Dr Nisha Arora

PyData Global

Python Meets Excel

Smarter Workflows for
Analysts and Data Teams

- ❑ Educator by heart & trainer by profession
- ❑ Believes learning should feel like discovery, not duty
- ❑ Works in the area of Data Science, AI & More

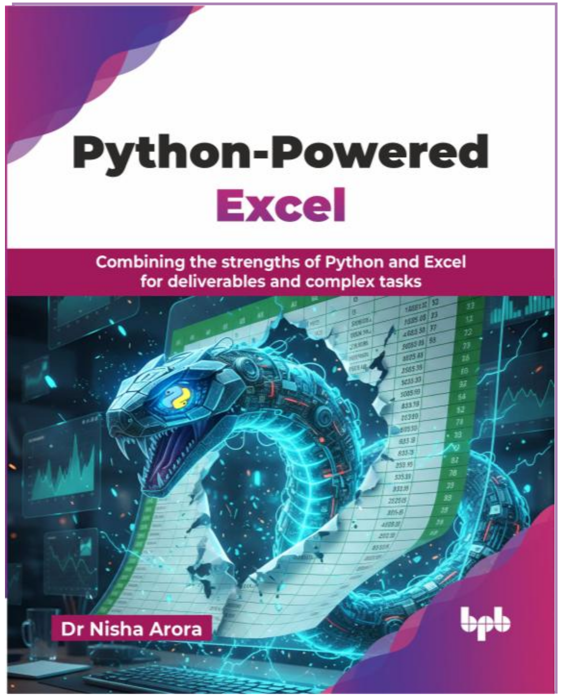
Dr Nisha Arora

Trainer | Author | Reviewer

Background

- ❑ MPhil, PhD in Mathematics.
- ❑ Taught across MBA classrooms, corporate boardrooms, and tech communities
- ❑ Mentor/Panelist/Speaker for Women in Tech Global & Women in Data Science, Silicon Valley

My Upcoming book!



Python-Powered Excel

Combining the strengths of Python and Excel for deliverables and complex tasks

Dr Nisha Arora

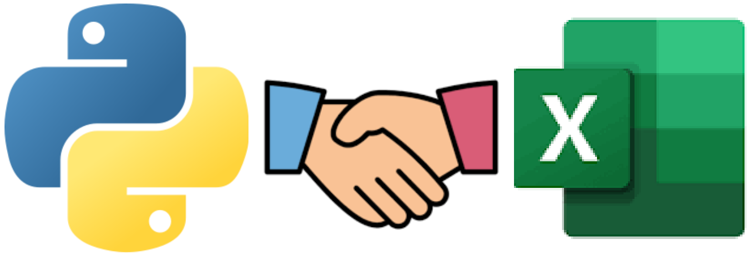
bpb

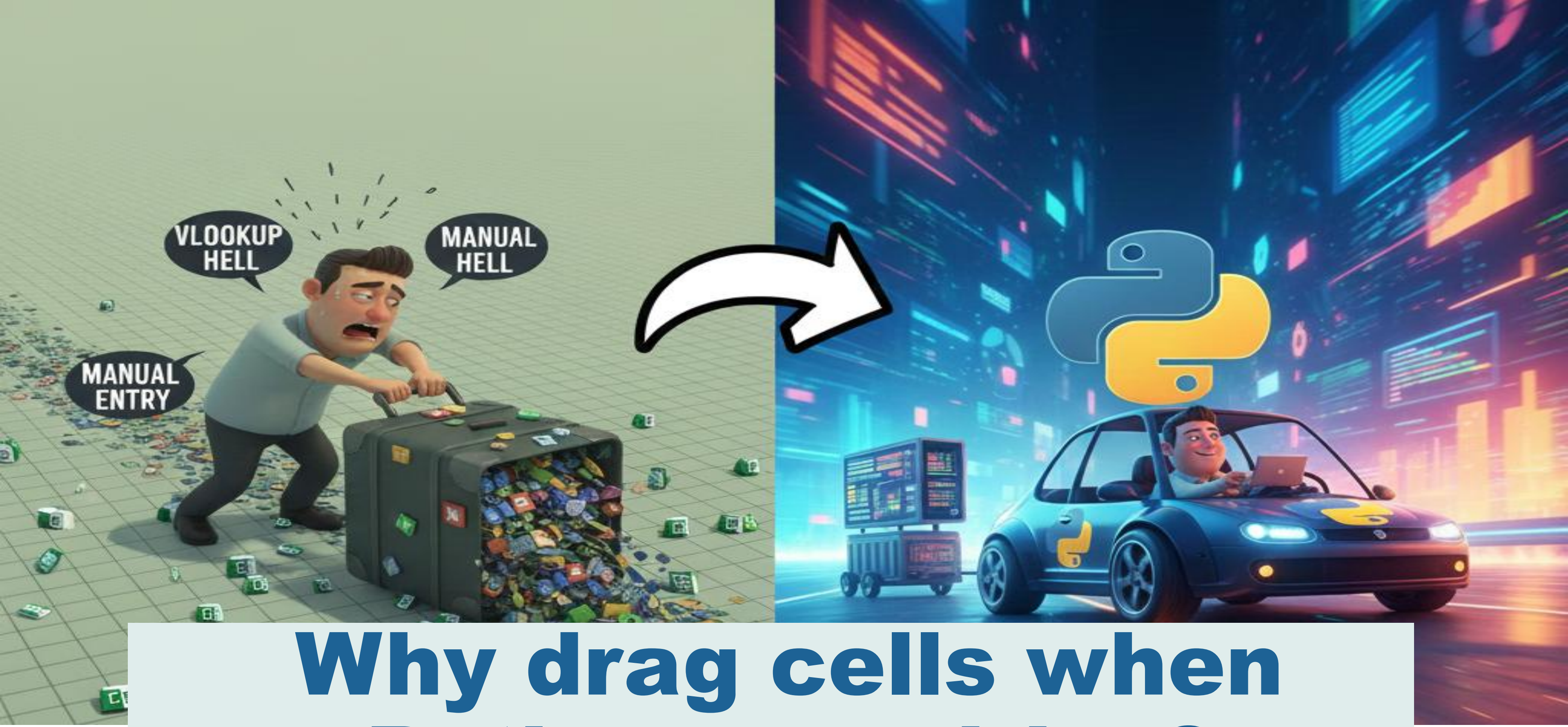
Community Contribution

- ❑ Content reached more than 1.7 million learners
- ❑ Conducted 50+ webinars and mentoring sessions worldwide
- ❑ Created courses on Udemy and Tutorials Point

