UNIVERSITY OF ZAGREB

Faculty of Electrical Engineering and Computing

pySolidWorks

A Python Interface to Solidworks

Mario Klanac

Version 0.1

2021

Contents

CHAPTER 1

Introduction

sec:intro

pySolidWorks is a collection of Python routines to interact with SolidWorks. Using pySolidWorks, you can open a SolidWorks model from within Python, modify feature dimensions, export DXF, carry out a finite element analysis, etc. pySolidWorks has been tested on Windows 10, SolidWorks 2019, Python 3.7.

For questions and feedback, please send an email to mario.klanac1995@gmail.com

1.1 Install and run example

You can install pySolidWorks using pip:

```
Microsoft Windows [Version 10.0.19041.928]
(c) Microsoft Corporation. All rights reserved.

c:\User\Mario>pip install pysolidworks
```

Try running this code, to check that everything was installed correctly:

```python
from pysolidworks import Solidworks
sw = Solidworks()
sw.OpenDoc(r'D:\Github\pySolid\manual\BoxWithHole.SLDPRT')
```

1.2 Outline

The rest of the text is organised as follows:

Chapter 2 explains scripts located in **examples** folder, so you can see the capabilities of this package.

Chapter 3 provides step by step introduction to pySolidWorks.

Chapter 4 doing optimization using automated SolidWorks Simulation.

Chapter 5 explains how to automate anything in Solidworks using Python.

Appendix B TODO.

CHAPTER 2

pySolidWorks examples

sec:second

Now the connection between Python and SolidWorks is established, you are now ready to run some examples.

2.1 Basic

In Basics demo, you will be able to carry out the following examples.

- Open TestPart.py if it is not open already
- Save the image of currently opened SolidWorks Part as Part.pdf
- Suppress a feature "SideHole" in TestPart.SLDPRT
- Unsuppress a feature "SideHole" in TestPart.SLDPRT
- Save the SolidWorks Part which is currently opened
- Create a new part, draw a pre-defined Sketch and extrude
- Insert a reference plane that is 10 units offset from the Front plane

Run “Basics.py” file and perform the above examples.

2.2 Advanced

In Advanced demo, you will be able to execute the following examples.

- Open TestPart.SLDPRT, compute mass properties, modify dimensions,recompute mass properties and revert back
- Open TestPart.SLDPRT, compute mass properties, modify dimensions,recompute mass properties and revert back
- Open TestPart.SLDPRT and display its Sketch1 of MainBody in Matlab as 2D figure
- Open TestPart.SLDPRT, create mesh and dumps it into a text file
- Finite Element Analysis on TestPart.SLDPRT

- Open testRevolve.SLDPRT and create a revolved feature

Run “Advanced.py” file in the pySolidWorks folder and perform the above examples.

CHAPTER 3

Getting started with pySolidWorks

sec:third

3.1 Opening a SolidWorks part

The first function introduced is the “OpenDoc” function. With the SolidWorks opened and pysolidworks initialized using this code:

```
from pysolidworks import Solidworks
sw = Solidworks()
```

the user can open a SolidWorks Part by calling this function:

```
sw.OpenDoc(r'full path')
```

In this function call, the r’full path’ is the full path of the part that user wants to open. If the user does not provide the full path, the function will look for the part in the current working directory.

3.2 Creating a SolidWorks Part

The “NewDoc” function enables the userto create a new part file with a specified name:

```
sw.NewDoc(modelPath, docType)
```

The ‘modelPath’ is the full path including the name of the part that the user wants to create with the name extension “.SLDPRT”. If the full path is not provided, the default directory is the current working directory.

3.3 Save a SolidWorks Part

To save the current part, we can call “SaveDoc” function:

```
sw.SaveDoc()
```

this function saves all the changes in current active part, assuming a part file has already been created. However, if the part needs to be saved in other forms than “.SLDPRT” or saved under other directories, “SaveDocAs” can be used:

```
sw.SaveDocAs(modelPath)
```

3.4 Querying Dimensions

In this section, we illustrate how the part parameters can be obtained using pySolidWorks. Using the part “BoxWithHole.SLDPRT” shown on figure 3.1, one can call the “GetValueOfDimension” function:

```
value = s.GetValueOfDimension(dimName)
```

where dimName is the name of dimension that can be found out by clicking on dimension, for example, if one clicks on dimensions that define box height D1 (to show this tag, select View⟶Hide/Show⟶Dimension Names) dimName should be:

```
dimName = 'D1@Sketch2'
```

The ‘D1’ is the auto-assigned tag of the dimension. The ‘Sketch2’ indicates the name of the sketch in which the dimension is defined. On figure 3.1 we have three D1 dimensions, one is defined on ’Sketch2’, second on ’Sketch3’, and the third one defines extrusion depth of feature called ’Box’.

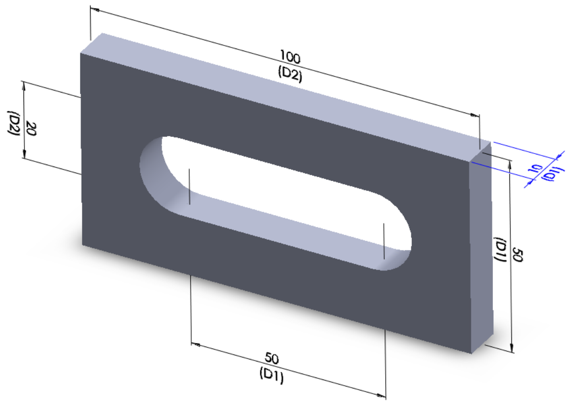

Figure 3.1: Dimensions associated with the opened part

figure: dimensions

Querying All Dimensions

If dimName is not given as input, this function isexpected to return the names andvalues of all the dimensions as indicated in Figure8:

Dimensions associated the MainBody feature.

Dimensions associated with the SideHole feature. Figure 8: Dimensions associated with TestPart.

The dimNames are formatted as the following: using’D3@Sketch1@TestPart.Part’asan example, the ‘D3’ is the auto-assigned tag of the dimension (To show this tag, selectView→Dimension Names). The ‘Sketch1’ indicates the name of the sketch inwhich thedimension is defined. Then, the ‘TestPart’ is the name of the file. While the dimValues corresponds to the values of the assigned dimensions in the unitsof the current model(To change the units within SolidWorks, use Tools→Options→Document Properties→Units.):

It should be noticed that only the dimensions assigned within the part are returned.Thus, for example, if a sketch is not dimensioned,then none of the (implicit) dimensions is returned.

f the variable dimName is included in the argument, then only the dimensionscorresponding to the input dimension name is returned. The dimName here can beeither one string or a cell of strings, and the strings must be in the form “dimensionNumber@featureName@fileName.Part”.

CHAPTER 4

Model Optimzation using FEM

sec:fourth

TODO

4.1 Creating Study

4.2 Applying materials

4.3 Creating mesh

4.4 Solving Study

4.5 Getting results

CHAPTER 5

The Fifth Chapter

sec:fifth

TODO

Appendices

APPENDIX A

The First Appendix

sec:first-app

TODO

APPENDIX B

The Second Appendix

sec:second-app

TODO