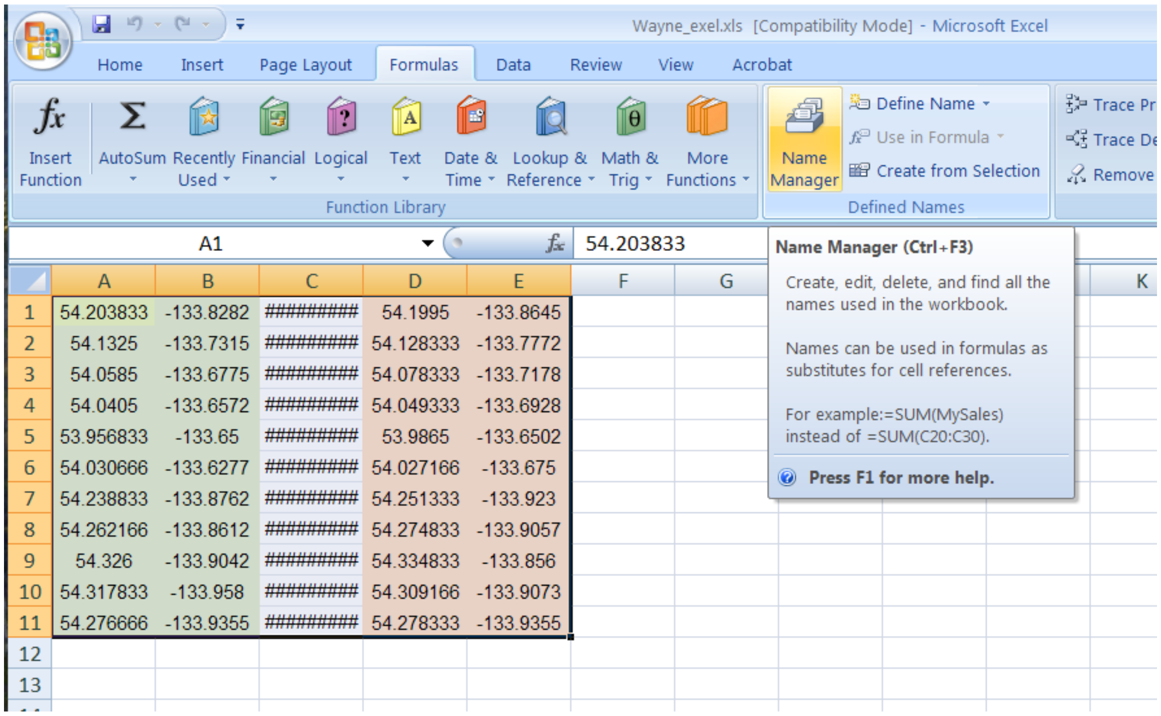
The Excel spreadsheet (xls) provided by Jane Hansen opens as in the following, missing the header and without the required initial 2 columns for ID and Date (the correct field order is {ID, DATE1, LAT1, LON1, DATE2, LAT2, LON2, … , DATEn, LATn, LONn}). Also, no defined ranges were found…in order to view the named ranges, go to the Formulas tab (or ribbon), and select Name Manager as shown:



	A	B	C	D	E
1	54.203833	-133.8282	#########	54.1995	-133.8645
2	54.1325	-133.7315	#########	54.128333	-133.7772
3	54.0585	-133.6775	#########	54.078333	-133.7178
4	54.0405	-133.6572	#########	54.049333	-133.6928
5	53.956833	-133.65	#########	53.9865	-133.6502
6	54.030666	-133.6277	#########	54.027166	-133.675
7	54.238833	-133.8762	#########	54.251333	-133.923
8	54.262166	-133.8612	#########	54.274833	-133.9057
9	54.326	-133.9042	#########	54.334833	-133.856
10	54.317833	-133.958	#########	54.309166	-133.9073
11	54.276666	-133.9355	#########	54.278333	-133.9355

Name Manager (Ctrl+F3)

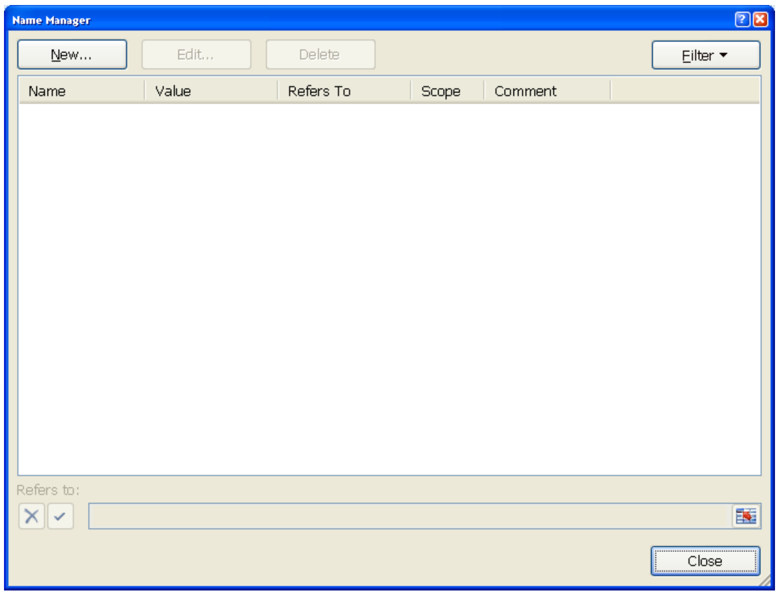
Create, edit, delete, and find all the names used in the workbook.

