Engineering Sketch Pad (ESP)

User Training Session 6
Pyscript Fundamentals

John F. Dannenhoffer, III
john@geocentrictech.com
Geocentric Technologies LLC

updated for v1.28

Overview

- Introduction
- `pyOCSM` Overview
- `pyESP` Overview
- Pyscript `example`
- Plotter
- Homework Exercise

Introduction to Pyscript

- Embedded Python interpeter
- Has access to `OpenCSM` via `pyOCSM`
- Has access to `EGADS` via `pyEGADS`
- Context-sensitive editor
- Reports Python's output in MessageWindow
- Reports Python's errors in a pop-up dialog and turns the MessageWindow yellow
  - double-clicking in yellow MessageWindow opens Pyscript editor to the offending line
- Can be launched:
  - directly from the `serveESP` command line
  - choosing **Tool→Pyscript**

pyOCSM and pyESP Overview

- pyOCSM
  - Python bindings for OpenCSM
    - one-to-one mapping with the OpenCSM API
- pyESP
  - Functions that allow pyOCSM to be linked with ESP

pyOCSM Methods (1)

```
Build(self, buildTo, nbody)
    ocsm.Build - build Bodys by executing the MODL up to a given Branch

Check(self)
    ocsm.Check - check that Branches are properly ordered

Clearance(self, ibody1, ibody2)
    ocsm.Clearance - compute clearance between Bodys

Copy(self)
    ocsm.Copy - copy a MODL (but not the Bodys)

DelBrch(self, ibrch)
    ocsm.DelBrch - delete a Branch (or whole Sketch if SKBEG)

DelPmtr(self, ipmtr)
    ocsm.DelPmtr = delete a Parameter

EvalExpr(self, express)
    ocsm.EvalExpr - evaluate an expression

FindEnt(self, ibody, seltype, entID)
    ocsm.FindEnt - find entity number given faceID or edgeID
```

pyOCSM Methods (2)

```
FindPmtr(self, name, type, nrow, ncol)
    ocsm.FindPmtr - find (or create) a Parameter

Free(self)
    ocsm.Free - free up all strorage associated with a MODL

GetArg(self, ibrch, iarg)
    ocsm.GetArg - get an Argument for a Branch

GetAttr(self, ibrch, aname)
    ocsm.RetAttr - return an Attribute for a Branch by name

GetBnds(self, ipmtr, irow, icol)
    ocsm.GetBnds - get the Bounds of a Parameter

GetBody(self, ibody)
    ocsm.GetBody - get info about a Body

GetBrch(self, ibrch)
    ocsm.GetBrch - get info about a Branch

GetCode(self, text)
    ocsm.GetCode - convert text to an OCSM numeric code
```

pyOCSM Methods (3)

```
GetCsys(self, ibrch, cname)
    ocsm.GetCsys - return a Csystem for a Branch by name

GetEgo(self, ibody, seltype, iselect)
    ocsm.GetEgo - get the EGO associated with the Body or its parts

GetFilelist(self)
    ocsm.GetFilelist - get a list of all .csm, .cpc, and .udc files

GetName(self, ibrch)
    ocsm.GetName - get the name of a Branch

GetNorm(self, ibody, iface, npnt, uv)
    ocsm.GetNorm - get the unit normals for a Face

GetPmtr(self, ipmtr)
    ocsm.GetPmtr - get info about a Parameter

GetSketch(self, ibrch, maxlen)
    ocsm.GetSketch - get string data associated with a Sketch

GetTessNpnt(self, ibody, seltype, iselect)
    ocsm.GetTessNpnt - get the number of tess points in an Edge or Face
```

pyOCSM Methods (4)

```
GetTessVel(self, ibody, seltype, iselect)
    ocsm.GetTessVel - get the tessellation velocities on a Node, Edge, or Face

GetText(self, code)
    ocsm.GetText - convert an OCSM numeric code to text

GetUV(self, ibody, seltype, iselect, npnt, xyz)
    ocsm.GetUV - get the parametric coordinates on an Edge or Face

GetValu(self, ipmtr, irow, icol)
    ocsm.GetValue - get the Value of a Parameter

GetValuS(self, ipmtr)
    ocsm.GetValue - get the Value of a string Parameter

GetVel(self, ibody, seltype, iselect, npnt, uv)
    ocsm.GetVel - get the velocities on a Node, Edge, or Face

GetXYZ(self, ibody, seltype, iselect, npnt, uv)
    ocsm.GetXYZ - get the coordinates of a Node, Edge, or Face

Info(self)
    ocsm.Info - get info about a MODL
```