Agenda

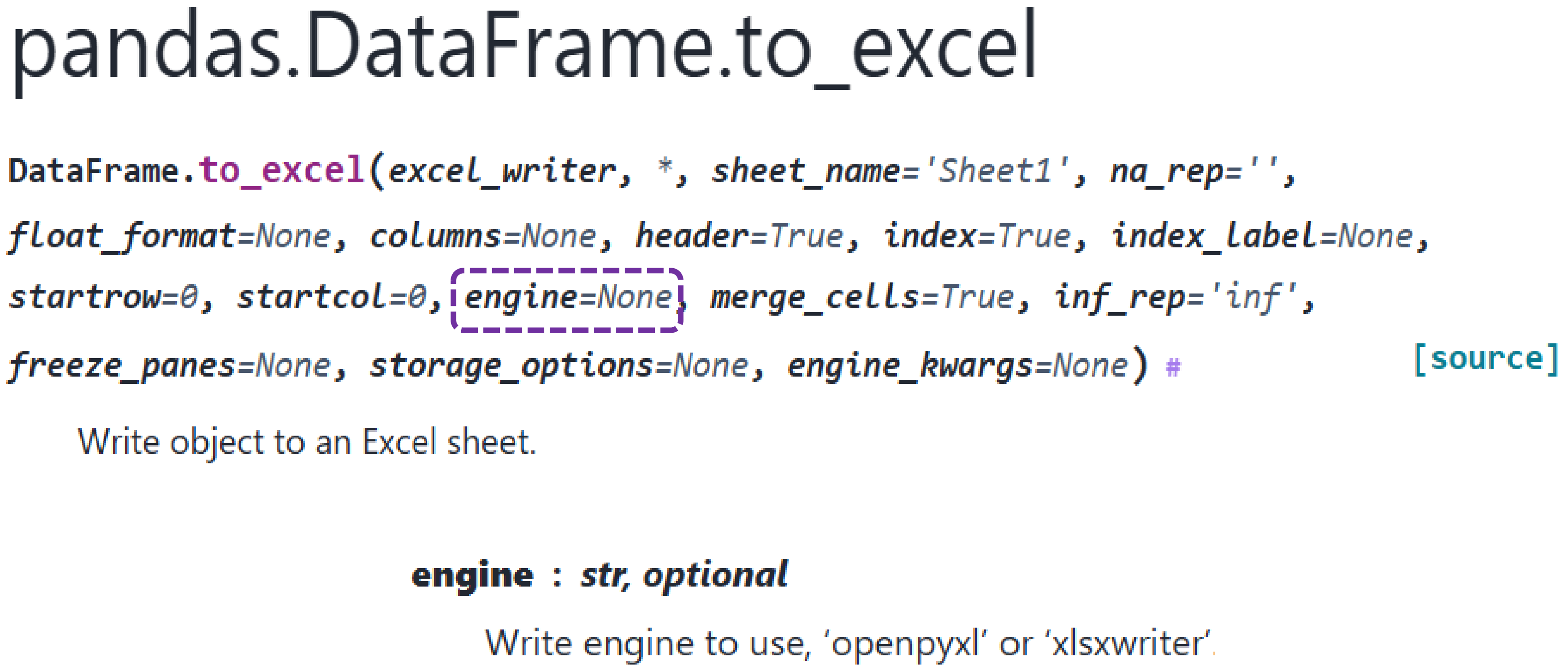
1. Do Python users need Excel?
2. Python meets Excel
3. Python Tools for Excel
4. Excel as UI and Python as the Engine
5. Let’s see it in action!





Why drag cells when Python can drive?

- Excel is everywhere
- Stockholders want deliverables in Excel
- Python solves Excel’s limitations
- Not everyone is a programmer
- Python with Excel creates a complete workflow



pandas.DataFrame.to_excel

```
DataFrame.to_excel(excel_writer, *, sheet_name='Sheet1', na_rep='',
float_format=None, columns=None, header=True, index=True, index_label=None,
startrow=0, startcol=0, engine=None, merge_cells=True, inf_rep='inf',
freeze_panes=None, storage_options=None, engine_kwargs=None) #          [source]
```

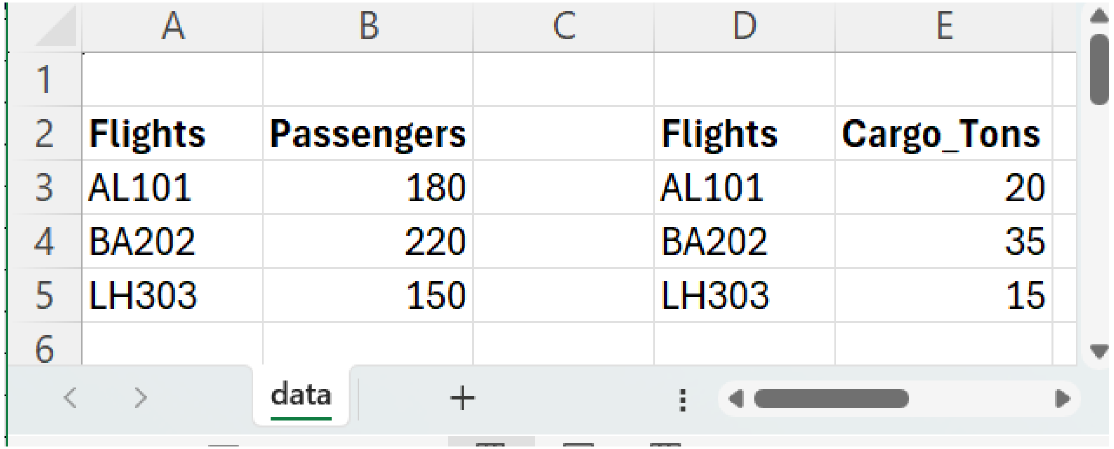
Write object to an Excel sheet.

engine : *str, optional*

Write engine to use, 'openpyxl' or 'xlsxwriter'.

