



# MD2PDF - Beautiful Markdown to PDF Converter


A comprehensive Python tool that converts Markdown files to impeccably beautiful PDFs with professional styling, syntax highlighting, and responsive layout. Features a dynamic style and theme system with automatic discovery and a powerful workflow for batch processing.

## ✨ Features

 **Dynamic Style System:** 5 beautiful typography templates with automatic discovery

 **Color Theme Engine:** 9 sophisticated color themes with CSS custom properties

 **Advanced Markdown Support:** Full syntax highlighting, tables, TOC, footnotes

 **Professional Layout:** Optimized for A4 paper with proper margins and page breaks

 **Batch Processing:** Default workflow for processing entire folders

 **Code Block Management:** Proper wrapping and syntax highlighting

 **Table Support:** Beautifully formatted tables with alternating row colors

 **Link Styling:** Elegant link formatting with hover effects

 **List Formatting:** Properly styled ordered and unordered lists

 **Blockquotes:** Distinguished quote styling with left border

 **Image Support:** Responsive image handling with rounded corners

📖 **Table of Contents:** Automatic TOC generation for better navigation

😊 **Emoji-Safe Rendering:** Twemoji SVG replacement for robust PDF output (local assets preferred, CDN fallback) 🖋️ **Reliable Syntax Highlighting:** Embed-

ded Pygments CSS, improved comment contrast per theme, and neutralized error tokens to avoid red boxes around ASCII art

## Quick Start

### Installation

1. **Clone the repository:** `bash git clone <repository-url> cd MD2PDF`
2. **Create virtual environment:** `bash python3 -m venv venv source venv/bin/activate #`  
On Windows: `venv\Scripts\activate`
3. **Install dependencies:** `bash pip install -r requirements.txt`

### Basic Usage

#### Single file conversion:

```
python md2pdf.py document.md
python md2pdf.py document.md --style modern --theme elegant
python md2pdf.py document.md -o custom_output.pdf
```

#### Batch processing (default workflow):

```
# Process all files in input/ folder
python md2pdf.py --style modern --theme sophisticated
python md2pdf.py --style technical --theme dark
```

## Glob patterns:

```
python md2pdf.py "*.md" --style whitepaper
python md2pdf.py "docs/*.md" --style story --theme sepia
```

## Project Structure

```
MD2PDF/
├── md2pdf.py      # Main converter script
├── style_loader.py # Dynamic style/theme loader
├── requirements.txt # Python dependencies
├── example.py     # Usage examples
├── quick_start.sh # Quick setup script
├── README.md      # This file
├── .gitignore     # Git ignore rules
├── input/         # Source markdown files (workflow)
├── output/        # Generated PDFs (workflow)
├── processed/     # Processed files (workflow)
├── styles/        # Style templates (CSS)
│   ├── technical.css # Technical documentation
│   ├── modern.css   # Modern, sophisticated
│   ├── whitepaper.css # Academic, authoritative
│   ├── story.css    # Literary, elegant
│   └── academic.css # Formal, scholarly
├── themes/        # Color themes (CSS)
│   ├── default.css  # Clean and professional
│   ├── minimal.css  # Sophisticated, timeless
│   ├── sophisticated.css # Refined light design
│   ├── elegant.css  # Sophisticated dark design
│   ├── dark.css     # Dark containers
│   ├── midnight.css # Dark containers with contrast
│   ├── oceanic.css  # Cool, calming blue tones
│   ├── forest.css   # Natural, earthy green palette
│   └── sepia.css    # Warm, vintage colors
```

# Style Templates

## Available Styles

STYLE	DESCRIPTION	BEST FOR
<b>Technical</b>	Clean, professional, code-friendly	Technical documentation, APIs, guides
<b>Modern</b>	Sophisticated, elegant, contemporary	Premium documentation, presentations
<b>White Paper</b>	Elegant, academic, authoritative	Research papers, business documents
<b>Story</b>	Literary, elegant, readable	Creative writing, narratives
<b>Academic</b>	Formal, scholarly, citation-friendly	Research papers, theses