pyOCSM Methods (5)

```
Load(self, filename)
    ocsm.Load - create a MODL by using __init__ method

LoadDict(self, dictname)
    ocsm.LoadDist - load dictionary from dictname

NewBrch(self, iafter, type, filename, linenum, arg1, arg2, arg3, arg4, arg5, arg6,
    ocsm.NewBrch - create a new Branch

NewPmtr(self, name, type, nrow, ncol)
    ocsm.NewPmtr - create a new Parameter

Perturb(self, npmtrs, ipmtrs, irows, icols, values)
    ocsm.Perturb - create a perturbed MODL

PrintAttrs(self, filename)
    ocsm.PrintAttrs - print global Attributes to file

PrintBodys(self, filename)
    ocsm.PrintBodys - print all Bodys to file

PrintBrchs(self, filename)
    ocsm.PrintBrchs - print Branches to a file
```

pyOCSM Methods (6)

```
PrintBrep(self, ibody, filename)
    ocsm.PrintBrep - print the BRep associated with a specific Body

PrintPmtrs(self, filename)
    ocsm.PrintPmtrs - print DESPMTRs and OUTPMTRs to file

RegMesgCB(self, callback)
    ocsm.RegMesgCB - register a callback function for exporting messages

RegSizeCB(self, callback)
    ocsm.RegSizeCB - register a callback function for DESPMTR size changes

RetAttr(self, ibrch, iattr)
    ocsm.RetAttr - return an Attribute for a Branch by index

RetCsys(self, ibrch, icsys)
    ocsm.RetCsys - return a Csystem for a Branch by index

Save(self, filename)
    ocsm.Save - save a MODL to a file

SaveDespmtrs(self, filename)
    ocsm.SaveDespmtrs - save CFGPMTRs and DESPMTRs to file
```

pyOCSM Methods (7)

```
SaveSketch(self, ibrch, vars, cons, segs)
    ocsm.SaveSketch - overwrite Branches associated with a Sketch

SetArg(self, ibrch, iarg, defn)
    ocsm.SetArg - set an Argument for a Branch

SetAttr(self, ibrch, aname, avalue)
    ocsm.SetAttr - set an Attribute for a Branch

SetBnds(self, ipmtr, irow, icol, lbound, ubound)
    ocsm.SetBnds - set teh Bounds of a Parameter

SetBrch(self, ibrch, actv)
    ocsm.SetBrch - set activity for a Branch

SetCsys(self, ibrch, cname, cvalue)
    ocsm.SetCsys - set a Csystem for a Branch

SetDtime(self, dtime)
    ocsm.SetDtime - set sensitivity FD time step (or select analytic)

SetEgg(self, eggname)
    ocsm.SetEgg - set up alternative tessellation by an external grid generator
```

pyOCSM Methods (8)

```
SetEgo(self, ibody, iselect, theEgo)
    ocsm.SetEgo - set a tessellation or EBody for a Body

SetName(self, ibrch, name)
    ocsm.SetName - set the name for a Branch

SetValu(self, ipmtr, irow, icol, defn)
    ocsm.SetValu - set a Value for a Parameter

SetValuD(self, ipmtr, irow, icol, value)
    ocsm.SetValuD - set the (double) value of a Parameter

SetVel(self, ipmtr, irow, icol, defn)
    ocsm.SetVel - set the velocity for a Parameter

SetVelD(self, ipmtr, irow, icol, dot)
    ocsm.SetVelD - set the (double) velocity for a Parameter

SolveSketch(self, vars_in, cons, vars_out)
    ocsm.SolveSketch - solve for new Sketch variables

UpdateDespmtrs(self, filename)
    ocsm.UpdateDespmtrs - update CFGPMTRs and DESPMTRs from filename
```

pyOCSM Methods & Functions (9)

```
UpdateTess(self, ibody, filename)
    ocsm.UpdateTess - update a tessellation from a file

SetOutLevel(ilevel)
    ocsm.SetOutLevel - set output level

Version()
    ocsm.Version - return current version
```

pyESP Functions (1)

```
GetCaps(esp)
    GetCaps - get serveESP's active CAPS

GetEsp(timName)
    ocsm.GetEsp - get the ESP structure

GetModl(esp)
    GetModl - get serveESP's active MODL

SetCaps(problem, esp)
    SetCaps - set serveESP's active CAPS

SetModl(modl, esp)
    SetModl - set serveESP's active MODL

TimBcst(timName, text)
    ocsm.TimBcst - broadcasts a message to all browsers

TimHold(timName, overlay)
    ocsm.TimHold - hold timName until overlay finished

TimLoad(timName, esp, data)
    ocsm.TimLoad - load a TIM
```

pyESP Functions (2)

```
TimMesg(timName, mesg)
    ocsm.TimMesg - sends a message to a TIM

TimQuit(timName)
    ocsm.TimQuit - quits a TIM

TimSave(timName)
    ocsm.TimSave - saves data and quits a TIM

UpdateESP()
    UpdateESP - update ESP after rebuilding

ViewModl(modl)
    ViewModl - make modl active and view it in serveESP
```

