

Python Script to Bulk Rename Tax Documents

Format: TYPE - FY - FolderName - OldFileName

Example: Client Name_AIS_1.xlsm → AIS - FY26 - Client Name - Client Name_AIS_1.xlsm

Run it

```
python rename_tax_files.py "C:\Path\To\Root Folder" # preview only
python rename_tax_files.py "C:\Path\To\Root Folder" --apply # actually
rename
```

- Requires Python 3.10+.
- Root folder → subfolders (client names) → files.
- Only files directly inside those subfolders are renamed.

Type detection (first match wins)

- FolderName_AIS → **AIS**
- 26as → **26AS**
- ais → **AIS**
- tis → **TIS**
- intimation → **143**
- report → **AIS**

No match → file is skipped (add `--include-unknown` to rename it as **UNKNOWN** instead).

FY detection

- 2026-27 in filename → FY26 (auto-works for any year)
- No year in that file's own name → borrows the first FY found among other files in the same subfolder
- Still none → FY-Unknown

Notes

- Duplicate new names get (1), (2), etc. - nothing is overwritten
- Nothing changes on disk unless you pass `--apply`
- Back up the folder before your first real `--apply` run

[rename_tax_files.py](#)

```

"""
PlanMyTax - Bulk Tax Document Renamer
-----
Walks through a root folder's subfolders and renames files based on
detected document type and financial year, using the format:

    TYPE - FY - FolderName - OldFileName

Detection rules (checked in this order per file):
    - filename contains "26as"                -> 26AS
    - filename contains "<foldername>_ais"      -> AIS
    - filename contains "ais"                  -> AIS
    - filename contains "tis"                  -> TIS
    - filename contains "intimation"           -> 143
    - filename contains "report"               -> AIS

Financial year:
    - Any "20XX-YY" pattern (e.g. 2026-27) in the filename becomes
      "FY" + last two digits of the first year (2026-27 -> FY26).

Usage:
    python rename_tax_files.py "C:\\path\\to\\root_folder"          #
dry run (preview only)
    python rename_tax_files.py "C:\\path\\to\\root_folder" --apply  #
actually rename

"""

import re
import argparse
from pathlib import Path

def detect_fy(filename: str) -> str | None:
    """Find a 20XX-YY pattern and return it as FY + last 2 digits of
    the first year."""
    match = re.search(r"(20\d{2})-(\d{2})", filename)
    if match:
        return f"FY{match.group(1)[-2:]}"
    return None

def detect_type(filename: str, folder_name: str) -> str | None:
    """Determine document type based on keywords in the filename."""
    lower = filename.lower()
    folder_lower = folder_name.lower()

    # Explicit "FolderName_AIS" pattern takes priority
    if f"{folder_lower}_ais" in lower:
        return "AIS"

```

```
    if "26as" in lower:
        return "26AS"
    if "ais" in lower:
        return "AIS"
    if "tis" in lower:
        return "TIS"
    if "intimation" in lower:
        return "143"
    if "report" in lower:
        return "AIS"
    return None

def build_new_name(doc_type: str, fy: str, folder_name: str,
original_name: str) -> str:
    return f"{doc_type} - {fy} - {folder_name} - {original_name}"

def unique_path(target: Path) -> Path:
    """Avoid overwriting an existing file by appending (1), (2), ...
before the extension."""
    if not target.exists():
        return target

    stem, ext = target.stem, target.suffix
    counter = 1
    while True:
        candidate = target.with_name(f"{stem} ({counter}){ext}")
        if not candidate.exists():
            return candidate
        counter += 1

def rename_files(root_dir: str, dry_run: bool = True, include_unknown:
bool = False):
    root_path = Path(root_dir)
    if not root_path.is_dir():
        raise NotADirectoryError(f"Not a valid folder: {root_dir}")

    renamed = []
    skipped = []

    for folder in sorted(p for p in root_path.iterdir() if p.is_dir()):
        folder_name = folder.name
        files_in_folder = sorted(p for p in folder.iterdir() if
p.is_file())

        # Folder-level fallback FY: the first FY found among any file
in this folder.
        folder_fallback_fy = None
        for f in files_in_folder:
```

```

        fy_found = detect_fy(f.name)
        if fy_found:
            folder_fallback_fy = fy_found
            break

    for file in files_in_folder:
        original_name = file.name

        doc_type = detect_type(original_name, folder_name)
        fy = detect_fy(original_name)

        if doc_type is None and not include_unknown:
            skipped.append((file, "no matching document type
keyword found"))
            continue

        doc_type = doc_type or "UNKNOWN"
        fy = fy or folder_fallback_fy or "FY-Unknown"

        new_name = build_new_name(doc_type, fy, folder_name,
original_name)
        new_path = unique_path(file.parent / new_name)

        renamed.append((file, new_path))
        if not dry_run:
            file.rename(new_path)

    return renamed, skipped

def main():
    parser = argparse.ArgumentParser(description="Rename tax document
files in subfolders")
    parser.add_argument("folder", help="Root folder containing
subfolders of files")
    parser.add_argument("--apply", action="store_true", help="Actually
rename files (default is dry-run preview)")
    parser.add_argument("--include-unknown", action="store_true",
                        help="Also rename files where no document type
keyword was matched (tagged UNKNOWN)")
    args = parser.parse_args()

    dry_run = not args.apply

    renamed, skipped = rename_files(args.folder, dry_run=dry_run,
include_unknown=args.include_unknown)

    mode = "DRY RUN (no files changed)" if dry_run else "APPLIED"
    print(f"\n=== {mode} ===\n")

```

```
print(f"{len(renamed)} file(s) to rename:\n")
for old, new in renamed:
    print(f"    {old.name}")
    print(f"    -> {new.name}\n")

if skipped:
    print(f"\n{len(skipped)} file(s) skipped:\n")
    for path, reason in skipped:
        print(f"    {path} ({reason})")

if dry_run:
    print("\nThis was a preview only. Re-run with --apply to
actually rename the files.")

if __name__ == "__main__":
    main()
```

From:

<https://planmytax.com/> - **PlanMyTax**

Permanent link:

<https://planmytax.com/doku.php?id=python-script-to-bulk-rename-tax-documents>

Last update: **2026/07/13 12:25**

