Essential Skills
Python Workshop

By Joseph Wright (@Wright4i)

Table of contents

01

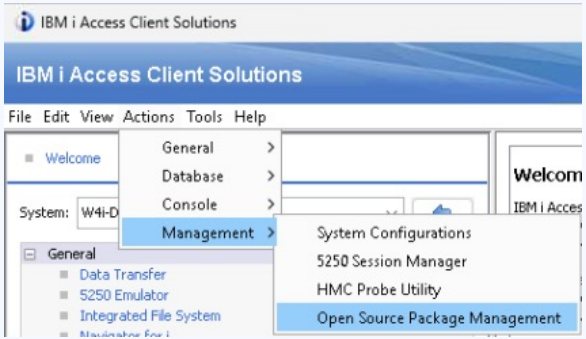
Getting Started

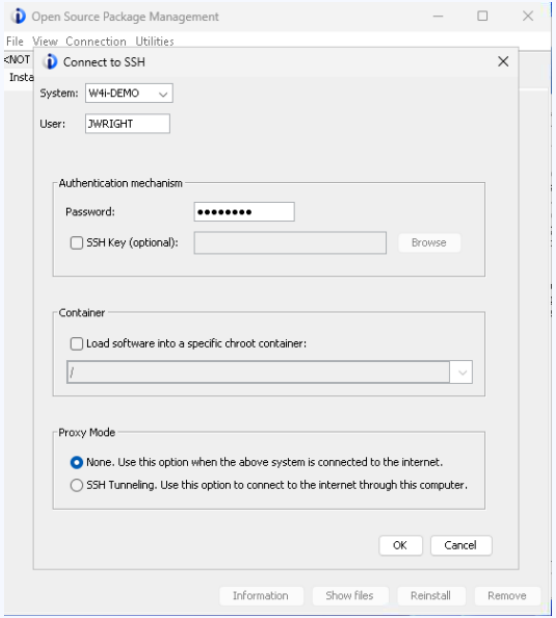
Using IBM Access Client Solutions to install Python &
Connecting to PASE with a terminal
Calling Python from CL & RPG

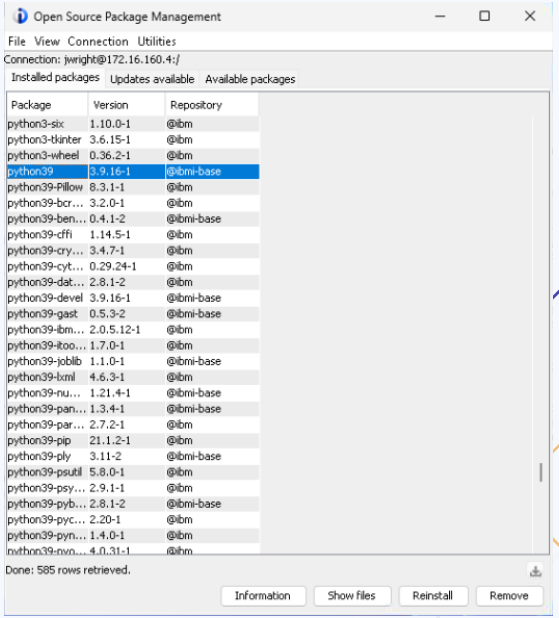
https://wright4i.com/guides/py/getting_started/

Installing Python

ACS > Open Source Package Manager







Connect to your system

Under “Install Packages” select **python39**

Calling Python from PASE

```
                                /QOpenSys/usr/bin/-sh

  $
> which python3
  /QOpenSys/pkgs/bin/python3
  $
> python3 --version
  Python 3.9.16
  $
> python3
  Python 3.9.16 (main, Feb  1 2023, 13:01:20)
  [GCC 6.3.0] on os400
  Type "help", "copyright", "credits" or "license" for more information.
  >>>
> print('Hello World')
  p
```

QP2TERM

Easy, Built-in, Buggy
Bourne Shell

BASH

```
SSH YOUR_USER@YOUR_SERVER
```

Linux/Mac: Built-in Terminal
Windows: Git Bash (git-scm.com)

```
$ ssh jwright@w4i.local
jwright@w4i.local's password:
-bash-5.1$ which python3
/QOpenSys/pkgs/bin/python3
-bash-5.1$ python3 --version
Python 3.9.16
-bash-5.1$ python3
Python 3.9.16 (main, Feb  1 2023, 13:01:20)
[GCC 6.3.0] on os400
Type "help", "copyright", "credits" or "license" f
or more information.
>>> print('Hello World')
Hello World
>>>
```

