That's an excellent project! Bi-directional conversion between Microsoft Word/Google Docs (.docx) and Markdown (.md) is a common need for modern documentation workflows.

While a single, perfect Python library for a **fully faithful, bi-directional** conversion of all formatting (especially complex tables and embedded objects) doesn't exist due to the complexity of the .docx format, the best starting points are specialized libraries.

Here are the top Python libraries for this task, with an emphasis on **MarkItDown** for DOCX-to-MD and **md2docx-python** for bi-directional conversion.

## 🐍 Recommended Python Libraries

| Direction | Primary Library | Key Features | URL |  
| :--- | :--- | :--- | :--- |  
| **DOCX → MD** | **MarkItDown** (by Microsoft) | Focuses on robust extraction for AI/LLM use. Handles headings, lists, tables, and images well. | GitHub: microsoft/markitdown |  
| **Bi-Directional** | **md2docx-python** | Simple, community-driven, bi-directional conversion. Good for basic Markdown elements (headings, lists, bold/italic). | PyPI: md2docx-python |  
| **Alternative** | **Pandoc** (CLI wrapper) | **Gold standard** for document conversion. Requires the Pandoc *software* to be installed, but offers the most reliable, high-fidelity conversion. | Pandoc Documentation |

## 🛠️ Setup and Code Snippets

### Option 1: Bi-Directional using md2docx-python

This is the most straightforward option for a pure Python bi-directional tool, especially if your documents use standard Markdown features.

1. Install the Library:  
   bash  
    pip install md2docx-python

**A. Convert DOCX to Markdown**

```python  
from md2docx\_python.src.docx2md\_python import word\_to\_markdown

def docx\_to\_md(word\_file: str, markdown\_file: str):  
 """Converts a Word (.docx) file to a Markdown (.md) file."""  
 try:  
 word\_to\_markdown(word\_file, markdown\_file)  
 print(f"✅ Successfully converted '{word\_file}' to '{markdown\_file}'")  
 except Exception as e:  
 print(f"❌ Error during DOCX to MD conversion: {e}")

# Example Usage

docx\_to\_md("input\_document.docx", "output\_document.md")  
```

**B. Convert Markdown to DOCX**

```python  
from md2docx\_python.src.md2docx\_python import markdown\_to\_word

def md\_to\_docx(markdown\_file: str, word\_file: str):  
 """Converts a Markdown (.md) file to a Word (.docx) file."""  
 try:  
 # Note: md2docx\_python creates the .docx file and writes the content  
 markdown\_to\_word(markdown\_file, word\_file)  
 print(f"✅ Successfully converted '{markdown\_file}' to '{word\_file}'")  
 except Exception as e:  
 print(f"❌ Error during MD to DOCX conversion: {e}")

# Example Usage

md\_to\_docx("input\_document.md", "output\_document.docx")  
```

### Option 2: High-Fidelity DOCX to Markdown using MarkItDown

If your primary challenge is getting *clean, structured* Markdown from complex Word documents, the Microsoft-backed MarkItDown is excellent. It leverages other conversion tools like mammoth internally.

1. Install the Library (including all dependencies for full functionality):  
   bash  
    # Use the [all] option for converting DOCX, PDF, Excel, etc.  
    pip install 'markitdown[all]'
2. Note: Requires Python 3.10+ for the latest version.

**Convert DOCX to Markdown**

```python  
from markitdown import MarkItDown  
import os

def markitdown\_docx\_to\_md(docx\_path: str, output\_dir: str):  
 """Converts a DOCX file to Markdown using MarkItDown."""

# Example Usage

# Create an output directory if it doesn't exist

os.makedirs("output\_md\_files", exist\_ok=True)  
markitdown\_docx\_to\_md("input\_document.docx", "output\_md\_files")  
```

### Key Considerations for Your Tool

1. Image Handling: Converting DOCX to MD often extracts embedded images and places them in a sub-folder, linking to them in the Markdown file (e.g., ![Image Description](images/image1.png)). Check the documentation for how each library manages this, especially MarkItDown which is good at it.
2. Tables: Markdown tables are simpler than Word tables. Complex Word tables (merged cells, nested tables) are often converted into HTML-style tables in the MD output, or sometimes poorly represented.
3. Styles: Word uses named styles (e.g., "Heading 1," "List Paragraph"). When converting to MD, these are mapped to Markdown syntax (#, \*). When converting back to DOCX, the md2docx-python approach typically relies on a default DOCX template to apply styles correctly.

This video gives a visual walkthrough of using MarkItDown for various document formats, which might be helpful for understanding its output quality: Convert Office Documents to Markdown with Python: A Simple Guide.

## 🚀 Enhanced CLI Tool: docx-md.py

We've improved the original docx-md.py script with a professional Click-based CLI interface for easier usage.

### Key Improvements:

1. Click CLI Interface: Added proper command-line argument parsing with --in and --out options
2. Automatic File Type Detection: The script now detects input file types (.md or .docx) and calls the appropriate conversion function
3. Auto-generated Output Paths: If --out is not specified, the script automatically generates the output filename with the correct extension
4. Enhanced Error Handling: Better error messages and proper exit codes
5. Directory Creation: Automatically creates output directories if they don't exist
6. Input Validation: Validates file existence and supported file types

### New Features:

* --in (required): Input file path (supports .md or .docx files)
* --out (optional): Output file path - auto-generates if not provided
* Help: Use --help or -h to see usage instructions

### Usage Examples:

```bash

# Convert docx to markdown (auto-generate output name)

python docx-md.py --in document.docx

# Convert markdown to docx with specific output

python docx-md.py --in document.md --out converted.docx

# Convert with explicit paths

python docx-md.py --in input.docx --out output.md

# Show help

python docx-md.py --help  
```

### Installation:

1. Install dependencies:  
    bash  
    pip install -r requirements.txt
2. The script automatically detects file types based on extensions and performs the appropriate conversion, making it much more user-friendly and robust than manual function calls.