The ExcelWriter Class

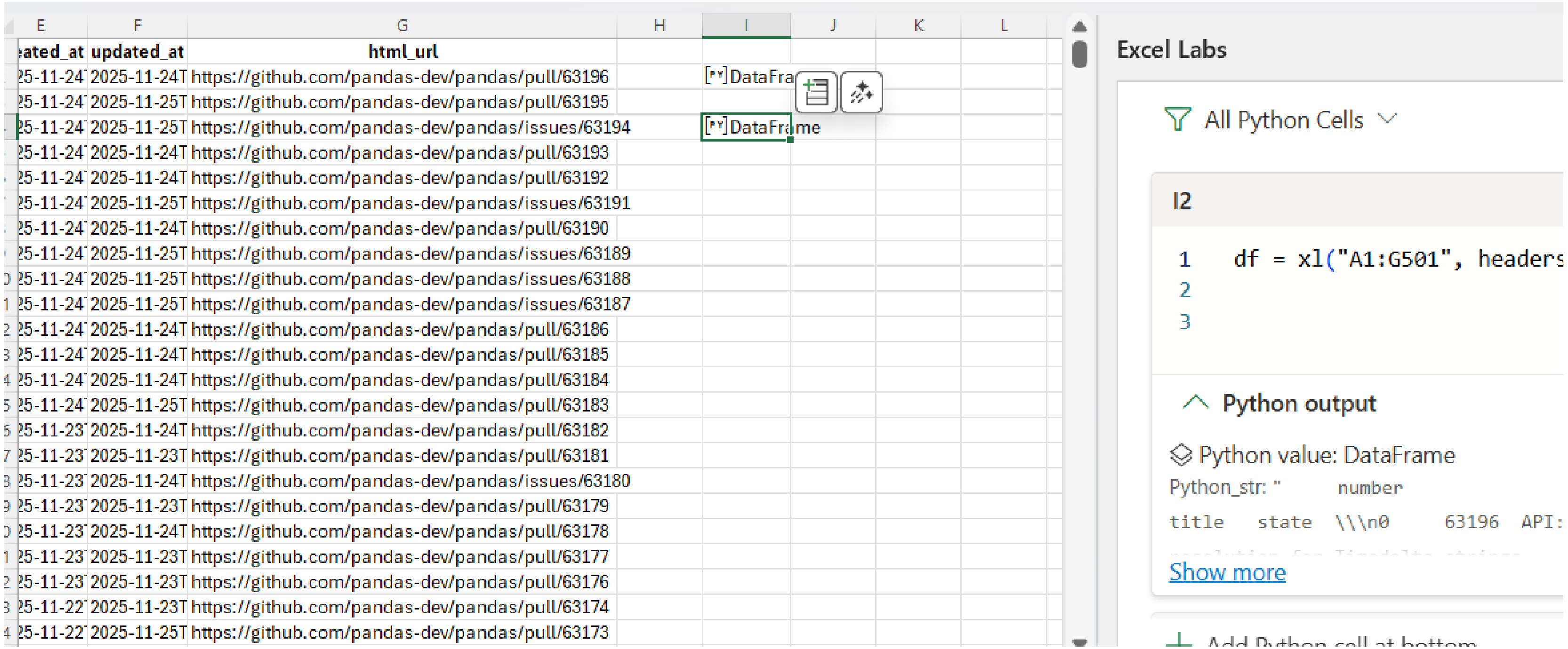
```
with pd.ExcelWriter("sample.xlsx", engine="openpyxl") as writer:
    flights.to_excel(writer, sheet_name="data", startrow=1, startcol=0, index=False)
    cargo.to_excel(writer, sheet_name="data", startrow=1, startcol=4, index=False)
```



	A	B	C	D	E
1					
2	**Flights**	**Passengers**		**Flights**	**Cargo_Tons**
3	AL101	180		AL101	20
4	BA202	220		BA202	35
5	LH303	150		LH303	15
6					

Python meets Excel

Python in Excel (Excel 365)



E	F	G	H	I	J	K	L
ated_at	updated_at	html_url					
25-11-24	2025-11-24T	https://github.com/pandas-dev/pandas/pull/63196		DataFra			
25-11-24	2025-11-25T	https://github.com/pandas-dev/pandas/pull/63195					
25-11-24	2025-11-25T	https://github.com/pandas-dev/pandas/issues/63194		DataFrame			
25-11-24	2025-11-24T	https://github.com/pandas-dev/pandas/pull/63193					
25-11-24	2025-11-24T	https://github.com/pandas-dev/pandas/pull/63192					
25-11-24	2025-11-25T	https://github.com/pandas-dev/pandas/issues/63191					
25-11-24	2025-11-24T	https://github.com/pandas-dev/pandas/pull/63190					
25-11-24	2025-11-25T	https://github.com/pandas-dev/pandas/issues/63189					
25-11-24	2025-11-25T	https://github.com/pandas-dev/pandas/issues/63188					
25-11-24	2025-11-25T	https://github.com/pandas-dev/pandas/issues/63187					
25-11-24	2025-11-24T	https://github.com/pandas-dev/pandas/pull/63186					
25-11-24	2025-11-24T	https://github.com/pandas-dev/pandas/pull/63185					
25-11-24	2025-11-24T	https://github.com/pandas-dev/pandas/pull/63184					
25-11-24	2025-11-25T	https://github.com/pandas-dev/pandas/pull/63183					
25-11-23	2025-11-24T	https://github.com/pandas-dev/pandas/pull/63182					
25-11-23	2025-11-23T	https://github.com/pandas-dev/pandas/pull/63181					
25-11-23	2025-11-24T	https://github.com/pandas-dev/pandas/issues/63180					
25-11-23	2025-11-23T	https://github.com/pandas-dev/pandas/pull/63179					
25-11-23	2025-11-24T	https://github.com/pandas-dev/pandas/pull/63178					
25-11-23	2025-11-23T	https://github.com/pandas-dev/pandas/pull/63177					
25-11-23	2025-11-23T	https://github.com/pandas-dev/pandas/pull/63176					
25-11-22	2025-11-23T	https://github.com/pandas-dev/pandas/pull/63174					
25-11-22	2025-11-25T	https://github.com/pandas-dev/pandas/pull/63173					

