

---

# lookatme

*Release v0.0.0*

Dec 30, 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 . . . . .	18
2.8	Builtin Extensions . . . . .	18
2.9	lookatme . . . . .	20
<b>3</b>	<b>Indices and tables</b>	<b>51</b>
	<b>Python Module Index</b>	<b>53</b>
	<b>Index</b>	<b>55</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 Support: Inline		
Markdown		Result
<code>*italic*</code>		<i>italic</i>
<code>_italic_</code>		<u>italic</u>
<code>**bold**</code>		<b>bold</b>
<code>__bold__</code>		<u>bold</u>
<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_paragraph_open`

- `render_paragraph_close`
- `render_inline`
- `render_ordered_list_open`
- `render_bullet_list_open`
- `render_list_open`
- `render_ordered_list_close`
- `render_bullet_list_close`
- `render_list_close`
- `render_list_item_open`
- `render_list_item_close`
- `render_heading_open`
- `render_heading_close`
- `render_blockquote_open`
- `render_blockquote_close`
- `render_fence`
- `render_table_open`
- `render_hr`
- `render_text`
- `render_em_open`
- `render_em_close`
- `render_strong_open`
- `render_strong_close`
- `render_s_open`
- `render_s_close`
- `render_link_open`
- `render_link_close`
- `render_image`
- `render_image_close`
- `render_hardbreak`
- `render_softbreak`
- `render_code_inline`
- `render_html_inline`
- `render_html_tag_default_open`
- `render_html_tag_default_close`
- `render_html_tag_u_open`
- `render_html_tag_i_open`

- `render_html_tag_b_open`
- `render_html_tag_em_open`
- `render_html_tag_blink_open`
- `render_html_tag_br_open`
- `render_html_tag_div_open`
- `render_html_tag_div_close`
- `render_html_tag_ol_open`
- `render_html_tag_ol_close`
- `render_html_tag_ul_open`
- `render_html_tag_ul_close`
- `render_html_tag_li_open`
- `render_html_tag_li_close`
- `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_fence(token: Dict[KT, VT], ctx:
                                             lookatme.render.context.Context)
```

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_fence(token, ctx: lookatme.render.context.Context)
```

```
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

### Subpackages

### lookatme.render.html package

### Submodules

### lookatme.render.html.templating module

This module provides basic templating functionality

```

lookatme.render.html.templating.comma_append(val, new_val)
lookatme.render.html.templating.css_filter(value: str) → str
lookatme.render.html.templating.get_filters() → Dict[KT, VT]
lookatme.render.html.templating.get_template_data(template_name: str) → str
    Return the template data or raise an error.
lookatme.render.html.templating.highlight_filter(value: str) → str
lookatme.render.html.templating.json_filter(value) → str
lookatme.render.html.templating.render(template_name: str, context: Dict[str, str]) → str
    Render the indicated template name using the provided context to resolve variables. An error will be thrown if
    variables are not resolved.

```

## Module contents

This module renders lookatme slides to HTML

```

class lookatme.render.html.HtmlContext
    Bases: object
    get_html()
    get_html_consumed()
    use_spec(spec: Optional[urwid.display_common.AttrSpec])
    use_tag(tag_name: str, classname: Optional[str] = None, id: Optional[str] = None, style: Optional[Dict[str, str]] = None, **other_attrs)
    write(content: str)

lookatme.render.html.canvas_to_html(ctx: lookatme.render.html.HtmlContext, canvas: urwid.canvas.Canvas, only_keep: Optional[str] = None)

lookatme.render.html.copy_tree_update(src: str, dst: str)
    Copy a directory from src to dst. For each directory within src, ensure that the directory exists, and then copy
    each individual file in. If other files exist in that same directory in dst, leave them alone.

lookatme.render.html.create_html_output(output_dir: str, slides_html: List[Tuple[str, str]], slides_titles: List[Tuple[str, Optional[urwid.canvas.Canvas]]],
                                         title_category_delim: str = ':')

```

## Submodules

### lookatme.render.asciinema module

### lookatme.render.context module

```

class lookatme.render.context.ContainerInfo(container: 'urwid.Widget', meta: 'Dict[str, Any]', source_token: 'Optional[Dict]')
    Bases: object

class lookatme.render.context.Context(loop: Optional[urwid.main_loop.MainLoop])
    Bases: object

```

**clean\_state\_snapshot()**

**clean\_state\_validate()**  
Validate that all stacks are empty, that everything unwound correctly

**clone()** → lookatme.render.context.Context  
Create a new limited clone of the current context.  
  
The only data that is actually cloned is the source and spec stacks.

**container**  
Return the current container

**container\_pop()** → lookatme.render.context.ContainerInfo  
Pop the last element off the stack. Returns the popped widget

**container\_push**(*new\_item: urwid.widget.Widget, is\_new\_block: bool, custom\_add: Optional[urwid.widget.Widget] = None*)  
Push to the stack and propagate metadata

**ensure\_new\_block()**  
Ensure that we are in a new block

**fake\_token**(*type: str, \*\*kwargs*) → Dict[KT, VT]

**get\_inline\_markup()**  
Return the current inline markup, ignoring any widgets that may have made it in

**get\_inline\_widgets()**  
Return the results of any inline rendering that has occurred in Widget form. The conversion here is necessary since inline rendering functions often produce urwid Text Markup instead of widgets, which need to be converted to ClickableText.

**inline\_clear()**  
Clear the inline rendered results

**inline\_convert\_all\_to\_widgets()**

**inline\_flush()**  
Add all inline widgets to the current container

**inline\_markup\_consumed**  
Return and clear the inline markup

**inline\_push**(*inline\_result: Union[urwid.widget.Widget, str, Tuple[Optional[urwid.display\_common.AttrSpec], str]]*)  
Push a new inline render result onto the stack. Either a widget, or text markup

**inline\_widgets\_consumed**  
Return and clear the inline widgets

**is\_literal**

**level\_inc()**

**log\_debug**(*msg*)

**meta**  
Return metadata associated with the current container, or None if the current container is None.

**source**  
Return the current markdown source

**source\_get\_token\_lines**(*token: Dict[KT, VT], extra: int = 5, with\_lines: bool = True, with\_marker: bool = True*) → List[str]

**source\_pop** () → str  
Pop the latest source off of the source stack

**source\_push** (new\_source: str)  
Push new markdown source onto the source stack

**spec\_general**  
Return the current fully resolved current AttrSpec

**spec\_peek** () → urwid.display\_common.AttrSpec  
Return the most recent spec, or None

**spec\_pop** ()  
Push a new AttrSpec onto the spec\_stack

**spec\_push** (new\_spec, text\_only=False)  
Push a new AttrSpec onto the spec\_stack

**spec\_text**

**spec\_text\_with** (other\_spec: Union[None, urwid.display\_common.AttrSpec]) → urwid.display\_common.AttrSpec

**tag**

**tag\_is\_ancestor** (ancestor\_tag\_name: str) → bool

**tag\_pop** ()  
Pop the most recent tag off of the tag stack

**tag\_push** (new\_tag: str, token: Dict[KT, VT], spec=None, text\_only\_spec=False)  
Push a new tag name onto the stack

**tag\_token**

**tokens**  
Return the current token iterator

**tokens\_pop** ()

**tokens\_push** (tokens: List[Dict[KT, VT]], inline: bool = False)

**unwind\_bookmark**

**unwind\_tokens**  
Generate a list of unwind (close) tokens from the token iterators in the stack

**unwind\_tokens\_consumed**  
Generate a list of unwind (close) tokens from the token iterators in the stack

**unwind\_tokens\_from** (bookmark: int) → List[Dict[KT, VT]]

**use\_container** (new\_container: urwid.widget.Widget, is\_new\_block: bool, custom\_add: Optional[urwid.widget.Widget] = None)  
Ensure that the container is pushed/popped correctly

**use\_container\_tmp** (new\_container: urwid.widget.Widget)  
Swap out the entire container stack for a new one with the new container as the only item, while keeping spec and tag stacks

**use\_spec** (new\_spec, text\_only=False)  
Ensure that specs are pushed/popped correctly

**use\_tokens** (tokens, inline: bool = False)  
Create a context manager for pushing/popping tokens via a with block

**widget\_add** (*w*: Union[List[urwid.widget.Widget], urwid.widget.Widget], *wrap*: Optional[bool] = False)

Add the provided widget to the current container.

**wrap\_widget** (*w*: urwid.widget.Widget, *spec*: Optional[urwid.display\_common.AttrSpec] = None) → urwid.widget.Widget

Wrap the provided widget with an AttrMap that will apply the current styling to the entire widget (using *spec\_general*)

## lookatme.render.markdown\_block module

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

**class** lookatme.render.markdown\_block.TableTokenExtractor

Bases: object

**curr\_parent**

**curr\_siblings**

**is\_table\_element\_close** (*token*: Dict[KT, VT]) → bool

**is\_table\_element\_open** (*token*: Dict[KT, VT]) → bool

**process\_token** (*token*: Dict[KT, VT])

**process\_tokens** (*tokens*: List[Dict[KT, VT]]) → Tuple[Optional[Dict[KT, VT]], Optional[Dict[KT, VT]]]

**set\_token\_alignment** (*token*: Dict[KT, VT])

lookatme.render.markdown\_block.**render** (*token*, *ctx*: lookatme.render.context.Context)

Render a single token

lookatme.render.markdown\_block.**render\_all** (*ctx*: lookatme.render.context.Context, *and\_unwind*: bool = False)

lookatme.render.markdown\_block.**render\_blockquote\_close** (*token*: Dict[KT, VT], *ctx*: lookatme.render.context.Context)

lookatme.render.markdown\_block.**render\_blockquote\_open** (*token*: Dict[KT, VT], *ctx*: lookatme.render.context.Context)

lookatme.render.markdown\_block.**render\_bullet\_list\_close** (*token*, *ctx*: lookatme.render.context.Context)

lookatme.render.markdown\_block.**render\_bullet\_list\_open** (*token*, *ctx*: lookatme.render.context.Context)

lookatme.render.markdown\_block.**render\_fence** (*token*: Dict[KT, VT], *ctx*: lookatme.render.context.Context)

Renders a code block using the Pygments library.

See [lookatme.tui.SlideRendererer.do\\_render](#) for additional argument and return value descriptions.

lookatme.render.markdown\_block.**render\_heading\_close** (*token*: Dict[KT, VT], *ctx*: lookatme.render.context.Context)

lookatme.render.markdown\_block.**render\_heading\_open** (*token*: Dict[KT, VT], *ctx*: lookatme.render.context.Context)

lookatme.render.markdown\_block.**render\_hr** (*token*, *ctx*: lookatme.render.context.Context)

Render a newline

See [lookatme.tui.SlideRendererer.do\\_render](#) for argument and return value descriptions.



