Python Script to Compare Folders and Move Files

Compares the subfolder names inside two parent folders. For every subfolder name that exists in BOTH parents, all files inside the subfolder under SOURCE_PARENT are moved into the matching subfolder under DEST_PARENT.

Usage:

```
  python move_matching_folders.py "C:\path\to\Source" "C:\path\to\Dest"
```

Optional flags:

1. -dry-run Show what WOULD be moved, without moving anything.
2. -recursive Also move files from sub-subfolders (keeps folder structure).
3. -overwrite Overwrite a file in destination if same filename exists.

(default: skip and keep both — renames with a suffix)

Example:

```
  python move_matching_folders.py "D:\Old\Clients" "D:\New\Clients" --dry-
run
```

---

move_matching_folders.py

```
r"""
move_matching_folders.py
-------------------------
Compares the subfolder names inside two parent folders. For every
subfolder
name that exists in BOTH parents, all files inside the subfolder under
SOURCE_PARENT are moved into the matching subfolder under DEST_PARENT.

Usage:
    python move_matching_folders.py "C:\path\to\Source"
"C:\path\to\Dest"

Optional flags:
    --dry-run     Show what WOULD be moved, without moving anything.
    --recursive   Also move files from sub-subfolders (keeps folder
structure).
    --overwrite   Overwrite a file in destination if same filename
exists.
                  (default: skip and keep both — renames with a suffix)

Example:
    python move_matching_folders.py "D:\Old\Clients" "D:\New\Clients" -
-dry-run
```

```
"""

import argparse
import shutil
import sys
from pathlib import Path


def unique_destination(dest_path: Path) -> Path:
    """If dest_path already exists, append (1), (2)... to avoid
overwrite."""
    if not dest_path.exists():
        return dest_path
    stem, suffix, parent = dest_path.stem, dest_path.suffix,
dest_path.parent
    counter = 1
    while True:
        candidate = parent / f"{stem} ({counter}){suffix}"
        if not candidate.exists():
            return candidate
        counter += 1


def move_files(src_folder: Path, dest_folder: Path, recursive: bool,
               overwrite: bool, dry_run: bool) -> tuple[int, int]:
    """Move all files from src_folder into dest_folder. Returns (moved,
skipped)."""
    moved = skipped = 0

    pattern = "**/*" if recursive else "*"
    for item in src_folder.glob(pattern):
        if item.is_dir():
            continue  # only move files, not subfolders themselves

        if recursive:
            # preserve relative sub-path under dest_folder
            rel_path = item.relative_to(src_folder)
            target = dest_folder / rel_path
            target.parent.mkdir(parents=True, exist_ok=True)
        else:
            target = dest_folder / item.name

        if target.exists() and not overwrite:
            target = unique_destination(target)

        if dry_run:
            print(f"  [DRY RUN] Would move: {item}  ->  {target}")
            moved += 1
            continue
```

```
        try:
            shutil.move(str(item), str(target))
            print(f"  Moved: {item.name}  ->  {target}")
            moved += 1
        except Exception as e:
            print(f"  ERROR moving {item}: {e}")
            skipped += 1

    return moved, skipped


def main():
    parser = argparse.ArgumentParser(
        description="Move files between matching-named subfolders of
two parent folders."
    )
    parser.add_argument("folder1", help="Path to the SOURCE parent
folder")
    parser.add_argument("folder2", help="Path to the DESTINATION parent
folder")
    parser.add_argument("--dry-run", action="store_true",
                        help="Preview actions without moving any
files")
    parser.add_argument("--recursive", action="store_true",
                        help="Include files in sub-subfolders too")
    parser.add_argument("--overwrite", action="store_true",
                        help="Overwrite existing files in
destination")
    args = parser.parse_args()

    folder1 = Path(args.folder1)
    folder2 = Path(args.folder2)

    if not folder1.is_dir():
        sys.exit(f"Error: '{folder1}' is not a valid folder.")
    if not folder2.is_dir():
        sys.exit(f"Error: '{folder2}' is not a valid folder.")

    # Get subfolder names (not full paths) under each parent
    sub1 = {p.name: p for p in folder1.iterdir() if p.is_dir()}
    sub2 = {p.name: p for p in folder2.iterdir() if p.is_dir()}

    matching_names = sorted(set(sub1.keys()) & set(sub2.keys()))

    if not matching_names:
        print("No matching subfolder names found. Nothing to do.")
        return

    print(f"Found {len(matching_names)} matching subfolder name(s):")
    for name in matching_names:
```

```
            print(f"  - {name}")
        print()

        total_moved = total_skipped = 0

        for name in matching_names:
            src = sub1[name]
            dest = sub2[name]
            print(f"Processing '{name}':")
            moved, skipped = move_files(src, dest, args.recursive,
args.overwrite, args.dry_run)
            total_moved += moved
            total_skipped += skipped
            print()

        action = "Would move" if args.dry_run else "Moved"
        print(f"Done. {action} {total_moved} file(s). Skipped/errors:
{total_skipped}.")


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