# Setting up Python / IfcOpenShell on your machine

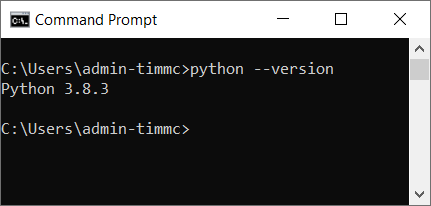
We have been using the RWTH viewer, which is great to get us started, but we do not always need the graphical interface and you might like to run python form the command line / terminal. The advantage to this approach is you can have better control of the script and the libraries that you are including. Ok now we have Python set up we can add the IfcOpenShell Library.

## Step 1: Check if python installed on your machine and what version it is.

To do this we need to add download the correct version of IfcOpenShell which depends on your operating system and the version of python you are using. To find this out.

1. Go to the command line in windows or the Terminal in mac and type:

python --version



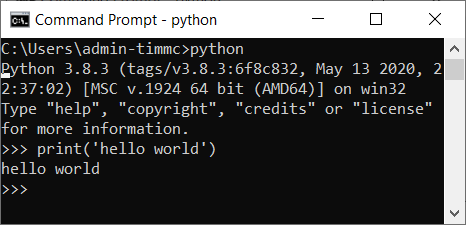
This should tell you the version of python that you are using. For instance you can see here that I am using 3.8.3. So I can download the correct version from <http://ifcopenshell.org/python>.

## Step 2: jump into and out of python from the command line / terminal (optional)

You can see that this time I have typed

python

And it has given me the python interpreter >>>

I can now type python commands into this and it will give me an output. for instance:

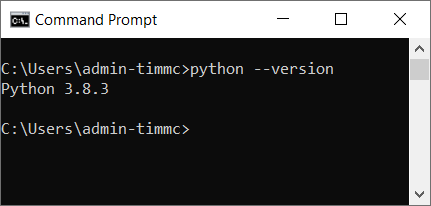
print(‘hello world’)

Produces the result

hello world

To get out of the python interpreter type

exit()

So you now shouldn’t be able to see the >>> prompt and it should instead be something more like  or the equivalent on your machine.

## Step 3: run your first python program.

It is a bit annoying to have to write out each line separately into the python interpreter, to get around this we can write a python script in a text editor and then save this a .py file.

So first we set up a location for your program. I am using c:/github as a root location on my machine, it is up to you where you choose, but it should be somewhere easy to find.

#### Step 3.1 creating a .bat file to get to your location (optional)

This is an optional step, but I find it makes it easier - I don’t have a mac equivalent yet, and it is not an essential step. so create a new file on your desktop and rename it to [name of the program].bat for instance mine is called rule.bat.

Open that file in a text editor for instance notepad++ or sublime and copy in the following.

@ECHO OFF

cmd.exe /K "cd C:\github\rule && c:"

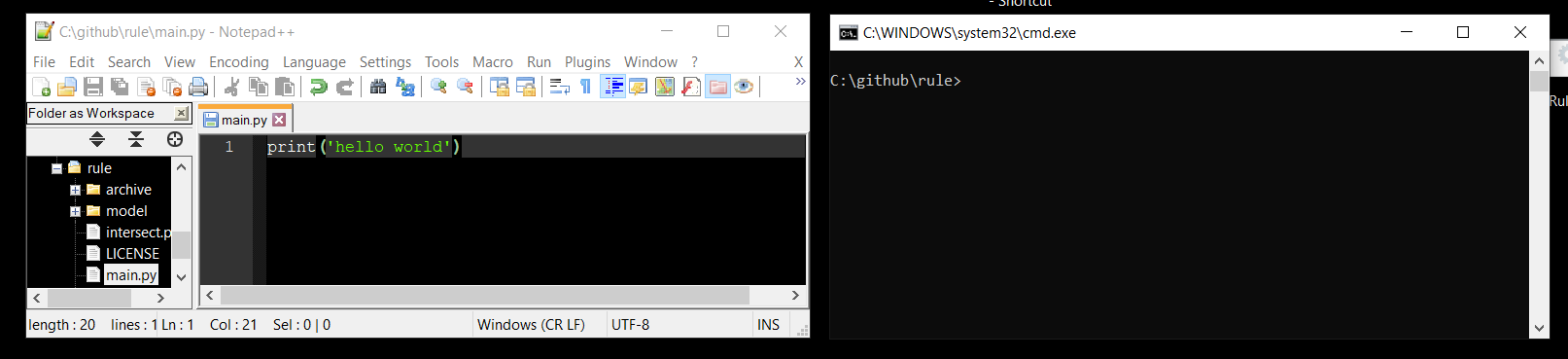
Save it and then when you double click it on your desktop it will automatically open the cmd line in the right place. Please note that the bat file uses backslashes.



