

---

**term-img**

**AnonymouX47**

**Jul 13, 2022**



## CONTENTS:

<b>1</b>	<b>Installation</b>	<b>1</b>
1.1	Requirements . . . . .	1
1.2	Steps . . . . .	1
1.3	Supported Terminal Emulators . . . . .	1
<b>2</b>	<b>Library Documentation</b>	<b>3</b>
2.1	Tutorial . . . . .	3
2.1.1	Creating an instance . . . . .	4
2.1.2	Rendering an image . . . . .	4
2.1.3	Drawing/Displaying an image to/in the terminal . . . . .	8
2.1.4	Image render size . . . . .	9
2.1.5	Image render scale . . . . .	11
2.2	Reference . . . . .	13
2.2.1	Core Library Definitions . . . . .	13
2.2.2	Context Manager Support . . . . .	19
2.2.3	Custom Exceptions . . . . .	19
2.2.4	Top-Level Functions . . . . .	19
2.2.5	Image Format Specification . . . . .	20
2.3	Known Issues . . . . .	21
2.4	Planned Features . . . . .	21
<b>3</b>	<b>Image viewer</b>	<b>23</b>
3.1	Text-based User Interface . . . . .	23
3.1.1	UI Components . . . . .	23
3.1.2	Contexts . . . . .	24
3.1.3	Actions . . . . .	25
3.2	Configuration . . . . .	25
3.2.1	Config Options . . . . .	25
3.2.2	Key Config . . . . .	26
3.3	Image sources . . . . .	32
3.4	Modes . . . . .	32
3.5	Usage . . . . .	32
3.6	Notifications . . . . .	33
3.7	Logging . . . . .	33
3.8	Known Issues . . . . .	34
3.9	Planned Features . . . . .	34
<b>4</b>	<b>FAQs</b>	<b>35</b>
<b>5</b>	<b>Glossary</b>	<b>37</b>

<b>6 Indices and tables</b>	<b>39</b>
<b>Python Module Index</b>	<b>41</b>
<b>Index</b>	<b>43</b>

## INSTALLATION

### 1.1 Requirements

- Operating System: Unix / Linux / MacOS X / Windows (partial, see the [FAQs](#))
- Python  $\geq 3.6$
- A terminal emulator with full Unicode support and ANSI 24-bit color support
  - Plans are in place to [partially] support terminals not meeting this requirements (see [Planned Features](#)).

### 1.2 Steps

The package can be installed from [PyPI](#) using `pip`:

```
pip install term-image
```

OR

Clone this repository using any method, then navigate into the project directory in a terminal and run:

```
pip install .
```

### 1.3 Supported Terminal Emulators

Some terminals emulators that have been tested to meet all major requirements are:

- **libvte**-based terminals such as:
  - Gnome Terminal
  - Terminator
  - Tilix
- Alacritty
- Kitty
- Windows Terminal
- Termux (on Android)
- xterm, uxterm (*256 colors*)

**Warning:** With some of these terminals, there's an issue where the TUI isn't cleared off the terminal screen after exiting.

Manually running `clear` should clear the terminal screen.

---

**Note:** If you've tested `term-img` on any other terminal emulator that meets all requirements, please mention the name in a new thread under [this discussion](#).

Also, if you're having an issue with terminal support, also report or view information about it in the discussion linked above.

---

See [here](#) for information about terminal colors and a list of potentially supported terminal emulators.

**Warning:** Some terminal emulators support 24-bit colors but have a 256-color palette. This will limit color reproduction

## LIBRARY DOCUMENTATION

### 2.1 Tutorial

This is a basic introduction to using the library. Please refer to the [Reference](#) for detailed description of the features and functionality provided by the library.

For this tutorial we'll be using the image below:



The image has a resolution of **288x288 pixels**.

---

**Note:** All the samples in this tutorial occurred in a terminal of **255 columns by 70 lines**.

---

### 2.1.1 Creating an instance

If the file is stored on your local filesystem:

```
from term_img.image import TermImage

image = TermImage.from_file("python.png")
```

You can also use a URL if you don't have the file stored locally:

```
from term_img.image import TermImage

image = TermImage.from_url("https://raw.githubusercontent.com/AnonymouX47/term-img/docs/
↪source/resources/python.png")
```

The library can also be used with PIL images:

```
from PIL import Image
from term_img.image import TermImage

img = Image.open("python.png")
image = TermImage(img)
```

### 2.1.2 Rendering an image

Rendering an image is simply the process of converting it (per-frame for *animated* images) into text (a string).

---

**Hint:** To display the rendered output in the following steps, just print the output string with `print()`.

---

There are two ways to render an image:

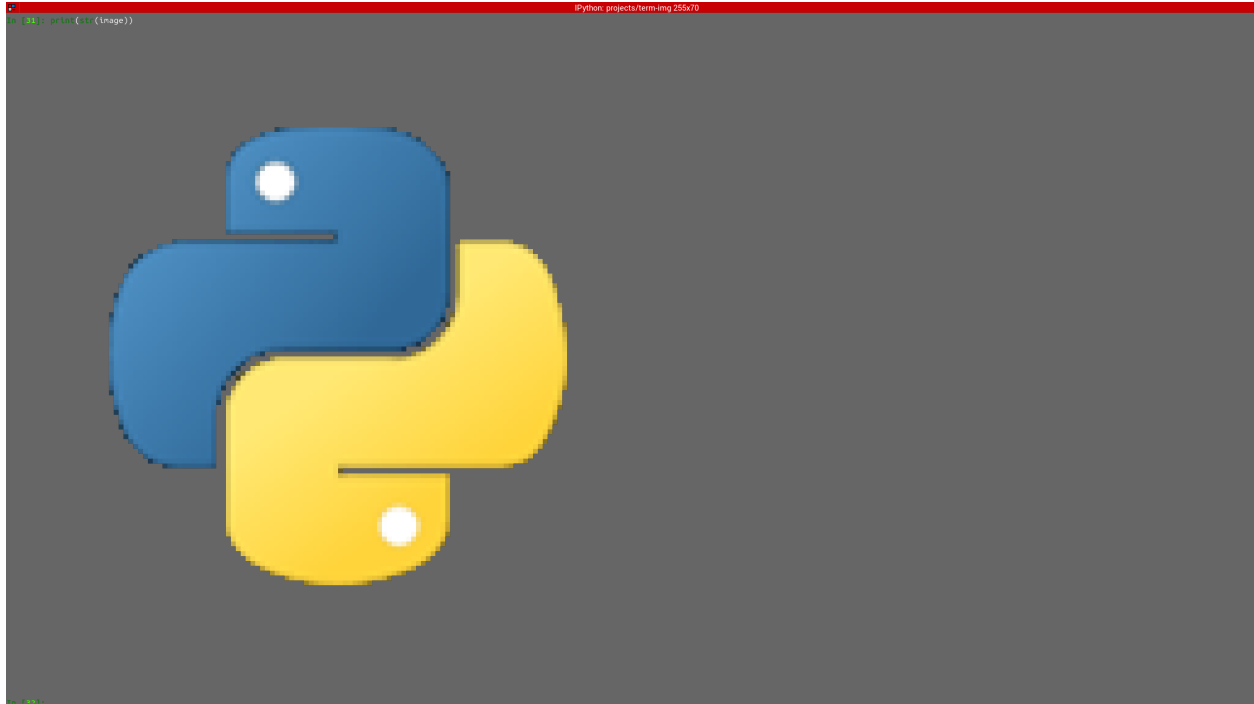
#### 1. Unformatted

```
str(image)
```

Renders the image without *padding/alignment* and with transparency enabled

The result should look like:





## 2. Formatted

**Note:** To see the effect of *alignment* in the steps below, please scale the image down using:

```
image.scale = 0.75
```

This simply sets the x-axis and y-axis *scales* of the image to 0.75. We'll see more about this *later*.

