Kojo Idrissa @Transition

Python & Spreadsheets: Earth Dog Edition Aug 2018

- DjangoCon US Orientation, ⚡Talks & Sprints Chair
- Jr. Developer for Decisio Health
- DEFNA North American Ambassador
- Formerly: Accountant & college instructor w/ an MBA

Outline

- Describe the problem
- Summarize OpenPyXL-based solution
- Opportunities and Problems

Mostly Simple Code

- Skimming over “standard Python”
- Focus on OpenPyXL specifics
- Interesting code provided by *your* specific use case

Input: Time Data

	A	B	C	D	E	F	G
1	employee_num	cost_center	division	Manager	date_worked	Employee Name	hours_worked
2	899591	52P02	1000	dirsa	6/12/2017	wrshwl	1.00
3	899591	52P02	1000	dirsa	6/13/2017	wrshwl	3.00
4	841596	57100	2000	kavi	6/3/2017	pplmbw	3.00
5	841596	57100	2000	kavi	6/4/2017	pplmbw	4.00
6	841596	57100	2000	kavi	6/5/2017	pplmbw	2.00
7	841596	57100	2000	kavi	6/6/2017	pplmbw	3.00
8	841596	57100	2000	kavi	6/10/2017	pplmbw	4.00
9	841596	57100	2000	kavi	6/11/2017	pplmbw	6.00
10	841596	57100	2000	kavi	6/12/2017	pplmbw	5.00
11	841596	57100	2000	kavi	6/13/2017	pplmbw	3.00
12	841596	57100	2000	kavi	6/14/2017	pplmbw	4.00

Names & OpenPyXL types

- Workbook
- Worksheet
- Cell

```
import openpyxl

demo_workbook = openpyxl.load_workbook(
    'demo_workbook.xlsx',
    data_only= True
)

demo_worksheet = demo_workbook.get_sheet_by_name(
    "clean_data"
)
```

```
employee_ids = set()
for row in demo_worksheet.rows:
    if row[0].value != 'employee_num':
        employee_ids.add(row[0].value)
```

```
employee_aggregate = {}
for employee in employee_ids:
    hours = [
        row[6].value
        for row in demo_worksheet.rows
        if employee == row[0].value
    ]
```

```
employee_aggregate[employee]={
    "hours": sum(hours),
    "cost_center": list(cost_center)[0],
    "division": list(division)[0],
    "manager": list(manager)[0]
}
```

```
header = [
    demo_worksheet["A1"].value,
    demo_worksheet["B1"].value,
    demo_worksheet["C1"].value,
    demo_worksheet["D1"].value,
    demo_worksheet["G1"].value
]
```

```
output_book = openpyxl.Workbook()
output_sheet = output_book.create_sheet(
    "Aggregate Time",
    0
)
```

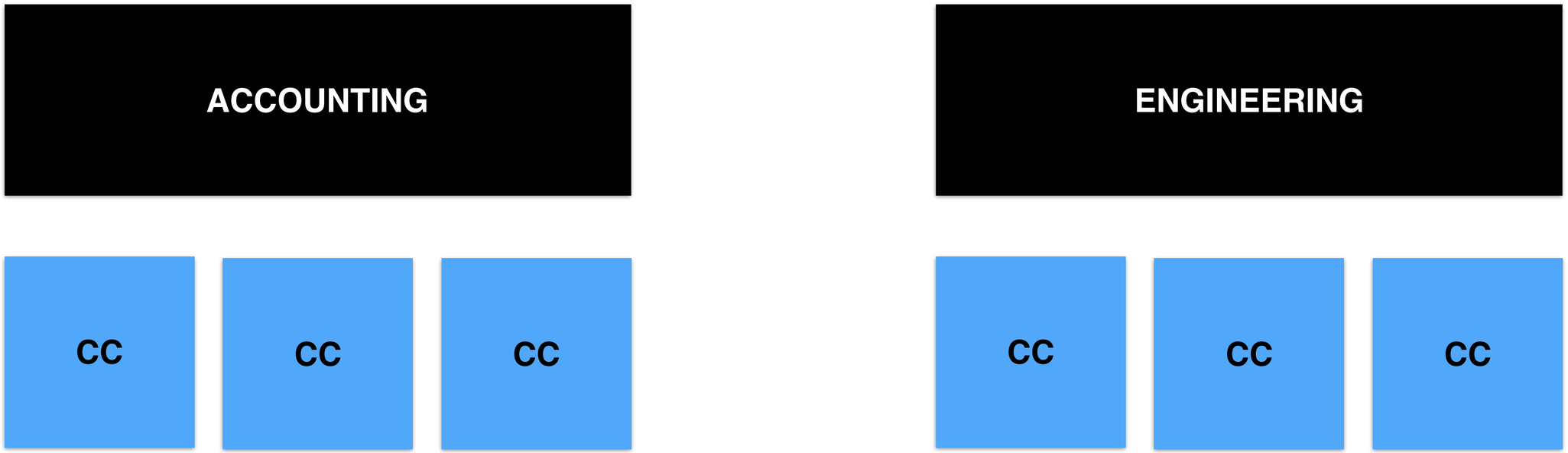
```
output_data = []
output_data.append(header)
```

```
for employee in employee_aggregate:
    new_row = []
    new_row.append(employee)
    new_row.append(employee_aggregate[employee]['cost_center'])
    new_row.append(employee_aggregate[employee]['division'])
    new_row.append(employee_aggregate[employee]['manager'])
    new_row.append(employee_aggregate[employee]['hours'])
    output_data.append(new_row)
```

```
# Write data to sheet object
for row in output_data:
    rowIn = output_data.index(row)
    for col in range(len(output_data[0])):
        # Adding 1 because spreadsheets count from 1, not 0
        output_sheet.cell(row = rowIn+1, column = col+1).value =
            output_data[rowIn][col]
```

```
output_book.save(
    filename = "done_pyconau.xlsx"
)
```

Extra Work: Multi-Level Sort



Original Method

- manual 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"]
}
```

ACCOUNTING

1

2

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

General Use Cases

- Single-Ended Input
- Single-Ended Output
- Double-Ended

What I’m NOT Showing You

- Styles
- Charts
- Good News/Bad News

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/pyconau_2018