# Bulk DOCX to PDF

give me bulk docx to bulk pdf automation with python, alignment, font size page size will be exactly same

You can **bulk convert DOCX files to PDF** with Python while preserving alignment, font size, and page size using the **Microsoft Word COM interface** (Windows only), or **LibreOffice** (cross-platform, but less precise with complex formatting). The **COM approach** using `pywin32` is the **most accurate** for preserving all formatting.

Here's a **step-by-step guide with full code** for both methods:

---

## Method 1: Using Microsoft Word (Best for Exact Formatting, Windows Only)

### Requirements

- Windows OS
- Microsoft Word installed
- Install Python package:

  ```bash
  pip install pywin32
  ```

### Bulk Conversion Script

```python
import os
import win32com.client

def docx_to_pdf_bulk(input_folder, output_folder):
    if not os.path.exists(output_folder):
        os.makedirs(output_folder)

    # Launch Word
    word = win32com.client.Dispatch('Word.Application')
    word.Visible = False

    for filename in os.listdir(input_folder):
        if filename.lower().endswith('.docx'):
            docx_path = os.path.join(input_folder, filename)
            pdf_path = os.path.join(output_folder, filename[:-5] + '.pdf')
            print(f"Converting {filename} ...")
            doc = word.Documents.Open(docx_path)
            doc.SaveAs(pdf_path, FileFormat=17)  # 17 = PDF format
            doc.Close()
    word.Quit()
    print("All files converted!")
```

```
# Usage:
input_dir = r"C:\path\to\input\docx"
output_dir = r"C:\path\to\output\pdf"
docx_to_pdf_bulk(input_dir, output_dir)
```

**Features:**

- Keeps all formatting, fonts, alignment, page size as in DOCX.
- Fast and reliable if you have MS Word.

---

## Method 2: Using LibreOffice (Cross-platform, Pretty Good)

### Requirements

- LibreOffice installed (`soffice` in PATH)
- Install Python package:

  ```
  bash

  pip install tqdm
  ```

- For **Ubuntu**, install LibreOffice:

  ```
  bash

  sudo apt-get install libreoffice
  ```

### Bulk Conversion Script

```
python

import os
import subprocess
from tqdm import tqdm

def bulk_convert_docx_to_pdf(input_folder, output_folder):
    if not os.path.exists(output_folder):
        os.makedirs(output_folder)

    files = [f for f in os.listdir(input_folder) if f.lower().endswith('.docx')]
    for filename in tqdm(files):
        docx_path = os.path.join(input_folder, filename)
        command = [
            "soffice",
            "--headless",
            "--convert-to", "pdf",
            "--outdir", output_folder,
            docx_path
        ]
        subprocess.run(command, stdout=subprocess.PIPE, stderr=subprocess.PIPE)

# Usage:
input_dir = "/path/to/docx"
output_dir = "/path/to/pdf"
bulk_convert_docx_to_pdf(input_dir, output_dir)
```

**Note:**

- LibreOffice is pretty good, but if you have complex equations, tables, or custom fonts, some formatting *may* shift.

---

## Summary Table

| Method | Platform | Formatting Fidelity | Speed | Requirements |
|---|---|---|---|---|
| MS Word + pywin32 | Windows Only | Perfect | Fast | MS Word, pywin32 |
| LibreOffice | Windows/Linux/Mac | Very Good | Fast | LibreOffice, Python |

---

## Recommendation

- **For exact formatting, always use the MS Word COM method (`pywin32`).**
- If on Linux/Mac, try LibreOffice, but check outputs for any small shifts.

---

If you want a **Drag-and-drop GUI**, let me know! Or if you need batch conversion for `.doc` as well as `.docx`, I can add that too.
Let me know your platform (Windows/Linux/Mac) if you want a tailored solution!

give me Method 1 with tikenter, also add pptx to pdf, in same folder have pptx and docx

Absolutely! Here's a **complete Python GUI app** (using `tkinter`) that lets you **select a folder**, and then **bulk converts all `.docx` and `.pptx` files in that folder to PDF** (in a selected output folder), while keeping all formatting, alignment, font size, and page size intact.

- It uses **MS Word** for DOCX → PDF, and **MS PowerPoint** for PPTX → PDF, via `pywin32`.
- **Works on Windows only** (requires MS Office installed).