```

lookatme.render.markdown_block.render_inline(token,          ctx:
                                              lookatme.render.context.Context)
lookatme.render.markdown_block.render_list_close(_,          ctx:
                                              lookatme.render.context.Context)
lookatme.render.markdown_block.render_list_item_close(_,      ctx:
                                              lookatme.render.context.Context)
lookatme.render.markdown_block.render_list_item_open(_,       ctx:
                                              lookatme.render.context.Context)
lookatme.render.markdown_block.render_list_open(token,        ctx:
                                              lookatme.render.context.Context,
                                              ordered: bool)
lookatme.render.markdown_block.render_ordered_list_close(token, ctx:
                                              lookatme.render.context.Context)
lookatme.render.markdown_block.render_ordered_list_open(token, ctx:
                                              lookatme.render.context.Context)
lookatme.render.markdown_block.render_paragraph_close(token,  ctx:
                                              lookatme.render.context.Context)
lookatme.render.markdown_block.render_paragraph_open(token,   ctx:
                                              lookatme.render.context.Context)
lookatme.render.markdown_block.render_table_open(token:       Dict[KT, VT], ctx:
                                              lookatme.render.context.Context)

```

### lookatme.render.markdown\_html module

This module attempts to parse basic html

```

class lookatme.render.markdown_html.Tag(tag_name: str, is_open: bool, attrs: Op-
                                         tional[Dict[str, str]] = None, style: Op-
                                         tional[Dict[str, str]] = None)

    Bases: object

    classmethod parse(text: str) → List[lookatme.render.markdown_html.Tag]
        Return a new Tag instance or None after parsing the text

    classmethod parse_style(style_contents)
        Parse the style contents

```

### lookatme.render.markdown\_inline module

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

```

lookatme.render.markdown_inline.markdown_block()

lookatme.render.markdown_inline.render(token, ctx: lookatme.render.context.Context)
    Render an inline token[""] These tokens come from “children” tokens of a block token[""]

lookatme.render.markdown_inline.render_all(ctx: lookatme.render.context.Context,
                                             and_unwind: bool = False)

lookatme.render.markdown_inline.render_code_inline(token,          ctx:
                                              lookatme.render.context.Context)

lookatme.render.markdown_inline.render_em_close(_,          ctx:
                                              lookatme.render.context.Context)

```

```
lookatme.render.markdown_inline.render_em_open (_, ctx: lookatme.render.context.Context)
lookatme.render.markdown_inline.render_hardbreak (_, ctx:
    lookatme.render.context.Context)
lookatme.render.markdown_inline.render_html_inline (token, ctx:
    lookatme.render.context.Context)
lookatme.render.markdown_inline.render_html_tag_b_open (token: Dict[KT, VT], tag:
    lookatme.render.markdown_html.Tag,
    ctx:
    lookatme.render.context.Context,
    style_spec: Optional[urwid.display_common.AttrSpec])
lookatme.render.markdown_inline.render_html_tag_blink_open (token: Dict[KT,
    VT], tag:
    lookatme.render.markdown_html.Tag,
    ctx:
    lookatme.render.context.Context,
    style_spec: Optional[urwid.display_common.AttrSpec])
lookatme.render.markdown_inline.render_html_tag_br_open (token: Dict[KT, VT], tag:
    lookatme.render.markdown_html.Tag,
    ctx:
    lookatme.render.context.Context,
    style_spec: Optional[urwid.display_common.AttrSpec])
lookatme.render.markdown_inline.render_html_tag_default_close (_, _tag:
    lookatme.render.markdown_html.Tag,
    ctx:
    lookatme.render.context.Context,
    _style_spec: Optional[urwid.display_common.AttrSpec])
lookatme.render.markdown_inline.render_html_tag_default_open (token: Dict[KT,
    VT], tag:
    lookatme.render.markdown_html.Tag,
    ctx:
    lookatme.render.context.Context,
    style_spec: Optional[urwid.display_common.AttrSpec])
lookatme.render.markdown_inline.render_html_tag_div_close (_, tag:
    lookatme.render.markdown_html.Tag,
    ctx:
    lookatme.render.context.Context,
    style_spec: Optional[urwid.display_common.AttrSpec])
lookatme.render.markdown_inline.render_html_tag_div_open (token: Dict[KT, VT], tag:
    lookatme.render.markdown_html.Tag,
    ctx:
    lookatme.render.context.Context,
    style_spec: Optional[urwid.display_common.AttrSpec])
```

```

lookatme.render.markdown_inline.render_html_tag_em_open(token: Dict[KT, VT], tag:
    lookatme.render.markdown_html.Tag,
    ctx:
    lookatme.render.context.Context,
    style_spec: Optional[urwid.display_common.AttrSpec])

lookatme.render.markdown_inline.render_html_tag_i_open(token: Dict[KT, VT], tag:
    lookatme.render.markdown_html.Tag,
    ctx:
    lookatme.render.context.Context,
    style_spec: Optional[urwid.display_common.AttrSpec])

lookatme.render.markdown_inline.render_html_tag_li_close(token: Dict[KT, VT], tag:
    lookatme.render.markdown_html.Tag,
    ctx:
    lookatme.render.context.Context,
    style_spec: Optional[urwid.display_common.AttrSpec])

lookatme.render.markdown_inline.render_html_tag_li_open(token: Dict[KT, VT], tag:
    lookatme.render.markdown_html.Tag,
    ctx:
    lookatme.render.context.Context,
    style_spec: Optional[urwid.display_common.AttrSpec])

lookatme.render.markdown_inline.render_html_tag_ol_close(_, tag:
    lookatme.render.markdown_html.Tag,
    ctx:
    lookatme.render.context.Context,
    style_spec: Optional[urwid.display_common.AttrSpec])

lookatme.render.markdown_inline.render_html_tag_ol_open(token: Dict[KT, VT], tag:
    lookatme.render.markdown_html.Tag,
    ctx:
    lookatme.render.context.Context,
    style_spec: Optional[urwid.display_common.AttrSpec])

lookatme.render.markdown_inline.render_html_tag_u_open(token, tag:
    lookatme.render.markdown_html.Tag,
    ctx:
    lookatme.render.context.Context,
    style_spec: Optional[urwid.display_common.AttrSpec])

lookatme.render.markdown_inline.render_html_tag_ul_close(_, tag:
    lookatme.render.markdown_html.Tag,
    ctx:
    lookatme.render.context.Context,
    style_spec: Optional[urwid.display_common.AttrSpec])

```

