Kojo Idrissa @Transition

Python & Spreadsheets Canadian Edition Jan 2018

- Now: QA Specialist for Decisio Health
- Then: Accountant & college instructor w/ an MBA
- https://github.com/kojoidrissa/pycascades_2018

Outline

- How I got here
- Describe problem space
- Describe OpenPyXL-based solution
- Discuss problems using spreadsheets via code

Kojo Idrissa @Transition

My Secret Origin!!

- In Progress: http://bit.ly/kojo_secret_origin
- tl;dr: Adding new contributors

Code

- Skipping “standard Python”
- Interesting code provided by *your* specific use case

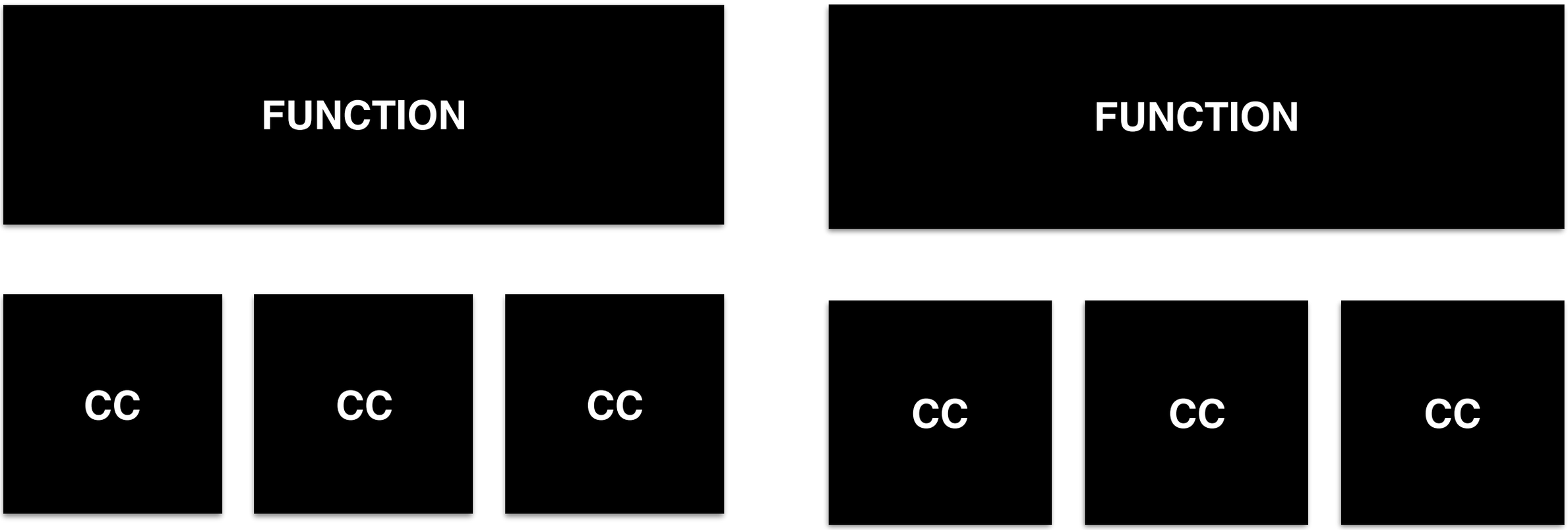
Step 0: Know Your Data!!

Let’s look at our spreadsheets

Input: Aggregate Time Utilization

	A	B	C	D	E	F	G
1	**division**	**cc**	**employe_num**	**employee_name**	**manager**	**doe**	**project**
2	1000	52P02	899591	Employee_01	dirsa	130	30
3	1000	52P10	811272	Employee_02	dirsa	93	67
4	1000	55335	810761	Employee_03	ffirs	38	122
5	1000	52P04	1098762	Employee_04	dirsa	156	4

Output: Sorted By Function



Output: Sorted By Function

- Accounting
  - 57100
  - 61101
- Engineering
  - 52P05
  - 55009

Output: Sorted By Cost Center

division	cc	employe_num	employee_name	manager	doe	project	Tot. Hours	DOE Util %	Proj. Util %
1000	52P05	1099700	Employee_21	dirsa	79	81	160	0.49375	0.50625
1000	52P05	1075893	Employee_22	dirsa	146	14	160	0.9125	0.0875
1000	52P05	899913	Employee_31	dirsa	130	30	160	0.8125	0.1875
division	cc	employe_num	employee_name	manager	doe	project	Tot. Hours	DOE Util %	Proj. Util %
1000	55099	801836	Employee_19	ffirs	49	111	160	0.30625	0.69375
1000	55099	801717	Employee_23	ffirs	158	2	160	0.9875	0.0125
1000	55099	810425	Employee_26	ffirs	84	76	160	0.525	0.475
1000	55099	806974	Employee_29	ffirs	34	126	160	0.2125	0.7875
1000	55099	807275	Employee_42	ffirs	90	70	160	0.5625	0.4375

