Extracting Data from Annoying Formats with Python

Jason Wise
Earth Data Scientist / Applications Developer / UAS Program Manager

Fall NEARC
(Northeast Arc Users Group conference)
October 2019

A Little About Notebooks

Notebook software lets you run code a little bit at a time and display the results.

It's great for experimenting or documenting your work.

In [4]:
```
public_content
```

Out[4]:
```
[<Item title:"Natuurrampen" type:Feature Layer Collection owner:Esri_NL_Content>,
 <Item title:"Collier County Emergency Services" type:Feature Layer Collection
owner:philsherman_collierbcc>,
 <Item title:"Intact Forest Landscapes (2000)" type:Feature Layer Collection owner:GlobalForestWatch>,
 <Item title:"Fire Perimeters" type:Feature Layer Collection owner:cferner_CALFIRE>,
 <Item title:"WRI_BeforeAndAfterPhotos" type:Feature Layer Collection owner:BuckEhler>]
```

The ArcGIS API for Python extends the Jupyter Notebook IDE to display ArcGIS Items in rich HTML notation. Thus, you can loop through each of the items in the search result and display it with thumbnails and metadata as shown below:

In [5]:
```
from IPython.display import display
for item in public_content:
    display(item)
```

Natuurrampen
Deze service toont de typen 'Natuurbrand' en 'Aardbeving' uit het thema 'Natuurrampen' van de risicokaart Feature Layer Collection by Esri_NL_Content
Last Modified: September 07, 2017
0 comments, 4,133 views

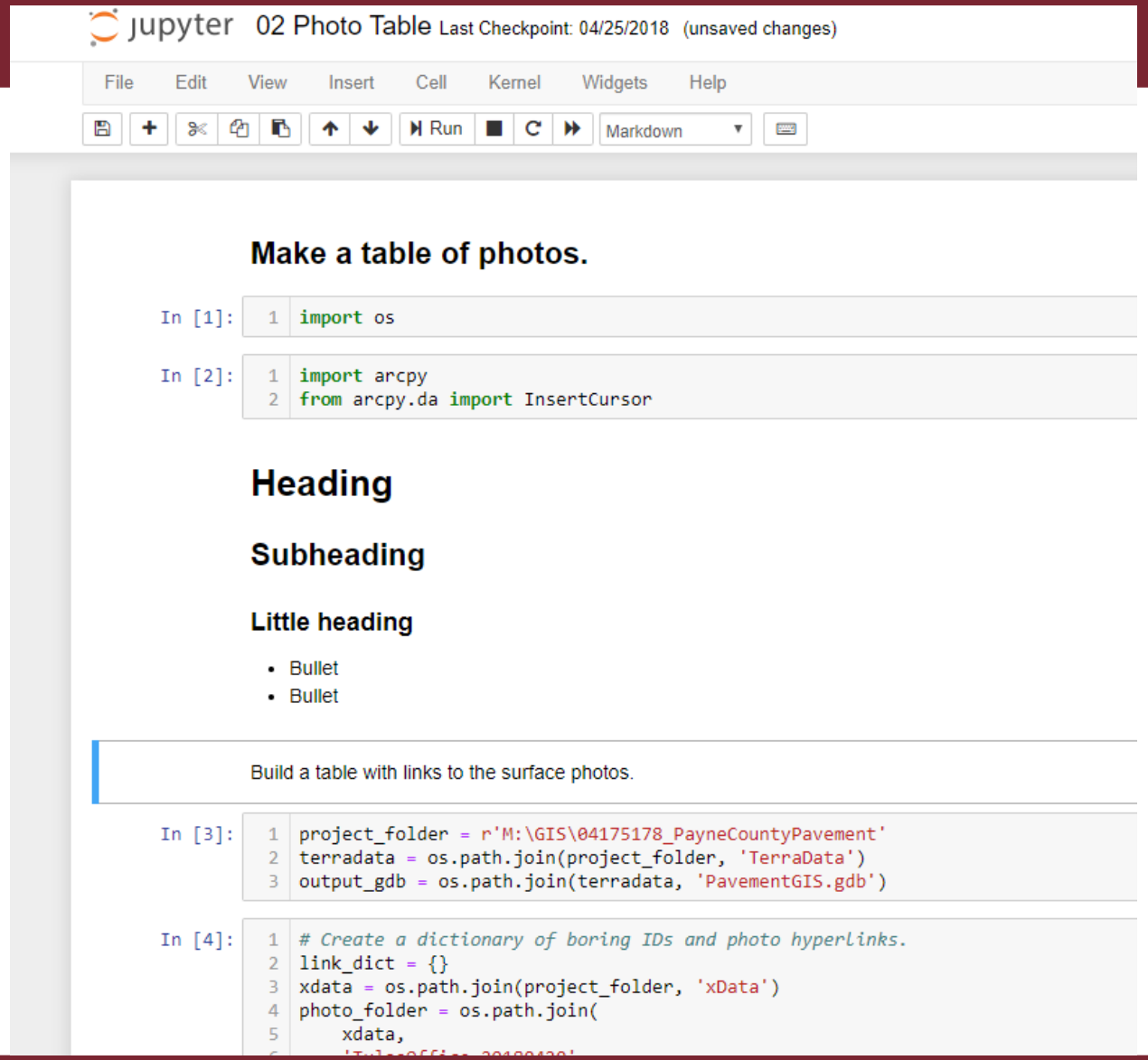
Collier County Emergency Services
Emergency Facilities in Collier County Feature Layer Collection by philsherman_collierbcc
Last Modified: May 22, 2014
0 comments, 4,436 views

Terracon

A Little About Notebooks

Jupyter Notebooks:

- Comes with ArcGIS
- Runs in a browser
- Enables nice formatting

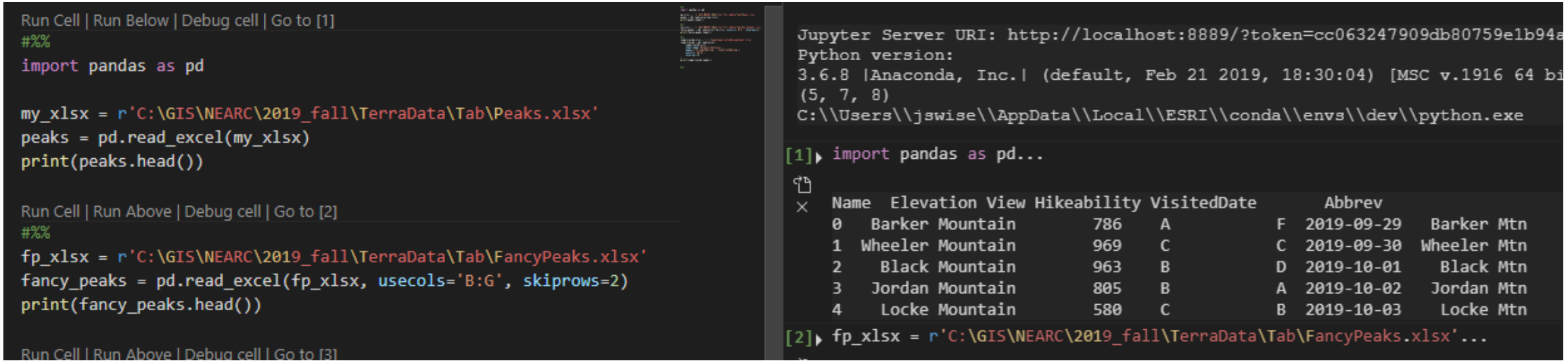


Terracon

A Little About Notebooks

Visual Studio Code:

- Real IDE + notebook functionality
- Stores code in regular Python files with special comments



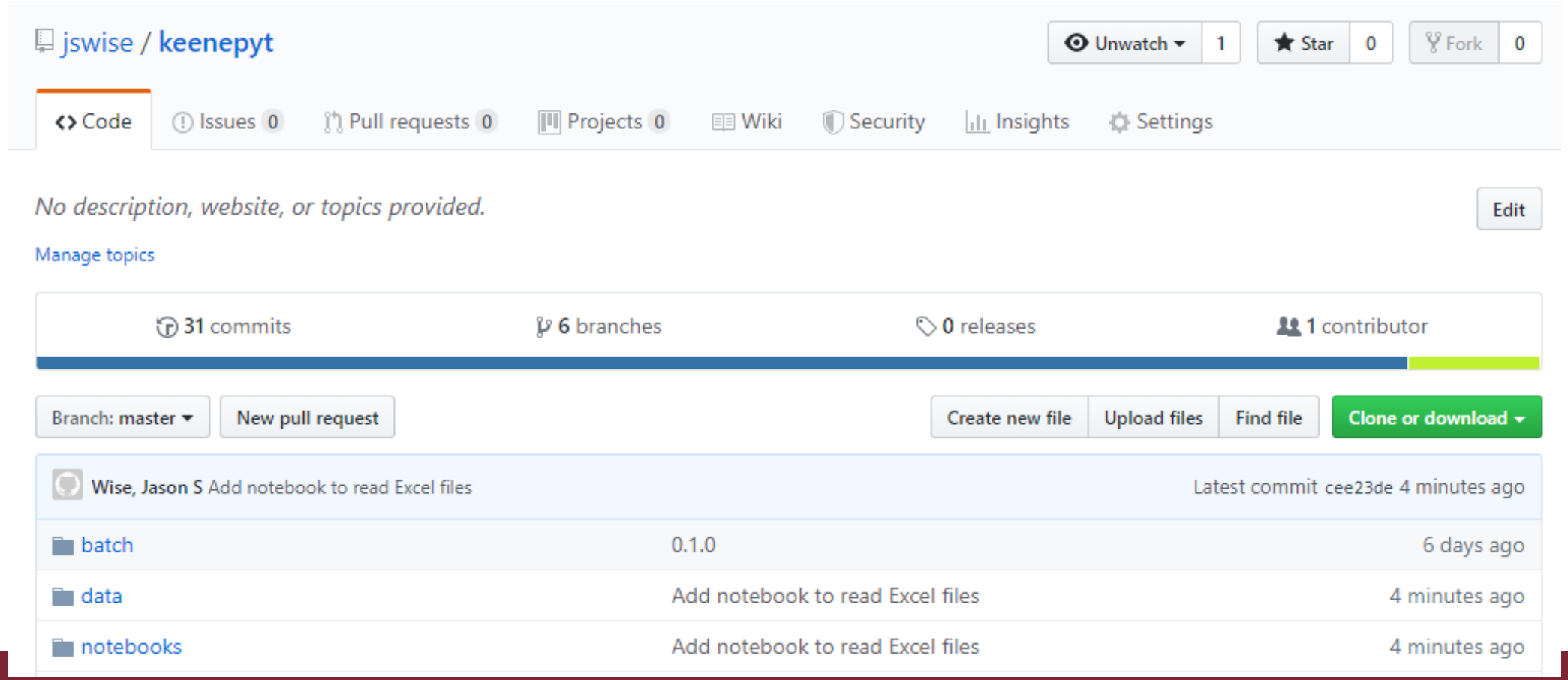
```
Run Cell | Run Below | Debug cell | Go to [1]
#%%
import pandas as pd

my_xlsx = r'C:\GIS\NEARC\2019_fall\TerraData\Tab\Peaks.xlsx'
peaks = pd.read_excel(my_xlsx)
print(peaks.head())

Run Cell | Run Above | Debug cell | Go to [2]
#%%
fp_xlsx = r'C:\GIS\NEARC\2019_fall\TerraData\Tab\FancyPeaks.xlsx'
fancy_peaks = pd.read_excel(fp_xlsx, usecols='B:G', skiprows=2)
print(fancy_peaks.head())

Run Cell | Run Above | Debug cell | Go to [3]
```

```
Jupyter Server URI: http://localhost:8889/?token=cc063247909db80759e1b94a
Python version:
3.6.8 |Anaconda, Inc.| (default, Feb 21 2019, 18:30:04) [MSC v.1916 64 bi
(5, 7, 8)
C:\\Users\\jswise\\AppData\\Local\\ESRI\\conda\\envs\\dev\\python.exe

[1] import pandas as pd...

   Name  Elevation View Hikeability VisitedDate       Abbrev
0   Barker Mountain        786    A           F  2019-09-29   Barker Mtn
1  Wheeler Mountain        969    C           C  2019-09-30  Wheeler Mtn
2    Black Mountain        963    B           D  2019-10-01    Black Mtn
3   Jordan Mountain        805    B           A  2019-10-02   Jordan Mtn
4    Locke Mountain        580    C           B  2019-10-03    Locke Mtn

[2] fp_xlsx = r'C:\GIS\NEARC\2019_fall\TerraData\Tab\FancyPeaks.xlsx'...
```

My GitHub

github.com/jswise/keenepyt

Spring 2019: I published *keenepyt* on Github for a NEARC workshop.
Fall 2019: I added a few things for this presentation.

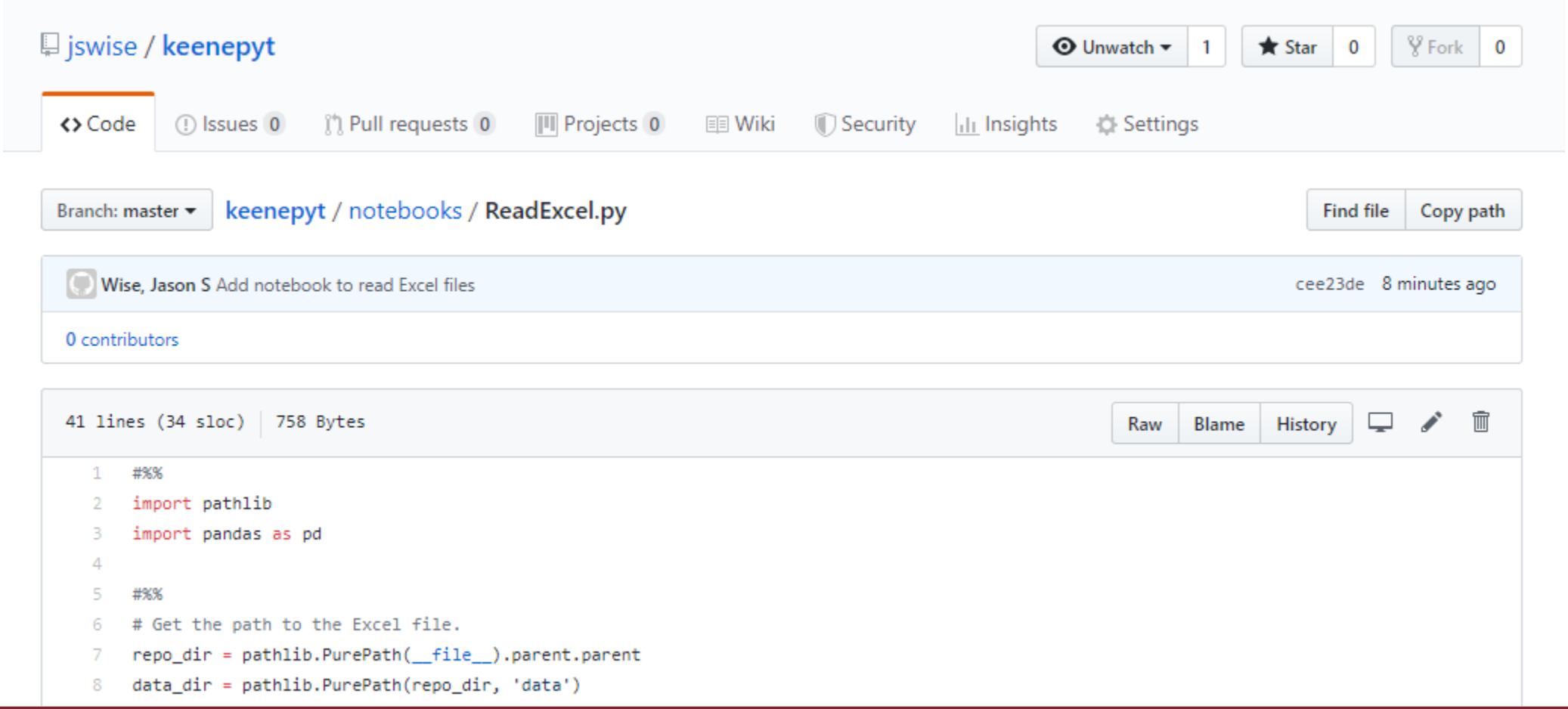


Terracon

My GitHub

github.com/jswise/keenepyt

Look at *notebooks/ReadExcel.py*.