Below are examples of formatted rendering:

```
format(image, "|200.^70#ffffff")
```

Renders the image with:

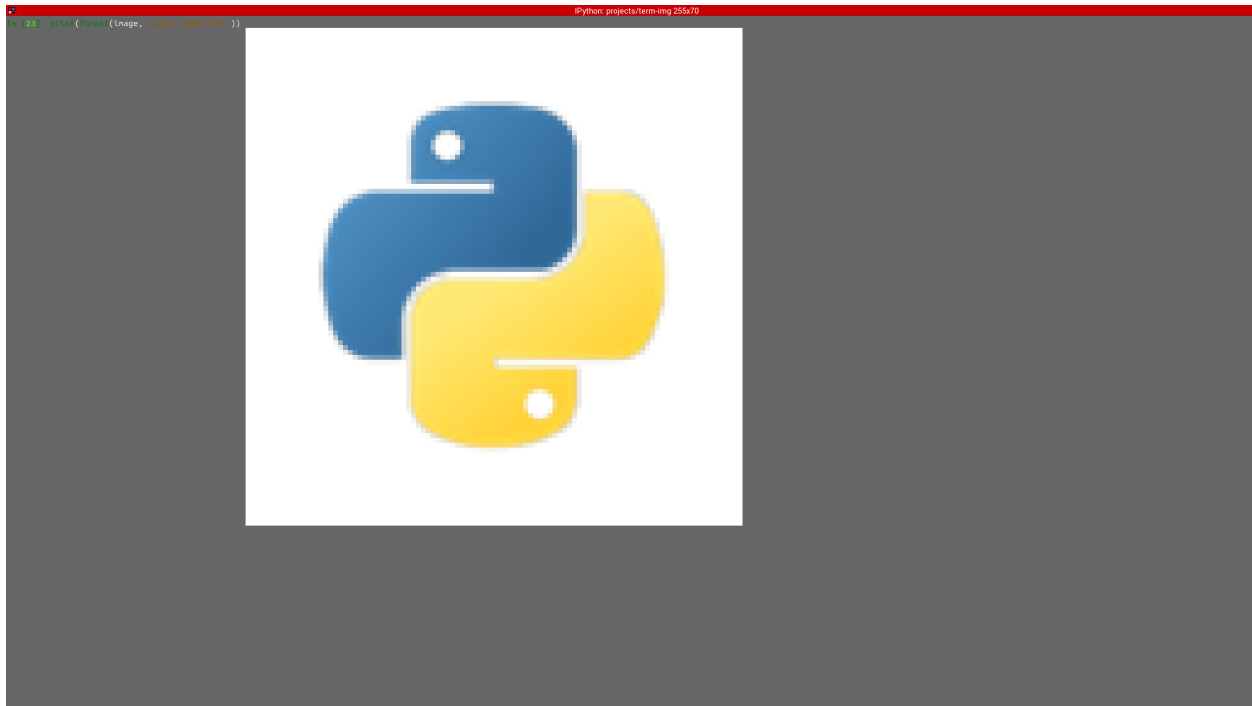
- **center** *horizontal alignment*
- a *padding width* of **200** columns
- **top** *vertical alignment*
- a *padding height* of **70** lines
- transparent background replaced with a **white** (`#ffffff`) background

**Note:** If you get an error while trying the step above, saying something like “padding width larger than...”, either:

- reduce the width (200) to something that'll fit into your terminal window, or
- increase the size of the terminal window

You might use your own *terminal height* instead of **70**.

The result should look like:



```
f"{image:>._#.5}"
```

Renders the image with:

- **right** *horizontal alignment*
- **automatic** *padding width* (the current *terminal width* minus *horizontal allowance*)
- **bottom** *vertical alignment*
- **automatic** *padding height* (the current *terminal height* minus *vertical allowance*)
- transparent background with **0.5** *alpha threshold*

The result should look like:



```
"{:1.1#}".format(image)
```

Renders the image with:

- **center** *horizontal alignment* (default)
- **no** horizontal *padding*, since 1 must be less than or equal to the image *width*
- **middle** *vertical alignment* (default)
- **no** vertical *padding*, since 1 is less than or equal to the image *height*
- transparency **disabled** (black background)

The result should look like:



You should also have a look at the complete *Image Format Specification*.

### 2.1.3 Drawing/Displaying an image to/in the terminal

There are two ways to draw an image to the terminal screen:

1. The `draw()` method

```
image.draw()
```

**NOTE:** `TermImage.draw()` has various parameters for *alignment/padding*, transparency and animation control.

2. Using `print()` with an image render output (i.e printing the rendered string)

```
print(image) # Uses str()
```

OR

```
print(f"{image:>200.^70#ffffff}") # Uses format()
```

---

**Note:** For *animated* images, only the first method animates the output, the second only draws the **current** frame (see `TermImage.seek()` and `TermImage.tell()`).

---

---

**Important:** All the examples above use automatic *sizing* and default *scale*.

---

### 2.1.4 Image render size

The *render size* of an image is the dimension with which an image is rendered.

The *render size* can be retrieved via the *size*, *width* and *height* properties.

The *render size* of an image can be in either of two states:

1. Set

The size is said to be *set* when the image has a fixed size.

In this state, the *size* property is a tuple of integers, the *width* and *height* properties are integers.

2. Unset

The size is said to be *unset* when the image doesn't have a fixed size.

In this case, the size with which the image is rendered is automatically calculated (based on the current *term: `terminal size`*) whenever the image is to be rendered.

In this state, the *size*, *width* and *height* properties are *None*.

The *render size* of an image can be set when creating the instance by passing valid values to **either** the *width* **or** the *height* **keyword-only** parameter.

For whichever axis is given, the other axis is calculated **proportionally**.