Ok so now we have set that up go to the program folder (in our case c:/github/rule). In that folder create a new file called main.py. Open up that folder in a text editor (notepad++ for instance) and in the file type:

print (‘hello world’)

Now save the file - you can keep the text editor open so that you have the text editor and cmd line side by side for instance…

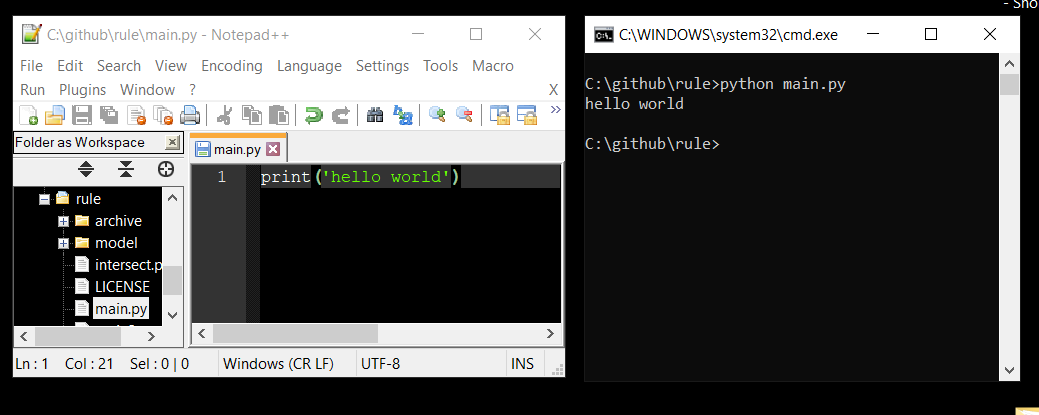


#### Step 3.2

Ok now, type into the cmd line

python main.py

This will run your program and you will see the result in the command line.



Cool eh? you just wrote and ran your first proper program in Python - good job!

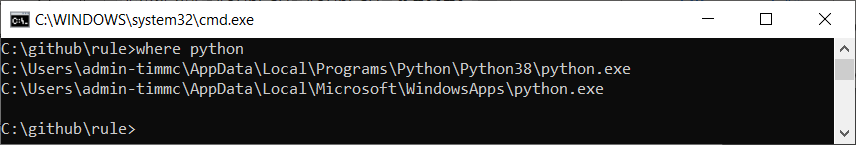
## Step 4 add IfcOpenShell to python so that you can include it in your program.

Having worked out what version of Python you have in step 1 we now need to work out where it is installed so that we can add the ifcOpenShell folder you downloaded in Step 1. (<http://ifcopenshell.org/python>). On a windows machine I will type …

where python

on a mac I can type:

which python

into Terminal which should give a similar result. On my machine it says two places, I choose the first one.