Simple `example`

- Configuration consists of BOX and CYLINDER
- CYLINDER length is a DESPMTR
- Generate a plot of volume vs. cylinder length and show intermediate configurations

example.csm

```
# example
# written by John Dannenhoffer

DESPMTR   myLen   0.1         # length of cylinder
OUTPMTR   myVol               # volume of final configuration

# make the configuration
BOX       -1  -1  -1  2  2  2
CYLINDER  -myLen/2 0 1  +myLen/2 0 1  0.25
UNION

# set the output parameter
SET       myVol  @volume

END
```

example.py (1)

```
###################################################################
#                                                                 #
# example.py --- start with either of the following:              #
#                      python example.py       (shows nothing)    #
#                         example.csm                             #
#                      serveESP example                           #
#                         Tool->Pyscript  example                 #
#                                                                 #
#             Written by John Dannenhoffer @ Syracuse University  #
#                                                                 #
###################################################################

# import the appropriate packages
from pyEGADS import egads
from pyOCSM  import ocsm
from pyOCSM  import esp

import os
```

example.py (2)

```
# callback functions
def pyMesgCB(text):
    print(" ")
    print("======= in pyMesgCB =======")
    print("   ", text.decode())
    print("============================")
    return

def pySizeCB(modl, ipmtr, nrow, ncol):
    print(" ")
    print("======= in pySizeCB =======")
    print("    ipmtr:", ipmtr)
    print("    nrow :", nrow )
    print("    ncol :", ncol )
    print("============================")
    return

# make a semi-colon-separated string from a list
def makeString(array):
    out = ""
    for i in array:
        out += str(i) + ";"
    return out
```

example.py (3)

```
# run quietly
ocsm.SetOutLevel(0)

# if we are running via serveESP, link to that MODL
try:
    modl = ocsm.Ocsm(esp.GetModl(esp.GetEsp("pyscript")))
    modl.RegMesgCB(pyMesgCB)
    modl.RegSizeCB(pySizeCB)
    print("==> getting MODL from ESP")

# an error means that we are probably running from the python prompt,
#    so get the filename from the user to create a new MODL
except ocsm.OcsmError:
    filename = ""
    while (".csm" not in filename):
        filename = input("Enter name of .csm file: ")
        if (not os.path.exists(filename)):
            print("\""+filename+"\" does not exist")
            filename = ""
    modl = ocsm.Ocsm(filename)
    modl.RegMesgCB(pyMesgCB)
    modl.RegSizeCB(pySizeCB)
    print("==> making new MODL from \""+filename+"\"")
```

example.py (4)

```
# check the MODL
modl.Check()

# we need a build so that we can get the parameter indices
modl.Build(0, 0)

# find the indicies of ESP's parameters
ilen = modl.FindPmtr("myLen", 0, 0, 0)
ivol = modl.FindPmtr("myVol", 0, 0, 0)

# load the viewer (so that we can see the configuration)
esp.TimLoad("viewer", esp.GetEsp("pyscript"), "")

# show the original configuration and wait for
#    user to press ExitViewer
esp.TimMesg("viewer", "MODL")

# initialize the list in which data to be plotted will be stored
myLen = []
myVol = []
```

example.py (5)

