mdutils Documentation

Release 1.6.0

Dídac Coll

Aug 11, 2023

Contents:

1 Overview 1

2 Features 3

3 Installation 5

4 Markdown File Example 7

5 Python Code of Markdown File Example 11

6 mdutils 21

7 Indices and tables 37

Python Module Index 39

Index 41

CHAPTER 1

Overview

This Python package contains a set of basic tools that can help to create a markdown file while running a Python code. Thus, if you are executing a Python code and you save the result in a text file, Why not format it? So using files such as Markdown can give a great look to those results. In this way, mdutils will make things easy for creating Markdown files.

CHAPTER 2

Features

There are some different features available on that version of mdutils:

2.1 Writing and Reading Files

- Write and Read Markdown files.
- Append data to the end of a Markdown file.
- Use markers to place text.

2.2 Markdown

- Implemented method to give format to the text: bold, italics, change color...
- Add headers of levels 1 til 6 (atx style) or 1 and 2 (setext style).
- Create tables.
- Create a table of contents.
- Add Links.
- Add Markdown Images.
- Add Html Images.

Note: Some available features will depen on which CSS you are using. For example, GitHub do not allows to give color to text.

CHAPTER 3

Installation

Use pip to install mdutils:

```
$ pip install mdutils
```

CHAPTER 4

Markdown File Example

4.1 Contents

- *Overview*
- *This is what you can do*
  - *Create Markdown files*
  - *Create Headers*
  - *Table of Contents*
  - *Paragraph and Text Format*
  - *Create a Table*

4.2 Overview

This is an example of markdown file created using mdutils python package. In this example you are going to see how to create a markdown file using this library. Moreover, you're finding the available features which makes easy the creation of this type of files while you are running Python code.

Note: Some features available on this library have no effect with the GitHub Markdown CSS. Some of them are: coloring text, centering text...

4.3 This is what you can do

4.3.1 Create Markdown files

```
import Mdutils


mdFile = MdUtils(file_name='Example_Markdown',title='Markdown File Example')
mdFile.create_md_file()
```

Note: `create_md_file()` is the last command that has to be called.

4.3.2 Create Headers

Using `new_header` method you can create headers of different levels depending on the style. There are two available styles: 'atx' and 'setext'. The first one has til 6 different header levels. Atx's levels 1 and 2 are automatically added to the table of contents unless the parameter `add_table_of_contents` is set to 'n'. The 'setext' style only has two levelsof headers.

```
mdFile.new_header(level=1, title='Atx Header 1')
mdFile.new_header(level=2, title='Atx Header 2')
mdFile.new_header(level=3, title='Atx Header 3')
mdFile.new_header(level=4, title='Atx Header 4')
mdFile.new_header(level=5, title='Atx Header 5')
mdFile.new_header(level=6, title='Atx Header 6')
```

4.4 Atx Header 1

4.4.1 Atx Header 2

Atx Header 3

Atx Header 4

Atx Header 5

Atx Header 6

```
mdFile.new_header(level=1, title='Setext Header 1', style='setext')
mdFile.new_header(level=2, title='Setext Header 2', style='setext')
```

4.5 Setext Header 1

4.5.1 Setext Header 2

4.5.2 Table of Contents

If you have defined some headers of level 1 and 2, you can create a table of contents invoking the following command (Normally, the method will be called at the end of the code before calling `create_md_file()`)

```
mdFile.new_table_of_contents(table_title='Contents', depth=2)
```

4.5.3 Paragraph and Text Format

mdutils allows you to create paragraph, line breaks or simply write text:

New Paragraph Method

```
mdFile.new_paragraph("Using ``new_paragraph`` method you can very easily add a new␣
↪paragraph"
                     " This example of paragraph has been added using this method.␣
↪Moreover,"
                     "``new_paragraph`` method make your live easy because it can␣
↪give format"
                     " to the text. Lets see an example:")
```

Using `new_paragraph` method you can very easily add a new paragraph on your markdown file. This example of paragraph has been added using this method. Moreover, `new_paragraph` method make your live easy because it can give format to the text. Lets see an example:

```
mdFile.new_paragraph("This is an example of text in which has been added color, bold␣
↪and italics text.", bold_italics_code='bi', color='purple')
```

New Line Method

`mdutils` has a method which can create new line breaks. Lets see it.

```
mdFile.new_line("This is an example of line break which has been created with ``new_
↪line`` method.")
```

This is an example of line break which has been created with `new_line` method.

As `new_paragraph`, `new_line` allows users to give format to text using `bold_italics_code` and `color` parameters:

```
mdFile.new_line("This is an inline code which contains bold and italics text and it␣
↪is centered", bold_italics_code='cib', align='center')
```

Write Method

`write` method writes text in a markdown file without jump lines `'\n'` and as `new_paragraph` and `new_line`, you can give format to text using the arguments `bold_italics_code`, `color` and `align`:

```
mdFile.write("The following text has been written with ``write`` method. You can use␣
↪markdown directives to write:"
             "**bold**, _italics_, ``inline_code``... or ")
mdFile.write("use the following available parameters:  \n")
```

The following text has been written with `write` method. You can use markdown directives to write: **bold**, *italics*, `inline_code`... or use the following available parameters:

```
mdFile.write('  \n')
mdFile.write('bold_italics_code', bold_italics_code='bic')
mdFile.write('  \n')
mdFile.write('Text color', color='green')
mdFile.write('  \n')
mdFile.write('Align Text to center', align='center')
```

