simpil Documentation

Hywel Thomas; Oli Davies

Oct 24, 2022

Contents:

Simpil simplifies the loading/creation of images and adding text using PIL/Pillow.

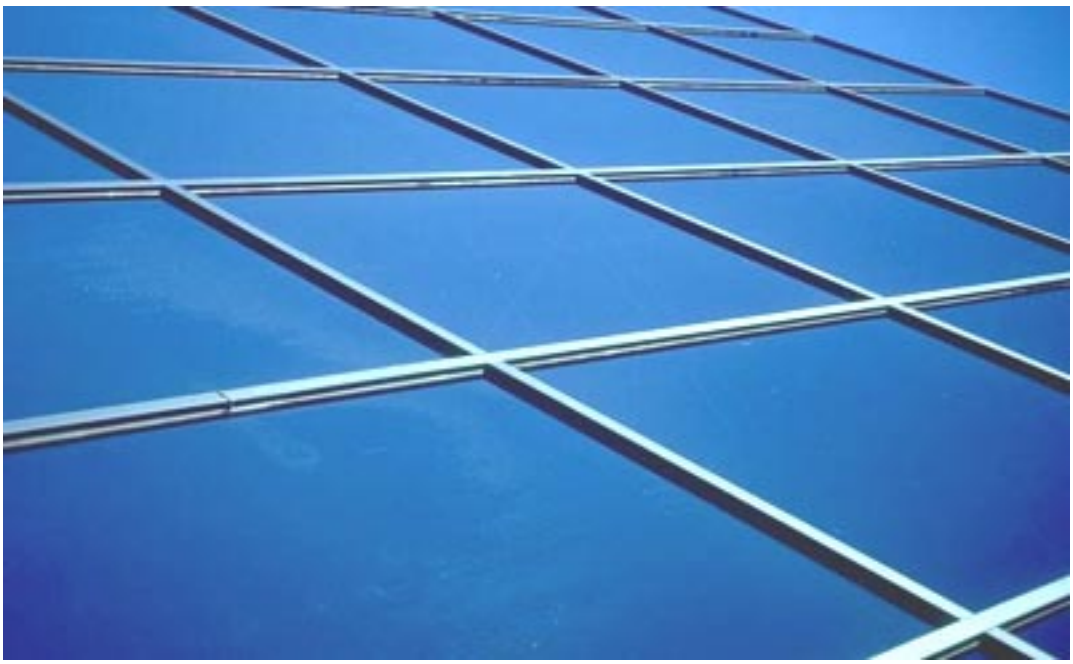
CHAPTER 1

Initialising

There's one class (`SimpilImage`) with several ways to instantiate.

1.1 Initialise from a URL

```
from simpil import SimpilImage

image = SimpilImage(source='https://jpeg.org/images/about.jpg')
```



1.2 Initialise from a file

```
from simpil import SimpilImage

image = SimpilImage(source='../images/orange_and_uke.jpg')
```

1.3 Create a new blank image

The colour is an RGB tuple. There are a few predefined constants:

- BLACK
- WHITE
- GREY
- RED
- GREEN
- BLUE
- YELLOW
- ORANGE

```
from simpil import SimpilImage, RED

image = SimpilImage(width=200,
                    height=100,
                    background_colour=RED)
```

1.4 Initialise from data

Use `bytes`, `BytesIO` (Py3) or `StringIO` (Py2). You can also use simpil.BytesIO on Py2.

```
from simpil import SimpilImage

image_data = b""
↪"\xff\xd8\xff\xe0\x00\x10JFIF\x00\x01\x01\x00\x00\x01\x00\x01\x00\x00\xff\xdb\x00C\x00\x08\x06\x06\
↪$.' ",#\x1c\x1c(7),01444\x1f'9=82<.
↪342\xff\xdb\x00C\x01\t\t\t\x0c\x0b\x0c\x18\r\r\x182!\x1c!
↪2222222222222222222222222222222222222222222222\xff\xc0\x00\x11\x08\x00d\x00\xc8\x03\x01
↪"\x00\x02\x11\x01\x03\x11\x01\xff\xc4\x00\x1f\x00\x00\x01\x05\x01\x01\x01\x01\x01\x01\x00\x00\x00\x
↪\x01\x02\x03\x00\x04\x11\x05\x12!1A\x06\x13Qa\x07"q\x142\x81\x91\xa1\x08
↪#B\xb1\xc1\x15R\xd1\xf0$3br\x82\t\n\x16\x17\x18\x19\x1a%&
↪'()*456789:CDEFGHIJSTUVWXYZcdefghijstuvwxyz\x83\x84\x85\x86\x87\x88\x89\x8a\x92\x93\x94\x95\x96\x9
↪1\x06\x12AQ\x07aq\x13"2\x81\x08\x14B\x91\xa1\xb1\xc1\t#3R\xf0\x15br\xd1\n\x16$4\xe1
↪%\xf1\x17\x18\x19\x1a&
↪'()*56789:CDEFGHIJSTUVWXYZcdefghijstuvwxyz\x82\x83\x84\x85\x86\x87\x88\x89\x8a\x92\x93\x94\x95\x96
↪\x00\xe2\xe8\xa2\x8a\xf9\x93\xf7\x10\xa2\x8a(\x00\xa2\x8a(\x00\xa2\x8a(\x00\xa2\x8a(\x00\xa2\x8a(\x
↪"""

image = SimpilImage(source=image_data)
```

