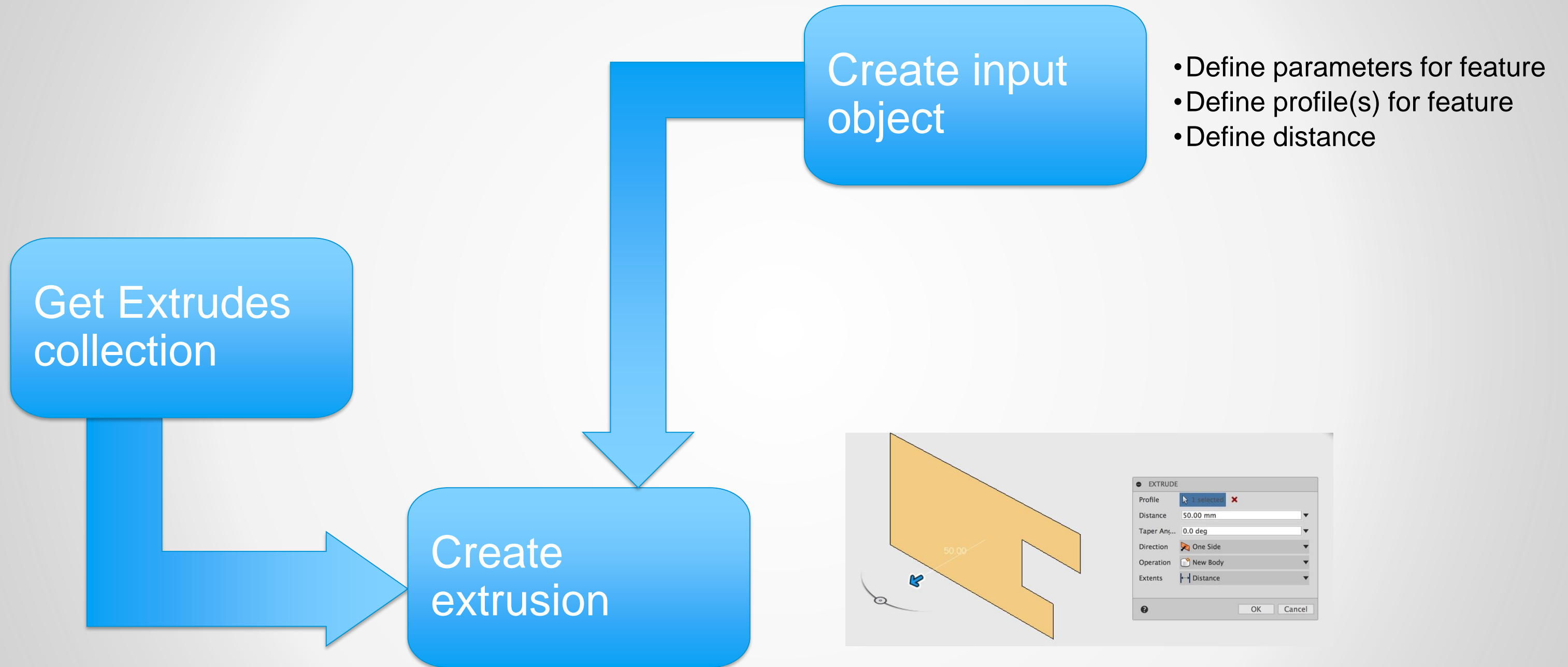
```
// Get reference to the sketches and plane
var sketches = component.sketches;
var xyPlane = component.xYConstructionPlane;

// Create a new sketch by adding to sketches collection
var sketch = sketches.add(xyPlane, null);

// Create first two points
var x, y, z;
x=0;y=0;z=0;
var point1 = adsk.core.Point3D.create(x, y, z);
x=1;y=0;z=0;
var point2 = adsk.core.Point3D.create(x, y, z);

// Get lines collection, add a new line
var lines = sketch.sketchCurves.sketchLines;
lines.addByTwoPoints(point1, point2);
```

