

---

# lookatme

*Release v2.5.3*

Oct 27, 2022



---

## Contents

---

<b>1</b>	<b>Tour</b>	<b>3</b>
<b>2</b>	<b>TL;DR Getting Started</b>	<b>5</b>
2.1	Getting Started . . . . .	6
2.2	Slides . . . . .	9
2.3	Dark Theme . . . . .	10
2.4	Light Theme . . . . .	11
2.5	Style Precedence . . . . .	12
2.6	Contrib Extensions . . . . .	14
2.7	Smart Slide Splitting . . . . .	17
2.8	Builtin Extensions . . . . .	18
2.9	lookatme . . . . .	20
<b>3</b>	<b>Indices and tables</b>	<b>41</b>
	<b>Python Module Index</b>	<b>43</b>
	<b>Index</b>	<b>45</b>



lookatme is an interactive, terminal-based markdown presentation tool that supports:

- Themes
- Syntax highlighting
- Styling and settings embedded within the Markdown YAML header
- Embedded terminals as part of a presentation
- Live and manual source reloading
- Contrib extensions
- Smart Slide Splitting



# CHAPTER 1

---

Tour

---





## CHAPTER 2

---

### TL;DR Getting Started

---

Install lookatme with:

```
pip install lookatme
```

Run lookatme on slides written in Markdown:

```
lookatme slides.md
```

Slides are separated with `---` hrules:

```
# Slide 1
```

```
Some text
```

```
---
```

```
# Slide 2
```

```
More text
```

A basic, optional YAML header may be included at the top of the slides:

```
---
title: Slides Presentation
author: Me Not You
date: 2019-12-02
---
```

```
# Slide 1
```

```
Some text
```

Slides can be progressively rendered by adding `<!-- stop -->` comments between block elements (paragraphs, tables, lists, etc.):

### # Progressive Slide

```
<!-- stop -->
```

Paragraph 1

```
<!-- stop -->
```

Paragraph 2

```
<!-- stop -->
```

Paragraph 3

## 2.1 Getting Started

### 2.1.1 Installation

lookatme can be installed with pip using the command:

```
pip install lookatme
```

### 2.1.2 Usage

The lookatme CLI has a few options to control it's behavior:

```
{{LOOKATME_HELP_OUTPUT_INDENTED}}
```

#### **--live / --live-reload**

This flag turns on live reloading within lookatme. If the input markdown is a filepath (and not stdin), the filepath with be watched for changes to its modification time. If a change to the file's modification time is observed, the slide deck is re-read and rendered, keeping the current slide in focus.

If your editor supports saving with every keystroke, instant slide updates are possible:

#### **-e EXT\_NAME1,EXT\_NAME2 / --exts EXT\_NAME1,EXT\_NAME2**

Allows a comma-separated list of extension names to be pre-loaded into lookatme without requiring them to be declared in the Markdown source.

#### **-s / --safe**

Do **NOT** load any new extensions specified in the markdown (ignore them). New extensions are extensions that have not manually been allowed via the `-e` argument or the `LOOKATME_EXTS` environment variable.

**--no-ext-warn**

Do not warn about new extensions that are to-be-loaded that are specified in the source markdown. New extensions are extensions that have not manually been allowed via the `-e` argument or the `LOOKATME_EXTS` environment variable.

**-i**

Ignore failure loading extensions. This does not ignore warnings, but ignores any hard-errors during import, such as `ImportError`.

**--single / --one**

Render the markdown source as a single slide, ignoring all hrules. Scroll overflowing slides with the up/down arrow keys and page up/page down.

**--debug and --log**

Turns on debug logging for lookatme. The debug log will be created in your platform's temporary directory by default and will be named `lookatme.log`:

```
$> lookatme slides.md --debug

# in another terminal
$> tail -f /tmp/lookatme.log
DEBUG:lookatme.RENDER:   Rendering token {'type': 'heading', 'level': 2, 'text': 'TOC'}
DEBUG:lookatme.RENDER:   Rendering token {'type': 'list_start', 'ordered': False}
DEBUG:lookatme.RENDER:   Rendering token {'type': 'list_item_start'}
DEBUG:lookatme.RENDER:   Rendering token {'type': 'text', 'text': '[Features] (
↪ #features) ' }
DEBUG:lookatme.RENDER:   Rendering token {'type': 'list_start', 'ordered': False}
DEBUG:lookatme.RENDER:   Rendering token {'type': 'list_item_start'}
```

You may set a custom log location with the `--log` flag

**--theme**

Themes in lookatme are pre-defined stylings. Lookatme comes with two built-in themes: `dark` and `light`. These themes are intended to look good on dark terminals and light terminals.

See the [Dark Theme](#) and [Light Theme](#) pages for more details. See the [Style Precedence](#) page for details on the order style overrides and settings are applied.

**--style**

This option overrides the [Pygments](#) syntax highlighting style to use. See the [Style Precedence](#) for details about style overriding order.

At the time of this writing, available Pygments style options include:

- default
- emacs
- friendly

- colorful
- autumn
- murphy
- manni
- monokai
- perldoc
- pastie
- borland
- trac
- native
- fruity
- bw
- vim
- vs
- tango
- rrt
- xcode
- igor
- paraiso-light
- paraiso-dark
- lovelace
- algol
- algol\_nu
- arduino
- rainbow\_dash
- abap
- solarized-dark
- solarized-light
- sas
- stata
- stata-light
- stata-dark

### --dump-styles

Print the final, resolved style definition that will be used to render the markdown as currently specified on the command-line. See the *Style Precedence* section for details on how this works.

E.g.:

```
lookatme examples/tour.md -theme --style solarized-dark --dump-styles
```

## 2.2 Slides

Slides in lookatme are:

- Separated by hrule elements: --- in Markdown
- Resized to fit the current window

### 2.2.1 Metadata

Slide metadata is contained within an optional YAML header:

```

---
title: TITLE
author: AUTHOR
date: 2019-12-02
extensions: []
styles: {}
---
```

Additional, unknown metadata fields are allowed at the top level. However, the `styles` field and subfields are strictly validated.

### Extensions

Extensions are lookatme contrib modules that redefine lookatme behavior. E.g., the `lookatmecontrib.calendar` example in the `examples` folder redefines the `render_code` function found in `lookatme/render/markdown_block.py`.

The original `render_code` function gives contrib extensions first-chance at handling any function calls. Contrib extensions are able to ignore function calls, and thus allow the default lookatme behavior, by raising the *IgnoredByContrib* exception:

```

import datetime
import calendar
import urwid

from lookatme.exceptions import IgnoredByContrib

def render_code(token, body, stack, loop):
    lang = token["lang"] or ""
    if lang != "calendar":
        raise IgnoredByContrib()
```

(continues on next page)

(continued from previous page)

```
today = datetime.datetime.utcnow()
return urwid.Text(calendar.month(today.year, today.month))
```

## Styles

In addition to the `--style` and `--theme` CLI options for lookatme, the slide metadata may explicitly override styling behaviors within lookatme:

```
---
title: TITLE
author: AUTHOR
date: 2019-12-02
styles:
  style: monokai
  table:
    column_spacing: 3
    header_divider: "-"
---

# Slide 1

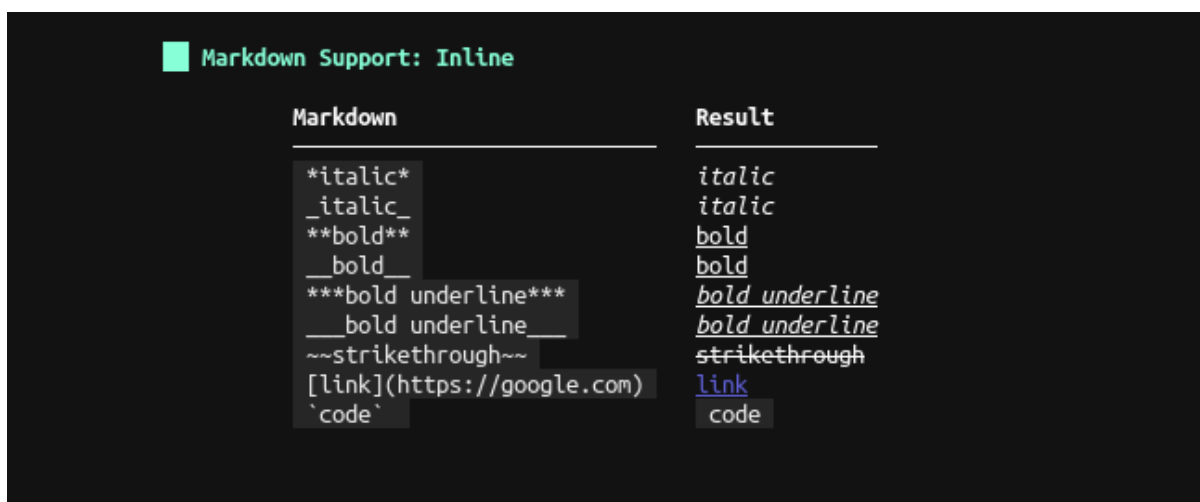
text
```

The final, resolved styling settings that will be used when displaying a markdown source is viewable by adding the `--dump-styles` flag as a command-line argument.

See the [Default Style Settings](#) for a full list of available, overrideable styles.

