Usage Example

In a vulgar display of sheer power (within reasonable limits), we shall generate a PDF from this very README via using the Makefile from the GitHub repository.

First, download the Makefile:

```
wget https://raw.githubusercontent.com/rldotai/makedown/master/makedown/Makefile
```

Then, setup the directory:

```
make initialize
```

Now, we shall copy this file to the source directory:

```
cp README.md -t source
```

Finally, we entreat the Makefile to generate the desired outputs:

```
make pdfs
```

Behold!

```
$ ls -t ./*
./README.md  ./Makefile

./build:

./output:
README.pdf

./source:
README.md
```

Beyond basic usage

Naturally there's some other stuff you can do as well...

Changing source/output directories

Due to self-imposed restrictions, we are, sadly, not using *this* particular file to generate the PDF, but instead a copy of it. But we can modify the `SOURCE_DIR` to point to the current directory, rather than `./source`.

```
# Backup the Makefile before modifying it
cp Makefile Makefile.bak

# Change the source directory definition in the Makefile
sed -i "s/SOURCE_DIR := source/SOURCE_DIR := ./g" Makefile
```

And we can see that it's now using this file as a source…

```
make list-sources
./README.md
```

Extraordinary. We can get the output to *also* be the current directory via a similar feat of derring-do:

```
# Change the output directory definition in the Makefile
sed -i "s/OUTPUT_DIR := output/OUTPUT_DIR := ./g" Makefile
```

…although personally I prefer to have the outputs in a separate directory.

Using templates

The default `pandoc` experience could benefit from a few tweaks, maybe using a template. There's a rule for downloading those:

```
make download-cool-templates
```

Now the increasingly cluttered directory should look like:

```
$ ls -1
build
Makefile
Makefile.bak
output
pandoc-latex.latex
pandoc-preamble.tex
README.md
```

```
README.pdf
source
```

We need to uncomment the lines in the Makefile to actually use the templates. As before, we make the changes using `sed` because I'm trying to keep things self-contained.

```
sed -i "s/# LATEX_TEMPLATE := pandoc-latex\.latex/LATEX_TEMPLATE := pandoc-latex\
sed -i "s/# LATEX_PREAMBLE := pandoc-preamble\.tex/LATEX_PREAMBLE := pandoc-preamb
```

Thilling indeed.

- Gaze upon
  - my slightly
    - improved
    - list
      - indentation
  - and
- despair.

Watching for changes

We can also use `entr` (assuming it's installed) to rebuild the PDF on every file change:

```
make watch
```

If you don't have it installed, the results will likely be far less incredible.