Feature Creation (Extrusion)



By adding a new extrude to the extrudes collection

Feature Creation

```
// Get reference to the first profile of the sketch
var profile = sketch.profiles.item(0);

// Get extrude features reference
var extrudes = component.features.extrudeFeatures;

// Create input object for the extrude feature
var extInput = extrudes.createInput(profile, adsk.fusion.FeatureOperations.NewBodyFeatureOperation);
var distance = adsk.core.ValueInput.createByReal(1);
extInput.setDistanceExtent(false, distance);

// Create extrusion
extrudes.add(extInput);
```

File System Interaction

- JavaScript does not allow for direct interaction with desktop files
- Special helper methods have been implemented, (see JavaScript Specific issues in documentation)

adsk.readFile
adsk.writeFile
adsk.copyFile
adsk.renameFile
adsk.removeFile
adsk.createDirectory
adsk.listDirectoryFiles

adsk.fileExists
adsk.fileIsDirectory
adsk.fileSize
adsk.tempDirectory
adsk.toBase64
adsk.fromBase64
adsk.utf8ToString

File System Interaction

```
// Setup the file open dialog
var dlg = ui.createFileDialog();
dlg.title = 'Open CSV File';
dlg.filter = 'Comma Separated Values (*.csv);;All Files (*.*)';

if (dlg.showOpen() !== adsk.core.DialogResults.DialogOK) {
    adsk.terminate();
    return;
}

// Read the file
var filename = dlg.filename;
var buffer = adsk.readFile(filename);

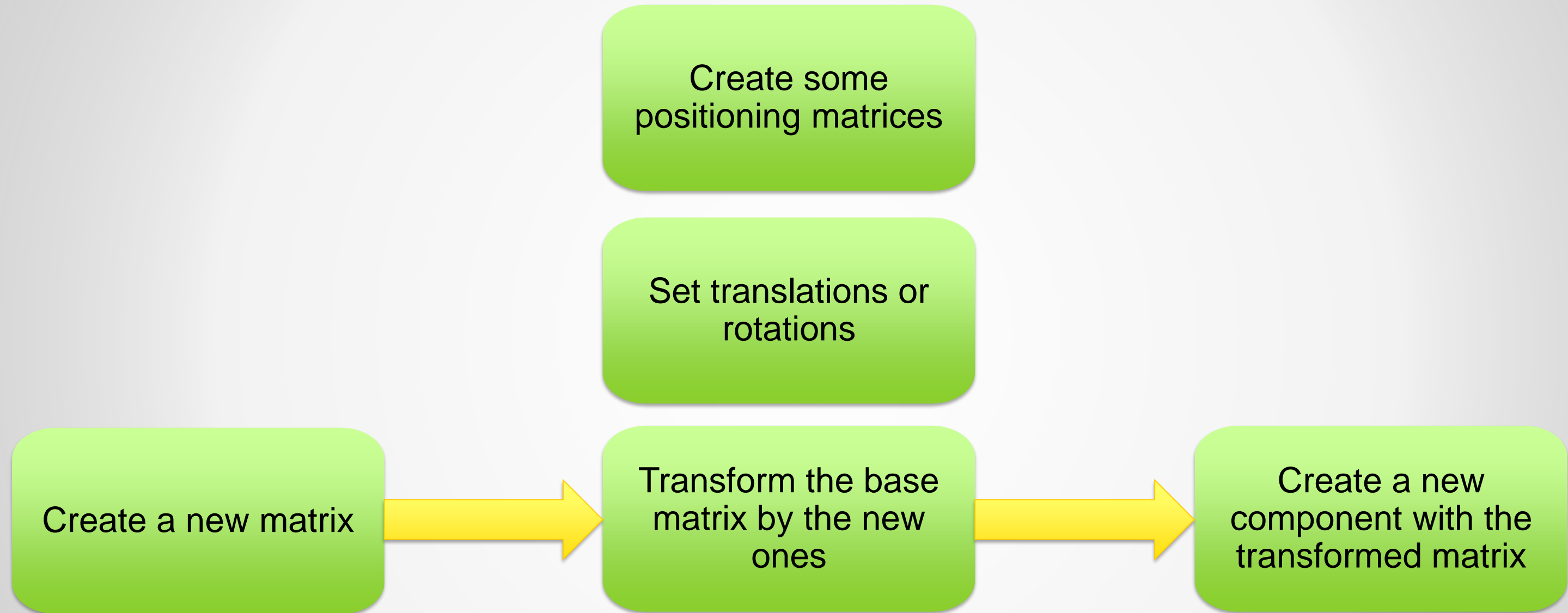
if (!buffer) {
    ui.messageBox('Failed to open ' + filename);
    adsk.terminate();
    return;
}
```