Names can be used in formulas as substitutes for cell references.

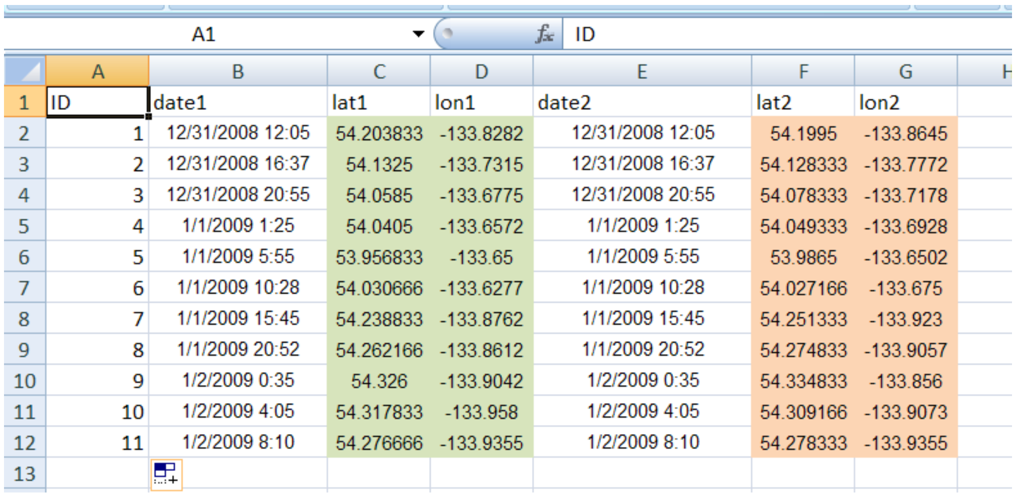
For example:=SUM(MySales) instead of =SUM(C20:C30).

Press F1 for more help.

The below shows nothing loaded in the Name Manager – now close the Name Manager to manipulate a few necessary changes on the sheet.



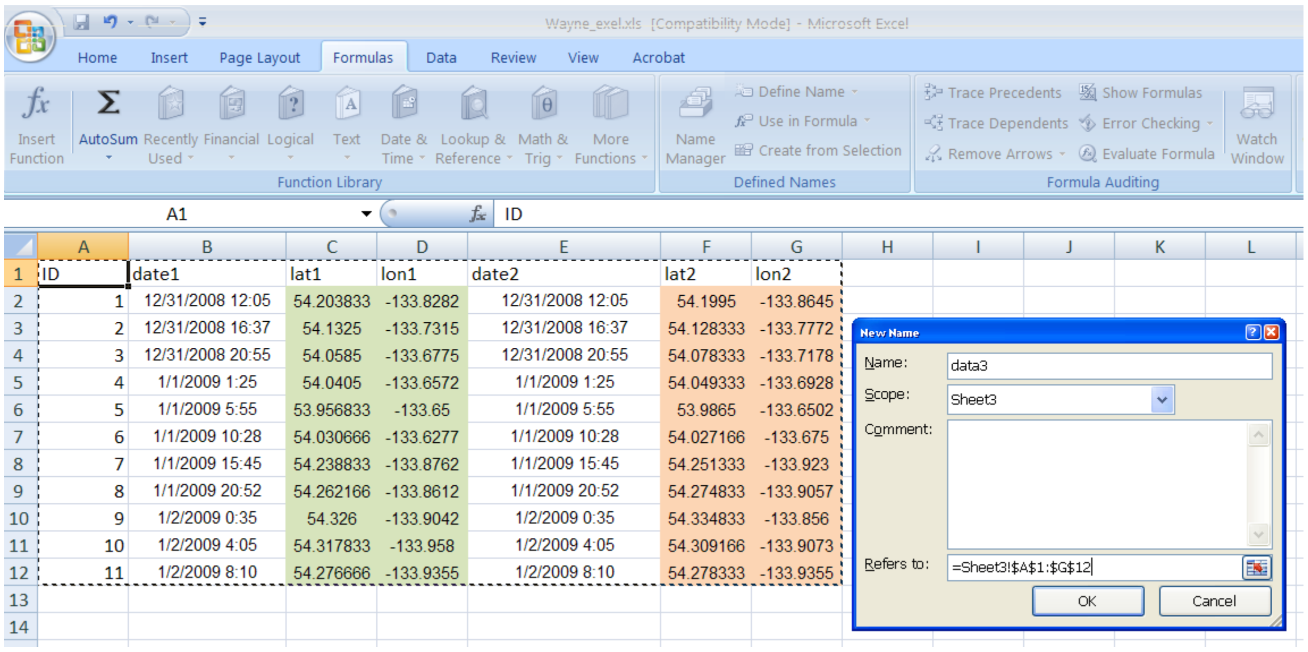
Correct the sheet to reflect the missing header and missing 2 columns for ID and Date1 (it doesn't matter what these fields are named – the script 'expects' to find lat/lon data in columns C, D and F, G, etc. (and is extensible). For purposes of this demo, the contained ID and Date values can be 'dummy' values. Below is how the corrected sheet may appear:



A1 ID

	A	B	C	D	E	F	G
1	ID	date1	lat1	lon1	date2	lat2	lon2
2	1	12/31/2008 12:05	54.203833	-133.8282	12/31/2008 12:05	54.1995	-133.8645
3	2	12/31/2008 16:37	54.1325	-133.7315	12/31/2008 16:37	54.128333	-133.7772
4	3	12/31/2008 20:55	54.0585	-133.6775	12/31/2008 20:55	54.078333	-133.7178
5	4	1/1/2009 1:25	54.0405	-133.6572	1/1/2009 1:25	54.049333	-133.6928
6	5	1/1/2009 5:55	53.956833	-133.65	1/1/2009 5:55	53.9865	-133.6502
7	6	1/1/2009 10:28	54.030666	-133.6277	1/1/2009 10:28	54.027166	-133.675
8	7	1/1/2009 15:45	54.238833	-133.8762	1/1/2009 15:45	54.251333	-133.923
9	8	1/1/2009 20:52	54.262166	-133.8612	1/1/2009 20:52	54.274833	-133.9057
10	9	1/2/2009 0:35	54.326	-133.9042	1/2/2009 0:35	54.334833	-133.856
11	10	1/2/2009 4:05	54.317833	-133.958	1/2/2009 4:05	54.309166	-133.9073
12	11	1/2/2009 8:10	54.276666	-133.9355	1/2/2009 8:10	54.278333	-133.9355

In order to set the current sheet's data range, you can 'name' the range - click 'New…' to open the New Name window from the Name Manager, with which we are going to name a range 'data3' to correspond to the Sheet3 it is found on – for the data on this sheet, use the range '=Sheet3!$A$1:$G$12' as shown below (you may set the range interactively as well, using the button to the bottom right just above 'Cancel'. Notice the scope is set to Sheet3.

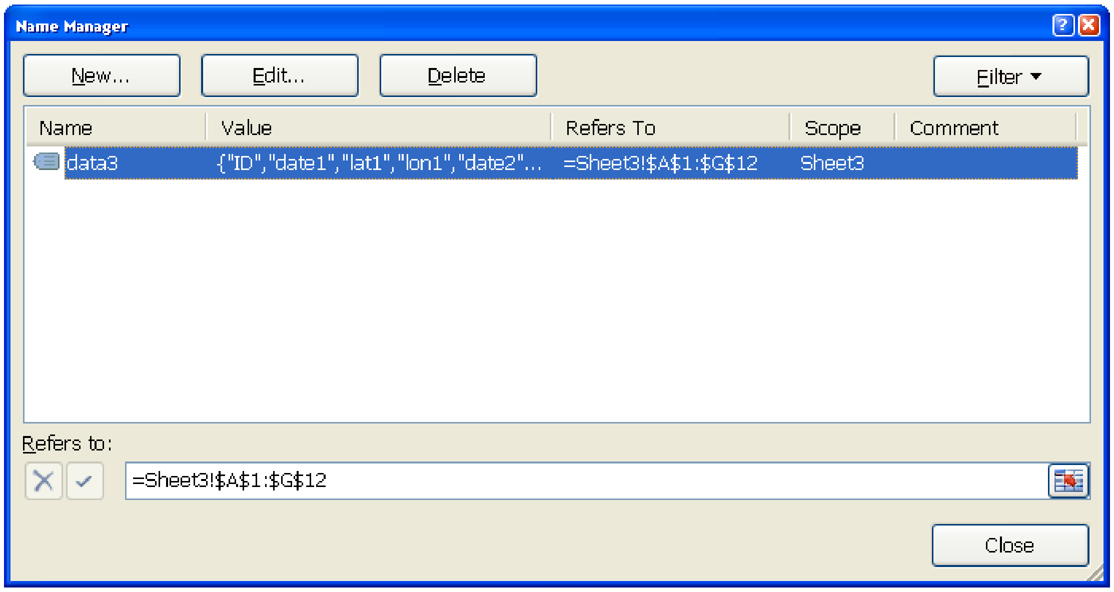


A1 ID

	A	B	C	D	E	F	G
1	ID	date1	lat1	lon1	date2	lat2	lon2
2	1	12/31/2008 12:05	54.203833	-133.8282	12/31/2008 12:05	54.1995	-133.8645
3	2	12/31/2008 16:37	54.1325	-133.7315	12/31/2008 16:37	54.128333	-133.7772
4	3	12/31/2008 20:55	54.0585	-133.6775	12/31/2008 20:55	54.078333	-133.7178
5	4	1/1/2009 1:25	54.0405	-133.6572	1/1/2009 1:25	54.049333	-133.6928
6	5	1/1/2009 5:55	53.956833	-133.65	1/1/2009 5:55	53.9865	-133.6502
7	6	1/1/2009 10:28	54.030666	-133.6277	1/1/2009 10:28	54.027166	-133.675
8	7	1/1/2009 15:45	54.238833	-133.8762	1/1/2009 15:45	54.251333	-133.923
9	8	1/1/2009 20:52	54.262166	-133.8612	1/1/2009 20:52	54.274833	-133.9057
10	9	1/2/2009 0:35	54.326	-133.9042	1/2/2009 0:35	54.334833	-133.856
11	10	1/2/2009 4:05	54.317833	-133.958	1/2/2009 4:05	54.309166	-133.9073
12	11	1/2/2009 8:10	54.276666	-133.9355	1/2/2009 8:10	54.278333	-133.9355

