Academic Sheet Automation Program (ASAP)

Overview

The Academic Sheet Automation Program (ASAP) automates the process of extracting student and semester data from multiple Excel files and populating a single CSV file (`UTD.csv`). This program is designed to simplify data entry tasks for academic purposes.

There are two versions of the program:

1. **`UTD.py`**: A script-based version that performs the automation without a graphical user interface (GUI).
2. **`GUI.py`**: A GUI-based version of the program that provides a user-friendly interface for selecting files and running the automation.

Both versions perform the same core functionality, but `GUI.py` includes a graphical interface for ease of use.

Conversion to Desktop Application

The `GUI.py` script has been converted into a standalone `.exe` desktop application. This allows users to run the program on their system without needing to install Python or any dependencies.

How to Use the .exe Application

1. Double-click the .exe file to launch the application.
2. Follow the same steps as described in the "Usage" section to select files and process data.
3. The output CSV file will be saved in the same directory as the selected master sheet.

The `.exe` file was created using **PyInstaller**.

Usage

`UTD.py`

To run the script-based version:

1. Ensure you have the required Excel files (`sem_sheet`, `all_sem_file`, `abc_file`) and the `UTD.csv` file in the same directory as the script.
2. Execute the script using Python:

```
python UTD.py
```

`GUI.py`

To run the GUI-based version:

1. Run the `GUI.py` script.
2. Use the "Browse" buttons to select the required input files (`sem_sheet`, `all_sem_file`, `abc_file`) and the output file (`UTD_file`).
3. Click the "Run" button to process the data.
4. **Important**: All files must be selected; otherwise, the program will not run.
5. The output CSV file will be saved in the same directory as the selected master sheet.

Output

The program generates a new CSV file named `UTD_<batch>_<current_semester>.csv` containing the processed student and semester data.

Notes

- Ensure the input files are in the correct format and contain the required data.
- Close the output CSV file if it is open in another program before running the script.
- For any issues, refer to the error messages displayed by the program.

Code Dependencies

- pandas
- roman
- tkinter (for `GUI.py`)
- warnings

Features

- **Automated Data Processing**:
    - Reads data from input Excel files.
    - Extracts and processes student and semester details.
    - Calculates grades, grade points, SGPA, CGPA, and other academic details.
    - Fills constant and variable data into the output CSV file.
- **File Validation**:

    - Ensures that input files are Excel files (`.xls` or `.xlsx`).
    - Ensures that the output file is a CSV file (`.csv`).
- **Error Handling**:
    - Displays appropriate error messages for missing files, invalid file formats, or processing errors.
- **Output File Location**:
    - The output CSV file is saved in the same directory as the selected master sheet (`sem_sheet`).

Steps Performed by the Program

Step 1: Read the Excel Files

The program reads the following input files:

- **Master Sheet for Semester (`sem_sheet`)**: Contains semester details.
- **Student Details File (`all_sem_file`)**: Contains student details.
- **ABC File (`abc_file`)**: Contains ABC IDs for students.

The data is loaded into pandas DataFrames without headers to allow access by row and column indices.

Step 2: Extract Data from Excel Files

- **Batch**: Extracted from the `all_sem_file`.
- **Parents' Names**: Extracted from the `all_sem_file`.
- **Roll Numbers**: Extracted from the `sem_sheet`.
- **Enrollment Numbers**: Extracted from the `sem_sheet`.
- **Student Names**: Extracted from the `sem_sheet`.
- **Semester Number**: Extracted from the roll numbers.
- **Attempts**: Extracted from the `all_sem_file`.
- **Credits, SGPA, and Results**: Extracted from the `sem_sheet`.
- **Subjects**: Extracted from the `sem_sheet`.
- **Grades and Grade Points**: Calculated from the subjects and credits.

Step 3: Read the CSV File

The program reads the `UTD.csv` file into a pandas DataFrame without headers.

Step 4: Ensure UTD.csv Has Enough Rows and Columns

The program calculates the required number of rows based on the length of the roll and enrollment numbers. If the CSV file does not have enough rows, additional rows are added.

Step 5: Fill Constant Data for All Students

The program fills columns A-F with constant values for all students:

- **Column A**: University Name
- **Column B**: College Name
- **Column C**: Course Name in Short
- **Column D**: Full Course Name
- **Column E**: Full Course Name in Detail
- **Column F**: Stream
- **Column H**: Session for Batch
- **Column S**: Year for Batch
- **Column T**: Month for Batch
- **Column X**: Semester Number
- **Column Y**: Semester Type
- **Column AB**: Total Credits

Step 6: Write the Data into UTD.csv

The program writes the extracted data into specific columns of the `UTD.csv` file:

- **Column I**: Enrollment Numbers
- **Column J**: Roll Numbers
- **Column K**: Student Names
- **Column N**: Fathers' Names
- **Column O**: Mothers' Names
- **Column Q**: Marksheet Status
- **Column R**: Results
- **Column U**: Division
- **Column AC**: Credits Earned
- **Column AF**: Credits Earned
- **Column AG**: CGPA
- **Column AI**: SGPA
- **Column AM**: Student Names

The program also fills the constant data for subjects, credits, and grades for each student in d.

Step 7: Save the Updated CSV File

The program saves the updated CSV file with a new name based on the session and current semester. The file is saved in the same directory as the selected master sheet (`sem_sheet`).

Conclusion

The Academic Sheet Automation Program (ASAP) is a powerful tool designed to simplify and automate the process of managing academic data. Whether you prefer the command-line interface of `UTD.py` or the user-friendly graphical interface of `GUI.py`, this program ensures accurate and efficient data processing.

By converting `GUI.py` into a standalone `.exe` application, users can now run the program on any system without the need for Python or additional dependencies. This makes the program accessible and convenient for a wide range of users.

For any issues or further assistance, please refer to the error messages displayed by the program or consult the documentation. Thank you for using ASAP to streamline your academic data entry tasks!