So I will go to

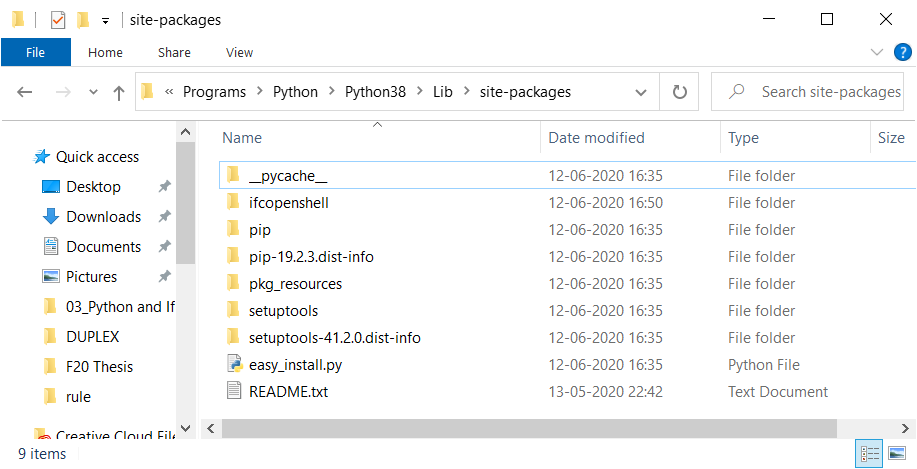
C:\Users\XXXX-XXXX\AppData\Local\Programs\Python\Python38\

n.b your location will not be XXXX-XXXX - this is an example!

and from there I will open the Lib/site-packages so I end up at

C:\Users\*admin-timmc*\AppData\Local\Programs\Python\Python38\Lib\site-packages

The downloaded ifcopenshell folder should go in there.



## Step 5: Test your setup:

Open up your program folder for me it was (c:/github/rule), but this could be anywhere for you. In the text editor open up the main.py file in your folder. Now in that folder type the following

import ifcopenshell

model = ifcopenshell.open('model\Duplex\_A\_20110907.ifc')

for obj in model.by\_type('IfcSlab'):

print(obj.Name)

Save that and then run it from the command line following Step 3.2 if this works, have a go at some of the code examples in the code examples document.

# Appendix F: Advanced Python scripts

*Appendix Firstly, as always massive thanks to Thomas Krijnen for most of this source, which has just been commented here, but is mostly intact from Thomas’s original examples from different locations.*

## Example 1: Get the property sets of an element

Only include the first chunk of text if you are NOT doing this in the RWTH viewer, in the viewer the import of the ifcopenshell library and getting the file has already been done for you and declared as model. Therefore in these examples we are also going to use model.

# ########### this is required if running from console ############

import ifcopenshell

model = ifcopenshell.open('model\Duplex\_A\_20110907.ifc')

# ########### end of required if running from console #############

# ############## code below needed in both cases ##################

# this just gets you the entity, defined here as wall

# feel free to change this to your needs

wall = model.by\_type('IfcWall')[0]

for definition in wall.IsDefinedBy:

# To support IFC2X3, we need to filter our results.

if definition.is\_a('IfcRelDefinesByProperties'):

property\_set = definition.RelatingPropertyDefinition

# Might return Pset\_WallCommon

print(property\_set.Name)

# ###################### end of example ###########################

## Example 2: Get the doors that bound a space (BoundedBy) - HARD

This example works to get you the doors (line 13) that bound the space (line 8)

# ########### this is required if running from console ############

import ifcopenshell

model = ifcopenshell.open('model\Duplex\_A\_20110907.ifc')

# ########### end of required if running from console #############

# ############## code below needed in both cases ##################

for space in model.by\_type("IfcSpace"):

near = space.BoundedBy

print("\n\t####{}\n".format(space.Name))

for objects in near:

if (objects.RelatedBuildingElement != None):

if (objects.RelatedBuildingElement.is\_a('IfcDoor')):

print(objects.RelatedBuildingElement.Name)

# ###################### end of example ###########################

## Example 3: Get the doors that bound a space (BoundedBy)

For this example we have to include an additional library, but it provides a really cool approach. Also please note that this example uses the optimized version of the Duplex model. This is also available in your models folder. Optimised versions of files are much smaller, they are optimized using a great tool ([Solibri IFC Optimizer](https://www.solibri.com/solibri-ifc-optimizer)) from Solibri. The idea is that it can be used to make IFC files easier to share.

