Ring Pull Strain Analysis (RPSA) Tutorial

Introduction

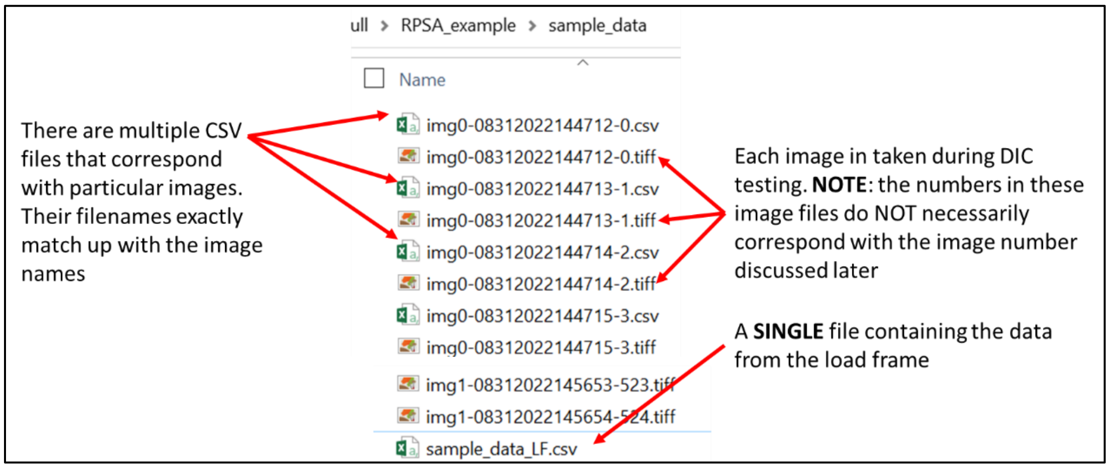
This tutorial is intended for a novice user to be able to run the Ring Pull Strain Analysis program.

Getting Started

Please download Python from anaconda.org. There are multiple ways to download Python, however, this one contains all the necessary modules. Features of Anaconda will be displayed heavily in this tutorial.

File Structure

The files that are needed to run this program are shown below


ull > RPSA_example > sample_data
Name
img0-08312022144712-0.csv
img0-08312022144712-0.tiff
img0-08312022144713-1.csv
img0-08312022144713-1.tiff
img0-08312022144714-2.csv
img0-08312022144714-2.tiff
img0-08312022144715-3.csv
img0-08312022144715-3.tiff
img1-08312022145653-523.tiff
img1-08312022145654-524.tiff
sample_data_LF.csv

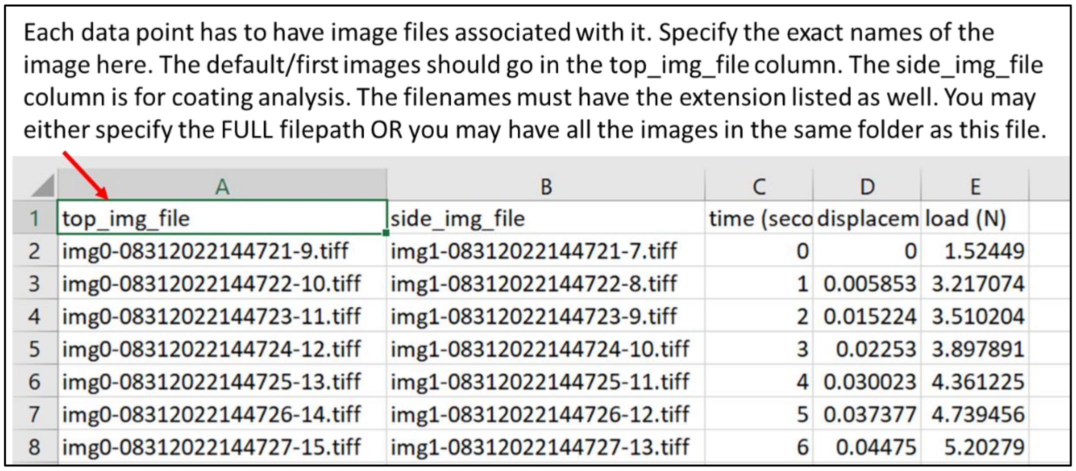
There are multiple CSV files that correspond with particular images. Their filenames exactly match up with the image names

Each image in taken during DIC testing. **NOTE**: the numbers in these image files do NOT necessarily correspond with the image number discussed later

A **SINGLE** file containing the data from the load frame

Below are some images showing the type of data present in each file. Note that the .csv files must have specific headers.