## Style Features

- **Typography:** Each style uses carefully selected fonts (Inter, JetBrains Mono, etc.)
- **Layout:** Optimized spacing, margins, and typography hierarchy
- **Code Blocks:** Proper syntax highlighting with theme-appropriate backgrounds
- **Print Optimization:** A4 page layout with proper page breaks
- **Responsive Design:** Adapts to different content types

## Available Themes

THEME	TYPE	DESCRIPTION
<b>Default</b>	Light	Clean and professional
<b>Minimal</b>	Light	Sophisticated, elegant, timeless
<b>Sophisticated</b>	Light	Refined light design with subtle accents
<b>Elegant</b>	Dark	Sophisticated dark design
<b>Dark</b>	Light	Dark containers with light text
<b>Midnight</b>	Light	Dark containers with high contrast
<b>Oceanic</b>	Light	Cool, calming blue tones
<b>Forest</b>	Light	Natural, earthy green palette
<b>Sepia</b>	Light	Warm, vintage book-like colors

## Theme Features

- **CSS Custom Properties:** Dynamic theming with CSS variables
- **Print Optimization:** All themes designed for PDF output

- **Code Block Styling:** Proper contrast for syntax highlighting
- **Consistent Design:** Unified color palette across all elements

## Workflow System

### Default Workflow

The default workflow processes all markdown files in the `input/` folder:

1. **Place files:** Add `.md` files to the `input/` folder
2. **Run converter:** `python md2pdf.py --style <style> --theme <theme>`
3. **Generated PDFs:** Appear in `output/` folder with naming: `{filename}_{style}_{theme}.pdf`
4. **Processed files:** Original files moved to `processed/` folder

### Workflow Features

- **Automatic folder creation:** Creates `input/` , `output/` , `processed/` if missing
- **Duplicate handling:** Appends numbers to avoid filename conflicts
- **Batch processing:** Converts all files in one command
- **Progress tracking:** Shows conversion status and results

## Example Workflow

```
# 1. Add files to input folder
cp my_document.md input/

# 2. Run workflow
python md2pdf.py --style modern --theme sophisticated

# 3. Check results
ls output/      # Generated PDFs
ls processed/   # Original files
```

## Advanced Usage

### Command Line Options

```
python md2pdf.py [input_file] [options]
```

Options:

- `input_file`      Input markdown file or glob pattern (optional **for** workflow)
- `-o, --output OUTPUT`    Output PDF file path
- `-s, --style STYLE`      Style template (default: technical)
- `-t, --theme THEME`      Color theme (default: default)
- `--list-styles`          List all available style templates
- `--list-themes`          List all available color themes
- `--list-combinations`   List all style + theme combinations

## Programmatic Usage

```
from md2pdf import MD2PDFConverter

# Single file conversion
converter = MD2PDFConverter('document.md', style='modern', theme='elegant')
converter.convert()

# Custom output path
converter = MD2PDFConverter('document.md', 'custom_output.pdf', 'story', 'sepia')
converter.convert()
```

## Style and Theme Discovery

```
from style_loader import style_loader

# List available styles
styles = style_loader.list_styles()
for name, style_name, description in styles:
    print(f'{name}: {description}')

# List available themes
themes = style_loader.list_themes()
for name, theme_name, description in themes:
    print(f'{name}: {description}')

# Get all combinations
combinations = style_loader.get_available_combinations()
for style, theme in combinations:
    print(f'{style} + {theme}')
```



# Customization

## Adding New Styles

1. Create a new CSS file in `styles/` folder
2. Add descriptive comment at the top: `/* Style Name - Description */`
3. Define CSS variables for theming
4. Style will be automatically discovered