Calling Python from CL

```
PGM
  /* Call Python Script */
  QSH CMD('python3 /path/to/hello.py')
ENDPGM
```

Calling Python from RPG

```
**free
  ctl-opt dftactgrp(*no);

  dcl-pr system extproc('system');
    *n pointer value options(*string);
  end-pr;

  dcl-s command varchar(256);

  command = 'QSH CMD(''python3 /path/to/hello.py'')';
  system(command);

  *inlr = *on;
```

02

Consuming APIs

Practical examples of using the **requests** library

https://wright4i.com/guides/py/requests/

Importance of APIs

APIs (Application Programming Interfaces) are sets of protocols and tools for building software and applications. They allow different software systems to communicate, exchanging data and commands.

Benefits for IBM i Users:

- **Unlock Data:** Gain access to a vast array of data previously unavailable to our system.

- **Bridge Gaps:** Seamlessly connect your applications with modern services, enhancing functionality and user experience.

The Python Advantage

Python stands out for its **simplicity** and **readability**, making it an ideal choice for interacting with web APIs. With its vast standard library and third-party modules, Python simplifies the process of sending requests to a web server and handling the responses.

Furthermore, Python's popularity in the development community means you're likely to encounter Python example code in API documentation, making it easier to understand and implement API integrations.

One such third-party module is **requests**, which we will use in this tutorial.

