以下に、あなたの要件に基づいて、

* ✅ **Pythonスクリプトを .exe に変換（PyInstaller）**
* ✅ **C#アプリをGUI（WinFormsまたはWPF）化**

について具体的に解説します。

**✅ 1. Pythonスクリプトを .exe に変換（PyInstaller）**

**🔧 ステップ 1: PyInstaller のインストール**

pip install pyinstaller

**🔧 ステップ 2: スクリプトを .exe に変換**

プロジェクトディレクトリに移動し、以下を実行：

pyinstaller --onefile --noconsole pdf\_process.py

| **オプション** | **説明** |
| --- | --- |
| --onefile | 実行ファイルを 1 個にまとめる |
| --noconsole | GUIアプリ用。ログ出さない（※CLIなら不要） |

**📁 生成物**

dist/

├─ pdf\_process.exe ← これを配布・実行可

build/

pdf\_process.spec

**注意**：

* .exe のサイズはやや大きめ（30〜70MB）
* 実行PCにPythonがインストールされていなくても動作します
* 必要なDLLやOCRモデル（tessdata/jpn.traineddataなど）は同梱するか、パスで読み込む工夫が必要です

**✅ 外部ファイルの扱いについて**

tessdata や poppler など外部ツールが必要な場合は、以下のように同梱用ディレクトリを用意し、コードで os.path を使って相対パス対応してください：

tesseract\_cmd = os.path.join(os.path.dirname(sys.executable), "Tesseract-OCR", "tesseract.exe")

pytesseract.pytesseract.tesseract\_cmd = tesseract\_cmd

**✅ 2. C#アプリを GUI 化（WinForms または WPF）**

**🔧 選択肢**

| **フレームワーク** | **UIスタイル** | **おすすめケース** |
| --- | --- | --- |
| **WinForms** | 従来のWindows UI | 初心者向け、構築が簡単 |
| **WPF** | モダンで柔軟なUI | デザイン重視・MVVM可能 |

**✅ WinForms版：GUIテンプレート（Visual Studio）**

**🔧 ステップ**

1. Visual Studio を開く
2. 新規プロジェクト → 「Windows フォーム アプリケーション (.NET 6/7)」を選択
3. 名前：PdfOcrSplitterApp
4. 「フォームデザイナ」で以下を配置：
   * Button：「PDFを読み込む」
   * TextBox：「ログ表示用」（Multiline）
   * ProgressBar（任意）

**🧩 処理連携コード（例）**

private void btnProcess\_Click(object sender, EventArgs e)

{

string pdfPath = @"C:\path\to\orders.pdf";

string outputDir = @"C:\output";

// 非同期で処理実行

Task.Run(() =>

{

var processor = new PdfProcessor(); // ← OCR & DB処理クラス

var result = processor.ProcessPdf(pdfPath, outputDir);

this.Invoke(() => {

txtLog.AppendText("完了: " + result + "\n");

});

});

}

PdfProcessor クラスには先ほどのOCR・DB登録処理を移植してください。

**✅ WPF版：GUIテンプレート（XAML + MVVM）**

WPFならUIとロジックをきれいに分離できます。

**プロジェクト作成**

* Visual Studio → WPF App (.NET 6/7)
* MainWindow.xaml に XAML記述：

<StackPanel Margin="10">

<Button Content="PDFを処理" Click="ProcessPdf\_Click" Width="200" Margin="0,10"/>

<TextBox x:Name="LogBox" Height="300" Margin="0,10" TextWrapping="Wrap" AcceptsReturn="True" VerticalScrollBarVisibility="Auto"/>

</StackPanel>

**C#コードビハインド（MainWindow.xaml.cs）**

private void ProcessPdf\_Click(object sender, RoutedEventArgs e)

{

Task.Run(() =>

{

var result = PdfProcessor.ProcessPdf(@"C:\input\orders.pdf", @"C:\output");

Dispatcher.Invoke(() =>

{

LogBox.AppendText(result + "\n");

});

});

}

**✅ 配布方法（WinForms/WPF共通）**

* Release ビルドで bin\Release\netX\ に .exe が生成されます
* 必要DLL込みで zip 化すれば他PCでも動作します（.NET 6/7がインストールされていれば）

**📦 オプション（希望あれば）**

* .exe 配布用インストーラー作成（Inno Setup / NSIS / Advanced Installer）
* アイコン付け (pyinstaller --icon=app.ico / C#のプロパティ設定)
* ドラッグ＆ドロップ対応（ファイルパスを取得して処理）

**🎯 まとめ**

| **要素** | **Python (.exe)** | **C# WinForms/WPF** |
| --- | --- | --- |
| 実行形式 | 単体 .exe（PyInstaller） | 単体 .exe（Visual Studio） |
| UIあり | △（Tkinter使えば可能） | ◎（標準でGUI） |
| スケジューラ対応 | ◎（CLIベース） | ◎（.exeを呼び出せばOK） |
| 配布のしやすさ | ◎（PyInstaller一括） | ◎（zipまたはインストーラー） |

どちらの方法についても、**テンプレートZIPの用意やカスタム実装**が可能です。  
必要であれば、どちらかのフルプロジェクト雛形をご用意します。ご希望はありますか？