## 2.3 Dark Theme

The dark theme is intended to appear well on terminals with dark backgrounds



Markdown	Result
<code>*italic*</code>	<i>italic</i>
<code>_italic_</code>	<i>italic</i>
<code>**bold**</code>	<b>bold</b>
<code>__bold__</code>	<b>bold</b>
<code>***bold underline***</code>	<b><u>bold underline</u></b>
<code>___bold underline___</code>	<b><u>bold underline</u></b>
<code>~~strikethrough~~</code>	<del>strikethrough</del>
<code>[link](https://google.com)</code>	<a href="https://google.com">link</a>
<code>`code`</code>	<code>code</code>

## Markdown Support: Headers

Heading 2

Heading 3

Heading 4

More text

## Markdown Support: Code Blocks & Quotes

Code blocks with language syntax highlighting

```
def a_function(arg1, arg2):
    """This is a function
    """
    print(arg1)
```

A quote is below:

```
[ This is a quote ]
```

## 2.4 Light Theme

The light theme is intended to appear well on terminals with light backgrounds

### Markdown Support: Inline

Markdown	Result
<code>*italic*</code>	<i>italic</i>
<code>_italic_</code>	<i>italic</i>
<code>**bold**</code>	<b>bold</b>
<code>__bold__</code>	<b>bold</b>
<code>***bold underline***</code>	<b><u>bold underline</u></b>
<code>__bold underline__</code>	<b><u>bold underline</u></b>
<code>~~strikethrough~~</code>	<del>strikethrough</del>
<code>[link](https://google.com)</code>	<a href="https://google.com">link</a>
<code>`code`</code>	<code>code</code>

## ■ Markdown Support: Headers

### ■ Heading 2

### ■ Heading 3

### ■ Heading 4

More text

## ■ Markdown Support: Code Blocks & Quotes

Code blocks with language syntax highlighting

```
def a_function(arg1, arg2):  
    """This is a function  
    """  
    print(arg1)
```

A quote is below:

```
[ This is a quote
```

## 2.5 Style Precedence

Styling may be set in three locations in lookatme:

1. In a theme
2. In a slide's YAML header
3. On the command-line

When constructing the final, resolved style set that will be used to render markdown, lookatme starts with the default style settings defined in `lookatme.schemas`, and then applies overrides in the order specified above.

Overrides are applied by performing a deep merge of nested dictionaries. For example, if the default styles defined in `schemas.py` were:

```
headings:  
  "1":  
    fg: "#33c,bold"  
    bg: "default"  
  "2":  
    fg: "#222,bold"  
    bg: "default"
```

... and if the style overrides defined by a theme were:



```
headings:
  "1":
    bg: "#f00"
```

... and if the style overrides defined in the slide YAML header were:

```
headings:
  "2":
    fg: "#f00,bold,underline"
```

The final, resolved style settings for rendering the markdown would be:

```
headings:
  "1":
    fg: "#33c,bold"
    bg: "#f00" # from the theme
  "2":
    fg: "#f00,bold,underline" # from the slide YAML header
    bg: "default"
```

## 2.5.1 Default Style Settings

The default styles and formats are defined in the marshmallow schemas in `lookatme.schemas`. The dark theme is an empty theme with no overrides (the defaults *are* the dark theme):

```
author:
  bg: default
  fg: '#f30'
bullets:
  '1': .
  '2':
  '3': o
  default: .
date:
  bg: default
  fg: '#777'
headings:
  '1':
    bg: default
    fg: '#9fc,bold'
    prefix: ' '
    suffix: ' '
  '2':
    bg: default
    fg: '#1cc,bold'
    prefix: ' '
    suffix: ' '
  '3':
    bg: default
    fg: '#29c,bold'
    prefix: ' '
    suffix: ' '
  '4':
    bg: default
    fg: '#559,bold'
```

(continues on next page)

(continued from previous page)

```

    prefix: ' '
    suffix: ' '
  default:
    bg: default
    fg: '#346,bold'
    prefix: ' '
    suffix: ' '
  hrule:
    char: -
    style:
      bg: default
      fg: '#777'
  link:
    bg: default
    fg: '#33c,underline'
  margin:
    bottom: 0
    left: 2
    right: 2
    top: 0
  numbering:
    '1': numeric
    '2': alpha
    '3': roman
    default: numeric
  padding:
    bottom: 0
    left: 10
    right: 10
    top: 0
  quote:
    bottom_corner: L
    side:
    style:
      bg: default
      fg: italics,#aaa
    top_corner:
  slides:
    bg: default
    fg: '#f30'
  style: monokai
  table:
    column_spacing: 3
    header_divider: -
  title:
    bg: default
    fg: '#f30,bold,italics'

```

## 2.6 Contrib Extensions

lookatme allows an extension to override and redefine how markdown is rendered. Extensions have first-chance opportunities to handle rendering function calls. Extensions also have the ability to ignore specific rendering function calls and allow original lookatme behavior (or other extensions) to handle the call to that rendering function.

For example, an extension may provide its own implementation of the render function `render_table` to provide

custom table rendering, such as sortable rows, alternating row background colors, etc.

## 2.6.1 Using Extensions

Extensions are namespace packages within `lookatme.contrib`. They are used by

1. Installing the extension with `pip install lookatme.contrib.XXX`
2. Adding the extension to the list of extensions required by your slides:

```
---
title: TITLE
author: AUTHOR
date: 2019-11-01
extensions:
  - XXX
---

# Slide 1

...
```

## 2.6.2 Extension Layout

It is highly recommended that you use the `lookatme.contrib-template` to create new extensions.

Extensions *must* be a namespaced module within the `lookatme.contrib` submodule. The basic tree layout for such an extension is below:

```
examples/calendar_contrib/
├── lookatme
│   └── contrib
│       └── calendar.py
└── setup.py
```

Notice that there is not an `__init__.py` file in the `contrib` path. This is using the [implicit namespace package](#) format for creating namespace packages, where an `__init__.py` is not needed.

## 2.6.3 Extension setup.py

Below is the `setup.py` from the `examples/calendar_contrib` extension:

```
"""
Setup for lookatme.contrib.calender example
"""

from setuptools import setup, find_namespace_packages
import os

setup(
    name="lookatme.contrib.calendar",
    version="0.0.0",
    description="Adds a calendar code block type",
```

(continues on next page)

(continued from previous page)

```
author="James Johnson",
author_email="d0c.s4vage@gmail.com",
python_requires=">=3.5",
packages=find_namespace_packages(include=["lookatme.*"]),
)
```

## 2.6.4 Overriding Behavior

Any function within lookatme that is decorated with `@contrib_first` may be overridden by an extension by defining a function of the same name within the extension module.

For example, to override the `render_code` function that is declared in lookatme in `lookatme/render/markdown_block.py`, the example calendar extension must declare its own function named `render_code` that accepts the same arguments and provides the same return values as the original function:

```
"""
Defines a calendar extension that overrides code block rendering if the
language type is calendar
"""

import datetime
import calendar
import urwid

from lookatme.exceptions import IgnoredByContrib

def user_warnings():
    """No warnings exist for this extension. Anything you want to warn the
    user about, such as security risks in processing untrusted markdown, should
    go here.
    """
    return []

def render_code(token, body, stack, loop):
    lang = token["lang"] or ""
    if lang != "calendar":
        raise IgnoredByContrib()

    today = datetime.datetime.utcnow()
    return urwid.Text(calendar.month(today.year, today.month))
```

Notice how the extension code above raises the `IgnoredByContrib` exception to allow the default lookatme behavior to occur.

## 2.6.5 Overrideable Functions

Below is an automatically generated list of all overrideable functions that are present in this release of lookatme. See the `lookatme.tui.SlideRenderer.do_render` function for details on `markdown_block` render function arguments and return values.

- `render_newline`

- `render_hruler`
- `render_heading`
- `render_table`
- `render_list_start`
- `render_list_end`
- `render_list_item_start`
- `render_loose_item_start`
- `render_list_item_end`
- `render_text`
- `render_paragraph`
- `render_block_quote_start`
- `render_block_quote_end`
- `render_code`
- `inline_html`
- `text`
- `escape`
- `autolink`
- `footnote_ref`
- `image`
- `link`
- `emphasis`
- `double_emphasis`
- `codespan`
- `linebreak`
- `strikethrough`
- `root_urwid_widget`

## 2.7 Smart Slide Splitting

lookatme will automatically split input markdown into separate slides if no `hrules` are present in the input markdown.

Slides are split automatically in two ways

1. If the lowest (e.g. `h1 < h2`) heading occurs only once, that heading is used as the title for the presentation. The next lowest heading will be used as the slide separator marker.
2. If the lowest (e.g. `h1 < h2`) heading occurs multiple times, that heading will be used as the slide separator marker and the heading will not be set.

E.g., below is the README.md of lookatme:

## 2.8 Builtin Extensions

lookatme comes with a few built-in extensions.

### 2.8.1 Builtin Extension Qualification

Builtin extensions must:

- Not require extra dependencies just for the extension
- Be generally useful in most cases

E.g., the qrcode extension has an extra dependency. This immediately disqualifies it from being a builtin extension.

### 2.8.2 Usage

Although builtin extensions are defined in the same way as external *Contrib Extensions*, builtin extensions do not need to be explicitly declared in the YAML header.

### 2.8.3 List of Builtin Extensions

#### Terminal Extension

The `lookatme.contrib.terminal` builtin extension allows terminals to be embedded within slides.

