**FamilyTreeReporter**

A utility to produce an ancestry report, in Microsoft Word (.docx) format, from a GEDCOM format file.

**Overview**

This utility allows the user to produce an ancestry report, in Microsoft Word (.docx) format, from a GEDCOM format file.

The report shows each family on a new page, starting with the family specified as a parameter.

Each page shows a family, starting with the family reference code (‘F’ followed by the family number, which is a sequential number starting at 1, i.e. F1, F2 …), then the husband, wife and children with birth, baptism, marriage, death and burial places and dates. Where a parent appears elsewhere in the report as a child, or vice versa, the name is followed by the family reference code, and has a hyperlink to the page. Where parents have associates document images, these follow the family page.

Each family page is followed by further pages showing the husband's family and ancestors, where known, then the wife's family and ancestors, where known.

Modules

* FamilyTreeReporter.pyw

The main module. This module presents the user with a Windows user interface, allowing them to edit parameters, process a GEDCOM format file or run the report.

* params.py

This module presents the user with a user interface to edit parameters. It contains the parameter variables used by other modules and the following subroutines:

*get\_params* reads parameter values from params.txt.

*save\_params* writes parameter values to params.txt.

*edit\_params* presents the user with a user interface to edit parameters.

Parameters are:

|  |  |
| --- | --- |
| GED File | The name of the GEDCOM format file to be processed. The file should be in the location where the utility runs. |
| Page Size | A4 or A5 |
| Initial Family | The first family in the report. To obtain this, first process a GEDCOM format file. This will produce Individuals.txt, Families.txt and Children.txt containing lists of individuals, families and children in the GEDCOM file. First, in Individuals.txt, find the IDs of the husband and wife for the first family in the report, then, in Families.txt, find the ID of the family. Note: each time a report is produced, the utility produces FamiliesToReport.txt containing a list of the family ID of the families in the report. Setting the Initial Family parameter to 0 will cause the utility to produce the report based on these families. FamiliesToReport.txt can be manually edited to remove unwanted families from the report, or to add additional families. |
| Title Page Required | Set to Y if a title page is required. |
| Contents Required | Set to Y if contents pages are required. |
| Index Required | Set to Y if index pages are required. |
| Document Images Required | Set to Y if document images are required |
| Website Path | Set to the full path of the folder containing an ‘images’ folder containing document images, e.g. if images are in C:\website\images, set this parameter to C:\website. Note: each document images must have a filename with the name and date of birth of an individual in the report, e.g. where for an individual called Mary Jane Swales, born in 1885, the Birth Index Register document image filename should be ‘Swales, Mary Jane b. 1885 – BIR.jpg’.  BIR is an image type (see write\_documents.py module for further explanation) |
| Title Line 1 | 1st line of text on title page |
| Title Line 2 | 2nd line of text on title page |
| Title Line 3 | 3rd line of text on title page |
| Title Line 4 | 4th line of text on title page |
| Country to remove 1 | 1st country to remove. Where many place names contain the same ‘home’ country, e.g. England, or United Kingdom, specify that country here to remove it. |
| Country to remove 2 | Specify a 2nd country here to remove it from place names. |
| Person1 | Parameter not used in FamilyTreeReporter. |
| Person2 | Parameter not used in FamilyTreeReporter. |
| Person3 | Parameter not used in FamilyTreeReporter. |
| Person4 | Parameter not used in FamilyTreeReporter. |

* process\_ged\_file.py

This module contains the *process\_ged\_file* subroutine that reads the GEDCOM file specified as a parameter and produces Individuals.txt, Families.txt and Children.txt containing lists of individuals, families and children, with attributes.

* individuals.py

This module contains the ‘individuals’ list and the following subroutines/functions:

*read\_individuals* reads Individuals.txt into the ‘individuals’ list.

write\_individuals writes the ‘individuals’ list to Individuals.txt.

*get\_person\_name* returns the name of the individual (id passed as a parameter) in the form ‘surname, forename(s)’.

*get\_birth\_date* returns the year of birth of the individual (id passed as a parameter) in form ‘b. yyyy’.

*get\_family\_where\_child* returns the id of the family where the individual (id passed as a parameter) is a child.

* families.py

This module contains the ‘families’ list and the following subroutines/functions:

*read\_families* reads Families.txt into the ‘families’ list.

*write\_families* writes the ‘families’ list to Families.txt.

*get\_father\_id* returns the id of the husband in the family where the individual (id passed as a parameter) is a child.

*get\_mother\_id* returns the id of the wife in the family where the individual (id passed as a parameter) is a child.

* children.py

This module contains the ‘children’ list and the following subroutines:

*read\_children* reads Children.txt into the ‘children’ list.

*write\_children* writes the ‘children’ list to Children.txt.

* create\_report.py

This module contains the *create\_report* subroutine that creates Report.docx based on the parameters and information in Individuals.txt, Families.txt and Children.txt.

*create\_report* opens a new document, calls *set\_page\_size* to define page attributes for the page size defined in params, then obtains the initial family id from params then calls *tree\_walk* to obtain a list of ‘families to report’, and writes this list to FamiliesToReport.txt. If the initial family was 0, it reads the existing list of ‘families to report’, from FamiliesToReport.txt and proceeds with them.