File Processing

```
// Process the data
var data = adsk.utf8ToString(buffer);
data = data.split('\n');
var i, j;
var points = []; /* Array to hold points from file */

for (i = 0; i < data.length; ++i) {
    data[i] = data[i].split(',');
    for (j = 0; j < data[i].length; ++j) {
        data[i][j] = parseFloat(data[i][j]);
    }
    if (data[i].length >= 3) {
        // Create a Point and add it to the array
        var point = adsk.core.Point3D.create(data[i][0], data[i][1], data[i][2]);
        points.push(point);
    }
}
```

Creating and positioning a component



Create/position Occurrences

```
// Loop to create pattern
for (var c = 1; c < patternNum; c++){

// Setup translation and apply it to the base matrix
    vector = adsk.core.Vector3D.create(x, y*c, z);
    trans.translation = vector;
    matrix.transformBy(trans);

// Setup Rotation and apply it to the base matrix
    rotation.setToRotation(angle*c, adsk.core.Vector3D.create(0,0,1), adsk.core.Point3D.create(0.5, 0.5, 0));
    matrix.transformBy(rotation);

//Create new occurrence of the part
    occurrence = rootComp.occurrences.addExistingComponent(component, matrix);

// Re-initialize the matrix
    matrix.setTolIdentity();
    trans.setTolIdentity();
    rotation.setTolIdentity();
}
```

```
// Pattern Component magic numbers
var x = 0, y = 2, z = 0;
var angle = Math.PI/6;
var patternNum = 30;

var matrix = adsk.core.Matrix3D.create();
var trans = adsk.core.Matrix3D.create();
var rotation = adsk.core.Matrix3D.create();
var vector;
```