My GitHub

github.com/jswise/keenepyt

Also see *tests/test_exkml.py* and the classes it imports (*KPWorkspace & ExKML*).

jswise / keenepyt

Code | Issues 0 | Pull requests 0 | Projects 0 | Wiki | Security

Branch: master ▾ keenepyt / tests / test_exkml.py

Wise, Jason S Extract popup info

0 contributors

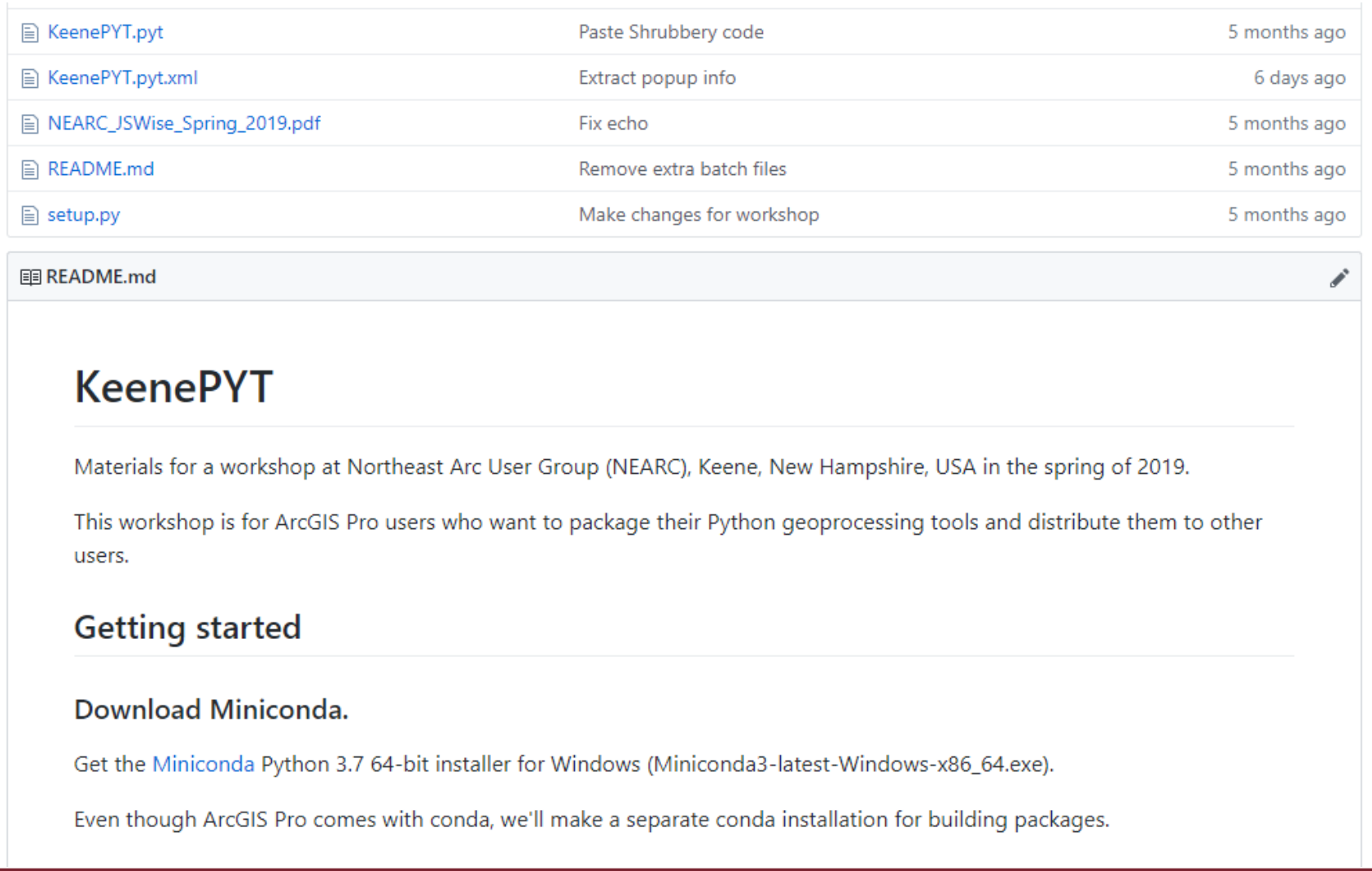
65 lines (52 sloc) | 1.77 KB

```
1  import logging
2  import os
3  import pathlib
4  import pytest
5
6  print('Importing ArcPy.')
7  import arcpy
8  from arcpy.da import SearchCursor # pylint: disable=no-name-in-module
9  print('Imported.')
10
11 from keenepyt.core.kpworkspace import KPWorkspace
12 from keenepyt.exkml import ExKML
```

My GitHub

github.com/jswise/keenepyt

Check out the spring workshop material.



KeenePYT.pyt	Paste Shrubbery code	5 months ago
KeenePYT.pyt.xml	Extract popup info	6 days ago
NEARC_JSWise_Spring_2019.pdf	Fix echo	5 months ago
README.md	Remove extra batch files	5 months ago
setup.py	Make changes for workshop	5 months ago

README.md

KeenePYT

Materials for a workshop at Northeast Arc User Group (NEARC), Keene, New Hampshire, USA in the spring of 2019.

This workshop is for ArcGIS Pro users who want to package their Python geoprocessing tools and distribute them to other users.

Getting started

Download Miniconda.

Get the Miniconda Python 3.7 64-bit installer for Windows (Miniconda3-latest-Windows-x86_64.exe).

Even though ArcGIS Pro comes with conda, we'll make a separate conda installation for building packages.

Mystery KMZ

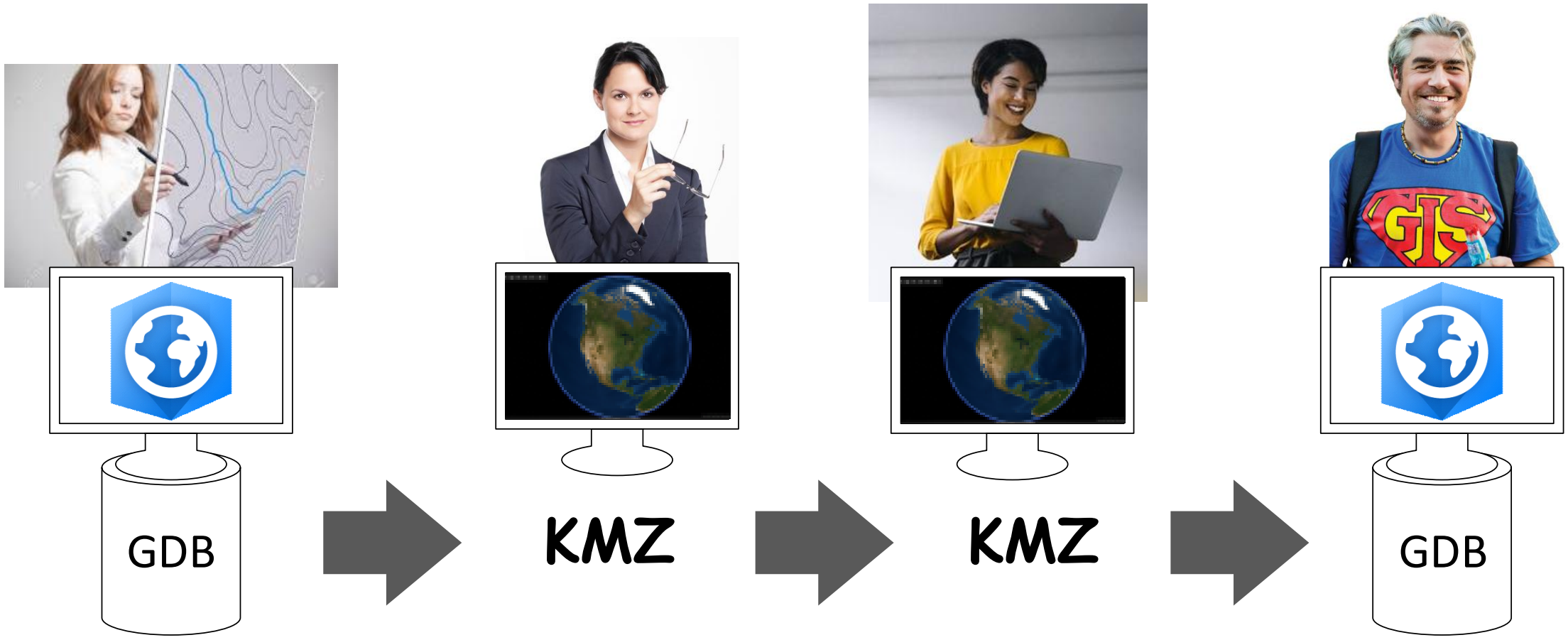
When we import a KMZ, why do its attributes disappear?

Points

PopupInfo
<html xmlns:fo="http://www.w3.org/1999/XSL/Format" xmlns:msxsl="urn:schemas-microsoft-com:xslt">
<html xmlns:fo="http://www.w3.org/1999/XSL/Format" xmlns:msxsl="urn:schemas-microsoft-com:xslt">
<html xmlns:fo="http://www.w3.org/1999/XSL/Format" xmlns:msxsl="urn:schemas-microsoft-com:xslt">
<html xmlns:fo="http://www.w3.org/1999/XSL/Format" xmlns:msxsl="urn:schemas-microsoft-com:xslt">
<html xmlns:fo="http://www.w3.org/1999/XSL/Format" xmlns:msxsl="urn:schemas-microsoft-com:xslt">
<html xmlns:fo="http://www.w3.org/1999/XSL/Format" xmlns:msxsl="urn:schemas-microsoft-com:xslt">
<html xmlns:fo="http://www.w3.org/1999/XSL/Format" xmlns:msxsl="urn:schemas-microsoft-com:xslt">

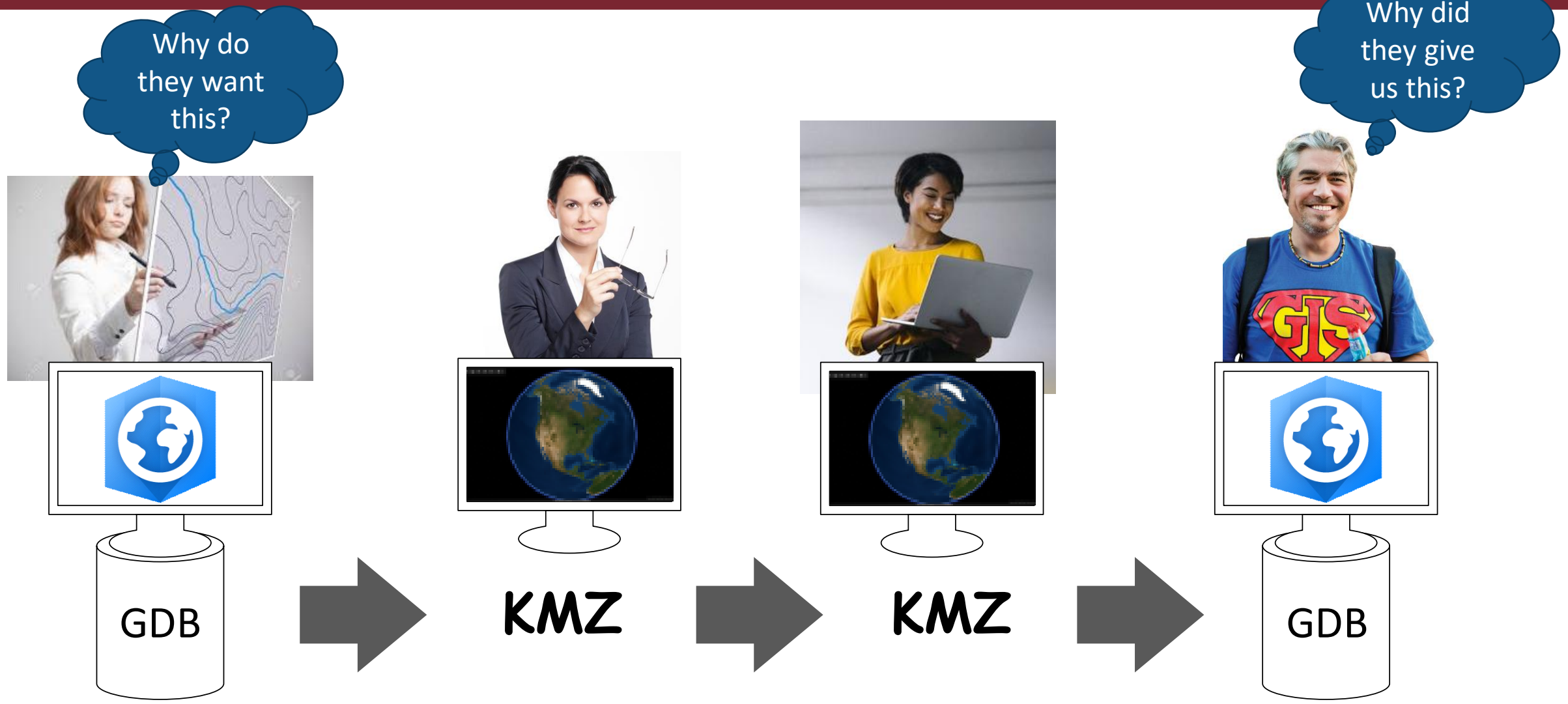
Mystery KMZ

Project managers love KMZ files.



Mystery KMZ

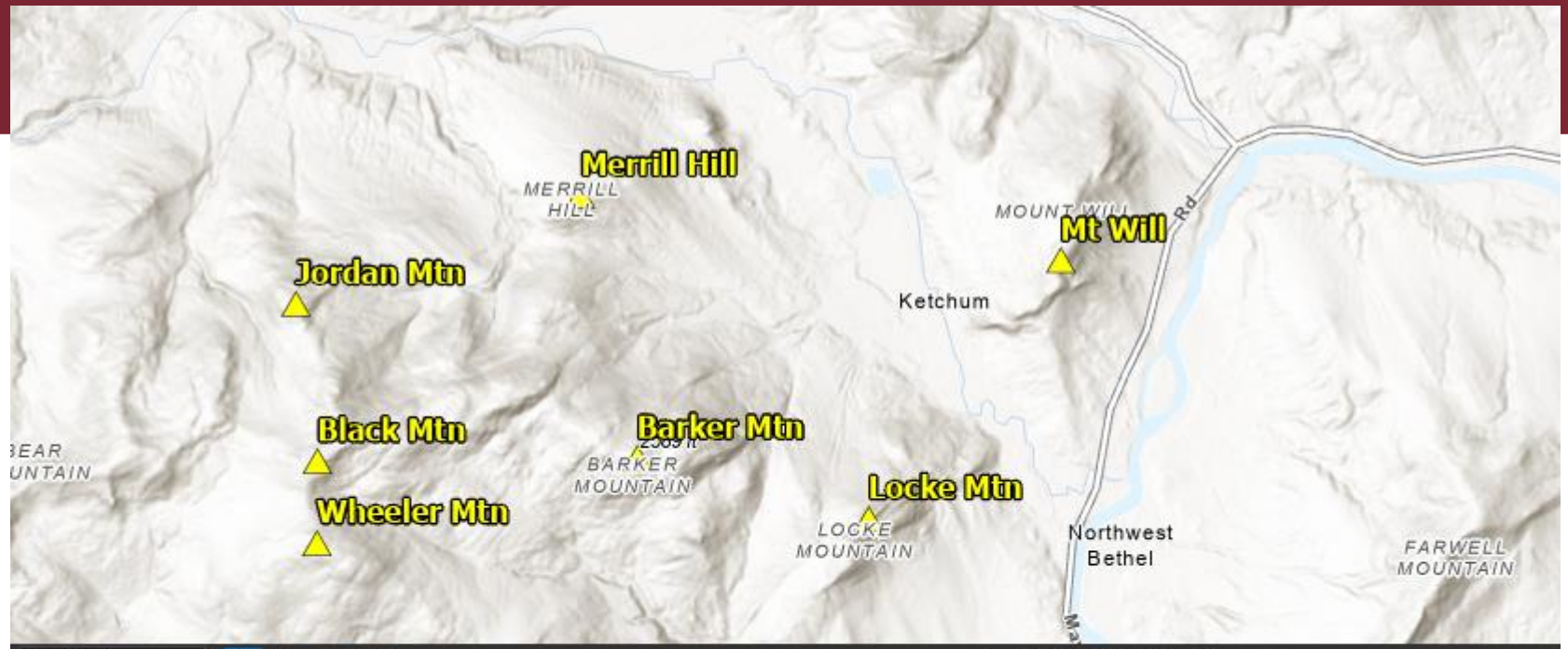
Project managers love KMZ files.