CHAPTER 2

Saving

2.1 Automatically saving

By default, saving is automatic when changes are made to a SimpilImage if it has a source or destination filename.

```
from simpil import SimpilImage

image = SimpilImage(source='https://jpeg.org/images/about.jpg',
                    destination='../images/saved_image.png')
```

For an image loaded from a local file, changes can be saved to another file.

```
from simpil import SimpilImage, LEFT

image = SimpilImage(source='../images/orange_and_uke.jpg',
                    destination='../images/saved_image.jpg')

image.text(text=("SimpilImage(source='../images/orange_and_uke.jpg',\n"
                 "            destination='../images/saved_image.jpg')"),
           justify=LEFT)
```

To disable automatic saving, initialise with autosave set to False:

```
from simpil import SimpilImage

image = SimpilImage(source='https://jpeg.org/images/about.jpg',
                    destination='../images/saved_image.jpg',
                    autosave=False)
```

2.2 Saving manually

To save an image, the SimpilImage needs a destination filename.

```
from simpil import SimpilImage, RED

image = SimpilImage(width=400,
                    height=200,
                    background_colour=RED)

image.save(filename="../images/red_rect.jpg")
```

When saving back to the same file, you don't need to supply the filename

```
from simpil import SimpilImage

image = SimpilImage(source='../images/orange_and_uke.jpg')

image.save()
```

CHAPTER 3

Properties

3.1 Image data

Get the image data in the desired format. This can be used when there's no need to save the file to disk, such as for images dynamically created from webserver requests.

```
from simpil import SimpilImage, PNG

image = SimpilImage(source='https://jpeg.org/images/about.jpg')

jpeg_data = image.image_data()
png_data = image.image_data(fmt=PNG)
```

3.2 Width and height

Width and height properties are available:

```
from simpil import SimpilImage

image = SimpilImage(source='https://jpeg.org/images/about.jpg')

print(f'width:{image.width}, height:{image.height}')
```

CHAPTER 4

Scaling

Scale by using `x` and `y`. Use a value of `2` to double the size. If only one of *x* or *y* is used, the aspect ratio is retained. If you want to scale on one axis only, set the other to ‘‘1‘.

```
from simpil import SimpilImage, CONSOLAS, RED, TOP, BOTTOM

image = SimpilImage(width=200,
                    height=50,
                    background_colour=RED,
                    destination='../images/scaled_image.jpg',
                    font=CONSOLAS[12])

image.text(text=f'This image was created\n'
                f'as {image.width}x{image.height}.',
           y=TOP)

image.scale(x=2,
            y=4)

# TODO: Figure out why text can't be added after scaling an image
image.text(text=f'This image has been\n'
                f'scaled to {image.width}x{image.height}.',
           y=BOTTOM)
```

Note from this example that there is an issue with adding text after an image is scaled.