---

#### Note:

1. The arguments can only be given **by keyword**.
  2. If neither is given, the size is *unset*.
  3. All methods of instantiation accept these arguments.
- 

For example:

```
>>> image = TermImage.from_file("python.png") # Unset
>>> image.size is None
True
>>> image = TermImage.from_file("python.png", width=60) # width is given
>>> image.size
(60, 60)
>>> image.height
60
>>> image = TermImage.from_file("python.png", height=56) # height is given
>>> image.size
(56, 56)
>>> image.width
56
```

The resulting size must fit into the terminal window

```
>>> image = TermImage.from_file("python.png", height=136) # (terminal_height - 2) * 2;
↳ Still OK
>>> image.size
(136, 136)
>>> image = TermImage.from_file("python.png", height=137) # Not OK
Traceback (most recent call last):
.
.
.
term_img.exceptions.InvalidSize: The resulting rendered size will not fit into the
↳ available size
```

An exception is raised when both *width* and *height* are given.

```
>>> image = TermImage.from_file("python.png", width=100, height=100)
Traceback (most recent call last):
.
.
.
ValueError: Cannot specify both width and height
```

The *width* and *height* properties are used to set the *render size* of an image after instantiation.

```
>>> image = TermImage.from_file("python.png") # Unset
>>> image.size is None
True
>>> image.width = 56
>>> image.size
(56, 56)
>>> image.height
56
>>> image.height = 136
>>> image.size
(136, 136)
>>> image.width
136
>>> image.width = 200 # Even though the terminal can contain this width, it can't
↳ contain the resulting height
Traceback (most recent call last):
.
.
.
term_img.exceptions.InvalidSize: The resulting rendered size will not fit into the
↳ available size
```

Setting width or height to None sets the size to that automatically calculated based on the current *terminal size*.

```
>>> image = TermImage.from_file("python.png") # Unset
>>> image.size is None
True
>>> image.width = None
>>> image.size
(136, 136)
```

(continues on next page)

(continued from previous page)

```
>>> image.width = 56
>>> image.size
(56, 56)
>>> image.height = None
>>> image.size
(136, 136)
```

The `size` property can only be set to one value, `None` and doing this *unsets* the *render size*.

```
>>> image = TermImage.from_file("python.png", width=100)
>>> image.size
(100, 100)
>>> image.size = None
>>> image.size is image.width is image.height is None
True
```

---

### Important:

1. The currently set *font ratio* is also taken into consideration when calculating or validating sizes.
2. **By default**, the resulting size must not exceed the *terminal size* i.e for both the given axis and the axis automatically calculated.
3. The *height* is actually **about twice the number of lines** that'll be used to draw the image, assuming the *y-axis scale* is 1.0 (we'll get to that).
4. There is a **default** 2-line *vertical allowance*, to allow for shell prompts or the likes.

Therefore, **by default**, only `terminal_height - 2` lines are available i.e the maximum height is `(terminal_height - 2) * 2`.

---

**Hint:** See `TermImage.set_size()` for advanced sizing control.

---

## 2.1.5 Image render scale

The *render scale* of an image is the **fraction** of the *render size* that'll actually be used to render the image. A valid scale value is a float in the range  $0 < x \leq 1$  i.e greater than zero and less than or equal to one.

The *render scale* can be retrieved via the properties `scale`, `scale_x` and `scale_y`.

The scale can be set at instantiation by passing a value to the `scale` **keyword-only** paramter.

```
>>> image = TermImage.from_file("python.png", scale=(0.75, 0.6))
>>> image.scale
>>> (0.75, 0.6)
```

The rendered result (using `image.draw()`) should look like:



If the *scale* argument is omitted, the default scale (1.0, 1.0) is used.

```
>>> image = Termimage.from_file("python.png")
>>> image.scale
>>> (1.0, 1.0)
```

The rendered result (using `image.draw()`) should look like:





The properties `scale`, `scale_x` and `scale_y` are used to set the *render scale* of an image after instantiation.

`scale` accepts a tuple of two scale values or a single scale value.

`scale_x` and `scale_y` each accept a single scale value.

```
>>> image = Termimage.from_file("python.png")
>>> image.scale = (.3, .56756)
>>> image.scale
(0.3, 0.56756)
>>> image.scale = .5
>>> image.scale
(0.5, 0.5)
>>> image.scale_x = .75
>>> image.scale
(0.75, 0.5)
>>> image.scale_y = 1.
>>> image.scale
(0.75, 1.0)
```

Finally, to explore more of the library's features and functionality, check out the [Reference](#) section.

## 2.2 Reference

### 2.2.1 Core Library Definitions

The `term_img.image` module defines the following:

**class** `term_img.image.TermImage`(*image*, \*, *width*=None, *height*=None, *scale*=(1.0, 1.0))

Bases: `object`

Text-printable image

#### Parameters

- **image** (*PIL.Image.Image*) – Image to be rendered.
- **width** (*Optional[int]*) – The width to render the image with.
- **height** (*Optional[int]*) – The height to render the image with.
- **scale** (*Tuple[float, float]*) – The image render scale on respective axes.

#### Raises

- **TypeError** – An argument is of an inappropriate type.
- **ValueError** – An argument has an unexpected/invalid value.

Propagates exceptions raised by `set_size()`, if *width* or *height* is given.

---

#### Note:

- *width* is not necessarily the exact number of columns that'll be used to render the image. That is influenced by the currently set *font ratio*.

- *height* is **2 times** the number of lines that'll be used in the terminal.
  - If neither is given or both are `None`, the size is automatically determined when the image is to be *rendered*, such that it can fit within the terminal.
  - The *size* is multiplied by the *scale* on each axis respectively before the image is *rendered*.
- 

**property closed**

Instance finalization status

**property frame\_duration**

Duration (in seconds) of a single frame for *animated* images

Setting this on non-animated images is simply ignored, no exception is raised.

**property height**

Image *render height*

None when *render size* is *unset*.

Settable values:

- `None`: Sets the render size to the automatically calculated one.
- A positive `int`: Sets the render height to the given value and the width proportionally.

The image is actually *rendered* using half this number of lines

**property is\_animated**

True if the image is *animated*. Otherwise, False.

**property original\_size**

Original image size

**property n\_frames**

The number of frames in the image

**property rendered\_height**

The number of lines that the drawn image will occupy in a terminal

**property rendered\_size: Tuple[int, int]**

The number of columns and lines (respectively) that the drawn image will occupy in a terminal

**property rendered\_width**

The number of columns that the drawn image will occupy in a terminal

**property scale**

Image *render scale*

Settable values are:

- A *scale value*; sets both axes.
- A tuple of two *scale values*; sets (*x*, *y*) respectively.

A scale value is a float in the range `0.0 < value <= 1.0`.

**property scale\_x**

x-axis *render scale*

A scale value is a float in the range `0.0 < x <= 1.0`.

**property scale\_y**

y-axis *render scale*

A scale value is a float in the range `0.0 < y <= 1.0`.