# We need all this code and we can’t run it from RWTH viewer

import ifcopenshell

# That was normal the new bit is this geom lib below

import ifcopenshell.geom

# ok, so we are calling it fn (for file name here) - lets stick to that

fn = "model/Duplex\_A\_20110907\_optimized.ifc"

# based on the fn we can now create the model which is called f

f = ifcopenshell.open(fn)

# a specific wall is defined here based on its GlobalID

# we are working with standard files so you should also be able to find this.

# the small diff is that this is working on the optimized version

wall = f["2O2Fr$t4X7Zf8NOew3FLPP"]

# This is the magic code that loads the geometry for the models into its own model - so that we can query the geometry (and in this case the

tree\_settings = ifcopenshell.geom.settings()

tree\_settings.set(tree\_settings.DISABLE\_OPENING\_SUBTRACTIONS, True)

t = ifcopenshell.geom.tree(f, tree\_settings)

~~# you need the code below in both RWTH and if running it directly.~~

~~for space in model.by\_type("IfcSpace"):~~

~~near = space.BoundedBy~~

~~print("\n\t####{}\n".format(space.Name))~~

~~for objects in near:~~

~~if (objects.RelatedBuildingElement != None):~~

~~if (objects.RelatedBuildingElement.is\_a('IfcDoor')):~~

~~print(objects.RelatedBuildingElement.Name)~~

## Example 4: Define a class and function to load models (Hard)

For this example we will work with classes and functions to load the model, the reason for this is it will make it much more simple when we try and load multiple models in the next example.

# We need all this code and we can’t run it from RWTH viewer

import ifcopenshell

import ifcopenshell.geom

# we need this module to tell us how long our code took to run

import time

start\_time=time.time()

# This is the tree settings - you shouldn't need to change this. keep it as is.

# it is a smart way to structure the geometry of the model,

# so that we can do super fast 'bounding box' tests, where the

# smallest possible box isnt drawn around the object.

# the we can then do fast clash detection on those boxes.

# i.e. does the bounding box of the slab intersect the bounding box of any columns?

tree\_settings = ifcopenshell.geom.settings()

tree\_settings.set(tree\_settings.DISABLE\_OPENING\_SUBTRACTIONS, True)

# This is a class, - your first one! good job!

# it defines an object (Model) that has attributes (i.e. name)- cool eh?

# using this means we can get attribute info about the geometry model

class Model:

def \_\_init\_\_(self,name, file, geometry, load\_time):

self.name = name

self.file = file

self.geometry = geometry

self.load\_time = load\_time

# this is a function, we use functions when we want to repeat the same process in the code

# in python we need to define (def) the function before we can call it.

# we call the function below with model=getGeometry('ARCHI',f\_ifc, tree\_settings)

# the function makes an instance of the model class defined above and 'returns'

# this to us in.

def getGeometry(model\_name,file,tree\_settings):

start\_time=time.time()

a = ifcopenshell.geom.tree()

a.add\_file(file, tree\_settings)

load\_time = time.time()-start\_time

mod = Model(model\_name,file,a,load\_time)

return mod

# now that we have done all that work above defining classes and functions,

# the code below can be quite simple (it is only 3 lines.

# you can see that it is important to name the functions and classes in a meaningful way to

# contribute to the legibility of your code.

f\_ifc = ifcopenshell.open("model/Duplex\_A\_20110907\_optimized.ifc")

f\_geo = getGeometry('ARCHI',f\_ifc, tree\_settings)

print("\n\t{} Model took {:06.2f} seconds to load".format(f\_geo.name,f\_geo.load\_time))

## Example 5: Compare geometry in different models

This code enables you to load in different models into the same geometry model / tree (line 16). The arch model is added on line 20 and the MECH model is added on line 27. Line 31 and 32 define the Ifc classes that you will use for your clash detection, in this example we are identifying clashes between IfcSpace and IfcFlowSegment as the IfcFlowSegment is something that definitely appears in the MECH model.

