gedcompy Documentation

Release 0.1

Rory McCann

Oct 27, 2017

Contents

CHAPTER 1

Reading GEDCOM files

The function *gedcom.parse()* reads a file, string, or file-like object and returns a *gedcom.GedcomFile*.

gedcom.**parse**(*obj*)

Parse and return this object, if it's a file.

If it's a filename, it calls `parse_filename()`, for file-like objects, `parse_fp`, for strings, calls `parse_string`.

Parameters **obj** – filename, open file-like object or string contents of GEDCOM file

Returns GedcomFile

CHAPTER 2

Writing GEDCOM files

gedcom.GedcomFile.save() saves a *gedcom.GedcomFile* to a specified filename, or file-like object.

GedcomFile.**save**(*fileout*)

Save the contents of this GEDCOM file to specified filename or file-like object.

Parameters **fileout** – Filename or open file-like object to save this to.

Raises **Exception** – if the filename exists

CHAPTER 3

Example Usage

```
>>> import gedcom
>>> gedcomfile = gedcom.parse("myfamilytree.ged")
>>> for person in gedcomfile.individuals:
...     firstname, lastname = person.name
...     print "{0} {1} is in the file".format(firstname, lastname)
```

CHAPTER 4

GEDCOM API

Library for reading and writing GEDCOM files.

https://en.wikipedia.org/wiki/GEDCOM

class gedcom.**Birth**(*level=None*, *tag=None*, *value=None*, *id=None*, *parent_id=None*, *parent=None*, *gedcom_file=None*)
Represents a birth (BIRT).

class gedcom.**Death**(*level=None*, *tag=None*, *value=None*, *id=None*, *parent_id=None*, *parent=None*, *gedcom_file=None*)
Represents a death (DEAT).

class gedcom.**Element**(*level=None*, *tag=None*, *value=None*, *id=None*, *parent_id=None*, *parent=None*, *gedcom_file=None*)
Generic represetation for a GEDCOM element.

Can be used as is, or subclassed for specific functionality.

add_child_element(*child_element*)
Add *child_element* as a child of this.

It sets the `parent` and `parent_id` of *child_element* to this element, but does not set the `level()`. See *`set_levels_downward()`* to correct that.

Parameters **child_element** (`Element`) – The Element you want to add as a child.

gedcom_lines()
Iterator over the encoded lines for this element.

Return type iterator over string

get_by_id(*other_id*)
Return an Element from the GEDCOM file with this id/pointer.

Parameters **other_id** (*str*) – ID/Pointer of element to search for

Returns Element with ID

Return type *Element*

Raises **KeyError** – If this id/pointer doesn't exist in the file

get_list(*tag*)
Return a list of all child elements that have this tag.

Parameters **tag** (*str*) – Tag to search for (e.g. 'DATE')

Returns list of any child nodes that have this tag

Return type list

note
Return the text of the Note (if any) of this Element.

Return None if there is no Note.

set_levels_downward()
Set all `level` attributes for all child elements recursively, based on the `level` for this object.

class gedcom.**Event**(*level=None*, *tag=None*, *value=None*, *id=None*, *parent_id=None*, *parent=None*, *gedcom_file=None*)
Generic base class for events, like *Birth* (BIRT) etc.

date
Get the Date of this event, from the 'DATE' tagged child element.

Returns date value

Return type string

Raises **KeyError** – if there is no DATE sub-element

place
Get the place of this event, from the 'PLAC' tagged child element.

Returns date value

Return type string

Raises **KeyError** – if there is no PLAC sub-element

class gedcom.**Family**(*level=None*, *tag=None*, *value=None*, *id=None*, *parent_id=None*, *parent=None*, *gedcom_file=None*)
Represents a family 'FAM' tag.

husbands
Return the *Husband*'s for this object.

Returns list

Return type list of *Husband*

partners
Return list of partners in this marriage. all HUSB/WIFE child elements. Not dereferenced.

Return type list of Husband or Wives

wives
Return the *Wife*'s for this object.

Returns list

Return type list of *Wife*

class gedcom.**GedcomFile**
Represents a GEDCOM file.

add_element(*element*)
Add an Element to this file.

If element.level is unset, it'll presume it's a top level element, and set the level and id appropriately.

:param *Element* element: Element to add

element(*tag*, ***kwargs*)
Return a new Element that is in this file.

Parameters

- **tag** (*str*) – tag name for this object
- ****kwargs** – Passed to Element constructor

Return type Element or subclass based on *tag*

ensure_header_trailer()
If GEDCOM file does not have a header (HEAD) or trailing element (TRLR), it will be added. If those exist they won't be added.

Call this method to ensure the file has these required elements.

ensure_levels()
Ensure that the levels for all elements in this file are sensible.

Sets the `Element.level` of all root elements to 0, and calls *Element.set_levels_downward()* on each one.

families
Iterator of all Family's in this file.

Returns iterator of Families's

Return type iterator

family(***kwargs*)
Create and return a Family that is in this file.

gedcom_lines()
Iterator that returns the lines in this file.

Returns iterator over lines

Return type iterator

gedcom_lines_as_string()
Return this file as a string.

Returns Full encoded text of this file

Return type string

individual(***kwargs*)
Create and return an Individual in this file.

individuals
Iterator of all Individual's in this file.