Terracon

Mystery KMZ

The original feature class:



Peaks

Field: Add Delete Calculate | Selection: Zoom To Switch Clear Delete Copy

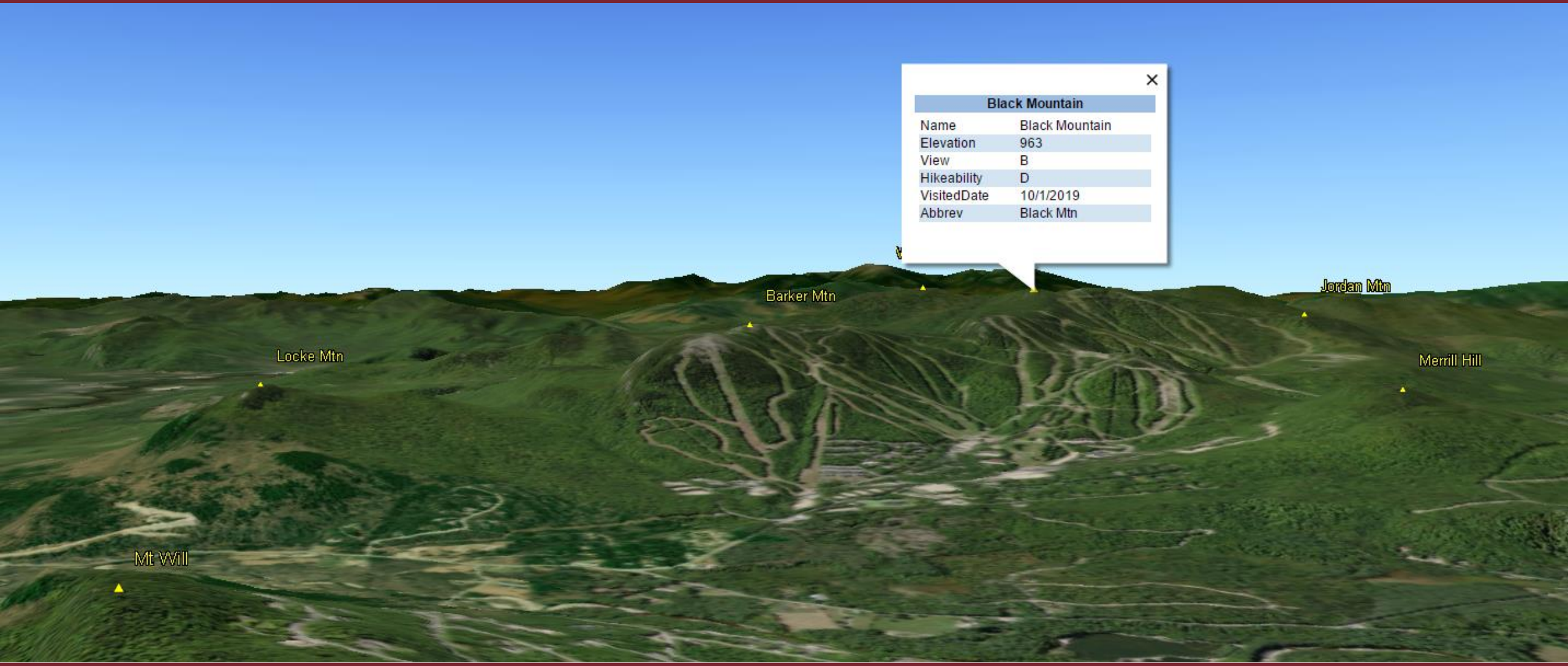
OBJECTID	Shape	Name	Elevation	View	Hikeability	VisitedDate	Abbrev
1	Point	Barker Mountain	786	A	F	9/29/2019	Barker Mtn
2	Point	Wheeler Mountain	969	C	C	9/30/2019	Wheeler Mtn
3	Point	Black Mountain				2019	Black Mtn
4	Point	Jordan Mountain				2019	Jordan Mtn
5	Point	Locke Mountain	580	C	B	10/3/2019	Locke Mtn
6	Point	Mount Will	526	A	B	10/4/2019	Mt Will
7	Point	Merrill Hill	572	A	A	10/5/2019	Merrill Hill

Note *Name* and *Abbrev* fields.

Terracon

Mystery KMZ

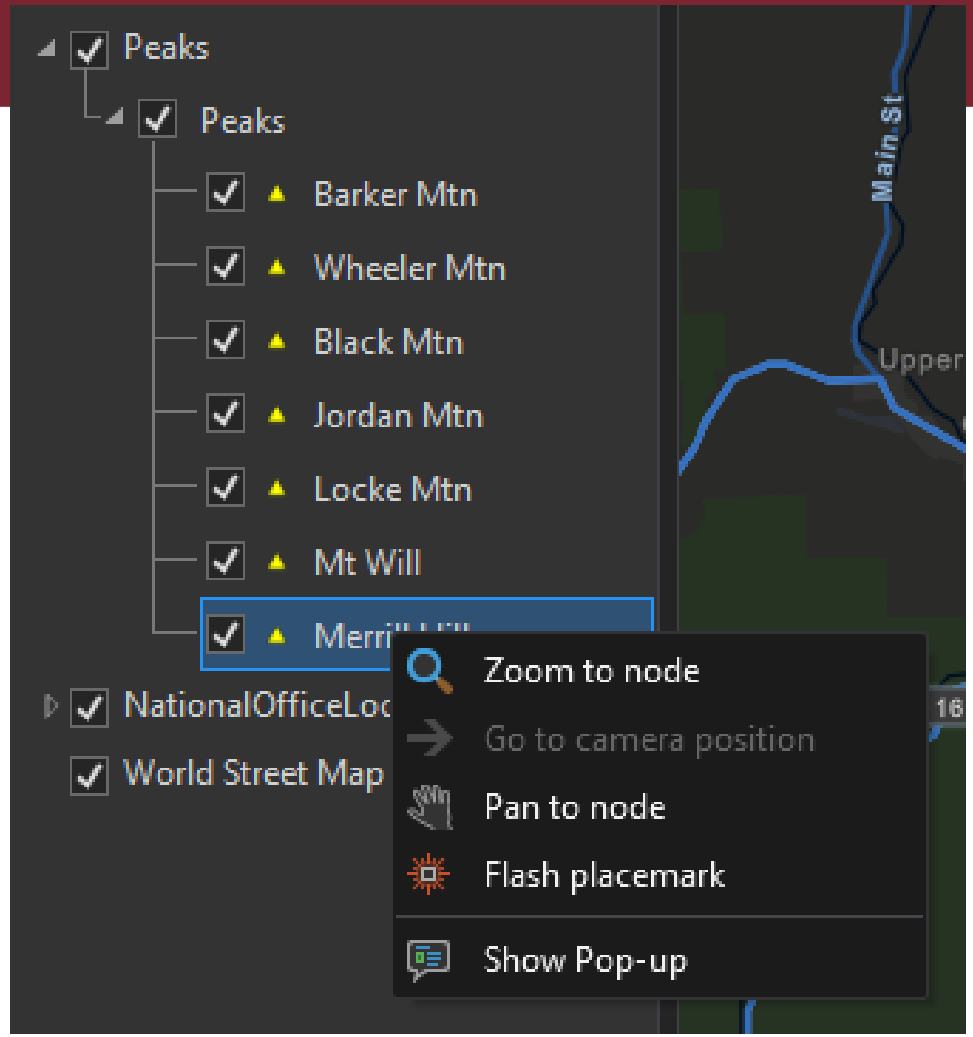
The KMZ:



Mystery KMZ

The extracted feature class:

You can load it into Pro (new feature?), but you can't do much with it.



Mystery KMZ

The extracted feature class:

Points

eld: Add Delete Calculate Selection: Zoom To Switch Clear Delete Copy

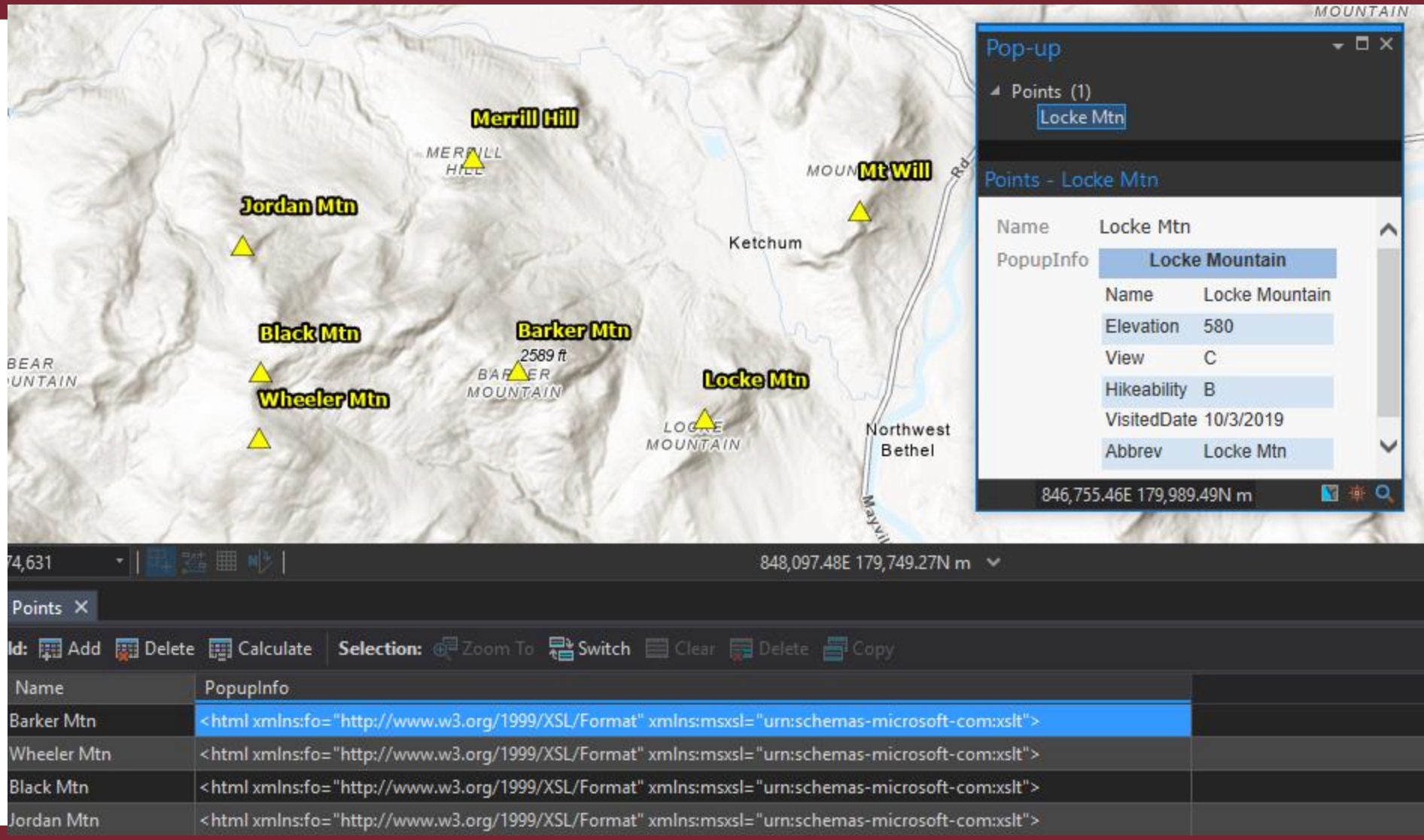
OID	Shape	Name	FolderPath	SymbolID	AltMode	Base	Snippet	PopupInfo	HasLabel	LabelID
1	Point Z	Barker Mtn	Peaks/Peaks	0	-1	0		<html xmlns:fo="htt...	-1	0
2	Point Z	Wheeler Mtn			-1	0		<html xmlns:fo="htt...	-1	0
3	Point Z	Black Mtn			-1	0		<html xmlns:fo="htt...	-1	0
4	Point Z	Jordan Mtn	Peaks/Peaks	0	-1	0		<html xmlns:fo="htt...	-1	0
5	Point Z	Locke Mtn	Peaks/Peaks	0	-1	0		<html xmlns:fo="htt...	-1	0
6	Point Z	Mt Will	Peaks/Peaks	0	-1	0		<html xmlns:fo="htt...	-1	0
7	Point Z	Merrill Hill	Peaks/Peaks	0	-1	0		<html xmlns:fo="htt...	-1	0

Name is really *Abbrev*.

Terracon

Mystery KMZ

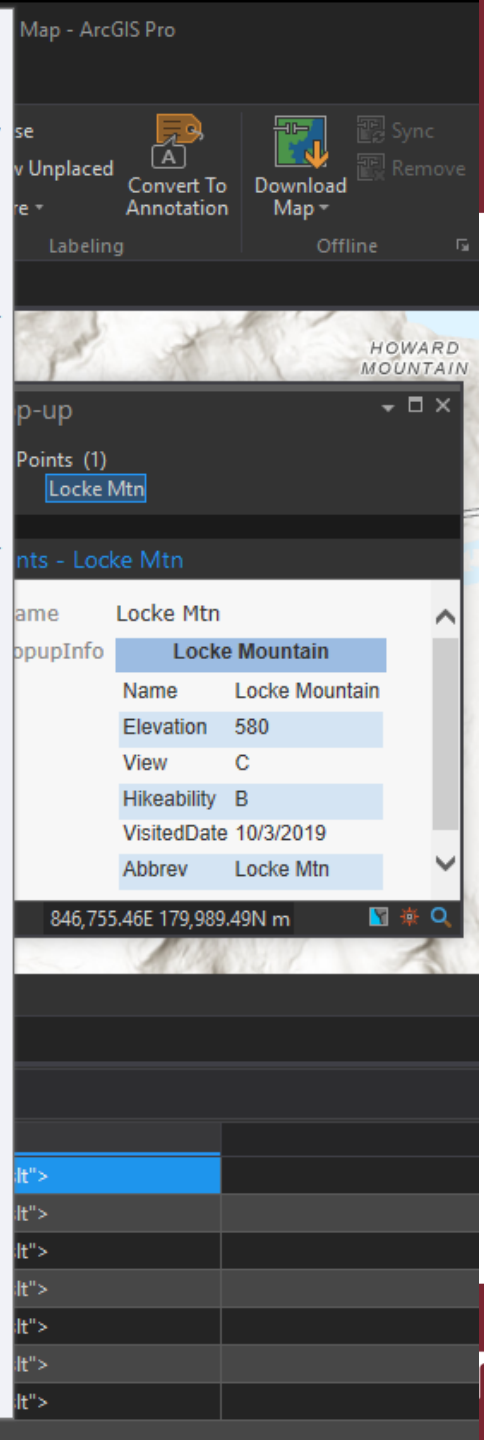
We have a popup, but where are the attributes?



