Extracting Data from Excel using Python

Speaker : Samuel Oranyeli

ABOUT ME

- Love open-source projects.
- Developer on pyjanitor, contributor to datatable and clumper.
- Love building data products.
- Blogger : https://samukweku.github.io/data-wrangling-blog/.
- Tennis Lover.

Contents

- Basics.
- Pivot table with multiple headers.
- Excel tables.
- Conditional formatting.
- Sheet with comments.
- Small multiples.

Third-Party Packages:

- Pandas
- Numpy
- OpenPyxl
- Pyjanitor

Import libraries

```
import pandas as pd
import numpy as np
import janitor
from openpyxl import load_workbook
from collections import defaultdict
```

BASICS

Read from a single sheet … a dataframe is returned :

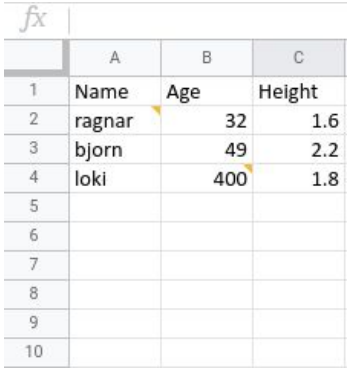
	A	B	C
1	Name	Age	Height
2	Tolu	24	2
3	Chukwuka	50	1.8
4	Ogor	15	1.5
5			
6			
7			
8			
9			

```
pd.read_excel(io = "names.xlsx", sheet_name = "naija")

        Name    Age Height
0       Tolu    24  2.0
1   Chukwuka    50  1.8
2       Ogor    15  1.5
```

Read from multiple sheets … a dictionary of dataframes is returned, with the sheet name as the key :

	A	B	C
1	Name	Age	Height
2	Tolu	24	2
3	Chukwuka	50	1.8
4	Ogor	15	1.5
5			
6			
7			
8			
9			



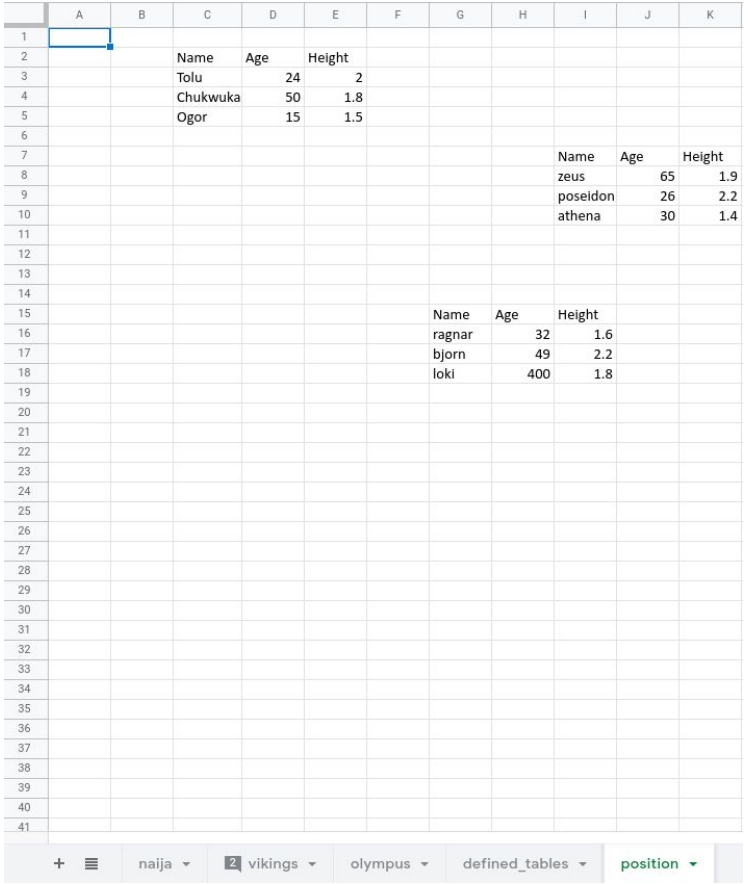
	A	B	C
1	Name	Age	Height
2	ragnar	32	1.6
3	bjorn	49	2.2
4	loki	400	1.8
5			
6			
7			
8			
9			
10			

```
pd.read_excel(io = "names.xlsx", sheet_name = ["naija", "vikings"])

{'naija':        Name  Age  Height
          0      Tolu   24     2.0
          1  Chukwuka   50     1.8
          2      Ogor   15     1.5,
 'vikings':       Name  Age  Height
          0    ragnar   32     1.6
          1     bjorn   49     2.2
          2      loki  400     1.8}

# read in all sheets :
pd.read_excel(io = "names.xlsx", sheet_name= None)
```

Read data that does not start from *A1* or the first row :



	Name	Age	Height
	Tolu	24	2
	Chukwuka	50	1.8
	Ogor	15	1.5

Name	Age	Height
zeus	65	1.9
poseidon	26	2.2
athena	30	1.4

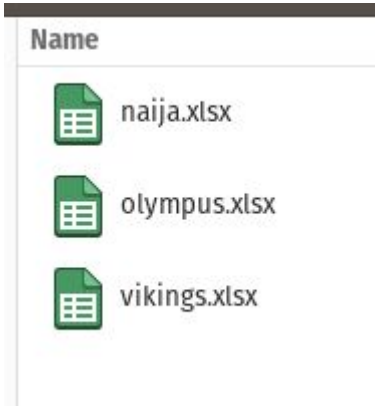
Name	Age	Height
ragnar	32	1.6
bjorn	49	2.2
loki	400	1.8

naija | vikings | olympus | defined_tables | position

```
pd.read_excel("names.xlsx",
              sheet_name = "position",
              usecols = "C:E",
              header = 1,
              nrows = 3)

        Name    Age Height
0       Tolu    24  2.0
1   Chukwuka    50  1.8
2       Ogor    15  1.5
```

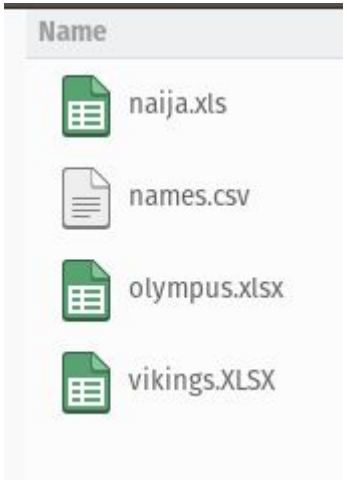
Read in all excel files in a folder :




```
# you can read it in as a dictionary
from pathlib import Path
folder = Path("excel_files")
excel_data = {filename.stem : pd.read_excel(filename)
                              for filename in folder.iterdir()}
excel_data

{'naija':        Name  Age  Height
          0      Tolu   24     2.0
          1  Chukwuka   50     1.8
          2      Ogor   15     1.5,
 'vikings':      Name  Age  Height
          0    ragnar   32     1.6
          1     bjorn   49     2.2
          2      loki  400     1.8,
 'olympus':      Name  Age  Height
          0      zeus   65     1.9
          1  poseidon   26     2.2
          2    athena   30     1.4}
```

Read in only the excel files in a folder :




```
folder = Path("mixed_files").rglob("*.[xX]*")
excel_data = {filename.stem : pd.read_excel(filename)
                              for filename in folder}
excel_data
```

```
{'naija':        Name  Age  Height
          0      Tolu   24     2.0
          1  Chukwuka   50     1.8
          2      Ogor   15     1.5,
 'vikings':      Name  Age  Height
          0    ragnar   32     1.6
          1     bjorn   49     2.2
          2      loki  400     1.8,
 'olympus':      Name  Age  Height
          0      zeus   65     1.9
          1  poseidon   26     2.2
          2    athena   30     1.4}
```

A Little more involved

Pivot Table with Multiple Headers

A	B	C	D	E	F	G	H
		Age	*Child*		*Adult*		
		Survived	*No*	*Yes*	*No*	*Yes*	
Class	**Sex**						
1st	*Male*		0	5	118	57	
	Female		0	1	4	140	
2nd	*Male*		0	11	154	14	
	Female		0	13	13	80	
3rd	*Male*		35	13	387	75	
	Female		17	14	89	76	
Crew	*Male*		0	0	670	192	
	Female		0	0	3	20	

How does it look in Pandas?

```
pd.read_excel('titanic.xlsx')
```

	Unnamed: 0	Unnamed: 1	Age	Child	Unnamed: 4	Adult	Unnamed: 6
0	NaN	NaN	Survived	No	Yes	No	Yes
1	Class	Sex	NaN	NaN	NaN	NaN	NaN
2	1st	Male	NaN	0	5	118	57
3	NaN	Female	NaN	0	1	4	140
4	2nd	Male	NaN	0	11	154	14
5	NaN	Female	NaN	0	13	13	80
6	3rd	Male	NaN	35	13	387	75
7	NaN	Female	NaN	17	14	89	76
8	Crew	Male	NaN	0	0	670	192
9	NaN	Female	NaN	0	0	3	20

Untidy; how do we fix this?