CHAPTER 5

Adding Text

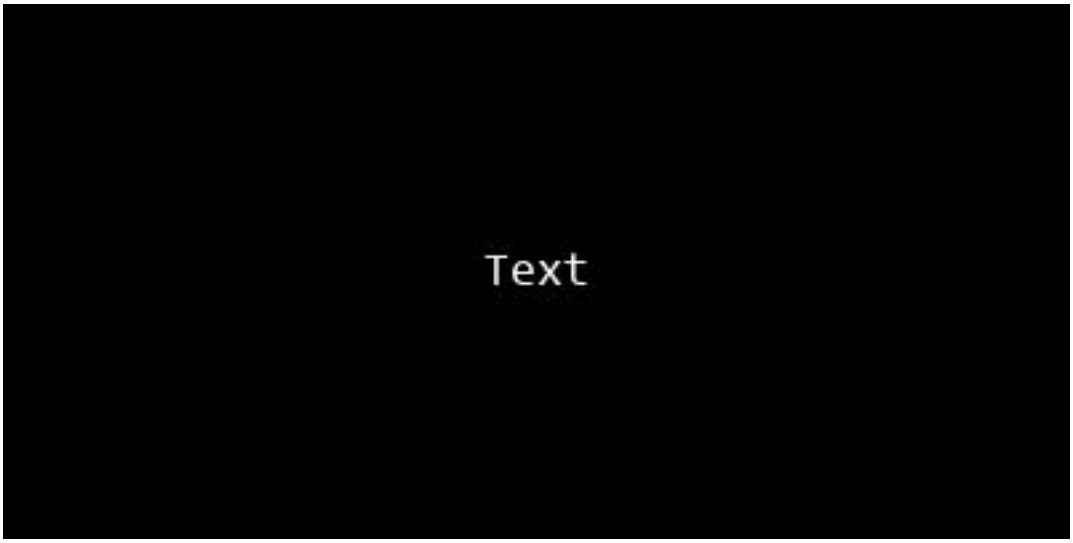
There are a few simple ways to add text to an image...

5.1 Single line

Use `SimpilImage.text()` to add text to an image.

```
from simpil import SimpilImage

image = SimpilImage(width=400,
                    height=200,
                    destination='../images/add_text.jpg')

image.text(text='Text')
```

This example shows the default font. The default `colour` is `WHITE`.



5.2 Declaring fonts

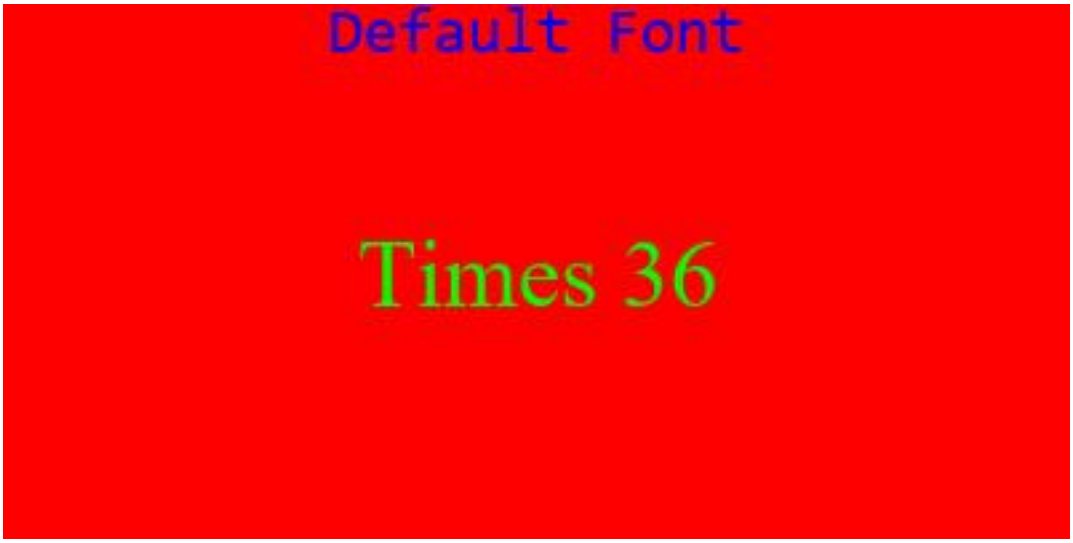
You can you use instances of `PIL.ImageFont`, but an easier way is to use the `fonts` object. This caches font instances in one place, accessed by name and size (in pixels). The name must match the filename, but you can often omit the file extension (`.ttf` and `.otf`).

```
from simpil import fonts, ImageFont

font_24 = fonts['consola'][24]
font_36 = fonts['consola'][36]
```

5.3 Using different fonts

Fonts can be supplied as a default for the SimpilImage or discretely to each text function:

```
from simpil import SimpilImage, fonts, RED, BLUE, GREEN, TOP

image = SimpilImage(width=400,
                    height=200,
                    background_colour=RED,
                    destination='../images/using_different_fonts.jpg',
                    font=fonts['consola'][24])

image.text(text='Default Font',
           y=TOP,
           colour=BLUE)

image.text(text='Times 36',
           colour=GREEN,
           font=fonts['times.ttf'][36])
```

