

/pdf

Source:

```
~/ .claude/skills/pdf/SKILL.md
```

name: pdf description: Use this skill whenever the user wants to do anything with PDF files. This includes reading or extracting text/tables from PDFs, combining or merging multiple PDFs into one, splitting PDFs apart, rotating pages, adding watermarks, creating new PDFs, filling PDF forms, encrypting/decrypting PDFs, extracting images, and OCR on scanned PDFs to make them searchable. If the user mentions a .pdf file or asks to produce one, use this skill. license: Proprietary. LICENSE.txt has complete terms

PDF Processing Guide

Overview

This guide covers essential PDF processing operations using Python libraries and command-line tools. For advanced features, JavaScript libraries, and detailed examples, see [REFERENCE.md](#). If you need to fill out a PDF form, read [FORMS.md](#) and follow its instructions.

Quick Start

```
from pypdf import PdfReader, PdfWriter

# Read a PDF
reader = PdfReader("document.pdf")
print(f"Pages: {len(reader.pages)}")

# Extract text
text = ""
for page in reader.pages:
    text += page.extract_text()
```

Python Libraries

pypdf - Basic Operations

Merge PDFs

```
from pypdf import PdfWriter, PdfReader

writer = PdfWriter()
for pdf_file in ["doc1.pdf", "doc2.pdf", "doc3.pdf"]:
    reader = PdfReader(pdf_file)
    for page in reader.pages:
        writer.add_page(page)
```

```
with open("merged.pdf", "wb") as output:
    writer.write(output)
```

Split PDF

```
reader = PdfReader("input.pdf")
for i, page in enumerate(reader.pages):
    writer = PdfWriter()
    writer.add_page(page)
    with open(f"page_{i+1}.pdf", "wb") as output:
        writer.write(output)
```

Extract Metadata

```
reader = PdfReader("document.pdf")
meta = reader.metadata
print(f"Title: {meta.title}")
print(f"Author: {meta.author}")
print(f"Subject: {meta.subject}")
print(f"Creator: {meta.creator}")
```

Rotate Pages

```
reader = PdfReader("input.pdf")
writer = PdfWriter()

page = reader.pages[0]
page.rotate(90) # Rotate 90 degrees clockwise
writer.add_page(page)

with open("rotated.pdf", "wb") as output:
    writer.write(output)
```

pdfplumber - Text and Table Extraction

Extract Text with Layout

```
import pdfplumber

with pdfplumber.open("document.pdf") as pdf:
```

```
for page in pdf.pages:
    text = page.extract_text()
    print(text)
```

Extract Tables

```
with pdfplumber.open("document.pdf") as pdf:
    for i, page in enumerate(pdf.pages):
        tables = page.extract_tables()
        for j, table in enumerate(tables):
            print(f"Table {j+1} on page {i+1}:")
            for row in table:
                print(row)
```

Advanced Table Extraction

```
import pandas as pd

with pdfplumber.open("document.pdf") as pdf:
    all_tables = []
    for page in pdf.pages:
        tables = page.extract_tables()
        for table in tables:
            if table: # Check if table is not empty
                df = pd.DataFrame(table[1:], columns=table[0])
                all_tables.append(df)

# Combine all tables
if all_tables:
    combined_df = pd.concat(all_tables, ignore_index=True)
    combined_df.to_excel("extracted_tables.xlsx", index=False)
```

reportlab - Create PDFs

Basic PDF Creation

```
from reportlab.lib.pagesizes import letter
from reportlab.pdfgen import canvas

c = canvas.Canvas("hello.pdf", pagesize=letter)
```

```
width, height = letter

# Add text
c.drawString(100, height - 100, "Hello World!")
c.drawString(100, height - 120, "This is a PDF created with reportlab")

# Add a line
c.line(100, height - 140, 400, height - 140)

# Save
c.save()
```

Create PDF with Multiple Pages

```
from reportlab.lib.pagesizes import letter
from reportlab.platypus import SimpleDocTemplate, Paragraph, Spacer, PageBreak
from reportlab.lib.styles import getSampleStyleSheet

doc = SimpleDocTemplate("report.pdf", pagesize=letter)
styles = getSampleStyleSheet()
story = []

# Add content
title = Paragraph("Report Title", styles['Title'])
story.append(title)
story.append(Spacer(1, 12))

body = Paragraph("This is the body of the report. " * 20, styles['Normal'])
story.append(body)
story.append(PageBreak())

# Page 2
story.append(Paragraph("Page 2", styles['Heading1']))
story.append(Paragraph("Content for page 2", styles['Normal']))

# Build PDF
doc.build(story)
```

Subscripts and Superscripts

IMPORTANT: Never use Unicode subscript/superscript characters (₀₁₂₃₄₅₆₇₈₉, ⁰¹²³⁴⁵⁶⁷⁸⁹) in ReportLab PDFs. The built-in fonts do not include these glyphs, causing them to render as solid black boxes.