- Read in data as a MultiIndex dataframe:

```
pd.read_excel('titanic.xlsx', header = [0,1], index_col=[0,1])
```

		Age	Child		Adult	
		Survived	No	Yes	No	Yes
Class	Sex					
1st	Male	NaN	0	5	118	57
	Female	NaN	0	1	4	140
2nd	Male	NaN	0	11	154	14
	Female	NaN	0	13	13	80
3rd	Male	NaN	35	13	387	75
	Female	NaN	17	14	89	76
Crew	Male	NaN	0	0	670	192
	Female	NaN	0	0	3	20

A bit better, but still room for improvement. What’s next?

- Rename columns axis with meaningful words:

```
(pd.read_excel('titanic.xlsx', header = [0,1], index_col=[0,1])
 .rename_axis(columns=["Adult_or_Child","Survived"])
)
```

	Adult_or_Child	Age	Child		Adult	
	Survived	Survived	No	Yes	No	Yes
Class	Sex					
1st	Male	NaN	0	5	118	57
	Female	NaN	0	1	4	140
2nd	Male	NaN	0	11	154	14
	Female	NaN	0	13	13	80
3rd	Male	NaN	35	13	387	75
	Female	NaN	17	14	89	76
Crew	Male	NaN	0	0	670	192
	Female	NaN	0	0	3	20

- Get rid of the completely empty column - it serves no purpose :

```
(pd.read_excel('titanic.xlsx', header = [0,1], index_col=[0,1])
 .rename_axis(columns=["Adult_or_Child","Survived"])
 .dropna(how='all',axis=1)
)
```

Adult_or_Child		Child		Adult	
Survived		No	Yes	No	Yes
Class	Sex				
1st	Male	0	5	118	57
	Female	0	1	4	140
2nd	Male	0	11	154	14
	Female	0	13	13	80
3rd	Male	35	13	387	75
	Female	17	14	89	76
Crew	Male	0	0	670	192
	Female	0	0	3	20

- Stack the columns :

```
(pd.read_excel('titanic.xlsx', header = [0,1], index_col=[0,1])
 .rename_axis(columns=["Adult_or_Child","Survived"])
 .dropna(how='all',axis=1)
 .stack(["Adult_or_Child","Survived"])
)

Class  Sex     Adult_or_Child  Survived
1st    Male    Adult           No          118
                               Yes          57
               Child           No           0
                               Yes          5
       Female  Adult           No           4
                               Yes        140
               Child           No           0
                               Yes          1
2nd    Male    Adult           No          154
                               Yes          14
               Child           No           0
                               Yes         11
       Female  Adult           No          13
                               Yes         80
               Child           No           0
                               Yes         13
3rd    Male    Adult           No          387
                               Yes         75
               Child           No          35
                               Yes         13
       Female  Adult           No          89
                               Yes         76
               Child           No          17
                               Yes         14
Crew   Male    Adult           No          670
                               Yes        192
               Child           No           0
                               Yes          0
       Female  Adult           No           3
                               Yes         20
               Child           No           0
                               Yes          0
dtype: int64
```

Photo by Iva Rajović on Unsplash

- Last step is to reset the index :

```
(pd.read_excel('titanic.xlsx', header = [0,1], index_col=[0,1])
 .rename_axis(columns=["Adult_or_Child","Survived"])
 .dropna(how='all',axis=1)
 .stack(["Adult_or_Child","Survived"])
 .reset_index(name="Numbers")
 .head(15)
)
```

	Class	Sex	Adult_or_Child	Survived	Numbers
0	1st	Male	Adult	No	118
1	1st	Male	Adult	Yes	57
2	1st	Male	Child	No	0
3	1st	Male	Child	Yes	5
4	1st	Female	Adult	No	4
5	1st	Female	Adult	Yes	140
6	1st	Female	Child	No	0
7	1st	Female	Child	Yes	1
8	2nd	Male	Adult	No	154
9	2nd	Male	Adult	Yes	14
10	2nd	Male	Child	No	0
11	2nd	Male	Child	Yes	11
12	2nd	Female	Adult	No	13
13	2nd	Female	Adult	Yes	80
14	2nd	Female	Child	No	0

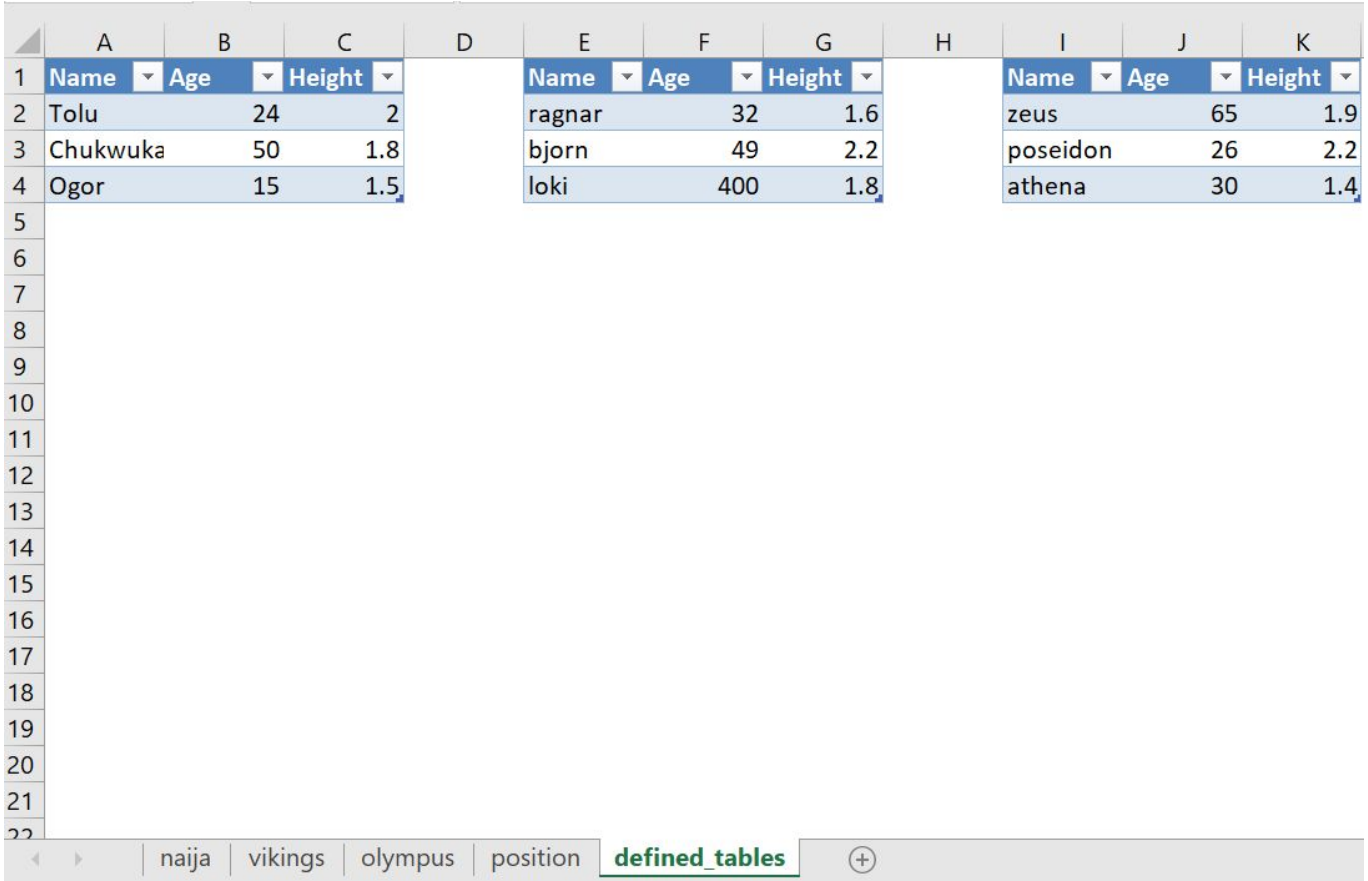
Before

		Age	*Child*		*Adult*		
		Survived	*No*	*Yes*	*No*	*Yes*	
Class	**Sex**						
1st	*Male*		0	5	118	57	
	Female		0	1	4	140	
2nd	*Male*		0	11	154	14	
	Female		0	13	13	80	
3rd	*Male*		35	13	387	75	
	Female		17	14	89	76	
Crew	*Male*		0	0	670	192	
	Female		0	0	3	20	

After

```
    Class   Sex     Adult_or_Child  Survived    Numbers
0   1st     Male    Adult               No      118
1   1st     Male    Adult               Yes     57
2   1st     Male    Child               No      0
3   1st     Male    Child               Yes     5
4   1st     Female  Adult               No      4
5   1st     Female  Adult               Yes     140
6   1st     Female  Child               No      0
7   1st     Female  Child               Yes     1
8   2nd     Male    Adult               No      154
9   2nd     Male    Adult               Yes     14
10  2nd     Male    Child               No      0
11  2nd     Male    Child               Yes     11
12  2nd     Female  Adult               No      13
13  2nd     Female  Adult               Yes     80
14  2nd     Female  Child               No      0
15  2nd     Female  Child               Yes     13
...
```

