Second PyPI example - Using PyExcel to read and write Excel files

Why is this important?

Excel is a pretty common tool that folks can use to sort and analyze data.

In this example we're going to take a look at `pyexcel` which is a package that allows us to read and write Excel files with Python.

Note: In future examples we're going to a more powerful package called `pandas` that allows us to do more complex data analysis.

What are we going to do?

We're going to create a virtual environment and install the `pyexcel` package from PyPI to see how it works. We're going to:

1. create an excel file with some game reviews data.
2. Add a new column to the file.

Steps

1. Just like the last example, create a virtual environment in your project folder.

```
python -m venv venv
```

This command creates a new directory called `venv` that contains the virtual environment.

2. Activate the virtual environment.

- On Windows:

```
venv\Scripts\activate
```

- On macOS and Linux:

```
source venv/bin/activate
```

When the virtual environment is activated, your command prompt will change to show the name of the environment (usually `venv`).

3. Let's install the `pyexcel` package from PyPI. This package allows us to read and write Excel files. A. install the package with pip:

```
pip install pyexcel pyexcel-xlsx
```

Note: this will install two packages, `pyexcel` and `pyexcel-xlsx`. The first is the core package and the second is a plugin that allows us to read and write Excel files specifically. Note: you can find the documentation for `pyexcel` here: PyExcel Documentation B. Ensure that you save the requirements for the project:

```
pip freeze > requirements.txt
```

The requiremments.txt file should look a little like below

```
et_xmlfile==2.0.0
lml==0.2.0
openpyxl==3.1.5
pyexcel==0.7.3
pyexcel-io==0.6.7
pyexcel-xlsx==0.6.1
texttable==1.7.0
```

Depending on when you take this course the versions may be different.

4. Now let's take a look at the code in `game_reviews.py` to see how we can use the `pyexcel` package to read and write Excel files.

```
# some times you can rename imports for convenience so that
# you don't have to type as much
import pyexcel as pex

def main():
    GAME_REVIEWS_FILE_NAME = "game_reviews.xlsx"
    # the data we're going to write to the excel file
    GAME_REVIEWS = [
        ["Title", "Platform", "Score"],
        ["Elden Ring", "PC", 95],
        ["Stardew Valley", "Switch", 89],
        ["Cyberpunk 2077", "PS5", 82],
        ["No Man's Sky", "PC", 90],
        ["Gollum", "PS5", 43],
    ]

if __name__ == "__main__":
```

```
    main()
```

You'll see this syntax quite often `import pyexcel as pex` which allows us to use `pex` as a shorthand for `pyexcel`.

5. Let's write a function to create the excel file with the game reviews data. reference to the documentation here

```
import pyexcel as pex


def create_game_reviews_excel_file(reviews, output_file):
    # saves the data to an excel file
    pex.save_as(array=reviews, dest_file_name=output_file)

def main():
    GAME_REVIEWS_FILE_NAME = "game_reviews.xlsx"
    GAME_REVIEWS = [
        ["Title", "Platform", "Score"],
        ["Elden Ring", "PC", 95],
        ["Stardew Valley", "Switch", 89],
        ["Cyberpunk 2077", "PS5", 82],
        ["No Man's Sky", "PC", 90],
        ["Gollum", "PS5", 43],
    ]
    # call our newly created function to create the excel file
    create_game_reviews_excel_file(GAME_REVIEWS, GAME_REVIEWS_FILE_NAME)
    print(f"Created {GAME_REVIEWS_FILE_NAME} with game reviews.")

if __name__ == "__main__":
    main()
```

Once you run the `game_reviews.py` script it will create a file called `game_reviews.xlsx` in the same directory as the script with the game reviews data. It should look something like this when you open it in Excel or another spreadsheet program:  game_reviews.xlsx image

6. Let's say that you got some feedback on your game reviews and you want to add a new column called "Recommended" that indicates whether the game is recommended based on the score, and you got a new requirement to add this column to the excel file. The recommendation criteria is:

- If the score is 90 or above, the value should be "Highly Recommended"
- If the score is between 80 and 89, the value should be "Recommended"
- If the score is between 70 and 79, the value should be "Average"
- If the score is below 70, the value should be "Not Recommended" We're going to take a look at docs on `get_sheet`

```
import pyexcel as pex

# ... other code ...

def add_recommendation_column_to_excel_file(input_file, output_file):
    # load the existing excel file
    sheet = pex.get_sheet(file_name=input_file)

    # add a new column for recommendations based on score
    recommendations = ["Recommendation"]
    rows_list = sheet.to_array()
    for row_index, row_values in enumerate(rows_list[1:]):  # skip header
          # skip header row
        score = row_values[2] # this is the score column (0-based index)
        if score >= 90:
            recommendations.append("Highly Recommended")
        elif score >= 80:
            recommendations.append("Recommended")
        elif score >= 70:
            recommendations.append("Average")
        else:
            recommendations.append("Not Recommended")

    sheet.column += recommendations

    # save the updated sheet to a new file
    sheet.save_as(output_file)

def main():
    # ... other code ...
    RECOMMENDED_GAME_REVIEWS_FILE_NAME = "recommended_game_reviews.xlsx"
    add_recommendation_column_to_excel_file(
        GAME_REVIEWS_FILE_NAME, RECOMMENDED_GAME_REVIEWS_FILE_NAME
    )
    print(f"Created {RECOMMENDED_GAME_REVIEWS_FILE_NAME} with
recommendations.")
```

Once you run the updated `game_reviews.py` script it will create a new file called `recommended_game_reviews.xlsx` in the same directory as the script with the new "Recommended" column added. It should look something like this when you open it in Excel or another spreadsheet program:
 recommended_game_reviews.xlsx image

Now when you run the `game_reviews.py` script it will create two excel files:

1. `game_reviews.xlsx` - the original file with game reviews.
2. `recommended_game_reviews.xlsx` - the updated file with the new "Recommended" column.

Summary

In this example we saw how to use the `pyexcel` package to read and write Excel files in Python. We created a virtual environment, installed the package, and wrote code to create and update an Excel file with game

reviews data.