Example - Short URL API

Here we import requests

GET from is.gd/create

Passing in our long URL

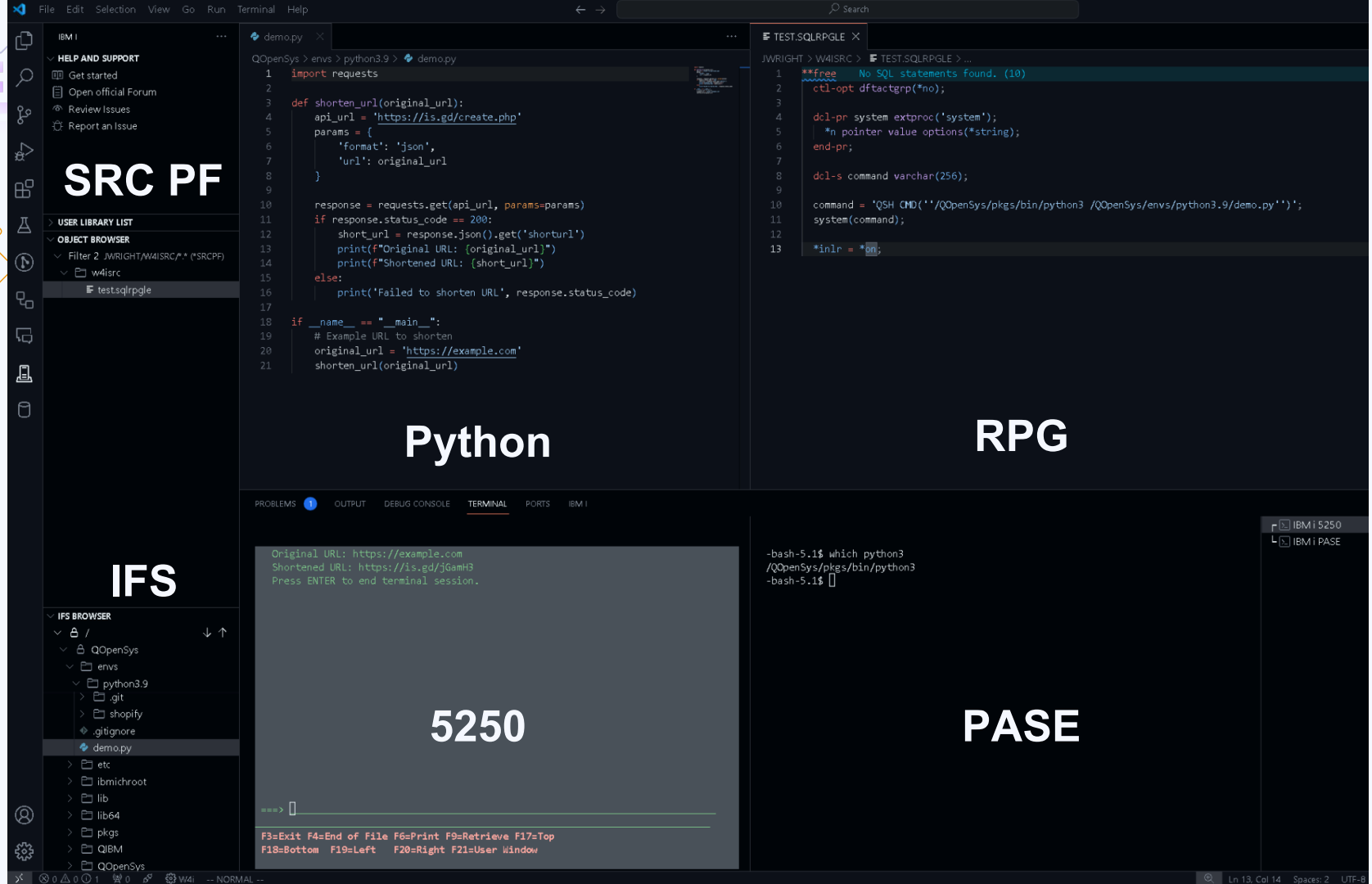
Do some error handling

Then print the short URL

```python
import requests

def shorten_url(original_url):
    api_url = 'https://is.gd/create.php'
    params = {
        'format': 'json',
        'url': original_url
    }

    response = requests.get(api_url, params=params)
    if response.status_code == 200:
        short_url = response.json().get('shorturl')
        print(f"Original URL: {original_url}")
        print(f"Shortened URL: {short_url}")
    else:
        print('Failed to shorten URL', response.status_code)

if __name__ == "__main__":
    # Example URL to shorten
    original_url = 'https://example.com'
    shorten_url(original_url)
```

The Results


PROBLEMS 1
OUTPUT
DEBUG CONSOLE
TERMINAL
PORTS
IBM i
Original URL: https://example.com
Shortened URL: https://is.gd/jGamH3
Press ENTER to end terminal session.
===>
F3=Exit F4=End of File F6=Print F9=Retrieve F17=Top
F18=Bottom F19=Left F20=Right F21=User Window
-bash-5.1$ which python3
/QOpenSys/pkgs/bin/python3
-bash-5.1$


File Edit Selection View Go Run Terminal Help
Search
IBM i
HELP AND SUPPORT
Get started
Open official Forum
Review Issues
Report an Issue
SRC PF
USER LIBRARY LIST
OBJECT BROWSER
Filter 2 JWRIGHT/W4ISRC/*.* (*SRCPF)
w4isrc
test.sqlrpgle
demo.py
QOpenSys > envs > python3.9 > demo.py
import requests
def shorten_url(original_url):
api_url = 'https://is.gd/create.php'
params = {
'format': 'json',