Excel Tables:



	A	B	C	D	E	F	G	H	I	J	K
1	Name	Age	Height		Name	Age	Height		Name	Age	Height
2	Tolu	24	2		ragnar	32	1.6		zeus	65	1.9
3	Chukwuka	50	1.8		bjorn	49	2.2		poseidon	26	2.2
4	Ogor	15	1.5		loki	400	1.8		athena	30	1.4
5											
6											
7											
8											
9											
10											
11											
12											
13											
14											
15											
16											
17											
18											
19											
20											
21											

naija | vikings | olympus | position | defined_tables

Pandas is not aware of tables:

```
pd.read_excel(io = "names.xlsx", sheet_name = "defined_tables")
```

	Name	Age	Height	Unnamed: 3	Name.1	Age.1	Height.1	Unnamed: 7	Name.2	Age.2	Height.2
0	Tolu	24	2.0	NaN	ragnar	32	1.6	NaN	zeus	65	1.9
1	Chukwuka	50	1.8	NaN	bjorn	49	2.2	NaN	poseidon	26	2.2
2	Ogor	15	1.5	NaN	loki	400	1.8	NaN	athena	30	1.4

This can be remedied using Openpyxl

Openpyxl has a *tables* method, that collects all tables in a sheet into a dictionary:

```
# read excel file
wb = load_workbook(filename = "names.xlsx")

# read specific sheet
ws = wb["defined_tables"]

# view keys in the tables_dictionary
# the keys are the table names
ws.tables.keys()

dict_keys(['naija', 'olympus', 'vikings'])
```

Let’s peek into values of one of the *tables* :

```
ws.tables['naija']

<openpyxl.worksheet.table.Table object>
Parameters:
id=1, name='naija', displayName='naija', comment=None, ref='A1:C4', tableType=None, headerRowCount=1,
insertRow=None, insertRowShift=None, totalsRowCount=None, totalsRowShown=False, published=None,
headerRowDxfId=None, dataDxfId=None, totalsRowDxfId=None, headerRowBorderDxfId=None,
tableBorderDxfId=None, totalsRowBorderDxfId=None, headerRowCellStyle=None, dataCellStyle=None,
totalsRowCellStyle=None, connectionId=None, autoFilter=<openpyxl.worksheet.filters.AutoFilter object>
Parameters:
ref='A1:C4', filterColumn=[], sortState=None, sortState=None, tableColumns=
[<openpyxl.worksheet.table.TableColumn object>
Parameters:
id=1, uniqueName=None, name='Name', totalsRowFunction=None, totalsRowLabel=None,
queryTableFieldId=None, headerRowDxfId=None, dataDxfId=None, totalsRowDxfId=None,
headerRowCellStyle=None, dataCellStyle=None, totalsRowCellStyle=None, calculatedColumnFormula=None,
totalsRowFormula=None, xmlColumnPr=None, extLst=None, <openpyxl.worksheet.table.TableColumn object>
Parameters:
id=2, uniqueName=None, name='Age', totalsRowFunction=None, totalsRowLabel=None, queryTableFieldId=None,
headerRowDxfId=None, dataDxfId=None, totalsRowDxfId=None, headerRowCellStyle=None, dataCellStyle=None,
totalsRowCellStyle=None, calculatedColumnFormula=None, totalsRowFormula=None, xmlColumnPr=None,
extLst=None, <openpyxl.worksheet.table.TableColumn object>
Parameters:
id=3, uniqueName=None, name='Height', totalsRowFunction=None, totalsRowLabel=None,
queryTableFieldId=None, headerRowDxfId=None, dataDxfId=None, totalsRowDxfId=None,
headerRowCellStyle=None, dataCellStyle=None, totalsRowCellStyle=None, calculatedColumnFormula=None,
totalsRowFormula=None, xmlColumnPr=None, extLst=None], tableStyleInfo=
<openpyxl.worksheet.table.TableStyleInfo object>
Parameters:
name='TableStyleMedium2', showFirstColumn=False, showLastColumn=False, showRowStripes=True,
showColumnStripes=False
```

data location

All that is left is to iterate through the dictionary and collate the values per key

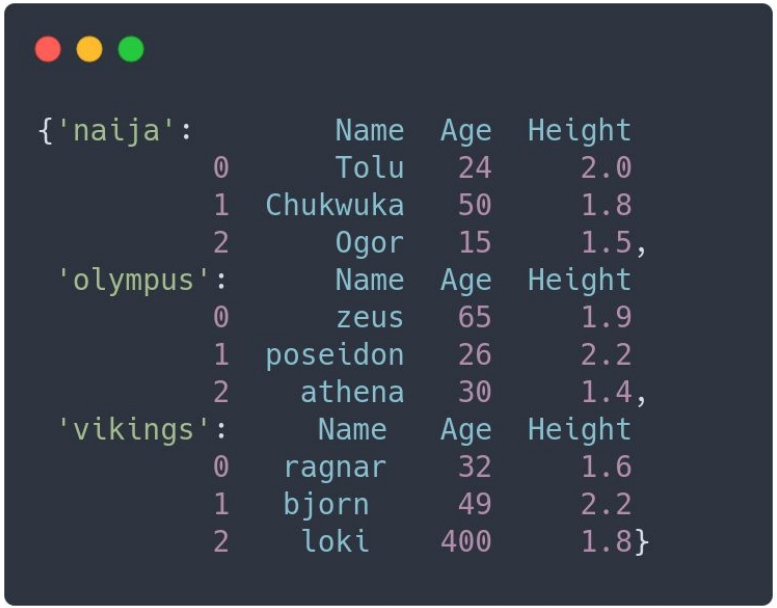
```
#read excel file
wb  = load_workbook(filename = "names.xlsx")

#read specific sheet
ws = wb['defined_tables']

named_tables = {}

# the value here is the *ref*
for table_name, value in ws.tables.items():
    dataframe = ws[value]
    header, *body = [[cell.value for cell in row]
                     for row in dataframe]
    dataframe = pd.DataFrame(body, columns = header)
    named_tables[table_name] = dataframe
```

Let’s peek into the *named_tables* dictionary :



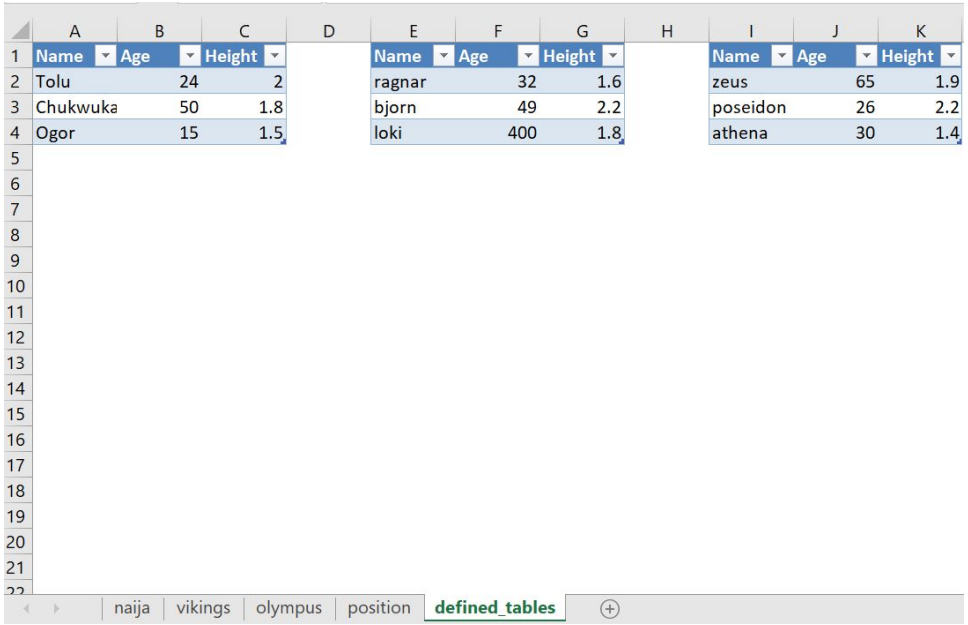
```
{'naija':        Name  Age  Height
          0      Tolu   24     2.0
          1  Chukwuka   50     1.8
          2      Ogor   15     1.5,
 'olympus':      Name  Age  Height
          0      zeus   65     1.9
          1  poseidon   26     2.2
          2    athena   30     1.4,
 'vikings':     Name   Age  Height
          0   ragnar    32     1.6
          1   bjorn     49     2.2
          2    loki    400     1.8}
```

We can combine into one dataframe:

```
pd.concat(named_tables).droplevel(-1).rename_axis(index="Table_name").reset_index()
```