Name: data3
Scope: Sheet3
Comment:
Refers to: =Sheet3!$A$1:$G$12

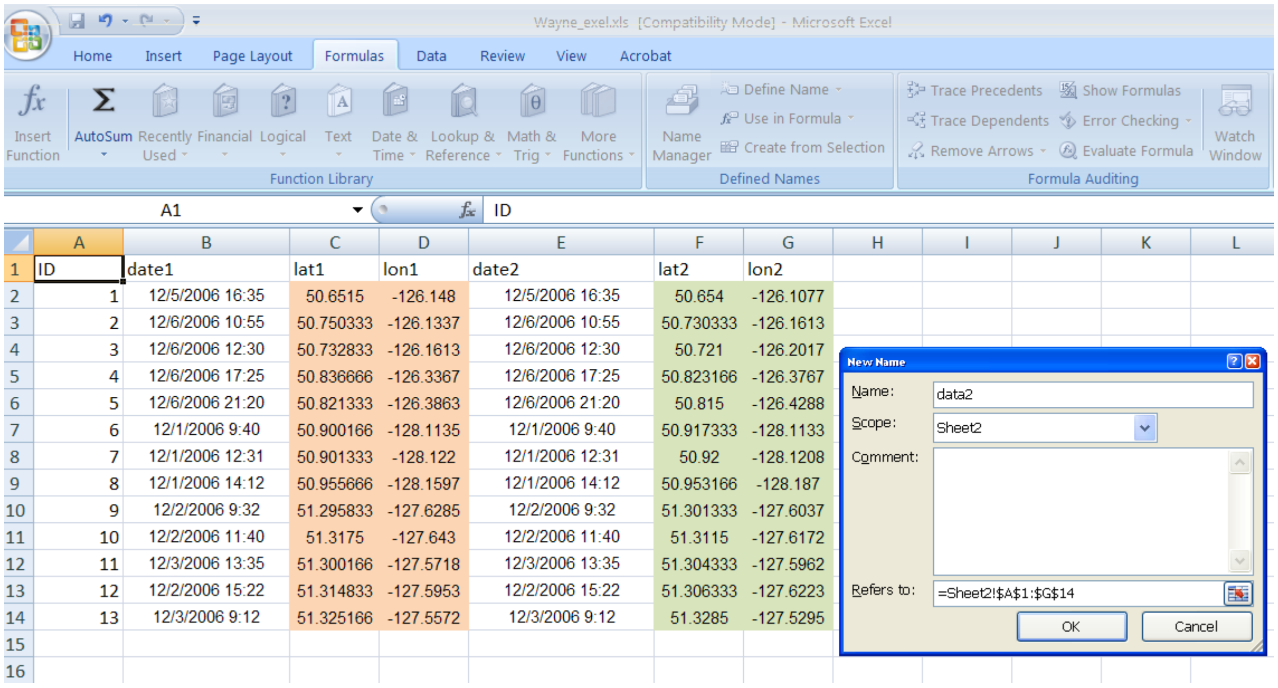
The Name Manager will list the new reference as follows:



Name	Value	Refers To	Scope	Comment
data3	{"ID","date1","lat1","lon1","date2"...	=Sheet3!$A$1:$G$12	Sheet3	

Refers to: =Sheet3!$A$1:$G$12

Do the same for Sheet2 (and get rid of Sheet1) – the following is how Sheet2 may look, including the New Name window entry:



	A	B	C	D	E	F	G
1	ID	date1	lat1	lon1	date2	lat2	lon2
2	1	12/5/2006 16:35	50.6515	-126.148	12/5/2006 16:35	50.654	-126.1077
3	2	12/6/2006 10:55	50.750333	-126.1337	12/6/2006 10:55	50.730333	-126.1613
4	3	12/6/2006 12:30	50.732833	-126.1613	12/6/2006 12:30	50.721	-126.2017
5	4	12/6/2006 17:25	50.836666	-126.3367	12/6/2006 17:25	50.823166	-126.3767
6	5	12/6/2006 21:20	50.821333	-126.3863	12/6/2006 21:20	50.815	-126.4288
7	6	12/1/2006 9:40	50.900166	-128.1135	12/1/2006 9:40	50.917333	-128.1133
8	7	12/1/2006 12:31	50.901333	-128.122	12/1/2006 12:31	50.92	-128.1208
9	8	12/1/2006 14:12	50.955666	-128.1597	12/1/2006 14:12	50.953166	-128.187
10	9	12/2/2006 9:32	51.295833	-127.6285	12/2/2006 9:32	51.301333	-127.6037
11	10	12/2/2006 11:40	51.3175	-127.643	12/2/2006 11:40	51.3115	-127.6172
12	11	12/3/2006 13:35	51.300166	-127.5718	12/3/2006 13:35	51.304333	-127.5962
13	12	12/2/2006 15:22	51.314833	-127.5953	12/2/2006 15:22	51.306333	-127.6223
14	13	12/3/2006 9:12	51.325166	-127.5572	12/3/2006 9:12	51.3285	-127.5295
15							
16							