---

### 1. Install Prerequisites

```bash
pip install pywin32
```

---

### 2. Full Tkinter App Code

```python
import os
import tkinter as tk
from tkinter import filedialog, messagebox
import win32com.client
```

```
def convert_files(input_folder, output_folder):
    word = win32com.client.Dispatch('Word.Application')
    ppt = win32com.client.Dispatch('PowerPoint.Application')
    word.Visible = False
    ppt.Visible = False

    count_docx, count_pptx = 0, 0

    for filename in os.listdir(input_folder):
        in_path = os.path.join(input_folder, filename)
        if filename.lower().endswith('.docx'):
            out_path = os.path.join(output_folder, filename[:-5] + '.pdf')
            try:
                doc = word.Documents.Open(in_path)
                doc.SaveAs(out_path, FileFormat=17)
                doc.Close()
                count_docx += 1
            except Exception as e:
                print(f"Error converting {filename}: {e}")
        elif filename.lower().endswith('.pptx'):
            out_path = os.path.join(output_folder, filename[:-5] + '.pdf')
            try:
                presentation = ppt.Presentations.Open(in_path, WithWindow=False)
                presentation.SaveAs(out_path, 32)  # 32 = PDF
                presentation.Close()
                count_pptx += 1
            except Exception as e:
                print(f"Error converting {filename}: {e}")
    word.Quit()
    ppt.Quit()
    return count_docx, count_pptx

def start_conversion():
    input_folder = input_folder_var.get()
    output_folder = output_folder_var.get()
    if not input_folder or not output_folder:
        messagebox.showerror("Error", "Please select both folders.")
        return
    msg.set("Converting, please wait...")
    root.update_idletasks()
    docx, pptx = convert_files(input_folder, output_folder)
    msg.set(f"Done! Converted {docx} DOCX and {pptx} PPTX files.")

def browse_input():
    folder = filedialog.askdirectory()
    if folder:
        input_folder_var.set(folder)

def browse_output():
    folder = filedialog.askdirectory()
    if folder:
        output_folder_var.set(folder)

root = tk.Tk()
root.title("Bulk DOCX & PPTX to PDF Converter (MS Office)")

input_folder_var = tk.StringVar()
output_folder_var = tk.StringVar()
msg = tk.StringVar()

tk.Label(root, text="Input Folder (contains .docx/.pptx):").grid(row=0, column=0, sticky="e")
tk.Entry(root, textvariable=input_folder_var, width=45).grid(row=0, column=1)
tk.Button(root, text="Browse", command=browse_input).grid(row=0, column=2)

tk.Label(root, text="Output Folder (PDFs will be saved here):").grid(row=1, column=0,
sticky="e")
tk.Entry(root, textvariable=output_folder_var, width=45).grid(row=1, column=1)
tk.Button(root, text="Browse", command=browse_output).grid(row=1, column=2)

tk.Button(root, text="Convert All", command=start_conversion, bg="#2ecc71", fg="white", font=
("Arial", 12, "bold")).grid(row=2, column=0, columnspan=3, pady=15)

tk.Label(root, textvariable=msg, fg="blue", font=("Arial", 11)).grid(row=3, column=0,
columnspan=3)
```

```
root.resizable(False, False)
root.mainloop()
```

---

### How to Use:

1. **Save the code as `bulk_converter.py`**.
2. **Run:**

   ```bash
   python bulk_converter.py
   ```

3. **Select the folder** containing your `.docx` and `.pptx` files.
4. **Select an output folder** for the PDFs.
5. **Click "Convert All"**.

---

### Note:

- Keeps all alignment, font, and page size exactly as in the originals (just like "Save As PDF" in MS Office).
- Needs Microsoft Word and PowerPoint installed on your Windows PC.
- Will **not** convert `.doc` or `.ppt` (old formats), but I can add that if needed!

---

Let me know if you want:

- Progress bar
- Support for `.doc`/`.ppt`
- Log file
- Or any other features!

Just ask!

```
PS D:\Office Files\Developer\Bulk Docx to PDF>  d:; cd 'd:\Office Files\Developer\Bulk Docx to PDF'; & 'c:\Python313\python.exe' 'c:\Users\Moaz_AFS\.vscode\extensions\ms-python.debugpy-2025.8.0-win32-x64\bundled\libs\debugpy\launcher' '54147' '--' 'D:\Office Files\Developer\Bulk Docx to PDF\docx_to_pdf_bulk.py'
Exception in Tkinter callback
Traceback (most recent call last):
  File "c:\Python313\Lib\tkinter\__init__.py", line 2068, in __call__
    return self.func(*args)
           ~~~~~~~~~^^^^^^^
  File "D:\Office Files\Developer\Bulk Docx to PDF\docx_to_pdf_bulk.py", line 46, in start_conversion
    docx, pptx = convert_files(input_folder, output_folder)
```

```
          ~~~~~~~~~~~~~^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
  File "D:\Office Files\Developer\Bulk Docx to PDF\docx_to_pdf_bulk.py", line 10, in convert_files
    ppt.Visible = False
    ^^^^^^^^^^^
  File "C:\Users\Moaz_AFS\AppData\Roaming\Python\Python313\site-
packages\win32com\client\dynamic.py", line 677, in __setattr__
    self._oleobj_.Invoke(entry.dispid, 0, invoke_type, 0, value)
    ~~~~~~~~~~~~~~~~~~~~^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
pywintypes.com_error: (-2147352567, 'Exception occurred.', (0, 'Microsoft PowerPoint',
'Application.Visible : Invalid request.  Hiding the application window is not allowed.', '', 0,
-2147188160), None)
```

