README Code of conduct MIT license Security

MarkItDown

> [!TIP]
> MarkItDown now offers an MCP (Model Context Protocol) server for integration with LLM applications like Claude Desktop. See markitdown-mcp for more information.

> [!IMPORTANT]
> Breaking changes between 0.0.1 to 0.1.0:
>
> - Dependencies are now organized into optional feature-groups (further details below). Use `pip install 'markitdown[all]'` to have backward-compatible behavior.
> - convert_stream() now requires a binary file-like object (e.g., a file opened in binary mode, or an io.BytesIO object). This is a breaking change from the previous version, where it previously also accepted text file-like objects, like io.StringIO.
> - The DocumentConverter class interface has changed to read from file-like streams rather than file paths. *No temporary files are created anymore*. If you are the maintainer of a plugin, or custom DocumentConverter, you likely need to update your code. Otherwise, if only using the MarkItDown class or CLI (as in these examples), you should not need to change anything.

MarkItDown is a lightweight Python utility for converting various files to Markdown for use with LLMs and related text analysis pipelines. To this end, it is most comparable to textract, but with a focus on preserving important document structure and content as Markdown (including: headings, lists, tables, links, etc.) While the output is often reasonably presentable and human-friendly, it is meant to be consumed by text analysis tools -- and may not be the best option for high-fidelity document conversions for human consumption.

MarkItDown currently supports the conversion from:

- PDF
- PowerPoint
- Word
- Excel
- Images (EXIF metadata and OCR)
- Audio (EXIF metadata and speech transcription)
- HTML
- Text-based formats (CSV, JSON, XML)
- ZIP files (iterates over contents)
- Youtube URLs
- EPubs
- ... and more!

Why Markdown?

Markdown is extremely close to plain text, with minimal markup or formatting, but still provides a way to represent important document structure. Mainstream LLMs, such as OpenAI's GPT-4o, natively "speak" Markdown, and often incorporate Markdown into their responses unprompted. This suggests that they have been trained on vast amounts of Markdown-formatted text, and understand it well. As a side benefit, Markdown conventions are also highly token-efficient.

Prerequisites

MarkItDown requires Python 3.10 or higher. It is recommended to use a virtual environment to avoid dependency conflicts.

With the standard Python installation, you can create and activate a virtual environment using the following commands:

```
python -m venv .venv
source .venv/bin/activate
```

If using `uv`, you can create a virtual environment with:

```
uv venv --python=3.12 .venv
source .venv/bin/activate
# NOTE: Be sure to use `uv pip install` rather than `pip install`
```

If you are using Anaconda, you can create a virtual environment with:

```
conda create -n markitdown python=3.12
conda activate markitdown
```

Installation

To install MarkItDown, use `pip`: `pip install 'markitdown[all]'`. Alternatively, you can install it from the source:

```
git clone git@github.com:microsoft/markitdown.gi
cd markitdown
pip install -e 'packages/markitdown[all]'
```

Usage

Command-Line

```
markitdown path-to-file.pdf > document.md
```

Or use `-o` to specify the output file:

```
markitdown path-to-file.pdf -o document.md
```

You can also pipe content:

```
cat path-to-file.pdf | markitdown
```

Optional Dependencies

MarkItDown has optional dependencies for activating various file formats. Earlier in this document, we installed all optional dependencies with the `[all]` option. However, you can also install them individually for more control. For example:

```
pip install 'markitdown[pdf, docx, pptx]'
```

will install only the dependencies for PDF, DOCX, and PPTX files.

At the moment, the following optional dependencies are available:

- `[all]` Installs all optional dependencies
- `[pptx]` Installs dependencies for PowerPoint files
- `[docx]` Installs dependencies for Word files
- `[xlsx]` Installs dependencies for Excel files
- `[xls]` Installs dependencies for older Excel files
- `[pdf]` Installs dependencies for PDF files
- `[outlook]` Installs dependencies for Outlook messages
- `[az-doc-intel]` Installs dependencies for Azure Document Intelligence
- `[audio-transcription]` Installs dependencies for audio transcription of wav and mp3 files
- `[youtube-transcription]` Installs dependencies for fetching YouTube video transcription

Plugins

MarkItDown also supports 3rd-party plugins. Plugins are disabled by default. To list installed plugins:

```
markitdown --list-plugins
```

To enable plugins use:

```
markitdown --use-plugins path-to-file.pdf
```

To find available plugins, search GitHub for the hashtag `#markitdown-plugin`. To develop a plugin, see `packages/markitdown-sample-plugin`.

markitdown-ocr Plugin

The `markitdown-ocr` plugin adds OCR support to PDF, DOCX, PPTX, and XLSX converters, extracting text from embedded images using LLM Vision — the same `llm_client` / `llm_model` pattern that MarkItDown already uses for image descriptions. No new ML libraries or binary dependencies required.