New Name

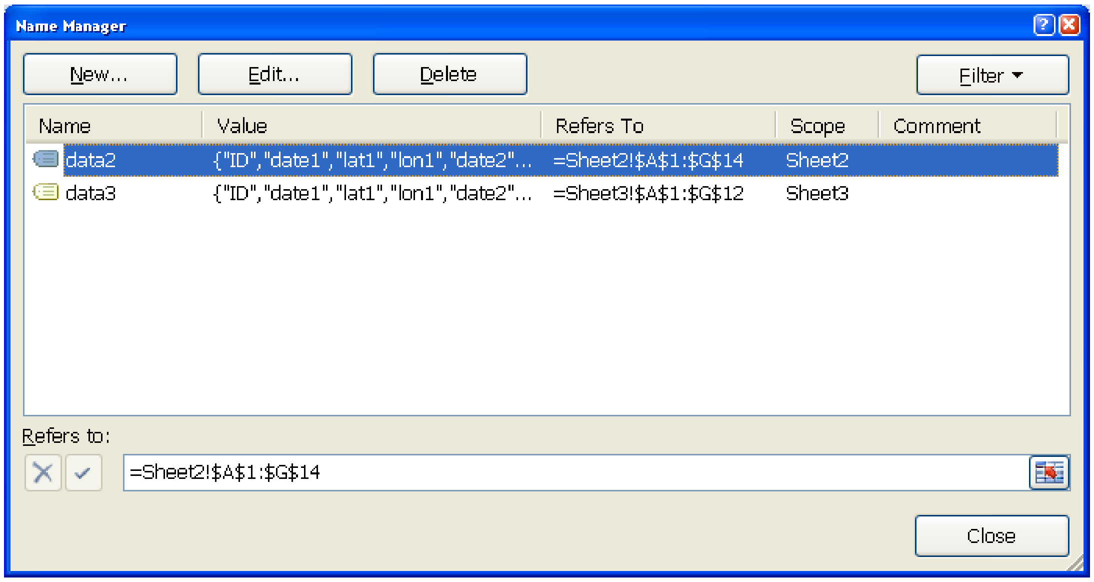
Name: data2

Scope: Sheet2

Comment:

Refers to: =Sheet2!$A$1:$G$14

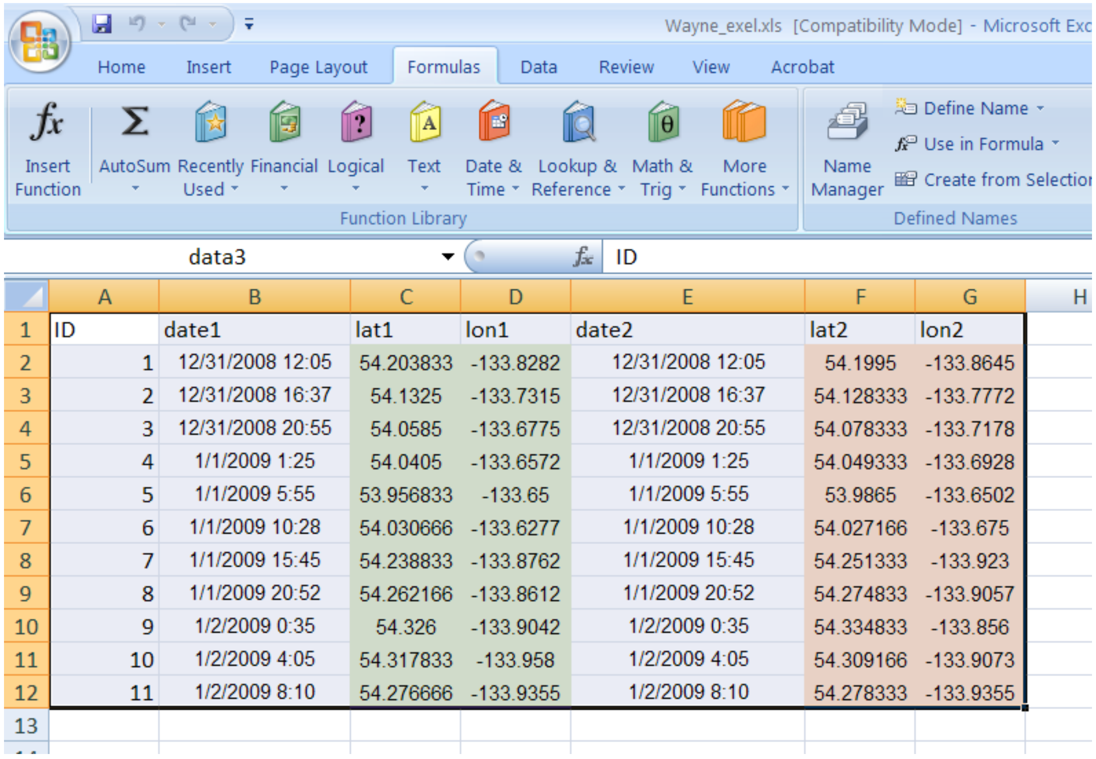
When the New Name window is dismissed, this is how the 2 'named' references appear in the Name Manager – notice that data2 reflects the range for 2 records (13 records + header) more than data3 (11 records + header):



Name	Value	Refers To	Scope	Comment
data2	{"ID","date1","lat1","lon1","date2"...	=Sheet2!$A$1:$G$14	Sheet2	
data3	{"ID","date1","lat1","lon1","date2"...	=Sheet3!$A$1:$G$12	Sheet3	

Refers to: =Sheet2!$A$1:$G$14

Also notice the Name box to the left of the cell address box (showing 'data3' below), that if you select the 'named' range reference, the range will highlight – this is a check that this naming of range references was done properly:



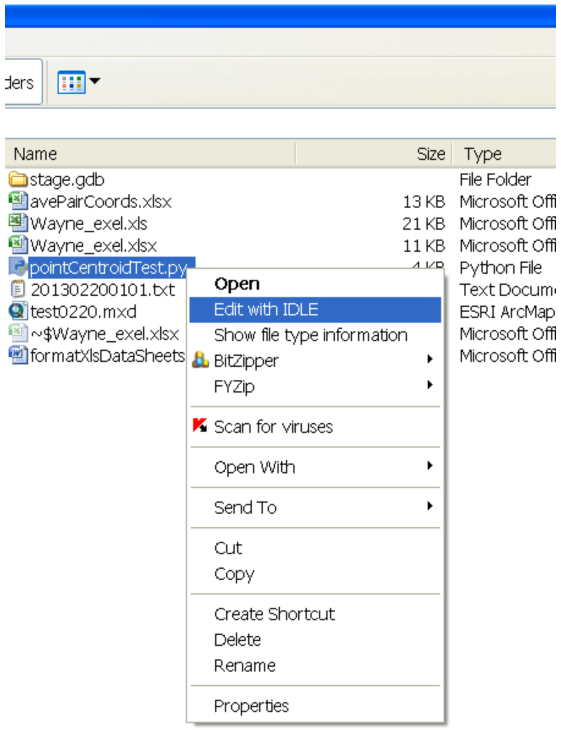
data3 | ID

	A	B	C	D	E	F	G
1	ID	date1	lat1	lon1	date2	lat2	lon2
2	1	12/31/2008 12:05	54.203833	-133.8282	12/31/2008 12:05	54.1995	-133.8645
3	2	12/31/2008 16:37	54.1325	-133.7315	12/31/2008 16:37	54.128333	-133.7772
4	3	12/31/2008 20:55	54.0585	-133.6775	12/31/2008 20:55	54.078333	-133.7178
5	4	1/1/2009 1:25	54.0405	-133.6572	1/1/2009 1:25	54.049333	-133.6928
6	5	1/1/2009 5:55	53.956833	-133.65	1/1/2009 5:55	53.9865	-133.6502
7	6	1/1/2009 10:28	54.030666	-133.6277	1/1/2009 10:28	54.027166	-133.675
8	7	1/1/2009 15:45	54.238833	-133.8762	1/1/2009 15:45	54.251333	-133.923
9	8	1/1/2009 20:52	54.262166	-133.8612	1/1/2009 20:52	54.274833	-133.9057
10	9	1/2/2009 0:35	54.326	-133.9042	1/2/2009 0:35	54.334833	-133.856
11	10	1/2/2009 4:05	54.317833	-133.958	1/2/2009 4:05	54.309166	-133.9073
12	11	1/2/2009 8:10	54.276666	-133.9355	1/2/2009 8:10	54.278333	-133.9355
13							

Save the xls to more recent Excel version, xlsx. The file is ready to run with the provided script, pointCentroidTest.py.

This test script, although it could be made to run as a script tool, is geared to run from the same folder location as the xlsx. Make sure that a copy of this spreadsheet is in the same location the zip file was 'unzipped' to with the accompanying file geodatabase (gdb).

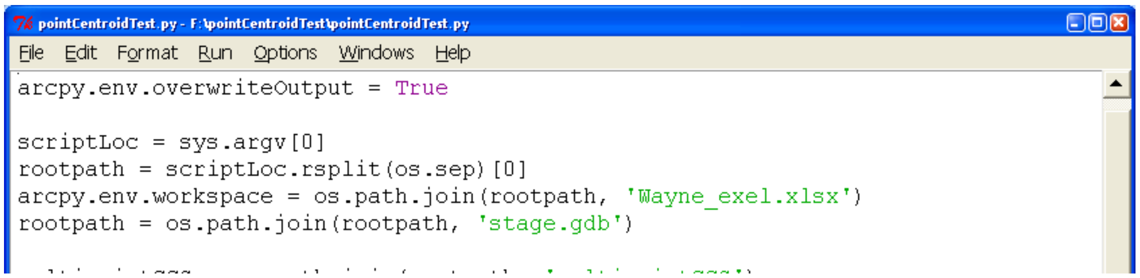
A line in the script file needs changing to reflect the 'new' input file – open the script file (you may use Windows Explorer) by right-clicking and selecting 'Edit with IDLE' (or you can use 'Open With' and select a text editor like Notepad). See below:



Change line 6 to this:

arcpy.env.workspace = os.path.join(rootpath, 'Wayne_exel.xlsx')

…so that the script file appears as follows:

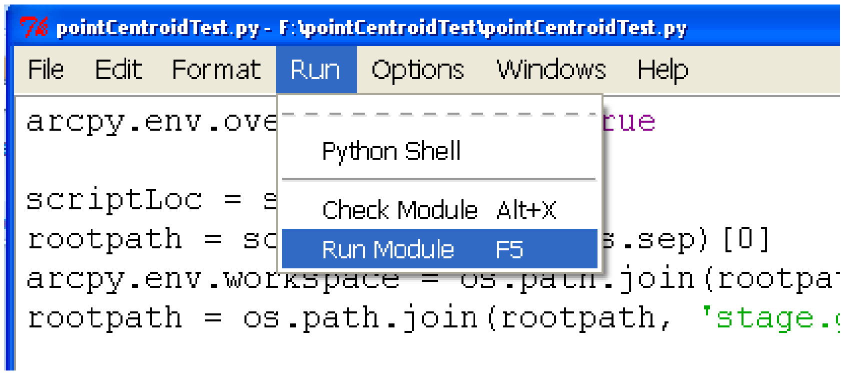

pointCentroidTest.py - F:\pointCentroidTest\pointCentroidTest.py
File Edit Format Run Options Windows Help

```
arcpy.env.overwriteOutput = True

scriptLoc = sys.argv[0]
rootpath = scriptLoc.rsplit(os.sep)[0]
arcpy.env.workspace = os.path.join(rootpath, 'Wayne_exel.xlsx')
rootpath = os.path.join(rootpath, 'stage.gdb')
```

