

```

1 import os
2 import argparse
3 from datetime import datetime
4
5 # --- EXISTING FUNCTIONS (sequential_rename, replace_text_rename) OMITTED FOR BREVITY ---
6 # ... (The existing sequential_rename and replace_text_rename functions go here)
7 # -----
8
9 def date_rename(folder_path, date_format="%Y%m%d_%H%M%S", prefix="", suffix="", use_modification_time=False):
10     """Renames files using their creation or last modification date/time."""
11     print(f"Starting date-based rename in: {folder_path}")
12
13     # Decide which timestamp to use
14     timestamp_func = os.path.getmtime if use_modification_time else os.path.getctime
15     time_source = "modification" if use_modification_time else "creation"
16
17     print(f"Using file {time_source} time.")
18
19     files = sorted(os.listdir(folder_path))
20
21     for filename in files:
22         old_path = os.path.join(folder_path, filename)
23
24         # Skip directories
25         if os.path.isdir(old_path):
26             continue
27
28         base_name, ext = os.path.splitext(filename)
29
30         try:
31             # 1. Get the timestamp (in seconds since the epoch)
32             timestamp_sec = timestamp_func(old_path)
33
34             # 2. Convert timestamp to a datetime object
35             file_date = datetime.fromtimestamp(timestamp_sec)
36
37             # 3. Format the date into a string

```

```

38 date_str = file_date.strftime(date_format)
39
40 # 4. Construct the new filename: prefix_YYYYMMDD_HHMMSS_suffix.ext
41 parts = [prefix, date_str, suffix]
42
43 # Filter out any empty strings and join the remaining parts
44 # We use '_' as a separator if both prefix and date are present, etc.
45 name_parts = [p for p in parts if p]
46 new_base_name = "_".join(name_parts)
47
48 new_filename = f"{new_base_name}{ext}"
49 new_path = os.path.join(folder_path, new_filename)
50
51 # Prevent overwrites
52 if os.path.exists(new_path):
53     # Append a counter to the suffix if a conflict occurs (e.g., if files were created in the same second)
54     i = 1
55     while os.path.exists(new_path):
56         conflict_suffix = f"{suffix}_{i}" if suffix else f"_{i}"
57         new_filename = f"{prefix}_{date_str}{conflict_suffix}{ext}"
58         new_path = os.path.join(folder_path, new_filename)
59         i += 1
60     print(f"⚠ Conflict resolved. Renamed: {filename} -> {new_filename}")
61
62 os.rename(old_path, new_path)
63 print(f"Renamed: {filename} -> {new_filename}")
64
65 except Exception as e:
66     print(f"❌ ERROR renaming {filename}: {e}")
67
68 print("\n✅ Date-based renaming complete.")
69
70 # -----
71

```

```

72 def main():
73     """Main function to handle command-line arguments."""
74     parser = argparse.ArgumentParser(
75         description="A versatile file-naming automation tool (now with humanized date features).",
76         epilog="Example: python rename_tool.py date c:\\Users\\...\\Photos -f %Y-%m-%d_ -p IMG -m"
77     )
78
79     subparsers = parser.add_subparsers(dest="mode", required=True, help="Renaming mode")
80
81     # --- Sequential Renaming Mode (seq) ---
82     # ... (Sequential parser definition)
83
84     # --- Text Replacement Mode (rep) ---
85     # ... (Replacement parser definition)
86
87     # --- NEW: Date-Based Renaming Mode (date) ---
88     parser_date = subparsers.add_parser('date', help='Date-based renaming mode')
89     parser_date.add_argument('folder_path', help='The path to the folder containing the files.')
90     parser_date.add_argument('-f', '--date_format', default="%Y%m%d_%H%M%S",
91                             help='Python strftime format string (e.g., %Y-%m-%d). Default: %Y%m%d_%H%M%S.')
92     parser_date.add_argument('-p', '--prefix', default="", help='Optional: Prefix to add before the date (e.g., "Photo").')
93     parser_date.add_argument('-s', '--suffix', default="", help='Optional: Suffix to add after the date.')
94     parser_date.add_argument('-m', '--use_modification_time', action='store_true',
95                             help='Use file last modification time (mtime) instead of creation time (ctime).')
96
97     args = parser.parse_args()
98
99     # Dispatch to the appropriate function
100     if args.mode == 'seq':
101         # sequential_rename(...)
102         pass # Placeholder for existing call
103     elif args.mode == 'rep':
104         # replace_text_rename(...)

```

```
104         # replace_text_rename(...)
105         pass # Placeholder for existing call
106     elif args.mode == 'date':
107         date_rename(args.folder_path, args.date_format, args.prefix, args.suffix, args.use_modification_time)
108
109     if __name__ == "__main__":
110         main()
```