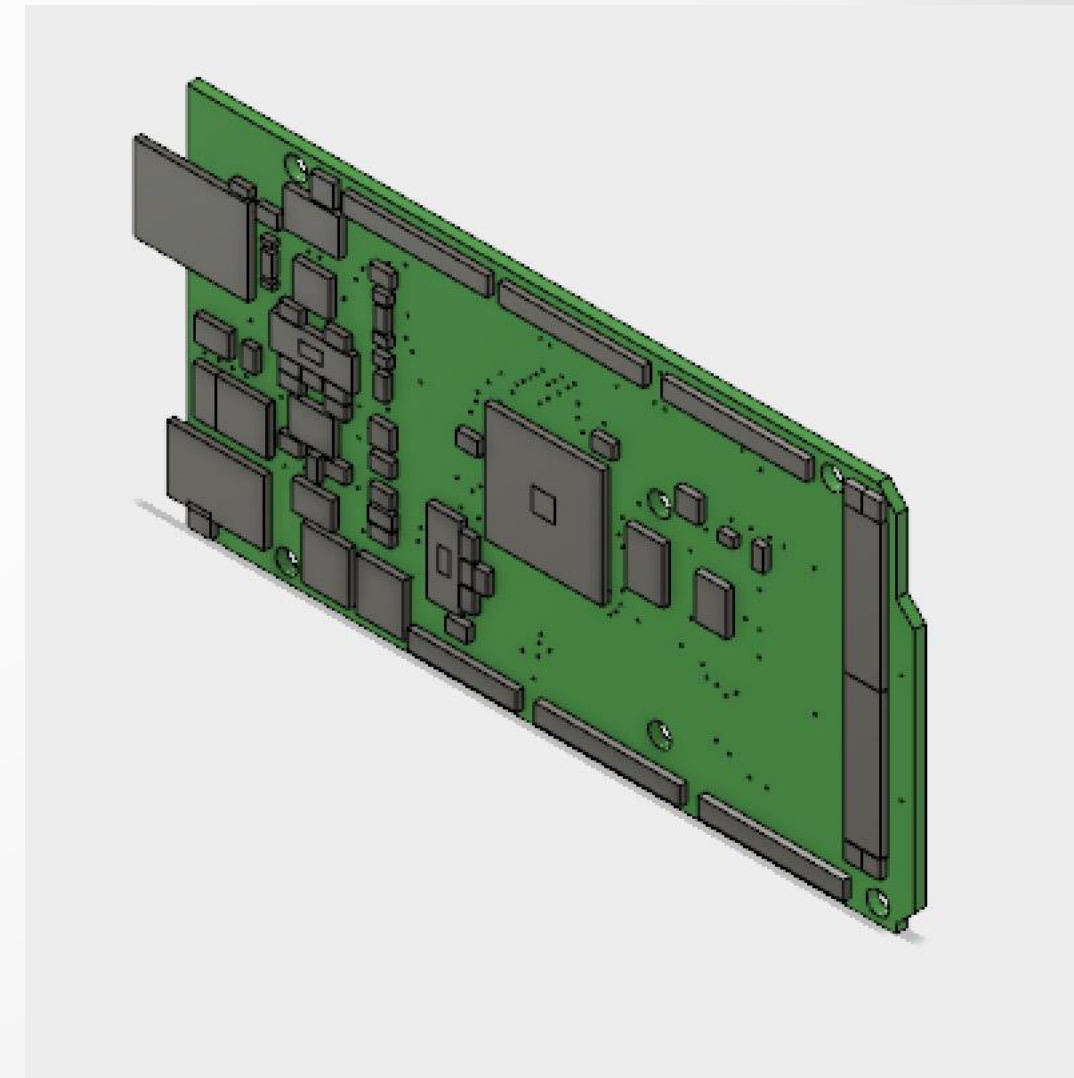