Mystery KMZ

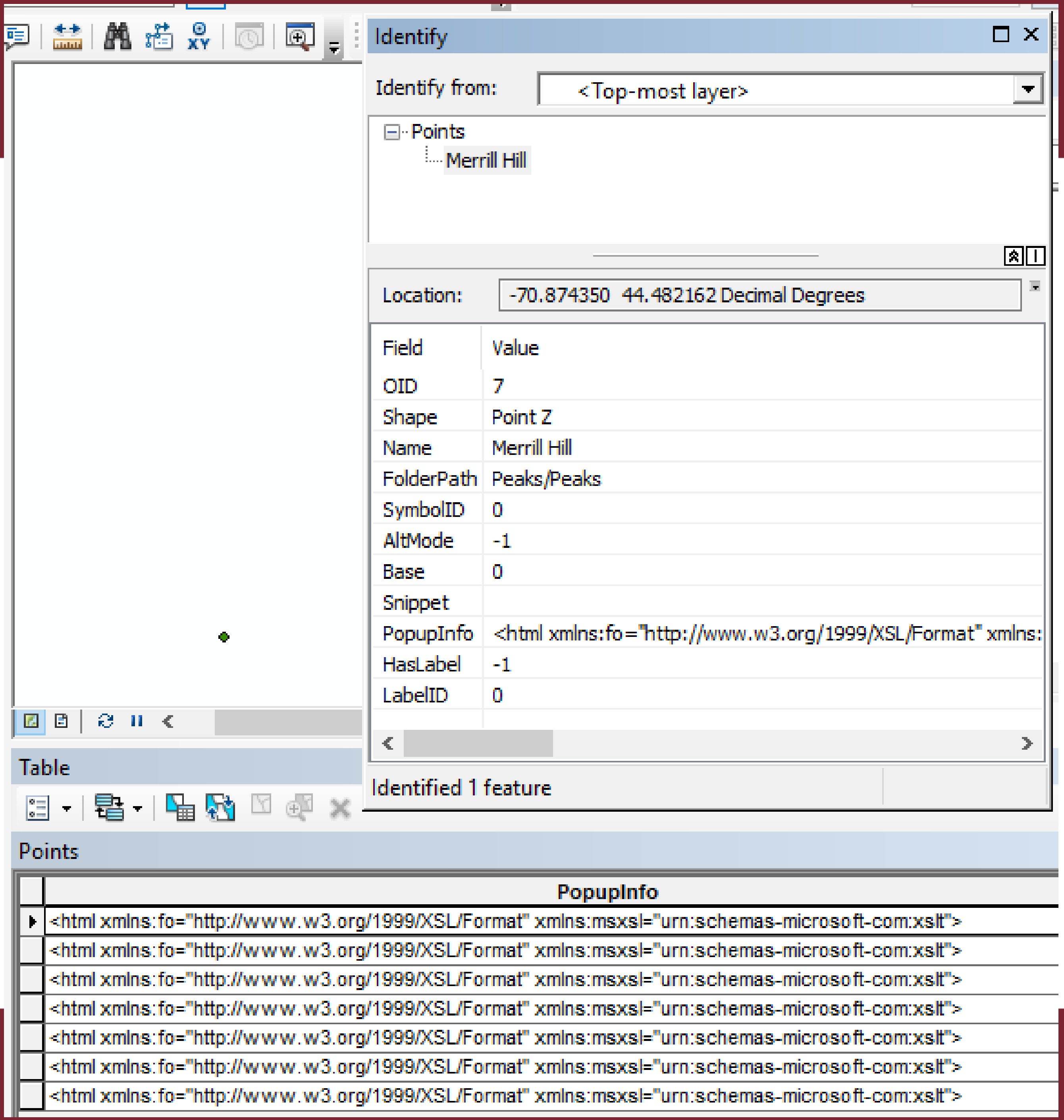
HTML with carriage returns


```
<html xmlns:fo="http://www.w3.org/1999/XSL/Format" xmlns:msxsl="urn:schemas-microsoft-com:xslt">
<head>
<META http-equiv="Content-Type" content="text/html">
<meta http-equiv="content-type" content="text/html; charset=UTF-8">
</head>
<body style="margin:0px 0px 0px 0px;overflow:auto;background:#FFFFFF;">
<table style="font-family:Arial,Verdana,Times;font-size:12px;text-align:left;width:100%;border-collapse:collapse;padding:3px 3px 3px 3px">
<tr style="text-align:center;font-weight:bold;background:#9CBCE2">
<td>Barker Mountain</td>
</tr>
<tr>
<td>
<table style="font-family:Arial,Verdana,Times;font-size:12px;text-align:left;width:100%;border-spacing:0px; padding:3px 3px 3px 3px">
<tr>
<td>Name</td>
<td>Barker Mountain</td>
</tr>
<tr bgcolor="#D4E4F3">
<td>Elevation</td>
<td>786</td>
</tr>
<tr>
<td>View</td>
<td>A</td>
</tr>
<tr bgcolor="#D4E4F3">
<td>Hikeability</td>
<td>F</td>
</tr>
<tr>
<td>VisitedDate</td>
<td>9/29/2019</td>
</tr>
<tr bgcolor="#D4E4F3">
<td>Abbrev</td>
<td>Barker Mtn</td>
</tr>
</table>
</td>
</tr>
</table>
</body>
</html>
```



Mystery KMZ

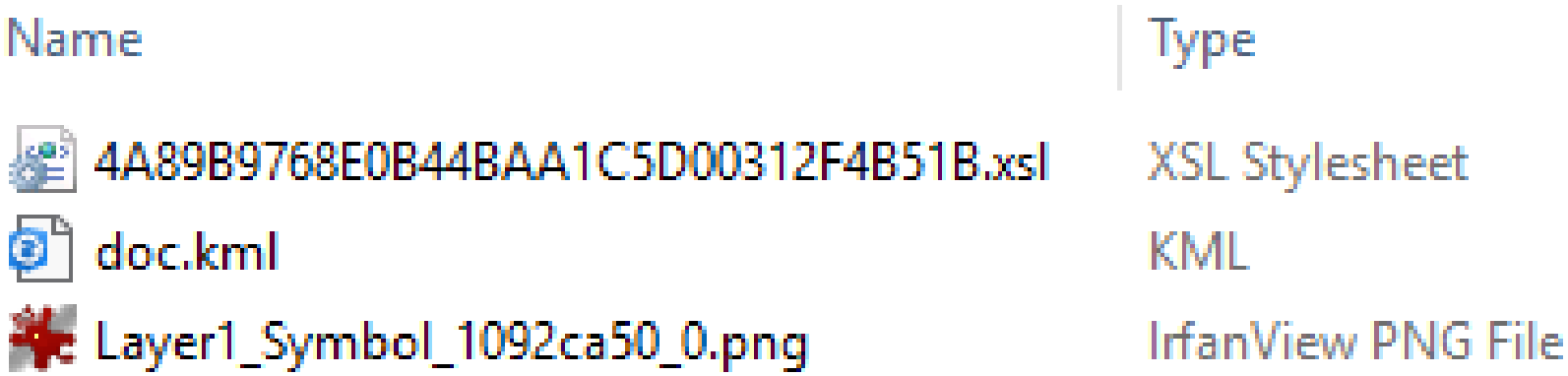
It’s even more mysterious in ArcMap.



Mystery KMZ

KMZ primer

A KMZ file is a zipped KML file.



Mystery KMZ

KMZ primer

A KML file is XML.

It’s not a table and doesn’t have fields.

```
<?xml version="1.0" encoding="UTF-8"?>
<kml xmlns="http://www.opengis.net/kml/2.2" xmlns:gx="http://www.google.com/kml/ext/2.2" xmlns:xsi="
http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="http://www.opengis.net/kml/2.2 http://schemas.opengis.net/kml/2.2.0/ogckml22.xsd
 http://www.google.com/kml/ext/2.2 http://code.google.com/apis/kml/schema/kml22gx.xsd">
<Document id="Peaks">
  <name>Peaks</name>
  <snippet></snippet>
  <description><![CDATA[Peaks]]></description>
  <Folder id="FeatureLayer1">
    <name>Peaks</name>
    <snippet></snippet>
    <description><![CDATA[Peaks]]></description>
    <Placemark id="ID_10000">
      <name>Barker Mtn</name>
      <snippet></snippet>
      <description><![CDATA[<html xmlns:fo="http://www.w3.org/1999/XSL/Format"
      xmlns:msxsl="urn:schemas-microsoft-com:xslt">

<head>

<META http-equiv="Content-Type" content="text/html">

<meta http-equiv="content-type" content="text/html; charset=UTF-8">

</head>

<body style="margin:0px 0px 0px 0px;overflow:auto;background:#FFFFFF;">
```

Terracon

Mystery KMZ

KMZ primer

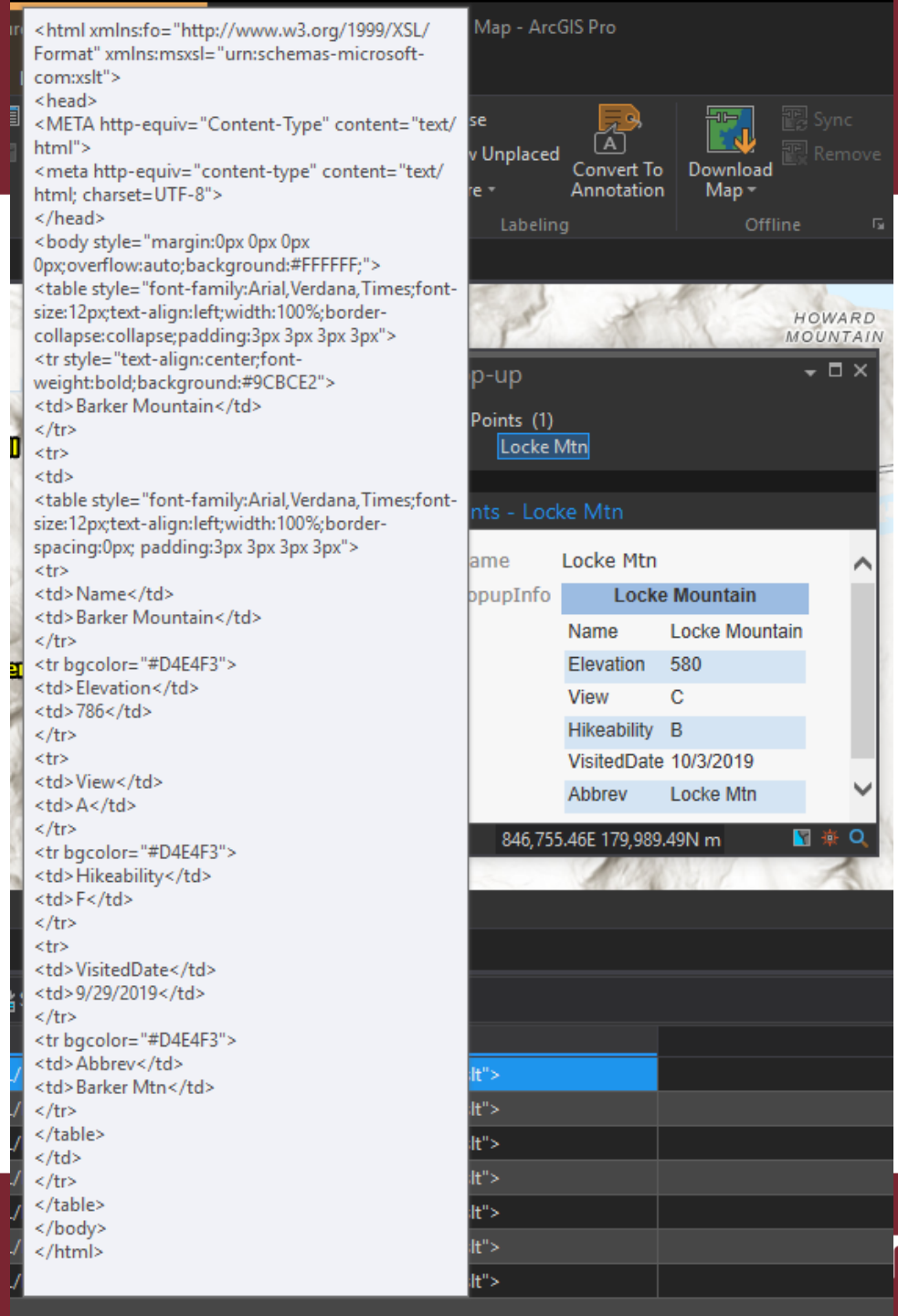
A popup is HTML inside the XML (a table for each feature).

```
<td>Name</td>
<td>Barker Mountain</td>
</tr>
<tr bgcolor="#D4E4F3">
<td>Elevation</td>
<td>786</td>
</tr>
<tr>
<td>View</td>
<td>A</td>
</tr>
<tr bgcolor="#D4E4F3">
<td>Hikeability</td>
<td>F</td>
</tr>
```

Mystery KMZ

KMZ primer

That’s why we end up with this instead of a real table.



Mystery KMZ

Python code (github.com/jswise/keenepyt)

Using Python, we can convert the HTML into feature attributes.

First, import the ElementTree library for working with XML & HTML.

```
4    import xml.etree.ElementTree as ET
```

Mystery KMZ

Python code (github.com/jswise/keenepyt)

Find, fix, and parse the HTML.

```
82          # Fix HTML that doesn't include the "body" tag.
83          if '<body' not in popup_html:
84              popup_html = '<body>' + popup_html + '</body>'
85
86          # Load the important part into an ElementTree object.
87          body_start = popup_html.find('<body')
88          body_end = popup_html.find('</body>')
89          body_text = popup_html[body_start:body_end + 7]
90          body_text = body_text.replace('<Null>', '')
91          body_text = body_text.replace('&', '&amp;')
92          try:
93              body_element = ET.fromstring(body_text)
94          except Exception as e:
95              self.raise_error('Failed to load HTML. {}'.format(e))
```

Mystery KMZ

Python code (github.com/jswise/keenepyt)

```
105         # Convert the table to a dictionary.
106         rows = table.findall('tr')
107         popup_dict = {}
108         for row in rows:
109             columns = row.findall('td')
110             raw_key = columns[0].text
111             key = str(raw_key).replace(' ', "_")
112             bold = columns[1].findall('b')
113             if len(bold) > 0:
114                 raw_val = bold[0].text
115             else:
116                 raw_val = columns[1].text
117             if raw_val is None:
118                 raw_val = ''
119             val = str(raw_val).replace('"', "'")
120             popup_dict[key] = val
```

Mystery KMZ

Python code (github.com/jswise/keenepyt)

```
105         # Convert the table to a dictionary.
106         rows = table.findall('tr')
107         popup_dict = {}
108         for row in rows:
                columns = row.findall('td')
                raw_key = columns[0].text
                key = str(raw_key).replace(' ', "_")
112             bold = columns[1].findall('b')
113             if len(bold) > 0:
114                 raw_val = bold[0].text
115             else:
116                 raw_val = columns[1].text
117             if raw_val is None:
118                 raw_val = ''
119             val = str(raw_val).replace('"', "'")
120             popup_dict[key] = val
```

pandas.read_html would also work.

Mystery KMZ

Reclaimed data

Add new fields with the extracted data.

Name	Elevation	View	Hikeability	VisitedDate	Abbrev
Barker Mountain	786	A	F	9/29/2019	Barker Mtn
Wheeler Mountain	969	C	C	9/30/2019	Wheeler Mtn
Black Mountain		B	D	10/1/2019	Black Mtn
Jordan Mountain	805	B	A	10/2/2019	Jordan Mtn
Locke Mountain	580	C	B	10/3/2019	Locke Mtn
Mount Will	526	A	B	10/4/2019	Mt Will
Merrill Hill	572	A	A	10/5/2019	Merrill Hill

Name is back!

Disclaimer: I haven't been on these mountains; the ratings are bogus.

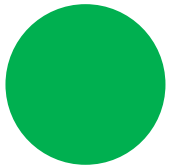
Excel Files

The terrain varies.