4.5.4 Create a Table

The library implements a method called `new_table` that can create tables using a list of strings. This method only needs: the number of rows and columns that your table must have. Optionally you can align the content of the table using the parameter `text_align`

```
list_of_strings = ["Items", "Descriptions", "Data"]
for x in range(5):
    list_of_strings.extend(["Item " + str(x), "Description Item " + str(x), str(x)])
mdFile.new_line()
mdFile.new_table(columns=3, rows=6, text=list_of_strings, text_align='center')
```

Items	Descriptions	Data
Item 0	Description Item 0	0
Item 1	Description Item 1	1
Item 2	Description Item 2	2
Item 3	Description Item 3	3
Item 4	Description Item 4	4

CHAPTER 5

Python Code of Markdown File Example

```
# Python
#
# This file implements an example.
#
# This file is part of mdutils. https://github.com/didix21/mdutils
#
# MIT License: (C) 2018 Dídac Coll


from mdutils.mdutils import MdUtils
from mdutils import Html

mdFile = MdUtils(file_name='Example_Markdown', title='Markdown File Example')

mdFile.new_header(level=1, title='Overview')  # style is set 'atx' format by default.

mdFile.new_paragraph("This is an example of markdown file created using mdutils␣
↪python package. In this example you "
                     "are going to see how to create a markdown file using this␣
↪library. Moreover, you're "
                     "finding the available features which makes easy the creation of␣
↪this type of files while you "
                     "are running Python code.")
mdFile.new_paragraph("**IMPORTANT:** some features available on this library have no␣
↪effect with the GitHub Markdown "
                     "CSS. Some of them are: coloring text, centering text...")
mdFile.new_paragraph()

# Available Features
mdFile.new_header(level=1, title="This is what you can do")

#␣
↪**********************************************************************************************************
# ************************************************* Markdown␣
↪*************************************************
```

(continues on next page)

(continued from previous page)

```
#␣
↪**********************************************************************************************
mdFile.new_header(level=2, title="Create Markdown files")
mdFile.new_paragraph("``create_md_file()`` is the last command that has to be called.
↪")
mdFile.insert_code("import Mdutils\n"
                   "\n"
                   "\n"
                   "mdFile = MdUtils(file_name=\'Example_Markdown\',title=\'Markdown␣
↪File Example\')\n"
                   "mdFile.create_md_file()", language='python')

#␣
↪**********************************************************************************************
# ************************************************* Headers␣
↪*************************************************
#␣
↪**********************************************************************************************
mdFile.new_header(level=2, title="Create Headers")
mdFile.new_paragraph("Using ``new_header`` method you can create headers of different␣
↪levels depending on the style. "
                     "There are two available styles: 'atx' and 'setext'. The first␣
↪one has til 6 different header "
                     "levels. Atx's levels 1 and 2 are automatically added to the␣
↪table of contents unless the "
                     "parameter ``add_table_of_contents`` is set to 'n'. The 'setext'␣
↪style only has two levels"
                     "of headers.")

mdFile.insert_code("mdFile.new_header(level=1, title='Atx Header 1')\n"
                   "mdFile.new_header(level=2, title='Atx Header 2')\n"
                   "mdFile.new_header(level=3, title='Atx Header 3')\n"
                   "mdFile.new_header(level=4, title='Atx Header 4')\n"
                   "mdFile.new_header(level=5, title='Atx Header 5')\n"
                   "mdFile.new_header(level=6, title='Atx Header 6')", language=
↪'python')

mdFile.new_header(level=1, title='Atx Header 1', add_table_of_contents='n')
mdFile.new_header(level=2, title='Atx Header 2', add_table_of_contents='n')
mdFile.new_header(level=3, title='Atx Header 3')
mdFile.new_header(level=4, title='Atx Header 4')
mdFile.new_header(level=5, title='Atx Header 5')
mdFile.new_header(level=6, title='Atx Header 6')

mdFile.insert_code("mdFile.new_header(level=1, title='Setext Header 1', style='setext
↪')\n"
                   "mdFile.new_header(level=2, title='Setext Header 2', style='setext
↪')", language='python')

mdFile.new_header(level=1, title='Setext Header 1', style='setext', add_table_of_
↪contents='n')
mdFile.new_header(level=2, title='Setext Header 2', style='setext', add_table_of_
↪contents='n')
mdFile.new_paragraph()  # Add two jump lines

#␣
↪**********************************************************************************************
```

(continues on next page)