```
lookatme.render.markdown_inline.render_html_tag_ul_open (token: Dict[KT, VT], tag:
                                                         lookatme.render.markdown_html.Tag,
                                                         ctx:
                                                         lookatme.render.context.Context,
                                                         style_spec: Optional[urwid.display_common.AttrSpec])

lookatme.render.markdown_inline.render_image (token,                                     ctx:
                                              lookatme.render.context.Context)

lookatme.render.markdown_inline.render_image_close (token,                             ctx:
                                                     lookatme.render.context.Context)

lookatme.render.markdown_inline.render_link_close (_,                               ctx:
                                                    lookatme.render.context.Context)

lookatme.render.markdown_inline.render_link_open (token,                             ctx:
                                                    lookatme.render.context.Context)

lookatme.render.markdown_inline.render_s_close (_, ctx: lookatme.render.context.Context)

lookatme.render.markdown_inline.render_s_open (_, ctx: lookatme.render.context.Context)

lookatme.render.markdown_inline.render_softbreak (_,                               ctx:
                                                    lookatme.render.context.Context)

lookatme.render.markdown_inline.render_strong_close (_,                             ctx:
                                                      lookatme.render.context.Context)

lookatme.render.markdown_inline.render_strong_open (_,                             ctx:
                                                      lookatme.render.context.Context)

lookatme.render.markdown_inline.render_text (token, ctx: lookatme.render.context.Context)
```

## lookatme.render.token\_iterator module

This module defines the TokenIterator class.

```
class lookatme.render.token_iterator.TokenIterator (tokens: List[Dict[KT, VT]], un-
                                                    wind_tokens: List[Tuple[bool,
                                                    Dict[KT, VT]]], inline: bool =
                                                    False)
```

Bases: object

This clas

**at\_offset** (idx\_offset: int) → Optional[Dict[KT, VT]]

**at\_offset\_v** (idx\_offset: int) → Dict[KT, VT]

**curr**

**next** () → Optional[Dict[KT, VT]]

**peek** () → Optional[Dict[KT, VT]]

**unwind\_has\_type** (unwind\_type: str) → bool

## 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.utils package

## Submodules

### lookatme.utils.colors module

Color-related utilities

`lookatme.utils.colors.ensure_contrast(spec: urwid.display_common.AttrSpec)`  
`lookatme.utils.colors.get_highlight_color(bg_color: str, percent: float = 0.1) → str`  
`lookatme.utils.colors.increase_brightness(color: str, percent: float) → str`  
`lookatme.utils.colors.luminance(color: str) → float`

## Module contents

`lookatme.utils.can_style_item(item)`  
 Return true/false if style\_text can work with the given item

`lookatme.utils.check_token_type(token: Optional[Dict[KT, VT]], expected_type: str)`

`lookatme.utils.core_widget(w) → urwid.widget.Widget`  
 Resolve a wrapped widget to its core widget value

`lookatme.utils.debug_print_tokens(tokens, level=1)`  
 Print the tokens DFS

`lookatme.utils.dict_deep_update(to_update, new_vals)`  
 Deeply update the to\_update dict with the new\_vals

`lookatme.utils.extract_hexcolor(spec_style: str) → str`

`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** (*SmartAttrSpec*) – A new spec to merge with existing styles

**Returns** list of tuples

```
lookatme.utils.get_fg_bg_styles (style)
lookatme.utils.get_meta (item)
lookatme.utils.int_to_roman (integer)
lookatme.utils.listbox_add (listbox, widgets)
lookatme.utils.non_empty_split (data)
lookatme.utils.override_spec (orig_spec, new_spec)
lookatme.utils.override_style (orig_style: Dict[str, str], new_style: Dict[str, str]) → Dict[str, str]
lookatme.utils.packed_widget_width (w: urwid.widget.Widget) → int
    Return the smallest size of the widget without wrapping
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_stack (spec_stack: list, filter_fn=None) → urwid.display_common.AttrSpec
lookatme.utils.spec_from_style (styles)
    Create an SmartAttrSpec 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.

---

## lookatme.widgets package

### Submodules

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

This module contains code for ClickableText

```
class lookatme.widgets.clickable_text.ClickableText (markup, *args, **kwargs)
    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\_target, orig\_spec, link\_type: str = 'link'*)

Bases: *lookatme.widgets.smart\_attr\_spec.SmartAttrSpec*

Used to track a link within an urwid.Text instance

**new\_for\_spec** (*new\_spec*)

Create a new LinkIndicatorSpec with the same link information but new AttrSpec

## lookatme.widgets.codeblock module

This module defines an urwid Widget that renders a codeblock

**class** lookatme.widgets.codeblock.**CodeBlock** (*source: str, lang: str = 'text', style\_name: str = 'monokai', line\_numbers: bool = False, start\_line\_number: int = 1, hl\_lines: Optional[List[range]] = None, default\_fg: Optional[str] = None, bg\_override: Optional[str] = None*)

Bases: *urwid.container.Pile*

**class** lookatme.widgets.codeblock.**StyleCache** (*default\_fg: Optional[str] = None, bg\_override: Optional[str] = None*)

Bases: *object*

Caches the highlight styles for loaded pygments syntax highlighting styles.

**get\_style** (*style\_name: str*) → lookatme.widgets.codeblock.SyntaxHlStyle

Return the highlight style for the specified pygments style name. If the style name isn't found, the "text" style will be used instead.

**is\_valid\_style** (*style\_name: str*) → bool

Return whether the style name is a valid pygments style

**load\_style** (*style\_name: str*) → lookatme.widgets.codeblock.SyntaxHlStyle

**class** lookatme.widgets.codeblock.**SyntaxHlStyle** (*name: str, styles: Dict[str, lookatme.widgets.smart\_attr\_spec.SmartAttrSpec], pygments\_style: pygments.style.StyleMeta, default\_fg: str, bg\_override: Optional[str] = None*)

Bases: *object*

Stores urwid styles for each token type for a specific pygments syntax highlighting style.

**get\_line\_number\_spec** (*do\_hl: bool = False*) → lookatme.widgets.smart\_attr\_spec.SmartAttrSpec

**get\_style\_spec** (*token\_type: str, highlight: bool*) → lookatme.widgets.smart\_attr\_spec.SmartAttrSpec  
 Attempt to find the closest matching style for the provided token type.

lookatme.widgets.codeblock.**clear\_cache** ()

lookatme.widgets.codeblock.**get\_lexer** (*lang, default='text'*) → *pygments.lexer.Lexer*

lookatme.widgets.codeblock.**get\_style\_cache** (*default\_fg: Optional[str] = None, bg\_override: Optional[str] = None*) → lookatme.widgets.codeblock.StyleCache

```
lookatme.widgets.codeblock.supported_langs() → Set[str]
lookatme.widgets.codeblock.supported_styles() → Mapping[str, str]
lookatme.widgets.codeblock.tokens_to_lines(tokens) → List[List[Tuple[str, str]]]
lookatme.widgets.codeblock.tokens_to_markup(line: List[Tuple[str, str]], style:
lookatme.widgets.codeblock.SyntaxHlStyle,
do_hl: bool = False) →
List[Tuple[lookatme.widgets.smart_attr_spec.SmartAttrSpec,
str]]
```

## lookatme.widgets.fancy\_box module

```
class lookatme.widgets.fancy_box.FancyBox(w, tl_corner: str = '', tr_corner:
str = '', bl_corner: str = 'L',
br_corner: str = '', tl_corner_spec: Op-
tional[urwid.display_common.AttrSpec]
= None, tr_corner_spec: Op-
tional[urwid.display_common.AttrSpec]
= None, bl_corner_spec: Op-
tional[urwid.display_common.AttrSpec]
= None, br_corner_spec: Op-
tional[urwid.display_common.AttrSpec]
= None, t_fill: str = '-', b_fill: str = '-',
l_fill: str = '|', r_fill: str = '|', t_fill_spec:
Optional[urwid.display_common.AttrSpec]
= None, b_fill_spec: Op-
tional[urwid.display_common.AttrSpec]
= None, l_fill_spec: Op-
tional[urwid.display_common.AttrSpec]
= None, r_fill_spec: Op-
tional[urwid.display_common.AttrSpec]
= None, tight: bool = False)
```

Bases: `urwid.decoration.WidgetDecoration`, `urwid.widget.WidgetWrap`

Test

```
b_fill = '-'
```

```
bl_corner = '| \n \n —'
```

```
br_corner = '| \n \n —'
```

```
l_fill = '|'
```

```
pack(size, focus=False)
```

See `Widget.render()` for parameter details.

**Returns** A “packed” size (*maxcol*, *maxrow*) for this widget

Calculate and return a minimum size where all content could still be displayed. Fixed widgets must implement this method and return their size when `()` is passed as the *size* parameter.

This default implementation returns the *size* passed, or the *maxcol* passed and the value of `rows()` as the *maxrow* when (*maxcol*,) is passed as the *size* parameter.



---

**Note:** This is a new method that hasn't been fully implemented across the standard widget types. In particular it has not yet been implemented for container widgets.

---

Text widgets have implemented this method. You can use `Text.pack()` to calculate the minimum columns and rows required to display a text widget without wrapping, or call it iteratively to calculate the minimum number of columns required to display the text wrapped into a target number of rows.