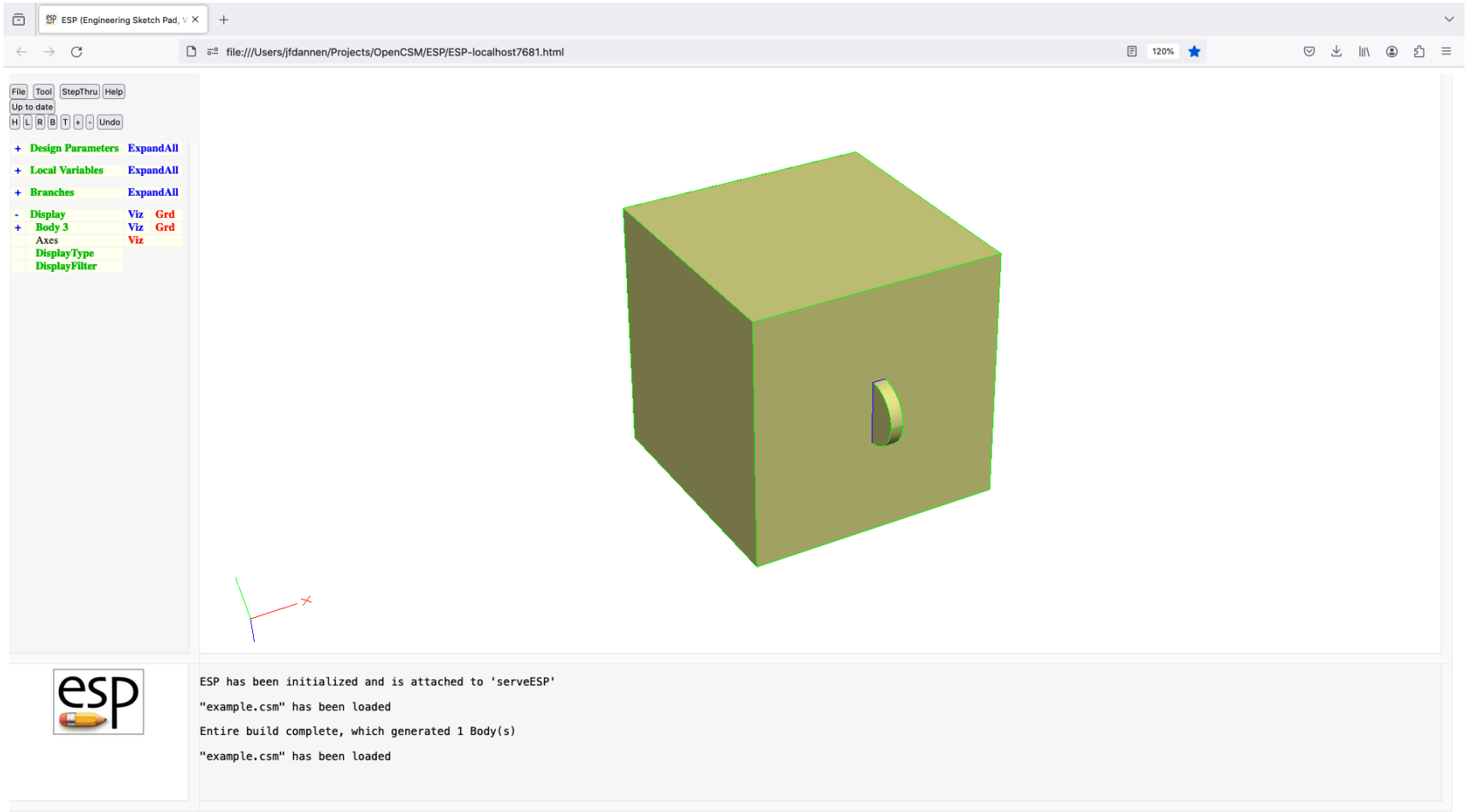
```
# loop through successively larger cylinders
for len in [0.5, 1.0, 1.5, 2.001, 2.5, 3.0]:

    # set the length
    modl.SetValuD(ilen, 1, 1, len)

    # build the model
    modl.Build(0, 0)

    # show the current configuration and wait to user to ExitViewer
    esp.TimMesg("viewer", "MODL")

    # add the data for this case to the data to be plotted
    myLen.append(modl.GetValu(ilen, 1, 0)[0])
    myVol.append(modl.GetValu(ivol, 1, 0)[0])
```

example.py (6)

```
# quit the viewer so that we can use the plotter
esp.TimQuit("viewer")

# load the plotter (so that can see a line plot)
esp.TimLoad("plotter", esp.GetEsp("pyscript"), "")

# new plot with given axis labels
esp.TimMesg("plotter", "new|example|len|vol|")

# add the line to the plot (solid black line with red + symbols)
esp.TimMesg("plotter", "add|"+makeString(myLen)+"|"+makeString(myVol)+"|k-|")
esp.TimMesg("plotter", "add|"+makeString(myLen)+"|"+makeString(myVol)+"|r+|")

# show the plot
esp.TimMesg("plotter", "show")

# exit the plotter when user presses ExitPlotter
esp.TimQuit("plotter")
```