(continued from previous page)

```
# ****************************************** Create a table of contents␣
↪******************************************
#␣
↪*********************************************************************************************
mdFile.new_header(level=2, title='Table of Contents')
mdFile.new_paragraph("If you have defined some headers of level 1 and 2, you can␣
↪create a table of contents invoking "
                     "the following command (Normally, the method will be called at␣
↪the end of the code before calling "
                     "``create_md_file()``)")
mdFile.insert_code("mdFile.new_table_of_contents(table_title='Contents', depth=2)",␣
↪language='python')

#␣
↪*********************************************************************************************
# ****************************************** Paragraph and Text format␣
↪******************************************
#␣
↪*********************************************************************************************
mdFile.new_header(level=2, title="Paragraph and Text Format")
mdFile.new_paragraph("mdutils allows you to create paragraph, line breaks or simply␣
↪write text:")
# ***************************************************** Paragraph␣
↪*****************************************************
mdFile.new_header(3, "New Paragraph Method")

mdFile.insert_code("mdFile.new_paragraph(\"Using ``new_paragraph`` method you can␣
↪very easily add a new paragraph\" \n"
                   "\t\t\t\t\t \" This example of paragraph has been added using this␣
↪method. Moreover,\"\n"
                   "\t\t\t\t\t \"``new_paragraph`` method make your live easy because␣
↪it can give format\" \n"
                   "\t\t\t\t\t \" to the text. Lets see an example:\")", language=
↪'python')

mdFile.new_paragraph("Using ``new_paragraph`` method you can very easily add a new␣
↪paragraph on your markdown file. "
                     "This example of paragraph has been added using this method.␣
↪Moreover, ``new_paragraph`` method "
                     "make your live easy because it can give format to the text.␣
↪Lets see an example:")

mdFile.insert_code("mdFile.new_paragraph(\"This is an example of text in which has␣
↪been added color, "
                   "bold and italics text.\", bold_italics_code='bi', color='purple')
↪", language='python')

mdFile.new_paragraph("This is an example of text in which has been added color, bold␣
↪and italics text.",
                     bold_italics_code='bi', color='purple')
# ************************************************ New Line␣
↪*****************************************************
mdFile.new_header(3, "New Line Method")

mdFile.new_paragraph("``mdutils`` has a method which can create new line breaks. Lets␣
↪see it.")
mdFile.insert_code("mdFile.new_line(\"This is an example of line break which has been␣
↪created with ``new_line`` "
```

(continues on next page)

(continued from previous page)

```
                    "method.\")", language='python')
mdFile.new_line("This is an example of line break which has been created with ``new_
↪line`` method.")
mdFile.new_paragraph("As ``new_paragraph``, ``new_line`` allows users to give format␣
↪to text using "
                     "``bold_italics_code`` and ``color`` parameters:")

mdFile.insert_code("mdFile.new_line(\"This is an inline code which contains bold and␣
↪italics text and it is centered\","
                   " bold_italics_code='cib', align='center')", language='python')

mdFile.new_line("This is an inline code which contains bold and italics text and it␣
↪is centered",
                bold_italics_code='cib', align='center')
# ********************************************** write␣
↪**********************************************************
mdFile.new_header(3, "Write Method")
mdFile.new_paragraph("``write`` method writes text in a markdown file without jump␣
↪lines ``'\\n'`` and as "
                     "``new_paragraph`` and ``new_line``, you can give format to text␣
↪using the arguments "
                     "``bold_italics_code``, ``color`` and ``align``: ")

mdFile.insert_code("mdFile.write(\"The following text has been written with ``write``␣
↪method. You can use markdown "
                   "directives to write:\"\n"
                   "\t\t\t \"**bold**, _italics_, ``inline_code``... or \")\n"
                   "mdFile.write(\"use the following available parameters:  \\n\")",␣
↪language='python')

mdFile.write("\n\nThe following text has been written with ``write`` method. You can␣
↪use markdown directives to write: "
             "**bold**, _italics_, ``inline_code``... or ")
mdFile.write("use the following available parameters:  \n")

mdFile.insert_code("mdFile.write('  \\n')\n"
                   "mdFile.write('bold_italics_code', bold_italics_code='bic')\n"
                   "mdFile.write('  \\n')\n"
                   "mdFile.write('Text color', color='green')\n"
                   "mdFile.write('  \\n')\n"
                   "mdFile.write('Align Text to center', align='center')", language=
↪'python')

mdFile.write('  \n')
mdFile.write('bold_italics_code', bold_italics_code='bic')
mdFile.write('  \n')
mdFile.write('Text color', color='green')
mdFile.write('  \n')
mdFile.write('Align Text to center', align='center')
mdFile.write('  \n')

#␣
↪**********************************************************************************************************
# ********************************************** Create a Table␣
↪**********************************************
#␣
↪**********************************************************************************************************
```

(continues on next page)