```
r_fill = '|'

set (tl_corner: Optional[str] = None, tr_corner: Optional[str] = None, bl_corner:
    Optional[str] = None, br_corner: Optional[str] = None, tl_corner_spec:
    Optional[urwid.display_common.AttrSpec] = None, tr_corner_spec: Op-
    tional[urwid.display_common.AttrSpec] = None, bl_corner_spec: Op-
    tional[urwid.display_common.AttrSpec] = None, br_corner_spec: Op-
    tional[urwid.display_common.AttrSpec] = None, t_fill: Optional[str] = None, b_fill: Optional[str]
    = None, l_fill: Optional[str] = None, r_fill: Optional[str] = None)

t_fill = '-'

tl_corner = '—\n \n |'
tr_corner = '—\n \n |'
```

### lookatme.widgets.line\_fill module

```
class lookatme.widgets.line_fill.LineFill (beg_chars: str, fill_char: str,
    end_chars: str, beg_spec: Op-
    tional[urwid.display_common.AttrSpec]
    = None, fill_spec: Op-
    tional[urwid.display_common.AttrSpec]
    = None, end_spec: Op-
    tional[urwid.display_common.AttrSpec]
    = None, orientation: str = 'vertical')
```

Bases: `urwid.widget.Widget`

Test

**HORIZONTAL** = 'horizontal'

**VERTICAL** = 'vertical'

**pack** (*size: Tuple, focus: bool = False*)

See `Widget.render()` for parameter details.

**Returns** A “packed” size (*maxcol, maxrow*) for this widget

Calculate and return a minimum size where all content could still be displayed. Fixed widgets must implement this method and return their size when `()` is passed as the *size* parameter.

This default implementation returns the *size* passed, or the *maxcol* passed and the value of `rows()` as the *maxrow* when (*maxcol,*) is passed as the *size* parameter.

---

**Note:** This is a new method that hasn't been fully implemented across the standard widget types. In particular it has not yet been implemented for container widgets.

---

Text widgets have implemented this method. You can use `Text.pack()` to calculate the minimum columns and rows required to display a text widget without wrapping, or call it iteratively to calculate the minimum number of columns required to display the text wrapped into a target number of rows.

**render** (*size*, *focus=False*)

**rows** (*size*, *focus=False*)

### **lookatme.widgets.scroll\_monitor module**

```
class lookatme.widgets.scroll_monitor.ScrollMonitor (main_widget:          ur-  
                                                    wid.widget.Widget,    scrollbar:  
                                                    lookatme.widgets.scrollbar.Scrollbar)
```

Bases: `urwid.container.Frame`

**render** (*size*, *focus=False*)

### **lookatme.widgets.scrollbar module**

```
class lookatme.widgets.scrollbar.Scrollbar (listbox: urwid.listbox.ListBox)
```

Bases: `urwid.widget.Widget`

A scrollbar that reflects the current scroll state of a list box. Non-selectable, non-interactive, informative only. Especially useful as an overlay or a box column so that you can have additional padding around your `ListBox`.

**render** (*size*, *focus=False*)

Please use `needs_scrollbar()` if manually deciding to render the `Scrollbar` (e.g. if you're overlaying the rendered results onto a canvas)

**should\_display** (*size*, *focus: bool = False*)

### **lookatme.widgets.smart\_attr\_spec module**

```
class lookatme.widgets.smart_attr_spec.SmartAttrSpec (fg, bg)
```

Bases: `urwid.display_common.AttrSpec`

### **lookatme.widgets.table module**

Defines a basic Table widget for `urwid`

```
class lookatme.widgets.table.SortSetting
```

Bases: `object`

**ASCENDING** = 1

**DEFAULT** = 0

**DESCENDING** = 2

**get\_casting** (*data: List[bytes]*) → Callable

**get\_header\_idx** () → int

**set\_header\_idx** (*idx*)

**set\_indicator** (*idx*, *text: str*)

```

sort_data (contents: List[Tuple[int, bytes]]) → List[int]
    Sort the provided data based on the bytes values in the tuples

class lookatme.widgets.table.Table (ctx: lookatme.render.context.Context, header: Dict[KT, VT], body: Optional[Dict[KT, VT]])
    Bases: urwid.container.Pile

    calc_column_maxes ()

    create_cells (rows, base_spec=None, header=False) → List[List[urwid.container.Pile]]
        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.

    gen_cells (row: List[urwid.container.Pile], row_specs: Tuple[urwid.display_common.AttrSpec, urwid.display_common.AttrSpec]) → List[Tuple[int, urwid.container.Pile]]

    gen_contents ()
        Calculate and set the column maxes for this table

    get_table_rows ()
        Get the current rows in this table. This is where sorting should happen!

    normalize_body ()
        Normalize the cells in the body - truncate all cells that go beyond the number of headers, and add blank cells if we're missing any

    on_header_click (w: urwid.widget.Widget)

    row_specs (row_idx: int, is_header: bool) → Tuple[urwid.display_common.AttrSpec, urwid.display_common.AttrSpec]
        Return the row-specific specs (text and general) for the given row

    signals = ['change']

    validate_row (row: Dict[KT, VT])
        Validate that the provided row is a tr_open, with th_open or td_open children.

    validate_row_container (container: Dict[KT, VT])
        Validate that the list of rows is valid. See validate_row for more details.

    watch_header (idx: int, w: urwid.widget.Widget)
        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]*) → *Dict[str, Any]*

Return the resulting style dict from the provided override values.

`lookatme.config.set_global_style_with_precedence` (*theme\_mod*, *direct\_overrides*) → *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*) → *Tuple[Dict[KT, VT], List[lookatme.slide.Slide], str]*

Parse the provided input data into a Presentation object

**Parameters** *input\_data* (*str*) – The input markdown presentation to parse

**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*) → *List[lookatme.slide.Slide]*

Parse the Slide out of the input data

**Parameters**

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

**Returns** *List[Slide]*

```

class lookatme.parser.SlideIsolator
    Bases: object

    create_slides (tokens, number) → List[lookatme.slide.Slide]

lookatme.parser.is_heading (token)

lookatme.parser.is_hrulerule (token)

lookatme.parser.md_to_tokens (md_text)

```

## lookatme.pres module

Defines Presentation specific objects

```

class lookatme.pres.Presentation (input_stream, theme: str = 'dark', live_reload=False, single_slide=False, preload_extensions=None, safe=False, no_ext_warn=False, ignore_ext_failure=False, no_threads=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!

    to_html (html_output_dir: str, options: Optional[Dict[KT, VT]] = None)

    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.BorderBoxSchema(*, 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.Nested(dump_default={'fg': 'bold', 'bg': '', 'text': ''})>`

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

```
class lookatme.schemas.CodeSchema(*, 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.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: lookatme.schemas.TextStyleFieldSchema

  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.Nested(dump_default={'fg': 'bold', 'bg': '', 'text':

  opts = <marshmallow.schema.SchemaOpts object>

class lookatme.schemas.ScrollbarGutterSchema(*, 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

  Schema for the slider on the scrollbar

  opts = <marshmallow.schema.SchemaOpts object>

class lookatme.schemas.ScrollbarSchema(*, 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

  Schema for the scroll bar

  opts = <marshmallow.schema.SchemaOpts object>

class lookatme.schemas.ScrollbarSliderSchema(*, 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

  Schema for the slider on the scrollbar

  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.TextStyleFieldSchema(*, 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

    get_title (ctx: lookatme.render.context.Context) → Tuple[str, Optional[urwid.canvas.Canvas]]
```

## lookatme.tui module

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

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

    init_ctx ()

    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

    render_without_scrollbar (width: int) → Tuple
        Return a tuple of three canvases: (header, body, footer)

    run ()

    set_slide_idx (slide_idx: int) → lookatme.slide.Slide

    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 (ctx: lookatme.render.context.Context)
    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()**

Clear everything out of the queue and the cache.

**queue\_render(slide)**

Queue up a slide to be rendered.

**render\_slide(slide)**

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, no_threads=False)`

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: float, 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.