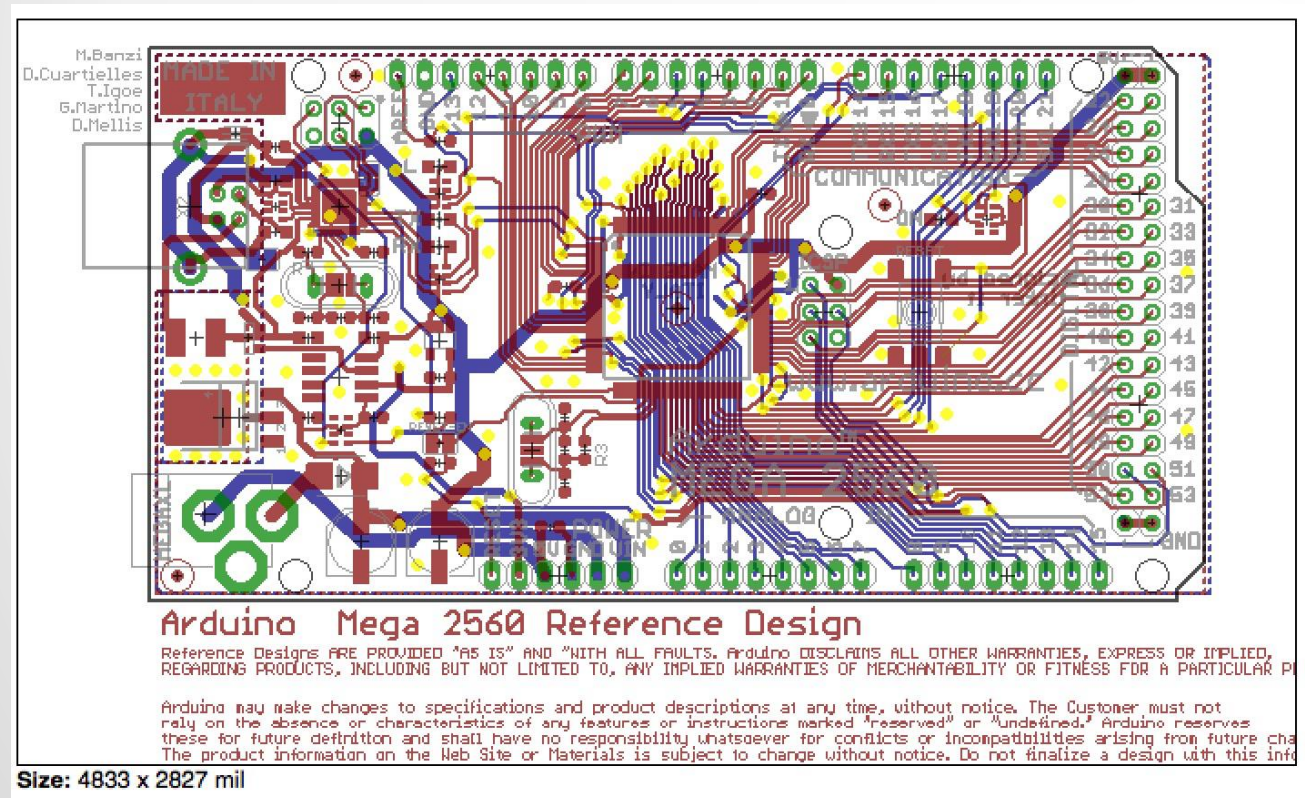