Example:

```
/* Custom Style - My Special Style */
@import url('https://fonts.googleapis.com/css2?family=...');

:root {
  --font-body: 'Your Font', sans-serif;
  --font-heading: 'Your Heading Font', serif;
  --font-code: 'Your Code Font', monospace;
  /* ... other variables */
}

/* ... rest of your CSS ... */
```

## Adding New Themes

1. Create a new CSS file in `themes/` folder
2. Add descriptive comment at the top: `/* Theme Name - Description */`
3. Define theme CSS variables
4. Theme will be automatically discovered

Example:

```
/* Custom Theme - My Special Theme */
:root {
  --theme-primary: #your-color;
  --theme-secondary: #your-color;
  --theme-text: #your-color;
  --theme-background: #your-color;
  --theme-surface: #your-color;
  --theme-code-bg: #your-color;
  --theme-code-text: #your-color;
  /* ... other theme variables */
}
```

## Requirements

- **Python:** 3.7+
- **Dependencies:** See requirements.txt
- markdown==3.5.1 - Markdown processing
- weasyprint==60.2 - PDF generation
- pygments==2.17.2 - Syntax highlighting
- pydyf==0.8.0 - PDF backend

## Aesthetic Considerations

### Typography

- High-legibility body fonts and clear monospace code fonts per style (e.g., Inter, Source Serif Pro, JetBrains Mono).

- Heading weights differ by style (lighter in Modern, stronger in Technical/Whitepaper) to signal hierarchy.

## Spacing & Padding

- A4 page margins defined via `@page` ; dark themes use a `.content` container with inner padding to provide breathing room.
- For multi-page documents, container padding is cloned across page fragments to maintain consistent spacing.

## First-line Indents

- Narrative styles (Story, Academic) indent paragraphs for readability but never the first paragraph after a heading.
- Technical, Modern, and Whitepaper avoid first-line indents by default.

## Text Alignment

- Body text alignment varies by style (left vs. justified), but list items are always ragged-right (left-aligned) to avoid rivers and awkward spacing.
- Emojis render as inline images and are aligned to baseline to prevent line breaks.

## Code & Syntax Highlighting

- Class-based Pygments with embedded CSS ensures consistent rendering across themes.
- Comment tokens have tuned contrast per theme; code blocks and inline code include background fills.
- Error token styling is neutralized to avoid red boxes around ASCII art/trees.

# Examples

## Technical Documentation

```
python md2pdf.py api_docs.md --style technical --theme dark
```

## Academic Paper

```
python md2pdf.py research_paper.md --style academic --theme sophisticated
```

## Creative Writing

```
python md2pdf.py short_story.md --style story --theme sepia
```

## Business Document

```
python md2pdf.py whitepaper.md --style whitepaper --theme minimal
```

## Batch Processing

```
# Process all documentation  
python md2pdf.py --style modern --theme elegant
```

# Troubleshooting

## Common Issues

**WeasyPrint errors:** Ensure all system dependencies are installed

```
# macOS
brew install cairo pango gdk-pixbuf libffi

# Ubuntu/Debian
sudo apt-get install build-essential python3-dev python3-pip python3-
setuptools python3-wheel python3-cffi libcairo2 libpango-1.0-0 libpangocairo
-1.0-0 libgdk-pixbuf2.0-0 libffi-dev shared-mime-info
```

**Font issues:** Ensure Google Fonts are accessible or use system fonts **Code block wrapping:** All styles include proper text wrapping for long lines

## Debug Mode

For troubleshooting, you can inspect the generated HTML:

```
converter = MD2PDFConverter('document.md')
html_content = converter._process_markdown(content)
print(html_content) # Inspect the HTML output
```

## License

[Add your license information here]

## Contributing

1. Fork the repository
2. Create a feature branch
3. Add your styles/themes to the respective folders
4. Test your changes
5. Submit a pull request

## Support

For issues, questions, or contributions, please open an issue on GitHub.

**MD2PDF** - Transform your markdown into beautiful, professional PDFs with style and elegance. ✨