	Table_name	Name	Age	Height
0	naija	Tolu	24	2.0
1	naija	Chukwuka	50	1.8
2	naija	Ogor	15	1.5
3	olympus	zeus	65	1.9
4	olympus	poseidon	26	2.2
5	olympus	athena	30	1.4
6	vikings	ragnar	32	1.6
7	vikings	bjorn	49	2.2
8	vikings	loki	400	1.8

Before



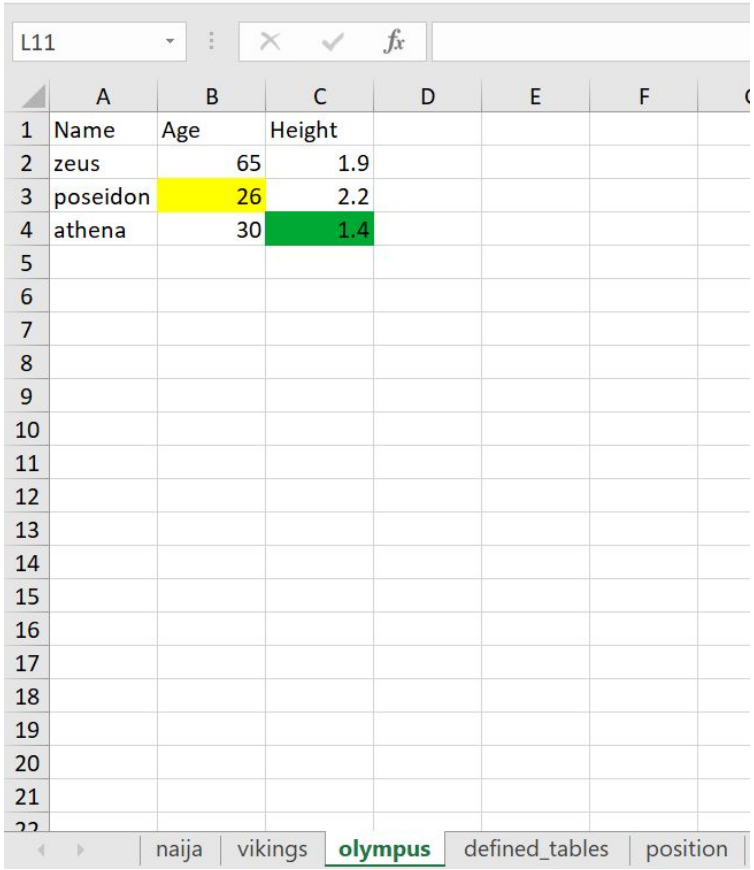
	A	B	C	D	E	F	G	H	I	J	K
1	Name	Age	Height		Name	Age	Height		Name	Age	Height
2	Tolu	24	2		ragnar	32	1.6		zeus	65	1.9
3	Chukwuka	50	1.8		bjorn	49	2.2		poseidon	26	2.2
4	Ogor	15	1.5		loki	400	1.8		athena	30	1.4
5											
6											
7											
8											
9											
10											
11											
12											
13											
14											
15											
16											
17											
18											
19											
20											
21											

naija | vikings | olympus | position | defined_tables

After

```
    Table_name  Name    Age Height
0   naija       Tolu    24  2.0
1   naija   Chukwuka    50  1.8
2   naija       Ogor    15  1.5
3   olympus     zeus    65  1.9
4   olympus poseidon    26  2.2
5   olympus     athena  30  1.4
6   vikings     ragnar  32  1.6
7   vikings     bjorn   49  2.2
8   vikings     loki    400 1.8
```

Conditional Formatting



Yellow could mean further investigation,
while green could mean approved.
The colours are meant to provide context.

Pandas has no idea about the cell metadata - it just focuses on the cell value:

```
pd.read_excel("names.xlsx", "olympus")

    Name        Age  Height
0   zeus         65  1.9
1   poseidon     26  2.2
2   athena       30  1.4
```

How do we get the colour information? Openpyxl !

- Read in data with Openpyxl:

```
#read excel file
wb  = load_workbook(filename = "names.xlsx")

#read specific sheet
ws = wb["olympus"]

data = defaultdict(list)
for column in ws.columns:
    for cell in column:
        if cell.data_type == "n":
          # keep the cell metadata close to the cell value
            cell_value = ",".join((str(cell.value),
                                   cell.fill.fgColor.rgb))
            data[f"C{cell.column}"].append(cell_value)
        else:
            data[f"C{cell.column}"].append(cell.value)

data

defaultdict(list,
            {'C1': ['Name', 'zeus', 'poseidon', 'athena'],
             'C2': ['Age', '65,00000000', '26,FFFFFF00', '30,00000000'],
             'C3': ['Height', '1.9,00000000', '2.2,00000000', '1.4,FF00A933']})
```

Read in data and tidy :

```
pd.DataFrame(data)
    C1          C2            C3
0   Name        Age           Height
1   zeus        65,00000000   1.9,00000000
2   poseidon    26,FFFFFF00   2.2,00000000
3   athena      30,00000000   1.4,FF00A933
```

- Promote first row as header :

```
(pd.DataFrame(data)
 .row_to_names(row_number = 0,remove_row = True)
)
```

	Name	Age	Height
1	zeus	65,00000000	1.9,00000000
2	poseidon	26,FFFFFF00	2.2,00000000
3	athena	30,00000000	1.4,FF00A933

- Set Name column as the index, in preparation for stacking :

```
(pd.DataFrame(data)
 .row_to_names(row_number = 0,remove_row = True)
 .set_index("Name")
)
```

Name	Age	Height
zeus	65,00000000	1.9,00000000
poseidon	26,FFFFFF00	2.2,00000000
athena	30,00000000	1.4,FF00A933

- Stack the columns, which will allow us separate the colour data from the numbers :

```
(pd.DataFrame(data)
 .row_to_names(row_number = 0,remove_row = True)
 .set_index("Name")
 .stack()
)
```

```
Name
zeus      Age         65,00000000
          Height    1.9,00000000
poseidon  Age         26,FFFFFF00
          Height    2.2,00000000
athena    Age         30,00000000
          Height    1.4,FF00A933
dtype: object
```

- Now, we can separate the numbers from the colour information :

```
(pd.DataFrame(data)
 .row_to_names(row_number = 0,remove_row = True)
 .set_index("Name")
 .stack()
 .str.split(",",expand=True)
)
```

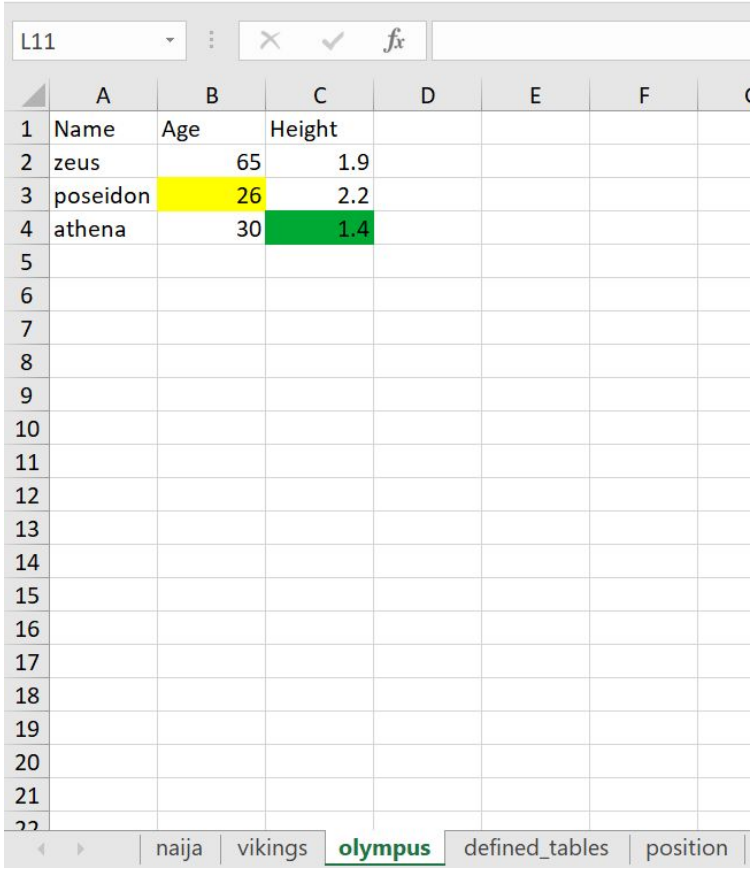
		0	1
Name			
zeus	Age	65	00000000
	Height	1.9	00000000
poseidon	Age	26	FFFFFF00
	Height	2.2	00000000
athena	Age	30	00000000
	Height	1.4	FF00A933

- Add column names and tidy up :

```
(pd.DataFrame(data)
 .row_to_names(row_number = 0,remove_row = True)
 .set_index("Name")
 .stack()
 .str.split(",",expand=True)
 .set_axis(["Value","Colour"], axis="columns")
 .rename_axis(index=["Name","Variable"])
 .reset_index()
)
```