#### Basic Format

The terminal extension modifies the code block markdown rendering by intercepting code blocks whose language has the format `terminal\d+`. The number following the `terminal` string indicates how many rows the terminal should use when rendered (the height).

#### Usage

E.g.

```
```terminal8
bash -il
```
```

The content of the code block is the command to be run in the terminal. Clicking inside of the terminal gives the terminal focus, which will allow you to interact with it, type in it, etc.

To escape from the terminal, press `ctrl+a`.

#### Extended Format

The terminal extension also has a *terminal-ex* mode that can be used as the language in a code block. When *terminal-ex* is used, the contents of the code block must be YAML that conforms to the *TerminalExSchema* schema.

The default schema is shown below:

```

command: "the command to run" # required
rows: 10                      # number of rows for the terminal (height)
init_text: null               # initial text to feed to the command. This is
                              # useful to, e.g., pre-load text on a
                              # bash prompt so that only "enter" must be
                              # pressed. Uses the `expect` command.
init_wait: null               # the prompt (string) to wait for with `expect`
                              # this is required if init_text is set.
init_codeblock: true          # show a codeblock with the init_text as its
                              # content
init_codeblock_lang: text     # the language of the init codeblock

```

## Usage

E.g.

```

```terminal-ex
command: bash -il
rows: 20
init_text: echo hello
init_wait: '$> '
init_codeblock_lang: bash
```

```

## File Loader Extension

The `lookatme.contrib.file_loader` builtin extension allows external files to be sourced into the code block, optionally being transformed and optionally restricting the range of lines to display.

## Format

The file loader extension modifies the code block markdown rendering by intercepting code blocks whose language equals `file`. The contents of the code block must be YAML that conforms to the `FileSchema` schema.

The default schema is shown below:

```

path: path/to/the/file # required
relative: true          # relative to the slide source directory
lang: text              # pygments language to render in the code block
transform: null         # optional shell command to transform the file data
lines:
  start: 0
  end: null

```

---

**Note:** The line range is only applied **AFTER** transformations are performed on the file data.

---

## Usage

E.g.

```
```file
path: ../source/main.c
lang: c
```
```

## 2.9 lookatme

### 2.9.1 lookatme package

#### Subpackages

#### lookatme.contrib package

#### Submodules

#### lookatme.contrib.file\_loader module

This module defines a built-in contrib module that enables external files to be included within the slide. This is extremely useful when having source code displayed in a code block, and then running/doing something with the source data in a terminal on the same slide.

```
class lookatme.contrib.file_loader.FileSchema(*only: types.StrSequenceOrSet | None
                                              = None, exclude: types.StrSequenceOrSet
                                              = (), many: bool = False, context: dict | None = None, load_only:
                                              types.StrSequenceOrSet = (), dump_only:
                                              types.StrSequenceOrSet = (), partial: bool
                                              | types.StrSequenceOrSet = False, unknown: str | None = None)
```

Bases: `marshmallow.schema.Schema`

**class Meta**

Bases: `object`

**render\_module**

alias of `YamlRender`

**load**(\*args, \*\*kwargs) → Dict[KT, VT]

Deserialize a data structure to an object defined by this Schema's fields.

#### Parameters

- **data** – The data to deserialize.
- **many** – Whether to deserialize *data* as a collection. If *None*, the value for *self.many* is used.
- **partial** – Whether to ignore missing fields and not require any fields declared. Propagates down to *Nested* fields as well. If its value is an iterable, only missing fields listed in that iterable will be ignored. Use dot delimiters to specify nested fields.
- **unknown** – Whether to exclude, include, or raise an error for unknown fields in the data. Use *EXCLUDE*, *INCLUDE* or *RAISE*. If *None*, the value for *self.unknown* is used.

**Returns** Deserialized data



New in version 1.0.0.

Changed in version 3.0.0b7: This method returns the deserialized data rather than a `(data, errors)` tuple. A `ValidationError` is raised if invalid data are passed.

**loads** (\*args, \*\*kwargs) → Dict[KT, VT]

Same as `load()`, except it takes a JSON string as input.

#### Parameters

- **json\_data** – A JSON string of the data to deserialize.
- **many** – Whether to deserialize *obj* as a collection. If *None*, the value for *self.many* is used.
- **partial** – Whether to ignore missing fields and not require any fields declared. Propagates down to *Nested* fields as well. If its value is an iterable, only missing fields listed in that iterable will be ignored. Use dot delimiters to specify nested fields.
- **unknown** – Whether to exclude, include, or raise an error for unknown fields in the data. Use *EXCLUDE*, *INCLUDE* or *RAISE*. If *None*, the value for *self.unknown* is used.

**Returns** Deserialized data

New in version 1.0.0.

Changed in version 3.0.0b7: This method returns the deserialized data rather than a `(data, errors)` tuple. A `ValidationError` is raised if invalid data are passed.

**opts** = <marshmallow.schema.SchemaOpts object>

```
class lookatme.contrib.file_loader.LineRange(*, only: types.StrSequenceOrSet | None
                                             = None, exclude: types.StrSequenceOrSet
                                             = (), many: bool = False, context: dict | None = None, load_only:
                                             types.StrSequenceOrSet = (), dump_only:
                                             types.StrSequenceOrSet = (), partial: bool |
                                             types.StrSequenceOrSet = False, unknown:
                                             str | None = None)
```

Bases: marshmallow.schema.Schema

**opts** = <marshmallow.schema.SchemaOpts object>

```
class lookatme.contrib.file_loader.YamlRender
```

Bases: object

**static dumps** (data)

**static loads** (data)

```
lookatme.contrib.file_loader.render_code(token, body, stack, loop)
```

Render the code, ignoring all code blocks except ones with the language set to *file*.

```
lookatme.contrib.file_loader.transform_data(transform_shell_cmd, input_data)
```

Transform the *input\_data* using the *transform\_shell\_cmd* shell command.

```
lookatme.contrib.file_loader.user_warnings()
```

Provide warnings to the user that loading this extension may cause shell commands specified in the markdown to be run.

## lookatme.contrib.terminal module

This module defines a built-in contrib module that enables terminal embedding within a slide.

```
class lookatme.contrib.terminal.TerminalExSchema(*, only: types.StrSequenceOrSet
| None = None, exclude:
types.StrSequenceOrSet = (),
many: bool = False, context:
dict | None = None, load_only:
types.StrSequenceOrSet = (),
dump_only: types.StrSequenceOrSet
= (), partial: bool |
types.StrSequenceOrSet = False,
unknown: str | None = None)
```

Bases: `marshmallow.schema.Schema`

The schema used for `terminal-ex` code blocks.

```
class Meta
```

Bases: `object`

```
render_module
```

alias of `YamlRender`

```
load(*args, **kwargs) → Dict[KT, VT]
```

Deserialize a data structure to an object defined by this Schema's fields.

#### Parameters

- **data** – The data to deserialize.
- **many** – Whether to deserialize *data* as a collection. If *None*, the value for *self.many* is used.
- **partial** – Whether to ignore missing fields and not require any fields declared. Propagates down to `Nested` fields as well. If its value is an iterable, only missing fields listed in that iterable will be ignored. Use dot delimiters to specify nested fields.
- **unknown** – Whether to exclude, include, or raise an error for unknown fields in the data. Use *EXCLUDE*, *INCLUDE* or *RAISE*. If *None*, the value for *self.unknown* is used.

**Returns** Deserialized data

New in version 1.0.0.

Changed in version 3.0.0b7: This method returns the deserialized data rather than a `(data, errors)` tuple. A `ValidationError` is raised if invalid data are passed.

```
loads(*args, **kwargs) → Dict[KT, VT]
```

Same as `load()`, except it takes a JSON string as input.

#### Parameters

- **json\_data** – A JSON string of the data to deserialize.
- **many** – Whether to deserialize *obj* as a collection. If *None*, the value for *self.many* is used.
- **partial** – Whether to ignore missing fields and not require any fields declared. Propagates down to `Nested` fields as well. If its value is an iterable, only missing fields listed in that iterable will be ignored. Use dot delimiters to specify nested fields.
- **unknown** – Whether to exclude, include, or raise an error for unknown fields in the data. Use *EXCLUDE*, *INCLUDE* or *RAISE*. If *None*, the value for *self.unknown* is used.

**Returns** Deserialized data

New in version 1.0.0.

Changed in version 3.0.0b7: This method returns the deserialized data rather than a (data, errors) tuple. A `ValidationError` is raised if invalid data are passed.

**opts** = <marshmallow.schema.SchemaOpts object>

**class** lookatme.contrib.terminal.YamlRender

Bases: object

**static dumps** (data)

**static loads** (data)

lookatme.contrib.terminal.render\_code (token, body, stack, loop)

lookatme.contrib.terminal.shutdown ()

lookatme.contrib.terminal.user\_warnings ()

Provide warnings to the user that loading this extension may cause shell commands specified in the markdown to be run.

## Module contents

This module handles loading and using lookatme\_contrib modules

**def loads(self, \*args, \*\*kwargs) -> Dict:** res = super(self.\_\_class\_\_, self).loads(\*args, \*\*kwargs) if res is None:

raise ValueError("Could not loads")

return res

**def load(self, \*args, \*\*kwargs) -> Dict:** res = super(self.\_\_class\_\_, self).load(\*args, \*\*kwargs) if res is None:

raise ValueError("Could not load")

return res

Contrib modules are directly used

lookatme.contrib.contrib\_first (fn)

A decorator that allows contrib modules to override default behavior of lookatme. E.g., a contrib module may override how a table is displayed to enable sorting, or enable displaying images rendered with ANSI color codes and box drawing characters, etc.