**property size**

Image *render size*

None when render size is unset.

Setting this to None *unsets* the *render size* (so that it's automatically calculated whenever the image is *rendered*) and resets the recognized advanced sizing options to their defaults.

**property source**

The *source* from which the instance was initialized

Can be a PIL image, file path or URL.

**property width**

Image *render width*

None when *render size* is *unset*.

Settable values:

- None: Sets the render size to the automatically calculated one.
- A positive int: Sets the render width to the given value and the height proportionally.

**close()**

Finalizes the instance and releases external resources.

**Note:**

- It's not necessary to explicitly call this method, as it's automatically called when necessary.
- This method can be safely called multiple times.
- If the instance was initialized with a PIL image, the PIL image is never finalized.

**Return type** None

**draw**(*h\_align=None, pad\_width=None, v\_align=None, pad\_height=None, alpha=0.1568627450980392, \*, animate=True, ignore\_oversize=False*)

Draws/Displays an image in the terminal, with optional *alignment* and *padding*.

**Parameters**

- **h\_align** (*Optional[str]*) – Horizontal alignment (“left”/”<”, “center”/”|” or “right”/”>”). Default: center.
- **pad\_width** (*Optional[int]*) – Number of columns within which to align the image.
  - Excess columns are filled with spaces.
  - default: terminal width.
- **v\_align** (*Optional[str]*) – Vertical alignment (“top”/”^”, “middle”/”-” or “bottom”/”\_”). Default: middle.
- **pad\_height** (*Optional[int]*) – Number of lines within which to align the image.
  - Excess lines are filled with spaces.
  - default: terminal height, with a 2-line allowance.
- **alpha** (*Optional[float]*) – Transparency setting.
  - If None, transparency is disabled (i.e black background).

- If a float ( $0.0 \leq x < 1.0$ ), specifies the alpha ratio **above** which pixels are taken as *opaque*.
- If a string, specifies a **hex color** with which transparent background should be replaced.
- **animate** (*bool*) – If False, disable animation i.e draw only the current frame of an *animated* image.
- **ignore\_oversize** (*bool*) – If True, do not verify if the image will fit into the *available terminal size* with it's currently set *render size*.

**Raises**

- **TypeError** – An argument is of an inappropriate type.
- **ValueError** – An argument has an unexpected/invalid value.
- **ValueError** – *Render size* or *scale* too small.
- **term\_img.exceptions.InvalidSize** – The terminal has been resized in such a way that the previously set size can no longer fit into it.
- **term\_img.exceptions.InvalidSize** – The image is *animated* and the previously set size won't fit into the *available terminal size*.

**Return type** None

---

**Note:**

- Animations, if not disabled, are infinitely looped but can be terminated with Ctrl-C (SIGINT or “KeyboardInterrupt”).
  - If *set\_size()* was previously used to set the *render size* (directly or not), the last values of its *check\_height*, *h\_allow* and *v\_allow* parameters are taken into consideration, with *check\_height* applying to only non-animated images.
  - For animated images, when *animate* is True:
    - *Render size* and *padding height* are always validated.
    - *ignore\_oversize* has no effect.
- 

**classmethod** *from\_file*(*filepath*, *\*\*kwargs*)Creates a *TermImage* instance from an image file.**Parameters**

- **filepath** (*str*) – Relative/Absolute path to an image file.
- **kwargs** (*Union[int, None, Tuple[float, float]]*) – Same keyword arguments as the class constructor.

**Returns** A new *TermImage* instance.**Raises**

- **TypeError** – *filepath* is not a string.
- **FileNotFoundError** – The given path does not exist.
- **IsADirectoryError** – Propagated from from *PIL.Image.open()*.
- **UnidentifiedImageError** – Propagated from from *PIL.Image.open()*.

**Return type** *term\_img.image.TermImage*

Also Propagates exceptions raised or propagated by the class constructor.

**classmethod** `from_url(url, **kwargs)`

Creates a [TermImage](#) instance from an image URL.

#### Parameters

- **url** (*str*) – URL of an image file.
- **kwargs** (*Union[int, None, Tuple[float, float]]*) – Same keyword arguments as the class constructor.

**Returns** A new [TermImage](#) instance.

#### Raises

- **TypeError** – *url* is not a string.
- **ValueError** – The URL is invalid.
- **[term\\_img.exceptions.URLNotFoundError](#)** – The URL does not exist.
- **PIL.UnidentifiedImageError** – Propagated from `PIL.Image.open()`.

**Return type** [term\\_img.image.TermImage](#)

Also propagates connection-related exceptions from `requests.get()` and exceptions raised or propagated by the class constructor.

---

**Note:** This method creates a temporary image file, but only after a successful initialization.

Proper clean-up is guaranteed except maybe in very rare cases.

To ensure 100% guarantee of clean-up, use the object as a [context manager](#).

---

**seek**(*pos*)

Changes current image frame.

**Parameters** **pos** (*int*) – New frame number.

#### Raises

- **TypeError** – An argument is of an inappropriate type.
- **ValueError** – An argument has an unexpected/invalid value but of an appropriate type.

**Return type** `None`

Frame numbers start from 0 (zero).

**set\_size**(*width=None, height=None, h\_allow=0, v\_allow=2, \*, maxsize=None, check\_width=True, check\_height=True*)

Sets the [render size](#) with advanced control.

#### Parameters

- **width** (*Optional[int]*) – [Render width](#) to use.
- **height** (*Optional[int]*) – [Render height](#) to use.
- **h\_allow** (*int*) – Horizontal allowance i.e minimum number of columns to leave unused.
- **v\_allow** (*int*) – Vertical allowance i.e minimum number of lines to leave unused.
- **maxsize** (*Optional[Tuple[int, int]]*) – If given (cols, lines), it's used instead of the terminal size.

- **check\_width** (*bool*) – If *False*, the validity of the resulting *rendered width* is not checked.
- **check\_height** (*bool*) – If *False*, the validity of the resulting *rendered height* is not checked.

**Raises**

- **TypeError** – An argument is of an inappropriate type.
- **ValueError** – An argument has an unexpected/invalid value but of an appropriate type.
- **ValueError** – Both *width* and *height* are specified.
- **ValueError** – The *available size* is too small.
- **term\_img.exceptions.InvalidSize** – The resulting *render size* is too small.
- **term\_img.exceptions.InvalidSize** – The resulting *rendered size* will not fit into the *available terminal size* (or *maxsize*, if given).

**Return type** None

If neither *width* nor *height* is given or anyone given is *None*:

- and *check\_height* and *check\_width* are both *True*, the size is automatically calculated to fit within the *available* terminal size (or *maxsize*, if given).
- and *check\_height* is *False*, the size is set such that the *rendered width* is exactly the *available* terminal width or *maxsize[0]* (assuming the *render scale* equals 1), regardless of the *font ratio*.
- and *check\_width* is *False* (and *check\_height* is *True*), the size is set such that the *rendered height* is exactly the *available* terminal height or *maxsize[1]* (assuming the *render scale* equals 1), regardless of the *font ratio*.