Save and run the script – this is a simple script designed only for testing purposes, includes no error trapping or special messaging…

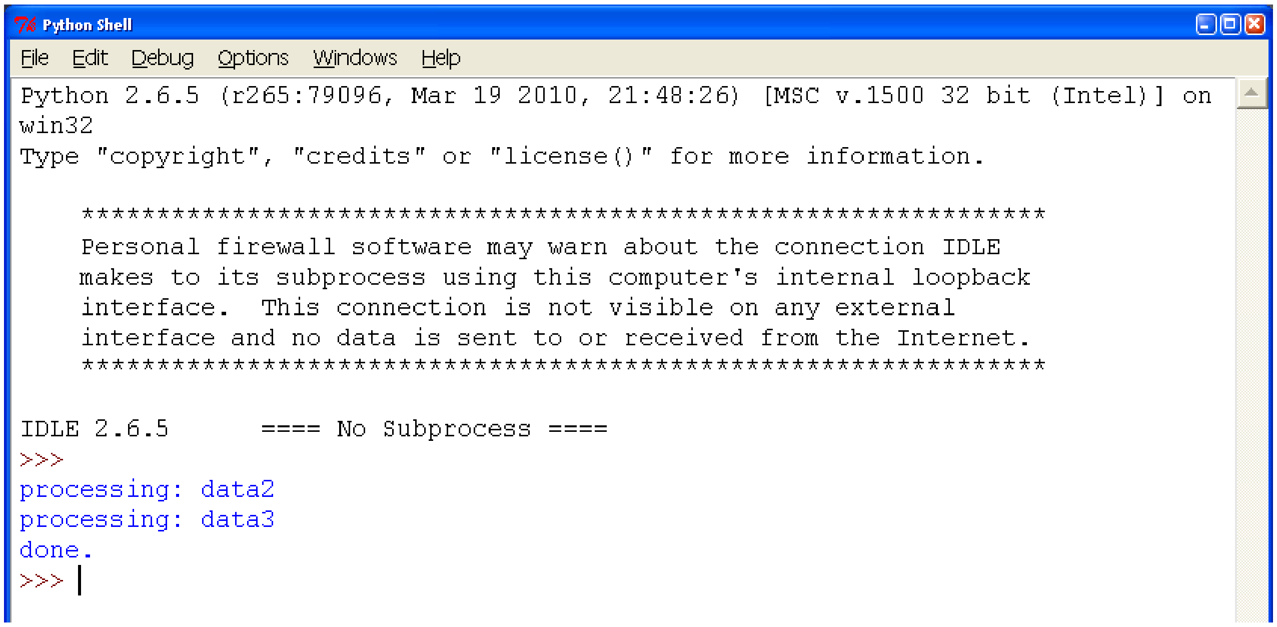
You can double-click to run the script from Windows Explorer, or if you already have it open in IDLE, you may go to the script file window and select 'Run', then 'Run Module' (or hit F5).



```
arcpy.env.ove                           rue

scriptLoc = s
rootpath = sc                          s.sep)[0]
arcpy.env.workspace = os.path.join(rootpa
rootpath = os.path.join(rootpath, 'stage.
```

I prefer to run in IDLE because I can then see any printed messages/errors without any added program lines to pause an execution window. If running via IDLE as I have shown and all runs well, this will be the only message output:



```
Python 2.6.5 (r265:79096, Mar 19 2010, 21:48:26) [MSC v.1500 32 bit (Intel)] on
win32
Type "copyright", "credits" or "license()" for more information.

    ****************************************************************
    Personal firewall software may warn about the connection IDLE
    makes to its subprocess using this computer's internal loopback
    interface.  This connection is not visible on any external
    interface and no data is sent to or received from the Internet.
    ****************************************************************

IDLE 2.6.5      ==== No Subprocess ====
>>> 
processing: data2
processing: data3
done.
>>> 
```

As an extremely general idea what the points look like, furthermost to the east of the dataset extent, they appear as follows in the map (shown with ESRI_Imagery_World_2D), see next page…

With the original script I posted, I was actually experimenting with loading multipoint geometry using an insert cursor and extracting the 'centroid' (via 'multiPoint.trueCentroid) which I could have copied to another feature class with another insert cursor, but I simply loaded that into another multipoint feature. As a result, every other multipoint (2, 4, 6, …, etc.) of the final projected multipoint feature class (multipointPCS_2) is the centroid.

Later, I decided to better clarify by simply appending code to the original code attached earlier to write out the multipoint centroid geometry to a regular point feature class (which I will attach along with this document). The new point feature class is called centroidsOnly and will simply be written to the same gdb.

Recall that this is a scenario for the purpose of testing the ability to get centroids from multipoint records… the script did not have to be written this way, so if there are further questions about the gdb output (or staged feature classes), the following list outlines it:

1 - multipointGCS_template

This is a template fc in Geographic Coordinate System, GCS_WGS_1984, used as input – probably not needed. I think I was initially having trouble with setting the Spatial Reference with the var, SR_GCS.

2 - multipointPCS_template

This is a template fc in Projected Coordinate System, NAD_1983_BC_Environment_Albers, used as input – also probably not needed....due to the same initial trouble setting Spatial Reference with the var, SR_PCS.

3- multipointGCS

This is the initially created GCS feature class, from the raw lat/lon coordinates.