Contrib modules may ignore chances to override default behavior by raising the lookatme.contrib.IgnoredByContrib exception.

lookatme.contrib.load\_contribs (contrib\_names, safe\_contribs, ignore\_load\_failure=False)

Load all contrib modules specified by contrib\_names. These should all be namespaced packages under the lookatmecontrib namespace. E.g. lookatmecontrib.calendar would be an extension provided by a contrib module, and would be added to an extensions list in a slide's YAML header as calendar.

safe\_contribs is a set of contrib names that are manually provided by the user by the -e flag or env variable of extensions to auto-load.

lookatme.contrib.shutdown\_contribs ()

Call the shutdown function on all contrib modules

lookatme.contrib.validate\_extension\_mod (\_ext\_name, ext\_mod)

Validate the extension, returns an array of warnings associated with the module

## lookatme.render package

### Submodules

#### lookatme.render.asciinema module

#### lookatme.render.markdown\_block module

Defines render functions that render lexed markdown block tokens into urwid representations

`lookatme.render.markdown_block.render_block_quote_end` (*token, body, stack, loop*)

Pops the block quote start `urwid.Pile()` from the stack, taking future renderings out of the block quote styling.

See `lookatme.tui.SlideRenderer.do_render` for additional argument and return value descriptions.

`lookatme.render.markdown_block.render_block_quote_start` (*token, body, stack, loop*)

Begins rendering of a block quote. Pushes a new `urwid.Pile()` to the stack that is indented, has styling applied, and has the quote markers on the left.

This function makes use of the styles:

```
quote:
  top_corner: ""
  bottom_corner: " L"
  side: ""
  style:
    bg: default
    fg: italics,#aaa
```

See `lookatme.tui.SlideRenderer.do_render` for additional argument and return value descriptions.

`lookatme.render.markdown_block.render_code` (*token, body, stack, loop*)

Renders a code block using the Pygments library.

See `lookatme.tui.SlideRenderer.do_render` for additional argument and return value descriptions.

`lookatme.render.markdown_block.render_heading` (*token, body, stack, loop*)

Render markdown headings, using the defined styles for the styling and prefix/suffix.

See `lookatme.tui.SlideRenderer.do_render` for argument and return value descriptions.

Below are the default stylings for headings:

```
headings:
  '1':
    bg: default
    fg: '#9fc,bold'
    prefix: " "
    suffix: ""
  '2':
    bg: default
    fg: '#1cc,bold'
    prefix: " "
    suffix: ""
```

(continues on next page)

(continued from previous page)

```
'3':
  bg: default
  fg: '#29c,bold'
  prefix: " "
  suffix: ""
'4':
  bg: default
  fg: '#66a,bold'
  prefix: " "
  suffix: ""
default:
  bg: default
  fg: '#579,bold'
  prefix: " "
  suffix: ""
```

**Returns** A list of urwid Widgets or a single urwid Widget

`lookatme.render.markdown_block.render_hruler(token, body, stack, loop)`

Render a newline

See `lookatme.tui.SlideRenderer.do_render` for argument and return value descriptions.

`lookatme.render.markdown_block.render_list_end(token, body, stack, loop)`

Pops the pushed `urwid.Pile()` from the stack (decreases indentation)

See `lookatme.tui.SlideRenderer.do_render` for argument and return value descriptions.

`lookatme.render.markdown_block.render_list_item_end(token, body, stack, loop)`

Pops the pushed `urwid.Pile()` from the stack (decreases indentation)

See `lookatme.tui.SlideRenderer.do_render` for argument and return value descriptions.

`lookatme.render.markdown_block.render_list_item_start(token, body, stack, loop)`

Render the start of a list item. This function makes use of the styles:

```
bullets:
  '1': "."
  '2': ""
  '3': "o"
default: "."
```

See `lookatme.tui.SlideRenderer.do_render` for argument and return value descriptions.

`lookatme.render.markdown_block.render_list_start(token, body, stack, loop)`

Handles the indentation when starting rendering a new list. List items themselves (with the bullets) are rendered by the `render_list_item_start` function.

See `lookatme.tui.SlideRenderer.do_render` for argument and return value descriptions.

`lookatme.render.markdown_block.render_loose_item_start(token, body, stack, loop)`

Render the start of a list item. This function makes use of the styles:

```
bullets:
  '1': "."
  '2': ""
  '3': "o"
default: "."
```

See `lookatme.tui.SlideRendererer.do_render` for argument and return value descriptions.

`lookatme.render.markdown_block.render_newline` (*token, body, stack, loop*)

Render a newline

See `lookatme.tui.SlideRendererer.do_render` for argument and return value descriptions.

`lookatme.render.markdown_block.render_paragraph` (*token, body, stack, loop*)

Renders the provided text with additional pre and post paddings.

See `lookatme.tui.SlideRendererer.do_render` for additional argument and return value descriptions.

`lookatme.render.markdown_block.render_table` (*token, body, stack, loop*)

Renders a table using the `Table` widget.

See `lookatme.tui.SlideRendererer.do_render` for argument and return value descriptions.

The table widget makes use of the styles below:

```
table:
  column_spacing: 3
  header_divider: "-"
```

**Returns** A list of urwid Widgets or a single urwid Widget

`lookatme.render.markdown_block.render_text` (*token=None, body=None, stack=None, loop=None, text=None*)

Renders raw text. This function uses the inline markdown lexer from mistune with the `lookatme.render.markdown_inline` render module to render the lexed inline markup to a list composed of widgets or `urwid Text markup`. The created list of widgets/Text markup is then used to create and return a list composed entirely of widgets and `ClickableText` instances.

Many other functions call this function directly, passing in the extra `text` argument and leaving all other arguments blank.

See `lookatme.tui.SlideRendererer.do_render` for additional argument and return value descriptions.

## lookatme.render.markdown\_inline module

Defines render functions that work with mistune's markdown inline lexer render interface

`lookatme.render.markdown_inline.autolink` (*link\_uri, is\_email=False*)

Renders a URI as a link

**Returns** list of `urwid Text markup` tuples.

`lookatme.render.markdown_inline.codespan` (*text, old\_styles*)

Renders inline code using the pygments renderer. This function also makes use of the coding style:

```
style: monokai
```

**Returns**

list of `urwid Text markup` tuples.

`lookatme.render.markdown_inline.double_emphasis` (*text, old\_styles*)

Renders double emphasis. Handles both `**word**` and `__word__`

**Returns**

list of [urwid Text markup](#) tuples.

```
lookatme.render.markdown_inline.emphasis (text, old_styles)
```

Renders double emphasis. Handles both `*word*` and `_word_`

**Returns**

list of [urwid Text markup](#) tuples.

```
lookatme.render.markdown_inline.escape (text)
```

Renders escapes

**Returns**

list of [urwid Text markup](#) tuples.

```
lookatme.render.markdown_inline.expanded_styles (fn)
```

```
lookatme.render.markdown_inline.footnote_ref (key, index)
```

Renders a footnote

**Returns**

list of [urwid Text markup](#) tuples.

```
lookatme.render.markdown_inline.image (link_uri, title, text)
```

Renders an image as a link. This would be a cool extension to render referenced images as scaled-down ansi pixel blocks.

**Returns**

list of [urwid Text markup](#) tuples.

```
lookatme.render.markdown_inline.inline_html (text)
```

Renders inline html as plaintext

**Returns**

list of [urwid Text markup](#) tuples.

```
lookatme.render.markdown_inline.linebreak ()
```

Renders a line break

**Returns**

list of [urwid Text markup](#) tuples.

```
lookatme.render.markdown_inline.link (link_uri, title, link_text)
```

Renders a link. This function does a few special things to make the clickable links happen. All text in lookatme is rendered using the [ClickableText](#) class. The `ClickableText` class looks for `urwid.AttrSpec` instances that are actually `LinkIndicatorSpec` instances within the Text markup. If an `AttrSpec` is an instance of `LinkIndicator` spec in the Text markup, `ClickableText` knows to handle clicks on that section of the text as a link.

**Returns**

list of [urwid Text markup](#) tuples.

```
lookatme.render.markdown_inline.placeholder ()
```

The starting point of the rendering. The final result will be this returned list with all inline markdown tokens translated into urwid objects

```
lookatme.render.markdown_inline.render_no_change (text)
```

Render inline markdown text with no changes

`lookatme.render.markdown_inline.strikethrough(text, old_styles)`

Renders strikethrough text (~~text~~)

### Returns

list of `urwid Text` markup tuples.

`lookatme.render.markdown_inline.text(text)`

Renders plain text (does nothing)

### Returns

list of `urwid Text` markup tuples.

## lookatme.render.pygments module

Pygments related rendering

**class** `lookatme.render.pygments.UrwidFormatter(**options)`

Bases: `pygments.formatter.Formatter`

Formatter that returns [(text,attrspec), ...], where text is a piece of text, and attrspec is an `urwid.AttrSpec`

**classmethod** `findclosest(colstr, colors=256)`

Takes a hex string and finds the nearest color to it.

Returns a string urwid will recognize.