Excel Labs

All Python Cells

I2

```
df = xl("A1:G501", headers
```

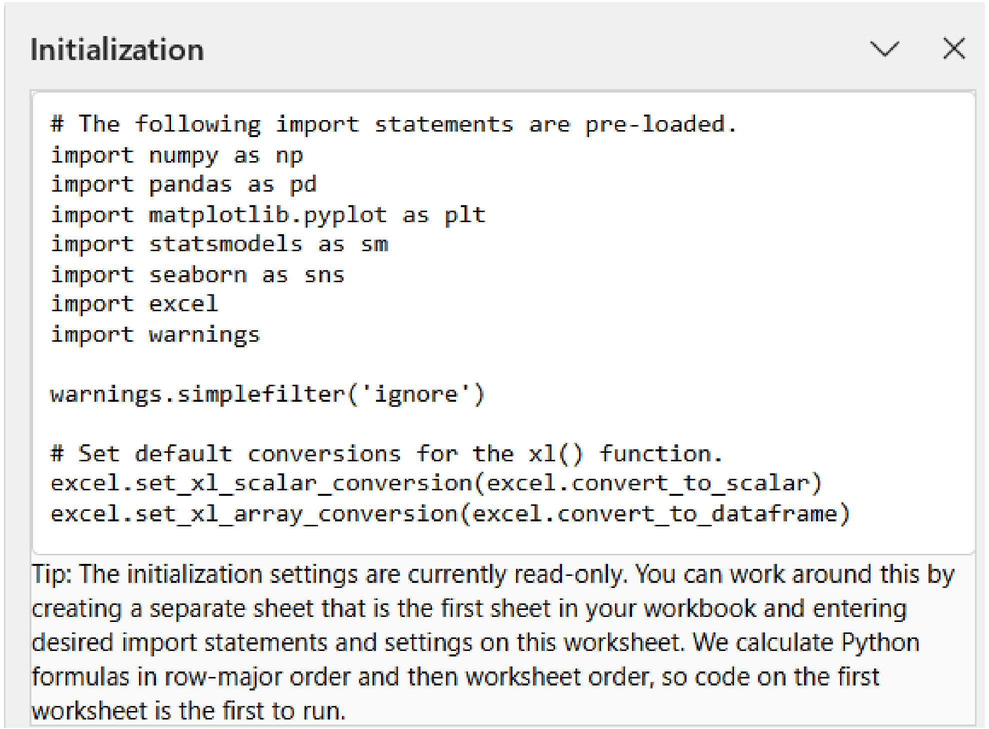
Python output

Python value: DataFrame

```
Python_str: '      number
title    state  \\\n0      63196   API:
```

Show more

Core Python in Excel libraries



```
# The following import statements are pre-loaded.
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import statsmodels as sm
import seaborn as sns
import excel
import warnings

warnings.simplefilter('ignore')

# Set default conversions for the xl() function.
excel.set_xl_scalar_conversion(excel.convert_to_scalar)
excel.set_xl_array_conversion(excel.convert_to_dataframe)
```

Tip: The initialization settings are currently read-only. You can work around this by creating a separate sheet that is the first sheet in your workbook and entering desired import statements and settings on this worksheet. We calculate Python formulas in row-major order and then worksheet order, so code on the first worksheet is the first to run.