For more code examples for detection check out this awesome page

import ifcopenshell

import ifcopenshell.geom

import time

# setup

start\_time=time.time()

tree\_settings = ifcopenshell.geom.settings()

tree\_settings.set(tree\_settings.DISABLE\_OPENING\_SUBTRACTIONS, True)

# this gets the architectural model

a\_ifc = ifcopenshell.open("model/Duplex\_A\_20110907\_optimized.ifc")

# this gets the mechanical model

m\_ifc = ifcopenshell.open("model/Duplex\_M\_20111024\_optimized.ifc")

# this is a sperate geometry model tree

t = ifcopenshell.geom.tree()

start\_time=time.time()

# this adds the architecture geometry to the tree

t.add\_file(a\_ifc, tree\_settings)

load\_time = time.time()-start\_time

print("\n\t{} Model took {:06.2f} seconds to load".format('ARCH',load\_time))

start\_time=time.time()

# this adds the mechanical geometry to the tree

t.add\_file(m\_ifc, tree\_settings)

load\_time = time.time()-start\_time

print("\n\t{} Model took {:06.2f} seconds to load".format('MECH',load\_time))

total\_clashes = 0

obj1 = "IfcSlab"

obj2 = "IfcFlowSegment"

start\_time=time.time()

# ok so in this example I want to take

clashed = a\_ifc.by\_type(obj1)

for clash\_object in clashed:

#print('### {}'.format(space.Name))

for obj in t.select\_box(clash\_object):

if (obj.is\_a(obj2)):

#print ('\t - {}'.format(obj.Name))

total\_clashes+=1

print("\n\tTotal clashes between {} and {} : {:6} ".format(obj1,obj2,total\_clashes))

load\_time = time.time()-start\_time

print("\n\tClash detection took {:06.6f} seconds to complete".format(load\_time))

## Example 5a: Other collision functions to explore

Extend the previous examples with the following commands ….

# extend the bounding box of the object being tested -> wall in this case

t.select\_box(wall, extend=1.)

# extend the bounding box of the object being tested

# and state that it has to be completely inside

t.select\_box(wall, extend=0.001, completely\_within=True)

# get the geometry in t that clash with point 0,0,0

t.select((0.,0.,0.))

# get the geometry in t that intersect a bounding box defined by (-1.,-1.,-1.) -> (1.,1.,1.)

t.select\_box(((-1.,-1.,-1.),(1.,1.,1.)))

# get the t geometry that intersect a bounding box defined by (-2.,-2.,-2.) -> (10.,10.,10.)

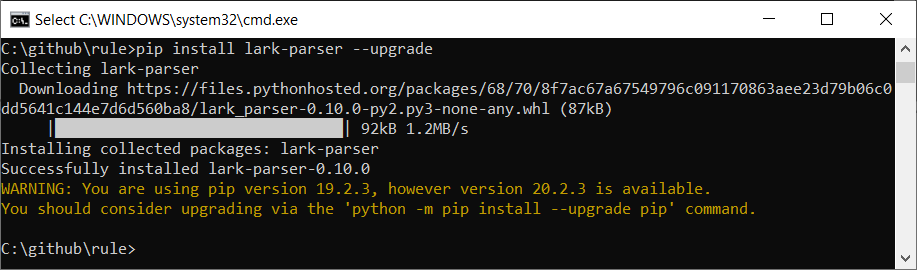
# and is completely contained by it

t.select\_box(((-2.,-2.,-2.),(10.,10.,10.)), completely\_within=True)

## Example 6: Super fast property queries using selector (Currently working on this)

Ok, so this is pretty straight forward, the only thing that might mess this is up, is that selector has a dependency called lark, to install lark you can follow the guide [here](https://github.com/lark-parser/lark). Or if feeling brave, just get the command line open and type:

pip install lark-parser --upgrade



You can look at the code examples here, to see what would be possible with it.

