Documentação do Projeto de Arquivamento e Comparação de Arquivos

Este projeto consiste em um conjunto de ferramentas para organização, arquivamento e comparação de arquivos, com foco especial em mensagens de email e documentos.

Ferramentas Disponíveis

1. Arquiva Email GUI (`arquiva_email_gui.py`)

Ferramenta com interface gráfica para organização de emails e outros arquivos em estrutura cronológica. Ver documentação detalhada.

2. Arquiva Email (`arquiva_email.py`)

Objetivos e Funcionalidades

- Versão de linha de comando para organização de arquivos `.eml` e outros tipos
- Organiza arquivos em estrutura cronológica (Ano/Ano-Mês)
- Extrai datas de cabeçalhos de emails para arquivos `.eml`
- Utiliza datas de modificação para outros tipos de arquivo Ver documentação detalhada.

Diferenças da versão GUI

- Não possui interface gráfica
- Configurado para uso via linha de comando ou importação em outros scripts
- Mantém a mesma lógica de processamento de arquivos

3. Arquiva Raiz (`arquiva_raiz.py`)

Objetivos e Funcionalidades

- Processa arquivos na pasta raiz e suas subpastas
- Move arquivos de subpastas para a raiz, sanitizando e renomeando conforme necessário
- Opção para remover pastas vazias após a movimentação dos arquivos
- Mantém contadores de arquivos processados, renomeados, movidos e erros Ver documentação detalhada.

Características Específicas

- Método `process_files_in_root()` para processar arquivos em toda a estrutura
- Método `remove_empty_folders()` para limpar pastas vazias após processamento
- Suporte a pastas excluídas que não serão processadas

4. Arquiva Subpastas (`arquiva_subpastas.py`)

Objetivos e Funcionalidades

- Versão especializada para processamento de arquivos em subpastas
- Mantém a estrutura de pastas original, mas organiza os arquivos dentro delas
- Implementa lógica avançada para tratamento de datas em emails
- Contabiliza pastas criadas durante o processamento Ver documentação detalhada.

Características Específicas

- Método `move_file_to_archive()` aprimorado para decisão entre mover, renomear ou ignorar
- Tratamento mais robusto de datas em emails com múltiplos formatos
- Contadores para pastas criadas durante o processamento

5. Comparador de Pastas (`pastas_diff.py`)

Objetivos e Funcionalidades

- Compara o conteúdo de duas pastas e gera relatório detalhado das diferenças
- Identifica arquivos presentes em apenas uma das pastas
- Detecta arquivos com mesmo nome mas conteúdo diferente
- Salva relatório de comparação em formato texto Ver documentação detalhada.

Características Específicas

- Interface para seleção das duas pastas a serem comparadas
- Método `compare_folders()` para análise detalhada das diferenças
- Sistema de logging para registro de erros durante a comparação
- Método `save_report()` para salvar o relatório em arquivo de texto

6. Verificador de Sequência Numérica e Unificador de Relatórios (`relatorio_mensagens.py`)

Objetivos e Funcionalidades

- **Análise de Sequência Numérica:**
  - Verifica a integridade de uma sequência numérica esperada em nomes de arquivos dentro de uma pasta.
  - Identifica números faltantes na sequência.
  - Lista arquivos com números fora do intervalo esperado.
  - Lista arquivos que não começam com um número.
- **Unificação de Relatórios:**
  - Opcionalmente, unifica múltiplos relatórios de verificação (arquivos `.txt`) de uma pasta em um único relatório HTML consolidado.
  - Os relatórios `.txt` individuais são salvos na pasta mãe da pasta analisada.
  - O relatório HTML unificado também é salvo na pasta mãe.
  - Após a unificação bem-sucedida, os arquivos `.txt` originais que foram combinados são excluídos.
- **Interface Gráfica:**
  - Ver documentação detalhada.
  - Utiliza Tkinter para uma interface amigável.
  - Permite selecionar a pasta para análise.

    - Permite definir o intervalo numérico (inicial e final).
    - Inclui uma opção (checkbox) para habilitar a unificação de relatórios.

Características Específicas

- **Geração de Relatório Individual (`.txt`):**
    - Salvo na pasta mãe da pasta analisada.
    - Nome do arquivo: `relatorio_verificacao_[nome_da_pasta_analisada].txt`.
    - Conteúdo: Detalhes da pasta, intervalo, números faltantes, números fora do intervalo, arquivos sem numeração.
- **Geração de Relatório Unificado (`.html`):**
    - Salvo na pasta mãe da pasta analisada.
    - Nome do arquivo: `Relatorio_Unificado_[nome_da_pasta_mae].html`.
    - Conteúdo: Combinação dos relatórios `.txt` formatados como HTML (Markdown).
    - Estilização básica para legibilidade.
- **Logging:**
    - Logs detalhados do processo de unificação são salvos em uma subpasta `LOGS_UNIFICADOR` dentro da pasta mãe.
- **Feedback ao Usuário:**
    - Mensagens de informação e erro via `messagebox`.

Modo de Usar (Resumido)

1. Execute `relatorio_mensagens.py`.
2. Na GUI, selecione a pasta a ser analisada.
3. Defina o "Número Inicial" e "Número Final" da sequência.
4. Marque "Unificar relatórios..." se desejar combinar relatórios `.txt` existentes na pasta mãe.
5. Clique em "Verificar Arquivos".
6. Os relatórios serão gerados na pasta mãe da pasta analisada.

Detalhes Técnicos Comuns

Sistema de Logging

- Todos os scripts utilizam o módulo `logging` para registro de erros
- Para as ferramentas de arquivamento e comparação, os logs são salvos em pasta "ERROS".
- Para o `relatorio_mensagens.py`, os logs da unificação são salvos em `LOGS_UNIFICADOR` na pasta mãe da analisada.
- Arquivos de log nomeados com timestamp para identificação única

Tratamento de Nomes de Arquivo

- Sanitização para remover caracteres inválidos e prefixos desnecessários
- Truncamento para respeitar limites de tamanho de caminho
- Resolução de conflitos para evitar sobrescrita de arquivos

Processamento de Datas

- Extração de datas de cabeçalhos de email para arquivos `.eml`

- Uso de data de modificação para outros tipos de arquivo
- Múltiplas estratégias de fallback para garantir processamento mesmo com formatos não padrão

Requisitos Técnicos

- Python 3.x (recomendado 3.6 ou superior)
- Bibliotecas padrão: `os`, `shutil`, `email`, `logging`, `re`, `datetime`, `tkinter`, `pathlib`
- Bibliotecas externas:
  - `markdown` (para `relatorio_mensagens.py`): Instalar com `pip install markdown`

Dicas de Uso

- Escolha a ferramenta adequada para sua necessidade específica:
  - `arquiva_email_gui.py` para organização simples com interface gráfica
  - `arquiva_raiz.py` para consolidar arquivos de subpastas na raiz
  - `arquiva_subpastas.py` para manter a estrutura de pastas original
  - `pastas_diff.py` para comparar o conteúdo de duas pastas
  - `relatorio_mensagens.py` para verificar sequências numéricas e unificar relatórios de verificação.
- Faça backup dos arquivos antes de executar qualquer ferramenta de movimentação
- Verifique os logs após o processamento para identificar possíveis problemas