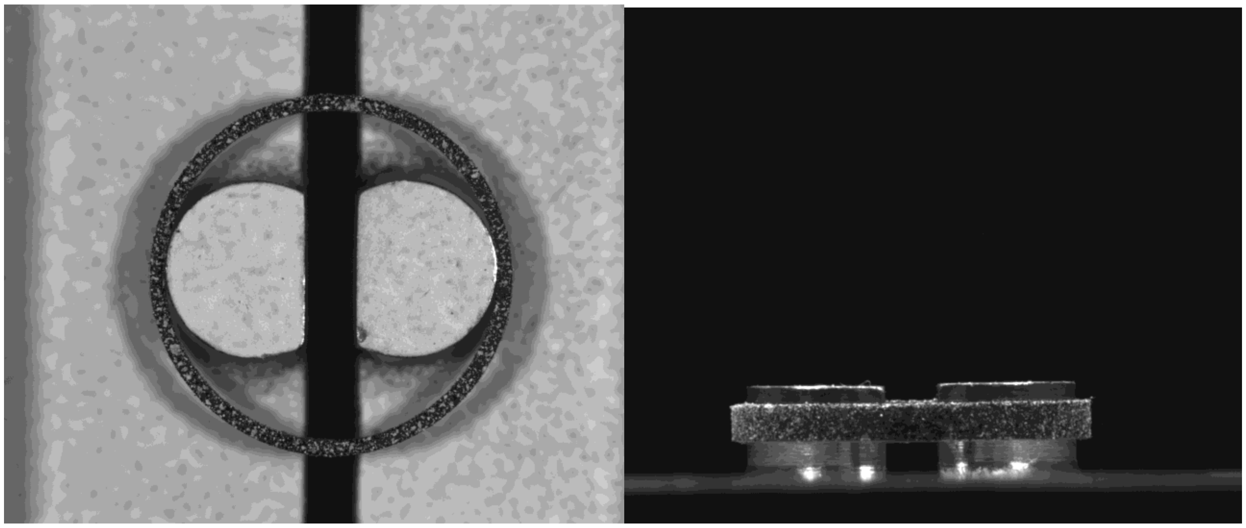
Load Frame File:



Each data point has to have image files associated with it. Specify the exact names of the image here. The default/first images should go in the top_img_file column. The side_img_file column is for coating analysis. The filenames must have the extension listed as well. You may either specify the FULL filepath OR you may have all the images in the same folder as this file.

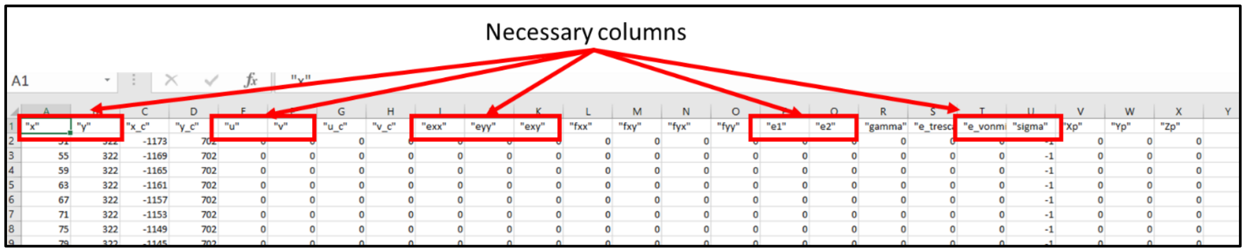
	A	B	C	D	E
1	top_img_file	side_img_file	time (seco	displacem	load (N)
2	img0-08312022144721-9.tiff	img1-08312022144721-7.tiff	0	0	1.52449
3	img0-08312022144722-10.tiff	img1-08312022144722-8.tiff	1	0.005853	3.217074
4	img0-08312022144723-11.tiff	img1-08312022144723-9.tiff	2	0.015224	3.510204
5	img0-08312022144724-12.tiff	img1-08312022144724-10.tiff	3	0.02253	3.897891
6	img0-08312022144725-13.tiff	img1-08312022144725-11.tiff	4	0.030023	4.361225
7	img0-08312022144726-14.tiff	img1-08312022144726-12.tiff	5	0.037377	4.739456
8	img0-08312022144727-15.tiff	img1-08312022144727-13.tiff	6	0.04475	5.20279

Image Files:



csv files:

- These files are most often generated from a third party digital image correlation software. Supported file modes are VIC-2D 6, VIC-2D 7, and DICengine.