4- multipointPCS_1

This was the 1st projected layer – the result of producing centroids on the GCS layer <u>before projecting</u>, for comparison with the placement of centroids <u>after projecting</u> (multipointPCS_2).

5- multipointPCS_2

This was the 2nd projected layer – as mentioned in item 4 above, this is the result of placing centroids <u>after projecting</u>, so that (as Dan Patterson mentioned in his post) the difference can be seen with the result in multipointPCS_1 (although not immediately apparent in this dataset – but there are differences, however minute in this case).

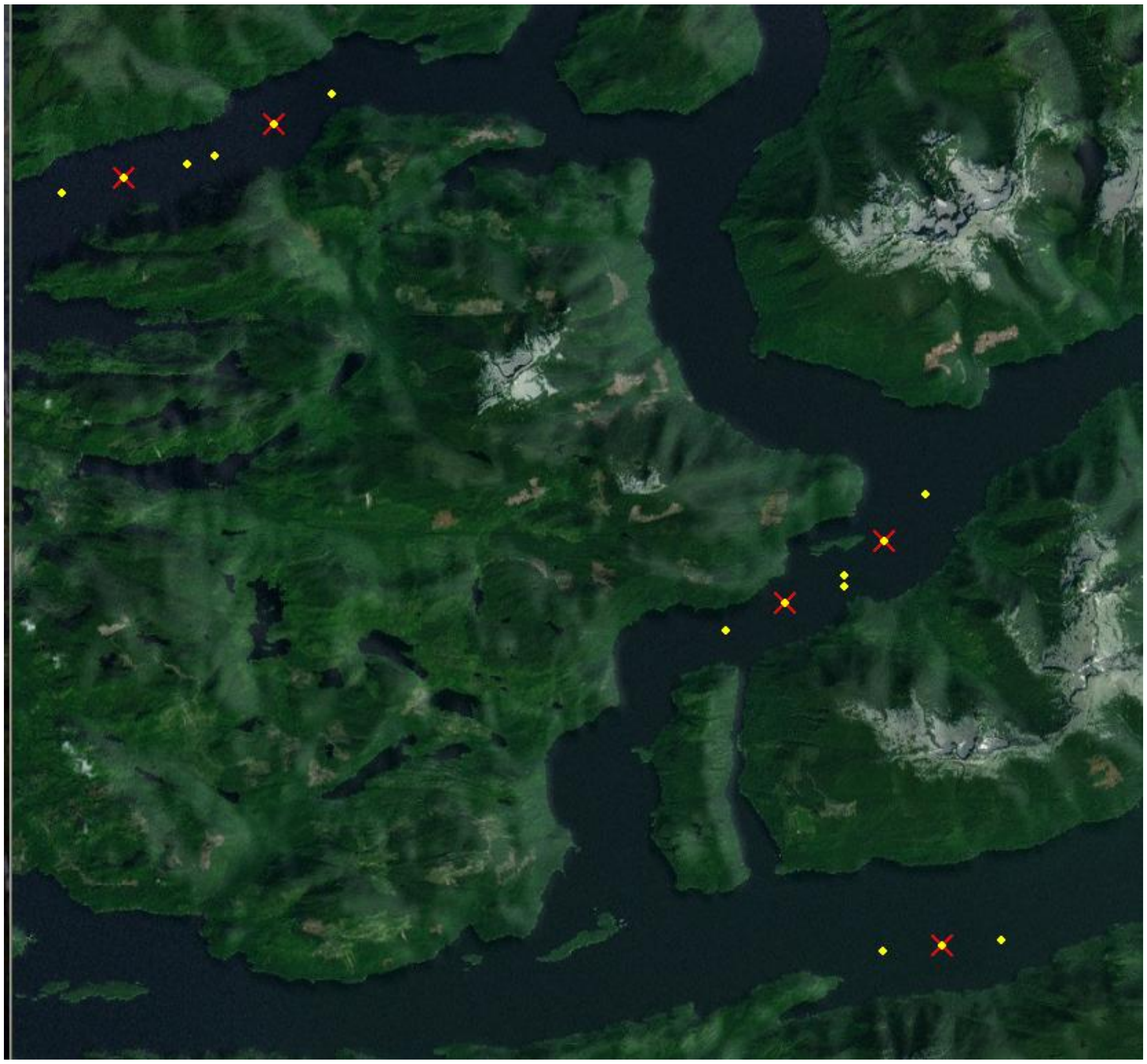
6- centroidsOnly

This was added code processing (see the code attached as well). Both multipointPCS_1 and multipointPCS_2 fcs contain multipoint geometry of the centroids (for comparison purposes) – everything was simply loaded on a single cursor passes. In case point geometry is desired, the fc called centroidsOnly was added as a 'user-friendlier' component. multipointPCS_2 was used to generate this output, copied to Point geometry via an insert cursor. (Another option would be to select every other record and execute the tool, Multipart To Singlepart.)

It is not necessary to delete any of these gdb-contained fcs – simply exit any apps that may be locking anything contained therein and execute the script at will – overwriteOutput is set True so that relevant fcs will be replaced.

As a 'final', more obvious depiction of the resultant centroid processing via these methods, see the next page (and you too can examine within ArcMap your own output as a result of using this script).

The yellow points are the multipoint result (including the centroid); the red 'X' is the centroid only extracted to its own point fc, projected of course. I am curious why the point sets are spread so far apart (over 1000 m), but of main concern in this exercise was to demonstrate working with geometry to produce centroids.



This concludes the test…

Enjoy,

Wayne