Examples for how to use it:

#import the module and setup

import ifcopenshell.util  
from ifcopenshell.util.selector import Selector  
model = ifcopenshell.open('model\Duplex\_A\_20110907.ifc')  
 selector = Selector()

#this is equivalent to model.by\_type(‘IfcWallStandardCase’)

walls = selector.parse(model, '.IfcWallStandardCase')  
 print(walls)  
 noWalls = len(walls)  
 print("There are {} walls in the model".format(noWalls))

#this is equivalent to searching by GlobalId

wall = selector.parse(model, '.IfcWallStandardCase'[GlobalId = "2O2Fr$t4X7Zf8NOew3FLOH"]')   
 print(wall)

#you can search for elements based on their properties  
#here you find all the external walls

extWalls = selector.parse(model, '.IfcWallStandardCase'[Pset\_WallCommon.IsExternal = "True"]')  
 noExtwalls = len(extWalls)  
 print("There are {} external walls in the model".format(noExtwalls))

#here you find all the spaces on level 1

spaces = selector.parse(model, '.IfcSpace[PSet\_Revit\_Constraints.Level = "Level 1"]')  
 noSpaces = len(spaces)  
 print("There are {} spaces on level 1".format(noSpaces))

#find all walls with a volume above 5

wallsVol = selector.parse(model, '.IfcWallStandardCase[PSet\_Revit\_Dimensions.Volume > "5"]')  
 noWallsVol = len(wallsVol)  
 print("{} walls out of {} walls in the model have a volume above 5".format(noWallsVol,noWalls))

## Example 7a: Check the NUMBER of stories in different models

This example checks to see if different models have the same number of stories.

import ifcopenshell

# this gets the architectural model

a\_ifc = ifcopenshell.open("model/Duplex\_A\_20110907\_optimized.ifc")

# this gets the mechanical model

m\_ifc = ifcopenshell.open("model/Duplex\_M\_20111024\_optimized.ifc")

# Ok so above we loaded 2 models...

a\_stories = a\_ifc.by\_type('IfcBuildingStorey')

m\_stories = m\_ifc.by\_type('IfcBuildingStorey')

**# first we need to check if they have the same number of stories**

**# check to see if the number of stories is the same in the different models...**

if (len(a\_stories) == len(m\_stories)):

print ('\n\tnumber of stories matches')

**# else if that is not true ... is a BIGGER than m ...**

elif (len(a\_stories) > len(m\_stories)):

print ('\n\tmodel\_a has more stories than model\_b')

**# else if that is not true ... is a SMALLLER than m ...**

elif (len(a\_stories) < len(m\_stories)):

print ('\n\tmodel\_a has less stories than model\_b')

## Example 7b: Compare the storey ELEVATIONS in different models

We use a **while** loop here which iterates through the len of a\_stories. This could cause a problem if there are less b\_stories than a\_stories. So we use the **try** command here, this enables us to try a piece of code, and if it doesn’t work it will trigger **except**, and enable us to define an error message to help us debug our program or provide feedback to the user.

import ifcopenshell

# this gets the architectural model

a\_ifc = ifcopenshell.open("model/Duplex\_A\_20110907\_optimized.ifc")

# this gets the mechanical model

m\_ifc = ifcopenshell.open("model/Duplex\_M\_20111024\_optimized.ifc")

# Ok so above we loaded 2 models...

a\_stories = a\_ifc.by\_type('IfcBuildingStorey')

m\_stories = m\_ifc.by\_type('IfcBuildingStorey')

# we are creating a counter here to help us iterate through the stories

count = 0

# we use a while loop here

while count < len(a\_stories):

# try is really cool and stops your code blowing up!

try:

print ('for {} level: a is at {:04.2f} m, and b is at {:04.2f} m'.format(count,a\_stories[count].Elevation,m\_stories[count].Elevation))