Returns iterator of Individual's

Return type iterator

save(*fileout*)
Save the contents of this GEDCOM file to specified filename or file-like object.

Parameters **fileout** – Filename or open file-like object to save this to.

Raises **Exception** – if the filename exists

class gedcom.**Husband**(*level=None*, *tag=None*, *value=None*, *id=None*, *parent_id=None*, *parent=None*, *gedcom_file=None*)

Represents pointer to a husband in a family.

class gedcom.**Individual**(*level=None*, *tag=None*, *value=None*, *id=None*, *parent_id=None*, *parent=None*, *gedcom_file=None*)

Represents and INDI (Individual) element.

aka

Return a list of 'also known as' names.

birth

Class representing the birth of this person.

death

Class representing the death of this person.

father

Calculate and return the individual represenating the father of this person.

Returns *None* if none found.

Returns the father, or *None* if not in file.

Raises **NotImplementedError** – If it cannot figure out who's the father.

Return type *Individual*

gender

Return the sex of this person, as the string 'M' or 'F'.

NB: This should probably support more sexes/genders.

Return type str

is_female

Return True iff this person is recorded as female.

is_male

Return True iff this person is recorded as male.

mother

Calculate and return the individual represenating the mother of this person.

Returns *None* if none found.

Returns the mother, or *None* if not in file.

Raises **NotImplementedError** – If it cannot figure out who's the mother.

Return type *Individual*

name

Return this person's name.

Returns a tuple of (firstname, lastname). If firstname or lastname isn't in the file, then None is returned. :returns: (firstname, lastname)

parents

Return list of parents of this person.

NB: There may be 0, 1, 2, 3, ... elements in this list.

Returns List of Individual's

set_sex(*sex*)
Set the sex for this person.

Parameters **sex** (*str*) – 'M' or 'F' for male or female resp.

Raises **TypeError** – if *sex* is invalid

sex
Return the sex of this person, as the string 'M' or 'F'.

NB: This should probably support more sexes/genders.

Return type str

title
Return the value of the Title (TITL) of this person, or None if no title.

class gedcom.**Marriage**(*level=None*, *tag=None*, *value=None*, *id=None*, *parent_id=None*, *parent=None*, *gedcom_file=None*)
Represents a marriage (MARR).

class gedcom.**Note**(*level=None*, *tag=None*, *value=None*, *id=None*, *parent_id=None*, *parent=None*, *gedcom_file=None*)
Represents a note (NOTE).

full_text
Return the full text of this note.

Internally, notes are stores across many child nodes, with child CONT/CONS child nodes that store the other lines. This method assembles these elements into one continuusous string.

class gedcom.**Spouse**(*level=None*, *tag=None*, *value=None*, *id=None*, *parent_id=None*, *parent=None*, *gedcom_file=None*)
Generic base class for HUSB/WIFE.

as_individual()
Return the *Individual* for this object.

Returns the individual

Return type *Individual*

Raises **KeyError** – if id/pointer not found in the file.

class gedcom.**Wife**(*level=None*, *tag=None*, *value=None*, *id=None*, *parent_id=None*, *parent=None*, *gedcom_file=None*)
Represents pointer to a wife in a family.

gedcom.**class_for_tag**(*tag*)
Return the class object for this *tag*.

Parameters **tag** (*str*) – tag (e.g. INDI)

Return type class (*Element* or something that's a subclass)

gedcom.**line_to_element**(***line_dict*)
Return an instance of *Element* (or subclass) based on these parsed out values from `line_regex`.

Return type *Element* or subclass

gedcom.**parse**(*obj*)
Parse and return this object, if it's a file.

If it's a filename, it calls *parse_filename()*, for file-like objects, `parse_fp`, for strings, calls `parse_string`.

Parameters **obj** – filename, open file-like object or string contents of GEDCOM file

Returns GedcomFile

gedcom.**parse_filename**(*filename*)
Parse filename and return GedcomFile.

Parameters **filename** (*string*) – Filename to parse

Returns GedcomFile instance

gedcom.**parse_fp**(*file_fp*)
Parse file and return GedcomFile.

Parameters **file_fp** (*filehandle*) – open file handle for input

Returns GedcomFile

gedcom.**parse_string**(*string*)
Parse filename and return GedcomFile.

Parameters **string** (*str*) – Filename to parse

Returns GedcomFile instance

gedcom.**register_tag**(*tag*)
Internal class decorator to mark a python class as to be the handler for this tag.

CHAPTER 5

gedcompy

Python library to parse and work with GEDCOM (genealogy/family tree) files.

It's goal is to support GEDCOM v5.5 (specification here).

This is released under the GNU General Public Licence version 3 (or at your option, a later version). See the file *LICENCE* for more.

Example Usage

```
>>> import gedcom
>>> gedcomfile = gedcom.parse("myfamilytree.ged")
>>> for person in gedcomfile.individuals:
...     firstname, lastname = person.name
...     print "{0} {1} is in the file".format(firstname, lastname)
```

Contributing

Run all unitttests with *tox*.

CHAPTER 6

Indices and tables

- genindex
- modindex
- search

Python Module Index

g

Index

S

T

W