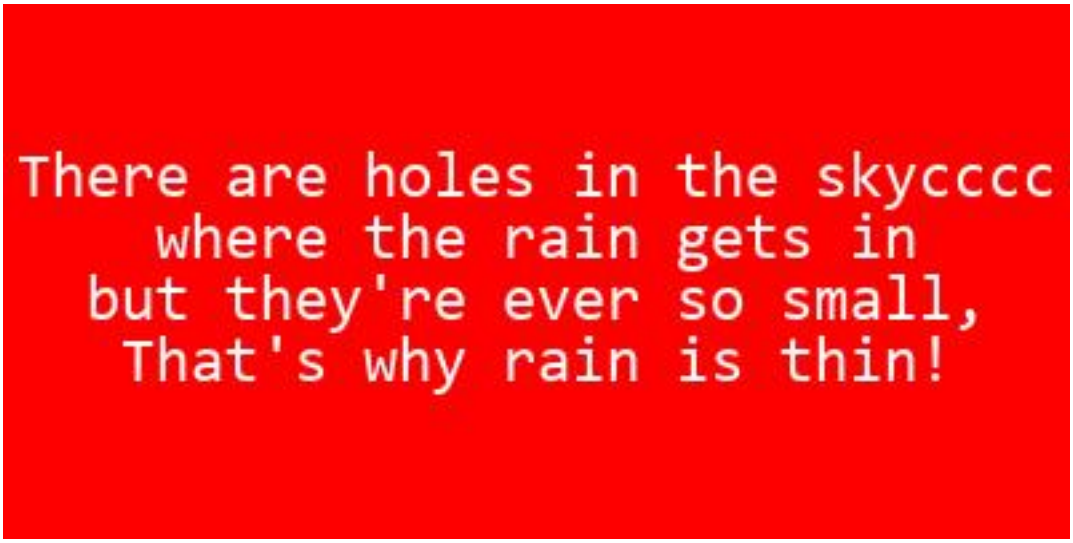


Note how different colours are used in this example.

5.4 Multiline Text

Newlines in strings will be respected. You can also use a list of strings, but if those strings should not contain newlines.

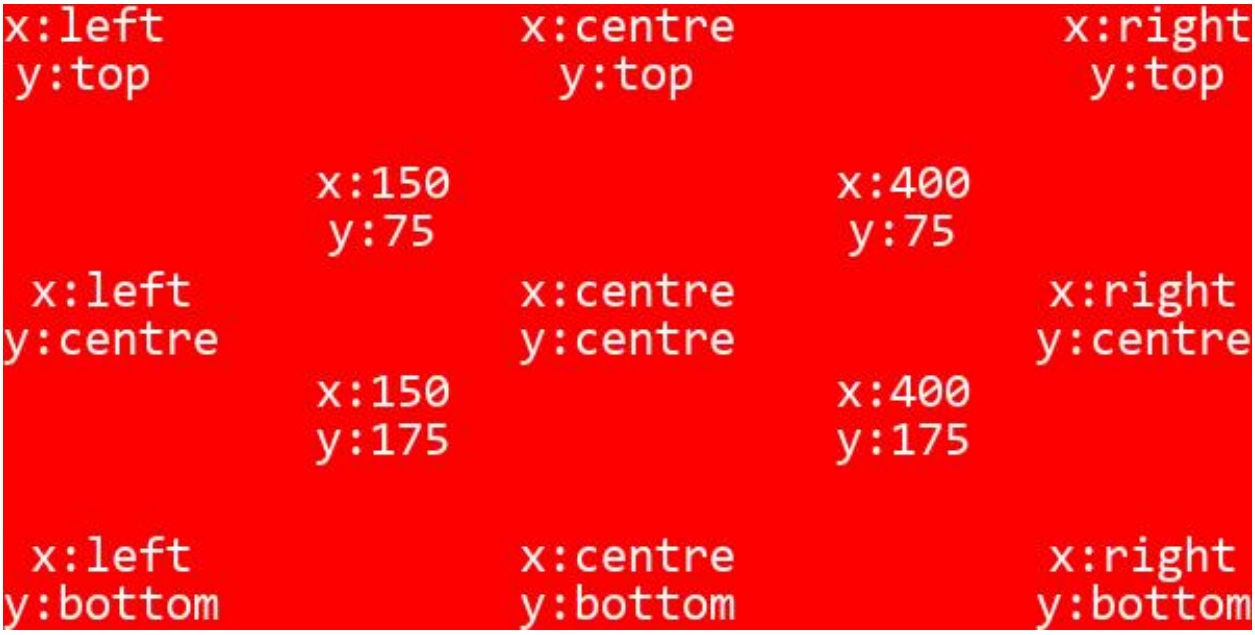
```
from simpil import SimpilImage, CONSOLAS, RED, WHITE

POEM = """There are holes in the sky
where the rain gets in
but they're ever so small,
That's why rain is thin!"""

image = SimpilImage(width=400,
                    height=200,
                    background_colour=RED,
                    destination='../images/multi_line_text.jpg')

image.text(text=POEM,
           font=CONSOLAS[24],
           colour=WHITE)
```