An aerial perspective of a cityscape. In the foreground, a multi-lane bridge with a rainbow-colored line along its edge spans a river. A red car is visible on the bridge. To the right of the bridge is a green park area with a blue oval field. In the background, a large stadium with a white roof is visible, surrounded by various city buildings and skyscrapers under a clear blue sky.

JavaScript Example: IDF Importer



Goal

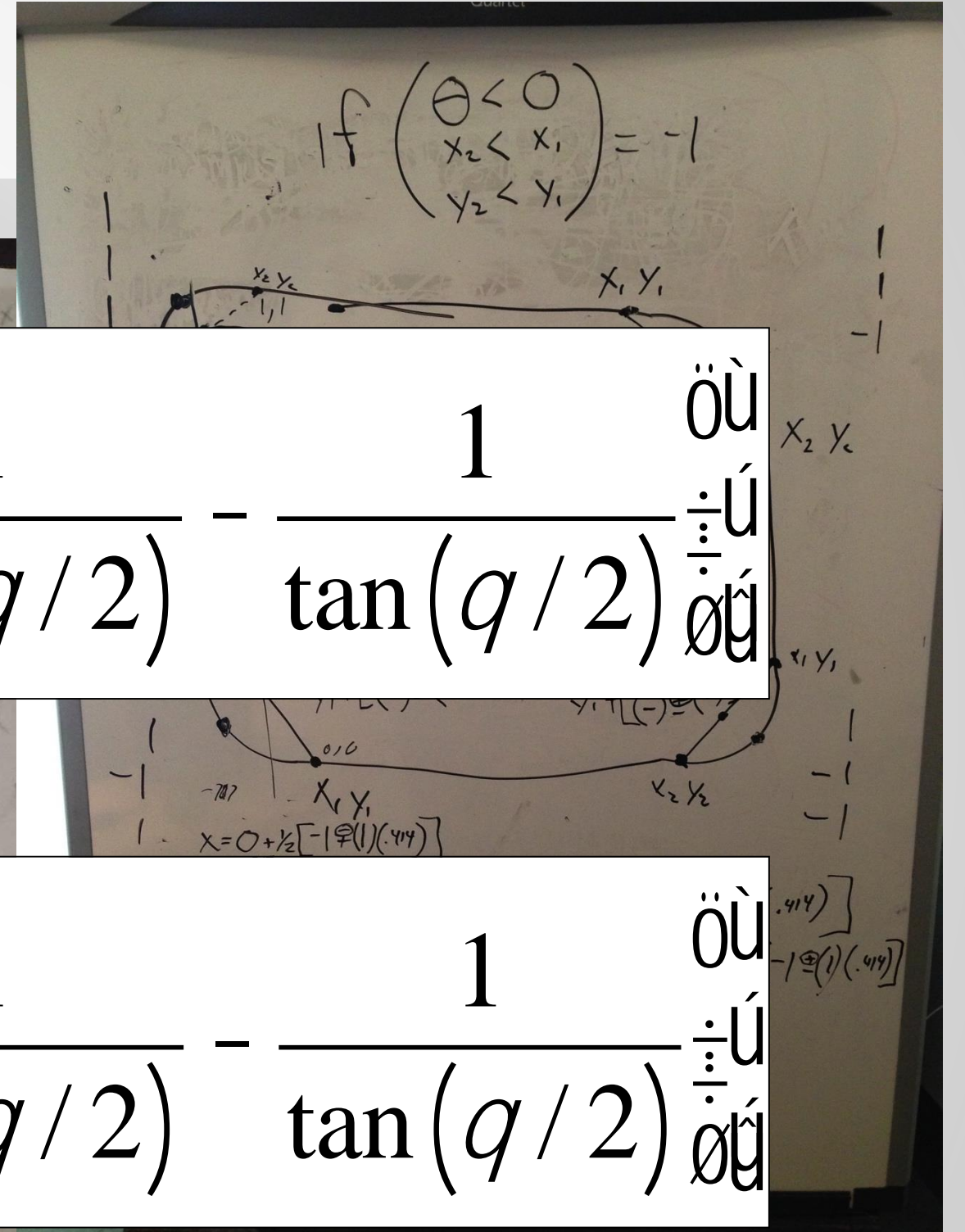
- IDF files are used to translate 2D circuit boards into 3D CAD systems.



Lost in translation...