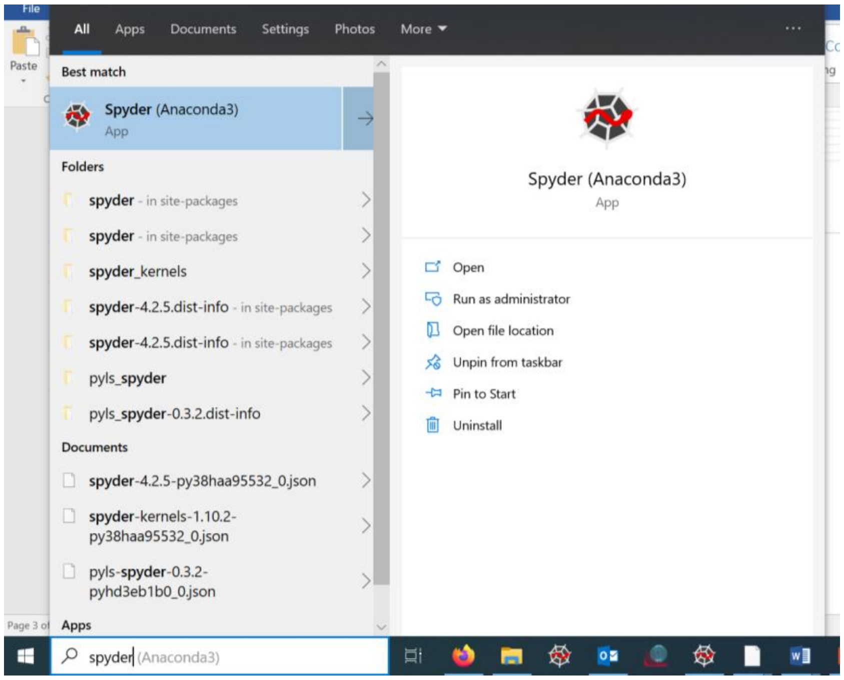
Necessary columns

A1 "x"

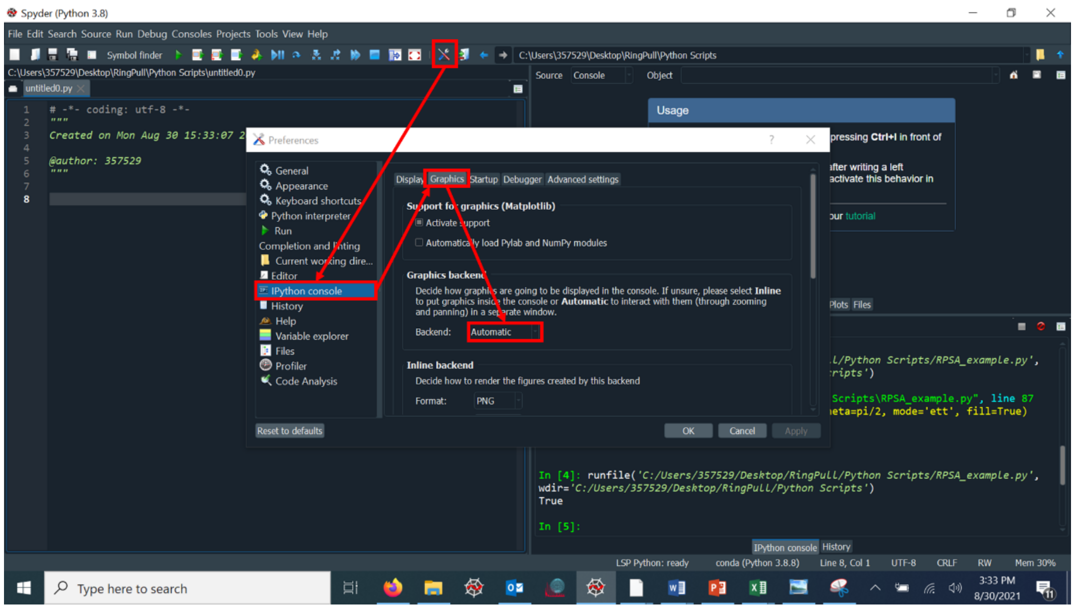
	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X
1	"x"	"y"	"x_c"	"y_c"	"u"	"v"	"u_c"	"v_c"	"exx"	"eyy"	"exy"	"fxx"	"fxy"	"fyx"	"fyy"	"e1"	"e2"	"gamma"	"e_tresc	"e_vonmi	"sigma"	"Xp"	"Yp"	"Zp"
2	51	322	-1173	702	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	-1	0	0	0
3	55	322	-1169	702	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	-1	0	0	0
4	59	322	-1165	702	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	-1	0	0	0
5	63	322	-1161	702	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	-1	0	0	0
6	67	322	-1157	702	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	-1	0	0	0
7	71	322	-1153	702	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	-1	0	0	0
8	75	322	-1149	702	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	-1	0	0	0
9	79	322	-1145	702	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	-1	0	0	0

Opening and running the script

Start by opening Spyder:


All
Apps
Documents
Settings
Photos
More
Best match
Spyder (Anaconda3)
App
Folders
spyder - in site-packages
spyder - in site-packages
spyder_kernels
spyder-4.2.5.dist-info - in site-packages
spyder-4.2.5.dist-info - in site-packages
pyls_spyder
pyls_spyder-0.3.2.dist-info
Documents
spyder-4.2.5-py38haa95532_0.json
spyder-kernels-1.10.2-py38haa95532_0.json
pyls-spyder-0.3.2-pyhd3eb1b0_0.json
Apps
Spyder (Anaconda3)
App
Open
Run as administrator
Open file location
Unpin from taskbar
Pin to Start
Uninstall
spyder (Anaconda3)

It is required to change the graphics backend to display the plots in a separate window. The backend cannot be inline, so any other setting should work. This will allow Spyder to create additional windows for any plots or figures that are made.



```
# -*- coding: utf-8 -*-
"""
Created on Mon Aug 30 15:33:07 2

@author: 357529
"""
```

Support for graphics (Matplotlib)

- [✓] Activate support
- [ ] Automatically load Pylab and NumPy modules

Graphics backend

Decide how graphics are going to be displayed in the console. If unsure, please select **Inline** to put graphics inside the console or **Automatic** to interact with them (through zooming and panning) in a separate window.