'url': original_url
}
response = requests.get(api_url, params=params)
if response.status_code == 200:
short_url = response.json().get('shorturl')
print(f"Original URL: {original_url}")
print(f"Shortened URL: {short_url}")
else:
print('Failed to shorten URL', response.status_code)
if __name__ == "__main__":
# Example URL to shorten
original_url = 'https://example.com'
shorten_url(original_url)
Python
TEST.SQLRPGLE
JWRIGHT > W4ISRC > TEST.SQLRPGLE > ...
**free No SQL statements found. (10)
ctl-opt dftactgrp(*no);
dcl-pr system extproc('system');
*n pointer value options(*string);
end-pr;
dcl-s command varchar(256);
command = 'QSH CMD(''/QOpenSys/pkgs/bin/python3 /QOpenSys/envs/python3.9/demo.py'')';
system(command);
*inlr = *on;
RPG
IFS
IFS BROWSER
/
QOpenSys
envs
python3.9
.git
shopify
.gitignore
demo.py
etc
ibmichroot
lib
lib64
pkgs
QIBM
QOpenSys
PROBLEMS 1 OUTPUT DEBUG CONSOLE TERMINAL PORTS IBM i
Original URL: https://example.com
Shortened URL: https://is.gd/jGamH3
Press ENTER to end terminal session.
5250
F3=Exit F4=End of File F6=Print F9=Retrieve F17=Top
F18=Bottom F19=Left F20=Right F21=User Window
IBM i 5250
IBM i PASE
-bash-5.1$ which python3
/QOpenSys/pkgs/bin/python3
-bash-5.1$
PASE
W4i -- NORMAL --
Ln 13, Col 14 Spaces: 2 UTF-8

03

Hosting APIs

Building a lightweight API web server with **flask**

https://wright4i.com/guides/py/flask/

What is flask

Flask is a minimalist yet powerful Python web framework

Akin to *Express* in Node, it offers simplicity and flexibility for developers without a lot of setup.

