Table of Contents

A set of python tools for convering org-roam files to a single document

- Can parse single or multiple org files, detecting and handling any roam links.
- Converts parsed org data into a tree structure, which is available for output in json format.
- Converts tree structure to html, preserving a large subset of the visual structure of the original files, and converting both internal and roam links into html links within the converted document
- Converts tree structure to latex, with with all the same features as the html version, except for differences in how document structures work in the two formats.
- Optionally converts the html to pdf using wkhtml2pdf or letex to PDF using pdflatex.
- Focused on the document creation aspect of org and org-roam, not on the todo lists, schedules, etc.
- Supports custom block surrounded with keywords `#+BEGIN_FILE_INCLUDE` ~~**BEGIN_FILE_INCLUDE~ and ~#**~ `#+BEGIN_FILE_INCLUDE` where each line in the block is treated as a file path, either full path starting at / or a path relative to the file that contains the include block. If the path resolves then whatever it contains replaces the include blog. Hopefully the contents are in org mode format, or all bets are off.
- On request, it generates a PDF that has special markup in the text and a cross reference at the end of the generated document that is designed to enable an AI (tested with Grok3) to figure out the internal links in the file after it has passed through OCR. This is particularly important if you are making the doc from a collection of org-roam files, as they tend to be writen as topic notes with minimal context and much of the available context arrises from the links between them. For a look at how the cross reference and annotations work you can read grok's note to future sessions in

HOW to use it

1. Until this becomes and actual package that includes an exectuable:
    1. PYTHONPATH=. ***src src*** roam2doc --help
2. There are two options for producing PDF, both of which require external tools
    1. The preferred method is to use the latex to pdf tool "pdflatex". This is the preferred way because it generates a decent table of contents and index, and can produce special markup for AIs (using the -grokify switch). The generated latex output uses various latex packages, all of which are available on ubuntu like this:
        1. sudo apt update
        2. sudo apt install texlive-full textlive-latex-extra
    2. Alternatively you can generate a pdf from html using wkhtmltopdf. It currently does not produce and index or the cross reference for AIs, but it does look nicer, at least to me. I may eventually add an index (if wkhtmltopdf supports it) and the AI cross reference (that is just work)
3. If you want to use wkhtmltopdf, then you will want to ensure that you have the patched QT version of wkhtml2pdf. The unpatched version will not handle links properly, nor produce a table of contents.
4. See some examples in action
    1. To see the result of combining roam files:
       `PYTHONPATH=./src src/roam2doc --help examples/roam/roam1/roam_combine1.list -o roam1.html --overwrite --doc_type=html`
       or
       `PYTHONPATH=./src src/roam2doc --help examples/roam/roam1/roam_combine1.list -o roam1.latex --overwrite --doc_type=latex`
       or
       `PYTHONPATH=./src src/roam2doc --help examples/roam/roam1/roam_combine1.list -o roam1.pdf --overwrite --doc_type=pdf --grokify`
    2. To see how the include mechansim works
       `PYTHONPATH=./src src/roam2doc --help examples/roam/roam2/roam_combine2.list -o roam2.pdf --overwrite --doc_type=pdf --grokify`
    3. To see the result of parsing a large number of org content types:
       `PYTHONPATH=./src src/roam2doc --help examples/plain/all_nodes.org -o all.html --overwrite`
    4. To see the result of parsing this file:
       `PYTHONPATH=./src src/roam2doc --help README.org -o readme.html --overwrite`

The full help:

```
usage: cli.py [-h] [-o OUTPUT] [-t {html,json,latex,pdf}] [-j] [-g] [-l {error,warning,info,debug}] [--overwrite] [--wk_pdf] input

Convert org-roam files to HTML documents.

positional arguments:
input                 Input file (.org), directory containing .org files, or file list with paths

options:
-h, --help            show this help message and exit
-o OUTPUT, --output OUTPUT
Output file path for HTML (default: print to stdout)
-t {html,json,latex,pdf}, --doc_type {html,json,latex,pdf}
Output file path for HTML (default: html)
-j, --include_json    Include a json version of the parsed document tree in the html head section
-g, --grokify         Produce a link cross reference table in pdf suitable for AI input
-l {error,warning,info,debug}, --logging {error,warning,info,debug}
Enable logging at provided level, has no effect if output goes to stdout
--overwrite           Allow overwriting existing output file (default: False)
--wk_pdf              Use wkhtmltopdf to convert output to PDF
```

Things to know

Things it does that might surpise you

- Org Keyword strings are stripped from the text during parsing. The only keyword that has
  any effect is the #+NAME: keyword, which (if at line beginning) is applied to the next non-keyword line. This allows you to name an element (e.g. a table) and then link to it by name

Things it doesn't do and probably should

- Footnotes are not parsed, they will be treated as ordinary text
- Drawers that are either property drawers at the beginning of a file or are property drawers for
  heaading are parsed, all other drawers are not parsed, just treated as text.

- Verbatim strings cannot contain equal sign "=", use ~ (inline code) if you need that in your text.

Things it doesn't do and maybe never will

- Parse and do something useful with the time management aspects of org files, this
- Inlinetasks are not parsed, they will be treated as headings an will make things ugly

Things it doesn't do and probably won't

- Run wkhtml2pdf on windows. Works on linux, will probably work on Mac
- Produce LaTex output, including any LaTex features found in the org files

Things that might be nice to add someday

- Provide option to allow uset to supply css and or javascript contents to
  be merged into the head of the html output. There is already an option to include a json object version of the parsed tree into the head, so you could write code to inspect that object and do interesting things. Of course you can do this just by editing the output directly.

History, what I wanted and why it lead to this.

What for?

I wanted to be able to take notes on a wide range of topics and relate them together into a book outline. Orgroam perfectly fit my style, so I started learning it.

First problem

I had also just started using the Grok3 AI to work on the research I was turning into notes, so I wanted to be able to load all the notes into the Grok context before submitting prompts. Grok informed me that orgroam files would not work as well as I wanted because it wouldn't do well interpreting the org files, and especially the links. Grok suggested that I would get much better results if I could collect the files into single document such as a PDF. So I needed a tool to do this. I prefer to look for python based solutions to such problems since I can modify or extend them if I need to, python being my favorite language.

The First Fix

I found the pyorg package at https: /***github.com*** nasa9084/py-org . Its main purpose was to export org content to html, and I have experience using wkhtmk2pdf to create PDFs, so that seemed workable. I forked to https: /***github.com*** dlparker/pyorg2 and was able to quickly modify it to add support for roam links. I got Grok to help me by updating the tests from nodetest to pytest. I then upgraded the tests to get 100% coverage. Seemed like a good start

Now I have two problems

As I started looking at how I wanted to use this, it became clear that I also wanted to support org internal links, which the orignal package did not. The linking to something part is simple, but the range of link targets that org supports lead to some complexity when thinking about adding it to the package. For example, you can link to a Table and almost any other element of an org file but giving it a name using a #NAME+ keyword like so:

```
#+NAME: my_table
| col 1         | col 2             |
| row 1 col 1 | row 1 col 2      |

[[my_table][link to my table]]
```

Also adding complexity to the needed changes is the fact that a link/reference can appear in many places other than plain text. Inside table cells, for example.

The original package's parsing had some other limitations as well, which may well have been the author's intention to keep the task at hand to a useful limited subset of org format. The full format is pretty rich. See https: /***orgmode.org*** worg/org-syntax.html

The scale of the modifications needed to achieve my goals convinced me that I was going to contort the structure so badly that it would be dificult to maintain. So I decided to start over.