Backend: Automatic

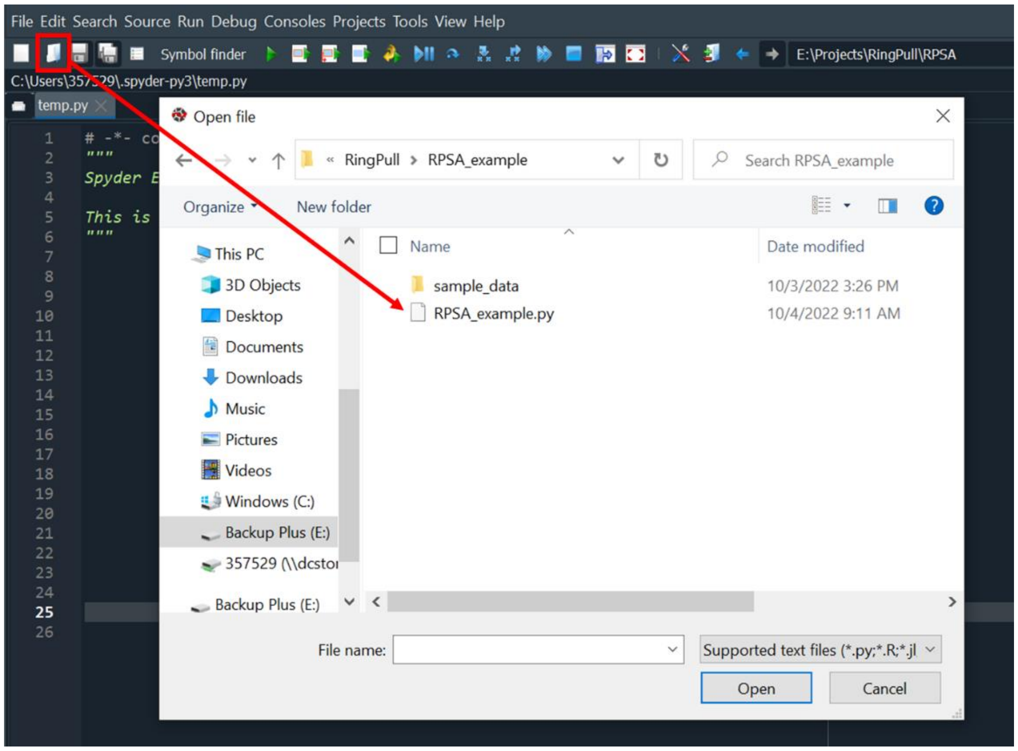
Inline backend

Decide how to render the figures created by this backend

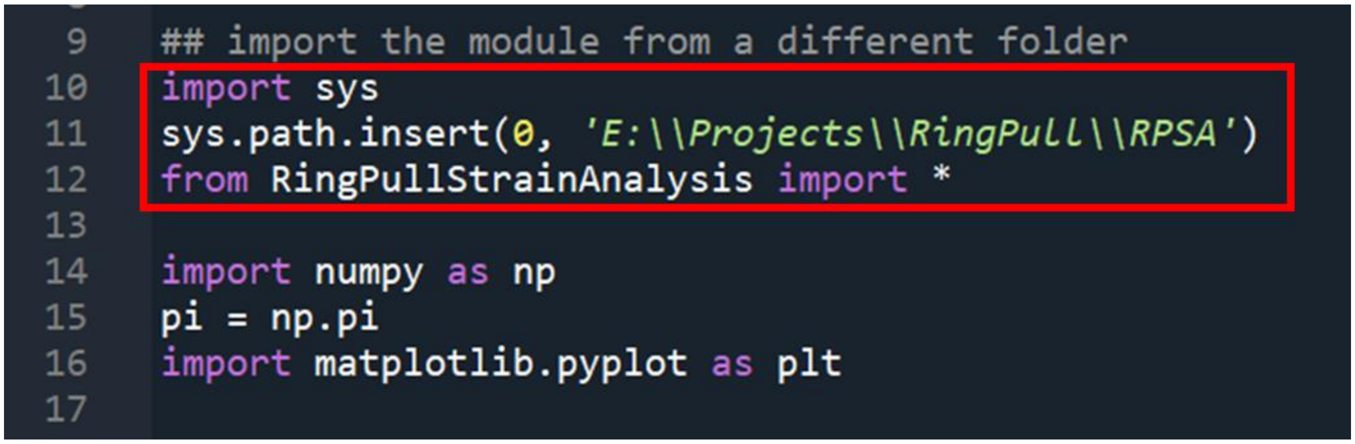
Format: PNG

```
In [4]: runfile('C:/Users/357529/Desktop/RingPull/Python Scripts/RPSA_example.py', wdir='C:/Users/357529/Desktop/RingPull/Python Scripts')
True

In [5]:
```

Now the RPSA example script can be opened.

