Flask App Documentation

Approach:
The app displays an upload file page to upload the files for text extraction. The filename is obtained from the post request data. Data sanitization checks are performed to discard files without extensions and files other than PDFs and Images. The uploaded file is saved temporarily to the local temp folder and then the process_files function (from the script used in Task 1) is called on this temporary file. This function returns a list of output file names of the CSV files which are stored in the same directory as the flask app (could have also saved this as temporary files but that would have needlessly complicated the task). The CSV files are then read and displayed in a webpage.

Design choices:
Used Flask over Django for obvious reasons. Wanted to keep the web app minimal so used the same script as in Task 1. The uploaded file is stored locally in a temp folder and not in the app directory. Only the generated CSV files are stored in the app directory (Files are saved as output_1, output_2, … etc based on the number of pages in PDF). The CSV files won't keep on accumulating after each run. Used Bootstrap for styling.

Instructions to run the web app:
Install the necessary dependencies and libraries

- Tesseract
- PDF2Image and Poppler
- Flask
- PyTesseract
- OpenCV
- Numpy
- pprint

There are debug print statements. Use debug mode to see them.

Run the app normally:

```
flask run
```

Run the app in debug mode:

```
flask --app app.py --debug run
```