Terracon

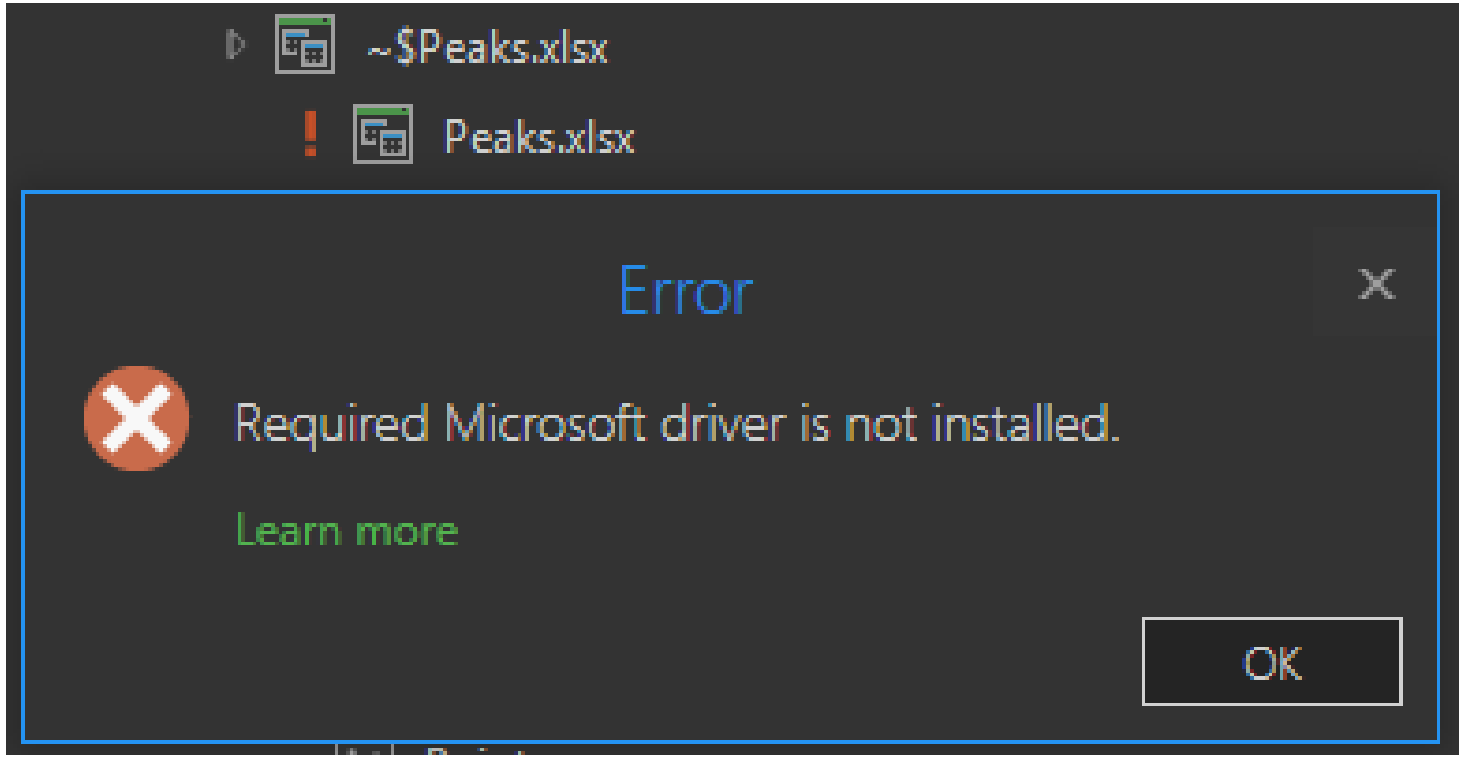
Excel Files

Well-behaved spreadsheets



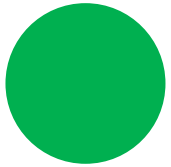
Easiest

Pro requires the Microsoft Access Database Engine driver.



Excel Files

Well-behaved spreadsheets



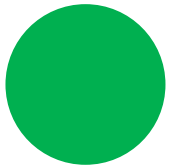
Easiest

Pro requires the Microsoft Access Database Engine driver.

```
C:\Users\jswise\Downloads>AccessDatabaseEngine.exe /quiet
```

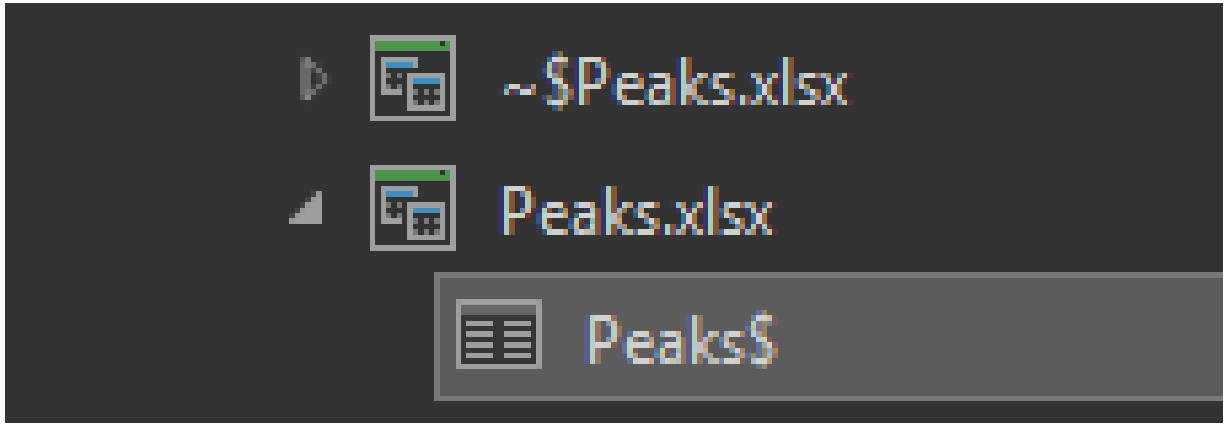
Excel Files

Well-behaved spreadsheets



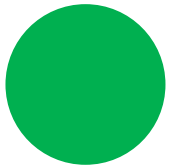
Easiest

Open the file in ArcGIS…



Excel Files

Well-behaved spreadsheets



Easiest

…or use Pandas,
and skip the driver!

```
Run Cell | Run Below | Debug cell | Go to [2]
#%%
import pandas as pd

my_xlsx = r'C:\GIS\NEARC\2019_fall\TerraData\Tab\Peaks.xlsx'
peaks = pd.read_excel(my_xlsx)
print(peaks.head())

```

Python Interactive

```
[2] import pandas as pd...
```

	Name	Elevation	View	Hikeability	VisitedDate		Abbrev
0	Barker Mountain	786	A	F	2019-09-29		Barker Mtn
1	Wheeler Mountain	969	C	C	2019-09-30		Wheeler Mtn
2	Black Mountain	963	B	D	2019-10-01		Black Mtn
3	Jordan Mountain	805	B	A	2019-10-02		Jordan Mtn
4	Locke Mountain	580	C	B	2019-10-03		Locke Mtn

Excel Files

Strangely-formatted sheets

More Difficult

Extra rows & columns?

	A	B	C	D	E	F	G	H
1		Nearby Peaks						
2		Fields		More Fields			One More	
3		Name	Elevation	View	Hikeability	VisitedDate	Abbrev	
4		Barker Mountain	786	A	F	9/29/2019	Barker Mtn	
5		Wheeler Mountain	969	C	C	9/30/2019	Wheeler Mtn	
6		Black Mountain	963	B	D	10/1/2019	Black Mtn	
7		Jordan Mountain	805	B	A	10/2/2019	Jordan Mtn	
8		Locke Mountain	580	C	B	10/3/2019	Locke Mtn	
9		Mount Will	526	A	B	10/4/2019	Mt Will	
10		Merrill Hill	572	A	A	10/5/2019	Merrill Hill	
11								

Excel Files

Strangely-formatted sheets

More Difficult

Pandas can handle it.

	A	B	C
1			Near
2		Fields	
3		Name	Elevation
4		Barker Mountain	786
5		Wheeler Mountain	969
6		Black Mountain	963
7		Jordan Mountain	805
8		Locke Mountain	580
9		Mount Will	526
10		Merrill Hill	572
11			

```
Run Cell | Run Above | Debug cell | Go to [8]
#%%
fp_xlsx = r'C:\GIS\NEARC\2019_fall\TerraData\Tab\FancyPeaks.xlsx'
fancy_peaks = pd.read_excel(fp_xlsx, usecols='B:G', skiprows=2)
print(fancy_peaks.head())

```

Python Interactive

```
[8] fp_xlsx = r'C:\GIS\NEARC\2019_fall\TerraData\Tab\FancyPeaks.xlsx'...

  Name  Elevation View Hikeability VisitedDate       Abbrev
0   Barker Mountain         786    A           F  2019-09-29   Barker Mtn
1  Wheeler Mountain         969    C           C  2019-09-30  Wheeler Mtn
2    Black Mountain         963    B           D  2019-10-01    Black Mtn
3   Jordan Mountain         805    B           A  2019-10-02   Jordan Mtn
4    Locke Mountain         580    C           B  2019-10-03    Locke Mtn
```

Terracon

Excel Files

Even stranger sheets

Difficult

Unhide Committed | Unhide Filters | Unhide Sidewalks | Unhide Zones | Unhide Distresses | Unhide ESA Calcs

Hide Committed | Hide Filters | Hide Sidewalks | Hide Zones | Hide Distresses | Hide ESA Calcs

Network Analysis Summary - Five Year Rehab Plan Development

Current PCI Date: 3/19/2018 | Analysis Start Date: 7/1/2018 (MM/DD/YYYY) | % of Budget Dedicated to Surface Treatments: 0.000 % | Restore PCI to Previous Date: 3/19/2018

Update PCI | Restore PCI | Run 10X Profile | Run Control

Program Year	Annual Budget ($)	Avg. Annual Budget ($)	Calendar Year	Block Count	Annual Expenditure ($)	Pavement Costs ($)	Peripheral Concrete Costs($)	Miles	PCI	Backlog (%)
Avg:	3,090,000	3090	2017	628	59,338,250	59,338,250	0	241.4	64	16.3
1	3,090,000		2018	24	3,082,687	3,082,687	0	6.8	64	
2	3,090,000		2019	17	3,085,320	3,085,320	0	7.0	64	
3	3,090,000		2020	30	3,088,590	3,088,590	0	6.4	64	
4	3,090,000		2021	15	3,084,275	3,084,275	0	7.3	64	
5	3,090,000		2022	24	3,085,821	3,085,821	0	7.2	64	18.0
			Totals:	110	15,426,693	15,426,693	0	34.6		

Projects Pavement Condition Summary (Based on PCI Aged to Current PCI Date)

Backlog: 16.3 18.5

Aged Network PCI: 64 64

IMS Easy Street Analysis

Create Inventory | Rehab Plan By Seg | Rehab Plan By Year | Need Year Analysis

GISID	Strength Rating	Condition Rating	LADD	NLAD	Override PCI (OPCI)	OPCI Date	Current PCI (CPCI)	Aged PCI (APCI)	OPCI Check	Project ID	Project Description	Prjct Block Count	Prjct Length (ft)	Prjct Area (yd2)	Prjctect PCI (P-CPCI)	Prjct Cndtn Rating	Prjct Strength Code	Prjct FunCl Code	Prjct Pavetype Code	Need Year	Committed Year	Year of First Selection	Segment Rehab Results	Rehab Activity Code	Rehab Activity	Avg Unit Rate ($/yd2)	Segment Peripheral Concrete Costs ($)	Segment Pavement Cost ($)	Segment Total Cost ($)	Project Cost ($)	Five Year Post Rehab PCI
5149	Mod	Poor	43	28			39	38		1000	1000 - KIRK AVE + Others	15	5,135	12,323	44	Marginal	2	5	1	1	0	0	Fall Thru Yr 1						0	0	29
5150	Mod	Poor	57	24			35	34		1000	1000 - KIRK AVE + Others	15	5,135	12,323	44	Marginal	2	5	1	1	0	0	Fall Thru Yr 1						0	0	25
5156	Mod	Poor	54	16			36	35		1000	1000 - KIRK AVE + Others	15	5,135	12,323	44	Marginal	2	5	1	1	0	0	Fall Thru Yr 1						0	0	26
7422	Weak	Fair	46	4			51	51		1020	1020 - 122ND ST	1	3,807	9,138	51	Fair	1	5	1	1	0	0	Fall Thru Yr 1						0	0	41
7421	Mod	Excellent	0	1			94	93		1040	1040 - 122ND ST	1	1,724	4,138	94	Excellent	2	5	1	6	0	0	Not Selected						0	0	88
7753	Mod	Poor	62	17			30	29		1060	1060 - 122ND ST	1	4,983	13,154	30	Poor	2	5	1	1	0	2	Selected Yr 2	56	FWM + Thick Overlay (> 2.0 - 3.0) + Strctrl Ptch	29.00	0	381,466	381,466	381,466	90
7718	Mod	Good	17	13			64	63		1080	1080 - 143RD ST + Others	5	2,794	8,549	74	V Good	2	3	1	2	0	2	Selected Yr 2	20	Surface Trtmnt / Chip Seal	5.40	0	11,324	11,324	46,165	82
7717	Mod	V Good	9	17			76	75		1100	1100 - 140TH ST	2	9,725	28,008	66	Good	2	3	1	3	0	0	Fall Thru Yr 3						0	0	68
7716	Mod	Fair	29	26			57	56		1100	1100 - 140TH ST	2	9,725	28,008	66	Good	2	3	1	3	0	0	Fall Thru Yr 3						0	0	45
1006	Mod	V Good	14	18			76	75		1120	1120 - 140TH ST	3	10,626	30,604	80	V Good	2	3	1	6	0	0	Not Selected						0	0	68
1005	Mod	V Good	7	17			80	79		1120	1120 - 140TH ST	3	10,626	30,604	80	V Good	2	3	1	6	0	0	Not Selected						0	0	73
7750	Mod	V Good	7	14			83	83		1120	1120 - 140TH ST	3	10,626	30,604	80	V Good	2	3	1	6	0	0	Not Selected						0	0	77
7739	Mod	V Good	8	10			84	84		1140	1140 - 140TH ST	3	10,602	30,534	77	V Good	2	3	1	4	0	0	Fall Thru Yr 4						0	0	78
7738	Mod	V Good	18	10			74	73		1140	1140 - 140TH ST	3	10,602	30,534	77	V Good	2	3	1	4	0	0	Fall Thru Yr 4						0	0	65
7737	Mod	Good	27	14			65	64		1140	1140 - 140TH ST	3	10,602	30,534	77	V Good	2	3	1	4	0	0	Fall Thru Yr 4						0	0	54
7724	Mod	Good	31	11			64	63		1160	1160 - 143RD ST + Others	2	4,704	13,546	56	Fair	2	3	1	2	0	0	Fall Thru Yr 2						0	0	53
4979	Mod	V Good	0	18			82	81		1080	1080 - 143RD ST + Others	5	2,794	8,549	74	V Good	2	3	1	2	0	2	Selected Yr 2	20	Surface Trtmnt / Chip Seal	5.40	0	5,854	5,854	46,165	82
7723	Mod	V Good	10	12			79	78		1080	1080 - 143RD ST + Others	5	2,794	8,549	74	V Good	2	3	1	2	0	2	Selected Yr 2	20	Surface Trtmnt / Chip Seal	5.40	0	11,966	11,966	46,165	82

TABS | Definitions | Summary | Comps | $ Plot | Equity Removal | Selections | Network Analysis | Parameters | Rehab Activities | Inventory | Rehab By Segment | Rehab By Year | Need Year Rehab | Analysis Results | Annual PCI ...

Terracon

Excel Files

Even stranger sheets

Difficult

Pandas can still handle it.

```
Run Cell | Run Above | Debug cell | Go to [16]
#%%
complicated_xlsx = r'C:\temp\ComplicatedSpreadsheet.xlsm'
complicated = pd.read_excel(
    complicated_xlsx,
    sheet_name='Network Analysis',
    names=['StrengthRating', 'ConditionRating'],
    usecols='AK:AL',
    skiprows=16
)
print(complicated.head())
```

Python Interactive

```
[16] complicated_xlsx = r'C:\temp\ComplicatedSpreadsheet.xlsm'...

  StrengthRating ConditionRating
0            Mod            Poor
1            Mod            Poor
2            Mod            Poor
3           Weak            Fair
4            Mod       Excellent
```

Terracon

Excel Files

Objects in sheets

Most Difficult

TOP

Terracon

CORE LOG

CORE NUMBER	B-1025
DATE CORED	10/3/2018
LANE DIRECTION	Southbound
LOCATION	South Mehan Road

CORE DATA

Surface Material Type: ☐ A.C. ☐ P.C.C. ☐ Continuously Reinforced Concrete

Stripping or Separation in Asphalt: ☐ Stripping ☐ Separation ☑ N/A

Honeycomb or "D" Cracking in PCC: ☐ Honeycomb ☐ "D" Cracking ☑ N/A

Stabilized Subgrade Beneath Pavement or Sub-base? ☐ Yes ☑ No ☐ Unknown

CORE LAYER DATA (FROM TOP TO BOTTOM):

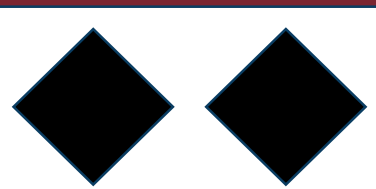
Sample No	Layer Type	Layer Thickness (in.)
1	Asphaltic Concrete	4 1/2
2	Clayey Sand (SC), Brown	3
	Refusal on Sandstone	
	Total Core Thickness	**4 1/2**
	Total Thickness / Depth	**7 1/2**

*** Asphalt type based on visual observation only**

Terracon

Excel Files

The terrain varies.