Python Tools for Excel

Image: The Python Strore

Open-source Python libraries

Openpyxl xlsxwriter xlwings

For legacy Excel(.xls) files

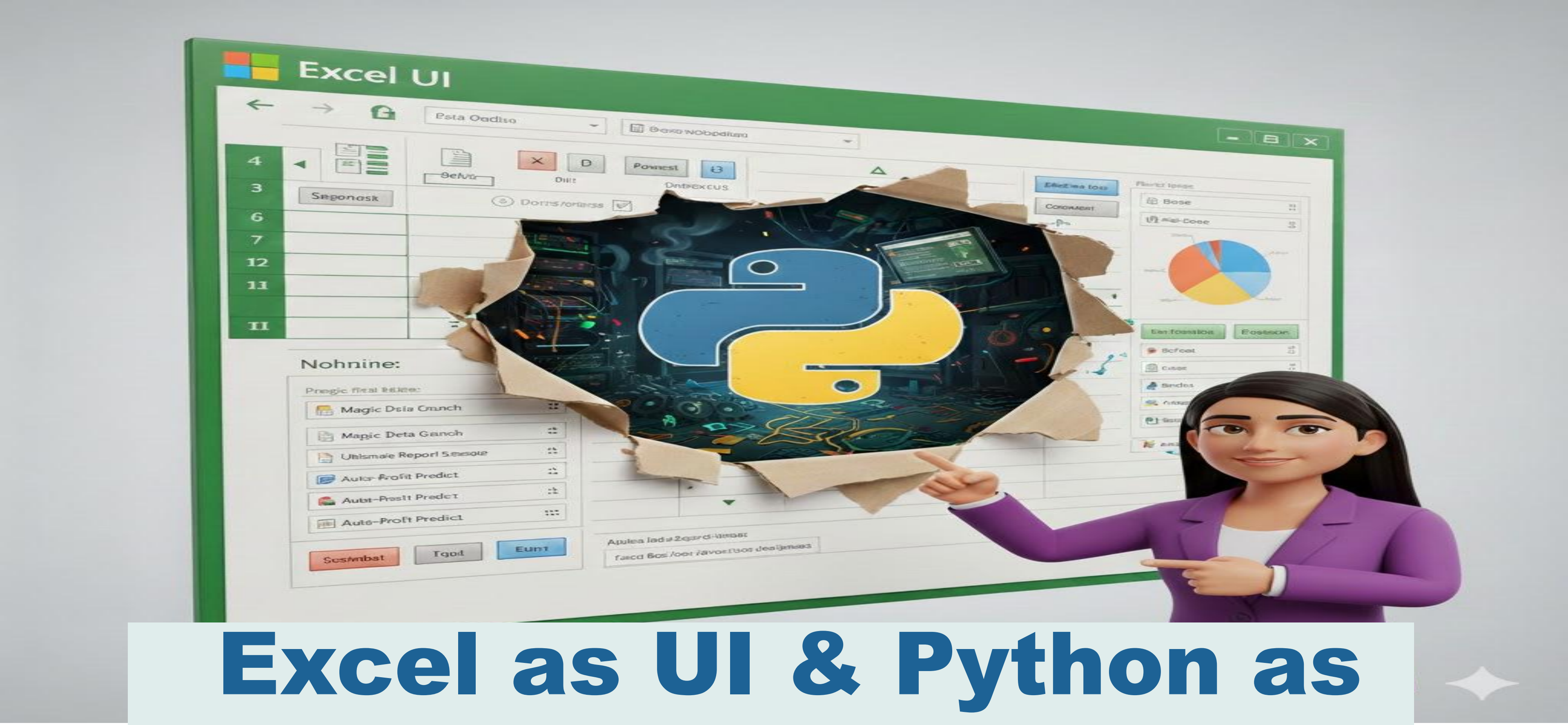
Xlrd xlwt xlutils

For binary Excel (.xlsb) files

pyxlsb

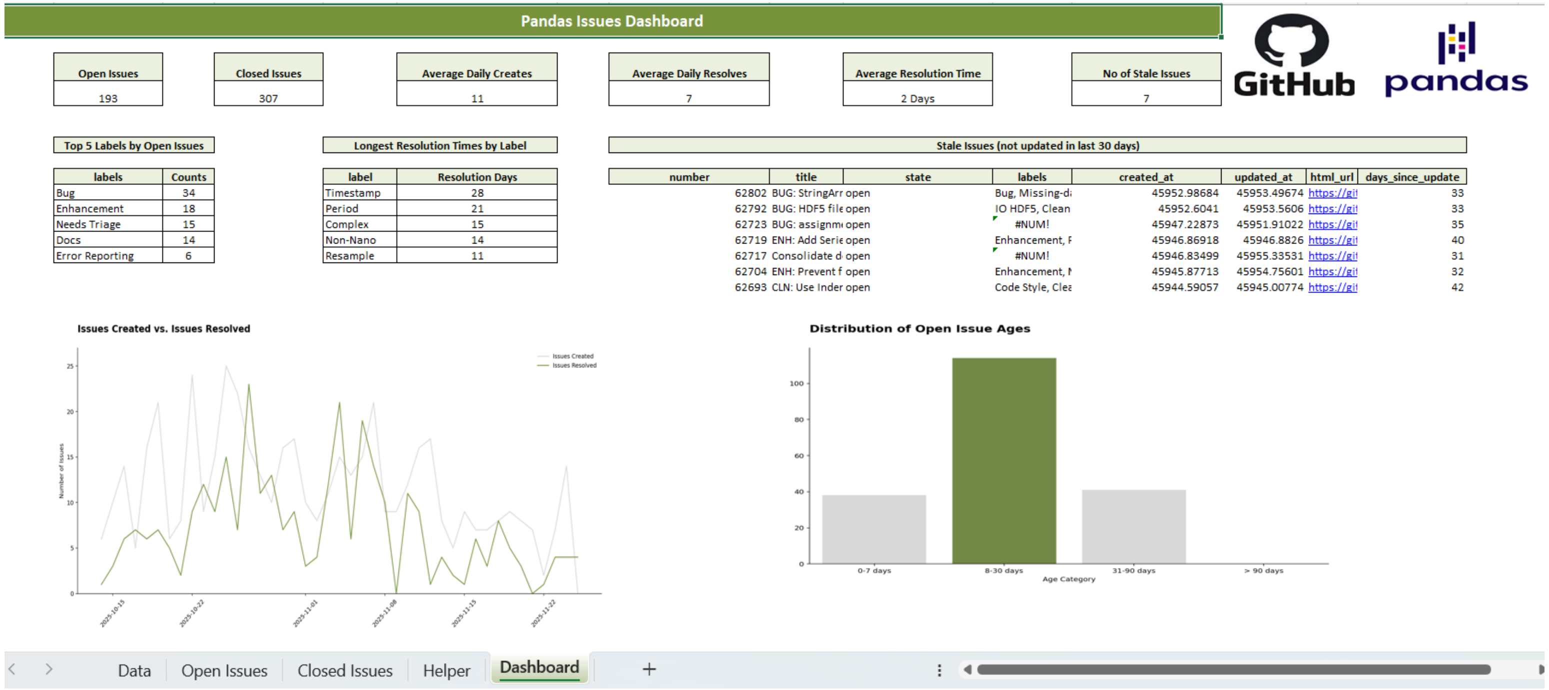
What can you do with xlwings?

- ❑ Read/write values, formulas, formats
- ❑ Interact with live Excel app (Hidden/Visible Mode, Multiple Instances)
- ❑ Pythonize workflows (not code-to-code, but you can replace VBA logic)
- ❑ Run Python functions from Excel (RunMain, Run Python, UDFs, buttons, ribbon)
- ❑ Run VBA macros from Python
- ❑ Control Excel like an automation engine



Excel as UI & Python as Engine

Report generated by Python



Pandas Issues Dashboard

GitHub pandas

Open Issues	Closed Issues	Average Daily Creates	Average Daily Resolves	Average Resolution Time	No of Stale Issues
193	307	11	7	2 Days	7

Top 5 Labels by Open Issues

labels	Counts
Bug	34
Enhancement	18
Needs Triage	15
Docs	14
Error Reporting	6

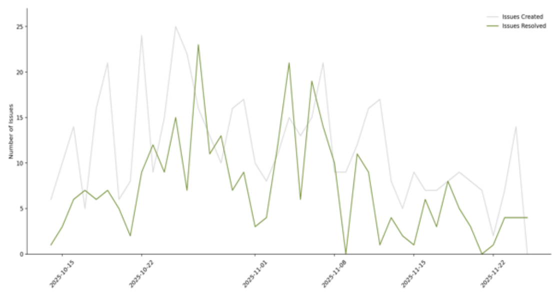
Longest Resolution Times by Label

label	Resolution Days
Timestamp	28
Period	21
Complex	15
Non-Nano	14
Resample	11

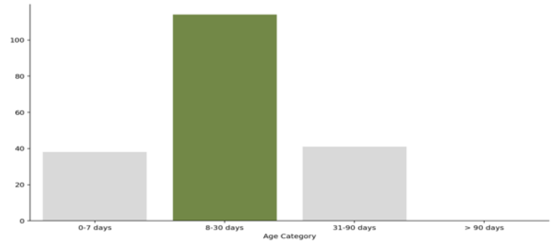
Stale Issues (not updated in last 30 days)

number	title	state	labels	created_at	updated_at	html_url	days_since_update
62802	BUG: StringArr	open	Bug, Missing-d	45952.98684	45953.49674	https://gi	33
62792	BUG: HDF5 file	open	IO HDF5, Clean	45952.6041	45953.5606	https://gi	33
62723	BUG: assignm	open	#NUM!	45947.22873	45951.91022	https://gi	35
62719	ENH: Add Serie	open	Enhancement, F	45946.86918	45946.8826	https://gi	40
62717	Consolidate d	open	#NUM!	45946.83499	45955.33531	https://gi	31
62704	ENH: Prevent f	open	Enhancement, N	45945.87713	45954.75601	https://gi	32
62693	CLN: Use Inder	open	Code Style, Clea	45944.59057	45945.00774	https://gi	42