*Allowance* does not apply when *maxsize* is given.

No *vertical allowance* when *check\_height* is *False*.

No *horizontal allowance* when *check\_width* is *False*.

The *check\_height* might be set to *False* to set the *render size* for vertically-oriented images (i.e images with *height* > *width*) such that the drawn image spans more columns but the terminal window has to be scrolled to view the entire image.

All image rendering and formatting methods recognize and respect the *check\_height*, *h\_allow* and *v\_allow* options, until the size is re-set or *unset*.

*check\_width* is only provided for completeness, it should probably be used only when the image will not be drawn to the current terminal. The value of this parameter is **not** recognized by any other method or operation.

**tell()**

Returns the current image frame number.

**Return type** int

---

**Note:** It's allowed to set properties for *animated* images on non-animated ones, the values are simply ignored.

---

## 2.2.2 Context Manager Support

TermImage instances are context managers i.e they can be used with the `with` statement such as in:

```
with TermImage.from_url(url) as image:
    ...
```

Using an instance as a context manager ensures **100% guarantee** to of **object finalization** (i.e clean-up/release of resources), especially for instances with URL sources (see [`TermImage.from\_url\(\)`](#)).

## 2.2.3 Custom Exceptions

The `term_img.exceptions` module defines the following:

**exception** `term_img.exceptions.TermImageException`

Bases: `Exception`

Package exception baseclass

**exception** `term_img.exceptions.URLNotFoundError`

Bases: `FileNotFoundError`, `term_img.exceptions.TermImageException`

Raised for 404 errors

**exception** `term_img.exceptions.InvalidSize`

Bases: `ValueError`, `term_img.exceptions.TermImageException`

Raised when the given/set image render size is larger than the terminal size

## 2.2.4 Top-Level Functions

`term_img.get_font_ratio()`

Return the set library-wide *font ratio*

**Return type** `float`

`term_img.set_font_ratio(ratio)`

Set the library-wide font ratio

**Parameters** `ratio` (*float*) – The aspect ratio of your terminal's font i.e *width / height* of a single character cell.

**Return type** `None`

This value is taken into consideration when rendering images in order for images drawn to the terminal to have a proper perceived scale.

If you can't determine this value from your terminal's configuration, you might have to try different values till you get a good fit. Normally, this value should be between 0 and 1, but not too close to either.

## 2.2.5 Image Format Specification

`[h_align] [width] [ . [v_align] [height] ] [ # [threshold | bgcolor] ]`

---

### Note:

- The spaces are only for clarity and not included in the syntax.
  - Fields within `[ ]` are optional.
  - `|` implies mutual exclusivity.
  - `width` and `height` are in units of columns and lines respectively.
  - If the *padding width* or *padding height* is less than or equal to the image's *rendered width* or *rendered height* respectively, the padding has **no effect**.
- 

- `h_align`: This can be one of:
  - `<` → left
  - `|` → center
  - `>` → right
  - *absent* → center
- `width`: Integer padding width (default: *terminal width* minus *horizontal allowance*)
  - Must not be greater than the *terminal width*.
- `v_align`: This can be one of:
  - `^` → top
  - `-` → middle
  - `_` → bottom
  - *absent* → middle
- `height`: Integer padding height (default: *terminal height* minus *vertical allowance*)
  - Must not be greater than the *terminal height* for *animated* images.
- `#`: Transparency setting:
  - If absent, transparency is enabled.
  - `threshold`: Alpha ratio above which pixels are taken as opaque e.g. `.0`, `.325043`, `.99999`. The value must be in the range `0.0 <= threshold < 1.0`.
  - `bgcolor`: Hex color with which transparent background should be replaced e.g. `ffffff`, `7faa52`.
  - If neither `threshold` nor `bgcolor` is present, but `#` is present, transparency is disabled i.e. the image has a **black background**.

See *Formatted rendering* for examples.



## 2.3 Known Issues

## 2.4 Planned Features

In no particular order:

- Performance improvements
- Greyscale render (Support for 256-color terminals)
- `AsciiImage` class, renders images in greyscale using ASCII characters only (Support for terminals without unicode or 24-bit color support)
- Add urwid widgets for displaying images
- Animated image iterators



## IMAGE VIEWER

### 3.1 Text-based User Interface

The TUI is developed using `urwid`.

#### 3.1.1 UI Components

The UI consists of various areas which are each composed using one or more widgets. The components of the UI might change depending on the current *context* and some *actions*.

The following are the key components that make up the UI.

- **Banner:**
  - At the top of the UI.
  - Fixed height of 4 lines.
  - Contains the project title with a surrounding color fill and a line-box decoration.
  - Hidden in full image views.
- **Viewer:**
  - Immediately below the title banner.
  - Consists of two sub-components (described below) arranged horizontally: \* Menu \* View
- **Menu:**
  - Sub-component of the *viewer* to the left.
  - Fixed width of 20 columns.
  - Contains a list of image and directory entries which can be scrolled through.
  - Used to scroll through images in a directory and navigate back and forth through directories, among other actions.
- **View:**
  - Sub-component of the *viewer* to the right.
  - Images are displayed in here.
  - The content can be one of these two, depending on the type of item currently selected in the *menu*: \* An image: When the item selected in the menu is an image. \* An image grid: When the item selected in the menu is a directory.

- The *view* component can also be used to scroll through images.

- **Notification Bar:**

- Immediately above the *Action/Key Bar*.
- Fixed height of 2 lines.
- Notifications about various events are displayed here.
- Hidden in full image views.

- **Action/Key Bar:**

- Contains a list of *actions* in the current *context*.
- Each action has the symbol of the assigned key beside its name.
- If the actions are too much to be listed on one line, the bar can be expanded/collapsed using the key indicated at the far right.

- **Overlays:**

- These are used for various purposes such as help menu, confirmations, etc.
- They are shown only when certain actions are triggered.

- **Info Bar:**

- Used for debugging.
- This is a 1-line bar immediately above the action/key bar.
- Only shows (in all views) when the `--debug` option is specified.

Full/Maximized image views consist of only the *view* and *action/key bar* components.

### 3.1.2 Contexts

A context is simply a set of *actions*.

The active context might change due to one of these:

- Triggering certain *actions*.
- Change of *viewer* sub-component (i.e *menu* or *view*) in focus.
- Type of menu entry selected.
- An overlay is shown.

The active context determines which actions are available and displayed in the *action/key bar* at the bottom of the UI.

