DJVU to PDF Converter

Important Packages

- `DjVuLibre` : You can download it from this website: http://djvu.sourceforge.net/
- `ImageMagick` : You can download it from this website: https://imagemagick.org/script/download.php

Outline:

1. **Import Libraries**:
    - Import the neccessary libraries (`os` for file-related ooperations and `subprocess` for running external commands.)
2. **Define Funtion 'Converter'**
    - Takes two parameters: `input_djvu` and `output_djvu` .
    - Checks if the input DJVU file exists.
    - Converts DJVU to PNM using `ddjvu` command line tool.
    - Checks if the PNM file was created successfully.
    - Converts PNM to PDF using `convert` command line tool.
    - Checks if the PDF file was created successfully.
    - Cleans up the intermediate PNM file.
    - Prints a success message.
3. **Handle Exceptions:**
    - Uses a `try-except` block to catch nay exceptions during the process.
    - Prints an error message if an exception occurs.

Step by Step explaination:

```
# Import necessary libraries
import os
import subprocess
```

These lines import the required Python libraries.

- `os` provides a way to interact with the operating system (checking file existence, removing files)
- `subprocess` is used to run external commands from within Python.

```
def converter(input_djvu, output_pdf):
        try:
                # Check if input DJVU file exists
                if not os.path.exists(input_djvu):
                        raise FileNotFoundError(f"The fi
le {input_djvu} does not exists.")
```

This function `converter` takes two parameters:

- `input_djvu` : the input djvu file
- `output_pdf` : the desire output PDF file

It starts with a check to see if the input DJVU file exists in the current folder. If not, it raises a `FileNotFoundError` with an appropriate message.

```
            # Use DjVuLibre's `ddjvu` command line tool
to convert DJVU to PNM format
        pnm_filename = output_pdf.replace(".pdf", ".pn
m")
        subprocess.run(["ddjvu", "-format=pnm", input_dj
vu, pnm_filename])
```

This part of the code calls the `ddjvu` command line tool with subprocess to convert the input DJVU file into PNM (Portable Anymap) format. It creates a temporary PNM file with a similar name to the output PDF.

▼ Why to we need to convert DJVU to PNM first?

The conversion from DJVU to PNM (Portable Anymap) is an intermediate step in the process because the `convert` command from ImageMagick, used for the final conversion to PDF, doesn't directly support DJVU input.

1. **DJVU to PNM:**
    - `ddjvu` is a command-line tool from DjVuLibre that converts DJVU files to PNM format.
    - PNM is a simple and versatile image format that stands for Portable Anymap. It's a common intermediate format used for various image processing tasks.
2. **PNM to PDF:**
    - The `convert` command from ImageMagick is then used to convert the PNM file to PDF.
    - ImageMagick supports a wide range of image formats, including PNM, and can easily convert them to PDF.

```
            # Check if PNM file was created successfully
        if not os.path.exists(pnm_filename):
            raise Exception("Conversion to PNM failed.")
```

After the conversion, it checks if the PNM file was created successfully. If not, it raises an exception indicating that the conversion failed.

```
        # Use ImageMagick's `convert` command line tool
to convert PNM to PDF
```

```
        subprocess.run(["convert", pnm_filename, output_
pdf])
```

This part of the code uses the `convert` command line tool from ImageMagick to convert the PNM file into the final PDF format.

```
        # Check if PDF file was created successfully
        if not os.path.exists(output_pdf):
            raise Exception("Conversion to PDF failed.")
```

It then checks if the PDF file was created successfully. If not, it raises an exception idicating that the conversion to PDF failed.

```
        # Clean up intermediate PNM file
        os.remove(pnm_filename)
```

After the successfull conversion, it removes the temporary PNM file to clean up.

```
   print(f"Conversion successful. PDF saved as {output_p
df}")
```

Finally, if everything is successful, it prints a message indicating the conversion was successful and the location of the saved PDF file.

```
    except Exception as e:
                print(f"Error: {e}")
```

The `try-except` block catches any exceptions that might occur during the process and prints an error message if there’s an issue.