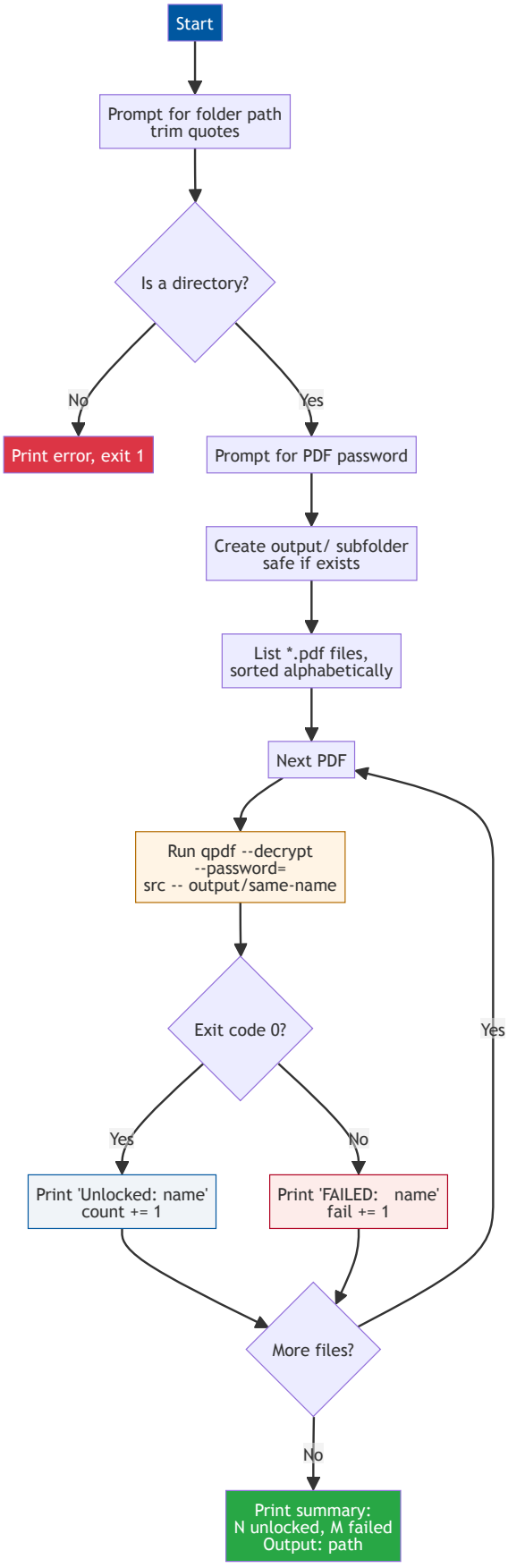
Unlock PDFs Tool

A command-line tool that takes a folder of password-protected PDFs and writes unlocked copies (same filename) into an `output/` subfolder. Uses `qpdf` under the hood — fast, preserves the original file layout exactly.

How It Works



Timing: Tool vs Manual

Per PDF, end-to-end:

Step	Tool	Manual (Preview / Acrobat)
Open the PDF		~3 s
Enter the password		~5 s
File → Export as PDF (or Print → Save as PDF)		~5 s
Save dialog: pick folder, confirm filename		~8 s
Close the file		~2 s
qpdf `--decrypt` per file	~0.1–0.3 s	
Total per file	**< 1 s**	**~25 s**

For a batch (e.g. 20 PDFs), the tool reuses the password once and loops through every file, so the whole run is typically **~5 s** — vs about **8 min** of clicking through dialogs manually.

Output quality is identical to manually doing "open with password → File → Export as PDF" — `qpdf --decrypt` only rewrites the encryption dictionary, so the unlocked file is byte-level equivalent to the original minus the lock (text selection, bookmarks, form fields, metadata, page sizes all preserved).

Output Files

For each input PDF, one unlocked PDF with the **same filename** is written to `<input-folder>/output/`.

```
<input-folder>/
├─ bank_statement_jan.pdf   <- still encrypted (untouched)
├─ bank_statement_feb.pdf
└─ output/
   ├─ bank_statement_jan.pdf   <- unlocked copy
   └─ bank_statement_feb.pdf
```

The tool never modifies the source files — you can re-run it safely.

How Each PDF Is Unlocked

1. Prompt for the **folder path** (trims surrounding quotes, validates it's a directory).
2. Prompt for the **password** (typed into stdin).
3. Create an `output/` subfolder inside the input folder (safe if it already exists).
4. Iterate every `*.pdf` file in the folder, sorted alphabetically.
5. For each file, run:

```
qpdf --decrypt --password=<pwd> <src> <output/same-name>
```

6. Capture `qpdf`'s exit code:
   - `0` → print `Unlocked: <name>`, increment success counter.
   - non-zero → print `FAILED: <name>`, increment failure counter (wrong password, corrupt PDF, etc.).
7. After the loop, print a summary: `N unlocked, M failed` and the output folder path.

Requirements

- Python 3.6+ (standard library only — no pip installs needed)
- `qpdf` on `PATH`
  - macOS: `brew install qpdf`
  - Ubuntu/Debian: `sudo apt install qpdf`
  - Windows: download from the qpdf releases page
- One shared password that opens every PDF in the folder (mixed passwords are not supported in a single run)

Usage

```
python unlock-pdfs.py
```

```
Enter folder path: ~/Downloads/locked-statements
Enter PDF password: ••••••••
Unlocked: statement_jan.pdf
Unlocked: statement_feb.pdf
FAILED:   statement_mar.pdf
Unlocked: statement_apr.pdf

Done. 3 unlocked, 1 failed.
Output: /Users/you/Downloads/locked-statements/output
```

Tips:

- You can drag-and-drop a folder from Finder into the terminal to paste its path — the script strips surrounding quotes.
- `FAILED` typically means the password is wrong for that specific PDF (or the file is corrupt). Run again with the correct password for just the failures.

Notes

- The password is read via plain `input()`, so it is visible on screen as you type. If that matters for your workflow, switch the prompt to `getpass.getpass()` (the `getpass` module is already imported).
- `qpdf --decrypt` strips the encryption layer but keeps any other PDF features (signatures, redactions, form data) intact.
- Failed files remain in the source folder; nothing is written to `output/` for them.