```
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: float = 99999.0, lazy_formatting: Optional[Callable] = None)
```

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

## Module contents



## CHAPTER 3

---

### Indices and tables

---

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





### I

- `lookatme`, 49
- `lookatme.ascii_art`, 39
- `lookatme.config`, 39
- `lookatme.contrib`, 24
- `lookatme.contrib.file_loader`, 21
- `lookatme.contrib.terminal`, 22
- `lookatme.exceptions`, 40
- `lookatme.log`, 40
- `lookatme.parser`, 40
- `lookatme.pres`, 41
- `lookatme.prompt`, 41
- `lookatme.render`, 32
- `lookatme.render.asciinema`, 25
- `lookatme.render.context`, 25
- `lookatme.render.html`, 25
- `lookatme.render.html.templating`, 24
- `lookatme.render.markdown_block`, 28
- `lookatme.render.markdown_html`, 29
- `lookatme.render.markdown_inline`, 29
- `lookatme.render.token_iterator`, 32
- `lookatme.schemas`, 41
- `lookatme.slide`, 47
- `lookatme.themes`, 33
- `lookatme.themes.dark`, 33
- `lookatme.themes.light`, 33
- `lookatme.tui`, 47
- `lookatme.tutorial`, 48
- `lookatme.utils`, 33
- `lookatme.utils.colors`, 33
- `lookatme.widgets`, 39
- `lookatme.widgets.clickable_text`, 34
- `lookatme.widgets.codeblock`, 35
- `lookatme.widgets.fancy_box`, 36
- `lookatme.widgets.line_fill`, 37
- `lookatme.widgets.scroll_monitor`, 38
- `lookatme.widgets.scrollbar`, 38
- `lookatme.widgets.smart_attr_spec`, 38
- `lookatme.widgets.table`, 38



## A

ASCENDING (*lookatme.widgets.table.SortSetting* attribute), 38  
*at\_offset()* (*lookatme.render.token\_iterator.TokenIterator* method), 32  
*at\_offset\_v()* (*lookatme.render.token\_iterator.TokenIterator* method), 32

## B

*b\_fill* (*lookatme.widgets.fancy\_box.FancyBox* attribute), 36  
*bl\_corner* (*lookatme.widgets.fancy\_box.FancyBox* attribute), 36  
*BlockQuoteSchema* (class in *lookatme.schemas*), 41  
*BorderBoxSchema* (class in *lookatme.schemas*), 42  
*br\_corner* (*lookatme.widgets.fancy\_box.FancyBox* attribute), 36  
*BulletsSchema* (class in *lookatme.schemas*), 42  
*BulletsSchema.Meta* (class in *lookatme.schemas*), 42

## C

*calc\_column\_maxes()* (*lookatme.widgets.table.Table* method), 39  
*can\_style\_item()* (in module *lookatme.utils*), 33  
*canvas\_to\_html()* (in module *lookatme.render.html*), 25  
*check\_token\_type()* (in module *lookatme.utils*), 33  
*clean\_state\_snapshot()* (*lookatme.render.context.Context* method), 25  
*clean\_state\_validate()* (*lookatme.render.context.Context* method), 26  
*clear\_cache()* (in module *lookatme.widgets.codeblock*), 35  
*ClickableText* (class in *lookatme.widgets.clickable\_text*), 34  
*clone()* (*lookatme.render.context.Context* method), 26

*CodeBlock* (class in *lookatme.widgets.codeblock*), 35  
*CodeSchema* (class in *lookatme.schemas*), 42  
*comma\_append()* (in module *lookatme.render.html.templating*), 24  
*container* (*lookatme.render.context.Context* attribute), 26  
*container\_pop()* (*lookatme.render.context.Context* method), 26  
*container\_push()* (*lookatme.render.context.Context* method), 26  
*ContainerInfo* (class in *lookatme.render.context*), 25  
*Context* (class in *lookatme.render.context*), 25  
*contrib\_first()* (in module *lookatme.contrib*), 24  
*copy\_tree\_update()* (in module *lookatme.render.html*), 25  
*core\_widget()* (in module *lookatme.utils*), 33  
*create\_cells()* (*lookatme.widgets.table.Table* method), 39  
*create\_html\_output()* (in module *lookatme.render.html*), 25  
*create\_log()* (in module *lookatme.log*), 40  
*create\_null\_log()* (in module *lookatme.log*), 40  
*create\_slides()* (*lookatme.parser.SlideIsolator* method), 41  
*create\_tui()* (in module *lookatme.tui*), 48  
*css\_filter()* (in module *lookatme.render.html.templating*), 25  
*curr* (*lookatme.render.token\_iterator.TokenIterator* attribute), 32  
*curr\_parent* (*lookatme.render.markdown\_block.TableTokenExtractor* attribute), 28  
*curr\_siblings* (*lookatme.render.markdown\_block.TableTokenExtractor* attribute), 28

## D

*daemon* (*lookatme.tui.SlideRenderer* attribute), 47  
*debug\_print\_tokens()* (in module *lookatme.utils*), 33  
*DEFAULT* (*lookatme.widgets.table.SortSetting* attribute), 38

DESCENDING (*lookatme.widgets.table.SortSetting attribute*), 38  
dict\_deep\_update() (*in module lookatme.utils*), 33  
do\_render() (*lookatme.tui.SlideRenderer method*), 47  
dump() (*lookatme.schemas.MetaSchema method*), 43  
dumps() (*lookatme.contrib.file\_loader.YamlRender static method*), 22  
dumps() (*lookatme.contrib.terminal.YamlRender static method*), 23  
dumps() (*lookatme.schemas.YamlRender static method*), 46

## E

ensure\_contrast() (*in module lookatme.utils.colors*), 33  
ensure\_defaults() (*in module lookatme.themes*), 33  
ensure\_new\_block() (*lookatme.render.context.Context method*), 26  
extract\_hexcolor() (*in module lookatme.utils*), 33

## F

fake\_token() (*lookatme.render.context.Context method*), 26  
FancyBox (*class in lookatme.widgets.fancy\_box*), 36  
FileSchema (*class in lookatme.contrib.file\_loader*), 21  
FileSchema.Meta (*class in lookatme.contrib.file\_loader*), 21  
flatten\_text() (*in module lookatme.utils*), 33  
flush\_cache() (*lookatme.tui.SlideRenderer method*), 48

## G

gen\_cells() (*lookatme.widgets.table.Table method*), 39  
gen\_contents() (*lookatme.widgets.table.Table method*), 39  
get\_casting() (*lookatme.widgets.table.SortSetting method*), 38  
get\_fg\_bg\_styles() (*in module lookatme.utils*), 34  
get\_filters() (*in module lookatme.render.html.templating*), 25  
get\_header\_idx() (*lookatme.widgets.table.SortSetting method*), 38  
get\_highlight\_color() (*in module lookatme.utils.colors*), 33  
get\_html() (*lookatme.render.html.HtmlContext method*), 25  
get\_html\_consumed() (*lookatme.render.html.HtmlContext method*), 25

get\_inline\_markup() (*lookatme.render.context.Context method*), 26  
get\_inline\_widgets() (*lookatme.render.context.Context method*), 26  
get\_lexer() (*in module lookatme.widgets.codeblock*), 35  
get\_line\_number\_spec() (*lookatme.widgets.codeblock.SyntaxHlStyle method*), 35  
get\_log() (*in module lookatme.config*), 39  
get\_md() (*lookatme.tutorial.Tutor method*), 49  
get\_meta() (*in module lookatme.utils*), 34  
get\_style() (*in module lookatme.config*), 39  
get\_style() (*lookatme.widgets.codeblock.StyleCache method*), 35  
get\_style\_cache() (*in module lookatme.widgets.codeblock*), 35  
get\_style\_spec() (*lookatme.widgets.codeblock.SyntaxHlStyle method*), 35  
get\_style\_with\_precedence() (*in module lookatme.config*), 39  
get\_table\_rows() (*lookatme.widgets.table.Table method*), 39  
get\_template\_data() (*in module lookatme.render.html.templating*), 25  
get\_title() (*lookatme.slide.Slide method*), 47  
get\_tui() (*lookatme.pres.Presentation method*), 41  
get\_tutorial\_help() (*in module lookatme.tutorial*), 49  
get\_tutorial\_md() (*in module lookatme.tutorial*), 49  