Great catch—**PowerPoint does not allow `Visible = False`** like Word does! That line causes the error:

```
'Application.Visible : Invalid request. Hiding the application window is not allowed.'
```

**Solution:**
Remove or comment out `ppt.Visible = False`.

---

## Here's the corrected section of your code:

```python
def convert_files(input_folder, output_folder):
    word = win32com.client.Dispatch('Word.Application')
    ppt = win32com.client.Dispatch('PowerPoint.Application')
    word.Visible = False
    # ppt.Visible = False  # REMOVE THIS LINE

    count_docx, count_pptx = 0, 0

    for filename in os.listdir(input_folder):
        in_path = os.path.join(input_folder, filename)
        if filename.lower().endswith('.docx'):
            out_path = os.path.join(output_folder, filename[:-5] + '.pdf')
            try:
                doc = word.Documents.Open(in_path)
                doc.SaveAs(out_path, FileFormat=17)
                doc.Close()
                count_docx += 1
            except Exception as e:
                print(f"Error converting {filename}: {e}")
        elif filename.lower().endswith('.pptx'):
            out_path = os.path.join(output_folder, filename[:-5] + '.pdf')
            try:
                presentation = ppt.Presentations.Open(in_path, WithWindow=False)
                presentation.SaveAs(out_path, 32)  # 32 = PDF
                presentation.Close()
                count_pptx += 1
            except Exception as e:
                print(f"Error converting {filename}: {e}")
    word.Quit()
    ppt.Quit()
    return count_docx, count_pptx
```

---

- Just **remove `ppt.Visible = False`** (or comment it out).
- All other logic remains the same.

- PowerPoint’s window may flash briefly but will close automatically.

---

**Try this fix and your program will run!**
If you want more advanced error handling or a progress bar, let me know!

```
PS D:\Office Files\Developer\Bulk Docx to PDF>  d:; cd 'd:\Office Files\Developer\Bulk Docx to PDF'; & 'c:\Python313\python.exe' 'c:\Users\Moaz_AFS\.vscode\extensions\ms-python.debugpy-2025.8.0-win32-x64\bundled\libs\debugpy\launcher' '54208' '--' 'D:\Office Files\Developer\Bulk Docx to PDF\docx_to_pdf_bulk.py'
Error converting AP - One Shot - Higher Mathematics - Class 10 - অধ্যায় ০৯ - সূচকীয় ও লগারিদমীয় ফাংশন - Lecture Slide.pptx: (-2147352567, 'Exception occurred.', (0, None, None, None, 0, -21474672   259), None)
Error converting AP_-_MCQ_Sheet_-_Class_6_-_Chapter_1.1^J_1.2^J_1.3^J_1.4^J_1.5^J_1.6_-_স্বাভাবিক_সংখ্যা_ও_ভগ্নাংশ.docx: (-2147352567, 'Exception occurred.', (0, 'Microsoft Word', "Sorry, we couldn  n't find your file. Was it moved, renamed, or deleted?\r (D:\\//Office%20Files/Developer/Bulk%20...)", 'wdmain11.chm', 24654, -2146823114), None)
Error converting AP_-_MCQ_Sheet_-_Class_6_-_Chapter_1.1^J_1.2^J_1.3^J_1.4^J_1.5^J_1.6_-_স্বাভাবিক_সংখ্যা_ও_ভগ্নাংশ_Formatted.docx: (-2147352567, 'Exception occurred.', (0, 'Microsoft Word', "Sorry,  , we couldn't find your file. Was it moved, renamed, or deleted?\r (D:\\//Office%20Files/Developer/Bulk%20...)", 'wdmain11.chm', 24654, -2146823114), None)
PS D:\Office Files\Developer\Bulk Docx to PDF>
```

You’re getting **two types of errors**:

---

## 1. PowerPoint Conversion Error

```plaintext
Error converting ... .pptx: (-2147352567, 'Exception occurred.', (0, None, None, None, 0,
-21474672259), None)
```

- This is a **generic PowerPoint error**. It often means the file is open, corrupted, password-protected, or there’s an unsupported feature.
- Sometimes it happens if file paths are too long, have special characters, or contain non-ASCII characters (like Bangla). PowerPoint COM can be **picky with Unicode paths**.