**findclosestattr** (*fgcolstr=None, bgcolstr=None, othersettings="", colors=256*)

Takes two hex colstring (e.g. 'ff00dd') and returns the nearest urwid style.

**format** (*tokensource, outfile*)

Format *tokensource*, an iterable of (*tokentype*, *tokenstring*) tuples and write it into *outfile*.

**formatgenerator** (*tokensource*)

Takes a token source, and generates (*tokenstring*, `urwid.AttrSpec`) pairs

**style**

`lookatme.render.pygments.get_formatter(style_name)`

`lookatme.render.pygments.get_lexer(lang, default='text')`

`lookatme.render.pygments.get_style(style_name)`

`lookatme.render.pygments.render_text(text, lang='text', style_name=None, plain=False)`

Render the provided text with the pygments renderer

## Module contents

### lookatme.themes package

#### Submodules

### lookatme.themes.dark module

Defines styles that should look good on dark backgrounds



## lookatme.themes.light module

### Module contents

Defines the built-in styles for lookatme

`lookatme.themes.ensure_defaults(mod) → Dict[str, Any]`  
Ensure that all required attributes exist within the provided module

## lookatme.widgets package

### Submodules

#### lookatme.widgets.clickable\_text module

This module contains code for ClickableText

```
class lookatme.widgets.clickable_text.ClickableText(markup, align='left',
                                                    wrap='space', layout=None)
    Bases: urwid.widget.Text
    Allows clickable/changing text to be part of the Text() contents
    mouse_event (size, event, button, x, y, focus)
        Handle mouse events!
    signals = ['click', 'change']
class lookatme.widgets.clickable_text.LinkIndicatorSpec(link_label, link_target,
                                                         orig_spec)
    Bases: urwid.display_common.AttrSpec
    Used to track a link within an urwid.Text instance
```

#### lookatme.widgets.table module

Defines a basic Table widget for urwid

```
class lookatme.widgets.table.Table(rows, headers=None, aligns: Optional[List[str]] = None)
    Bases: urwid.container.Pile
    Create a table from a list of headers, alignment values, and rows.
    calc_column_maxes()
    create_cells(body_rows, modifier=None)
        Create the rows for the body, optionally calling a modifier function on each created cell Text. The modifier
        must accept an urwid.Text object and must return an urwid.Text object.
    render(size, focus=False)
        Do whatever needs to be done to render the table
    set_column_maxes()
        Calculate and set the column maxes for this table
    signals = ['change']
    watch(w)
        Watch the provided widget w for changes
```

## Module contents

### Submodules

#### lookatme.ascii\_art module

Misc ASCII art

#### lookatme.config module

Config module for lookatme

`lookatme.config.get_log()` → `logging.Logger`

`lookatme.config.get_style()` → `Dict[KT, VT]`

`lookatme.config.get_style_with_precedence` (*theme\_mod: module, direct\_overrides: Dict[KT, VT], style\_override: str*) → `Dict[str, Any]`

Return the resulting style dict from the provided override values.

`lookatme.config.set_global_style_with_precedence` (*theme\_mod, direct\_overrides, style\_override*) → `Dict[str, Any]`

Set the `lookatme.config.STYLE` value based on the provided override values

#### lookatme.exceptions module

Exceptions used within lookatme

**exception** `lookatme.exceptions.IgnoredByContrib`

Bases: `Exception`

Raised when a contrib module's function chooses to ignore the function call.

#### lookatme.log module

Logging module

`lookatme.log.create_log(log_path)`

Create a new log that writes to `log_path`

`lookatme.log.create_null_log()`

Create a logging object that does nothing

#### lookatme.parser module

This module defines the parser for the markdown presentation file

**class** `lookatme.parser.Parser` (*single\_slide=False*)

Bases: `object`

A parser for markdown presentation files

**parse** (*input\_data*)

Parse the provided input data into a `Presentation` object

**Parameters** `input_data` (*str*) – The input markdown presentation to parse

**Returns** Presentation

**parse\_meta** (*input\_data*) → Tuple[AnyStr, Dict[KT, VT]]

Parse the PresentationMeta out of the input data

**Parameters** `input_data` (*str*) – The input data string

**Returns** tuple of (remaining\_data, meta)

**parse\_slides** (*meta*, *input\_data*)

Parse the Slide out of the input data

**Parameters**

- **meta** (*dict*) – The parsed meta values
- **input\_data** (*str*) – The input data string

**Returns** tuple of (remaining\_data, slide)

`lookatme.parser.is_progressive_slide_delimiter_token` (*token*)

Returns True if the token indicates the end of a progressive slide

**Parameters** `token` (*dict*) – The markdown token

**Returns** True if the token is a progressive slide delimiter

## lookatme.pres module

Defines Presentation specific objects

```
class lookatme.pres.Presentation (input_stream,          theme,          style_override=None,
                                live_reload=False,        single_slide=False,
                                preload_extensions=None, safe=False, no_ext_warn=False,
                                ignore_ext_failure=False)
```

Bases: object

Defines a presentation

**get\_tui** () → lookatme.tui.MarkdownTui

**reload** (*data=None*)

Reload this presentation

**Parameters** `data` (*str*) – The data to render for this slide deck (optional)

**reload\_watcher** ()

Watch for changes to the input filename, automatically reloading when the modified time has changed.

**run** (*start\_slide=0*)

Run the presentation!

**warn\_exts** (*exts*)

Warn about source-provided extensions that are to-be-loaded

## lookatme.prompt module

Basic user-prompting helper functions

`lookatme.prompt.yes` (*msg*)

Prompt the user for a yes/no answer. Returns bool

## lookatme.schemas module

Defines all schemas used in lookatme

```
class lookatme.schemas.BlockQuoteSchema(*, only: types.StrSequenceOrSet | None =
                                         None, exclude: types.StrSequenceOrSet = (),
                                         many: bool = False, context: dict | None =
                                         None, load_only: types.StrSequenceOrSet = (),
                                         dump_only: types.StrSequenceOrSet = (), partial:
                                         bool | types.StrSequenceOrSet = False, unknown:
                                         str | None = None)

    Bases: marshmallow.schema.Schema

    opts = <marshmallow.schema.SchemaOpts object>

class lookatme.schemas.BulletsSchema(*, only: types.StrSequenceOrSet | None = None,
                                         exclude: types.StrSequenceOrSet = (), many:
                                         bool = False, context: dict | None = None,
                                         load_only: types.StrSequenceOrSet = (), dump_only:
                                         types.StrSequenceOrSet = (), partial: bool |
                                         types.StrSequenceOrSet = False, unknown: str |
                                         None = None)

    Bases: marshmallow.schema.Schema

    class Meta
        Bases: object

        include = {'1': <fields.String(dump_default='•', attribute=None, validate=None, re

        opts = <marshmallow.schema.SchemaOpts object>

class lookatme.schemas.HeadingStyleSchema(*, only: types.StrSequenceOrSet | None =
                                         None, exclude: types.StrSequenceOrSet = (),
                                         many: bool = False, context: dict | None =
                                         None, load_only: types.StrSequenceOrSet = (),
                                         dump_only: types.StrSequenceOrSet = (), partial:
                                         bool | types.StrSequenceOrSet = False, unknown:
                                         str | None = None)

    Bases: marshmallow.schema.Schema

    opts = <marshmallow.schema.SchemaOpts object>

class lookatme.schemas.HeadingsSchema(*, only: types.StrSequenceOrSet | None = None,
                                         exclude: types.StrSequenceOrSet = (), many:
                                         bool = False, context: dict | None = None,
                                         load_only: types.StrSequenceOrSet = (), dump_only:
                                         types.StrSequenceOrSet = (), partial: bool |
                                         types.StrSequenceOrSet = False, unknown: str |
                                         None = None)

    Bases: marshmallow.schema.Schema

    class Meta
        Bases: object

        include = {'1': <fields.Nested(dump_default={'fg': '#9fc,bold', 'bg': 'default',

        opts = <marshmallow.schema.SchemaOpts object>
```

```
class lookatme.schemas.HruleSchema(*, only: types.StrSequenceOrSet | None = None, exclude:
    types.StrSequenceOrSet = (), many: bool = False, context:
    dict | None = None, load_only: types.StrSequenceOrSet =
    (), dump_only: types.StrSequenceOrSet = (), partial: bool
    | types.StrSequenceOrSet = False, unknown: str | None =
    None)
```

Bases: `marshmallow.schema.Schema`

**opts** = `<marshmallow.schema.SchemaOpts object>`

```
class lookatme.schemas.MetaSchema(*, only: types.StrSequenceOrSet | None = None, exclude:
    types.StrSequenceOrSet = (), many: bool = False, context:
    dict | None = None, load_only: types.StrSequenceOrSet =
    (), dump_only: types.StrSequenceOrSet = (), partial: bool
    | types.StrSequenceOrSet = False, unknown: str | None =
    None)
```

Bases: `marshmallow.schema.Schema`

The schema for presentation metadata

```
class Meta
```

Bases: `object`

**render\_module**

alias of `YamlRender`

**unknown** = `'include'`

```
dump(*args, **kwargs) → Dict[KT, VT]
```

Serialize an object to native Python data types according to this Schema's fields.

**Parameters**

- **obj** – The object to serialize.
- **many** – Whether to serialize *obj* as a collection. If *None*, the value for *self.many* is used.

**Returns** Serialized data

New in version 1.0.0.