Installation:

```
pip install markitdown-ocr
pip install openai  # or any OpenAI-compatible c
```

Usage:

Pass the same `llm_client` and `llm_model` you would use for image descriptions:

```
from markitdown import MarkItDown
from openai import OpenAI

md = MarkItDown(
    enable_plugins=True,
    llm_client=OpenAI(),
    llm_model="gpt-4o",
)
result = md.convert("document_with_images.pdf")
print(result.text_content)
```

If no `llm_client` is provided the plugin still loads, but OCR is silently skipped and the standard built-in converter is used instead.

See `packages/markitdown-ocr/README.md` for detailed documentation.

Azure Document Intelligence

To use Microsoft Document Intelligence for conversion:

```
markitdown path-to-file.pdf -o document.md -d -e
```

More information about how to set up an Azure Document Intelligence Resource can be found here

Python API

Basic usage in Python:

```
from markitdown import MarkItDown

md = MarkItDown(enable_plugins=False) # Set to T
result = md.convert("test.xlsx")
print(result.text_content)
```

Document Intelligence conversion in Python:

```
from markitdown import MarkItDown

md = MarkItDown(docintel_endpoint="<document_int>
result = md.convert("test.pdf")
print(result.text_content)
```

To use Large Language Models for image descriptions (currently only for pptx and image files), provide `llm_client` and `llm_model`:

```
from markitdown import MarkItDown
from openai import OpenAI

client = OpenAI()
md = MarkItDown(llm_client=client, llm_model="gp
result = md.convert("example.jpg")
print(result.text_content)
```

Docker

```
docker build -t markitdown:latest .
docker run -rm -i markitdown:latest < ~/your-fi
```

Contributing

This project welcomes contributions and suggestions. Most contributions require you to agree to a Contributor License Agreement (CLA) declaring that you have the right to, and actually do, grant us the rights to use your contribution. For details, visit https://cla.opensource.microsoft.com.

When you submit a pull request, a CLA bot will automatically determine whether you need to provide a CLA and decorate the PR appropriately (e.g., status check, comment). Simply follow the instructions provided by the bot. You will only need to do this once across all repos using our CLA.

This project has adopted the Microsoft Open Source Code of Conduct. For more information see the Code of Conduct FAQ or contact opencode@microsoft.com with any additional questions or comments.

How to Contribute

You can help by looking at issues or helping review PRs. Any issue or PR is welcome, but we have also marked some as 'open for contribution' and 'open for reviewing' to help

About

Python tool for converting files and office documents to Markdown.

markdown pdf openai microsoft-office autogen langchain autogen-extension

Readme

MIT license

Code of conduct

Security policy

Activity

Custom properties

104k stars

373 watching

6.5k forks

Report repository

Releases 18

v0.1.5 Latest on Feb 20

+ 17 releases

Packages

No packages published

Used by 2.4k

+ 2,422

Contributors 74

+ 60 contributors

Languages

Python 99.7%

Dockerfile 0.3%

facilitate community contributions. These are of course just suggestions and you are welcome to contribute in any way you like.

	All	Especially Needs Help from Community
Issues	All Issues	Issues open for contribution
PRs	All PRs	PRs open for reviewing

Running Tests and Checks

- Navigate to the Mark3tDown package:

  ```
  cd packages/markitdown
  ```

- Install `hatch` in your environment and run tests:

  ```
  pip install hatch  # Other ways of installing: https://hatch.pypa.io/dev/install/
  hatch shell
  hatch test
  ```

  (Alternative) Use the Devcontainer which has all the dependencies installed:

  ```
  # Reopen the project in Devcontainer and run:
  hatch test
  ```

- Run pre-commit checks before submitting a PR: `pre-commit run --all-files`

Contributing 3rd-party Plugins

You can also contribute by creating and sharing 3rd party plugins. See `packages/markitdown-sample-plugin` for more details.

Trademarks

This project may contain trademarks or logos for projects, products, or services. Authorized use of Microsoft trademarks or logos is subject to and must follow Microsoft's Trademark & Brand Guidelines. Use of Microsoft trademarks or logos in modified versions of this project must not cause confusion or imply Microsoft sponsorship. Any use of third-party trademarks or logos are subject to those third-party's policies.