5.5 Positioning Text on the Image

You can specify positions `x` which can use constants `LEFT`, `CENTRE` or `RIGHT` and *y* which can use the constants `TOP`, `CENTRE` or `BOTTOM`. `x` and `y` can also take absolute values for the top left of the text.

```
from simpil import SimpilImage, CONSOLAS, RED, LEFT, CENTRE, RIGHT, TOP, BOTTOM

image = SimpilImage(width=600,
                    height=300,
                    background_colour=RED,
                    destination='../images/positioning.jpg',
                    font=CONSOLAS[24])

for x in (LEFT, CENTRE, RIGHT):
    for y in (TOP, CENTRE, BOTTOM):
        image.text(text=f'x:{x}\ny:{y}',
                   x=x,
                   y=y)

for x in (150, 400):
    for y in (75, 175):
        image.text(text=f'x:{x}\ny:{y}',
                   x=x,
                   y=y)
```



5.6 Justifying a block of text

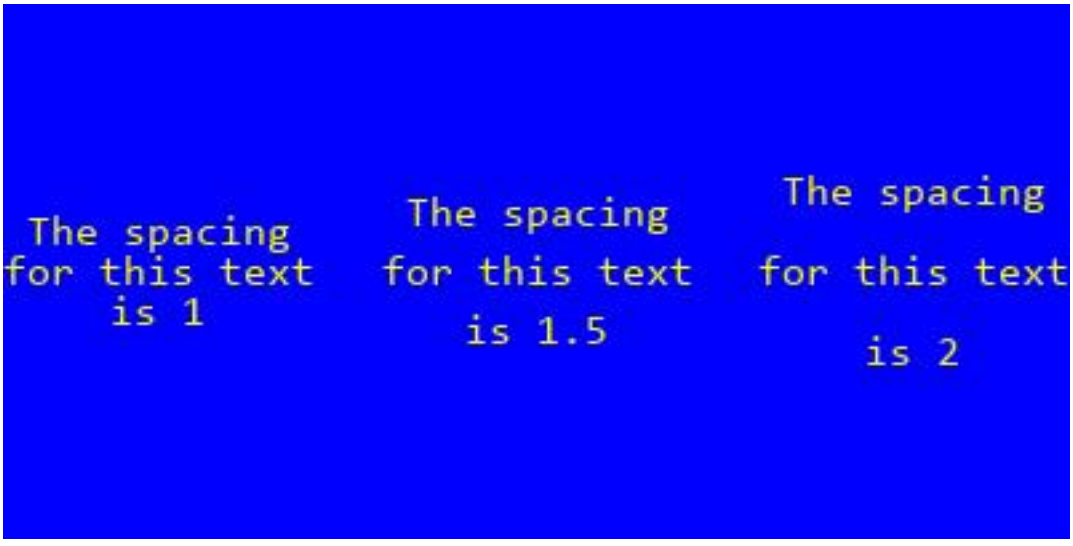
You can control how the text is justified by passing value to `justify`. This should be `LEFT`, `CENTRE` or `RIGHT`. The default is `CENTRE`.

```
from simpil import SimpilImage, fonts, CONSOLAS, YELLOW, BLUE, LEFT, RIGHT, CENTRE

image = SimpilImage(width=400,
                    height=200,
                    background_colour=BLUE,
                    destination='../images/justified_text_block.jpg',
                    font=CONSOLAS[20])

for justification in (LEFT, CENTRE, RIGHT):
    image.text(text=f'this\ntext\nis\n{justification}\njustified',
               justify=justification,
               x=justification,  # can use LEFT, CENTRE, RIGHT for x
               colour=YELLOW)
```