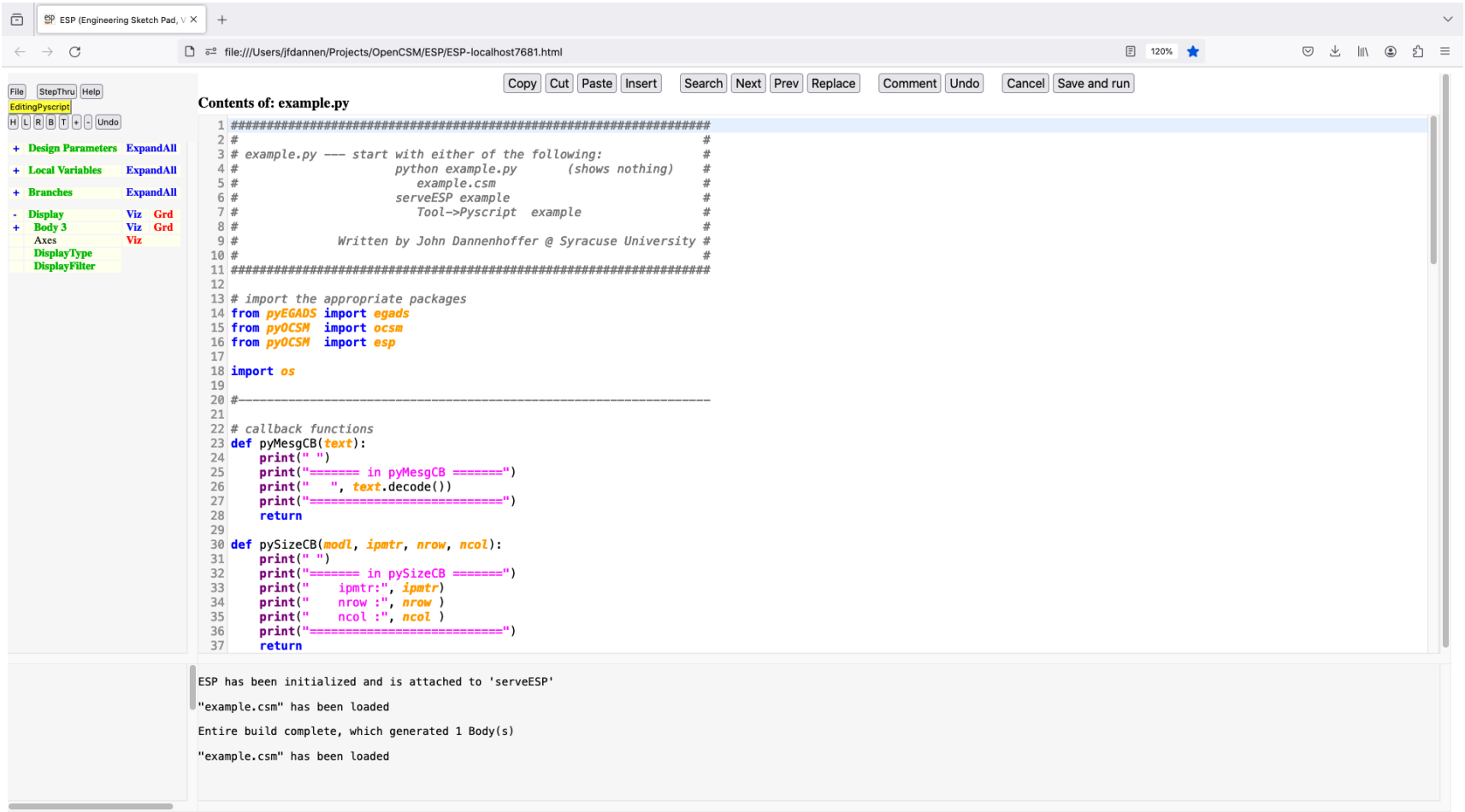
example Screenshot (1)

Body before Pyscript starts



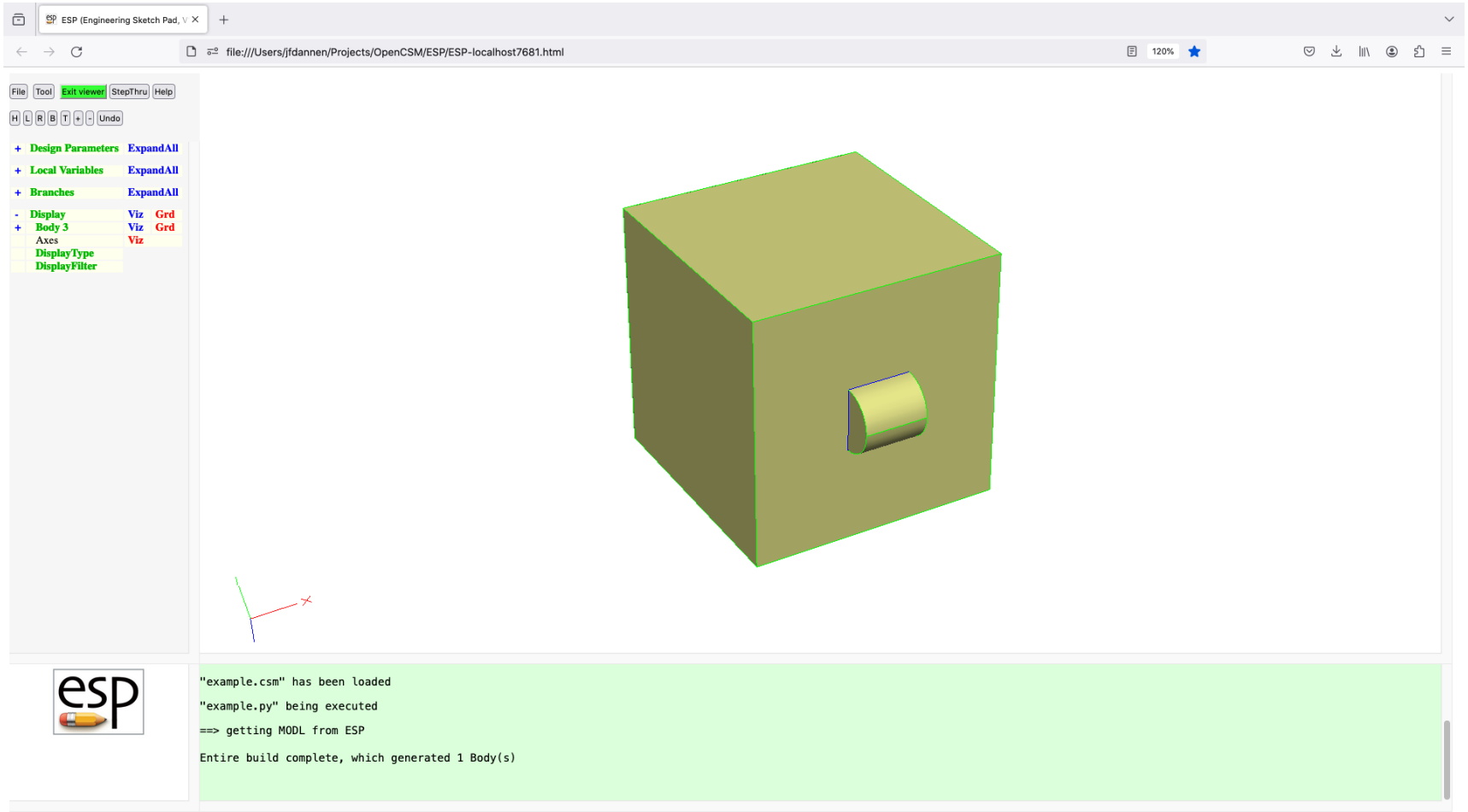
example Screenshot (2)