The following are the contexts available:

- **global:** The actions in this context are available when any other context is active, with a few exceptions.
- **menu:** This context is active when the *menu* UI component is in focus and non-empty.
- **image:** This context is active if the *view* UI component is in focus and was switched to (from the *menu*) while an image entry was selected.
- **image-grid:** This context is active if the *view* UI component is in focus and was switched to (from the *menu*) while a directory entry was selected.
- **full-image:** This context is active when an image entry is maximized from the *image* context (using the *Maximize* action) or from the *menu* context using the *Open* action.

- **full-grid-image:** This context is active when an image grid cell is maximized from the `image-grid` context (using the `Open` action).
- **confirmation:** This context is active only when specific actions that require confirmation are triggered e.g the `Delete` action in some contexts.
- **overlay:** This context is active only when an overlay UI component (e.g the help menu) is shown.

### 3.1.3 Actions

An action is a single entry in a *context*, it represents a functionality available in that context.

An action has the following defining properties:

- **name:** The name of the action.
- **key:** The key/combination used to trigger the action.
- **symbol:** A string used to represent the *key*.
- **description:** A brief description of what the action does.
- **visibility:** Determines if the action is displayed in the *action/key bar* or not.
- **state:** Determines if the action is enabled or not. \* If an action is disabled, pressing its *key* will trigger the terminal bell.

---

**Note:** All contexts and their actions (with default properties) are defined in `_context_keys` in the `term_img.config` sub-module.

---

## 3.2 Configuration

The configuration is divided into the following categories:

- Options
- Keys

The configuration is stored in the JSON format in a file located at `~/.term_img/config.json`.

### 3.2.1 Config Options

These are fields whose values control various behaviours of the viewer. They are as follows:

- **cell width:** The initial width of (no of columns for) grid cells, in the TUI.
  - Type: integer
  - Valid values:  $x > 0$
- **font ratio:** The *font ratio* used when `--font-ratio` CLI option is not specified.
  - Type: float
  - Valid values:  $x > 0.0$
- **frame duration:** The the time (in seconds) between frames of an animated image, when `--frame-duration` CLI option is not specified.

- Type: float
- Valid values:  $x > 0.0$
- **max pixels:** The maximum amount of pixels in images to be displayed in the TUI, when `--max-pixels` CLI option is not specified.
  - Type: integer
  - Valid values:  $x > 0$
  - Any image having more pixels than the specified maximum will be replaced with a placeholder when displayed but can still be forced to display or viewed externally.
  - Note that increasing this will have adverse effects on performance.

---

**Important:** The `version` field is not a config option, it's used for config file updates and should not be tampered with.

---

### 3.2.2 Key Config

The key assigned to every *action* can be modified in the config file.

The "keys" field in the configuration holds a mapping containing fields each mapping a *context* to a mapping of *actions* to their properties.

The format of the "keys" mapping is thus:

```
{
  "<context>": {
    "<action>": [
      "<key>",
      "<symbol>"
    ],
    ...
  },
  ...
}
```

'...' means continuous repetition of the format occurs.

*action* is the *name* of the action. It should not be modified.

Any or both of *key* and *symbol* can be changed. Both must be valid Python strings, hence Unicode characters are supported.

---

**Hint:** If using a Unicode character that occupies multiple columns in *symbol*, then add spaces after it as required to cover-up for the extra columns.

---

---

**Note:** The `navigation` field is not actually a *context*, instead it's the universal navigation controls configuration from which navigation *actions* in actual *contexts* are updated.

---

Here is a pre-configured config with Vim-style key-bindings (majorly navigation).  
*Remember to rename the file to `config.json`.*

Below is a list of all **valid** values for *key*:

```
" "  
"! "  
"" "  
"# "  
"$ "  
"% "  
"& "  
" " "  
" (" "  
") "  
"* "  
" + "  
" ' "  
" _ "  
" . "  
" / "  
" 0 "  
" 1 "  
" 2 "  
" 3 "  
" 4 "  
" 5 "  
" 6 "  
" 7 "  
" 8 "  
" 9 "  
" : "  
" ; "  
" < "  
" = "  
" > "  
" ? "  
" @ "  
" [ "  
" \ "  
" ] "  
" ^ "  
" _ "  
" ` "  
" A "  
" a "
```

(continues on next page)

(continued from previous page)

```
"ctrl a"  
"B"  
"b"  
"ctrl b"  
"C"  
"c"  
"D"  
"d"  
"ctrl d"  
"E"  
"e"  
"ctrl e"  
"F"  
"f"  
"ctrl f"  
"G"  
"g"  
"ctrl g"  
"H"  
"h"  
"ctrl h"  
"I"  
"i"  
"ctrl i"  
"J"  
"j"  
"ctrl j"  
"K"  
"k"  
"ctrl k"  
"L"  
"l"  
"ctrl l"  
"M"  
"m"  
"ctrl m"  
"N"  
"n"  
"ctrl n"  
"O"  
"o"  
"ctrl o"  
"P"  
"p"  
"ctrl p"  
"Q"  
"q"  
"ctrl q"  
"R"  
"r"  
"ctrl r"  
"S"
```

(continues on next page)



(continued from previous page)

```
"s"
"ctrl s"
"T"
"t"
"ctrl t"
"U"
"u"
"ctrl u"
"V"
"v"
"ctrl v"
"W"
"w"
"ctrl w"
"X"
"x"
"ctrl x"
"Y"
"y"
"ctrl y"
"Z"
"z"
"{"
"|"
"}"
"~"
"f1"
"ctrl f1"
"shift f1"
"shift ctrl f1"
"f2"
"ctrl f2"
"shift f2"
"shift ctrl f2"
"f3"
"ctrl f3"
"shift f3"
"shift ctrl f3"
"f4"
"ctrl f4"
"shift f4"
"shift ctrl f4"
"f5"
"ctrl f5"
"shift f5"
"shift ctrl f5"
"f6"
"ctrl f6"
"shift f6"
"shift ctrl f6"
"f7"
"ctrl f7"
```

(continues on next page)

(continued from previous page)

```
"shift f7"
"shift ctrl f7"
"f8"
"ctrl f8"
"shift f8"
"shift ctrl f8"
"f9"
"ctrl f9"
"shift f9"
"shift ctrl f9"
"up"
"ctrl up"
"shift up"
"shift ctrl up"
"end"
"ctrl end"
"shift end"
"shift ctrl end"
"esc"
"f10"
"ctrl f10"
"shift f10"
"shift ctrl f10"
"f11"
"ctrl f11"
"shift f11"
"shift ctrl f11"
"f12"
"ctrl f12"
"shift f12"
"shift ctrl f12"
"tab"
"down"
"ctrl down"
"shift down"
"shift ctrl down"
"home"
"ctrl home"
"shift home"
"shift ctrl home"
"left"
"ctrl left"
"shift left"
"shift ctrl left"
"enter"
"right"
"ctrl right"
"shift right"
"shift ctrl right"
"delete"
"ctrl delete"
"shift delete"
```

(continues on next page)

