Show your work!

`show-your-work` is a Python module to help with writing solutions to physics problems with numerical answers. It relies on other modules to do most of the heavy lifting.

Installing

Show-your-work is available on PyPi. The package is named `show-your-work-utils` to avoid potential name conflicts. Install it with your favorite package manager (mine is `uv`).

```
$ pip install show-your-work-utils
$ uv add show-your-work-utils
$ rye add show-your-work-utils
$ poetry add show-your-work-utils
```

Motivation

When writing detailed example problems for a physics class, we often need to plug numerical values into an equation and compute a numerical result. `show-your-work` allows you to write an equation in latex, add numerical substitutions for each term in the equations, produce the latex of your equation with numerical values substituted in, and then evaluate the equation with the given numerical values.

It is especially handy when used with tools like compudoc.

From the `doc/examples` directory

```
% main.tex.cd
\documentclass[]{article}

\usepackage{siunitx}
\usepackage{physics}
\usepackage{graphicx}
\usepackage{fullpage}

\author{C.D. Clark III}
\title{show-your-work/compudoc example}

\begin{document}
\maketitle

% {{{
% from show_your_work import Expression, Equation
% import pint
% ureg = pint.UnitRegistry()
```

```
% Q_ = ureg.Quantity
% def Lx_filter(input,fmt=""):
%   text = fmt_filter(input,fmt+"Lx")
%   return text
% jinja2_env.filters["Lx"] = Lx_filter
% line_eq = Equation(r"\\v{y} = \\v{m}\\v{x} + \\v{b}")
% print(line_eq.latex)
% }}}
$$
{{ line_eq.latex }}
$$
This gives the relationship between a dependent variable $y$ on an independent variable $x$.
% {{{
% slope = 3
% y_intercept = 2
% val = 10
% }}}
For example, for a slope $m = {{slope}}$ and y-intercept $b = {{y_intercept}}$, the value fo
% {{{
% line_eq.rhs.add_substitution('m',slope)
% line_eq.rhs.add_substitution('b',y_intercept)
% line_eq.rhs.add_substitution('x',val)
% }}}
$$
{{ line_eq.latex_with_substitutions }} = {{line_eq.rhs.eval()}}
$$
% {{{
% kinematics_eq1 = Equation( r"\\v{x} = \\v{v} \\v{t} + \\v{x_0}", "meter")
% v = Q_(60,'mph').to("km/hr")
% t = Q_(40,'min')
% x0 = Q_(150,'yd').to("m")
% kinematics_eq1.rhs.add_substitution('v',v)
% kinematics_eq1.rhs.add_substitution('t',t)
% kinematics_eq1.rhs.add_substitution('x_0',x0)
% }}}
Now, when dealing with physical quantities, we will have to consider units. Take an example
position of an object traveling at constant speed is given by the equation,
$$
{{kinematics_eq1.latex}},
$$
which is just the equation for a line with $v$ as the slope and $x0$ as the y-intercept. If
$v = {{v|Lx}}$ and we didn't start the stop watch until we were {{x0|Lx}} out of town, the c
after traveling for {{t|Lx}} would be:
$$
{{kinematics_eq1.latex_with_substitutions}} = {{kinematics_eq1.rhs.eval(Q_)|Lx}}
$$
```

```
\end{document}
```

This template file can be rendered to LaTeX with `compudoc` and then compiled to a PDF with `latexmk`:

```
$ compudoc main.tex.cd
$ latexmk main
```

which will give this PDF