Python script to modify the Model



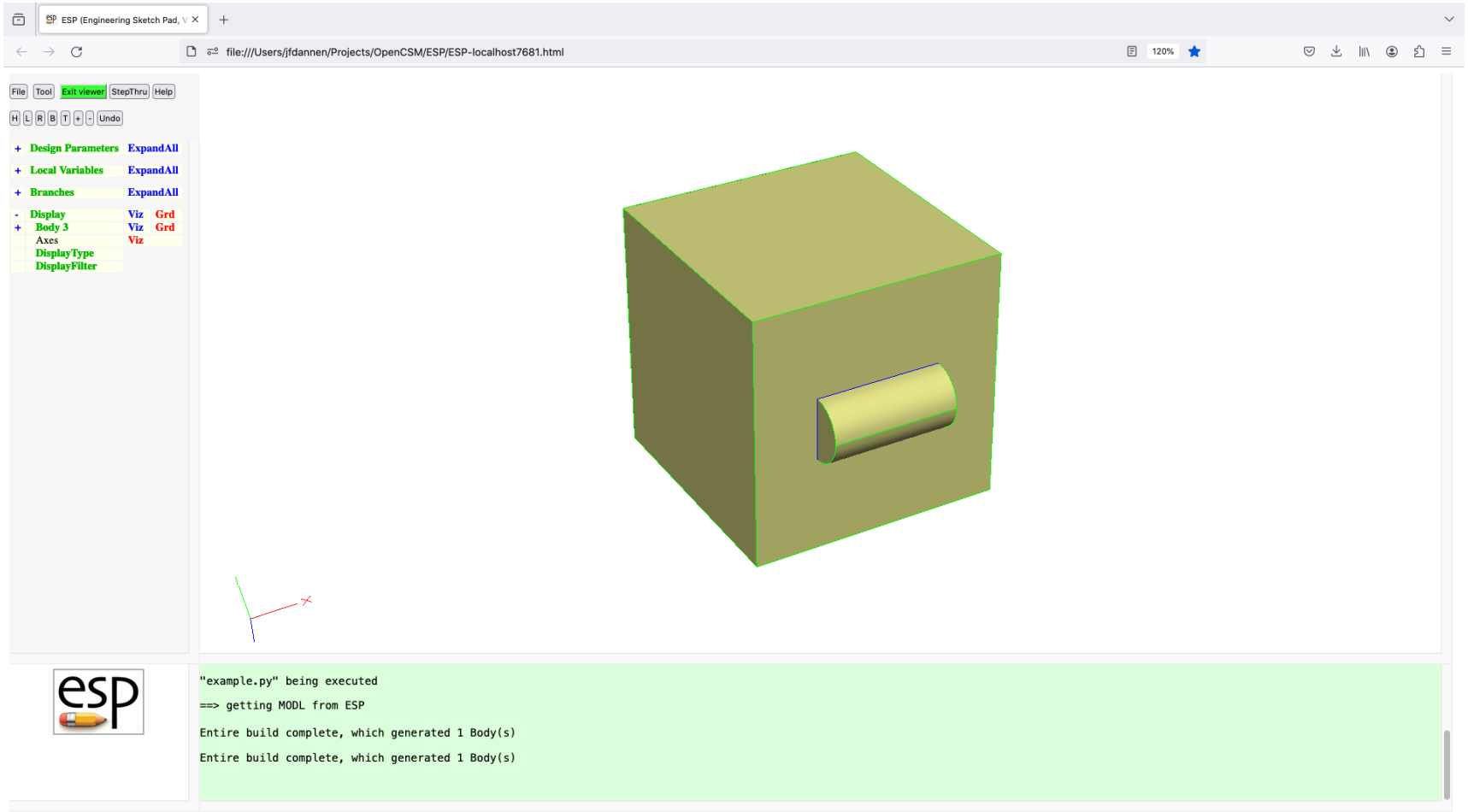
example Screenshot (3)