(continued from previous page)

```
"shift ctrl delete"
"insert"
"backspace"
"page up"
"ctrl page up"
"page down"
"ctrl page down"
```

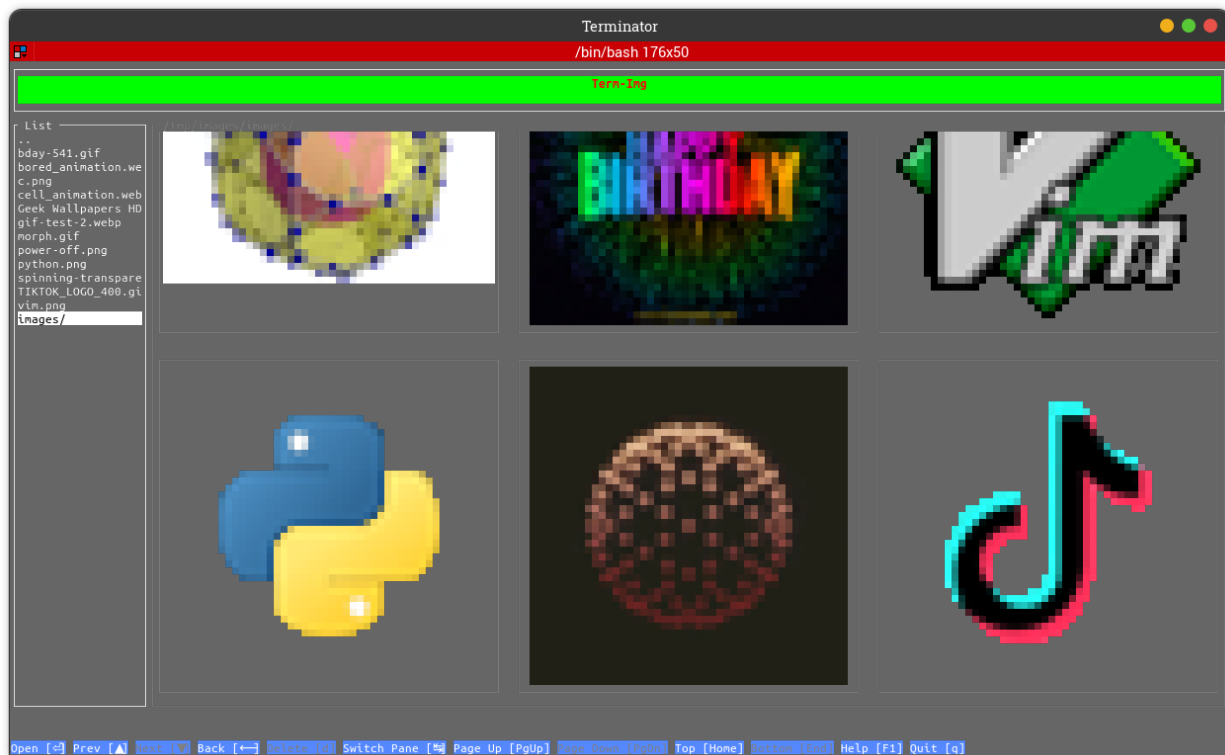
Any values other than these will be flagged as invalid and the default will be used instead (if possible), for that session.

### Important:

1. Keys used in navigation or global contexts cannot be used in any other context.
2. All keys in a context must be unique.
3. If a key is invalid or already used, the default is tried as a fallback but if that fails (because it already used), the session is terminated.

The package comes with a standalone in-terminal image viewer based on the library.

The image viewer is started from the command line using either the `term-img` command (only works if the Python scripts directory is on PATH) or `python -m term_img`.



## 3.3 Image sources

The viewer accepts the following kinds of sources:

- An image file on a local filesystem.
- A directory on a local filesystem.
- An Image URL.

Any other thing given as a *source* is simply reported as invalid.

## 3.4 Modes

The viewer can be used in two modes:

### 1. CLI mode

In this mode, images are directly printed to standard output.

This mode is used whenever there is only a single image source or when the `--cli` option is specified.

### 2. TUI mode

In this mode, a Terminal/Text-based User Interface is launched, within which images and directories can be browsed and viewed in different ways.

This mode is used whenever there are multiple image sources or at least one directory source, or when the `--tui` option is specified.

## 3.5 Usage

Run `term-img` with the `--help` option to see the usage info and help text.

All arguments and options are described there.

Note that some options are only applicable to a specific mode. If used with the other mode, they're simply ignored.

Some options have a `[N]` (where *N* is a number) behind their description, it indicates that the option has a footnote attached.

All footnotes are at the bottom of the help text.

## 3.6 Notifications

Notifications are event reports meant to be brought to the immediate knowledge of the user. Notifications have two possible destinations:

- Standard output: This is used while the TUI is **not** launched.
- TUI *notification bar*: This is used while the TUI is launched.

Notifications sent to the TUI's *notification bar* automatically disappear after 5 seconds.

## 3.7 Logging

Logs are more detailed event reports meant for troubleshooting and debugging purposes.

Logs are written to a file on a local filesystem. The default log file is `~/ .term_img/term_img.log` but a different file can be specified (for a single session) using the `--log` CLI option.

A log entry has the following format:

```
(<pid>) (<date> <time>) [<level>] <module>: <function>: <message>
```

- *pid*: The process ID of the term-img session.
- *date* and *time*: Current system date and time in the format `%d-%m-%Y %H:%M:%S`.
- *level*: The level of the log entry, this indicates it's importance.
- *module*: The package sub-module from which it originated.
- *function*: The function from which it originated.
  - Only present when running on **Python 3.8+** and *logging level* is set to `DEBUG` (either by `--debug` or `--log-level=DEBUG`).
- *message*: The actual report describing the event that occurred.

---

### Note:

- Certain logs and some extra info are only provided when *logging level* is set to `DEBUG`.
  - Log files are **appended to**, so it's safe use the same file for multiple sessions.
  - Logs are rotated upon reaching a size of **1MiB**.
    - Only the current and immediate previous log file are kept.
  - The Process ID of the `term-img` instance precedes every log entry, so this can be used to distinguish and track logs from different sessions running simultaneously while using the same log file.
-

## 3.8 Known Issues

## 3.9 Planned Features

In no particular order:

- Performance improvements
- Overlay support for Image widgets
- Open in external viewer
- Theme customization
- Grid cell animations
- Config menu
- Pattern-based file and directory exclusion
- Minimum and maximum file size
- Optionally skipping symlinks
- Grid cell showing number of subfolders
- Slideshow
- Zoom/Pan
- Frame duration adjustment per animated image
- Find
- Filter
- Alpha background adjustment per image
- **Copy:**
  - Image
  - File/Directory name
  - Full path
  - Parent directory path

**Why?**

- Why not?
- To improve and extend the capabilities of CLI and TUI applications.
- Terminals are here to stay!