Changed in version 3.0.0b7: This method returns the serialized data rather than a `(data, errors)` tuple. A `ValidationError` is raised if *obj* is invalid.

Changed in version 3.0.0rc9: Validation no longer occurs upon serialization.

```
load(*args, **kwargs) → Dict[KT, VT]
```

Deserialize a data structure to an object defined by this Schema's fields.

**Parameters**

- **data** – The data to deserialize.
- **many** – Whether to deserialize *data* as a collection. If *None*, the value for *self.many* is used.
- **partial** – Whether to ignore missing fields and not require any fields declared. Propagates down to `Nested` fields as well. If its value is an iterable, only missing fields listed in that iterable will be ignored. Use dot delimiters to specify nested fields.
- **unknown** – Whether to exclude, include, or raise an error for unknown fields in the data. Use `EXCLUDE`, `INCLUDE` or `RAISE`. If *None*, the value for *self.unknown* is used.

**Returns** Deserialized data

New in version 1.0.0.

Changed in version 3.0.0b7: This method returns the deserialized data rather than a `(data, errors)` tuple. A `ValidationError` is raised if invalid data are passed.

**load\_partial\_styles** (\*args, \*\*kwargs)

**loads** (\*args, \*\*kwargs) → Dict[KT, VT]

Same as `load()`, except it takes a JSON string as input.

#### Parameters

- **json\_data** – A JSON string of the data to deserialize.
- **many** – Whether to deserialize *obj* as a collection. If *None*, the value for *self.many* is used.
- **partial** – Whether to ignore missing fields and not require any fields declared. Propagates down to *Nested* fields as well. If its value is an iterable, only missing fields listed in that iterable will be ignored. Use dot delimiters to specify nested fields.
- **unknown** – Whether to exclude, include, or raise an error for unknown fields in the data. Use *EXCLUDE*, *INCLUDE* or *RAISE*. If *None*, the value for *self.unknown* is used.

**Returns** Deserialized data

New in version 1.0.0.

Changed in version 3.0.0b7: This method returns the deserialized data rather than a `(data, errors)` tuple. A `ValidationError` is raised if invalid data are passed.

**loads\_partial\_styles** (\*args, \*\*kwargs)

**opts** = <marshmallow.schema.SchemaOpts object>

**class** lookatme.schemas.NoDatesSafeLoader (stream)

Bases: `yaml.loader.SafeLoader`

**classmethod** **remove\_implicit\_resolver** (tag\_to\_remove)

Remove implicit resolvers for a particular tag

Takes care not to modify resolvers in super classes.

We want to load datetimes as strings, not dates, because we go on to serialise as json which doesn't have the advanced types of yaml, and leads to incompatibilities down the track.

**yaml\_implicit\_resolvers** = {'': [('tag:yaml.org,2002:null', re.compile('^(?: ~\n |nul

```
class lookatme.schemas.NumberingSchema(*, only: types.StrSequenceOrSet | None =
    None, exclude: types.StrSequenceOrSet = (),
    many: bool = False, context: dict | None =
    None, load_only: types.StrSequenceOrSet = (),
    dump_only: types.StrSequenceOrSet = (), partial:
    bool | types.StrSequenceOrSet = False, unknown:
    str | None = None)
```

Bases: `marshmallow.schema.Schema`

**class** **Meta**

Bases: `object`

**include** = {'1': <fields.String(dump\_default='numeric', attribute=None, validate=<C

**opts** = <marshmallow.schema.SchemaOpts object>

```

class lookatme.schemas.SpacingSchema(*, only: types.StrSequenceOrSet | None = None,
                                     exclude: types.StrSequenceOrSet = (), many:
                                     bool = False, context: dict | None = None,
                                     load_only: types.StrSequenceOrSet = (), dump_only:
                                     types.StrSequenceOrSet = (), partial: bool |
                                     types.StrSequenceOrSet = False, unknown: str |
                                     None = None)

    Bases: marshmallow.schema.Schema

    opts = <marshmallow.schema.SchemaOpts object>

class lookatme.schemas.StyleFieldSchema(*, only: types.StrSequenceOrSet | None =
                                         None, exclude: types.StrSequenceOrSet = (),
                                         many: bool = False, context: dict | None =
                                         None, load_only: types.StrSequenceOrSet = (),
                                         dump_only: types.StrSequenceOrSet = (), partial:
                                         bool | types.StrSequenceOrSet = False, unknown:
                                         str | None = None)

    Bases: marshmallow.schema.Schema

    opts = <marshmallow.schema.SchemaOpts object>

class lookatme.schemas.StyleSchema(*, only: types.StrSequenceOrSet | None = None, exclude:
                                     types.StrSequenceOrSet = (), many: bool = False, context:
                                     dict | None = None, load_only: types.StrSequenceOrSet =
                                     (), dump_only: types.StrSequenceOrSet = (), partial: bool
                                     | types.StrSequenceOrSet = False, unknown: str | None =
                                     None)

    Bases: marshmallow.schema.Schema

    Styles schema for themes and style overrides within presentations

    class Meta
        Bases: object

        render_module
            alias of YamlRender

        unknown = 'raise'

    opts = <marshmallow.schema.SchemaOpts object>

class lookatme.schemas.TableSchema(*, only: types.StrSequenceOrSet | None = None, exclude:
                                     types.StrSequenceOrSet = (), many: bool = False, context:
                                     dict | None = None, load_only: types.StrSequenceOrSet =
                                     (), dump_only: types.StrSequenceOrSet = (), partial: bool
                                     | types.StrSequenceOrSet = False, unknown: str | None =
                                     None)

    Bases: marshmallow.schema.Schema

    opts = <marshmallow.schema.SchemaOpts object>

class lookatme.schemas.YamlRender
    Bases: object

    static dumps(data)

    static loads(data)

```

## lookatme.slide module

Slide info holder

```
class lookatme.slide.Slide (tokens, number=0)
    Bases: object
```

This class defines a single slide. It operates on mistune's lexed tokens from the input markdown

## lookatme.tui module

This module defines the text user interface (TUI) for lookatme

```
class lookatme.tui.MarkdownTui (pres, start_idx=0)
    Bases: urwid.container.Frame
```

```
    keypress (size, key)
        Handle keypress events
```

```
    prep_pres (pres, start_idx=0)
        Prepare the presentation for displaying/use
```

```
    reload ()
        Reload the input, keeping the current slide in focus
```

```
    run ()
```

```
    update ()
```

```
    update_body ()
        Render the provided slide body
```

```
    update_creation ()
        Update the author and date
```

```
    update_slide_num ()
        Update the slide number
```

```
    update_slide_settings ()
        Update the slide margins and paddings
```

```
    update_title ()
        Update the title
```

```
class lookatme.tui.SlideRenderer (loop)
    Bases: threading.Thread
```

```
    daemon = True
```

```
    do_render (to_render, slide_num)
        Perform the actual rendering of a slide. This is done by:
        

- parsing the slide into tokens (should have occurred already)
- iterating through each parsed markdown token
- calling the appropriately-named render function for the token["type"] in lookatme.render.markdown_block

```

Each render function must have the signature:

```
def render_XXX(token, body, stack, loop):
    pass
```



The arguments to the render function are described below:

- `token` - the lexed markdown token - a dictionary
- `body` - the current `urwid.Pile()` that return values will be added to (same as `stack[-1]`)
- `stack` - The stack of `urwid.Pile()` used during rendering. E.g., when rendering nested lists, each nested list will push a new `urwid.Pile()` to the stack, each wrapped with its own additional indentation.
- `loop` - the `urwid.MainLoop` instance being used by lookatme. This won't usually be used, but is available if needed.

Main render functions (those defined in `markdown_block.py`) may have three types of return values:

- `None` - nothing is added to `stack[-1]`. Perhaps the render function only needed to add additional indentation by pushing a new `urwid.Pile()` to the stack.
- `list(urwid.Widget)` - A list of widgets to render. These will automatically be added to the `Pile` at `stack[-1]`
- `urwid.Widget` - A single widget to render. Will be added to `stack[-1]` automatically.

**flush\_cache()**

Clea everything out of the queue and the cache.

**queue\_render(*slide*)**

Queue up a slide to be rendered.

**render\_slide(*slide*, *force=False*)**

Render a slide, blocking until the slide completes. If `force` is `True`, rerender the slide even if it is in the cache.

**run()**

Run the main render thread

**stop()**

`lookatme.tui.create_tui(pres, start_slide=0)`

Run the provided presentation

**Parameters** `start_slide(int)` - 0-based slide index

`lookatme.tui.root_urwid_widget(to_wrap)`

This function is overridable by contrib extensions that need to specify the root urwid widget.

The return value *must* return either the `to_wrap` widget itself, or another widget that wraps the provided `to_wrap` widget.

`lookatme.tui.text(style, data, align='left')`

## lookatme.tutorial module

Functions and sources for the markdown tutorial slides!

**class** `lookatme.tutorial.Tutor` (*name: str*, *group: str*, *slides\_md: str*, *impl\_fn: Callable*, *order: int*, *lazy\_formatting: Optional[Callable] = None*)

Bases: `object`

A class to handle/process tutorials for specific functionality

In addition to name, group, and slides content of the tutor, each Tutor must also be associated with the implementation.

**get\_md** (*rendered\_example=True*) → str

Get the tutor's markdown text after resolving any special markup contained in it.