Most Difficult

Those checkboxes…

CORE DATA

Surface Material Type: ☐ A.C. ☐ P.C.C. ☐ Continuously Reinforced Concrete

Stripping or Separation in Asphalt: ☐ Stripping ☐ Separation ☑ N/A

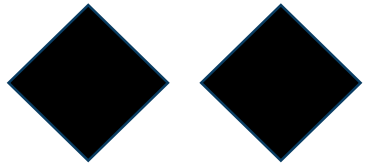
Honeycomb or "D" Cracking in PCC: ☐ Honeycomb ☐ "D" Cracking ☑ N/A

Stabilized Subgrade Beneath Pavement or Sub-base? ☐ Yes ☑ No ☐ Unknown

Terracon

Excel Files

The terrain varies.



Most Difficult

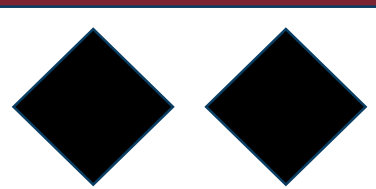
…are not cells.

CORE DATA	☐ P.C.C.			
Surface Material Type:	☐ A.C.	☐ Continuously Reinforced Concrete		
Stripping or Separation in Asphalt:	☐ Stripping	☐ Separation	☑ N/A	
Honeycomb or "D" Cracking in PCC:	☐ Honeycomb	☐ "D" Cracking	☑ N/A	
Stabilized Subgrade Beneath Pavement or Sub-base?	☐ Yes	☑ No	☐ Unknown	

Terracon

Excel Files

The terrain varies.



Most Difficult

If you really need to mess with an Excel file, use pywin32 to access the COM system.

```
from win32com.client import gencache
```

```
def extract_checkboxes(worksheet, attribute_dict):
    for shape in worksheet.Shapes:

        # See if the shape is a form control.
        if shape.Type != 8:
            continue

        # See if it's named as a checkbox.
        if not 'Check Box' in shape.Name:
            continue

        checkbox_name = shape.AlternativeText.lstrip()
        if shape.ControlFormat.Value == 1.0:
            attribute_dict[checkbox_name] = 1
        else:
            attribute_dict[checkbox_name] = 0
```

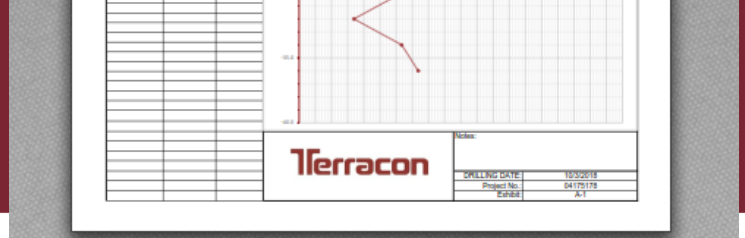
Semantic Note

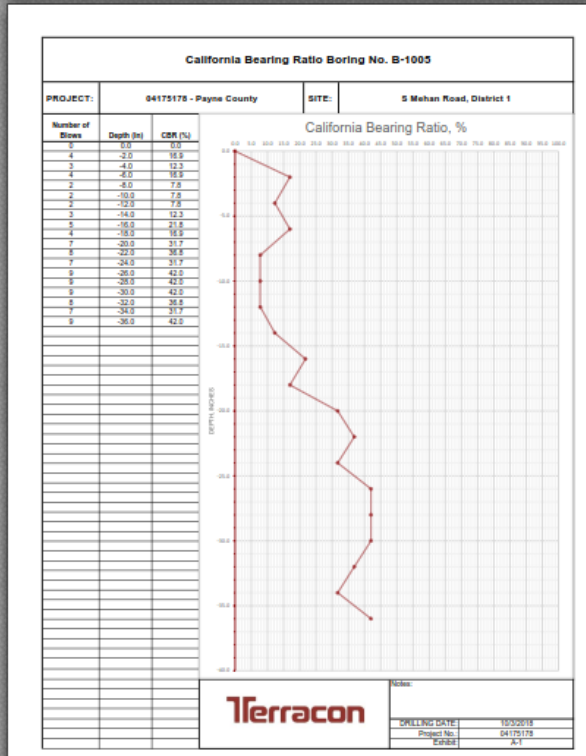
This is what I mean by “boring data.”

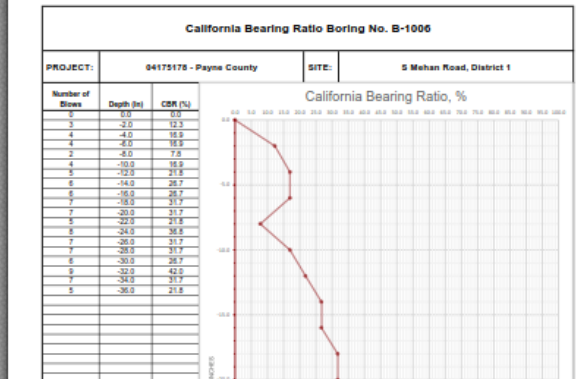
PDF Files

Reading & slicing

Each page is for a different location.







Terracon

PDF Files

Reading & slicing

Use two Python libraries.

- PyPDF2 manages pages.
- PDFMiner reads text.

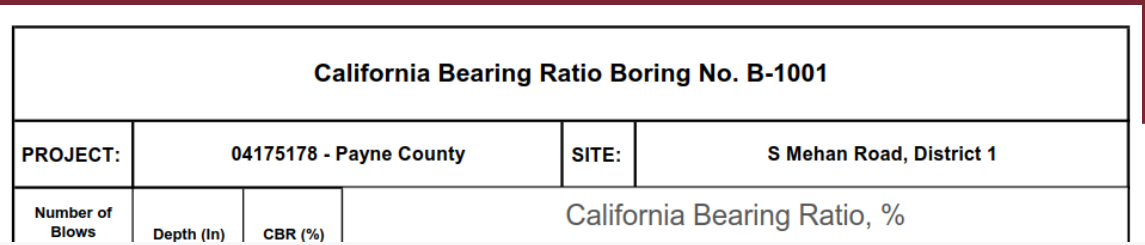
PDF Files

Reading & slicing

Use PyPDF2's PdfFileReader to read the file and get each page.

Use PDFMiner's PDFPageInterpreter to extract text from the page.

Search for text like "Boring No. B-*"



California Bearing Ratio Boring No. B-1001				
PROJECT:	04175178 - Payne County		SITE:	S Mehan Road, District 1
Number of Blows	Depth (In)	CBR (%)	California Bearing Ratio, %	

```
# Use the PDFMiner library to set up a PDFPageInterpreter.
resourcemanager = PDFResourceManager()
output = StringIO()
converter = TextConverter(resourcemanager, output, laparams=LAParams())
interpreter = PDFPageInterpreter(resourcemanager, converter)

# Get the text from the PDF.
infile = open(inputpdf, 'rb')
pageset = set([pagenumber])
generator = PDFPage.get_pages(infile, pageset)
page = next(generator)
interpreter.process_page(page)
converter.close()
infile.close()
text = output.getvalue()
output.close
```

Terracon

PDF Files

Reading & slicing

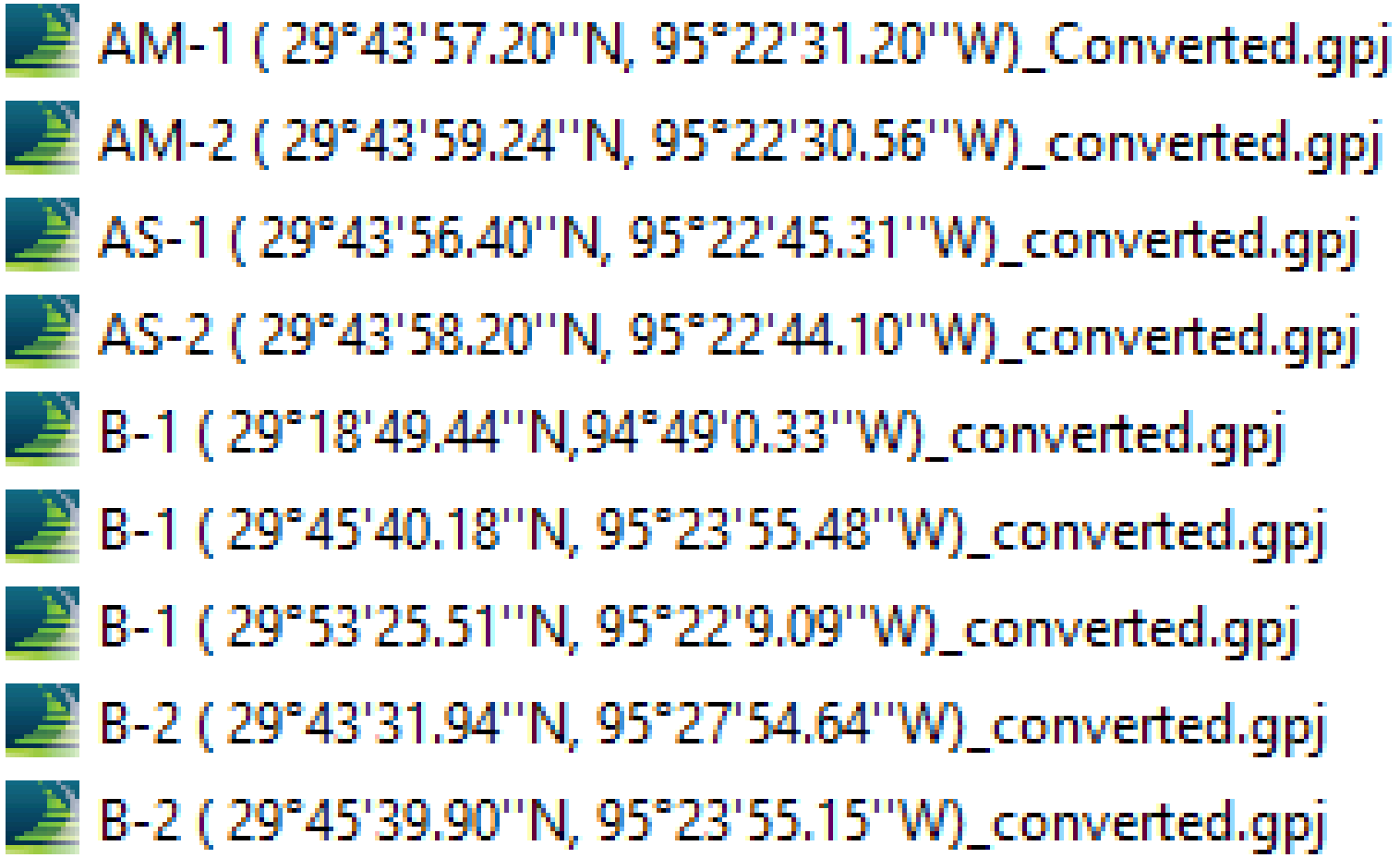
Use PyPDF2's PdfFileWriter to write each page to a new file with the ID in the name.

```
def export_page(inputpdf, pageindex, outfilename, outputfolder=None):
    if outputfolder is None:
        outputfolder = os.path.dirname(inputpdf)

    reader = PdfFileReader(open(inputpdf, "rb"))
    writer = PdfFileWriter()
    page = reader.getPage(pageindex)
    writer.addPage(page)
    outfilepath = os.path.join(outputfolder, outfilename)
    print('Writing 1 page to %s.' % (outfilepath))
    outputstream = open(outfilepath, 'wb')
    writer.write(outputstream)
    outputstream.close()
    return True
```

Lat/Long File Names

Seriously?

Yes, this really happened.



Terracon

Lat/Long File Names

Seriously?

```
import re
```

AM-1 (29°43'57.20"N, 95°22'31.20"W)_Converted.gpj

Use a regular expression to split the name into parts.

```
def parse_file_name(self, file_name):
    """Split the name of a file into boring ID, latitude, & longitude.

    :param file_name: The name of a file

    :return: A list containing boring ID, latitude, longitude, & the file name
    """

    raw_id, raw_lat, raw_lon, _ = re.split('[(),]+', file_name)
    boring_id = raw_id.strip()
    lat = self.convert_dms(raw_lat)
    lon = self.convert_dms(raw_lon)
    return [boring_id, lat, lon, file_name]
```

Terracon

Lat/Long File Names

Seriously?

Split each part at the degree symbol & quotes.

If it's west or south, make it negative.

```
def convert_dms(self, dms):
    """Convert degrees-minutes-seconds to decimal degrees.

    :param dms: A latitude or longitude in degrees-minutes-seconds (e.g. 29°51'53.14''N)

    :return: A decimal number
    """

    raw_deg, raw_min, raw_sec, _, raw_dir = re.split("[\xb0']", dms)
    dec_deg = float(raw_deg.strip())
    dec_deg += float(raw_min) / 60
    dec_deg += float(raw_sec) / 3600
    if raw_dir in 'WS':
        dec_deg = -dec_deg
    return dec_deg
```

Terracon

Lat/Long File Names

Seriously?

```python
def write_csv(self, folder):
    """Write a CSV file containing boring IDs & coordinates.

    :param folder: The path to a folder (directory).

    :return: The path to a new CSV file
    """

    borings = self.read(folder)
    output_file = os.path.join(folder, 'Borings.csv')
    print('Writing {}.'.format(output_file))
    with open(output_file, 'w') as f:
        f.write('BoringID,Latitude,Longitude,FileName\n')
        for boring in borings:
            boring_id, lat, lon, file_name = boring
            f.write('{},{},{},"{}"\n'.format(boring_id, lat, lon, file_name))
    return output_file
```

BoringID	Latitude	Longitude	FileName
AM-1	29.73255556	-95.37533333	AM-1 (29°43'57.20"N, 95°22'31.20"W)_Converted.gpj
AM-2	29.73312222	-95.37515556	AM-2 (29°43'59.24"N, 95°22'30.56"W)_converted.gpj
AS-1	29.73233333	-95.37925278	AS-1 (29°43'56.40"N, 95°22'45.31"W)_converted.gpj
AS-2	29.73283333	-95.37891667	AS-2 (29°43'58.20"N, 95°22'44.10"W)_converted.gpj
B-1	29.31373333	-94.81675833	B-1 (29°18'49.44"N,94°49'0.33"W)_converted.gpj
B-1	29.76116111	-95.39874444	B-1 (29°45'40.18"N, 95°23'55.48"W)_converted.gpj
B-1	29.89041944	-95.36919167	B-1 (29°53'25.51"N, 95°22'9.09"W)_converted.gpj
B-2	29.72553889	-95.46517778	B-2 (29°43'31.94"N, 95°27'54.64"W)_converted.gpj
B-2	29.76108333	-95.39865278	B-2 (29°45'39.90"N, 95°23'55.15"W)_converted.gpj

Terracon

Secret MS Access Databases

e.g. gINT GPJ files

Many programs (especially older ones) have proprietary databases.

Typically, these are just Microsoft Access databases with different file extensions.

ArcMap can usually read these, but not always.