get\_tutors() (*in module lookatme.tutorial*), 49

## H

HeadingsSchema (*class in lookatme.schemas*), 42  
HeadingsSchema.Meta (*class in lookatme.schemas*), 42  
HeadingStyleSchema (*class in lookatme.schemas*), 42  
highlight\_filter() (*in module lookatme.render.html.templating*), 25  
HORIZONTAL (*lookatme.widgets.line\_fill.LineFill attribute*), 37  
HrRuleSchema (*class in lookatme.schemas*), 43  
HtmlContext (*class in lookatme.render.html*), 25

## I

IgnoredByContrib, 40  
include (*lookatme.schemas.BulletsSchema.Meta attribute*), 42  
include (*lookatme.schemas.HeadingsSchema.Meta attribute*), 43

include (*lookatme.schemas.NumberingSchema.Meta attribute*), 45  
 increase\_brightness() (in module *lookatme.utils.colors*), 33  
 init\_ctx() (*lookatme.tui.MarkdownTui method*), 47  
 inline\_clear() (*lookatme.render.context.Context method*), 26  
 inline\_convert\_all\_to\_widgets() (*lookatme.render.context.Context method*), 26  
 inline\_flush() (*lookatme.render.context.Context method*), 26  
 inline\_markup\_consumed (*lookatme.render.context.Context attribute*), 26  
 inline\_push() (*lookatme.render.context.Context method*), 26  
 inline\_widgets\_consumed (*lookatme.render.context.Context attribute*), 26  
 int\_to\_roman() (in module *lookatme.utils*), 34  
 is\_heading() (in module *lookatme.parser*), 41  
 is\_hrulerule() (in module *lookatme.parser*), 41  
 is\_literal (*lookatme.render.context.Context attribute*), 26  
 is\_table\_element\_close() (*lookatme.render.markdown\_block.TableTokenExtractor method*), 28  
 is\_table\_element\_open() (*lookatme.render.markdown\_block.TableTokenExtractor method*), 28  
 is\_valid\_style() (*lookatme.widgets.codeblock.StyleCache method*), 35  
**J**  
 json\_filter() (in module *lookatme.render.html.templating*), 25  
**K**  
 keypress() (*lookatme.tui.MarkdownTui method*), 47  
**L**  
 l\_fill (*lookatme.widgets.fancy\_box.FancyBox attribute*), 36  
 level\_inc() (*lookatme.render.context.Context method*), 26  
 LineFill (class in *lookatme.widgets.line\_fill*), 37  
 LineRange (class in *lookatme.contrib.file\_loader*), 22  
 LinkIndicatorSpec (class in *lookatme.widgets.clickable\_text*), 35  
 listbox\_add() (in module *lookatme.utils*), 34  
 load() (*lookatme.contrib.file\_loader.FileSchema method*), 21  
 load() (*lookatme.contrib.terminal.TerminalExSchema method*), 23  
 load() (*lookatme.schemas.MetaSchema method*), 43  
 load\_contribs() (in module *lookatme.contrib*), 24  
 load\_partial\_styles() (*lookatme.schemas.MetaSchema method*), 44  
 load\_style() (*lookatme.widgets.codeblock.StyleCache method*), 35  
 loads() (*lookatme.contrib.file\_loader.FileSchema method*), 21  
 loads() (*lookatme.contrib.file\_loader.YamlRender static method*), 22  
 loads() (*lookatme.contrib.terminal.TerminalExSchema method*), 23  
 loads() (*lookatme.contrib.terminal.YamlRender static method*), 23  
 loads() (*lookatme.schemas.MetaSchema method*), 44  
 loads() (*lookatme.schemas.YamlRender static method*), 46  
 loads\_partial\_styles() (*lookatme.schemas.MetaSchema method*), 44  
 log\_debug() (*lookatme.render.context.Context method*), 26  
 lookatme (module), 49  
 lookatme.ascii\_art (module), 39  
 lookatme.config (module), 39  
 lookatme.contrib (module), 24  
 lookatme.contrib.file\_loader (module), 21  
 lookatme.contrib.terminal (module), 22  
 lookatme.exceptions (module), 40  
 lookatme.log (module), 40  
 lookatme.parser (module), 40  
 lookatme.pres (module), 41  
 lookatme.prompt (module), 41  
 lookatme.render (module), 32  
 lookatme.render.asciinema (module), 25  
 lookatme.render.context (module), 25  
 lookatme.render.html (module), 25  
 lookatme.render.html.templating (module), 24  
 lookatme.render.markdown\_block (module), 28  
 lookatme.render.markdown\_html (module), 29  
 lookatme.render.markdown\_inline (module), 29  
 lookatme.render.token\_iterator (module), 32  
 lookatme.schemas (module), 41  
 lookatme.slide (module), 47  
 lookatme.themes (module), 33  
 lookatme.themes.dark (module), 33  
 lookatme.themes.light (module), 33  
 lookatme.tui (module), 47  
 lookatme.tutorial (module), 48  
 lookatme.utils (module), 33

lookatme.utils.colors (module), 33  
 lookatme.widgets (module), 39  
 lookatme.widgets.clickable\_text (module), 34  
 lookatme.widgets.codeblock (module), 35  
 lookatme.widgets.fancy\_box (module), 36  
 lookatme.widgets.line\_fill (module), 37  
 lookatme.widgets.scroll\_monitor (module), 38  
 lookatme.widgets.scrollbar (module), 38  
 lookatme.widgets.smart\_attr\_spec (module), 38  
 lookatme.widgets.table (module), 38  
 luminance () (in module lookatme.utils.colors), 33

## M

markdown\_block () (in module lookatme.render.markdown\_inline), 29  
 MarkdownTui (class in lookatme.tui), 47  
 md\_to\_tokens () (in module lookatme.parser), 41  
 meta (lookatme.render.context.Context attribute), 26  
 MetaSchema (class in lookatme.schemas), 43  
 MetaSchema.Meta (class in lookatme.schemas), 43  
 mouse\_event () (lookatme.widgets.clickable\_text.ClickableText method), 34

## N

new\_for\_spec () (lookatme.widgets.clickable\_text.LinkIndicatorSpec method), 35  
 next () (lookatme.render.token\_iterator.TokenIterator method), 32  
 NoDatesSafeLoader (class in lookatme.schemas), 44  
 non\_empty\_split () (in module lookatme.utils), 34  
 normalize\_body () (lookatme.widgets.table.Table method), 39  
 NumberingSchema (class in lookatme.schemas), 44  
 NumberingSchema.Meta (class in lookatme.schemas), 44

## O

on\_header\_click () (lookatme.widgets.table.Table method), 39  
 opts (lookatme.contrib.file\_loader.FileSchema attribute), 22  
 opts (lookatme.contrib.file\_loader.LineRange attribute), 22  
 opts (lookatme.contrib.terminal.TerminalExSchema attribute), 23  
 opts (lookatme.schemas.BlockQuoteSchema attribute), 42  
 opts (lookatme.schemas.BorderBoxSchema attribute), 42  
 opts (lookatme.schemas.BulletsSchema attribute), 42  
 opts (lookatme.schemas.CodeSchema attribute), 42

opts (lookatme.schemas.HeadingsSchema attribute), 43  
 opts (lookatme.schemas.HeadingStyleSchema attribute), 42  
 opts (lookatme.schemas.HruleSchema attribute), 43  
 opts (lookatme.schemas.MetaSchema attribute), 44  
 opts (lookatme.schemas.NumberingSchema attribute), 45  
 opts (lookatme.schemas.ScrollbarGutterSchema attribute), 45  
 opts (lookatme.schemas.ScrollbarSchema attribute), 45  
 opts (lookatme.schemas.ScrollbarSliderSchema attribute), 45  
 opts (lookatme.schemas.SpacingSchema attribute), 45  
 opts (lookatme.schemas.StyleFieldSchema attribute), 46  
 opts (lookatme.schemas.StyleSchema attribute), 46  
 opts (lookatme.schemas.TableSchema attribute), 46  
 opts (lookatme.schemas.TextStyleFieldSchema attribute), 46  
 overwrite\_spec () (in module lookatme.utils), 34  
 overwrite\_style () (in module lookatme.utils), 34

## P

pack () (lookatme.widgets.fancy\_box.FancyBox method), 36  
 pack () (lookatme.widgets.line\_fill.LineFill method), 37  
 packed\_widget\_width () (in module lookatme.utils), 34  