	Name	Variable	Value	Colour
0	zeus	Age	65	00000000
1	zeus	Height	1.9	00000000
2	poseidon	Age	26	FFFFFF00
3	poseidon	Height	2.2	00000000
4	athena	Age	30	00000000
5	athena	Height	1.4	FF00A933

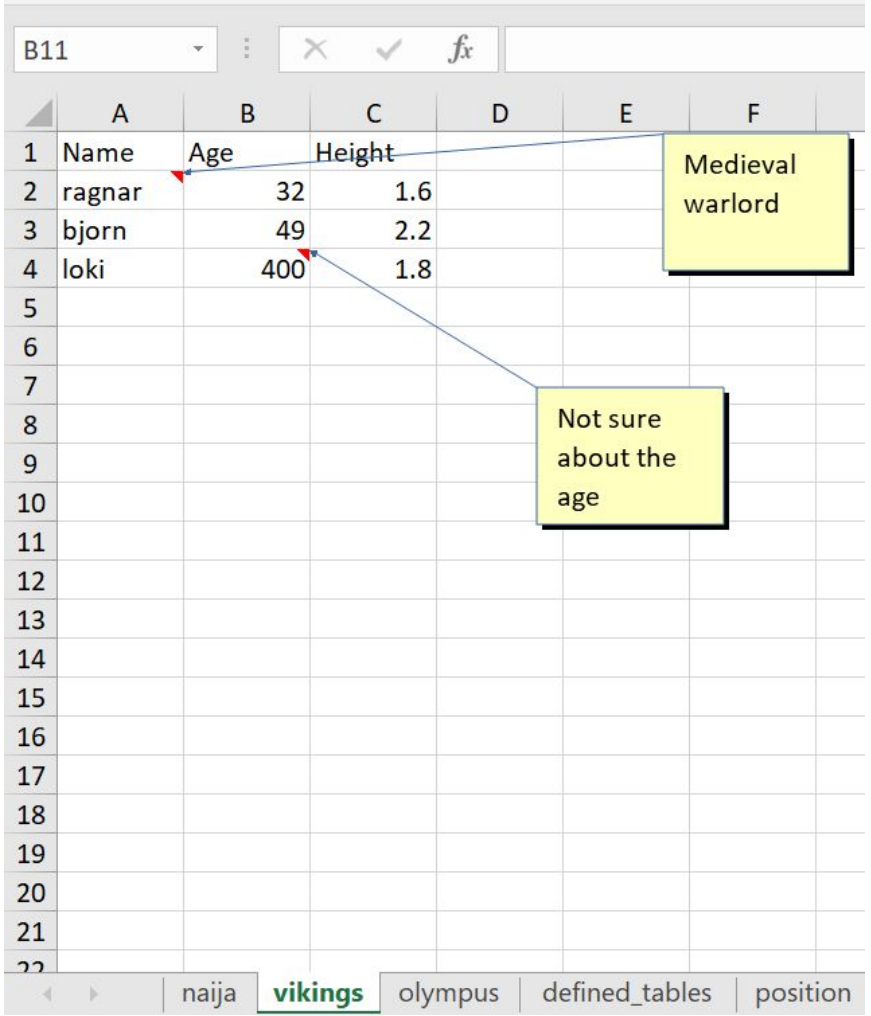
Before



After

```
       Name    Variable    Value    Colour
0      zeus    Age         65       00000000
1      zeus    Height      1.9      00000000
2  poseidon    Age         26       FFFFFF00
3  poseidon    Height      2.2      00000000
4    athena    Age         30       00000000
5    athena    Height      1.4      FF00A933
```

Read data with embedded comments :



Comments can be accessed with Openpyxl:

```
#read excel file
wb  = load_workbook(filename = "names.xlsx")

#read specific sheet
ws = wb["vikings"]

data = defaultdict(list)
comments = {}
for column in ws.columns:
    for cell in column:
        if cell.comment:
            comments[cell.row] = cell.comment.text
        data[cell.row].append(cell.value)

data
defaultdict(list,
            {1: ['Name', 'Age', 'Height'],
             2: ['ragnar', 32, 1.6],
             3: ['bjorn', 49, 2.2],
             4: ['loki', 400, 1.8]})

comments
{2: 'Medieval warlord', 4: 'Not sure about the age'}
```

Let’s get the data into one dictionary:

```
new_data = {}
for key, value in data.items():
    value.append(comments.get(key, None))
    new_data[key] = value

new_data
```

```
{1: ['Name', 'Age', 'Height', None],
 2: ['ragnar', 32, 1.6, 'Medieval warlord'],
 3: ['bjorn', 49, 2.2, None],
 4: ['loki', 400, 1.8, 'Not sure about the age']}
```

Now we can read the data into a dataframe and clean :

```
pd.DataFrame(new_data.values())
```

	0	1	2	3
0	Name	Age	Height	None
1	ragnar	32	1.6	Medieval warlord
2	bjorn	49	2.2	None
3	loki	400	1.8	Not sure about the age

Let’s replace the first *None* with something meaningful:

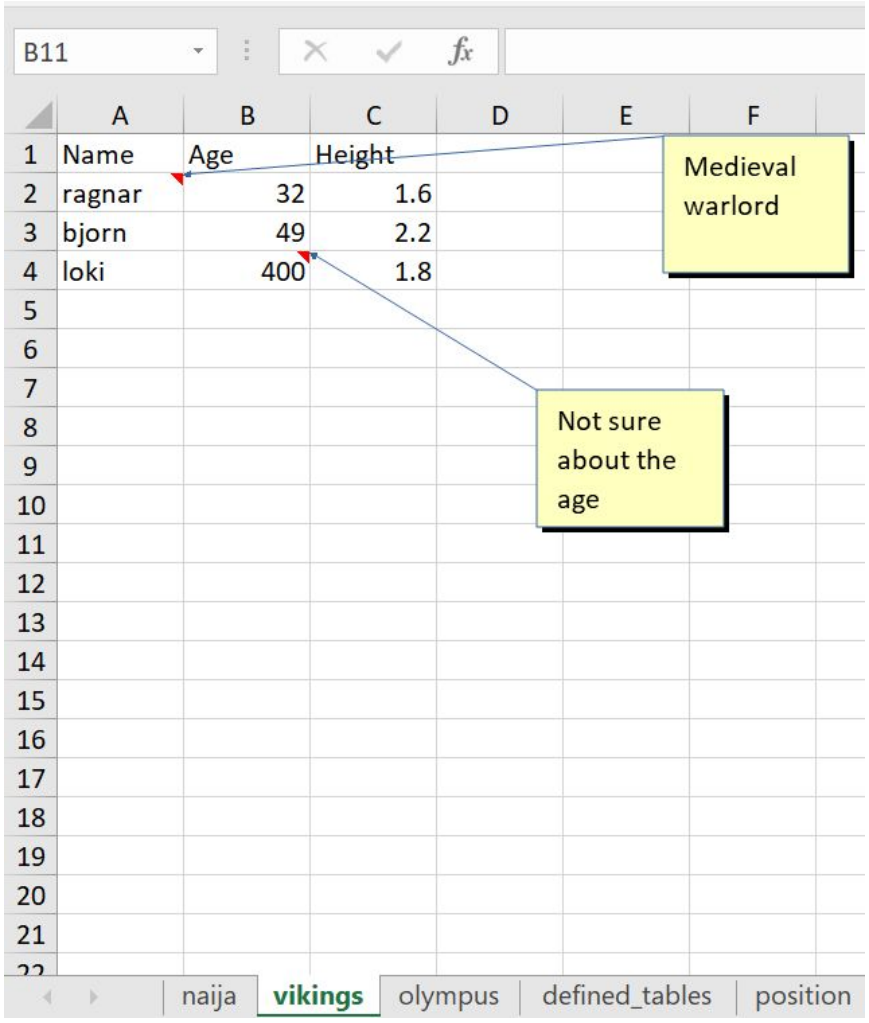
```
(pd.DataFrame(new_data.values())
 .fillna("comments", limit=1))
```

	0	1	2	3
0	Name	Age	Height	comments
1	ragnar	32	1.6	Medieval warlord
2	bjorn	49	2.2	None
3	loki	400	1.8	Not sure about the age

Promote rows to headers :

```
(pd.DataFrame(new_data.values())
 .fillna("comments", limit=1)
 .row_to_names(0,True)
)
```

```
    Name    Age Height  comments
1   ragnar  32  1.6     Medieval warlord
2   bjorn   49  2.2     None
3   loki    400 1.8     Not sure about the age
```

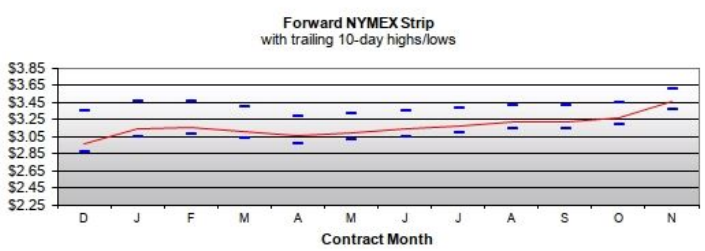



```
    Name    Age Height  comments
1   ragnar  32  1.6     Medieval warlord
2   bjorn   49  2.2     None
3   loki    400 1.8     Not sure about the age
```