(continued from previous page)

```
mdFile.new_header(2, "Create a Table")
mdFile.new_paragraph("The library implements a method called ``new_table`` that can␣
↪create tables using a list of "
                     "strings. This method only needs: the number of rows and columns␣
↪that your table must have. "
                     "Optionally you can align the content of the table using the␣
↪parameter ``text_align``")

mdFile.insert_code("list_of_strings = [\"Items\", \"Descriptions\", \"Data\"]\n"
                   "for x in range(5):\n"
                   "\tlist_of_strings.extend([\"Item \" + str(x), \"Description Item \
↪" + str(x), str(x)])\n"
                   "mdFile.new_line()\n"
                   "mdFile.new_table(columns=3, rows=6, text=list_of_strings, text_
↪align='center')", language='python')

list_of_strings = ["Items", "Descriptions", "Data"]
for x in range(5):
    list_of_strings.extend(["Item " + str(x), "Description Item " + str(x), str(x)])
mdFile.new_line()
mdFile.new_table(columns=3, rows=6, text=list_of_strings, text_align='center')

#␣
↪**********************************************************************************************
# ********************************************** Create Link␣
↪**************************************************
#␣
↪**********************************************************************************************

mdFile.new_header(2, "Create Links")

# ************************************************ Inline link␣
↪**********************************************************

mdFile.new_header(3, "Create inline links")

link = "https://github.com/didix21/mdutils"
text = "mdutils"

mdFile.new_paragraph("``new_inline_link`` method allows you to create a link of the␣
↪style: "
                     "``[mdutils](https://github.com/didix21/mdutils)``.\n")
mdFile.new_paragraph("Moreover, you can add bold, italics or code in the link text.␣
↪Check the following examples: \n")

mdFile.insert_code("mdFile.new_line('  - Inline link: '"
                   " + mdFile.new_inline_link(link='{}', text='{}')) \n".format(link,␣
↪text) +
                   "mdFile.new_line('  - Bold inline link: ' "
                   "+ mdFile.new_inline_link(link='{}', text='{}', bold_italics_code=
↪'b') \n".format(link, text) +
                   "mdFile.new_line('  - Italics inline link: ' "
                   "+ mdFile.new_inline_link(link='{}', text='{}', bold_italics_code=
↪'i') \n".format(link, text) +
                   "mdFile.new_line('  - Code inline link: ' "
                   "+ mdFile.new_inline_link(link='{}', text='{}', bold_italics_code=
↪'i') \n".format(link, text) +
```

(continues on next page)

(continued from previous page)

```
                    "mdFile.new_line('  - Bold italics code inline link: ' "
                    "+ mdFile.new_inline_link(link='{}', text='{}', bold_italics_code=
↪'cbi') \n".format(link, text) +
                    "mdFile.new_line('  - Another inline link: ' + mdFile.new_inline_
↪link(link='{}') \n".format(link),
                    language='python')

mdFile.new_line('  - Inline link: ' + mdFile.new_inline_link(link=link, text=text))
mdFile.new_line('  - Bold inline link: ' + mdFile.new_inline_link(link=link,␣
↪text=text, bold_italics_code='b'))
mdFile.new_line('  - Italics inline link: ' + mdFile.new_inline_link(link=link,␣
↪text=text, bold_italics_code='i'))
mdFile.new_line('  - Code inline link: ' + mdFile.new_inline_link(link=link,␣
↪text=text, bold_italics_code='c'))
mdFile.new_line(
    '  - Bold italics code inline link: ' + mdFile.new_inline_link(link=link,␣
↪text=text, bold_italics_code='cbi'))
mdFile.new_line('  - Another inline link: ' + mdFile.new_inline_link(link=link))

# ********************************************* Reference link␣
↪**************************************************
mdFile.new_header(3, "Create reference links")

mdFile.new_paragraph("``new_reference_link`` method allows you to create a link of␣
↪the style: "
                     "``[mdutils][1]``. All references will be added at the end of␣
↪the markdown file automatically as: \n")

mdFile.insert_code("[1]: https://github.com/didix21/mdutils", language="python")
mdFile.new_paragraph("Lets check some examples: \n")

link = "https://github.com/didix21/mdutils"

mdFile.insert_code("mdFile.write('\\n  - Reference link: ' "
                    "+ mdFile.new_reference_link(link='{}', text='mdutils', reference_
↪tag='1')\n".format(link) +
                    "mdFile.write('\\n  - Reference link: ' "
                    "+ mdFile.new_reference_link(link='{}', text='another reference',␣
↪reference_tag='md')\n".format(
                        link) +
                    "mdFile.write('\\n  - Bold link: ' "
                    "+ mdFile.new_reference_link(link='{}', text='Bold reference',␣
↪reference_tag='bold', bold_italics_code='b')\n".format(
                        link) +
                    "mdFile.write('\\n  - Italics link: ' "
                    "+ mdFile.new_reference_link(link='{}', text='Bold reference',␣
↪reference_tag='italics', bold_italics_code='i')\n".format(
                        link),
                    language="python")

mdFile.write("\n  - Reference link: " + mdFile.new_reference_link(link=link, text=
↪'mdutils', reference_tag='1'))
mdFile.write(
    "\n  - Reference link: " + mdFile.new_reference_link(link=link, text='another␣
↪reference', reference_tag='md'))
mdFile.write("\n  - Bold link: " + mdFile.new_reference_link(link=link, text='Bold␣
↪reference', reference_tag='bold',
```

(continues on next page)