Instead, use ReportLab's XML markup tags in Paragraph objects:

```
from reportlab.platypus import Paragraph
from reportlab.lib.styles import getSampleStyleSheet

styles = getSampleStyleSheet()

# Subscripts: use <sub> tag
chemical = Paragraph("H<sub>2</sub>O", styles['Normal'])

# Superscripts: use <sup> tag
squared = Paragraph("x<sup>2</sup> + y<sup>2</sup>", styles['Normal'])
```

For canvas-drawn text (not Paragraph objects), manually adjust font the size and position rather than using Unicode subscripts/superscripts.

Command-Line Tools

pdftotext (poppler-utils)

```
# Extract text
pdftotext input.pdf output.txt

# Extract text preserving layout
pdftotext -layout input.pdf output.txt

# Extract specific pages
pdftotext -f 1 -l 5 input.pdf output.txt # Pages 1-5
```

qpdf

```
# Merge PDFs
qpdf --empty --pages file1.pdf file2.pdf -- merged.pdf

# Split pages
```

```
qpdf input.pdf --pages . 1-5 -- pages1-5.pdf
qpdf input.pdf --pages . 6-10 -- pages6-10.pdf

# Rotate pages
qpdf input.pdf output.pdf --rotate=+90:1 # Rotate page 1 by 90 degrees

# Remove password
qpdf --password=mypassword --decrypt encrypted.pdf decrypted.pdf
```

pdftk (if available)

```
# Merge
pdftk file1.pdf file2.pdf cat output merged.pdf

# Split
pdftk input.pdf burst

# Rotate
pdftk input.pdf rotate least output rotated.pdf
```

Common Tasks

Extract Text from Scanned PDFs

```
# Requires: pip install pytesseract pdf2image
import pytesseract
from pdf2image import convert_from_path

# Convert PDF to images
images = convert_from_path('scanned.pdf')

# OCR each page
text = ""
for i, image in enumerate(images):
    text += f"Page {i+1}:\n"
    text += pytesseract.image_to_string(image)
    text += "\n\n"
```

```
print(text)
```

Add Watermark

```
from pypdf import PdfReader, PdfWriter

# Create watermark (or load existing)
watermark = PdfReader("watermark.pdf").pages[0]

# Apply to all pages
reader = PdfReader("document.pdf")
writer = PdfWriter()

for page in reader.pages:
    page.merge_page(watermark)
    writer.add_page(page)

with open("watermarked.pdf", "wb") as output:
    writer.write(output)
```

Extract Images

```
# Using pdfimages (poppler-utils)
pdfimages -j input.pdf output_prefix

# This extracts all images as output_prefix-000.jpg, output_prefix-001.jpg, etc.
```

Password Protection

```
from pypdf import PdfReader, PdfWriter

reader = PdfReader("input.pdf")
writer = PdfWriter()

for page in reader.pages:
    writer.add_page(page)
```

```
# Add password
writer.encrypt("userpassword", "ownerpassword")

with open("encrypted.pdf", "wb") as output:
    writer.write(output)
```

Quick Reference

Task	Best Tool	Command/Code
Merge PDFs	pypdf	<code>writer.add_page(page)</code>
Split PDFs	pypdf	One page per file
Extract text	pdfplumber	<code>page.extract_text()</code>
Extract tables	pdfplumber	<code>page.extract_tables()</code>
Create PDFs	reportlab	Canvas or Platypus
Command line merge	qpdf	<code>qpdf --empty --pages ...</code>
OCR scanned PDFs	pytesseract	Convert to image first
Fill PDF forms	pdf-lib or pypdf (see FORMS.md)	See FORMS.md

Next Steps

- For advanced pypdfium2 usage, see REFERENCE.md
- For JavaScript libraries (pdf-lib), see REFERENCE.md
- If you need to fill out a PDF form, follow the instructions in FORMS.md
- For troubleshooting guides, see REFERENCE.md

Revision #5

Created 2026-02-18 08:39:52 UTC by John

Updated 2026-06-21 20:00:50 UTC by John