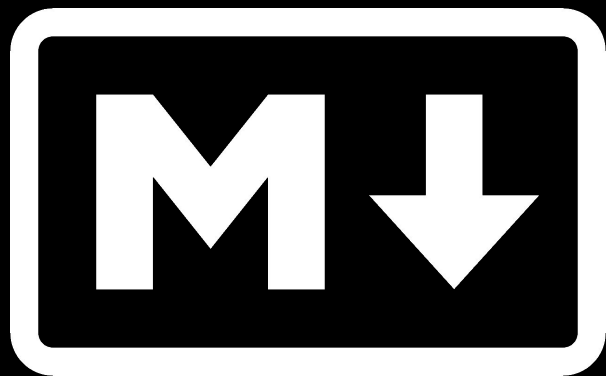


Markdown Converter

by Mehmet Fatih SARAÇ



Valid Email

```
^[a-zA-Z0-9.!#$%&'*+\\/=/?^_`{|}~-]+@[a-zA-Z0-9](?:[a-zA-Z0-9-]{0,61}[a-zA-Z0-9])?(?:\\.[a-zA-Z0-9](?:[a-zA-Z0-9-]{0,61}[a-zA-Z0-9])?)*$
```

Libraries

```
import argparse
```

```
import os
```

```
import sys
```

```
import re
```

Functions

```
def main()
```

```
def get_content(file)
```

```
def convert_to_raw_html(content)
```

```
def export_as_html(filename, content)
```

```
def export_as_txt(filename, content)
```

```
def format_content(content)
```

```
def get_content(file), explained
```

```
try:
```

```
    with open(file) as f:
```

```
        return f.read()
```

```
except FileNotFoundError:
```

```
    return "Error 1: FileNotFoundError"
```

```
def format_content(content), explained
```

```
start = f'{"=" * 25} START {"=" * 25}'
```

```
end = f'{"=" * 26} END {"=" * 26}'
```

```
return f"{start}\n\n{content}\n{end}"
```

```
def export_as_txt(filename, content), explained
```

```
with open(f"{filename}.txt", "w") as file:  
    file.write(content)
```



```
def export_as_html(filename, content), explained
```

```
# upper_part -> HTML Boilerplate and some style
```

```
# lower_part -> Closes the body and html tags
```

```
with open(f"{filename}.html", "w") as file:
```

```
    file.write(upper_part.format(filename.capitalize()))
```

```
    file.write(content)
```

```
    file.write(lower_part)
```

```
def convert_to_raw_html(content), explained
```

```
# Convert to paragraph
```

```
content = re.sub(  
    r"^([^\n]*\n)*$", r"<p>\1</p>",  
    content,  
    flags=re.MULTILINE  
)
```

Pattern Explanation: `r"^([^\#!*_\-\+\n]*.*)$"`

`^`: Asserts the position at the start of a line

`(`: Start of a capturing group

`[^\#!*_\-\+\n]`: A negated character class. It matches any character that is not #, !, *, _, -, +, or a newline (`\n`)

`.*`: Matches any character (except for a newline) zero or more times

`)`: End of the capturing group

`$`: Asserts the position at the end of a line

foo



<p>foo</p>

```
def convert_to_raw_html(content), continued
```

```
# Convert to headings
```

```
content = re.sub(  
    r"^(#{1,6}) +(.*)$",  
    lambda m: f"<h{len(m.group(1))}>{m.group(2)}</h{len(m.group(1))}>",  
    content,  
    flags=re.MULTILINE  
)
```

Bar



<h1>Bar</h1>

```
def convert_to_raw_html(content), continued
```

```
# Convert to bold
```

```
content = re.sub(  
    r"\*{2}([^\*]+)\*{2}", r"<strong>\1</strong>", content, flags=re.MULTILINE  
)
```

```
# Convert to italic
```

```
content = re.sub(r"_([^\_]+)_(?: |\n)", r"<em>\1</em> ", content, flags=re.MULTILINE)
```

```
# Convert to strikethrough
```

```
content = re.sub(r"~{2}([^\~]+)~{2}", r"<s>\1</s>", content, flags=re.MULTILINE)
```

```
def convert_to_raw_html(content), continued
```

```
# Convert to link
```

```
content = re.sub(  
    r"\([([^\]]+)\)\((.+?(?=\) ))\)",  
    r'<a href="\2">\1</a>',  
    content,  
    flags=re.MULTILINE,  
)
```



```
def convert_to_raw_html(content), continued
```

```
# Convert to image
```

```
content = re.sub(  
    r"!\\([\\^\\]*)\\]\\((\\.+?(?=\\)\\n))\\)",  
    r'',  
    content,  
    flags=re.MULTILINE,  
)
```

```
def convert_to_raw_html(content), continued
```

```
# Convert to unordered lists
```

```
content = re.sub(r"^(?:-|\*|\+)(.*)$", r"<li>\1</li>", content, flags=re.MULTILINE)
```

```
# Convert to code
```

```
content = re.sub(r"```([^\`]+)```", r"<code>\1</code>", content, flags=re.MULTILINE)
```

def main(), explained

```
parser = argparse.ArgumentParser(description="Convert a markdown file to HTML")
parser.add_argument(
    "-f", "--filename", help="filename to be converted", metavar="", type=str
)
parser.add_argument(
    "-o",
    "--output",
    default="str",
    help="output type (html, str or txt)",
    metavar="",
    type=str,
)
args = parser.parse_args()
```

```
def main(), continued
```

```
file = args.filename
```

```
filename, file_extension = os.path.splitext(file)
```

```
if file_extension != ".md":
```

```
    sys.exit("File extension must be .md.")
```

```
content = helpers.get_content(file)
```

```
if content == "Error 1: FileNotFoundError":
```

```
    sys.exit(f"The {file} file could not be found.")
```

```
def main(), continued
```

```
output_type: str = args.output.strip().lower()
```

```
match output_type:
```

```
    case "html" | "h":
```

```
        helpers.export_as_html(filename, converted_content)
```

```
    case "txt" | "t":
```

```
        helpers.export_as_txt(filename, converted_content)
```

```
    case "str" | "s":
```

```
        print(helpers.format_content(converted_content))
```

```
    case _:
```

```
        print(f"Cannot convert to {output_type}")
```



Thank you
for listening!
¡Gracias por
escuchar!
Merci de votre
attention!
Dziękuję za
wysłuchanie!