---

## 2. Word Conversion Error

```plaintext
Error converting ... .docx: (-2147352567, 'Exception occurred.', (0, 'Microsoft Word', "Sorry,
we couldn't find your file. Was it moved, renamed, or deleted?", ...))
```

- This is usually a **path issue**:

- Double slashes or forward slashes in Windows paths
- Non-ASCII characters in file names or folders (some old Office COM versions have trouble)
- File doesn’t exist (was renamed/moved/deleted)
- File name too long

---

# How to Fix & Debug

## A. Normalize Path Handling

Use `os.path.normpath()` on all paths sent to Office:

```python
in_path = os.path.normpath(os.path.join(input_folder, filename))
out_path = os.path.normpath(os.path.join(output_folder, filename[:-5] + '.pdf'))
```

## B. Handle Special Characters in Paths

- Office COM sometimes fails with Unicode or very long file names.
- Try with a simple test: Place a `.docx` or `.pptx` named `test.docx` in `C:\Temp` and see if it works.

## C. Add More Error Info

Print the `in_path` and `out_path` before opening/saving, so you can see what path is being used.

```python
print(f"Input: {in_path}")
print(f"Output: {out_path}")
```

## D. Files Should Not Be Open

- Close all files in Word/PowerPoint before converting.

---

# Updated Example: With Debugging

```python
for filename in os.listdir(input_folder):
    in_path = os.path.normpath(os.path.join(input_folder, filename))
    if filename.lower().endswith('.docx'):
        out_path = os.path.normpath(os.path.join(output_folder, filename[:-5] + '.pdf'))
        try:
            print(f"Converting Word: {in_path} → {out_path}")
            doc = word.Documents.Open(in_path)
            doc.SaveAs(out_path, FileFormat=17)
            doc.Close()
            count_docx += 1
        except Exception as e:
            print(f"Error converting {filename}: {e}")
    elif filename.lower().endswith('.pptx'):
        out_path = os.path.normpath(os.path.join(output_folder, filename[:-5] + '.pdf'))
        try:
            print(f"Converting PPT: {in_path} → {out_path}")
            presentation = ppt.Presentations.Open(in_path, WithWindow=False)
            presentation.SaveAs(out_path, 32)  # 32 = PDF
            presentation.Close()
            count_pptx += 1
        except Exception as e:
            print(f"Error converting {filename}: {e}")
```

---

## Recommendations

1. **Test on a short, ASCII-named file in a simple path first.**
2. **Move problematic files to a short path (e.g., `C:\Temp`) and rename to `test.docx` or `test.pptx` and retry.**
3. If it works, your issue is **file name/path/character** related.
    - If not, the file may be corrupted.

---

## If You Want to Automatically Skip Unopenable Files

Add a check:

```python
if not os.path.isfile(in_path):
    print(f"File does not exist: {in_path}")
    continue
```

---

**Try these fixes and check your paths—report back if there's a specific file that still fails even with a simple name/location. If so, upload a minimal example (name/path), and I'll help debug further!**

```
PS D:\Office Files\Developer\Bulk Docx to PDF> pyinstaller --onefile docx_to_pdf_bulk.py
pyinstaller : The term 'pyinstaller' is not recognized as the name of a cmdlet, function, script file, or
operable program. Check the spelling of the name, or if a path was
included, verify that the path is correct and try again.
At line:1 char:1
+ pyinstaller --onefile docx_to_pdf_bulk.py
+ ~~~~~~~~~~~
    + CategoryInfo          : ObjectNotFound: (pyinstaller:String) [], CommandNotFoundException
    + FullyQualifiedErrorId : CommandNotFoundException

PS D:\Office Files\Developer\Bulk Docx to PDF>
```

The error message means **PyInstaller is not installed** in your system or in your current Python environment.

---

# How to fix

## 1. Install PyInstaller

Open your terminal/PowerShell and run:

```bash
pip install pyinstaller
```

If your computer has multiple Python installations, make sure you are installing it for the correct one. For your Python 3.13, you may need to use:

```bash
python -m pip install pyinstaller
```

or

```bash
c:\Python313\python.exe -m pip install pyinstaller
```

---

## 2. Try the Command Again

After installation, try building your exe:

```bash
pyinstaller --onefile docx_to_pdf_bulk.py
```

If **`pyinstaller`** is still not recognized, run it with python:

```bash
python -m PyInstaller --onefile docx_to_pdf_bulk.py
```

or

```bash
c:\Python313\python.exe -m PyInstaller --onefile docx_to_pdf_bulk.py
```

---

# Summary

- Install PyInstaller with **`pip install pyinstaller`**
- Run the command again as above.

---

**Tip:**
If you're in VS Code, make sure your terminal is using the same Python interpreter as your project (check bottom left in VS Code).

Let me know if you face any more issues!