Secret MS Access Databases

e.g. GPJ files

Import pyodbc.

```
conn_str = (
    r'DRIVER={Microsoft Access Driver (*.mdb, *.accdb)};'
    r'DBQ='
) + gpj_path
cnxn = pyodbc.connect(conn_str)
cnxn.setencoding('utf-8')
cursor = cnxn.cursor()
cursor.execute('select PointID, [Latitude Decimal], [Longitude Decimal] from POINT')
cursor.fetchone()
cursor.execute('update POINT set [Latitude Decimal] = {}, [Longitude Decimal] = {}'.format(latitude, longitude))
cnxn.commit()
print('Set {} to {}, {}.'.format(boring_id, longitude, latitude))
```

Terracon

Low-Distortion Coordinate Systems

Smaller than state plane

GNSS surveying makes distortions in coordinate systems more obvious.

There’s a movement to divide state plane zones into smaller zones with less distortion.

For us, it means there are more coordinate systems to deal with.

Low-Distortion Coordinate Systems

Virginia has already had two of them.

Sec. 10.11 **The VDOT Project Coordinate System**

Beginning June 1, 2014 all new VDOT Projects will be based on the new VDOT Project Coordinate System outlined below (Now known as "VDOT Project Coordinates-2014").
To convert Virginia State Plane Coordinates (based on the US Survey Foot) to VDOT Project Coordinates-2014, the coordinates will need to be multiplied by the combined Scale & Elevation Factor for each specific project. One method of obtaining the scale factor for each project will be to submit GPS data to OPUS (NGS utility) for each primary control point on the project. Submitting "Static" data to OPUS (minimum 2- hour occupations per point) will be required. Once the OPUS results are obtained, take the average of the combined factors under the State Plane Coordinates for the primary control points. Once this step is done, the inverse function (1/x) should be applied, resulting in the Combined Scale Factor for the project (9 decimal places- Example= 1.000000009).

This is only one method of obtaining the scale factor for a project. Regardless of the method used, the procedure shall be described in detail in the project notes as well in the Project Deliverables (Sec. 10.06).◊

Special Note on Projects that predate June 1, 2014:◊

Projects completed or started prior to January 1, 2014 should continue to use the former language below.

The VDOT Coordinate System is based on **NAD83 METRIC values** as defined in **The Code of Virginia §55-292** (see Figure 10-H). To convert NAD83 METRIC to VDOT Project coordinates (Imperial Units), first depending on the zone you are working in, subtract 1,000,000 meters from the South Zone Northing value (or 2,000,000 meters from the North Zone Northing value). Next, subtract 2,500,000 meters from the Easting value. Next, multiply the Northing and Easting values by 3.28083333333 (the conversion for the U. S. Survey Foot as defined in **The Code of Virginia §55-290,** see Figure 10-M). Last, multiply the Northing and Easting values by the Combined County Scale & Elevation Factor. Figure 10-N is a list of the combined scale and elevation factor for the counties. This produces VDOT Project Coordinates (in Imperial Units) for a given project. A reverse of this procedure will transform VDOT Project Coordinates back the original NAD83 METRIC values. See Figure 10-F, showing the use of the above procedures as depicted on a LD-200 Horizontal Control Station Reference Card.

Terracon

Low-Distortion Coordinate Systems

Do this conversion backward.

To convert NAD83 METRIC to VDOT Project coordinates (Imperial Units), first depending on the zone you are working in, subtract 1,000,000 meters from the South Zone Northing value (or 2,000,000 meters from the North Zone Northing value). Next, subtract 2,500,000 meters from the Easting value. Next, multiply the Northing and Easting values by 3.28083333333 (the conversion for the U. S. Survey Foot as defined in The Code of Virginia §55-290, see Figure 10-M). Last, multiply the Northing and Easting values by the Combined County Scale & Elevation Factor. Figure 10-N is a list of the combined scale and elevation factor for the counties. This produces VDOT Project Coordinates (in Imperial Units) for a given project. A reverse of this procedure will transform VDOT Project Coordinates back the original NAD83 METRIC values. See Figure 10-F, showing the use of the above procedures as depicted on a LD-200 Horizontal Control Station Reference Card.

Low-Distortion Coordinate Systems

Make a configuration file (in YAML format) to look up county parameters.

```yaml
counties:
  Arlington:
    zone: north
    scale: 1.00006
  Accomack:
    zone: south
    scale: 1.00004
  Albemarle:
    zone: south
    scale: 1.00002
  Alleghany:
    zone: south
    scale: 1.00015
  Amelia:
    zone: south
    scale: 1.00007
  Amherst:
    zone: south
    scale: 1.00009
  Appomattox:
    zone: south
    scale: 1.00008
```

Low-Distortion Coordinate Systems

Read a CSV file, then use Pandas to do the calculations in a dataframe.

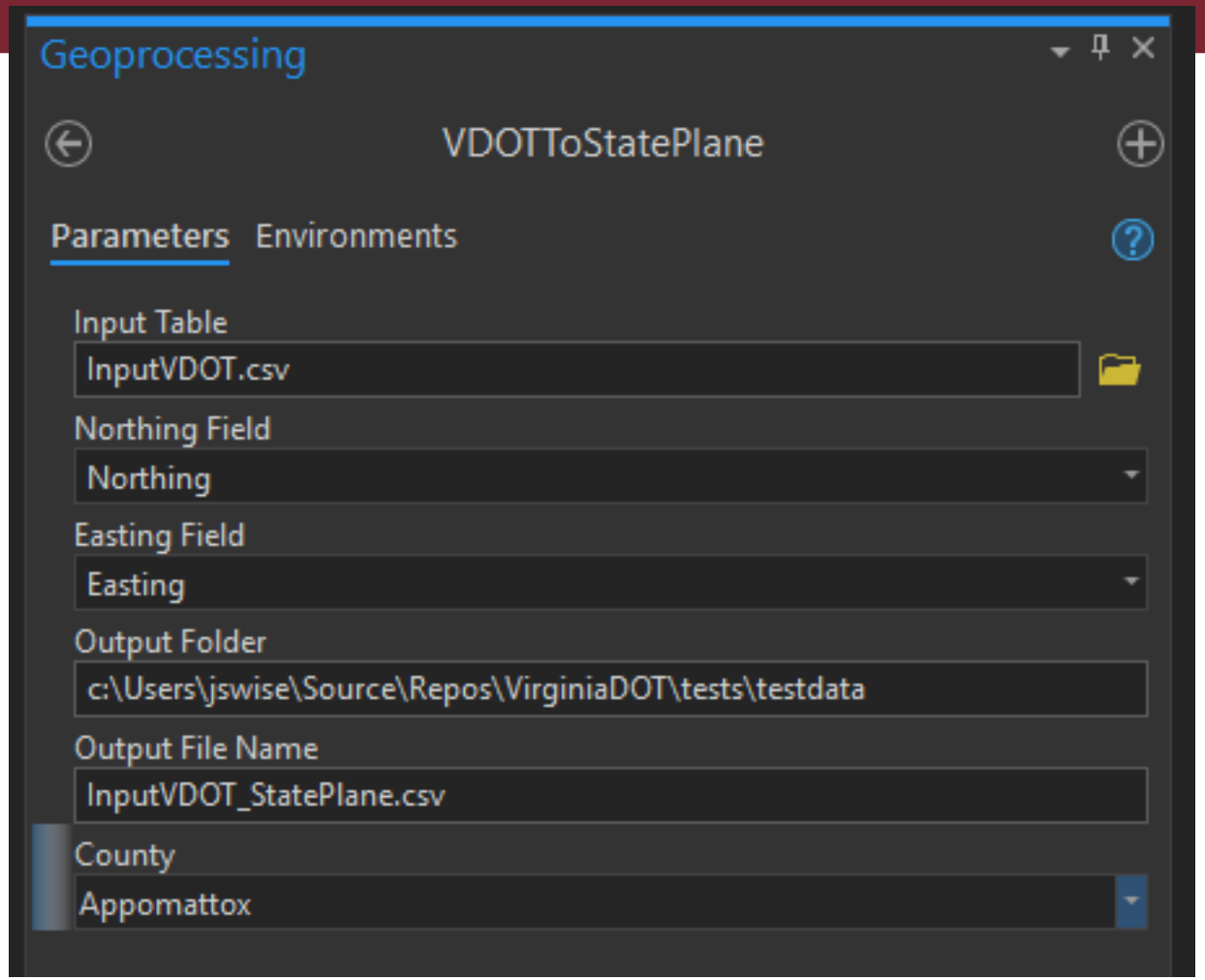
Convert the dataframe to an XY layer.

```
y_scaled_feet = df[y_field] / county_params['scale']
df['NorthingSPFeet'] = y_scaled_feet + county_params['false_northing_feet']
df.drop(y_field, axis=1, inplace=True)

x_scaled_feet = df[x_field] / county_params['scale']
df['EastingSPFeet'] = x_scaled_feet + county_params['false_easting_feet']
df.drop(x_field, axis=1, inplace=True)

df['NorthingSPMeters'] = df['NorthingSPFeet'] / 3.28083333333
df['EastingSPMeters'] = df['EastingSPFeet'] / 3.28083333333

self.info('Writing state plane (meters) coordinates to {}.'.format(output_file))
df.to_csv(output_file, index=False)

# Make an event layer for viewing.
if county_params['zone'] == 'north':
    wkid = '6592'
else:
    wkid = '6594'
return arcpy.management.MakeXYEventLayer(
    output_file,
    'EastingSPMeters',
    'NorthingSPMeters',
    os.path.basename(output_file),
    wkid
)
```

Terracon

Low-Distortion Coordinate Systems

Make it into a nice geoprocessing tool.



Low-Distortion Coordinate Systems

The user immediately tried to run it on this.

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T
1	Boring Nu	Northing	Easting	Station	Main Station		TRUE STAT	Adjusted	Station Sn	Offset	LT or RT							Station		
2	18BR-01	########	########	1365+19.9	1365	1365	136520	136520	19.98	1.7834	RT							1365+19.98	1.78 ft. Right	
3	18BR-02	########	########	1366+25.5	1366	1366	136625.6	136625.6	25.59	-4.1641	LT							1366+25.59	4.16 ft. Left	
4	18BR-03	########	########	1366+27.6	1366	1366	136627.6	136627.6	27.64	-48.6163	LT							1366+27.64	48.62 ft. Left	
5	18BR-04	########	########	1367+27.4	1367	1367	136727.5	136727.5	27.48	-2.9966	LT							1367+27.48	3.00 ft. Left	
6	18BR-05	########	########	1367+21.7	1367	1367	136721.8	136721.8	21.77	-50.4625	LT							1367+21.77	50.46 ft. Left	
7	18BR-06	########	########	1368+02.6	1368	1368	136802.6	136802.6	2.64	-10.3892	LT							1368+02.64	10.39 ft. Left	
8	18BR-07	########	########	1368+05.8	1368	1368	136805.8	136805.8	5.84	-41.8722	LT							1368+05.84	41.87 ft. Left	
9	18CPT-01	########	########	1207+01.2	1207	1207	120701.2	120701.2	1.24	-132.522	LT							1207+01.24	132.52 ft. Left	
10	18CPT-02	########	########	1329+75.0	1329	1329	132975	132975	75.02	130.1751	RT							1329+75.02	130.18 ft. Right	
11	18CPT-03	########	########	1332+05.4	1332	1332	133205.4	133205.4	5.4	-120.901	LT							1332+05.40	120.90 ft. Left	
12	18CPT-04	########	########	1341+31.7	1341	1341	134131.8	134131.8	31.75	126.0207	RT							1341+31.75	126.02 ft. Right	
13	18CPT-05	########	########	1346+05.0	1346	1346	134605.1	134605.1	5.06	-36.7828	LT							1346+05.06	36.78 ft. Left	
14	18CPT-06	########	########	1394+65.1	1394	1394	139465.2	139465.2	65.16	-85.9203	LT							1394+65.16	85.92 ft. Left	
15	18CPT-07	########	########	1451+31.5	1451	1451	145131.5	145131.5	31.5	-101.066	LT							1451+31.50	101.07 ft. Left	
16	18DMT-01	########	########	1201+63.9	1201	1201	120164	120164	63.98	-128.801	LT							1201+63.98	128.80 ft. Left	
17	18DMT-02	########	########	1345+95.0	1345	1345	134595.1	134595.1	95.05	65.0505	RT							1345+95.05	65.05 ft. Right	
18	18DMT-03	########	########	1356+81.2	1356	1356	135681.3	135681.3	81.29	61.9962	RT							1356+81.29	62.00 ft. Right	
19	18DMT-04	########	########	1359+69.6	1359	1359	135969.7	135969.7	69.65	38.6837	RT							1359+69.65	38.68 ft. Right	
20	18DMT-05	########	########	1360+10.9	1360	1360	136010.9	136010.9	10.91	-28.0871	LT							1360+10.91	28.09 ft. Left	
21	18DMT-06	########	########	1363+81.5	1363	1363	136381.6	136381.6	81.58	48.8654	RT							1363+81.58	48.87 ft. Right	
22	18DMT-07	########	########	1364+81.2	1364	1364	136481.3	136481.3	81.26	38.7565	RT							1364+81.26	38.76 ft. Right	
23	18DMT-08	########	########	1363+88.8	1363	1363	136388.9	136388.9	88.87	-17.8367	LT							1363+88.87	17.82 ft. Left	
24	18DMT-09	########	########	1363+97.0	1363	1363	136397.1	136397.1	97.05	-83.7162	LT							1363+97.05	83.72 ft. Left	
25	18DMT-10	########	########	1368+60.5	1368	1368	136860.6	136860.6	60.59	-10.6026	LT							1368+60.59	10.60 ft. Left	
26	18DMT-11	########	########	1369+08.7	1369	1369	136908.7	136908.7	8.74	-113.166	LT							1369+08.74	113.17 ft. Left	
27	18DMT-12	########	########	1370+69.8	1370	1370	137069.8	137069.8	69.8	66.7028	RT							1370+69.80	66.70 ft. Right	
28	18DMT-13	########	########	1372+43.1	1372	1372	137243.2	137243.2	43.17	-15.6473	LT							1372+43.17	15.65 ft. Left	
29	18DMT-14	########	########	1373+53.2	1373	1373	137353.2	137353.2	53.23	44.9873	RT							1373+53.23	44.99 ft. Right	
30	18DMT-15	########	########	1396+10.0	1396	1396	139610	139610	10	-119.468	LT							1396+10.00	119.47 ft. Left	
31	18DP-001	DO NOT STAKE																		
32	18DP-002																			
33	18DP-003																			
34	18DP-004																			
35	18DP-005																			

Terracon

UAS Data

Large quantities of geotagged photos

The project: Photogrammetric modeling of really large quarry walls.

Terracon

UAS Data

Large quantities of geotagged photos

We needed to manage thousands of photos and group them by location.

The first step was to make these points.

UAS Data

Large quantities of geotagged photos

Install Pillow, which is based on the Python Image Library (PIL).

```
from PIL import Image
from PIL.ExifTags import GPSTAGS
from PIL.ExifTags import TAGS
```

Get an image's EXIF header.

```
image = Image.open(path)
self.exif = image._getexif()
```

Find the GPSInfo tag in the EXIF header.

```
tag_dict = {}
for numeric_key, val in self.exif.items():
    text_key = TAGS.get(numeric_key)
    if text_key:
        tag_dict[text_key] = val
gps_info = tag_dict.get('GPSInfo')
if not gps_info:
    return
```

Terracon

UAS Data

Large quantities of geotagged photos

Convert coordinates.

```
# Convert the latitude to decimal degrees.
yref = self.gps_info[1]
ydms = self.gps_info[2]
ydd = self.dms_to_dd(ydms)
if yref == 'S':
    ydd = -ydd

# Convert the longitude to decimal degrees.
xref = self.gps_info[3]
xdms = self.gps_info[4]
xdd = self.dms_to_dd(xdms)
if xref == 'W':
    xdd = -xdd

# Convert the altitude to meters.
zm = self.tuple_to_float(self.gps_info[6])

self.coord_list = [xdd, ydd, zm]
```

Scanned Documents

The messiest data

You can use Python-tesseract (pytesseract) for optical character recognition (OCR).