Small Multiples :

Forward Prices (US$/MMBtu)

	NYMEX		IF NWPL Rocky Mountains			
			Fixed Price		Basis	
	SETTLE	Δ	BID	OFFER	BID	OFFER
Cash			1.890	1.910		
ROM			2.060	2.080		
Dec-01	2.960	0.090	2.395	2.415	(0.565)	(0.545)
Dec-01 to Mar-02	3.088	0.083	2.594	2.614	(0.494)	(0.474)
Apr-02 to Oct-02	3.166	0.084	2.581	2.601	(0.585)	(0.565)
Nov-02 to Mar-03	3.651	0.090	3.356	3.376	(0.295)	(0.275)
One Year Strip*	3.165	0.084	2.634	2.654	(0.530)	(0.510)



	IF CIG Rocky Mountains				IF EL Paso Permian				IF EL Paso San Juan			
	Fixed Price		Basis		Fixed Price		Basis		Fixed Price		Basis	
	BID	OFFER	BID	OFFER	BID	OFFER	BID	OFFER	BID	OFFER	BID	OFFER
Cash	1.940	1.960			2.375	2.395			2.450	2.470		
ROM	1.960	1.980			2.420	2.440			2.350	2.370		
Dec-01	2.345	2.365	(0.615)	(0.595)	2.700	2.720	(0.260)	(0.240)	2.560	2.580	(0.400)	(0.380)
Dec-01 to Mar-02	2.548	2.568	(0.540)	(0.520)	2.855	2.875	(0.233)	(0.213)	2.743	2.763	(0.345)	(0.325)
Apr-02 to Oct-02	2.471	2.491	(0.695)	(0.675)	3.009	3.029	(0.158)	(0.138)	2.801	2.821	(0.365)	(0.345)
Nov-02 to Mar-03	3.311	3.331	(0.340)	(0.320)	3.499	3.519	(0.153)	(0.133)	3.421	3.441	(0.230)	(0.210)
One Year Strip*	2.551	2.571	(0.614)	(0.594)	2.982	3.002	(0.182)	(0.162)	2.817	2.837	(0.347)	(0.327)

	AECO / NIT				IF NWPL Canadian Border (Sumas)				IF PEPL TX-OK			
	Fixed Price		Basis		Fixed Price		Basis		Fixed Price		Basis	
	BID	OFFER	BID	OFFER	BID	OFFER	BID	OFFER	BID	OFFER	BID	OFFER
Cash	2.376	2.396			2.480	2.500			2.530	2.550		
ROM	2.398	2.418			2.460	2.480			2.530	2.550		
Dec-01	2.552	2.572	(0.408)	(0.388)	2.800	2.820	(0.160)	(0.140)	2.828	2.848	(0.133)	(0.113)
Dec-01 to Mar-02	2.616	2.636	(0.472)	(0.452)	2.892	2.912	(0.196)	(0.176)	2.958	2.978	(0.130)	(0.110)
Apr-02 to Oct-02	2.661	2.681	(0.505)	(0.485)	2.796	2.816	(0.370)	(0.350)	3.046	3.066	(0.120)	(0.100)
Nov-02 to Mar-03	3.216	3.236	(0.435)	(0.415)	3.706	3.726	0.055	0.075	3.531	3.551	(0.120)	(0.100)
One Year Strip*	2.676	2.696	(0.488)	(0.468)	2.880	2.900	(0.285)	(0.265)	3.041	3.061	(0.123)	(0.103)

	NGI Socal (South Cal Border)				NGI Malin (North Cal Border)				PG&E City Gate			
	Fixed Price		Basis		Fixed Price		Basis		Fixed Price		Basis	
	BID	OFFER	BID	OFFER	BID	OFFER	BID	OFFER	BID	OFFER	BID	OFFER
Cash	2.580	2.600			2.550	2.570			2.570	2.590		
ROM	2.500	2.520			2.480	2.500			2.520	2.540		

Image is from [unpivotr]

Forward Prices (US$/MMBtu)

	NYMEX	
	SETTLE	Δ
Cash		
ROM		
Dec-01	2.960	0.090
Dec-01 to Mar-02	3.088	0.083
Apr-02 to Oct-02	3.166	0.084
Nov-02 to Mar-03	3.651	0.090
One Year Strip*	3.165	0.084

	IF NWPL Rocky Mountains			
	Fixed Price		Basis	
	BID	OFFER	BID	OFFER
	1.890	1.910		
	2.060	2.080		
	2.395	2.415	(0.565)	(0.545)
	2.594	2.614	(0.494)	(0.474)
	2.581	2.601	(0.585)	(0.565)
	3.356	3.376	(0.295)	(0.275)
	2.634	2.654	(0.530)	(0.510)

	IF CIG Rocky Mountains			
	Fixed Price		Basis	
	BID	OFFER	BID	OFFER
Cash	1.940	1.960		
ROM	1.960	1.980		
Dec-01	2.345	2.365	(0.615)	(0.595)
Dec-01 to Mar-02	2.548	2.568	(0.540)	(0.520)
Apr-02 to Oct-02	2.471	2.491	(0.695)	(0.675)
Nov-02 to Mar-03	3.311	3.331	(0.340)	(0.320)
One Year Strip*	2.551	2.571	(0.614)	(0.594)

	IF EL Paso Permian			
	Fixed Price		Basis	
	BID	OFFER	BID	OFFER
Cash	2.375	2.395		
ROM	2.420	2.440		
Dec-01	2.700	2.720	(0.260)	(0.240)
Dec-01 to Mar-02	2.855	2.875	(0.233)	(0.213)
Apr-02 to Oct-02	3.009	3.029	(0.158)	(0.138)
Nov-02 to Mar-03	3.499	3.519	(0.153)	(0.133)
One Year Strip*	2.982	3.002	(0.182)	(0.162)

	IF EL Paso San Juan			
	Fixed Price		Basis	
	BID	OFFER	BID	OFFER
Cash	2.450	2.470		
ROM	2.350	2.370		
Dec-01	2.560	2.580	(0.400)	(0.380)
Dec-01 to Mar-02	2.743	2.763	(0.345)	(0.325)
Apr-02 to Oct-02	2.801	2.821	(0.365)	(0.345)
Nov-02 to Mar-03	3.421	3.441	(0.230)	(0.210)
One Year Strip*	2.817	2.837	(0.347)	(0.327)

	AECO / NIT			
	Fixed Price		Basis	
	BID	OFFER	BID	OFFER
Cash	2.376	2.396		
ROM	2.398	2.418		
Dec-01	2.552	2.572	(0.408)	(0.388)
Dec-01 to Mar-02	2.616	2.636	(0.472)	(0.452)
Apr-02 to Oct-02	2.661	2.681	(0.505)	(0.485)
Nov-02 to Mar-03	3.216	3.236	(0.435)	(0.415)
One Year Strip*	2.676	2.696	(0.488)	(0.468)

	IF NWPL Canadian Border (Sumas)			
	Fixed Price		Basis	
	BID	OFFER	BID	OFFER
Cash	2.480	2.500		
ROM	2.460	2.480		
Dec-01	2.800	2.820	(0.160)	(0.140)
Dec-01 to Mar-02	2.892	2.912	(0.196)	(0.176)
Apr-02 to Oct-02	2.796	2.816	(0.370)	(0.350)
Nov-02 to Mar-03	3.706	3.726	0.055	0.075
One Year Strip*	2.880	2.900	(0.285)	(0.265)

	IF PEPL TX-OK			
	Fixed Price		Basis	
	BID	OFFER	BID	OFFER
Cash	2.530	2.550		
ROM	2.530	2.550		
Dec-01	2.828	2.848	(0.133)	(0.113)
Dec-01 to Mar-02	2.958	2.978	(0.130)	(0.110)
Apr-02 to Oct-02	3.046	3.066	(0.120)	(0.100)
Nov-02 to Mar-03	3.531	3.551	(0.120)	(0.100)
One Year Strip*	3.041	3.061	(0.123)	(0.103)

	NGI Socal (South Cal Border)			
	Fixed Price		Basis	
	BID	OFFER	BID	OFFER
Cash	2.580	2.600		
ROM	2.500	2.520		

	NGI Malin (North Cal Border)			
	Fixed Price		Basis	
	BID	OFFER	BID	OFFER
Cash	2.550	2.570		
ROM	2.480	2.500		

	PG&E City Gate			
	Fixed Price		Basis	
	BID	OFFER	BID	OFFER
Cash	2.570	2.590		
ROM	2.520	2.540		

Forward Prices (US$/MMBtu)

	NYMEX SETTLE	NYMEX Δ
Cash		
ROM		
Dec-01	2.960	0.090
Dec-01 to Mar-02	3.088	0.083
Apr-02 to Oct-02	3.166	0.084
Nov-02 to Mar-03	3.651	0.090
One Year Strip*	3.165	0.084