**opts. Dict[str, Any]** slides. Current can include ‘{

`lookatme.tutorial.get_tutorial_help()` → str

`lookatme.tutorial.get_tutorial_md(groups_or_tutors: List[str])` → Union[None, str]

`lookatme.tutorial.get_tutors(group_or_tutor: str)` → List[lookatme.tutorial.Tutor]

`lookatme.tutorial.pretty_close_match(str1, str2)`

`lookatme.tutorial.print_tutorial_help()`

`lookatme.tutorial.tutor(group: str, name: str, slides_md: str, order: int = 99999, lazy_formatting: Optional[Callable] = None)`

Define tutorial slides by using this as a decorator on a function!

## lookatme.utils module

`lookatme.utils.can_style_item(item)`

Return true/false if `style_text` can work with the given item

`lookatme.utils.dict_deep_update(to_update, new_vals)`

Deeply update the `to_update` dict with the `new_vals`

`lookatme.utils.flatten_text(text, new_spec=None)`

Return a flattend list of tuples that can be used as the first argument to a new `urwid.Text()`.

### Parameters

- **text** (*urwid.Text*) – The text to flatten
- **new\_spec** (*urwid.AttrSpec*) – A new spec to merge with existing styles

**Returns** list of tuples

`lookatme.utils.get_fg_bg_styles(style)`

`lookatme.utils.int_to_roman(integer)`

`lookatme.utils.listbox_add(listbox, widgets)`

`lookatme.utils.override_spec(orig_spec, new_spec)`

`lookatme.utils.pile_add(pile, widgets)`

`lookatme.utils.pile_or_listbox_add(container, widgets)`

Add the widget/widgets to the container

`lookatme.utils.prefix_text(text: str, prefix: str, split: str = '\n')` → str

`lookatme.utils.resolve_bag_of_text_markup_or_widgets(items)`

Resolve the list of items into either contiguous `urwid.Text()` instances, or pre-existing `urwid.Widget` objects

`lookatme.utils.row_text(rendered_row)`

Return all text joined together from the rendered row

`lookatme.utils.spec_from_style(styles)`

Create an `urwid.AttrSpec` from a {fg: "", bg: ""} style dict. If `styles` is a string, it will be used as the foreground

`lookatme.utils.styled_text(text, new_styles, old_styles=None, supplement_style=False)`

Return a styled text tuple that can be used within `urwid.Text`.

---

**Note:** If an `urwid.Text` instance is passed in as the `text` parameter, alignment values will be lost and must be explicitly re-added by the caller.

---

## Module contents



## CHAPTER 3

---

### Indices and tables

---

- `genindex`
- `modindex`
- `search`



### I

- `lookatme`, 39
- `lookatme.ascii_art`, 30
- `lookatme.config`, 30
- `lookatme.contrib`, 23
- `lookatme.contrib.file_loader`, 20
- `lookatme.contrib.terminal`, 21
- `lookatme.exceptions`, 30
- `lookatme.log`, 30
- `lookatme.parser`, 30
- `lookatme.pres`, 31
- `lookatme.prompt`, 31
- `lookatme.render`, 28
- `lookatme.render.asciinema`, 24
- `lookatme.render.markdown_block`, 24
- `lookatme.render.markdown_inline`, 26
- `lookatme.render.pygments`, 28
- `lookatme.schemas`, 32
- `lookatme.slide`, 36
- `lookatme.themes`, 29
- `lookatme.themes.dark`, 28
- `lookatme.themes.light`, 29
- `lookatme.tui`, 36
- `lookatme.tutorial`, 37
- `lookatme.utils`, 38
- `lookatme.widgets`, 30
- `lookatme.widgets.clickable_text`, 29
- `lookatme.widgets.table`, 29





**A**

`autolink()` (in module `lookatme.render.markdown_inline`), 26

**B**

`BlockQuoteSchema` (class in `lookatme.schemas`), 32

`BulletsSchema` (class in `lookatme.schemas`), 32

`BulletsSchema.Meta` (class in `lookatme.schemas`), 32

**C**

`calc_column_maxes()` (`lookatme.widgets.table.Table` method), 29

`can_style_item()` (in module `lookatme.utils`), 38

`ClickableText` (class in `lookatme.widgets.clickable_text`), 29

`codespan()` (in module `lookatme.render.markdown_inline`), 26

`contrib_first()` (in module `lookatme.contrib`), 23

`create_cells()` (`lookatme.widgets.table.Table` method), 29

`create_log()` (in module `lookatme.log`), 30

`create_null_log()` (in module `lookatme.log`), 30

`create_tui()` (in module `lookatme.tui`), 37

**D**

`daemon` (`lookatme.tui.SlideRenderer` attribute), 36

`dict_deep_update()` (in module `lookatme.utils`), 38

`do_render()` (`lookatme.tui.SlideRenderer` method), 36

`double_emphasis()` (in module `lookatme.render.markdown_inline`), 26

`dump()` (`lookatme.schemas.MetaSchema` method), 33

`dumps()` (`lookatme.contrib.file_loader.YamlRender` static method), 21

`dumps()` (`lookatme.contrib.terminal.YamlRender` static method), 23

`dumps()` (`lookatme.schemas.YamlRender` static method), 35

**E**

`emphasis()` (in module `lookatme.render.markdown_inline`), 27

`ensure_defaults()` (in module `lookatme.themes`), 29

`escape()` (in module `lookatme.render.markdown_inline`), 27

`expanded_styles()` (in module `lookatme.render.markdown_inline`), 27

**F**

`FileSchema` (class in `lookatme.contrib.file_loader`), 20

`FileSchema.Meta` (class in `lookatme.contrib.file_loader`), 20

`findclosest()` (`lookatme.render.pygments.UrwidFormatter` class method), 28

`findclosestattr()` (`lookatme.render.pygments.UrwidFormatter` method), 28

`flatten_text()` (in module `lookatme.utils`), 38

`flush_cache()` (`lookatme.tui.SlideRenderer` method), 37

`footnote_ref()` (in module `lookatme.render.markdown_inline`), 27

`format()` (`lookatme.render.pygments.UrwidFormatter` method), 28

`formatgenerator()` (`lookatme.render.pygments.UrwidFormatter` method), 28

**G**

`get_fg_bg_styles()` (in module `lookatme.utils`), 38

`get_formatter()` (in module `lookatme.render.pygments`), 28

`get_lexer()` (in module `lookatme.render.pygments`), 28

`get_log()` (in module `lookatme.config`), 30

`get_md()` (`lookatme.tutorial.Tutor` method), 37

`get_style()` (in module `lookatme.config`), 30

`get_style()` (in module *lookatme.render.pygments*), 28  
`get_style_with_precedence()` (in module *lookatme.config*), 30  
`get_tui()` (*lookatme.pres.Presentation* method), 31  
`get_tutorial_help()` (in module *lookatme.tutorial*), 38  
`get_tutorial_md()` (in module *lookatme.tutorial*), 38  
`get_tutors()` (in module *lookatme.tutorial*), 38

## H

*HeadingsSchema* (class in *lookatme.schemas*), 32  
*HeadingsSchema.Meta* (class in *lookatme.schemas*), 32  
*HeadingStyleSchema* (class in *lookatme.schemas*), 32  
*HruleSchema* (class in *lookatme.schemas*), 32

## I

*IgnoredByContrib*, 30  
`image()` (in module *lookatme.render.markdown\_inline*), 27  
`include` (*lookatme.schemas.BulletsSchema.Meta* attribute), 32  
`include` (*lookatme.schemas.HeadingsSchema.Meta* attribute), 32  
`include` (*lookatme.schemas.NumberingSchema.Meta* attribute), 34  
`inline_html()` (in module *lookatme.render.markdown\_inline*), 27  
`int_to_roman()` (in module *lookatme.utils*), 38  
`is_progressive_slide_delimiter_token()` (in module *lookatme.parser*), 31

## K

`keypress()` (*lookatme.tui.MarkdownTui* method), 36

## L

`linebreak()` (in module *lookatme.render.markdown\_inline*), 27  
*LineRange* (class in *lookatme.contrib.file\_loader*), 21  
`link()` (in module *lookatme.render.markdown\_inline*), 27  
*LinkIndicatorSpec* (class in *lookatme.widgets.clickable\_text*), 29  
`listbox_add()` (in module *lookatme.utils*), 38  
`load()` (*lookatme.contrib.file\_loader.FileSchema* method), 20  
`load()` (*lookatme.contrib.terminal.TerminalExSchema* method), 22  
`load()` (*lookatme.schemas.MetaSchema* method), 33  
`load_contribs()` (in module *lookatme.contrib*), 23