It enables seamless API creation, making it **effortless** to setup a quick API.

Flask also provides users with extensive customization options, ensuring tailored solutions for web services and applications when its time to go to Production.

Example – API to serve DB2 data

Import **flask** and our DB2 connection library

Setup your Flask app

Add a **GET** route

Query the DB using **empno**

```python
from flask import Flask, jsonify, request
import ibm_db_dbi

app = Flask(__name__)

@app.route('/employee/<empno>', methods=['GET'])
def get_employee(empno):
    try:
        conn = ibm_db_dbi.connect() # Include Connection details here if not *LOCAL
        cursor = conn.cursor()
        sql = """
        SELECT EMPNO, FIRSTNME, MIDINIT, LASTNAME, WORKDEPT, PHONENO,
               HIREDATE, JOB, EDLEVEL, SEX, BIRTHDATE, SALARY, BONUS, COMM
        FROM SAMPLE.EMPLOYEE
        WHERE EMPNO = ?
        """
        cursor.execute(sql, (empno,))
        employee = cursor.fetchone()
        cursor.close()
        conn.close()

```

Example – API to serve DB2 data

If the Query returned results

Create an object with our data

Jsonify our object to the caller

Error Handling:
404 – Not Found
500 – Internal Server Error

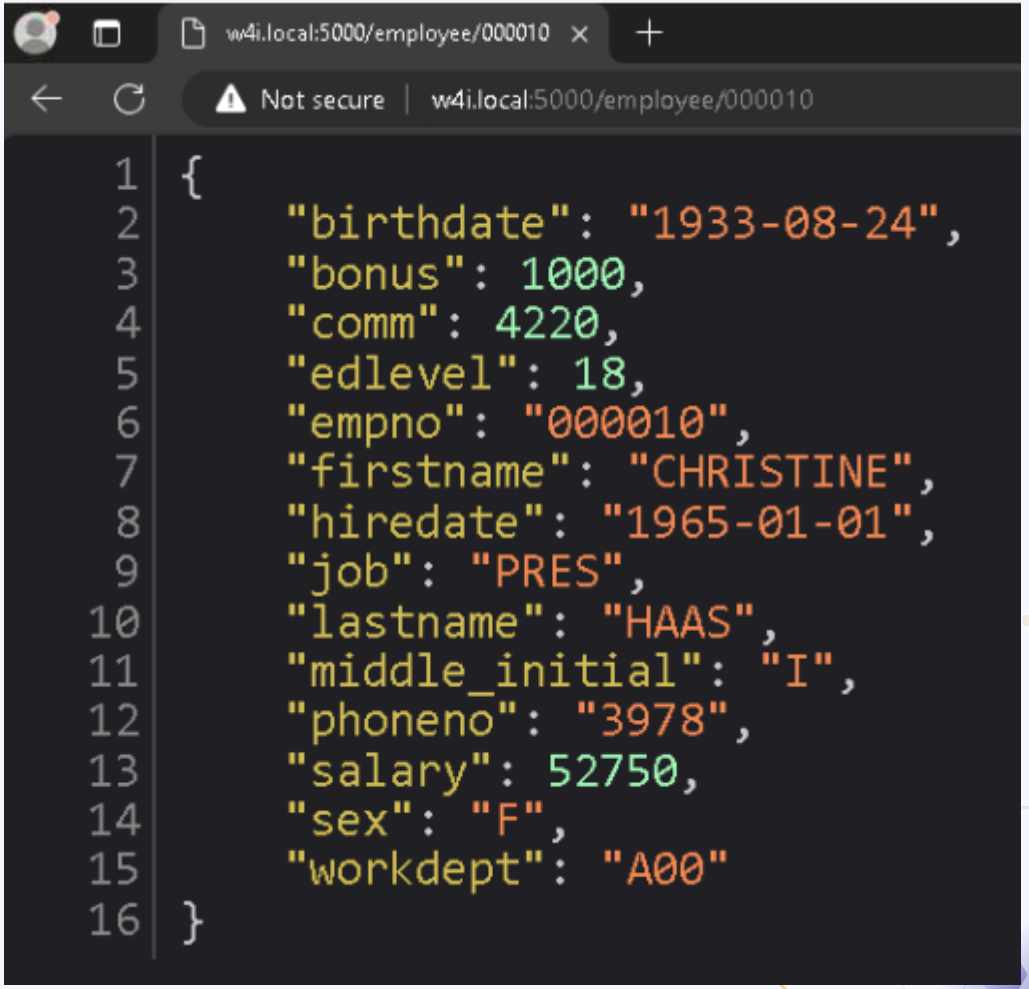
```
        if employee:
            employee_data = {
                'empno': employee[0].strip(),
                'firstname': employee[1].strip(),
                'middle_initial': employee[2].strip(),
                'lastname': employee[3].strip(),
                'workdept': employee[4].strip() if employee[4] else '',
                'phoneno': employee[5].strip(),
                'hiredate': employee[6].isoformat() if employee[6] else '',
                'job': employee[7].strip(),
                'edlevel': employee[8],
                'sex': employee[9].strip(),
                'birthdate': employee[10].isoformat() if employee[10] else '',
                'salary': float(employee[11]),
                'bonus': float(employee[12]),
                'comm': float(employee[13])
            }
            return jsonify(employee_data)
        else:
            return jsonify({'error': 'Employee not found'}), 404
    except Exception as e:
        return jsonify({'error': str(e)}), 500

if __name__ == '__main__':
    app.run(debug=True)
```