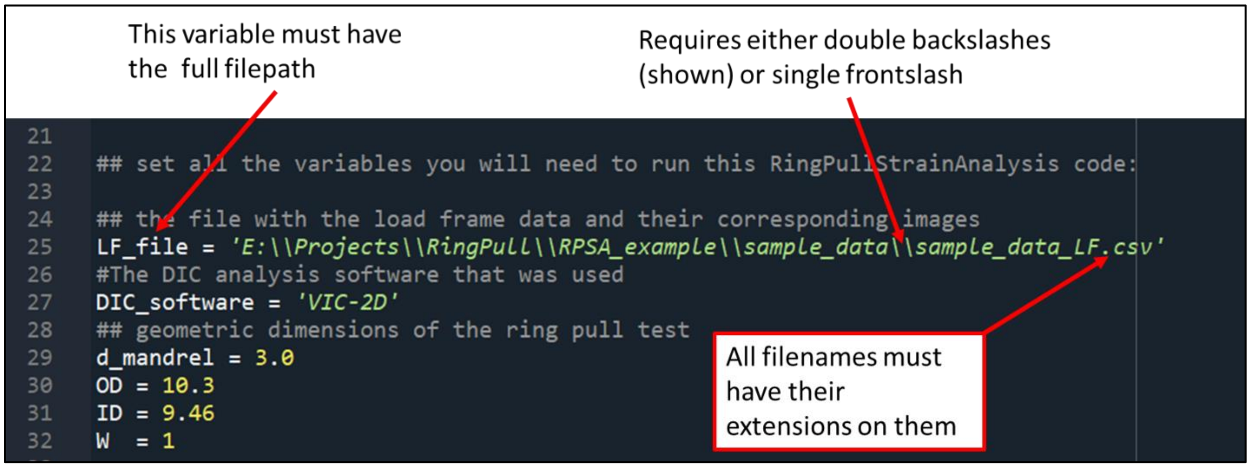


Now, walking through the script. Here are some imported packages. If you downloaded Anaconda, these should all be available with minimal hassle. We also define pi as we will often use it. Note the red box – this is where we import the module RingPullStrainAnalysis.py from a different folder.



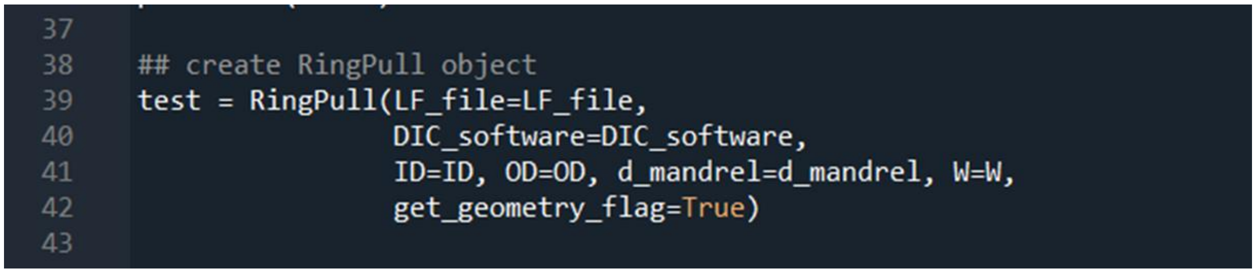
```
## import the module from a different folder
import sys
sys.path.insert(0, 'E:\\Projects\\RingPull\\RPSA')
from RingPullStrainAnalysis import *

import numpy as np
pi = np.pi
import matplotlib.pyplot as plt

```

Here are the variables you can change in the code. The most important are the directory and the filename for the load frame data. Here is also where you specify geometric dimensions of the ring.


This variable must have the full filepath
Requires either double backslashes (shown) or single frontslash
All filenames must have their extensions on them