 parse () (lookatme.parser.Parser method), 40  
 parse () (lookatme.render.markdown\_html.Tag class method), 29  
 parse\_meta () (lookatme.parser.Parser method), 40  
 parse\_slides () (lookatme.parser.Parser method), 40  
 parse\_style () (lookatme.render.markdown\_html.Tag class method), 29  
 Parser (class in lookatme.parser), 40  
 peek () (lookatme.render.token\_iterator.TokenIterator method), 32  
 pile\_add () (in module lookatme.utils), 34  
 pile\_or\_listbox\_add () (in module lookatme.utils), 34  
 prefix\_text () (in module lookatme.utils), 34  
 prep\_pres () (lookatme.tui.MarkdownTui method), 47  
 Presentation (class in lookatme.pres), 41  
 pretty\_close\_match () (in module lookatme.tutorial), 49  
 print\_tutorial\_help () (in module lookatme.tutorial), 49  
 process\_token () (lookatme.render.markdown\_block.TableTokenExtra method), 28  
 process\_tokens () (lookatme.render.markdown\_block.TableTokenExtra method), 28



## Q

`queue_render()` (*lookatme.tui.SlideRenderer* method), 48

## R

`r_fill` (*lookatme.widgets.fancy\_box.FancyBox* attribute), 37

`reload()` (*lookatme.pres.Presentation* method), 41

`reload()` (*lookatme.tui.MarkdownTui* method), 47

`reload_watcher()` (*lookatme.pres.Presentation* method), 41

`remove_implicit_resolver()` (*lookatme.schemas.NoDatesSafeLoader* class method), 44

`render()` (in module *lookatme.render.html.templating*), 25

`render()` (in module *lookatme.render.markdown\_block*), 28

`render()` (in module *lookatme.render.markdown\_inline*), 29

`render()` (*lookatme.widgets.line\_fill.LineFill* method), 38

`render()` (*lookatme.widgets.scroll\_monitor.ScrollMonitor* method), 38

`render()` (*lookatme.widgets.scrollbar.Scrollbar* method), 38

`render_all()` (in module *lookatme.render.markdown\_block*), 28

`render_all()` (in module *lookatme.render.markdown\_inline*), 29

`render_blockquote_close()` (in module *lookatme.render.markdown\_block*), 28

`render_blockquote_open()` (in module *lookatme.render.markdown\_block*), 28

`render_bullet_list_close()` (in module *lookatme.render.markdown\_block*), 28

`render_bullet_list_open()` (in module *lookatme.render.markdown\_block*), 28

`render_code_inline()` (in module *lookatme.render.markdown\_inline*), 29

`render_em_close()` (in module *lookatme.render.markdown\_inline*), 29

`render_em_open()` (in module *lookatme.render.markdown\_inline*), 29

`render_fence()` (in module *lookatme.contrib.file\_loader*), 22

`render_fence()` (in module *lookatme.contrib.terminal*), 23

`render_fence()` (in module *lookatme.render.markdown\_block*), 28

`render_hardbreak()` (in module *lookatme.render.markdown\_inline*), 30

`render_heading_close()` (in module *lookatme.render.markdown\_block*), 28

`render_heading_open()` (in module *lookatme.render.markdown\_block*), 28

`render_hr()` (in module *lookatme.render.markdown\_block*), 28

`render_html_inline()` (in module *lookatme.render.markdown\_inline*), 30

`render_html_tag_b_open()` (in module *lookatme.render.markdown\_inline*), 30

`render_html_tag_blink_open()` (in module *lookatme.render.markdown\_inline*), 30

`render_html_tag_br_open()` (in module *lookatme.render.markdown\_inline*), 30

`render_html_tag_default_close()` (in module *lookatme.render.markdown\_inline*), 30

`render_html_tag_default_open()` (in module *lookatme.render.markdown\_inline*), 30

`render_html_tag_div_close()` (in module *lookatme.render.markdown\_inline*), 30

`render_html_tag_div_open()` (in module *lookatme.render.markdown\_inline*), 30

`render_html_tag_em_open()` (in module *lookatme.render.markdown\_inline*), 30

`render_html_tag_i_open()` (in module *lookatme.render.markdown\_inline*), 31

`render_html_tag_li_close()` (in module *lookatme.render.markdown\_inline*), 31

`render_html_tag_li_open()` (in module *lookatme.render.markdown\_inline*), 31

`render_html_tag_ol_close()` (in module *lookatme.render.markdown\_inline*), 31

`render_html_tag_ol_open()` (in module *lookatme.render.markdown\_inline*), 31

`render_html_tag_u_open()` (in module *lookatme.render.markdown\_inline*), 31

`render_html_tag_ul_close()` (in module *lookatme.render.markdown\_inline*), 31

`render_html_tag_ul_open()` (in module *lookatme.render.markdown\_inline*), 31

`render_image()` (in module *lookatme.render.markdown\_inline*), 32

`render_image_close()` (in module *lookatme.render.markdown\_inline*), 32

`render_inline()` (in module *lookatme.render.markdown\_block*), 28

`render_link_close()` (in module *lookatme.render.markdown\_inline*), 32

`render_link_open()` (in module *lookatme.render.markdown\_inline*), 32

`render_list_close()` (in module *lookatme.render.markdown\_block*), 29

`render_list_item_close()` (in module *lookatme.render.markdown\_block*), 29

`render_list_item_open()` (in module *lookatme.render.markdown\_block*), 29

[render\\_list\\_open\(\)](#) (in module [ScrollMonitor](#) (class in [lookatme.render.markdown\\_block](#)), 29 [lookatme.widgets.scroll\\_monitor](#)), 38  
[render\\_module\(\)](#) ([lookatme.contrib.file\\_loader.FileSchema.Meta](#) (lookatme.widgets.fancy\_box.FancyBox method), attribute), 21 37  
[render\\_module\(\)](#) ([lookatme.contrib.terminal.TerminalException](#) (class in module [lookatme.config](#)), 40 attribute), 23  
[render\\_module\(\)](#) ([lookatme.schemas.MetaSchema.Meta](#) (lookatme.widgets.table.SortSetting method), 38 attribute), 43  
[render\\_module\(\)](#) ([lookatme.schemas.StyleSchema.Meta](#) (lookatme.widgets.table.SortSetting method), 38 attribute), 46  
[render\\_ordered\\_list\\_close\(\)](#) (in module [lookatme.render.markdown\\_block](#)), 29  
[render\\_ordered\\_list\\_open\(\)](#) (in module [lookatme.render.markdown\\_block](#)), 29  
[render\\_paragraph\\_close\(\)](#) (in module [lookatme.render.markdown\\_block](#)), 29  
[render\\_paragraph\\_open\(\)](#) (in module [lookatme.render.markdown\\_block](#)), 29  
[render\\_s\\_close\(\)](#) (in module [lookatme.render.markdown\\_inline](#)), 32  
[render\\_s\\_open\(\)](#) (in module [lookatme.render.markdown\\_inline](#)), 32  
[render\\_slide\(\)](#) ([lookatme.tui.SlideRenderer](#) method), 48  
[render\\_softbreak\(\)](#) (in module [lookatme.render.markdown\\_inline](#)), 32  
[render\\_strong\\_close\(\)](#) (in module [lookatme.render.markdown\\_inline](#)), 32  
[render\\_strong\\_open\(\)](#) (in module [lookatme.render.markdown\\_inline](#)), 32  
[render\\_table\\_open\(\)](#) (in module [lookatme.render.markdown\\_block](#)), 29  
[render\\_text\(\)](#) (in module [lookatme.render.markdown\\_inline](#)), 32  
[render\\_without\\_scrollbar\(\)](#) ([lookatme.tui.MarkdownTui](#) method), 47  
[resolve\\_bag\\_of\\_text\\_markup\\_or\\_widgets\(\)](#) (in module [lookatme.utils](#)), 34  
[root\\_urwid\\_widget\(\)](#) (in module [lookatme.tui](#)), 48  
[row\\_specs\(\)](#) ([lookatme.widgets.table.Table](#) method), 39  
[row\\_text\(\)](#) (in module [lookatme.utils](#)), 34  