(continued from previous page)

```
                                                                    bold_italics_code='b'))
mdFile.write(
    "\n  - Italics link: " + mdFile.new_reference_link(link=link, text='Italics␣
↪reference', reference_tag='italics',
                                                        bold_italics_code='i'))

#␣
↪*****************************************************************************************************
# ************************************************** Create Lists␣
↪*****************************************************
#␣
↪*****************************************************************************************************
mdFile.new_header(2, "Create Lists")
# ********************************************** Unordered Lists␣
↪*****************************************************
mdFile.new_header(3, "Create unordered lists")
mdFile.new_paragraph(
    "You can add Mark down unordered list using ``mdFile.new_list(items, marked_
↪with)``. Lets check an example: ")
items = ["Item 1", "Item 2", "Item 3", "Item 4", ["Item 4.1", "Item 4.2", ["Item 4.2.1
↪", "Item 4.2.2"],
                                                    "Item 4.3", ["Item 4.3.1"]], "Item 5
↪"]
mdFile.insert_code(f'items = {items}\n'
                   f'mdFile.new_list(items)\n')
mdFile.new_list(items=items)

# ********************************************** Ordered Lists␣
↪*****************************************************
mdFile.new_header(3, "Create ordered lists")
mdFile.new_paragraph("You can add ordered ones easily, too: ``mdFile.new_list(items,␣
↪marked_with='1')``")
mdFile.new_list(items=items, marked_with='1')

mdFile.new_paragraph("Moreover, you can add mixed list, for example: ")
items = ["Item 1", "Item 2", ["1. Item 2.1", "2. Item 2.2"], "Item 3"]
mdFile.insert_code(f'items = {items}\n'
                   f'mdFile.new_list(items)\n')
mdFile.new_list(items)
mdFile.new_paragraph("Maybe you want to replace the default hyphen ``-`` by a ``+``␣
↪or ``*`` then you can do: "
                     "``mdFile.new_list(items, marked_with='*')``.")

#␣
↪*****************************************************************************************************
# ************************************************** Add Images␣
↪*****************************************************
#␣
↪*****************************************************************************************************

mdFile.new_header(2, "Add images")

# ********************************************** Inline Image␣
↪*****************************************************

image_text = "snow trees"
path = "./doc/source/images/photo-of-snow-covered-trees.jpg"
```

(continues on next page)

(continued from previous page)

```
mdFile.new_header(3, "Inline Images")

mdFile.new_paragraph("You can add inline images using ``new_inline_image`` method.␣
↪Method will return: "
                     "``[image](../path/to/your/image.png)``. Check the following␣
↪example: ")
mdFile.insert_code("mdFile.new_line(mdFile.new_inline_image(text='{}', path='{}'))".
↪format(image_text, path))
mdFile.new_line(mdFile.new_inline_image(text=image_text, path=path))

# ********************************************** Reference Image␣
↪**************************************************
mdFile.new_header(3, "Reference Images")
mdFile.new_paragraph("You can add inline images using ``new_reference_image`` method.␣
↪Method will return: "
                     "``[image][im]``. Check the following example: ")
mdFile.insert_code(
    "mdFile.new_line(mdFile.new_reference_image(text='{}', path='{}', reference_tag=
↪'im'))".format(image_text, path))
mdFile.new_line(mdFile.new_reference_image(text=image_text, path=path, reference_tag=
↪'im'))

# ************************************************ Html Image␣
↪**************************************************

mdFile.new_header(2, "Add HTML images")

# ********************************************** Size Image␣
↪**************************************************

mdFile.new_header(3, "Change size to images")
path = "./doc/source/images/sunset.jpg"

mdFile.new_paragraph("With ``Html.image`` you can change size of images in a markdown␣
↪file. For example you can do"
                     "the following for changing width: ``mdFile.new_paragraph(Html.
↪image(path=path, size='200'))``")

mdFile.new_paragraph(Html.image(path=path, size='200'))

mdFile.new_paragraph(
    "Or maybe only want to change height: ``mdFile.new_paragraph(Html.image(path=path,
↪ size='x300'))``")
mdFile.new_paragraph(Html.image(path=path, size='x300'))

mdFile.new_paragraph("Or change width and height: ``mdFile.new_paragraph(Html.
↪image(path=path, size='300x300'))``")
mdFile.new_paragraph(Html.image(path=path, size='300x300'))
mdFile.write('\n')

# ********************************************** Align Image␣
↪**************************************************

mdFile.new_header(3, "Align images")
mdFile.new_paragraph("Html.image allow to align images, too. For example you can run:
↪"
```

(continues on next page)

(continued from previous page)

```
                    "``mdFile.new_paragraph(Html.image(path=path, size='300x200',␣
↪align='center'))``")

mdFile.new_paragraph(Html.image(path=path, size='300x200', align='center'))

# Create a table of contents
mdFile.new_table_of_contents(table_title='Contents', depth=2)
mdFile.create_md_file()
```

CHAPTER 6

mdutils

6.1 mdutils package

6.1.1 Subpackages

mdutils.fileutils package

Submodules

mdutils.fileutils.fileutils module

class mdutils.fileutils.fileutils.**MarkDownFile**(*name=''*, *dirname: Optional[str] = None*)

Bases: object

MarkDownFile class creates a new file of MarkDown extension.

Features available are:

- Create a file.
- Rewrite a file with new data.
- Write at the end of the file.

append_after_second_line(*data: str*)

Write after the file's first line.

Parameters **data** (*str*) – is a string containing all the data that is written in the markdown file.

append_end(*data: str*)

Write at the last position of a Markdown file.

Parameters **data** (*str*) – is a string containing all the data that is written in the markdown file.

static read_file(*file_name: str*) → str
Read a Markdown file using a file name. It is not necessary to add *.md extension.

Parameters **file_name** (*str*) – Markdown file's name.

Returns return all file's data.

Return type str

rewrite_all_file(*data: str*)
Rewrite all the data of a Markdown file by `data`.

Parameters **data** (*str*) – is a string containing all the data that is written in the markdown file.

Module contents

mdutils.tools package

Submodules

mdutils.tools.Header module

class mdutils.tools.Header.**AtxHeaderLevel**
Bases: enum.Enum

An enumeration.

