```
NAME
    process_messages - Process the files in the messages folder.

DESCRIPTION
    Synopsis:
        process_messages.py

    Description:
        Process the files in the messages folder and any attachments.
        For each message file, read the contents and extract the commands
        and data. Generate from this information the trigger files.
        If the command needs any attachments, add that information to
        the trigger file.
        If the command is for a system admin task, generate a trigger file
        for the system admin task.
        All files are processed first and when finished, the trigger files
        will start the necessary scripts.

    Arguments:
        None

    Return:
        0 -- Normal exit

    Run info:
        This program must be run by the process_messages.sh script only.

    See also:
        dotenv -- dotenv file for configuration information.
        process_messages.sh -- Script that executes this program.
        html2text -- Utility to convert HTML to text.
        signage.conf -- Configuration data providing environment variables
                        when running this inside a BASH script.

    Version:
       1.0

    Copyright (C) 2023 Gert Bakker, Coleshill,  UK

    This file is part of the <Digital Signage> project.

    Digital Signage is free software: you can redistribute it and/or modify
    it under the terms of the GNU General Public License as published by the
    Free Software Foundation, version 3.

    Digital Signage is distributed in the hope that it will be useful, but
    WITHOUT ANY WARRANTY; without even the implied warranty of
    MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General
    Public License for more details.

    You should have received a copy of the GNU General Public License
    along with this program. If not, see <https://www.gnu.org/licenses/>.

FUNCTIONS
    get_filename_ext(filepath)
        Get the filename and it’s extension from a given filename.

        Extracts from the given filepath, the filename and the extension and
        return them as a two elements.

        Arguments:
            filepath -- The file path name from which to extract the
                        filename and extension (.extn)

        Return:
            filename, fileext -- Return tuple with filename and extension

    initialise()
```

```
        Initialise this program.

        From the dotenv set of variables, get the logging level.
        Set the logging level depending on the DEBUG variable from the
        dotenv secrets file.

        Arguments:
            None

        Return:
            None

    main()
        Main control of the program.

        Initialise the program setting global variable values.
        Process the message files and return a list of message files that
        can be deleted. This process creates one or more 'command' files
        that are used by other scripts to perform specific functions.
        If there are any message files to be deleted, remove them.

        Arguments:
            None

        Return:
            0 -- OK

    process_data(text_data, mesg_prefix)
        Process the data taken from a file in the messages folder.

        The data consists of the lines of text taken from the email message.
        Each line is considered to be an instruction with the first word
        being the command and the rest of the line the data to be used.
        The command is tested in lowercase to avoid any upper and
        lower case problems. Appropriate trigger files are created depending
        on the command and data provided.

        Arguments:
            text_data -- The text data (multiple lines) to be processed.
            mesg_prefix -- Prefix of the message file being processed.

        Return:
            None

    process_files()
        Process all files in the "messages" folder.

        If no files in the message folder, exit this function with empty
        list of files to remove. This function only deals with message files
        with the extension ".html" and ".txt". Any files not having the
        correct file extension are ignored and are deleted.
        HTML files are converted to text using the "html2text" utility.
        If a .txt file has a corresponding .html file, it is ignored,
        otherwise the file contents is used.
        The text data is then processed by the "process_data" function.

        Arguments:
            None

        Return
            remove_list -- The list of message files that are to be deleted.

    remove_files(remove_list)
        Remove all the files that are in the removal list given.

        Arguments:
            remove_list -- The list of files to be removed.
        Return:
            None

    update_debug_var(param_list)
        Update the DEBUG variable in the dotenv file.

        Arguments:
            param_list -- List of words that were on the same line as the
                          DEBUG command.

        Return:
            None.

    update_email_auth(email_list)
        Update the authorised email list in the dotenv file.

        This given list will be converted to a list variable. The email
        addresses are checked for having at least one @ sign it them.
        If not correct, then the variable is not updated.

        Arguments:
            email_list -- list of words that were on the same line as the
                          EMAIL_AUTH command.

        Return:
            None

    write_adm_file(parm_cmnd, parm_data, parm_atch)
        Write the date to the System Admin management trigger file.

        Arguments:
            parm_cmnd -- Command to write to the file
            parm_data -- Data to write to the file
            parm_atch -- Attachment file name to write to the file

        Return:
            None

    write_google_data(rest_of_list)
        Write the running_next data for a Google slides show.

        Using the data in the "rest_of_list" parameter, check first that it
        is for a Google Slide show by ensuring it has the right DNS name
        in the URL. If not, error message and remove any existing run_next
        file.
        Write the two environment variables to the next presentation to run.
        The variables are RUN_CMND and RUN_DATA, where RUN_CMND is set to
        'google'. Add the command 'export' to the variables so that they can
        be used in a Python or other program,

        Arguments:
            rest_of_list -- The remainder of the line from the email data

        Return:
            None

    write_manage_next(p_fileprefix, p_cmnd, p_data)
        Write the System Admin manage_next file.

        The p_data is unravelled using the csv module methods. This will
        result in a list within a list. It contains the command and data
        provided.
        The attachment files are checked and the one with the filename that
        starts with the file prefix parameter is selected. If found, that is
        used in the manage_next trigger file.

        Arguments:
            p_fileprefix -- Filename prefix used for finding attachment
            p_cmnd -- Management command
            p_data -- Data for the management command

        Return:
            None

    write_next_file(parm_cmnd, parm_data)
        Write the command and data to the running next trigger file.

        Arguments:
            parm_cmnd -- Command to write to the file
            parm_data -- Data to write to the file

        Return:
            None

    write_pp_impress_data(fileprefix)
        Write a PowerPoint or Impress running_next file.

        Get a list of attachments from the attachment folder and search
        this list for a file that starts with the file same prefix as the
        message and ends with the suffix for PowerPoint or Impress.

        If not found, exit the function with a False value. Ensure that
        the message filename goes into the list of messages to delete.
        Log this condition.

        Arguments:
            fileprefix -- The string that should be at the start of the
                          attachment file.

        Return:
            None

    write_run_file(p_filedata)
        Get the RUN filename and copy from permanent file to running_next

        Using the data given with the RUN command, if there is any data and
        if the data is a file name in the permanent folder, copy the
        permanent file to the running-next file. This will trigger the start
        of that presentation.

        Arguments:
            p_filedata -- Filename data from the RUN command in the message

        Return:
            None
```