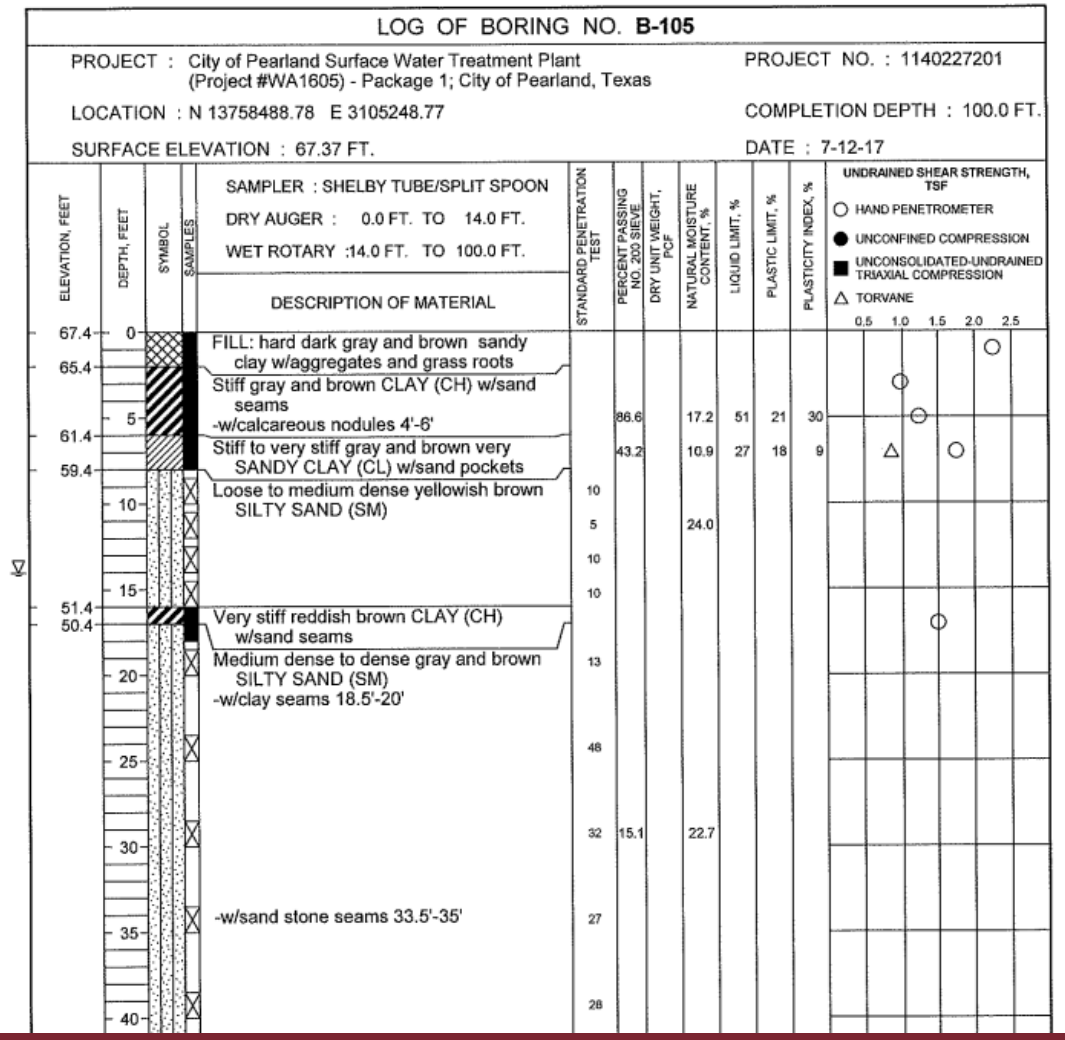
For typical text documents, this is straightforward.

Scanned Documents

The messiest data

◆◆ Most Difficult

For specially-formatted documents, it's not.



LOG OF BORING NO. **B-105**

PROJECT : City of Pearland Surface Water Treatment Plant (Project #WA1605) - Package 1; City of Pearland, Texas

PROJECT NO. : 1140227201

LOCATION : N 13758488.78 E 3105248.77

COMPLETION DEPTH : 100.0 FT.

SURFACE ELEVATION : 67.37 FT.

DATE : 7-12-17

SAMPLER : SHELBY TUBE/SPLIT SPOON

DRY AUGER : 0.0 FT. TO 14.0 FT.

WET ROTARY :14.0 FT. TO 100.0 FT.

ELEVATION, FEET	DEPTH, FEET	DESCRIPTION OF MATERIAL	STANDARD PENETRATION TEST	PERCENT PASSING NO. 200 SIEVE	DRY UNIT WEIGHT, PCF	NATURAL MOISTURE CONTENT, %	LIQUID LIMIT, %	PLASTIC LIMIT, %	PLASTICITY INDEX, %
67.4	0	FILL: hard dark gray and brown sandy clay w/aggregates and grass roots							
65.4		Stiff gray and brown CLAY (CH) w/sand seams		86.6		17.2	51	21	30
	5	-w/calcareous nodules 4'-6'							
61.4		Stiff to very stiff gray and brown very SANDY CLAY (CL) w/sand pockets		43.2		10.9	27	18	9
59.4		Loose to medium dense yellowish brown SILTY SAND (SM)	10						
	10		5			24.0			
			10						
	15		10						
51.4		Very stiff reddish brown CLAY (CH) w/sand seams							
50.4		Medium dense to dense gray and brown SILTY SAND (SM) -w/clay seams 18.5'-20'	13						
	20								
			48						
	25								
			32	15.1		22.7			
	30								
		-w/sand stone seams 33.5'-35'	27						
	35								
			28						
	40								

UNDRAINED SHEAR STRENGTH, TSF

○ HAND PENETROMETER

● UNCONFINED COMPRESSION

■ UNCONSOLIDATED-UNDRAINED TRIAXIAL COMPRESSION

△ TORVANE

0.5 1.0 1.5 2.0 2.5

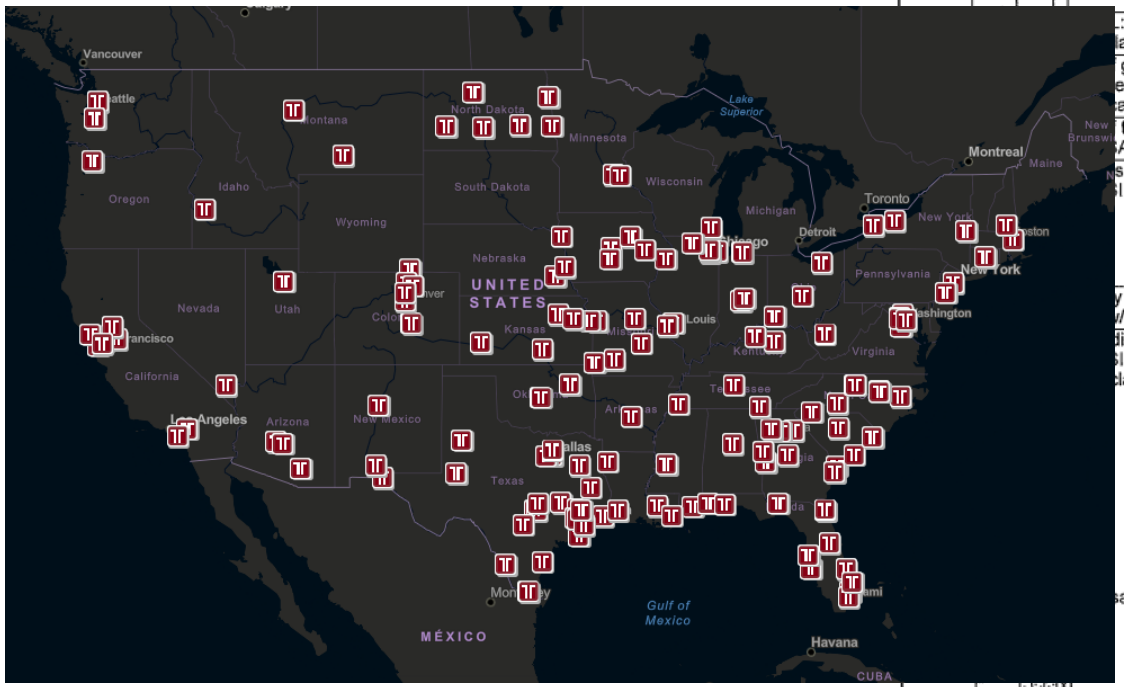
Terracon

Scanned Documents

The messiest data

◆◆ Most Difficult

We have a lot of offices with decades' worth of logs in varying formats.



LOG OF BORING NO. **B-105**

PROJECT : City of Pearland Surface Water Treatment Plant (Project #WA1605) - Package 1; City of Pearland, Texas — PROJECT NO. : 1140227201

LOCATION : N 13758488.78 E 3105248.77 — COMPLETION DEPTH : 100.0 FT.

SURFACE ELEVATION : 67.37 FT. — DATE : 7-12-17

SAMPLER : SHELBY TUBE/SPLIT SPOON
DRY AUGER : 0.0 FT. TO 14.0 FT.
WET ROTARY :14.0 FT. TO 100.0 FT.

ELEVATION, FEET	DEPTH, FEET	SYMBOL	SAMPLES	DESCRIPTION OF MATERIAL	STANDARD PENETRATION TEST	PERCENT PASSING NO. 200 SIEVE	DRY UNIT WEIGHT, PCF	NATURAL MOISTURE CONTENT, %	LIQUID LIMIT, %	PLASTIC LIMIT, %	PLASTICITY INDEX, %
				: hard dark gray and brown sandy lay w/aggregates and grass roots							
				gray and brown CLAY (CH) w/sand eams		86.6		17.2	51	21	30
				alcareous nodules 4'-6'		43.2		10.9	27	18	9
				to very stiff gray and brown very ANDY CLAY (CL) w/sand pockets							
				se to medium dense yellowish brown ILTY SAND (SM)	10						
					5			24.0			
					10						
					10						
				y stiff reddish brown CLAY (CH) w/sand seams							
				dium dense to dense gray and brown ILTY SAND (SM) lay seams 18.5'-20'	13						
					48						
					32	15.1		22.7			
				sand stone seams 33.5'-35'	27						
					28						
	40										

UNDRAINED SHEAR STRENGTH, TSF
○ HAND PENETROMETER
● UNCONFINED COMPRESSION
■ UNCONSOLIDATED-UNDRAINED TRIAXIAL COMPRESSION
△ TORVANE
0.5 1.0 1.5 2.0 2.5

Terracon

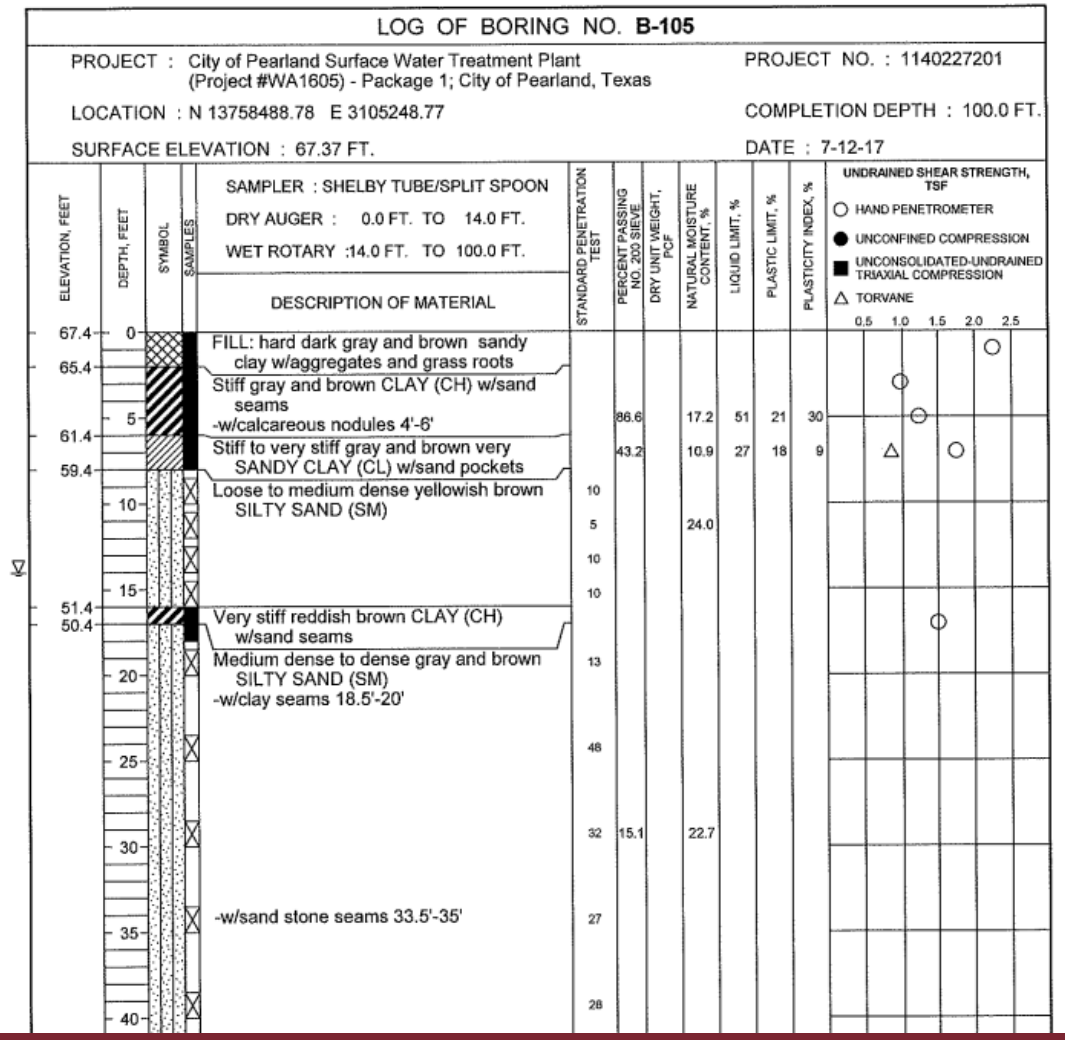
Scanned Documents

The messiest data

◆◆ Most Difficult

My ambition: Extract subsurface data from these.

I'm not sure if it's possible.



LOG OF BORING NO. **B-105**

PROJECT : City of Pearland Surface Water Treatment Plant (Project #WA1605) - Package 1; City of Pearland, Texas

PROJECT NO. : 1140227201

LOCATION : N 13758488.78 E 3105248.77

COMPLETION DEPTH : 100.0 FT.

SURFACE ELEVATION : 67.37 FT.

DATE : 7-12-17

SAMPLER : SHELBY TUBE/SPLIT SPOON

DRY AUGER : 0.0 FT. TO 14.0 FT.

WET ROTARY :14.0 FT. TO 100.0 FT.

ELEVATION, FEET	DEPTH, FEET	DESCRIPTION OF MATERIAL	STANDARD PENETRATION TEST	PERCENT PASSING NO. 200 SIEVE	DRY UNIT WEIGHT, PCF	NATURAL MOISTURE CONTENT, %	LIQUID LIMIT, %	PLASTIC LIMIT, %	PLASTICITY INDEX, %
67.4	0	FILL: hard dark gray and brown sandy clay w/aggregates and grass roots							
65.4		Stiff gray and brown CLAY (CH) w/sand seams		86.6		17.2	51	21	30
	5	-w/calcareous nodules 4'-6'							
61.4		Stiff to very stiff gray and brown very SANDY CLAY (CL) w/sand pockets		43.2		10.9	27	18	9
59.4		Loose to medium dense yellowish brown SILTY SAND (SM)	10						
	10		5			24.0			
			10						
	15		10						
51.4		Very stiff reddish brown CLAY (CH) w/sand seams							
50.4		Medium dense to dense gray and brown SILTY SAND (SM) -w/clay seams 18.5'-20'	13						
	20								
			48						
	25								
			32	15.1		22.7			
	30								
		-w/sand stone seams 33.5'-35'	27						
	35								
			28						
	40								

UNDRAINED SHEAR STRENGTH, TSF

○ HAND PENETROMETER

● UNCONFINED COMPRESSION

■ UNCONSOLIDATED-UNDRAINED TRIAXIAL COMPRESSION

△ TORVANE

0.5 1.0 1.5 2.0 2.5

Terracon

Jason Wise
jason.wise@terracon.com
github.com/jswise/keenepyt

RESPONSIVE | RESOURCEFUL | RELIABLE

terracon.com