HEADING = 2

LEASTHEADING = 6

MINORHEADING = 5

SUBHEADING = 3

SUBSUBHEADING = 4

TITLE = 1

class mdutils.tools.Header.**Header**(*level: int*, *title: str*, *style: mdutils.tools.Header.HeaderStyle*, *header_id: str = None*)
Bases: object

Contain the main methods to define Headers on a Markdown file.

Features available: - Create Markdown Titles: *atx* and *setext* formats are available. - Create Header Anchors.

Example

```
>>> str(Header(level=1, title='New Header', style=HeaderStyle.ATX))
```

static atx(*level: mdutils.tools.Header.AtxHeaderLevel*, *title: str*, *header_id: str = None*) → str
Return an atx-style header.

Parameters

- **title** – Text title.
- **level** – HeaderLevel enum member, e.g., TITLE, HEADING, SUBHEADING, etc.
- **header_id** – ID of the header for extended Markdown syntax (optional)

Returns an atx-style header string

static choose_header(*level: int*, *title: str*, *style: str = 'atx'*, *header_id: str = ''*) → str

This method choose the style and the header level.

Examples

```
>>> from mdutils.tools.Header import Header
>>> Header.choose_header(level=1, title='New Header', style='atx')
'\n# New Header\n'
```

```
>>> Header.choose_header(level=2, title='Another Header 1', style=
↪'setext')
'\nAnother Header 1\n----------------\n'
```

Parameters

- **level** – Header Level, For Atx-style 1 til 6. For Setext-style 1 and 2 header levels.
- **title** – Header Title.
- **style** – Header Style atx or setext.
- **header_id** – ID of the header for extended Markdown syntax

Returns

static header_anchor(*text: str*, *link: str = None*) → str

Create an internal link to a defined header level in the markdown file.

Parameters

- **text** – Displayed text for the link.
- **link** – Internal link (optional). If not provided, it is generated based on the text.

Returns a header anchor string

Examples:

1. Using the default generated link based on the text:

```
>>> header_link = Header.header_anchor("Section 1")
>>> print(header_link)
[Section 1](#section-1)
```

2. Providing a custom link:

```
>>> header_link = Header.header_anchor("Section 1", "custom-link")
>>> print(header_link)
[Section 1](#custom-link)
```

3. Providing a link with an existing '#' symbol:

```
>>> header_link = Header.header_anchor("Section 1", "#existing-link")
>>> print(header_link)
[Section 1](#existing-link)
```

static setext(*level: mdutils.tools.Header.SetextHeaderLevel*, *title: str*) → str

Return a setext-style header.

Parameters

- **title** – Text title.
- **level** – HeaderLevel enum member, e.g., TITLE, HEADING.

Returns a setext-style header string

class mdutils.tools.Header.**HeaderStyle**
Bases: enum.Enum

An enumeration.

ATX = 1

SETEXT = 2

class mdutils.tools.Header.**SetextHeaderLevel**
Bases: enum.Enum

An enumeration.

HEADING = 2

TITLE = 1

mdutils.tools.Html module

class mdutils.tools.Html.**Html**
Bases: object

classmethod image(*path: str*, *size: str = None*, *align: str = None*) → str

Parameters

- **path** (*str*) –
- **size** (*str*) – (In px) for width write `'<int>'`, for height write `'x<int>'` or width and height `'<int>x<int>`.
- **align** (*str*) – can be `'left'`, `'center'` or `'right'`.

Returns html format

Return type str

Example

```
>>> Html.image(path='../image.jpg', size='200', align='center')
>>> Html.image(path='../image.jpg', size='x200', align='left')
>>> Html.image(path='../image.jpg', size='300x400')
```

static paragraph(*text: str*, *align: str = None*) → str

Parameters

- **text** –
- **align** (*str*) – `center` or `right`.

Returns `"<p align="{}">\n {}\n</p>".format(align, text)`.

Return type str

class mdutils.tools.Html.**HtmlSize**
Bases: object

classmethod size_to_width_and_height(*size: str*) → str

exception mdutils.tools.Html.**SizeBadFormat**(*message*)
Bases: Exception

Raise exception when size does not match the expected format

mdutils.tools.Image module

class mdutils.tools.Image.**Image**(*reference: mdutils.tools.Link.Reference*)
Bases: object

static new_inline_image(*text: str*, *path: str*, *tooltip: str = None*) → str

Parameters

- **text** (*str*) – Text that is going to be displayed in the markdown file as a iamge.
- **path** (*str*) – Image's path / link.
- **tooltip** (*str*) –

Returns return the image in markdown format `'![ + text + '](' + path + 'tooltip' + ')'`.

Return type str

new_reference_image(*text: str*, *path: str*, *reference_tag: str = None*, *tooltip: str = None*) → str

Parameters

- **text** (*str*) – Text that is going to be displayed in the markdown file as a image.
- **path** (*str*) – Image's path / link.
- **reference_tag** (*str*) – Tag that will be placed at the end of the markdown file jointly with the image's path.
- **tooltip** (*str*) –

Returns return the image in markdown format `'![ + text + '][' + reference_tag + ']'`.

Return type str

mdutils.tools.Link module

class mdutils.tools.Link.**Inline**
Bases: object

static new_link(*link: str*, *text: str = None*, *tooltip: str = None*)

Parameters

- **link** (*str*) –
- **text** (*str*) – Text that is going to be displayed in the markdown file as a link.
- **tooltip** (*str*) – Add a tool tip on the image.