	IF NWPL Rocky Mountains			
	Fixed Price		Basis	
	BID	OFFER	BID	OFFER
Cash	1.890	1.910		
ROM	2.060	2.080		
Dec-01	2.395	2.415	(0.565)	(0.545)
Dec-01 to Mar-02	2.594	2.614	(0.494)	(0.474)
Apr-02 to Oct-02	2.581	2.601	(0.585)	(0.565)
Nov-02 to Mar-03	3.356	3.376	(0.295)	(0.275)
One Year Strip*	2.634	2.654	(0.530)	(0.510)

	IF CIG Rocky Mountains			
	Fixed Price		Basis	
	BID	OFFER	BID	OFFER
Cash	1.940	1.960		
ROM	1.960	1.980		
Dec-01	2.345	2.365	(0.615)	(0.595)
Dec-01 to Mar-02	2.548	2.568	(0.540)	(0.520)
Apr-02 to Oct-02	2.471	2.491	(0.695)	(0.675)
Nov-02 to Mar-03	3.311	3.331	(0.340)	(0.320)
One Year Strip*	2.551	2.571	(0.614)	(0.594)

	IF EL Paso Permian			
	Fixed Price		Basis	
	BID	OFFER	BID	OFFER
Cash	2.375	2.395		
ROM	2.420	2.440		
Dec-01	2.700	2.720	(0.260)	(0.240)
Dec-01 to Mar-02	2.855	2.875	(0.233)	(0.213)
Apr-02 to Oct-02	3.009	3.029	(0.158)	(0.138)
Nov-02 to Mar-03	3.499	3.519	(0.153)	(0.133)
One Year Strip*	2.982	3.002	(0.182)	(0.162)

	IF EL Paso San Juan			
	Fixed Price		Basis	
	BID	OFFER	BID	OFFER
Cash	2.450	2.470		
ROM	2.350	2.370		
Dec-01	2.560	2.580	(0.400)	(0.380)
Dec-01 to Mar-02	2.743	2.763	(0.345)	(0.325)
Apr-02 to Oct-02	2.801	2.821	(0.365)	(0.345)
Nov-02 to Mar-03	3.421	3.441	(0.230)	(0.210)
One Year Strip*	2.817	2.837	(0.347)	(0.327)

	AECO / NIT			
	Fixed Price		Basis	
	BID	OFFER	BID	OFFER
Cash	2.376	2.396		
ROM	2.398	2.418		
Dec-01	2.552	2.572	(0.408)	(0.388)
Dec-01 to Mar-02	2.616	2.636	(0.472)	(0.452)
Apr-02 to Oct-02	2.661	2.681	(0.505)	(0.485)
Nov-02 to Mar-03	3.216	3.236	(0.435)	(0.415)
One Year Strip*	2.676	2.696	(0.488)	(0.468)

	IF NWPL Canadian Border (Sumas)			
	Fixed Price		Basis	
	BID	OFFER	BID	OFFER
Cash	2.480	2.500		
ROM	2.460	2.480		
Dec-01	2.800	2.820	(0.160)	(0.140)
Dec-01 to Mar-02	2.892	2.912	(0.196)	(0.176)
Apr-02 to Oct-02	2.796	2.816	(0.370)	(0.350)
Nov-02 to Mar-03	3.706	3.726	0.055	0.075
One Year Strip*	2.880	2.900	(0.285)	(0.265)

	IF PEPL TX-OK			
	Fixed Price		Basis	
	BID	OFFER	BID	OFFER
Cash	2.530	2.550		
ROM	2.530	2.550		
Dec-01	2.828	2.848	(0.133)	(0.113)
Dec-01 to Mar-02	2.958	2.978	(0.130)	(0.110)
Apr-02 to Oct-02	3.046	3.066	(0.120)	(0.100)
Nov-02 to Mar-03	3.531	3.551	(0.120)	(0.100)
One Year Strip*	3.041	3.061	(0.123)	(0.103)

	NGI Socal (South Cal Border)			
	Fixed Price		Basis	
	BID	OFFER	BID	OFFER
Cash	2.580	2.600		
ROM	2.500	2.520		

	NGI Malin (North Cal Border)			
	Fixed Price		Basis	
	BID	OFFER	BID	OFFER
Cash	2.550	2.570		
ROM	2.480	2.500		

	PG&E City Gate			
	Fixed Price		Basis	
	BID	OFFER	BID	OFFER
Cash	2.570	2.590		
ROM	2.520	2.540		

Get the locations of the tables:

```
#read excel file
wb  = load_workbook(filename = "enron.xlsx")

#read specific sheet
ws = wb["Report"]

rows = set()
columns = set()
data = []
for row in ws.rows:
    for cell in row:
        if cell.value in ("BID", "OFFER") :
            rows.add(cell.row)
            columns.add(cell.column)
        data.append((cell.value, cell.row, cell.column))

rows
{16, 27, 38, 49}

columns
{7, 8, 9, 10, 12, 13, 14, 15, 17, 18, 19, 20}
```

Let’s peek into *data* :

```
[(None, 1, 1),
 ('Enron North America - West Gas', 1, 2),
 (None, 1, 3),
 (None, 1, 4),
 (None, 1, 5),
 (None, 1, 6),
 (None, 1, 7),
 (None, 1, 8),
 (None, 1, 9),
 (None, 1, 10),
 (None, 1, 11),
 (None, 1, 12),
 (None, 1, 13),
 (None, 1, 14),
 (None, 1, 15),
 (None, 1, 16),
 (None, 1, 17),
 (None, 1, 18),
 (None, 1, 19),
 (None, 1, 20),
 (None, 1, 21),
 (None, 1, 22),
 (None, 1, 23),
 (None, 1, 24),
 (None, 1, 25),
 (None, 1, 26),
 ('NYMEX Month', 1, 27),
 (None, 1, 28),
 ...
```

Create the row and column boundaries:

```
# two rows above BID/OFFER, eight rows below
rows = [range(n-2, n+9) for n in sorted(rows)]
rows
[range(14, 25), range(25, 36), range(36, 47), range(47, 58)]

columns = sorted(columns)
# each table has four columns
columns = [columns[slice(n,n+4)] for n in range(0, len(columns), 4)]
columns
[[7, 8, 9, 10], [12, 13, 14, 15], [17, 18, 19, 20]]
```

Get values within pairs of row and column boundaries:

```
from itertools import product

data_boundary = defaultdict(list)

for (value, row, column), row_range, column_list in product(data, rows, columns):
    if row in row_range and column in column_list:
        data_boundary[(row_range, tuple(column_list))].append(value)
```

Peek into *data_boundary* :

```
defaultdict(list,
            {(range(14, 25), (7, 8, 9, 10)): [None,
              None,
              None,
              None,
              None,
              None,
              None,
              None,
              None,
              None,
              None,
              None,
              None,
              None,
              None,
              None,
              None,
              None,
              None,
              None,
              None,
              None,
              None,
              None,
              None,
              None,
              None,
              None,
              None,
              None,
              None,
              None,
              None,
              None,
              None,
              None,
              None,
              ...
```

Reshape and read into dataframe:

```
                # remember each table has four columns
numpy_data = np.vstack([np.reshape(value,(-1,4))
                        for key, value
                        in data_boundary.items()])

df = pd.DataFrame(numpy_data).remove_empty().add_prefix("A_")
```

Peek into *numpy_data* :

```
numpy_data
array([[None, None, None, None],
       [None, None, None, None],
       [None, None, None, None],
       [None, None, None, None],
       [None, None, None, None],
       [None, None, None, None],
       [None, None, None, None],
       [None, None, None, None],
       [None, None, None, None],
       [None, None, None, None],
       [None, None, None, None],
       [None, None, None, None],
       [None, None, None, None],
       [None, None, None, None],
       [None, None, None, None],
       [None, None, None, None],
       [None, None, None, None],
       [None, None, None, None],
       [None, None, None, None],
       [None, None, None, None],
       [None, None, None, None],
       [None, None, None, None],
       [None, None, None, None],
       ['IF NWPL Rocky Mountains', None, None, None],
       ['Fixed Price', None, 'Basis', None],
       ['BID', 'OFFER', 'BID', 'OFFER'],
       [1.89, 1.91, None, None],
       [2.06, 2.08, None, None], ...
```

Peek into *df* :

```
df.head(10)
```

	A_0	A_1	A_2	A_3
0	IF NWPL Rocky Mountains	None	None	None
1	Fixed Price	None	Basis	None
2	BID	OFFER	BID	OFFER
3	1.89	1.91	None	None
4	2.06	2.08	None	None
5	2.395	2.415	-0.565	-0.545
6	2.594	2.614	-0.49375	-0.47375
7	2.58129	2.60129	-0.585	-0.565
8	3.356	3.376	-0.295	-0.275
9	2.63408	2.65408	-0.530417	-0.510417

