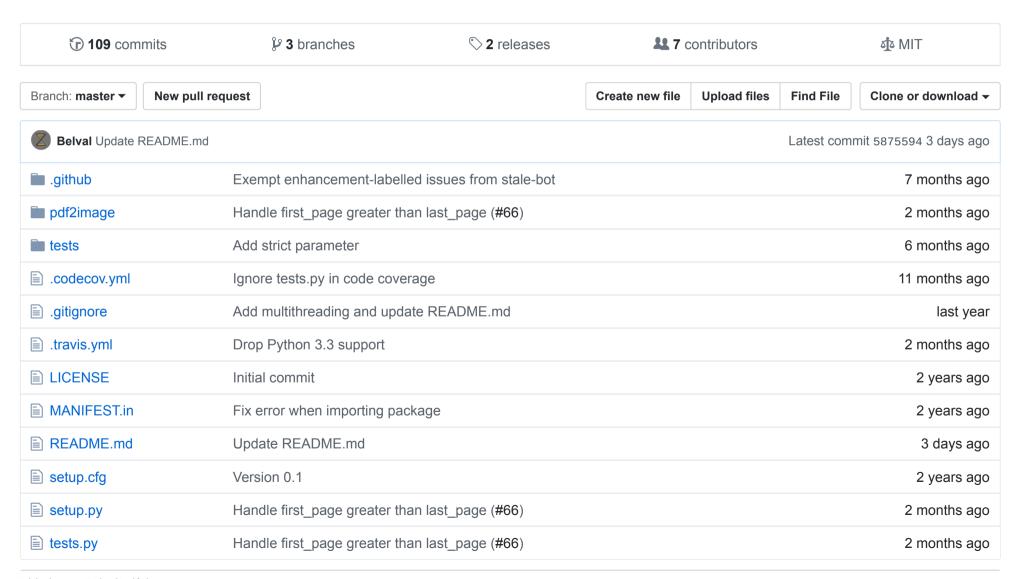
#### Belval / pdf2image

A python module that wraps the pdftoppm utility to convert PDF to PIL Image object

#pdf #pil #pil-image #convert #poppler



https://github.com/Belval/pdf2image 1/5

**■ README.md** 

# pdf2image build passing pypi package 1.5.4 Codecov 99% downloads/month 72k

A python 2.7 and 3.4+ module that wraps pdftoppm and pdftocairo to convert PDF to a PIL Image object

#### How to install

#### First you need poppler-utils

pdftoppm and pdftocairo are the piece of software that do the actual magic. It is distributed as part of a greater package called poppler.

#### Using pip

Windows users will have to install poppler for Windows, then add the bin/ folder to PATH.

Mac users will have to install poppler for Mac.

Linux users will have both tools pre-installed with Ubuntu 16.04+ and Archlinux. If it's not, run sudo apt install poppler-utils

#### Using conda

conda install -c conda-forge poppler

#### Then you can install the pip package!

pip install pdf2image

Install Pillow if you don't have it already with pip install pillow

#### How does it work?

```
from pdf2image import convert_from_path, convert_from_bytes
 from pdf2image.exceptions import (
      PDFInfoNotInstalledError,
      PDFPageCountError,
      PDFSyntaxError
Then simply do:
 images = convert_from_path('/home/kankroc/example.pdf')
OR
 images = convert_from_bytes(open('/home/kankroc/example.pdf', 'rb').read())
OR better yet
 import tempfile
 with tempfile.TemporaryDirectory() as path:
       images_from_path = convert_from_path('/home/kankroc/example.pdf', output_folder=path)
       # Do something here
images will be a list of PIL Image representing each page of the PDF document.
```

https://github.com/Belval/pdf2image 3/5

Here are the definitions:

```
convert_from_path(pdf_path, dpi=200, output_folder=None, first_page=None, last_page=None, fmt='ppm',
thread_count=1, userpw=None, use_cropbox=False, strict=False, transparent=False, poppler_path=None)
convert_from_bytes(pdf_file, dpi=200, output_folder=None, first_page=None, last_page=None, fmt='ppm',
thread_count=1, userpw=None, use_cropbox=False, strict=False, transparent=False, poppler_path=None)
```

#### What's new?

- Allow the user to specify poppler's installation path with poppler\_path
- Fixed a bug where PNGs buffer with a non-terminating I-E-N-D sequence would throw an exception
- Fixed a bug that left open file descriptors when using <code>convert\_from\_bytes()</code> (Thank you @FabianUken)
- fmt='tiff' parameter allows you to create .tiff files (You need pdftocairo for this)
- transparent parameter allows you to generate images with no background instead of the usual white one (You need pdftocairo for this)
- strict parameter allows you to catch pdftoppm syntax error with a custom type PDFSyntaxError
- use\_cropbox parameter allows you to use the crop box instead of the media box when converting ( -cropbox in pdftoppm's CLI)
- userpw parameter allows you to set a password to unlock the converted PDF ( -upw in pdftoppm's CLI)

### **Performance tips**

- Using an output folder is significantly faster if you are using an SSD. Otherwise i/o usually becomes the bottleneck.
- Using multiple threads can give you some gains but avoid more than 4 as this will cause i/o bottleneck (even on my NVMe SSD!).
- If i/o is your bottleneck, using the JPEG format can lead to significant gains.
- PNG format is pretty slow, this is because of the compression.

https://github.com/Belval/pdf2image 4/5

• If you want to know the best settings (most settings will be fine anyway) you can clone the project and run python tests.py to get timings.

## **Limitations / known issues**

• A relatively big PDF will use up all your memory and cause the process to be killed (unless you use an output folder)

https://github.com/Belval/pdf2image 5/5