- Original Method
  - manually filter/copy/paste
  - 500-600 employees
  - 10-15 functions
  - ~ 80 cost centers

Kojo Idrissa @Transition

JSON To The Rescue!

```
{
    "Project Comptroller":
        ["55221","52A04", "52A09", "52A10"],
    "Project Management":
        ["52A02", "52A12", "52P01", "52P02", "52P04", "52P06", "52P07
            ",
        "52P08", "52P09", "52P10", "52P11", "52P14","52P17", "52P18",
            "52P19", "52P21", "52P22", "52P23"],
    "Quality":
        ["52A03", "52A11", "51346", "55054", "57165"],
    "Accounting":
        ["57100", "61101"],
    "HR":
        ["51247", "51270", "55308", "55358", "52A01"],
    "Purchasing":
        ["55236", "52A05"],
    "Engineering":
        ["52P05", "55099"],
    "Gotham City":
        ["55903", "55147", "55335", "55292"]
}
```

Kojo Idrissa @Transition

Code

```
from openpyxl import load_workbook
from openpyxl.workbook import Workbook

wb = load_workbook('headcount_summary.xlsx')
source = wb.get_sheet_by_name('monthly_headcount_summary')
```

OpenPyXL Data Types

- Workbook
- Worksheet
- Cell

Create a header

```
#creating the header row
header =[]
# In Py3, source.rows returns a generator object. I had to use the
next() function on it.
for cell in next(source.rows):
    header.append(cell.value)
header.extend(['Tot. Hours', 'DOE Util %', 'Proj. Util %'])
```

Creating Functional Groups

```
with open('costCenter_Function_map.json', 'r') as json_map:
    dept_dict = json.load(json_map)

sheet_dict = {}
for key in dept_dict.keys():
    sheet_dict.update({key : functionTable(dept_dict[key])})
```

functionTable: docstring

```
def functionTable(list):
    '''
    list ==> list of lists

    Input a list of cost centers which make up a Functional Tab;
    return a list of lists where each inner list is a row of
    representing one person in that Cost Center
    '''
```

functionTable: content

```
tempTable = []
for row in source.rows:
    # row[1] is the position of the Cost Center
    if str(row[1].value) in list:
        temprow = []
        for cell in row:
            temprow.append(cell.value)
```

functionTable: content

```
        #Total Hours: sum of DOE & Proj Hours
        temprow.append(temprow[-1]+temprow[-2])
        #DOE Util%; DOE Hours / newly added Total
        temprow.append(temprow[-3]/float(temprow[-1]))
        #Proj Util%; Proj. Hours / Total
        temprow.append(temprow[-3]/float(temprow[-2]))
        tempTable.append(temprow)
return tempTable
```

Phase 1 complete

```
with open('costCenter_Function_map.json', 'r') as json_map:
    dept_dict = json.load(json_map)

sheet_dict = {}
for key in dept_dict.keys():
    sheet_dict.update({key : functionTable(dept_dict[key])})
```

Phase 2.0
Create a workbook

```
##Creating the target workbook
target = Workbook()
dest_filename = r'headcount_by_function.xlsx'
```

Phase 2.1
create_tabs

```
for key in sorted(sheet_dict.keys(), reverse = True):
    create_tabs(sheet_dict[key], key)
```

Phase 2.1
create_tabs

```
def create_tabs(functable, tabname):
    '''
    list of lists, string --> list of lists

    Takes in nested list for each functional area (created by
    functionTable) and a string (tabname)
    each inner list represents a row of data; creates a worksheet in
    memory, writes those rows to the worksheet
    The string becomes the name of the worksheet
    '''
```

Phase 2.1
create_tabs

```
ws = target.create_sheet(tabname, 0)
```

```
spacer = [None for i in range(len(functable[0]))]
```

The Phantom Step

- I’m skipping the creation of ‘*headcount_sorted*’, which we see in the next slide.

- It’s *functable* with a semi-fancy sort
- It’s not that interesting

Visually Formatted Output

```
for r in headcount_sorted:
    ri = headcount_sorted.index(r)
    #If this is the FIRST row, append the header, then the row
    if ri == 0:
        ws.append(header)
        ws.append(r)
    #If the Cost Center in THIS row is the same as the one
    before it, append the row
    elif headcount_sorted[ri][1] == headcount_sorted[(ri-1)][1]:
        ws.append(r)
    #If this Cost Center is different than the prior row, it's a
    new Cost Center
    else:
        ws.append(spacer) #spacer for readability
        ws.append(header)
        ws.append(r)
```

Kojo Idrissa @Transition

The Most Important Line Of Code

```
target.save(dest_filename)
```

Why You'll Hate Spreadsheets

- Spreadsheets as visual medium
- Unstructured input data
- “Visual" output requirements

Make New Friends

- Help co-workers automate, make a new Pythonista
- Will they “standardize” for a 90+% time reduction?
- Use cell style attributes to read and create ‘visual’ spreadsheets

Questions/Comments?

- @Transition on Twitter
- kojoidrissa.com
- https://github.com/kojoidrissa/pycascades_2018