[rows\(\)](#) ([lookatme.widgets.line\\_fill.LineFill](#) method), 38  
[run\(\)](#) ([lookatme.pres.Presentation](#) method), 41  
[run\(\)](#) ([lookatme.tui.MarkdownTui](#) method), 47  
[run\(\)](#) ([lookatme.tui.SlideRenderer](#) method), 48

## S

[Scrollbar](#) (class in [lookatme.widgets.scrollbar](#)), 38  
[ScrollbarGutterSchema](#) (class in [lookatme.schemas](#)), 45  
[ScrollbarSchema](#) (class in [lookatme.schemas](#)), 45  
[ScrollbarSliderSchema](#) (class in [lookatme.schemas](#)), 45  
[ScrollMonitor](#) (class in [lookatme.widgets.scroll\\_monitor](#)), 38  
[set\\_header\\_idx\(\)](#) ([lookatme.widgets.table.SortSetting](#) method), 38  
[set\\_indicator\(\)](#) ([lookatme.widgets.table.SortSetting](#) method), 38  
[set\\_slide\\_idx\(\)](#) ([lookatme.tui.MarkdownTui](#) method), 47  
[set\\_token\\_alignment\(\)](#) ([lookatme.render.markdown\\_block.TableTokenExtractor](#) method), 28  
[should\\_display\(\)](#) ([lookatme.widgets.scrollbar.Scrollbar](#) method), 38  
[shutdown\(\)](#) (in module [lookatme.contrib.terminal](#)), 23  
[shutdown\\_contribs\(\)](#) (in module [lookatme.contrib](#)), 24  
[signals](#) ([lookatme.widgets.clickable\\_text.ClickableText](#) attribute), 35  
[signals](#) ([lookatme.widgets.table.Table](#) attribute), 39  
[Slide](#) (class in [lookatme.slide](#)), 47  
[SlideIsolator](#) (class in [lookatme.parser](#)), 40  
[SlideRenderer](#) (class in [lookatme.tui](#)), 47  
[SmartAttrSpec](#) (class in [lookatme.widgets.smart\\_attr\\_spec](#)), 38  
[sort\\_data\(\)](#) ([lookatme.widgets.table.SortSetting](#) method), 38  
[SortSetting](#) (class in [lookatme.widgets.table](#)), 38  
[source](#) ([lookatme.render.context.Context](#) attribute), 26  
[source\\_get\\_token\\_lines\(\)](#) ([lookatme.render.context.Context](#) method), 26  
[source\\_pop\(\)](#) ([lookatme.render.context.Context](#) method), 26  
[source\\_push\(\)](#) ([lookatme.render.context.Context](#) method), 27  
[SpacingSchema](#) (class in [lookatme.schemas](#)), 45  
[spec\\_from\\_stack\(\)](#) (in module [lookatme.utils](#)), 34  
[spec\\_from\\_style\(\)](#) (in module [lookatme.utils](#)), 34  
[spec\\_general](#) ([lookatme.render.context.Context](#) attribute), 27  
[spec\\_peek\(\)](#) ([lookatme.render.context.Context](#) method), 27  
[spec\\_pop\(\)](#) ([lookatme.render.context.Context](#) method), 27  
[spec\\_push\(\)](#) ([lookatme.render.context.Context](#) method), 27  
[spec\\_text](#) ([lookatme.render.context.Context](#) attribute), 27  
[spec\\_text\\_with\(\)](#) ([lookatme.render.context.Context](#) method), 27



- stop() (*lookatme.tui.SlideRenderer method*), 48  
 StyleCache (*class in lookatme.widgets.codeblock*), 35  
 styled\_text() (*in module lookatme.utils*), 34  
 StyleFieldSchema (*class in lookatme.schemas*), 45  
 StyleSchema (*class in lookatme.schemas*), 46  
 StyleSchema.Meta (*class in lookatme.schemas*), 46  
 supported\_langs() (*in module lookatme.widgets.codeblock*), 35  
 supported\_styles() (*in module lookatme.widgets.codeblock*), 36  
 SyntaxHlStyle (*class in lookatme.widgets.codeblock*), 35
- ## T
- t\_fill (*lookatme.widgets.fancy\_box.FancyBox attribute*), 37  
 Table (*class in lookatme.widgets.table*), 39  
 TableSchema (*class in lookatme.schemas*), 46  
 TableTokenExtractor (*class in lookatme.render.markdown\_block*), 28  
 Tag (*class in lookatme.render.markdown\_html*), 29  
 tag (*lookatme.render.context.Context attribute*), 27  
 tag\_is\_ancestor() (*lookatme.render.context.Context method*), 27  
 tag\_pop() (*lookatme.render.context.Context method*), 27  
 tag\_push() (*lookatme.render.context.Context method*), 27  
 tag\_token (*lookatme.render.context.Context attribute*), 27  
 TerminalExSchema (*class in lookatme.contrib.terminal*), 22  
 TerminalExSchema.Meta (*class in lookatme.contrib.terminal*), 22  
 text() (*in module lookatme.tui*), 48  
 TextStyleFieldSchema (*class in lookatme.schemas*), 46  
 tl\_corner (*lookatme.widgets.fancy\_box.FancyBox attribute*), 37  
 to\_html() (*lookatme.pres.Presentation method*), 41  
 TokenIterator (*class in lookatme.render.token\_iterator*), 32  
 tokens (*lookatme.render.context.Context attribute*), 27  
 tokens\_pop() (*lookatme.render.context.Context method*), 27  
 tokens\_push() (*lookatme.render.context.Context method*), 27  
 tokens\_to\_lines() (*in module lookatme.widgets.codeblock*), 36  
 tokens\_to\_markup() (*in module lookatme.widgets.codeblock*), 36  
 tr\_corner (*lookatme.widgets.fancy\_box.FancyBox attribute*), 37  
 transform\_data() (*in module lookatme.contrib.file\_loader*), 22  
 Tutor (*class in lookatme.tutorial*), 48  
 tutor() (*in module lookatme.tutorial*), 49
- ## U
- unknown (*lookatme.schemas.MetaSchema.Meta attribute*), 43  
 unknown (*lookatme.schemas.StyleSchema.Meta attribute*), 46  
 unwind\_bookmark (*lookatme.render.context.Context attribute*), 27  
 unwind\_has\_type() (*lookatme.render.token\_iterator.TokenIterator method*), 32  
 unwind\_tokens (*lookatme.render.context.Context attribute*), 27  
 unwind\_tokens\_consumed (*lookatme.render.context.Context attribute*), 27  
 unwind\_tokens\_from() (*lookatme.render.context.Context method*), 27  
 update() (*lookatme.tui.MarkdownTui method*), 47  
 update\_body() (*lookatme.tui.MarkdownTui method*), 47  
 update\_creation() (*lookatme.tui.MarkdownTui method*), 47  
 update\_slide\_num() (*lookatme.tui.MarkdownTui method*), 47  
 update\_slide\_settings() (*lookatme.tui.MarkdownTui method*), 47  
 update\_title() (*lookatme.tui.MarkdownTui method*), 47  
 use\_container() (*lookatme.render.context.Context method*), 27  
 use\_container\_tmp() (*lookatme.render.context.Context method*), 27  
 use\_spec() (*lookatme.render.context.Context method*), 27  
 use\_spec() (*lookatme.render.html.HtmlContext method*), 25  
 use\_tag() (*lookatme.render.html.HtmlContext method*), 25  
 use\_tokens() (*lookatme.render.context.Context method*), 27  
 user\_warnings() (*in module lookatme.contrib.file\_loader*), 22  
 user\_warnings() (*in module lookatme.contrib.terminal*), 23
- ## V
- validate\_extension\_mod() (*in module lookatme.contrib*), 24

`validate_row()` (*lookatme.widgets.table.Table method*), 39  
`validate_row_container()` (*lookatme.widgets.table.Table method*), 39  
`VERTICAL` (*lookatme.widgets.line\_fill.LineFill attribute*), 37

## W

`warn_exts()` (*lookatme.pres.Presentation method*), 41  
`watch_header()` (*lookatme.widgets.table.Table method*), 39  
`widget_add()` (*lookatme.render.context.Context method*), 27  
`wrap_widget()` (*lookatme.render.context.Context method*), 28  
`write()` (*lookatme.render.html.HtmlContext method*), 25

## Y

`yaml_implicit_resolvers` (*lookatme.schemas.NoDatesSafeLoader attribute*), 44  
`YamlRender` (*class in lookatme.contrib.file\_loader*), 22  
`YamlRender` (*class in lookatme.contrib.terminal*), 23  
`YamlRender` (*class in lookatme.schemas*), 46  
`yes()` (*in module lookatme.prompt*), 41