**What about Windows support?**

- Firstly, only the new [Windows Terminal](#) seems to have proper ANSI support and modern terminal emulator features.
- The library and CLI-only mode currently work on Windows (i.e using CMD or Powershell) if the other requirements are satisfied but can't guarantee it'll always be so.
- The TUI doesn't work due to lack of [urwid](#) support.
- If stuck on Windows, you could use WSL + Windows Terminal.

**Why are colours not properly reproduced?**

- Some terminals support 24-bit colors but have a **256-color palette**. This limits color reproduction.

**Why do images look out-of-scale in my terminal?**

- Simply adjust your *font ratio* setting appropriately.

**Why is the TUI unresponsive or slow in drawing images?**

- Drawing (not rendering) speed is **entirely** dependent on the terminal emulator itself.
- Some terminal emulators block upon input, so rapidly repeated input could cause the terminal to be unresponsive.





## GLOSSARY

Below are definitions of terms used across the library's public interface, exception messages, CLI help text and the documentation.

---

**Note:** For contributors, these terms are also used in the source code, as variable names, in comments, docstrings, etc.

---

**alignment** The position to place a rendered image within its padding.

**allowance** The amount of space to be left un-used in a given maximum size.

**alpha threshold** Alpha ratio/value above which a pixel is taken as **opaque**.

**animated** Having multiple frames.

The frames of an animated image are generally meant to be displayed in rapid succession, to give the effect of animation.

**available height** The remainder after vertical allowance is subtracted from the maximum amount of lines.

**available size** The remainder after *allowances* are subtracted from the maximum size.

**available width** The remainder after horizontal allowance is subtracted from the maximum amount of columns.

**font ratio** The **aspect ratio** of a terminal's font i.e the ratio of **width to height** of a single **character cell** on the terminal.

See also: *get\_font\_ratio()* and *set\_font\_ratio()*.

**horizontal alignment** The position to place a rendered image within its *padding width*.

**horizontal allowance** The amount of **columns** to be left un-used in a given maximum amount of columns.

### padding

**padding width** Amount of columns within which to fit an image. Excess columns on either or both sides of the image (depending on the *horizontal alignment*) will be filled with spaces.

**padding height** Amount of columns within which to fit an image. Excess columns on either or both sides of the image (depending on the *vertical alignment*) will be filled with spaces.

**pixel ratio** It is equivalent to the *font ratio* multiplied by 2, since there are two pixels (arranged vertically) in one character cell.

### render

**rendered** To convert image pixel data into a **string** (optionally including escape sequences to produce colour and transparency).

**render height** The real **vertical** dimension (in pixels) with which an image is rendered.

**render size** The real dimension (in pixels) with which an image is rendered.

**render width** The real **horizontal** dimension (in pixels) with which an image is rendered.

**rendered height** The amount of **lines** that'll be occupied by a rendered image **when drawn onto a terminal screen**.

**rendered size** The amount of space (columns and lines) that'll be occupied by a rendered image **when drawn onto a terminal screen**.

This is determined by the *render size* and *scale* of an image and the global *font ratio*.

**rendered width** The amount of **columns** that'll be occupied by a rendered image **when drawn onto a terminal screen**.

**scale**

**render scale** The fraction of an image's *render size* that'll actually be used to *render* it.

See also: *Image render scale*.

**source** The resource from which an image is derived.

**terminal height** The amount of lines on a terminal screen at a time i.e without scrolling.

**terminal size** The amount of columns and lines on a terminal screen at a time i.e without scrolling.

**terminal width** The amount of columns on a terminal screen at a time.

**vertical alignment** The position to place a rendered image within its *padding height*.

**vertical allowance** The amount of **lines** to be left un-used in a given maximum amount of lines.

## INDICES AND TABLES

- *Glossary*
- `genindex`
- `modindex`
- `search`



## PYTHON MODULE INDEX

### t

`term_img.exceptions`, [19](#)

`term_img.image`, [13](#)



## A

alignment, [37](#)  
 allowance, [37](#)  
 alpha threshold, [37](#)  
 animated, [37](#)  
 available height, [37](#)  
 available size, [37](#)  
 available width, [37](#)

## C

close() (*term\_img.image.TermImage* method), [15](#)  
 closed (*term\_img.image.TermImage* property), [14](#)

## D

draw() (*term\_img.image.TermImage* method), [15](#)

## F

font ratio, [37](#)  
 frame\_duration (*term\_img.image.TermImage* property), [14](#)  
 from\_file() (*term\_img.image.TermImage* class method), [16](#)  
 from\_url() (*term\_img.image.TermImage* class method), [17](#)

## G

get\_font\_ratio() (in module *term\_img*), [19](#)

## H

height (*term\_img.image.TermImage* property), [14](#)  
 horizontal alignment, [37](#)  
 horizontal allowance, [37](#)

## I

InvalidSize, [19](#)  
 is\_animated (*term\_img.image.TermImage* property), [14](#)

## M

module  
     *term\_img.exceptions*, [19](#)  
     *term\_img.image*, [13](#)

## N

n\_frames (*term\_img.image.TermImage* property), [14](#)

## O

original\_size (*term\_img.image.TermImage* property), [14](#)

## P

padding, [37](#)  
 padding height, [37](#)  
 padding width, [37](#)  
 pixel ratio, [37](#)

## R

render, [37](#)  
 render height, [37](#)  
 render scale, [38](#)  
 render size, [38](#)  
 render width, [38](#)  
 rendered, [37](#)  
 rendered height, [38](#)  
 rendered size, [38](#)  
 rendered width, [38](#)  
 rendered\_height (*term\_img.image.TermImage* property), [14](#)  
 rendered\_size (*term\_img.image.TermImage* property), [14](#)  
 rendered\_width (*term\_img.image.TermImage* property), [14](#)

## S

scale, [38](#)  
 scale (*term\_img.image.TermImage* property), [14](#)  
 scale\_x (*term\_img.image.TermImage* property), [14](#)  
 scale\_y (*term\_img.image.TermImage* property), [14](#)  
 seek() (*term\_img.image.TermImage* method), [17](#)  
 set\_font\_ratio() (in module *term\_img*), [19](#)  
 set\_size() (*term\_img.image.TermImage* method), [17](#)  
 size (*term\_img.image.TermImage* property), [14](#)  
 source, [38](#)  
 source (*term\_img.image.TermImage* property), [15](#)

## T

`tell()` (*term\_img.image.TermImage* method), [18](#)

`term_img.exceptions`

    module, [19](#)

`term_img.image`

    module, [13](#)

`TermImage` (class in *term\_img.image*), [13](#)

`TermImageException`, [19](#)

terminal height, [38](#)

terminal size, [38](#)

terminal width, [38](#)

## U

`URLNotFoundError`, [19](#)

## V

vertical alignment, [38](#)

vertical allowance, [38](#)

## W

`width` (*term\_img.image.TermImage* property), [15](#)