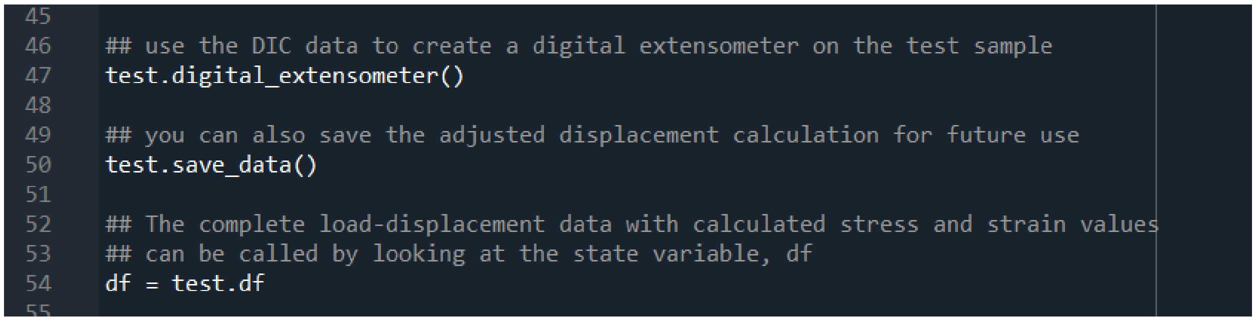
```python

## set all the variables you will need to run this RingPullStrainAnalysis code:

## the file with the load frame data and their corresponding images
LF_file = 'E:\\Projects\\RingPull\\RPSA_example\\sample_data\\sample_data_LF.csv'
#The DIC analysis software that was used
DIC_software = 'VIC-2D'
## geometric dimensions of the ring pull test
d_mandrel = 3.0
OD = 10.3
ID = 9.46
W  = 1
```

This line of code creates an instance of the RingPull class which is intended to encompass an entire test. It loads in the load frame data and stores it in memory.

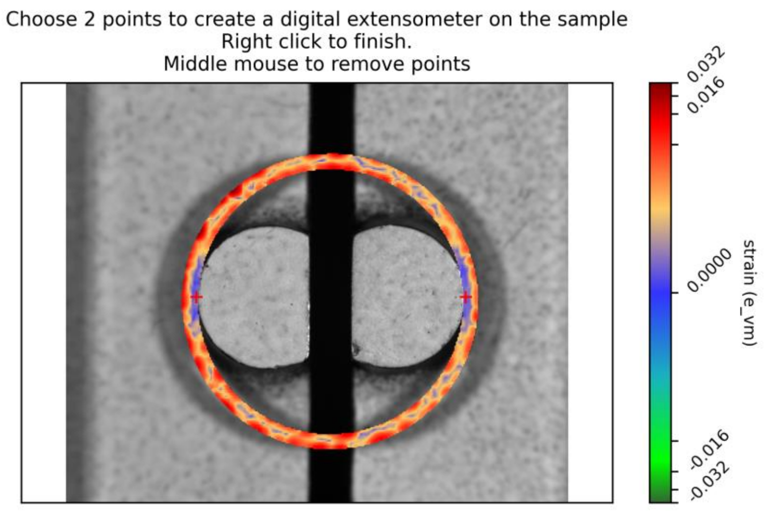


```python

## create RingPull object
test = RingPull(LF_file=LF_file,
                DIC_software=DIC_software,
                ID=ID, OD=OD, d_mandrel=d_mandrel, W=W,
                get_geometry_flag=True)

```

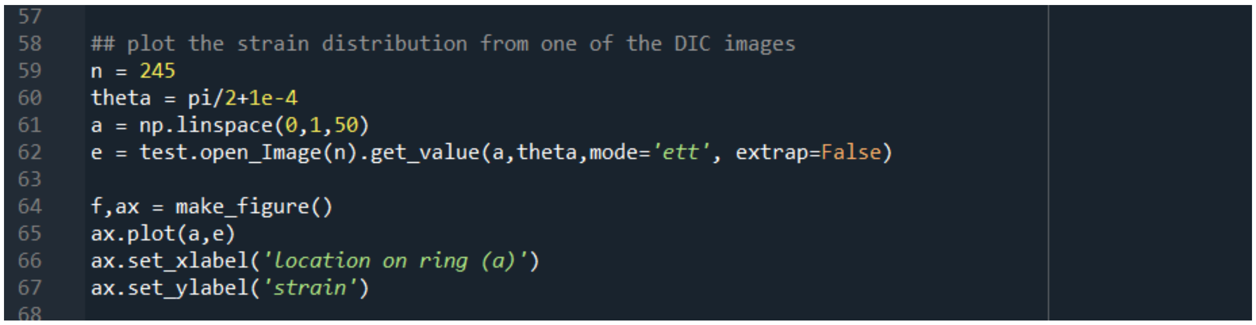
You can put a digital extensometer on the sample to get an adjusted displacement. This pops up a user interface allowing the user to select two points to track. This will put an additional column in the load-displacement DataFrame internally in Python. The user can save this data externally to a .csv file for future use.