Issues Created vs. Issues Resolved

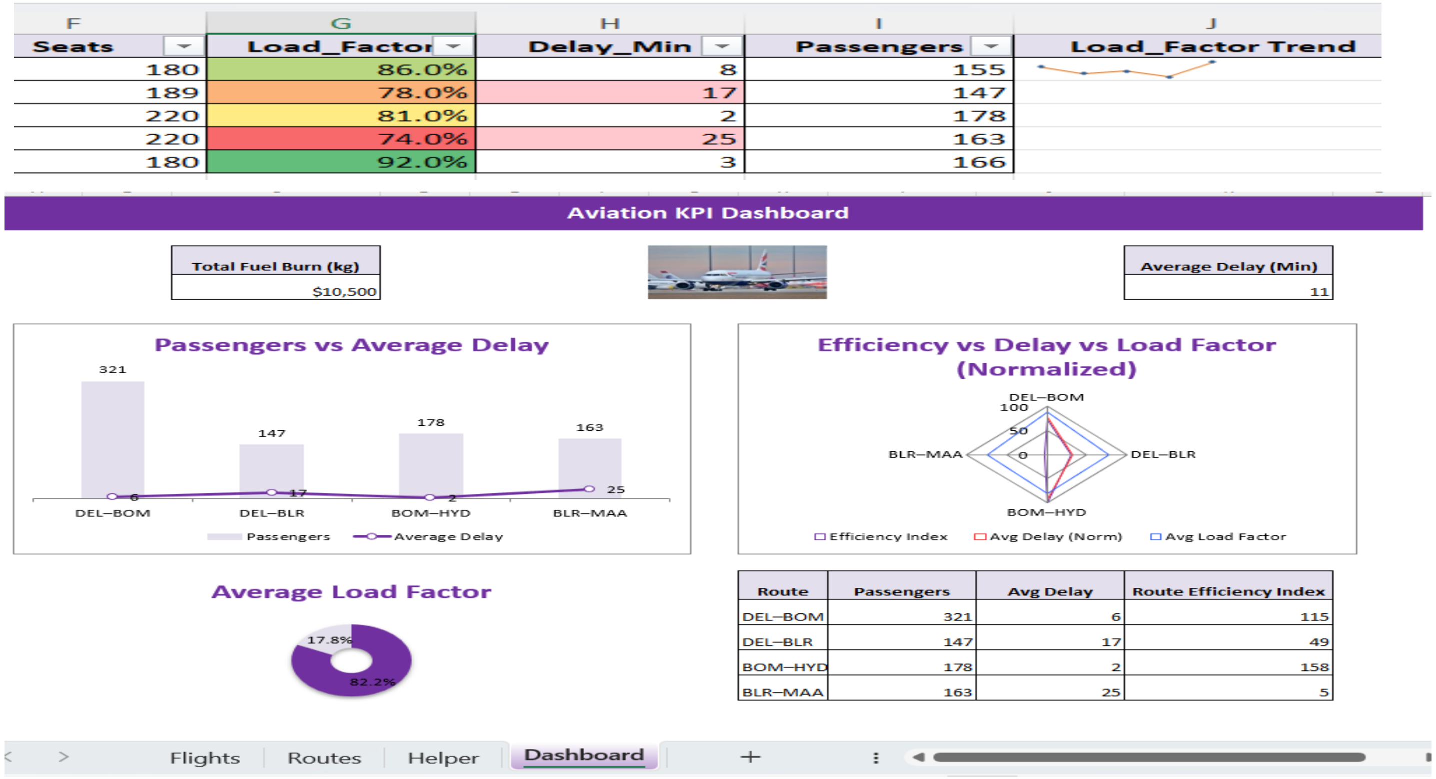


Distribution of Open Issue Ages



Data | Open Issues | Closed Issues | Helper | Dashboard | +

Report generated by Python

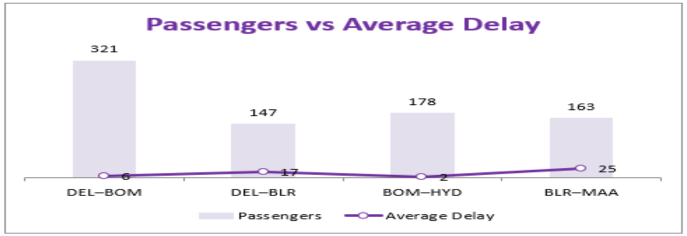


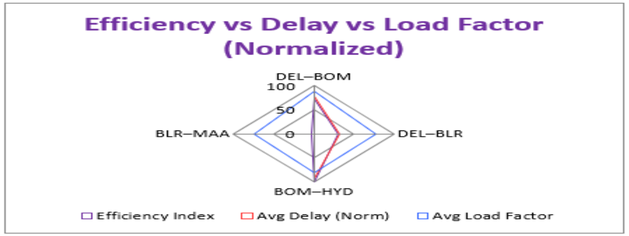
F	G	H	I	J
Seats	Load_Factor	Delay_Min	Passengers	Load_Factor Trend
180	86.0%	8	155	
189	78.0%	17	147	
220	81.0%	2	178	
220	74.0%	25	163	
180	92.0%	3	166	

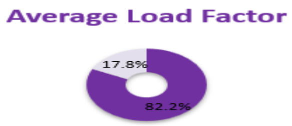
Aviation KPI Dashboard

Total Fuel Burn (kg)
$10,500

Average Delay (Min)
11







Route	Passengers	Avg Delay	Route Efficiency Index
DEL–BOM	321	6	115
DEL–BLR	147	17	49
BOM–HYD	178	2	158
BLR–MAA	163	25	5

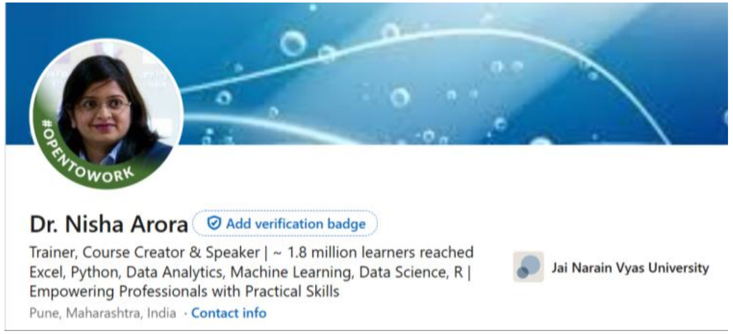
Flights | Routes | Helper | Dashboard | +

Let’s see that in action!

1	Using Python function in Excel (any version)
2	Clicking a button to perform analysis
3	Creating stunning Excel report in python

Connect with me!