Body after first modification



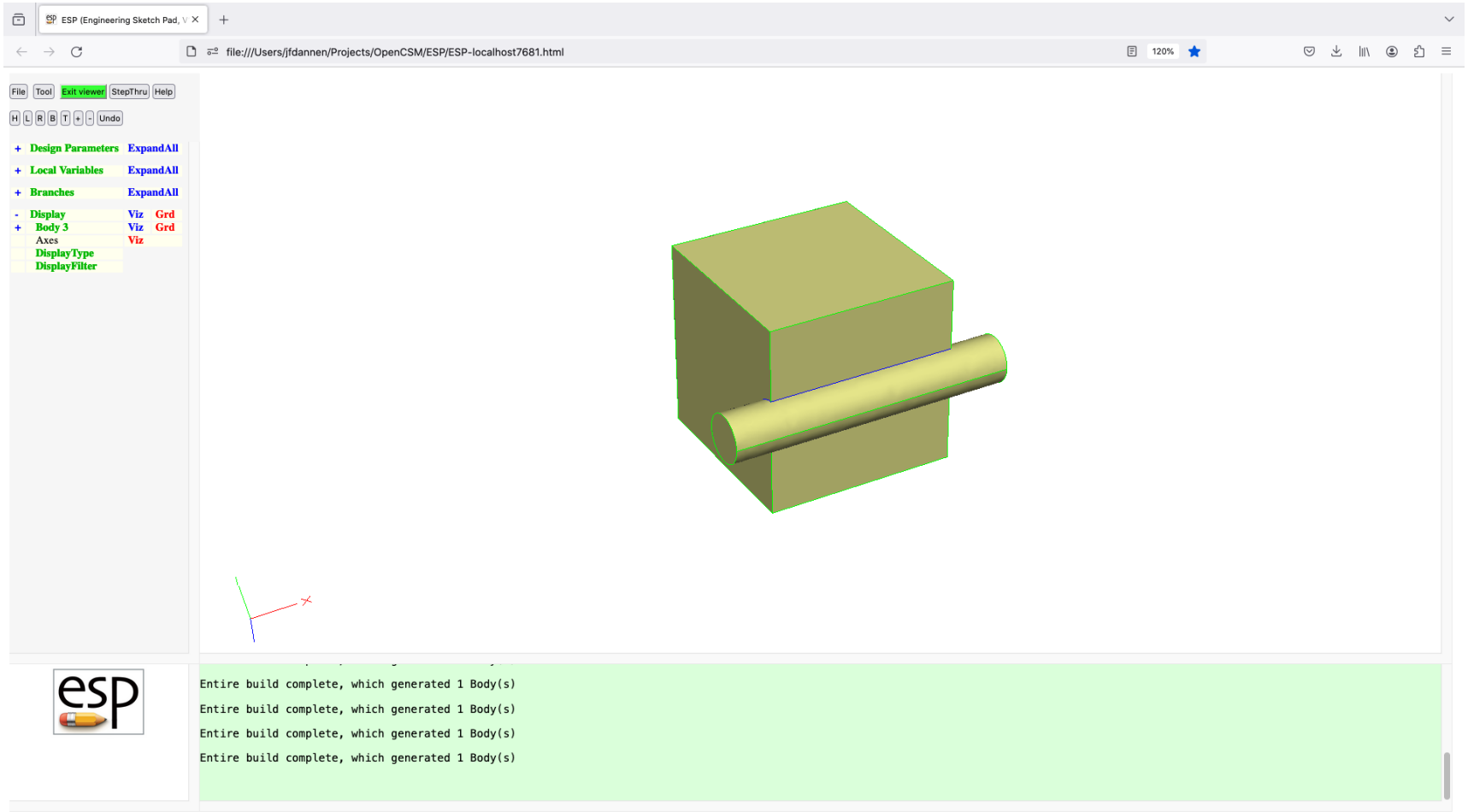
example Screenshot (4)