`load_partial_styles()` (*lookatme.schemas.MetaSchema* method), 34  
`loads()` (*lookatme.contrib.file\_loader.FileSchema* method), 21  
`loads()` (*lookatme.contrib.file\_loader.YamlRender* static method), 21  
`loads()` (*lookatme.contrib.terminal.TerminalExSchema* method), 22  
`loads()` (*lookatme.contrib.terminal.YamlRender* static method), 23  
`loads()` (*lookatme.schemas.MetaSchema* method), 34  
`loads()` (*lookatme.schemas.YamlRender* static method), 35  
`loads_partial_styles()` (*lookatme.schemas.MetaSchema* method), 34  
*lookatme* (module), 39  
*lookatme.ascii\_art* (module), 30  
*lookatme.config* (module), 30  
*lookatme.contrib* (module), 23  
*lookatme.contrib.file\_loader* (module), 20  
*lookatme.contrib.terminal* (module), 21  
*lookatme.exceptions* (module), 30  
*lookatme.log* (module), 30  
*lookatme.parser* (module), 30  
*lookatme.pres* (module), 31  
*lookatme.prompt* (module), 31  
*lookatme.render* (module), 28  
*lookatme.render.asciinema* (module), 24  
*lookatme.render.markdown\_block* (module), 24  
*lookatme.render.markdown\_inline* (module), 26  
*lookatme.render.pygments* (module), 28  
*lookatme.schemas* (module), 32  
*lookatme.slide* (module), 36  
*lookatme.themes* (module), 29  
*lookatme.themes.dark* (module), 28  
*lookatme.themes.light* (module), 29  
*lookatme.tui* (module), 36  
*lookatme.tutorial* (module), 37  
*lookatme.utils* (module), 38  
*lookatme.widgets* (module), 30  
*lookatme.widgets.clickable\_text* (module), 29  
*lookatme.widgets.table* (module), 29

## M

*MarkdownTui* (class in *lookatme.tui*), 36  
*MetaSchema* (class in *lookatme.schemas*), 33  
*MetaSchema.Meta* (class in *lookatme.schemas*), 33  
`mouse_event()` (*lookatme.widgets.clickable\_text.ClickableText* method), 29

## N

NoDatesSafeLoader (class in lookatme.schemas), 34  
 NumberingSchema (class in lookatme.schemas), 34  
 NumberingSchema.Meta (class in lookatme.schemas), 34

## O

opts (lookatme.contrib.file\_loader.FileSchema attribute), 21  
 opts (lookatme.contrib.file\_loader.LineRange attribute), 21  
 opts (lookatme.contrib.terminal.TerminalExSchema attribute), 23  
 opts (lookatme.schemas.BlockQuoteSchema attribute), 32  
 opts (lookatme.schemas.BulletsSchema attribute), 32  
 opts (lookatme.schemas.HeadingsSchema attribute), 32  
 opts (lookatme.schemas.HeadingStyleSchema attribute), 32  
 opts (lookatme.schemas.HruleSchema attribute), 33  
 opts (lookatme.schemas.MetaSchema attribute), 34  
 opts (lookatme.schemas.NumberingSchema attribute), 34  
 opts (lookatme.schemas.SpacingSchema attribute), 35  
 opts (lookatme.schemas.StyleFieldSchema attribute), 35  
 opts (lookatme.schemas.StyleSchema attribute), 35  
 opts (lookatme.schemas.TableSchema attribute), 35  
 overwrite\_spec() (in module lookatme.utils), 38

## P

parse() (lookatme.parser.Parser method), 30  
 parse\_meta() (lookatme.parser.Parser method), 31  
 parse\_slides() (lookatme.parser.Parser method), 31  
 Parser (class in lookatme.parser), 30  
 pile\_add() (in module lookatme.utils), 38  
 pile\_or\_listbox\_add() (in module lookatme.utils), 38  
 placeholder() (in module lookatme.render.markdown\_inline), 27  
 prefix\_text() (in module lookatme.utils), 38  
 prep\_pres() (lookatme.tui.MarkdownTui method), 36  
 Presentation (class in lookatme.pres), 31  
 pretty\_close\_match() (in module lookatme.tutorial), 38  
 print\_tutorial\_help() (in module lookatme.tutorial), 38

## Q

queue\_render() (lookatme.tui.SlideRenderer method), 37

## R

reload() (lookatme.pres.Presentation method), 31

reload() (lookatme.tui.MarkdownTui method), 36  
 reload\_watcher() (lookatme.pres.Presentation method), 31  
 remove\_implicit\_resolver() (lookatme.schemas.NoDatesSafeLoader class method), 34  
 render() (lookatme.widgets.table.Table method), 29  
 render\_block\_quote\_end() (in module lookatme.render.markdown\_block), 24  
 render\_block\_quote\_start() (in module lookatme.render.markdown\_block), 24  
 render\_code() (in module lookatme.contrib.file\_loader), 21  
 render\_code() (in module lookatme.contrib.terminal), 23  
 render\_code() (in module lookatme.render.markdown\_block), 24  
 render\_heading() (in module lookatme.render.markdown\_block), 24  
 render\_hrule() (in module lookatme.render.markdown\_block), 25  
 render\_list\_end() (in module lookatme.render.markdown\_block), 25  
 render\_list\_item\_end() (in module lookatme.render.markdown\_block), 25  
 render\_list\_item\_start() (in module lookatme.render.markdown\_block), 25  
 render\_list\_start() (in module lookatme.render.markdown\_block), 25  
 render\_loose\_item\_start() (in module lookatme.render.markdown\_block), 25  
 render\_module (lookatme.contrib.file\_loader.FileSchema.Meta attribute), 20  
 render\_module (lookatme.contrib.terminal.TerminalExSchema.Meta attribute), 22  
 render\_module (lookatme.schemas.MetaSchema.Meta attribute), 33  
 render\_module (lookatme.schemas.StyleSchema.Meta attribute), 35  
 render\_newline() (in module lookatme.render.markdown\_block), 26  
 render\_no\_change() (in module lookatme.render.markdown\_inline), 27  
 render\_paragraph() (in module lookatme.render.markdown\_block), 26  
 render\_slide() (lookatme.tui.SlideRenderer method), 37  
 render\_table() (in module lookatme.render.markdown\_block), 26  
 render\_text() (in module lookatme.render.markdown\_block), 26  
 render\_text() (in module lookatme.render.pygments), 28  
 resolve\_bag\_of\_text\_markup\_or\_widgets()

*(in module lookatme.utils)*, 38  
`root_urwid_widget()` *(in module lookatme.tui)*, 37  
`row_text()` *(in module lookatme.utils)*, 38  
`run()` *(lookatme.pres.Presentation method)*, 31  
`run()` *(lookatme.tui.MarkdownTui method)*, 36  
`run()` *(lookatme.tui.SlideRenderer method)*, 37

## S

`set_column_maxes()` *(lookatme.widgets.table.Table method)*, 29  
`set_global_style_with_precedence()` *(in module lookatme.config)*, 30  
`shutdown()` *(in module lookatme.contrib.terminal)*, 23  
`shutdown_contribs()` *(in module lookatme.contrib)*, 23  
`signals` *(lookatme.widgets.clickable\_text.ClickableText attribute)*, 29  
`signals` *(lookatme.widgets.table.Table attribute)*, 29  
`Slide` *(class in lookatme.slide)*, 36  
`SlideRenderer` *(class in lookatme.tui)*, 36  
`SpacingSchema` *(class in lookatme.schemas)*, 34  
`spec_from_style()` *(in module lookatme.utils)*, 38  
`stop()` *(lookatme.tui.SlideRenderer method)*, 37  
`strikethrough()` *(in module lookatme.render.markdown\_inline)*, 27  
`style` *(lookatme.render.pygments.UrwidFormatter attribute)*, 28  
`styled_text()` *(in module lookatme.utils)*, 38  
`StyleFieldSchema` *(class in lookatme.schemas)*, 35  
`StyleSchema` *(class in lookatme.schemas)*, 35  
`StyleSchema.Meta` *(class in lookatme.schemas)*, 35

## T

`Table` *(class in lookatme.widgets.table)*, 29  
`TableSchema` *(class in lookatme.schemas)*, 35  
`TerminalExSchema` *(class in lookatme.contrib.terminal)*, 21  
`TerminalExSchema.Meta` *(class in lookatme.contrib.terminal)*, 22  
`text()` *(in module lookatme.render.markdown\_inline)*, 28  
`text()` *(in module lookatme.tui)*, 37  
`transform_data()` *(in module lookatme.contrib.file\_loader)*, 21  
`Tutor` *(class in lookatme.tutorial)*, 37  
`tutor()` *(in module lookatme.tutorial)*, 38

## U

`unknown` *(lookatme.schemas.MetaSchema.Meta attribute)*, 33  
`unknown` *(lookatme.schemas.StyleSchema.Meta attribute)*, 35  
`update()` *(lookatme.tui.MarkdownTui method)*, 36

`update_body()` *(lookatme.tui.MarkdownTui method)*, 36  
`update_creation()` *(lookatme.tui.MarkdownTui method)*, 36  
`update_slide_num()` *(lookatme.tui.MarkdownTui method)*, 36  
`update_slide_settings()` *(lookatme.tui.MarkdownTui method)*, 36  
`update_title()` *(lookatme.tui.MarkdownTui method)*, 36  
`UrwidFormatter` *(class in lookatme.render.pygments)*, 28  
`user_warnings()` *(in module lookatme.contrib.file\_loader)*, 21  
`user_warnings()` *(in module lookatme.contrib.terminal)*, 23

## V

`validate_extension_mod()` *(in module lookatme.contrib)*, 23

## W

`warn_exts()` *(lookatme.pres.Presentation method)*, 31  
`watch()` *(lookatme.widgets.table.Table method)*, 29

## Y

`yaml_implicit_resolvers` *(lookatme.schemas.NoDatesSafeLoader attribute)*, 34  
`YamlRender` *(class in lookatme.contrib.file\_loader)*, 21  
`YamlRender` *(class in lookatme.contrib.terminal)*, 23  
`YamlRender` *(class in lookatme.schemas)*, 35  
`yes()` *(in module lookatme.prompt)*, 31