My LinkTree

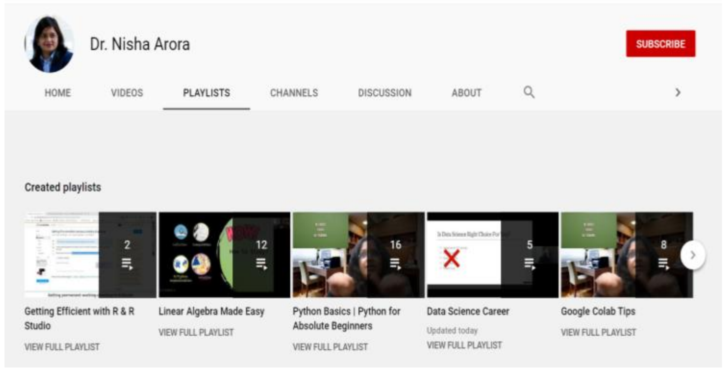


Dr. Nisha Arora

Trainer, Course Creator & Speaker | ~ 1.8 million learners reached Excel, Python, Data Analytics, Machine Learning, Data Science, R | Empowering Professionals with Practical Skills

Pune, Maharashtra, India · Contact info

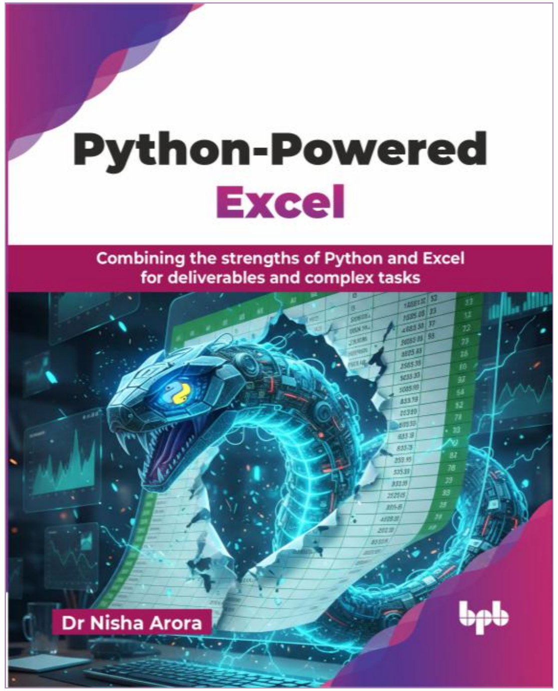
Jai Narain Vyas University



Dr. Nisha Arora

Created playlists

2	Getting Efficient with R & R Studio VIEW FULL PLAYLIST
12	Linear Algebra Made Easy VIEW FULL PLAYLIST
16	Python Basics | Python for Absolute Beginners VIEW FULL PLAYLIST
5	Data Science Career Updated today VIEW FULL PLAYLIST
8	Google Colab Tips VIEW FULL PLAYLIST

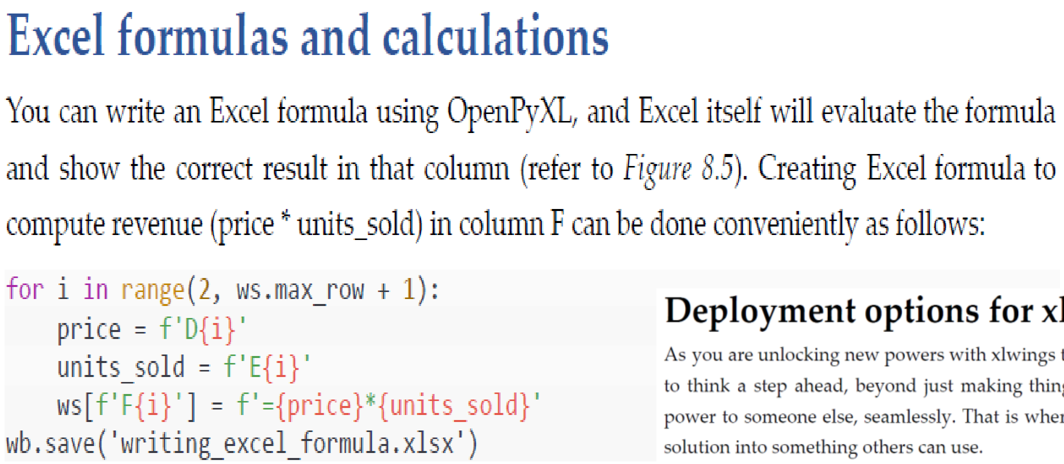

Python-Powered
Excel
Combining the strengths of Python and Excel for deliverables and complex tasks
Dr Nisha Arora
bpb

Thank you for your support as this book takes shape!

Early Interest Form

LinkedIn: drnishaarora
YouTube: @DrNishaArora

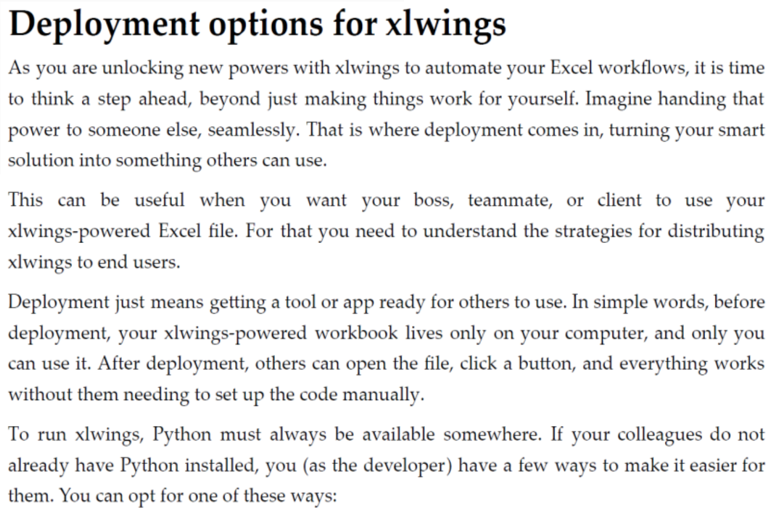
Sneak Peek: Python-Powered Excel



Excel formulas and calculations

You can write an Excel formula using OpenPyXL, and Excel itself will evaluate the formula and show the correct result in that column (refer to *Figure 8.5*). Creating Excel formula to compute revenue (price * units_sold) in column F can be done conveniently as follows:

```
for i in range(2, ws.max_row + 1):
    price = f'D{i}'
    units_sold = f'E{i}'
    ws[f'F{i}'] = f'={price}*{units_sold}'
wb.save('writing_excel_formula.xlsx')
```



Deployment options for xlwings

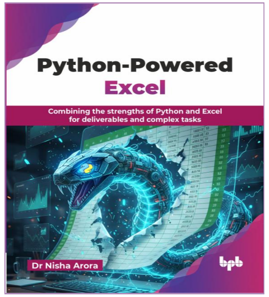
As you are unlocking new powers with xlwings to automate your Excel workflows, it is time to think a step ahead, beyond just making things work for yourself. Imagine handing that power to someone else, seamlessly. That is where deployment comes in, turning your smart solution into something others can use.