# each time this loop runs it adds one to count and gets

# closer to the len of m\_stories

except:

print ('something went wrong')

count+= 1

## Example 7c: Are the ELEVATIONS in diff models the same?

This example is a combination of the logic in 7a and 7b

import ifcopenshell

# this gets the architectural model

a\_ifc = ifcopenshell.open("model/Duplex\_A\_20110907\_optimized.ifc")

# this gets the mechanical model

m\_ifc = ifcopenshell.open("model/Duplex\_M\_20111024\_optimized.ifc")

# Ok so above we loaded 2 models...

a\_stories = a\_ifc.by\_type('IfcBuildingStorey')

m\_stories = m\_ifc.by\_type('IfcBuildingStorey')

# we are creating a counter here to help us iterate through the stories

count = 0

# we use a while loop here

while count < len(a\_stories):

# try is really cool and stops your code blowing up!

try:

# here we grab the elevations and also round them up to 3 decimal places

a\_elev= round(a\_stories[count].Elevation,3)

m\_elev= round(m\_stories[count].Elevation,3)

# we can then print the elevations as we did in example 7A

print ('\n[{} level] a : {:04.3f}m | b : {:04.3f}m'.format(count,a\_elev,m\_elev))

# test if the rounded values are equal

if (a\_elev == m\_elev):

print ('- PASS these match!')

else:

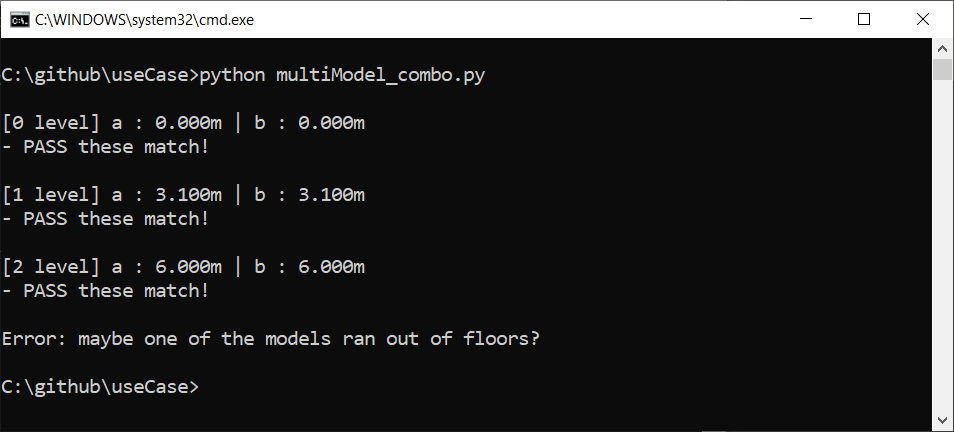
print ('- FAIL these do not match')

except:

print ('\nError: maybe one of the models ran out of floors?')

count+= 1

Example output from 7c - does this match what you got? - why does it error? is the answer in 7a?!?



## Example 8a: Property check

Descirption and comments to follow

import ifcopenshell

model = ifcopenshell.open("model\Duplex\_A\_20110907.ifc")

for entity in model.by\_type("IfcFooting"):

ele\_at\_bottom = False

#we need to get the attributes

for relDefinesByProperties in entity.IsDefinedBy:

for prop in relDefinesByProperties.RelatingPropertyDefinition.HasProperties:

#and then get the attribute we are looking for

if prop.Name == 'Elevation at Bottom':

#add the length to the total length

ele\_at\_bottom = True

if (ele\_at\_bottom):

print ('[X] {}'.format(entity.Name))

else:

print ('[ ] {}'.format(entity.Name))

## Example 8b: Generic Property list

OK so this is my best bet its not perfect but you should be able to adapt it to your needs and it works for windows and doors 😊

import ifcopenshell

model = ifcopenshell.open("model\Duplex\_A\_20110907.ifc")

print ('\n## search all properties of a type ##\n')