5.7 Linespace in a block of text

By default, there's no gap between lines, so descenders might meet ascenders. If you want a bit more spce, use a value for `spacing`. The default is `1`. Using''1.5'' adds a spacing of half a line height between each line.

```
from simpil import SimpilImage, CONSOLAS, YELLOW, BLUE, LEFT, RIGHT, CENTRE

image = SimpilImage(width=400,
                    height=200,
                    background_colour=BLUE,
                    destination='../images/line_spacing.jpg',
                    font=CONSOLAS[16])

for spacing, x in zip((1, 1.5, 2), (LEFT, CENTRE, RIGHT)):
    image.text(text=f'The spacing\nfor this text\nis {spacing}',
               spacing=spacing,
               x=x,
               colour=YELLOW)
```



5.8 Text with a Drop Shadow

Use `SimpilImage.shadowed_text()` to add text to an image with a bottom-right shadow. Useful for making text stand out again a background that is similar to the text colour

```
from simpil import SimpilImage, CONSOLAS, BLACK, WHITE, BLUE, LEFT, RIGHT

image = SimpilImage(width=400,
                    height=200,
                    background_colour=BLUE,
                    destination='../images/shadowed_text.jpg',
                    font=CONSOLAS[48])

image.shadowed_text(text=['Shadowed', 'Text'],
                    justify=LEFT,
                    x=RIGHT,
                    colour=WHITE,
                    shadow_colour=BLACK,
                    shadow_size=3)
```

Note how the colour of the shadow can be set with `shadow_colour` and the shadow depth with `shadow_size`.

5.9 Text with an Outline

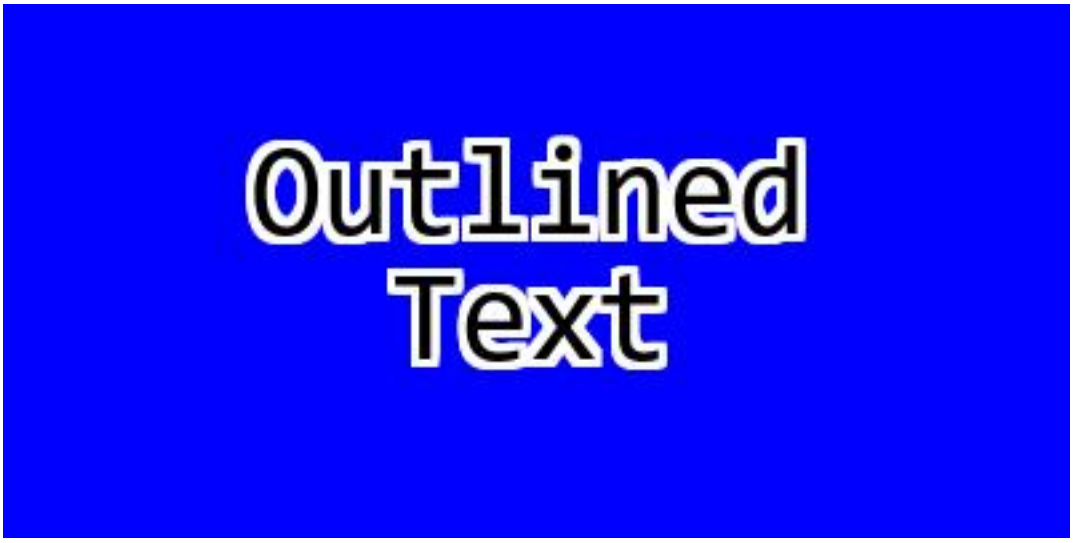
Use `SimpilImage.outlined_text()` to add plain text to an image with an outline. Useful for making text stand out against a background that is similar to the text colour

```
from simpil import SimpilImage, CONSOLAS, BLACK, WHITE, BLUE

image = SimpilImage(width=400,
                    height=200,
                    background_colour=BLUE,
                    destination='../images/outlined_text.jpg',
                    font=CONSOLAS[48])

image.outlined_text(text='Outlined\nText',
```

(continues on next page)

(continued from previous page)

```
                    colour=BLACK,
                    outline_colour=WHITE,
                    outline_size=3)
```



Note how the colour of the outline can be set with `outline_colour` and the outline size with `outline_size`.