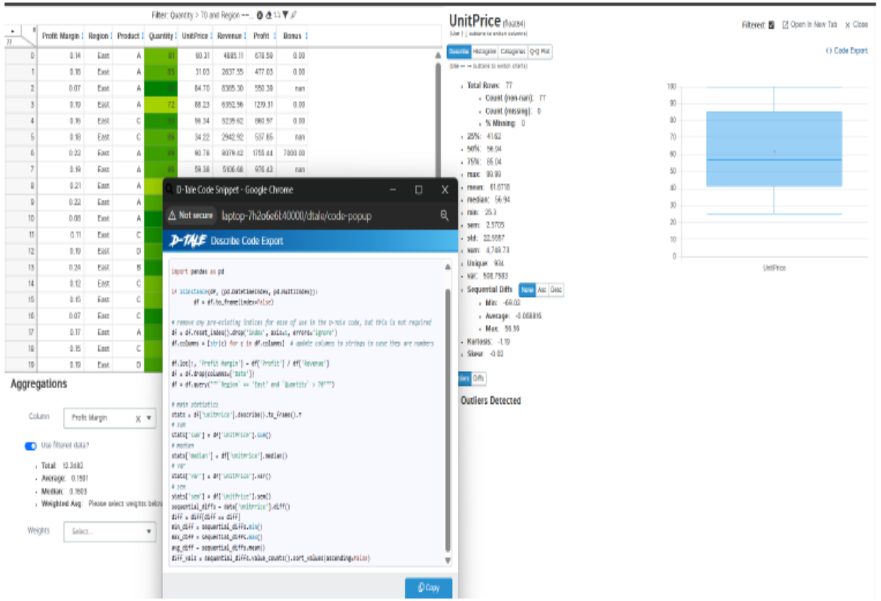
This can be useful when you want your boss, teammate, or client to use your xlwings-powered Excel file. For that you need to understand the strategies for distributing xlwings to end users.

Deployment just means getting a tool or app ready for others to use. In simple words, before deployment, your xlwings-powered workbook lives only on your computer, and only you can use it. After deployment, others can open the file, click a button, and everything works without them needing to set up the code manually.

To run xlwings, Python must always be available somewhere. If your colleagues do not already have Python installed, you (as the developer) have a few ways to make it easier for them. You can opt for one of these ways:

Sneak Peek: Python-Powered Excel


Python-Powered
Excel
Combining the strengths of Python and Excel for deliverables and complex tasks
Dr Nisha Arora
bpb



	Profit Margin	Region	Product	Quantity	UnitPrice	Revenue	Profit	Bonus
0	0.14	East	A	81	60.31	4885.11	678.59	0.00
1	0.18	East	A	85	31.03	2637.55	477.03	0.00
2	0.07	East	A	[illegible]	84.70	8385.30	550.39	nan
3	0.19	East	A	72	88.23	6352.56	1219.31	0.00
4	0.16	East	C	[illegible]	56.34	5239.62	860.97	0.00
5	0.18	East	C	86	34.22	2942.92	537.85	nan
6	0.22	East	A	89	90.78	8079.42	1755.44	7000.00
7	0.19	East	A	86	59.38	5106.68	976.43	nan
8	0.21	East	A					
9	0.22	East	A					
10	0.08	East	A					
11	0.11	East	C					
12	0.19	East	D					
13	0.24	East	B					
14	0.12	East	C					
15	0.15	East	C					
16	0.07	East	C					
17	0.17	East	A					
18	0.15	East	C					
19	0.19	East	D					

- Total: 13.3683
- Average: 0.1501
- Median: 0.1803
- Weighted Avg: Please select weights below

```
import pandas as pd

if isinstance(df, (pd.DatetimeIndex, pd.MultiIndex)):
        df = df.to_frame(index=False)

# remove any pre-existing indices for ease of use in the D-Tale code, but this is not required
df = df.reset_index().drop('index', axis=1, errors='ignore')
df.columns = [str(c) for c in df.columns]  # update columns to strings in case they are numbers

df.loc[:, 'Profit Margin'] = df['Profit'] / df['Revenue']
df = df.drop(columns=['Date'])
df = df.query("""`Region` == 'East' and `Quantity` > 70""")

# main statistics
stats = df['UnitPrice'].describe().to_frame().T
# sum
stats['sum'] = df['UnitPrice'].sum()
# median
stats['median'] = df['UnitPrice'].median()
# var
stats['var'] = df['UnitPrice'].var()
# sem
stats['sem'] = df['UnitPrice'].sem()
sequential_diffs = data['UnitPrice'].diff()
diff = diff[diff == diff]
min_diff = sequential_diffs.min()
max_diff = sequential_diffs.max()
avg_diff = sequential_diffs.mean()
diff_vals = sequential_diffs.value_counts().sort_values(ascending=False)
```

- Total Rows: 77
  - Count (non-nan): 77
  - Count (missing): 0
  - % Missing: 0
- 25%: 41.62
- 50%: 56.94
- 75%: 65.04
- max: 99.99
- mean: 61.6718
- median: 56.94
- min: 25.3
- sem: 2.5705
- std: 22.5557
- sum: 4,748.73
- Unique: 934
- var: 508.7583
- Sequential Diffs: Asc Desc
  - Min: -68.02
  - Average: -0.068816
  - Max: 55.55
- Kurtosis: -1.19
- Skew: -0.02

```
# Net Present Value (NPV)

# Initial investment = ₹1,00,000
# Returns over 5 years = ₹20k, ₹30k, ₹30k, ₹20k, ₹10k
cashflows = [-100000, 20000, 30000, 40000, 25000, 15000]
rate = 0.10

npv = npf.npv(rate, cashflows)

# Present Value (PV)

future_value = 100000
rate = 0.08
n_years = 3

pv = npf.pv(rate=rate, nper=n_years, pmt=0, fv=-future_value)

# Internal Rate of Return (IRR)

irr = npf.irr(cashflows)
```

Thank You!