# lets search all property sets

for pset in model.by\_type("IfcPropertySet"):

# and all single values within those property sets (that have props)

for prop in pset.HasProperties:

# and check if that property matches the one we are looking for...

if prop.is\_a("IfcPropertySingleValue"):

obj = model.get\_inverse(pset)

# ok cool so now get the related objects

# so then we get the entity related to that property

for part in obj[0].RelatedObjects:

# then we check if the entity is a window

if (part.is\_a('IfcWindow')):

# print the property sets that you have found

print ('{} : {}'.format(prop.Name,prop.NominalValue.wrappedValue))

## Example 8c: Find entities based on a singlevalue property

I think this one is pretty cool, I was trying to write an example for 8b and ended up with this. The logic is that we define the property and value we are looking for , get it and then seeks its inverse entities.

import ifcopenshell

model = ifcopenshell.open(" model\Duplex\_A\_20110907.ifc")

propName ='Unbounded Height'

value= 2.6

print ('\n## search entity by property and value ##\n')

print ('Searching for...')

print ('\tIfcPropertySingleValue = [ {} : {} ]'.format(propName,value))

# lets search all property sets

count = 0

print ('Results:')

for pset in model.by\_type("IfcPropertySet"):

# and all single values within those property sets (that have props)

for prop in pset.HasProperties:

# and check if that property matches the one we are looking for...

if prop.is\_a("IfcPropertySingleValue") and prop.Name == propName and prop.NominalValue.wrappedValue == value:

# then get the objects inverse relations

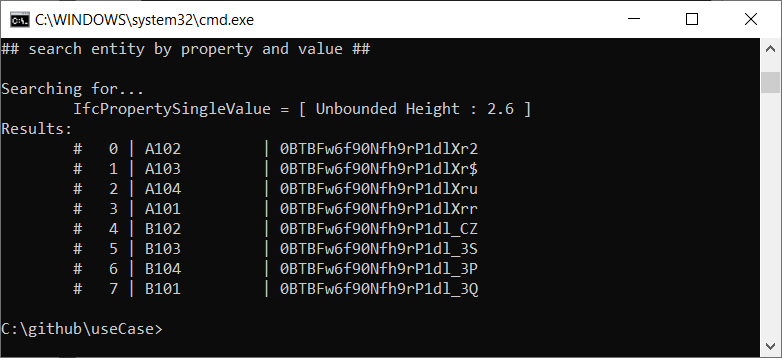
obj = model.get\_inverse(pset)

# ok cool so now get the related object …

for part in obj[0].RelatedObjects:

print('\t# {:3} | {:12} | {}'.format(count,part.Name,part.GlobalId))

count+=1



## 

## Example 9a: Door code check

This is an edit of Kallina’s door code check, its a good example.

import ifcopenshell

model = ifcopenshell.open("model\Duplex\_A\_20110907.ifc")

###Doors###

doors\_required = 14 ### <- Expected value of doors ###

doors\_in\_model = len(model.by\_type("IfcDoor"))

min\_width\_door = 0.77

valid\_doors=0

invalid\_doors=0

print ('\n')

# initial check to establish if we have the 'correct' number of doors

if doors\_required == doors\_in\_model:

print("Result matches expected value ({})".format(doors\_required))

elif doors\_required > doors\_in\_model:

print("There are more doors than expected")

elif doors\_required < doors\_in\_model:

print("There are less doors than expected")

print ('\n')

# check each door to see if it complies and count the valid ones

for door in model.by\_type("IfcDoor"):

print ("door with width: "+str(door.OverallWidth))

if door.OverallWidth>=min\_width\_door:

valid\_doors+=1

# now we have finished the counting we can pull back the indents and print the result

print("\nThe width of {} doors is according to the Danish Regulations".format(valid\_doors))

print("The width of {} doors is not according to the Danish Regulations".format(doors\_in\_model-valid\_doors))