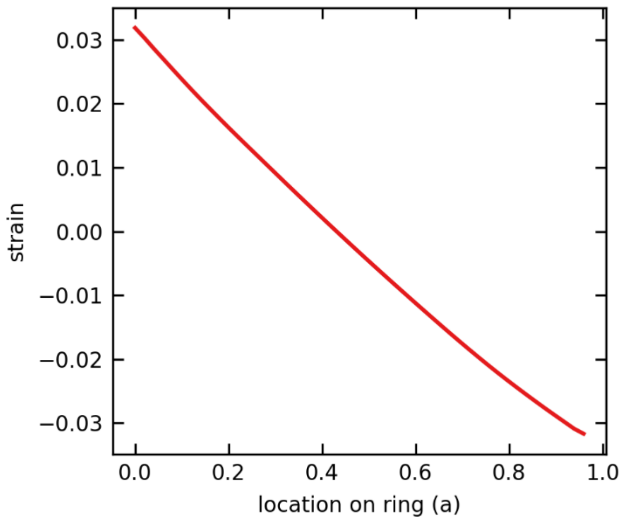
```python

## use the DIC data to create a digital extensometer on the test sample
test.digital_extensometer()

## you can also save the adjusted displacement calculation for future use
test.save_data()

## The complete load-displacement data with calculated stress and strain values
## can be called by looking at the state variable, df
df = test.df
```


Choose 2 points to create a digital extensometer on the sample
Right click to finish.
Middle mouse to remove points
0.032
0.016
0.0000
-0.016
-0.032
strain (e_vm)

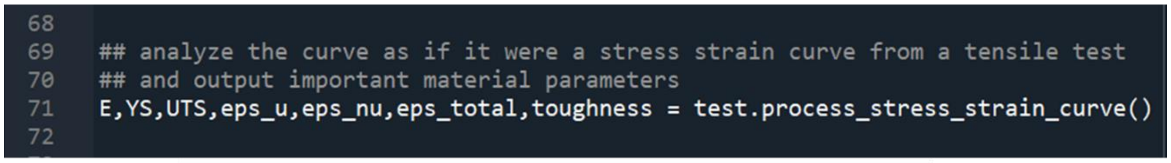
Strain values at arbitrary points can be probed. Here is an example of strain values being obtained from the DIC mesh and plotted.



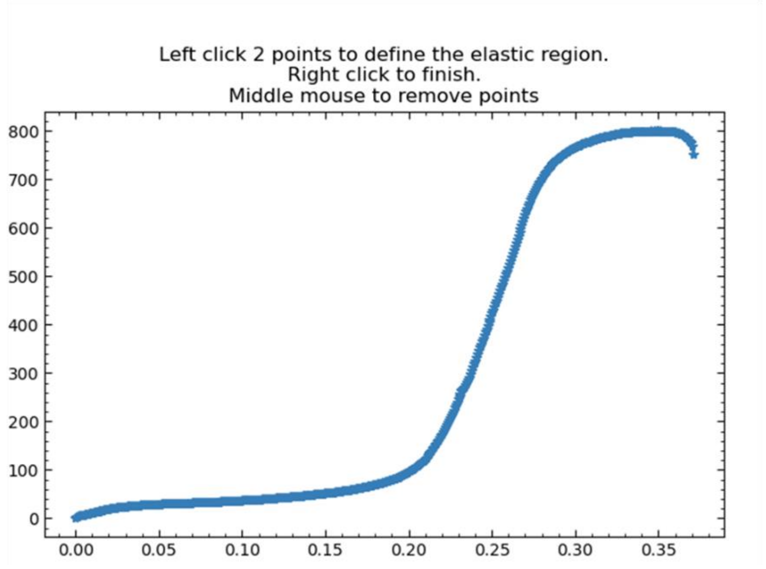
```

## plot the strain distribution from one of the DIC images
n = 245
theta = pi/2+1e-4
a = np.linspace(0,1,50)
e = test.open_Image(n).get_value(a,theta,mode='ett', extrap=False)

f,ax = make_figure()
ax.plot(a,e)
ax.set_xlabel('location on ring (a)')
ax.set_ylabel('strain')

```

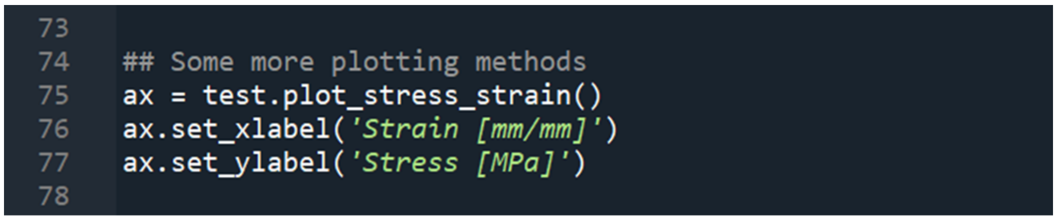


You can also analyze this curve as if it were a tensile curve. When this command is run, user graphical input is requested.