Body after second modification



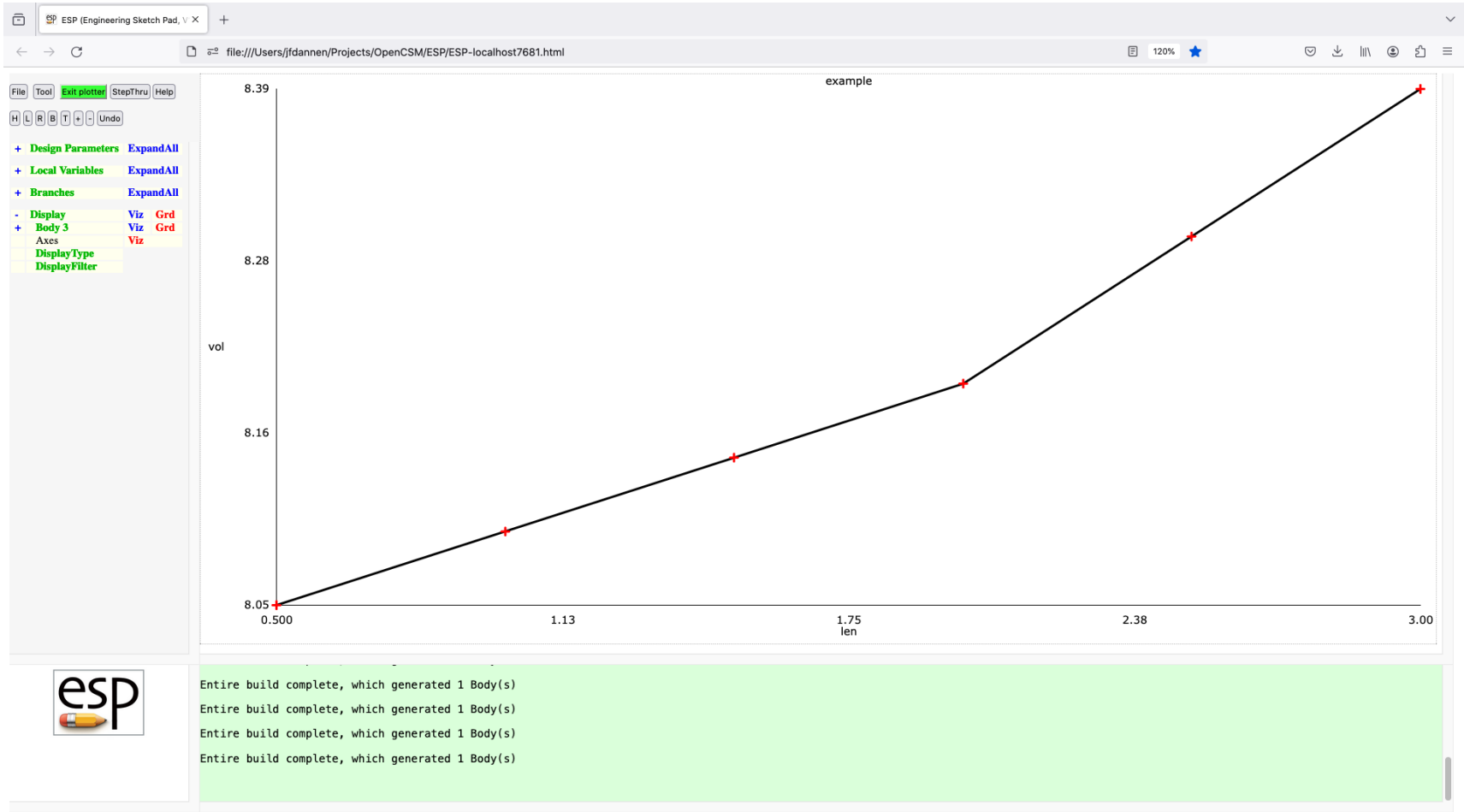
example Screenshot (5)

Body after final modification



example Screenshot (6)

Line plot of vol vs. len



Plotter

```
timLoad|plotter|

timMesg|plotter|new|title|xlabel|ylabel|ylabel2|

timMesg|plotter|add|xvalue1;xvalue2;...|yvalue1;yvalue2;...|type|

type:
  r red       - solid      o circle        2 ylabel2
  g green     : dotted     x x-mark
  b blue      _ dashed     + plus
  c cyan      ; dot-dash   * star
  m magenta                s square
  y yellow                 ^ triangle-up
  k black                  v triangle-down
  w white

timMesg|plotter|show|
timMesg|plotter|show|nohold|
```

Homework Exercise

- Start with vehicle in `$ESP_ROOT/training/exercises/session06/areaRule.csm`
- Write a Pyscript to adjust the entries in `fuse:radius` so that the `fuse:xsect` satisfies the Sears-Haack area rule

$$A_{\text{sears}}(x) = A_{\max}\left[4\frac{x}{x_{\max}}\left(1-\frac{x}{x_{\max}}\right)\right]^{3/2}$$

- Plot the actual and Sears-Haack areas as a function of x (after each iteration)
- Hint: you can use the update formula

$$r_{\texttt{new}}(x) = r_{\texttt{old}}(x)\sqrt{\frac{A_{\text{sears}}(x)}{A_{\text{xsect}}(x)}}$$