

Downloaded Python(x, y): <https://python-xy.github.io/>

Did a google search for “how to remove noise from data” and found an interesting Quora page:

<https://www.quora.com/What-are-some-methods-to-reduce-the-noise-of-a-dataset-if-the-data-is-highly-turbulent>

The “best” answer mentions several methods. One of them is Wiener filtering, which SciPy has a package for:

<https://docs.scipy.org/doc/scipy-0.14.0/reference/generated/scipy.signal.wiener.html>

According to SciPy API, the filter can be implemented by calling the function **scipy.signal.wiener**(*im*, *mysize=None*, *noise=None*)

Where *im* is an *ndarray*, i.e. an N-dimensional array,

mysize is an *int* or *arraylike*, *optional*

noise is an optional float,

and

out is the filtered result, the output and the same shape as *im*, the input.

Note that, if *mysize* is a scalar, then it “is used as the size in each dimension”.

Otherwise, it is “an N-length list giving the size of the Wiener filter window in each dimension”.

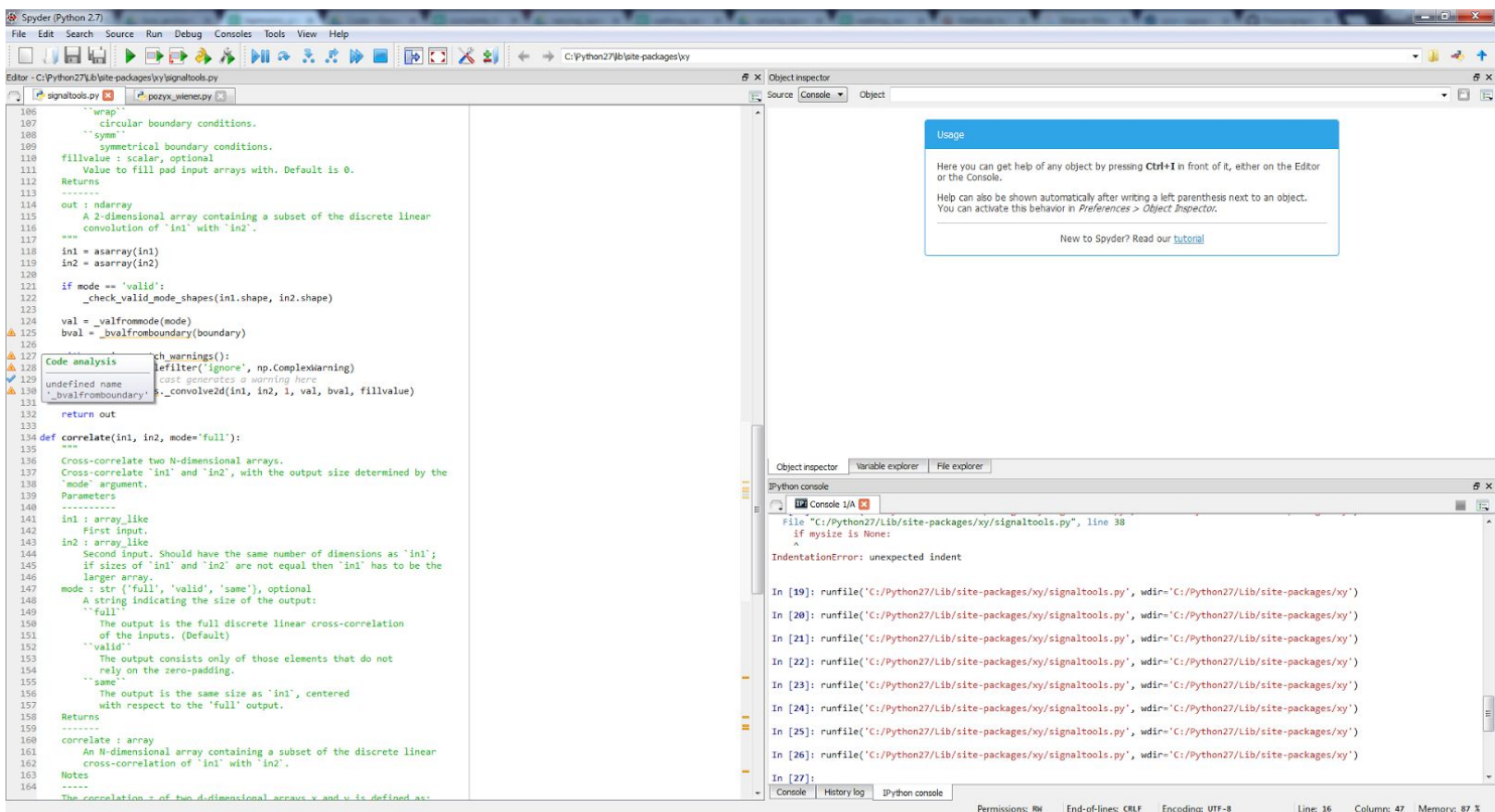
Copied the Pozyx code from graphFromText2.py

github.com/yawnypaws/Pozyx/blob/master/house_code/original_programs/graphFromText2.py

and pasted it into a local file to modify (I also created a local backup file of the original .py script).

In a separate python file - assembled all of the source code to use the Wiener function, *scipy.signal.wiener*.