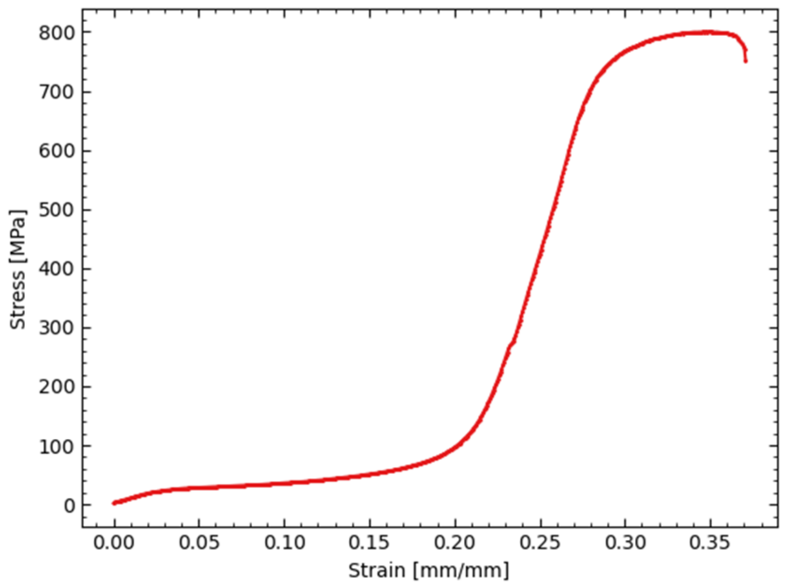


```
## analyze the curve as if it were a stress strain curve from a tensile test
## and output important material parameters
E,YS,UTS,eps_u,eps_nu,eps_total,toughness = test.process_stress_strain_curve()
```



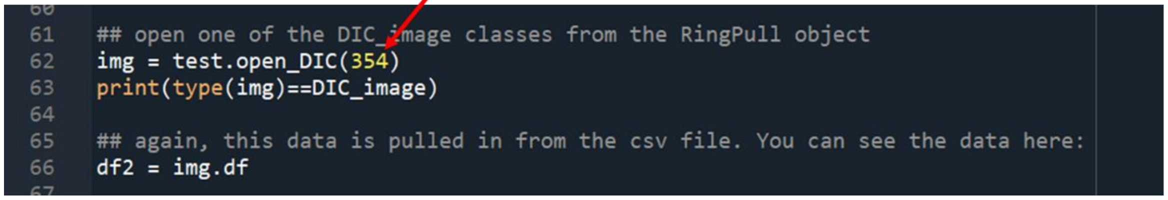


```

## Some more plotting methods
ax = test.plot_stress_strain()
ax.set_xlabel('Strain [mm/mm]')
ax.set_ylabel('Stress [MPa]')

```



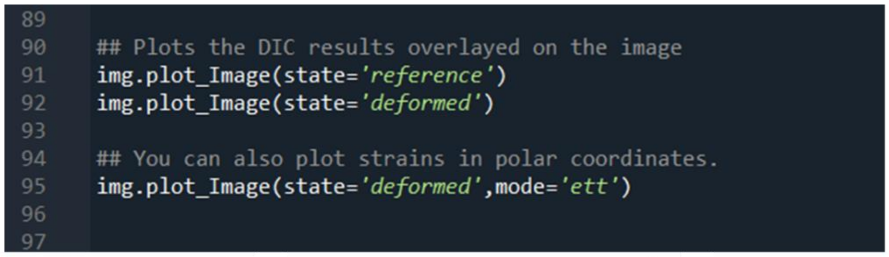
Now, we can open a single image file to look at a specific point in the test. You can also access the data in spyder's variable editor.

We opened a random image (number 354). Note that the image numbers correspond with their place in the load frame file, and NOT with any numbers in their filenames

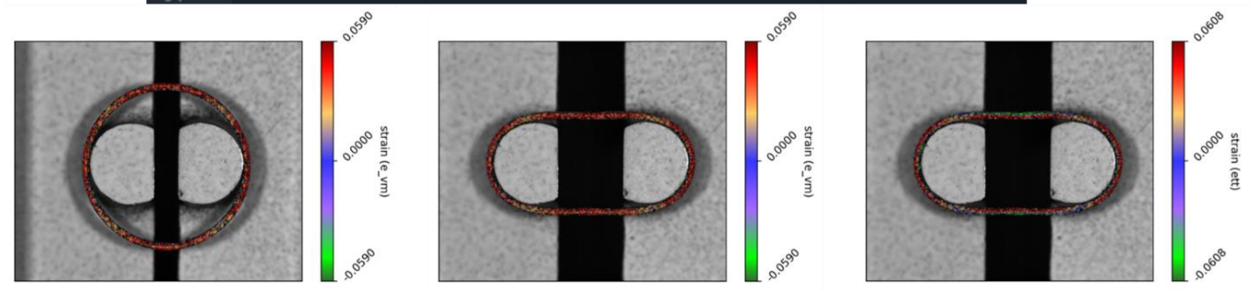


```python
## open one of the DIC_image classes from the RingPull object
img = test.open_DIC(354)
print(type(img)==DIC_image)

## again, this data is pulled in from the csv file. You can see the data here:
df2 = img.df
```

Now we can also plot a few more things, shown below.



```python

## Plots the DIC results overlayed on the image
img.plot_Image(state='reference')
img.plot_Image(state='deformed')

## You can also plot strains in polar coordinates.
img.plot_Image(state='deformed',mode='ett')


```


strain (e_vm)
strain (e_vm)
strain (ett)