Starting the Flask Web App

```
-bash-5.1$ python3 demo2.py
 * Serving Flask app 'demo2'
 * Debug mode: on
WARNING: This is a development server. Do not use it in a production deployment. Us
e a production WSGI server instead.
 * Running on http://172.16.160.4:5000
Press CTRL+C to quit
 * Restarting with stat
 * Debugger is active!
 * Debugger PIN: 894-290-679
10.4.1.2 - - [29/Mar/2024 19:24:21] "GET /employee/000010 HTTP/1.1" 200 -
10.4.1.2 - - [29/Mar/2024 19:24:31] "GET /employee/000010 HTTP/1.1" 200 -
```

Successful call to the API

/employee/000010



Results of an invalid employee number

/employee/003953

04

Excel Templates

Inject data into preformatted Excel files with **openpyxl**

https://wright4i.com/guides/py/openpyxl/

Concepts

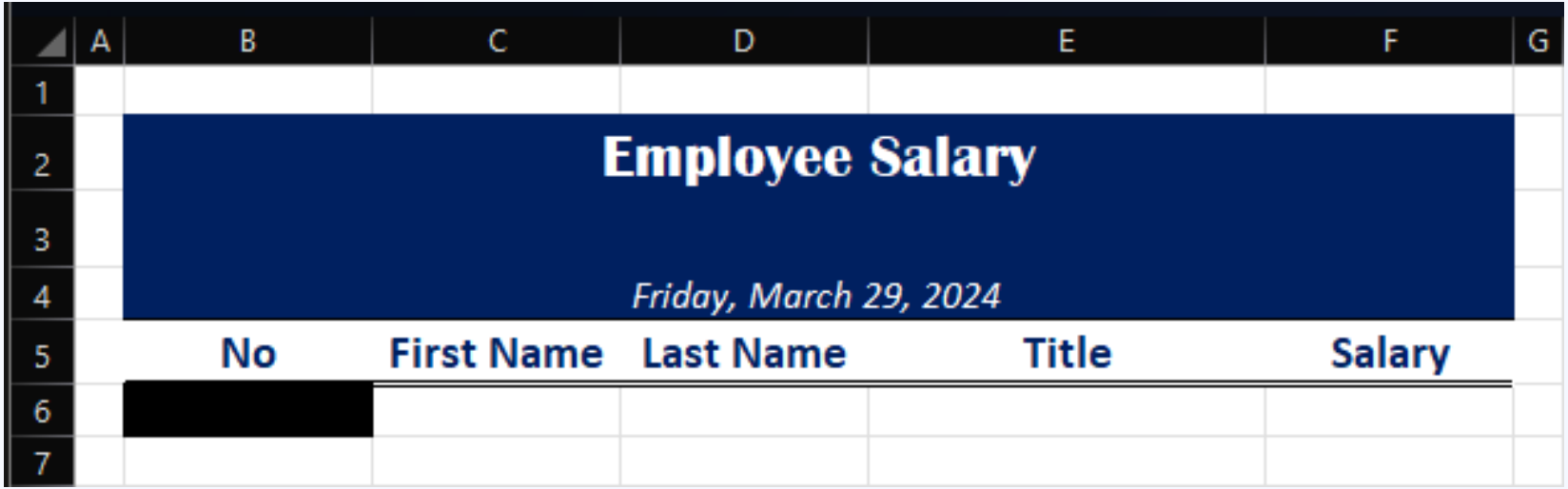
Template File

An .xlsx file with formatting – Colors, Headers, Merged Cells, Formulas, Images, and more.

Python Data Merge

Use Python and SQL to provide data to **openpyxl** which inserts it into a starting cell

Example - openpyxl template.xlsx



	A	B	C	D	E	F	G
1							
2		Employee Salary					
3							
4		*Friday, March 29, 2024*					
5		No	First Name	Last Name	Title	Salary	
6							
7							

Example – openpyxl copy function

Import our libraries

Define a function to copy all the formatting from the source to target cell

```
1   import argparse
2   import openpyxl
3   import ibm_db_dbi
4   from copy import copy
5
6   def copy_cell_formatting(source_cell, target_cell):
7       """
8       Copies all formatting from source_cell to target_cell.
9       """
10      target_cell.font = copy(source_cell.font)
11      target_cell.border = copy(source_cell.border)
12      target_cell.fill = copy(source_cell.fill)
13      target_cell.number_format = copy(source_cell.number_format)
14      target_cell.protection = copy(source_cell.protection)
15      target_cell.alignment = copy(source_cell.alignment)
16
```

Example – openpyxl generate

Load workbook

Connect to DB2 and run our Query (this could be an arg)

Determine the start position using the data_start parm.

Loop through the SQL result set writing to the worksheet while copying cell formatting

Close DB and output .xlsx

```python
def generate_report(template_path, output_path, data_start="A1"):
    # Load the workbook and select the active worksheet
    wb = openpyxl.load_workbook(template_path)
    ws = wb.active

    # Connect to the database
    conn = ibm_db_dbi.connect()
    cursor = conn.cursor()

    # Execute the query to fetch employee data
    sql = "SELECT EMPNO, FIRSTNME, LASTNAME, JOB, SALARY FROM SAMPLE.EMPLOYEE"
    cursor.execute(sql)

    # Parse the data_start to get row and column correctly for any column
    column, row = data_start[0], data_start[1:]
    start_row = int(row)
    start_col = openpyxl.utils.column_index_from_string(column)

    # Insert data into the worksheet starting from data_start
    for row_index, db_row in enumerate(cursor, start=start_row):
        for col_index, value in enumerate(db_row, start=start_col):
            # The template cell for formatting is in the first row (data_start row)
            template_cell = ws.cell(row=start_row, column=col_index)
            target_cell = ws.cell(row=row_index, column=col_index, value=value)
            # Copy formatting from the template cell to the new cell
            copy_cell_formatting(template_cell, target_cell)

    # Close the database cursor and connection
    cursor.close()
    conn.close()

    # Save the modified workbook to the specified output path
    wb.save(output_path)

