## Contents

1 Examples ........................................... 3
  1.1 Split an image .................................. 3
  1.2 Control tile saving .................................. 3
  1.3 Processing tile images .......................... 3
  1.4 Keep it in memory ............................... 4

2 Functions ........................................... 5

3 Installation ......................................... 7

4 Command-line tools ................................ 9
  4.1 slice-image ...................................... 9
  4.2 join-tiles ....................................... 9

5 Methods ............................................. 11
  5.1 Methodology .................................... 11
  5.2 Limitations ..................................... 11

6 Development ....................................... 13

7 Troubleshooting ................................... 15
  7.1 IOError: decoder %s not available .............. 15

8 Dependencies ...................................... 17

9 Indices and tables .................................. 19

Python Module Index ................................. 21
Slice images into tiles and rejoin them. Compatible with **Python 2.6+, 3.2+**. Relies on **Pillow** for image manipulation.
CHAPTER 1

Examples

Split an image

Save tiles to the same directory as the image using the original filename as a prefix:

```python
>>> import image_slicer
>>> image_slicer.slice('cake.jpg', 4)
(<Tile #1 - cake_01_01.png>, <Tile #2 - cake_01_02.png>, <Tile #3 - cake_02_01.png>, <Tile #4 - cake_02_02.png>)
```

Control tile saving

Need more control over saving? Pass `save=False` and then use `save_tiles()`:

```python
>>> import image_slicer
>>> tiles = image_slicer.slice('cake.jpg', 4, save=False)
>>> image_slicer.save_tiles(tiles, directory='~/cake_slices', prefix='slice', ext='jpg')
(<Tile #1 - slice_01_01.jpg>, <Tile #2 - slice_01_02.jpg>, <Tile #3 - slice_02_01.jpg>, <Tile #4 - slice_02_02.jpg>)
```

Processing tile images

You can perform further processing of the images in between calling `slice()` and `save_tiles()`. The PIL Image object can be accessed with `Tile.image`. Let's overlay the tile number on each tile:

```python
import image_slicer
from PIL import ImageDraw, ImageFont
```
tiles = image_slicer.slice('cake.jpg', 4, save=False)

for tile in tiles:
    overlay = ImageDraw.Draw(tile.image)
    overlay.text((5, 5), str(tile.number), (255, 255, 255),
                  ImageFont.load_default())

image_slicer.save_tiles(tiles)

Keep it in memory

If the tile image files are not the final product and performance is a concern, consider using BytesIO to create file-like objects instead of saving each of the files to disk. Let’s use the zipfile module to create a zip archive, 'tiles.zip':

Example courtesy of 'slice-image.net'

```python
import io
import zipfile
import image_slicer

tiles = image_slicer.slice('cake.jpg', 4, save=False)

with zipfile.ZipFile('tiles.zip', 'w') as zip:
    for tile in tiles:
        with io.BytesIO() as data:
            tile.save(data)
            zip.writestr(tile.generate_filename(path=False),
                         data.getvalue())
```
CHAPTER 2

Functions

The most important functions are:

- image_slicer.main.split_image
- image_slicer.main.save_tiles
- image_slicer.main.join_tiles
To download and install the latest release:

$ pip install image_slicer

Or, for developers, to get the bleeding-edge, unreleased version:

$ pip install -e git://github.com/samdobson/image-slicer.git#egg=image-slicer

Run tests:

$ python setup.py test
Two CLI tools are provided: `slice-image` and `join-image`. These will be added to your PATH and can thus be called from any directory.

### slice-image

**Usage:**

```
$ slice-image image num_tiles
```

Unless an output directory is specified with `--dir` or `-d` tiles will be saved in the same location as the image. The original filename will be used as a prefix unless overridden with `--prefix` or `-p`.

### join-tiles

**Usage:**

```
$ join-tiles tile
```

Any of the tile images can be used as an argument - the others will be discovered automatically. Unless an output directory is specified with `--dir` or `-d` the image will be saved in the same location as the tiles. The prefix of the tiles will be used to save the image unless this is overridden with `--filename` or `-f`. 
CHAPTER 5

Methods

See all functions.

**Methodology**

Images are always split into exactly equal parts, even if this means creating more than the requested number.

**Note:** In future versions this behaviour will be overridable.

Tile filenames are appended with a 2-digit representation of the tile’s grid position (e.g image_03_02.jpg).

**Limitations**

The maximum number of tiles that can be produced is **9800**. This is an arbitrary limit which ensures that row and column numbers can be conveniently represented by two digits. Increasing it would break `get_columns_rows()` and consequently, `join_tiles`().
Fork the repository on GitHub, commit your changes and send a pull request.
Troubleshooting

If the following doesn’t help then open an issue.

**IOError: decoder %s not available**

You are missing some of the libraries required for Pillow. The Pillow documentation will be able to help you. Try starting with the platform-specific instructions.
CHAPTER 8

Dependencies

Just one: Pillow. It will be installed automatically by `pip` or `python setup.py`.
CHAPTER 9

Indices and tables

- genindex
- search
Python Module Index

i

image_slicer, 1
image_slicer (module), 1