Returns `'[' + text +'](' + link + 'tooltip' + ')'` or if link is only defined: `` `<` + link '>' ``.

Return type str

class mdutils.tools.Link.**Reference**
Bases: object

get_references() → dict

Returns

Return type dict

get_references_as_markdown() → str

Returns

Return type str

new_link(*link: str*, *text: str*, *reference_tag: str = None*, *tooltip: str = None*) → str

Parameters

- **link** (*str*) –
- **text** (*str*) – Text that is going to be displayed in the markdown file as a link.
- **reference_tag** (*str*) – Reference that will be saved on reference dict.
- **tooltip** (*str*) – Add a tooltip on the link.

Returns `'[' + text + '][' + reference_tag + ']'` or if reference_Tag is not defined: `'[' + text + ']'`.

Return type str

mdutils.tools.MDList module

class mdutils.tools.MDList.**MDCheckbox**(*items*, *checked: bool = False*)
Bases: *mdutils.tools.MDList.MDListHelper*

This class allows to create checkbox MarkDown list.

get_md() → str

class mdutils.tools.MDList.**MDList**(*items*, *marked_with: str = '-'*)
Bases: *mdutils.tools.MDList.MDListHelper*

This class allows to create unordered or ordered MarkDown list.

get_md() → str
Get the list in markdown format.

Returns

Return type str

class mdutils.tools.MDList.**MDListHelper**
Bases: object

mdutils.tools.Table module

class mdutils.tools.Table.**Table**
Bases: object

create_table(*columns: int, rows: int, text: List[str], text_align: Union[str, list, None] = None*)
This method takes a list of strings and creates a table.

Using arguments `columns` and `rows` allows to create a table of *n* columns and *m* rows. The `columns * rows` operations has to correspond to the number of elements of `text` list argument.

Parameters

- **columns** (*int*) – number of columns of the table.
- **rows** (*int*) – number of rows of the table.
- **text** (*list of str*) – a list of strings.
- **text_align** (*str_or_list*) – text align argument. Values available are: `'right'`, `'center'`, `'left'` and `None` (default). If text_align is a list then individual alignment can be set for each column.

Returns a markdown table.

Return type str

Example

```
>>> from mdutils.tools.Table import Table
>>> text_list = ['List of Items', 'Description', 'Result', 'Item 1',
↪'Description of item 1', '10', 'Item 2', 'Description of item 2', '0']
>>> table = Table().create_table(columns=3, rows=3, text=text_list, text_
↪align='center')
>>> print(repr(table))
'\n|List of Items|Description|Result|\n| :---: | :---: | :---: |\n|Item␣
↪1|Description of item 1|10|\n|Item 2|Description of item 2|0|\n'
```

Table 1: **Table result on Markdown**

List of Items	Description	Results
Item 1	Description of Item 1	10
Item 2	Description of Item 2	0

mdutils.tools.TableOfContents module

class mdutils.tools.TableOfContents.**TableOfContents**

Bases: object

create_table_of_contents(*array_of_title_contents: List[str], depth: int = 1*) → str

This method can create a table of contents using an array of the different titles. The depth can be changed. :param array_of_title_contents: a string list with the different headers. :type array_of_title_contents: list :param depth: allows to include atx headers 1 through 6. Possible values: 1, 2, 3, 4, 5, or 6. :type depth: int :return: return a string ready to be written to a Markdown file. :rtype: str

mdutils.tools.TextUtils module

class mdutils.tools.TextUtils.**TextUtils**

Bases: object

This class helps to create bold, italics and change color text.

static add_tooltip(*link: str, tip: str*) → str

Parameters

- **link** (*str*) –
- **tip** (*str*) –

return: `link + "'" + format + "'"`

static bold(*text: str*) → str
Bold text converter.

Parameters **text** (*str*) – a text string.

Returns a string like this example: `'**text**'`

Return type str

static center_text(*text: str*) → str
Place a text string to center.

Parameters **text** (*str*) – a text string.

Returns a string like this exampple: `'<center>text</center>'`

static inline_code(*text: str*) → str
Inline code text converter.

Parameters **text** (*str*) – a text string.

Returns a string like this example: ``'``text''``

Return type str

static insert_code(*code: str*, *language: str = ''*) → str
This method allows to insert a peace of code.

Parameters **code** – code string.

:type code:str :param language: code language: python. c++, c#... :type language: str :return: markdown style. :rtype: str

static italics(*text: str*) → str
Italics text converter.

Parameters **text** (*str*) – a text string.

Returns a string like this example: `'_text_'`

Return type str

static text_color(*text: str*, *color: str = 'black'*) → str
Change text color.

Parameters

- **text** (*str*) – it is the text that will be changed its color.
- **color** (*str*) – it is the text color: `'orange'`, `'blue'`, `'red'`... or a **RGB** color such as `'#ffce00'`.

Returns a string like this one: `'<font color='color'>'text'</font>'`

Return type str

static text_external_link(*text: str*, *link: str = ''*) → str
Using this method can be created an external link of a file or a web page.

Parameters

- **text** (*str*) – Text to be displayed.
- **link** (*str*) – External Link.