*create\_report* then calls *write\_title\_page*, to write the title page, if required, and calls *write\_contents*, to write contents pages, if required. Then for each family in the ‘families to report’ list, it calls *add\_family\_to\_index*, to write the family to the index list, then calls *write\_page*, to write the family pages. After it has written all the family pages, it then calls *write\_index*, to write the index pages, if required, then saves the open document as Report.docx.

*tree\_walk* writes the family (id passed as a parameter) to the list of ‘families to report’. Then, if the family has a husband, it gets the id of the family where the husband was a child and calls *tree\_walk* recursively with this family id. Then, if the family has a wife, if gets the family where the wife was a child and calls *tree\_walk* recursively with this family id.

*add\_family\_to\_index* calls *update\_index* to add the husband to the ‘index’ list, used to print the index at the end of the report, then calls *update\_index* again to add the wife to the list, then for each child in the family it calls *update\_index* again to add the child to the list.

*update\_index* creates an index entry(key) for the individual (id passed as a parameter), using surname, forename(s), year of birth and id, then obtains the family references where the individual is a spouse and where they are a child. It then checks if the individual has already exists in the ‘index’ list. If not, it adds a record to the ‘index’ list, containing the key and links to the pages where the individual is a spouse and child. Otherwise, it updates the existing record with the links.

*write\_family\_page* creates a table (1 row by 5 columns) calls *add\_*bookmark, to writes a bookmark, e.g. [F1]. It then calls *write\_individual* to write the husband details, and again for the wife details. It then writes marriage details and the ‘Children’ heading. Then for each child in the family calls *write\_individual* to write the child details. It then calls *set\_col\_*widths to set the column widths for the family table. It then calls *write\_documents to* writes document associated with the husband and wife. Each time a new row of information is written, a new table row is added. Other text is written using a *write\_text* subroutine to ensure consistent text attributes (font name, font size etc)

*write\_individual* write information relating to an individual (id passed as a parameter). The individual’s type (Husband, Wife, Child), name, ‘Born’ heading are always written. Other headings (Baptism, Death, Burial) and attributes (Date and Place) are only written if that information is available. Name is written using the *write\_link* subroutine, where the individual appears elsewhere in the report. Otherwise name and all other text is written using the *write\_text* subroutine.

*write\_text* writes text to the specified cell and sets font name, normal font size, bold and alignment.

*write\_link* extracts the family reference (bookmark text) from the name and calls *add\_link* to write the text as a hyperlink to the required family page.

*write\_place* removes any text defined as a ‘country to remove’ parameter, then writes the place using the *write\_text* subroutine.

*set\_col\_widths* sets the width of the 5 family page table columns, for each row, to 1.80cm, 5.40cm, 2.40cm, 2.48cm and 4.48cm respectively.

*add\_bookmark* writes the specified bookmark name to the specified paragraph (both passed as parameters).

*add\_link* write the specified text as a hyperlink to the specified family page, to the specified paragraph (all passed as parameters).

*write\_documents* writes documents associated with the specified individual (id passed as a parameter). For each image type it searches the images folder within the website (set as a param) for a files with the correct name, forname(s), birth year and image type. If found, it reads the image, decides whether it should be show in portrait or landscape format, adjusts the image size to fit the page, then writes the image to the open document. It then calls *write\_*footer to write the individual’s names, birth year and image type as a footer. Note: the subroutine scans for all image types in the order they are defined in ImageTypes.txt so as to show them in a useful order, i.e. birth, baptism, marriage, census in date order, death, burial, will, probate.

*write\_footer* write text as a footer to the current section (both specified as parameters)

*write\_title\_page* writes the four title lines specified in params, and the current date, to the open document. Title lines are written using the *write\_title\_text* subroutine.

*write\_title\_text* writes text to the specified cell and sets font name, title font size, bold and alignment.

*write\_contents* creates a table in the open document, then writes the family number and husband and wife names for each family in the list of ‘families to report’ to a table cell. Each record includes a hyperlink to a bookmark identifying the family number.

*write\_index* sorts the ‘index’ list, then writes each item in the index to a table cell. If it’s writing the first item on a page, it creates a new table with 2 columns and the maximum number of rows for the page size, with the ‘Index’ heading. It then determines the row and column of the next available cell, writing all on the left, then all on the right. It them forms the index text from the surname, forename(s) and year of birth, replacing surname with ‘-----’ where it repeats the previous one. It then appends the hyperlinks to the families where the individual is a parent and child, then writes the whole entry to the next available index cell.

*set\_page\_attributes* sets global variables i.e. page height and width, margin sizes, header and footer lengths font sizes, the number of table rows on title and index pages and maximum image height, width and ration for document images, according to the size defined in params.

*set\_section\_attributes* sets section attributes i.e. orientation, page height and width, based on page attributes, and orientation (passed as a parameter). Also sets the section margins, header and footer attributes.

*new\_section* is called each time a new page is started, to create the section and call *set\_section\_attributes*.

* families\_to\_report.py

This module contains the ‘families to report’ list and the following subroutines/functions:

*read\_families\_to\_report* reads FamiliesToReport.txt into the ‘families to report’ list.

*write\_families\_to\_report* writes the ‘families to report’ list to FamiliesToReport.txt.

*get\_family\_number* returns the family number for the family (id passed as a parameter).

* image\_types.py

Contains the ‘image\_types’ list and subroutine read\_image\_types to read ImageTypes.txt into the ‘image\_types’ list

* index.py

Contains the ‘index’ list and subroutine write\_index to write and index entry to the ‘index’ list.