```

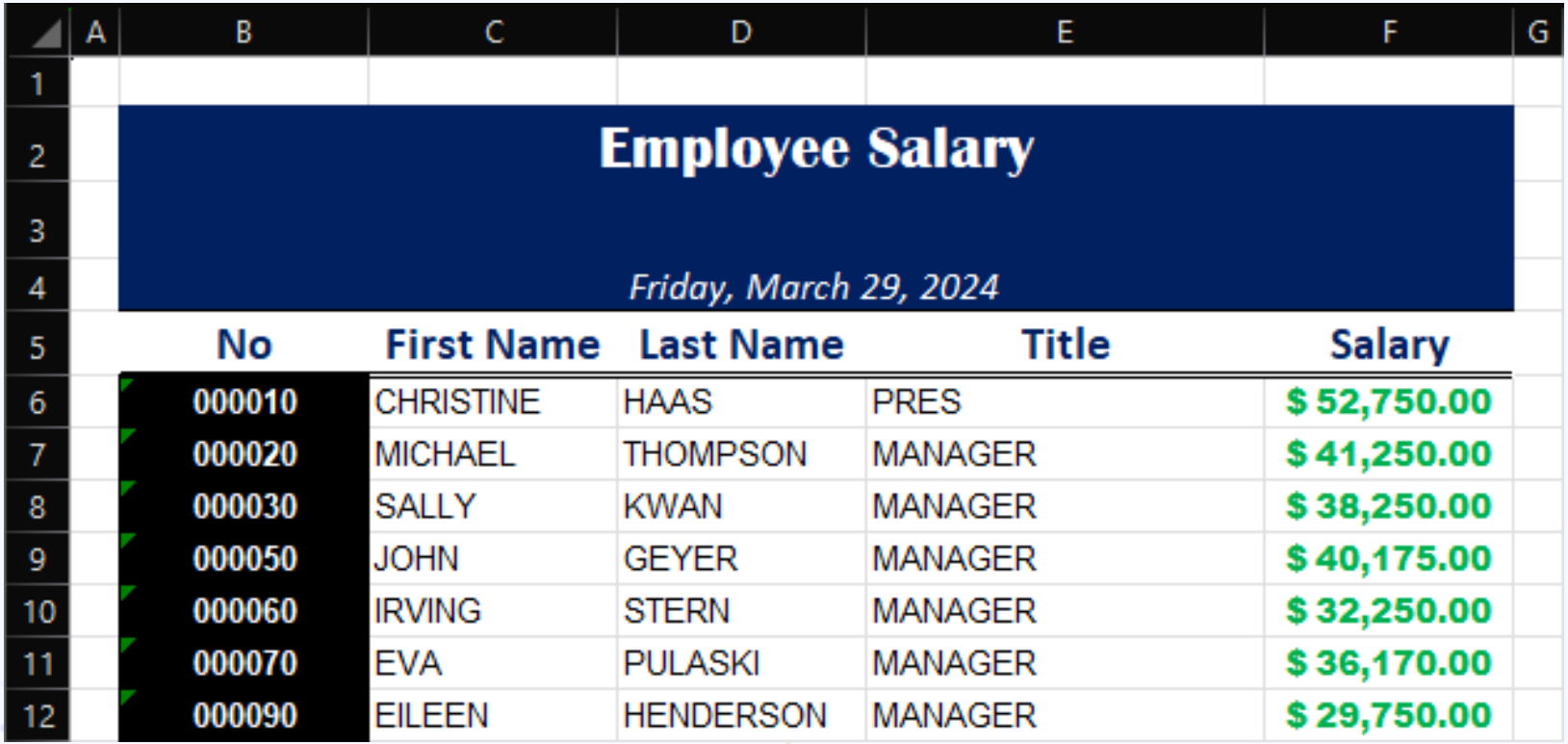
Example - openpyxl main

Setup the Argument Parser
template_path = **Required**, path of the template file
output_path = **Required**, path to save the new file
--data-start = **Optional**, starting cell for data merge (**A1** is Default)

Generate the report.

```
51  if __name__ == "__main__":
52      parser = argparse.ArgumentParser(description="Generate an Excel report from DB2 data.")
53      parser.add_argument("template_path", help="Path to the Excel template file.")
54      parser.add_argument("output_path", help="Path where the output Excel file will be saved.")
55      parser.add_argument("--data-start", default="A1", help="Cell position where data insertion begins (e.g., 'A6').")
56
57      args = parser.parse_args()
58
59      generate_report(args.template_path, args.output_path, args.data_start)
60
```

Example – openpyxl output.xlsx



	A	B	C	D	E	F	G
1							
2		**Employee Salary**					
3							
4		*Friday, March 29, 2024*					
5		**No**	**First Name**	**Last Name**	**Title**	**Salary**	
6		000010	CHRISTINE	HAAS	PRES	$ 52,750.00	
7		000020	MICHAEL	THOMPSON	MANAGER	$ 41,250.00	
8		000030	SALLY	KWAN	MANAGER	$ 38,250.00	
9		000050	JOHN	GEYER	MANAGER	$ 40,175.00	
10		000060	IRVING	STERN	MANAGER	$ 32,250.00	
11		000070	EVA	PULASKI	MANAGER	$ 36,170.00	
12		000090	EILEEN	HENDERSON	MANAGER	$ 29,750.00	

Q&A

Thank you OCEAN!

Presented by:
Joseph Wright
Wright4i.com / @Wright4i