Debugged the program until I could call *scipy.signal.wiener* without any warnings or errors, such as the following:



Screenshot 1. Picture of a common bug when getting the necessary source code for the wiener function.

The warning in Screenshot 1 says, "Code analysis undefined name '`_bvalfromboundary`'. Using the find function (CTRL + F) I just copied all of the absent source code into the python file. For this example, I did a find for `_bvalfromboundary`, which led me to some source code that was missing in order to use the wiener function with the minimal amount of code.

Made a copy of the `signaltools.py`

Everything looked OK, except for the following error:

ValueError: Attempted relative import in non-package

and its corresponding instruction

```
15 from . import sigtools
```

Trying to use a potential solution:

groups.google.com/forum/#!topic/winpython/de69CrhR3XU

I found an example of someone importing sigtools:

```
1 '''
2 Created on Mar 16, 2013
3
4 @author: tiago
5 '''
6
7 import numpy as np
8 from scipy.interpolate import UnivariateSpline
9 from scipy.signal import wiener, filtfilt, butter, gaussian, fr
10 from scipy.ndimage import filters
11 import scipy.optimize as op
12 import matplotlib.pyplot as plt
13
14 def ssqe(sm, s, npts):
15     return np.sqrt(np.sum(np.power(s-sm,2)))/npts
```

Screenshot 2. From solution to the ValueError error.

Found this at the following site:

nehalem labs.net/prototype/blog/2013/04/09/an-introduction-to-smoothing-time-series-in-python-part-ii-wiener-filter-and-smoothing-splines/

Testing the graphFromText2.py script and the local file path did not work.

Uploaded the local .csv file to the git repository.

Used the link to the git file (for the .csv) and the file path worked.

Got a new error:

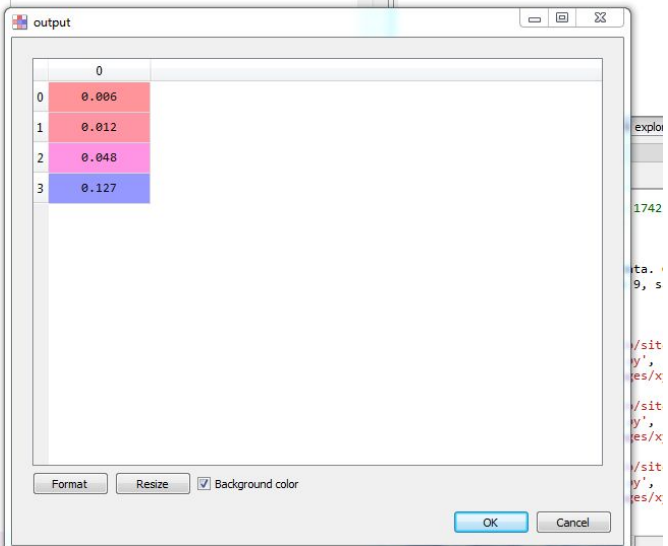
```
CParserError: Error tokenizing data. C error: Expected 2 fields in line 9, saw 3
```

```
35 df=pd.read_csv('https://github.com/etcy1/SciPozyx/blob/master/Data/walking_varying_speed_multi_test_0')
```

Stopped working on resolving this issue.

Started writing small inputs to test the wiener function:

```
191 # don't use _valfrommode, since correlate should not accept numeric modes
192 try:
193     val = _modedict[mode]
194 except KeyError:
195     raise ValueError("Acceptable mode flags are 'valid',",
196                     " 'same', or 'full'.")
197
198 if in1.ndim == in2.ndim == 0:
199     return in1 * in2
200 elif not in1.ndim == in2.ndim:
201     raise ValueError("in1 and in2 should have the same dimensionality")
202
203 if mode == 'valid':
204     _check_valid_mode_shapes(in1.shape, in2.shape)
205     ps = [i - j + 1 for i, j in zip(in1.shape, in2.shape)]
206     out = np.empty(ps, in1.dtype)
207
208     z = sigtools._correlateND(in1, in2, out, val)
209 else:
210     ps = [i + j - 1 for i, j in zip(in1.shape, in2.shape)]
211     # zero pad input
212     in1padded = np.zeros(ps, in1.dtype)
213     sc = [slice(0, i) for i in in1.shape]
214     in1padded[sc] = in1.copy()
215
216     if mode == 'full':
217         out = np.empty(ps, in1.dtype)
218     elif mode == 'same':
219         out = np.empty(in1.shape, in1.dtype)
220
221     z = sigtools._correlateND(in1padded, in2, out, val)
222
223 return z
224
225 data = [0, 0.017993, 0.018665, 0.18156]
226
227 output = wiener(data)
```



| | 0 |
|---|-------|
| 0 | 0.006 |
| 1 | 0.012 |
| 2 | 0.048 |
| 3 | 0.127 |

Screenshot 3. Verification that the wiener function works without errors, processing a 4 element array called “data”. The output of the function is called “output”, and is shown in the box labeled “output” (right in the picture).

Looking into a Kalman (and Bayesian) filter in Python:

github.com/rlabbe/Kalman-and-Bayesian-Filters-in-Python/blob/master/07-Kalman-Filter-Math.ipynb