Let’s tidy the dataframe :

```
result = (df
          .assign(company_name = np.where(df.isna().sum(1)==3,
                                          df.iloc[:,0],
                                          None))
           # pyjanitor
          .fill_direction({"company_name":"down"})
          .set_axis(["Fixed_Price_Bid", "Fixed_Price_Offer",
                     "Basis_Bid", "Basis_Offer", "company_name"],
                     axis='columns')
         .query("Fixed_Price_Bid != company_name and Fixed_Price_Bid != ['Fixed Price', 'BID']")
           # pyjanitor
         .reorder_columns(["company_name"])
         .reset_index(drop=True)
         )
```

A peek into the results dataframe :

```
result.head(10)
```

	company_name	Fixed_Price_Bid	Fixed_Price_Offer	Basis_Bid	Basis_Offer
0	IF NWPL Rocky Mountains	1.89	1.91	None	None
1	IF NWPL Rocky Mountains	2.06	2.08	None	None
2	IF NWPL Rocky Mountains	2.395	2.415	-0.565	-0.545
3	IF NWPL Rocky Mountains	2.594	2.614	-0.49375	-0.47375
4	IF NWPL Rocky Mountains	2.58129	2.60129	-0.585	-0.565
5	IF NWPL Rocky Mountains	3.356	3.376	-0.295	-0.275
6	IF NWPL Rocky Mountains	2.63408	2.65408	-0.530417	-0.510417
7	IF CIG Rocky Mountains	1.94	1.96	None	None
8	IF CIG Rocky Mountains	1.96	1.98	None	None
9	IF CIG Rocky Mountains	2.345	2.365	-0.615	-0.595

The complete code :

```
from itertools import product

wb  = load_workbook(filename = "enron.xlsx")
ws = wb["Report"]

rows = set()
columns = set()

data = []
for row in ws.rows:
    for cell in row:
        if cell.value in ("BID", "OFFER") :
            rows.add(cell.row)
            columns.add(cell.column)
        data.append((cell.value, cell.row, cell.column))

rows = [range(n-2, n+9) for n in sorted(rows)]

columns = sorted(columns)
columns = [columns[slice(n,n+4)] for n in range(0, len(columns), 4)]

data_boundary = defaultdict(list)
for (value, row, column), row_range, column_list in product(data, rows, columns):
    if row in row_range and column in column_list:
        data_boundary[(row_range, tuple(column_list))].append(value)

numpy_data = np.vstack([np.reshape(value,(-1,4))
                        for key, value
                        in data_boundary.items()])

df = (pd.DataFrame(numpy_data)
     .remove_empty()
     .add_prefix("A_")
     .assign(company_name = lambda df: np.where( df.isna().sum(1)==3,
                                                 df.iloc[:,0],
                                                 None))
     .fill_direction({"company_name":"down"})
     .set_axis(["Fixed_Price_Bid", "Fixed_Price_Offer",
                "Basis_Bid", "Basis_Offer", "company_name"],
               axis='columns')
      .query("Fixed_Price_Bid != company_name and Fixed_Price_Bid != ['Fixed Price', 'BID']")
     .reorder_columns(["company_name"])
     .reset_index(drop=True)
    )
```

Before

Forward Prices (US$/MMBtu)

	NYMEX		IF NWPL Rocky Mountains			
	SETTLE	Δ	Fixed Price BID	Fixed Price OFFER	Basis BID	Basis OFFER
Cash			1.890	1.910		
ROM			2.060	2.080		
Dec-01	2.960	0.090	2.395	2.415	(0.565)	(0.545)
Dec-01 to Mar-02	3.088	0.083	2.594	2.614	(0.494)	(0.474)
Apr-02 to Oct-02	3.166	0.084	2.581	2.601	(0.585)	(0.565)
Nov-02 to Mar-03	3.651	0.090	3.356	3.376	(0.295)	(0.275)
One Year Strip*	3.165	0.084	2.634	2.654	(0.530)	(0.510)

Forward NYMEX Strip
with trailing 10-day highs/lows

	IF CIG Rocky Mountains				IF EL Paso Permian				IF EL Paso San Juan			
	Fixed Price BID	Fixed Price OFFER	Basis BID	Basis OFFER	Fixed Price BID	Fixed Price OFFER	Basis BID	Basis OFFER	Fixed Price BID	Fixed Price OFFER	Basis BID	Basis OFFER
Cash	1.940	1.960			2.375	2.395			2.450	2.470		
ROM	1.960	1.980			2.420	2.440			2.350	2.370		
Dec-01	2.345	2.365	(0.615)	(0.595)	2.700	2.720	(0.260)	(0.240)	2.560	2.580	(0.400)	(0.380)
Dec-01 to Mar-02	2.548	2.568	(0.540)	(0.520)	2.855	2.875	(0.233)	(0.213)	2.743	2.763	(0.345)	(0.325)
Apr-02 to Oct-02	2.471	2.491	(0.695)	(0.675)	3.009	3.029	(0.158)	(0.138)	2.801	2.821	(0.365)	(0.345)
Nov-02 to Mar-03	3.311	3.331	(0.340)	(0.320)	3.499	3.519	(0.153)	(0.133)	3.421	3.441	(0.230)	(0.210)
One Year Strip*	2.551	2.571	(0.614)	(0.594)	2.982	3.002	(0.182)	(0.162)	2.817	2.837	(0.347)	(0.327)

	AECO / NIT				IF NWPL Canadian Border (Sumas)				IF PEPL TX-OK			
	Fixed Price BID	Fixed Price OFFER	Basis BID	Basis OFFER	Fixed Price BID	Fixed Price OFFER	Basis BID	Basis OFFER	Fixed Price BID	Fixed Price OFFER	Basis BID	Basis OFFER
Cash	2.376	2.396			2.480	2.500			2.530	2.550		
ROM	2.398	2.418			2.460	2.480			2.530	2.550		
Dec-01	2.552	2.572	(0.408)	(0.388)	2.800	2.820	(0.160)	(0.140)	2.828	2.848	(0.133)	(0.113)
Dec-01 to Mar-02	2.616	2.636	(0.472)	(0.452)	2.892	2.912	(0.196)	(0.176)	2.958	2.978	(0.130)	(0.110)
Apr-02 to Oct-02	2.661	2.681	(0.505)	(0.485)	2.796	2.816	(0.370)	(0.350)	3.046	3.066	(0.120)	(0.100)
Nov-02 to Mar-03	3.216	3.236	(0.435)	(0.415)	3.706	3.726	0.055	0.075	3.531	3.551	(0.120)	(0.100)
One Year Strip*	2.676	2.696	(0.488)	(0.468)	2.880	2.900	(0.285)	(0.265)	3.041	3.061	(0.123)	(0.103)

	NGI Socal (South Cal Border)				NGI Malin (North Cal Border)				PG&E City Gate			
	Fixed Price BID	Fixed Price OFFER	Basis BID	Basis OFFER	Fixed Price BID	Fixed Price OFFER	Basis BID	Basis OFFER	Fixed Price BID	Fixed Price OFFER	Basis BID	Basis OFFER
Cash	2.580	2.600			2.550	2.570			2.570	2.590		
ROM	2.500	2.520			2.480	2.500			2.520	2.540		

After

```
    company_name           Fixed_Price_Bid Fixed_Price_Offer   Basis_Bid   Basis_Offer
0   IF NWPL Rocky Mountains        1.89            1.91          None        None
1   IF NWPL Rocky Mountains        2.06            2.08          None        None
2   IF NWPL Rocky Mountains       2.395           2.415        -0.565      -0.545
3   IF NWPL Rocky Mountains       2.594           2.614       -0.49375    -0.47375
4   IF NWPL Rocky Mountains     2.58129         2.60129        -0.585      -0.565
5   IF NWPL Rocky Mountains       3.356           3.376        -0.295      -0.275
6   IF NWPL Rocky Mountains     2.63408         2.65408     -0.530417   -0.510417
11  IF CIG Rocky Mountains      2.47129         2.49129        -0.695      -0.675
12  IF CIG Rocky Mountains        3.311           3.331         -0.34       -0.32
13  IF CIG Rocky Mountains      2.55075         2.57075     -0.61375    -0.59375
14  IF EL Paso Permian            2.375           2.395          None        None
15  IF EL Paso Permian             2.42            2.44          None        None
16  IF EL Paso Permian              2.7            2.72         -0.26       -0.24
17  IF EL Paso Permian          2.85463         2.87462    -0.233125   -0.213125
18  IF EL Paso Permian          3.00879         3.02879       -0.1575     -0.1375
19  IF EL Paso Permian           3.4985          3.5185       -0.1525     -0.1325
20  IF EL Paso Permian          2.98221         3.00221    -0.182292   -0.162292
...
```

Python is powerful!

Resources:

- Pandas Documentation
- Duncan Garmonsway's Spreadsheet Munging Strategies
- *Titanic.xlsx* data was from tidyxl repo
- *Enron.xlsx* data is from unpivotr repo
- My encounters with spreadsheets 😉

Code plus slides

https://github.com/samukweku/Extracting-Data-from-Excel-with-Python

Q & A

https://samukweku.github.io/data-wrangling-blog/

github.com/samukweku

@samukweku

linkedin.com/samuel-oranyeli