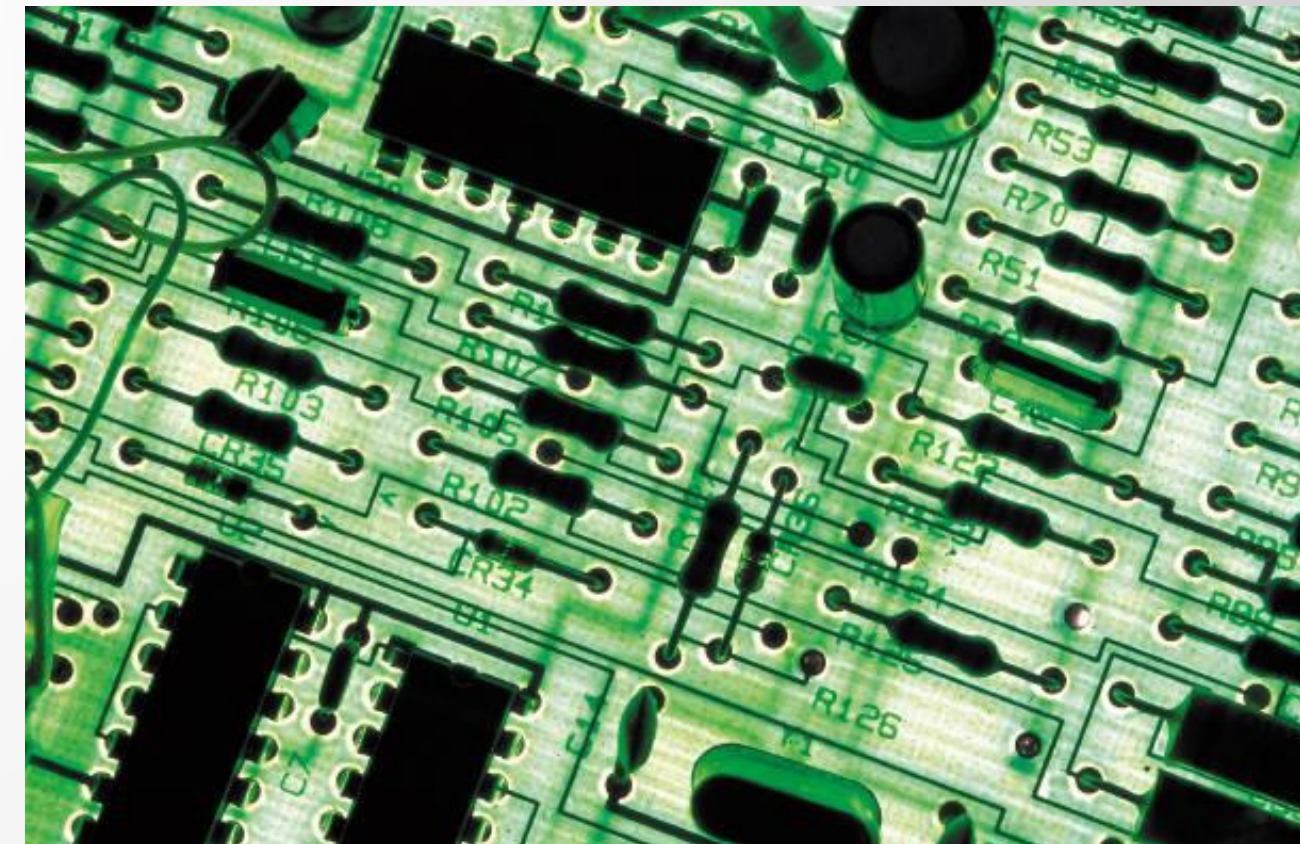
$$x_3 = x_1 + \frac{1}{2} \hat{e} \left(x_2 - x_1 \right) \pm \left(y_2 - y_1 \right) \frac{1}{\sin(q/2)} - \frac{1}{\tan(q/2)}$$

$$y_3 = y_1 + \frac{1}{2} \hat{e} \left(y_2 - y_1 \right) \pm \left(x_2 - x_1 \right) \frac{1}{\sin(q/2)} - \frac{1}{\tan(q/2)}$$





Script Overview


- Launch options menu
- Select IDF files (.emn, emp)
- Process Library files
- Process Board File
 - Draw board outline
 - Draw holes
 - Draw Keep-outs
 - Place components





Getting help and feedback


 **AUTODESK® FUSION 360™**


 **COMMUNITY**

 GALLERY

 LEARNING

 BLOG

 FEATURES

 SIGN IN

Autodesk Community > Fusion 360 > Preview: API and Scripts

Preview: API and Scripts

Create a New Post →

Options ↓

Quick Links
[Register](#)
[Sign In](#)
Recently Solved
Simple Python Example Request
Preview: API and Scripts
Python editing and debugging does not work
Preview: API and Scripts
Python API Documentation
Preview: API and Scripts
Availability of javascript standard libraries such...

Cloud Developer Accelerator

- 2 Week intensive in San Francisco
- We pay the hotel
- jim.quanci@autodesk.com

THANKS!



patrick.rainsberry@autodesk.com

PRAINSBERRY

robert.angus@autodesk.com



@prainsberry



<http://www.linkedin.com/in/patrickrainsberry/>

Session Feedback

- Via the Survey Stations, email or mobile device
- AU 2015 passes given out each day!
- Best to do it right after the session
- Instructors see results in real-time







Students, educators, and schools now have

FREE access to Autodesk design software & apps.

Download at www.autodesk.com/education



Earn your professional Autodesk Certification at AU

Visit the [AU Certification Lab](#)