Returns return a string like this: `'[Text to be shown](https://write.link.com)'`

Return type str

static text_format(*text: str*, *bold_italics_code: str = ''*, *color: str = 'black'*, *align: str = ''*) → str

Text format helps to write multiple text format such as bold, italics and color.

Parameters

- **text** (*str*) – it is a string in which will be added the mew format
- **bold_italics_code** (*str*) – using *'b'*: **bold**, *'i'*: _italics_ and *'c'*: *inline_code*.
- **color** (*str*) – Can change text color. For example: 'red', 'green, 'orange'...
- **align** (*str*) – Using this parameter you can align text.

Returns return a string with the new text format.

Return type str

Example

```
>>> from mdutils.tools.TextUtils import TextUtils
>>> TextUtils.text_format(text='Some Text Here', bold_italics_code='bi',␣
↪color='red', align='center')
'***<center><font color="red">Some Text Here</font></center>***'
```

Module contents

6.1.2 Submodules

6.1.3 mdutils.mdutils module

Module **mdutils**

The available features are:

- Create Headers, Til 6 sub-levels.
- Auto generate a table of contents.
- Create List and sub-list.
- Create paragraph.
- Generate tables of different sizes.
- Insert Links.
- Insert Code.
- Place text anywhere using a marker.

class mdutils.mdutils.**MdUtils**(*file_name: str*, *title: str = ''*, *author: str = ''*)

Bases: object

This class give some basic methods that helps the creation of Markdown files while you are executing a python code.

The `__init__` variables are:

- **file_name:** it is the name of the Markdown file.
- **author:** it is the author fo the Markdown file.
- **header:** it is an instance of Header Class.
- **textUtils:** it is an instance of TextUtils Class.
- **title:** it is the title of the Markdown file. It is written with Setext-style.
- **table_of_contents:** it is the table of contents, it can be optionally created.
- **file_data_text:** contains all the file data that will be written on the markdown file.

create_marker(*text_marker: str*) → str

This will add a marker to `file_data_text` and returns the marker result in order to be used whenever you need.

Markers allows to place them to the string data text and they can be replaced by a peace of text using `place_text_using_marker` method.

Parameters **text_marker** (*str*) – marker name.

Returns return a marker of the following form: `'##--[' + text_marker + ']--##'`

Return type str

create_md_file() → mdutils.fileutils.fileutils.MarkDownFile
It creates a new Markdown file. :return: return an instance of a MarkDownFile.

get_md_text() → str
Instead of writing the markdown text into a file it returns it as a string.

Returns return a string with the markdown text.

insert_code(*code: str*, *language: str = ''*) → str
This method allows to insert a peace of code on a markdown file.

Parameters

- **code** (*str*) – code string.
- **language** (*str*) – code language: python, c++, c#...

Returns

Return type str

new_checkbox_list(*items: List[str]*, *checked: bool = False*)
Add checkbox list in MarkDown file.

Parameters

- **items** (*[str]*) – Array of items for generating the checkbox list.
- **checked** (*bool*) – if you set this to `True`. All checkbox will be checked. By default is `False`.

Returns

new_header(*level: int*, *title: str*, *style: str = 'atx'*, *add_table_of_contents: str = 'y'*, *header_id: str = ''*) → str

Add a new header to the Markdown file.

Parameters

- **level** (*int*) – Header level. *atx* style can take values 1 til 6 and *setext* style take values 1 and 2.
- **title** (*str*) – Header title.
- **style** (*str*) – Header style, can be `'atx'` or `'setext'`. By default `'atx'` style is chosen.
- **add_table_of_contents** (*str*) – by default the atx and setext headers of level 1 and 2 are added to the table of contents, setting this parameter to 'n'.
- **header_id** (*str*) – ID of the header for extended Markdown syntax

Example

```
>>> from mdutils import MdUtils
>>> mdfile = MdUtils("Header_Example")
>>> print(mdfile.new_header(level=2, title='Header Level 2 Title', style='atx
↪', add_table_of_contents='y'))
'\n## Header Level 2 Title\n'
>>> print(mdfile.new_header(level=2, title='Header Title', style='setext'))
'\nHeader Title\n------------\n'
```

static new_inline_image(*text: str*, *path: str*) → str

Add inline images in a markdown file. For example `[MyImage](../MyImage.jpg)`.

Parameters

- **text** (*str*) – Text that is going to be displayed in the markdown file as a iamge.
- **path** (*str*) – Image's path / link.

Returns return the image in markdown format `'![ + text + '](' + path + ')'`.

Return type str

new_inline_link(*link: str*, *text: Optional[str] = None*, *bold_italics_code: str = ''*, *align: str = ''*) → str

Creates a inline link in markdown format.

Parameters

- **link** (*str*) –
- **text** (*str*) – Text that is going to be displayed in the markdown file as a link.
- **bold_italics_code** (*str*) – Using `'b'`: **bold**, `'i'`: *italics* and `'c'`: `inline_code`...
- **align** (*str*) – Using this parameter you can align text. For example `'right'`, `'left'` or `'center'`.

Returns returns the link in markdown format `'[ + text + '](' + link + ')'`. If text is not defined returns `'<' + link + '>'`.

Return type str

Note: If param text is not provided, link param will be used instead.

new_line(*text: str = ''*, *bold_italics_code: str = ''*, *color: str = 'black'*, *align: str = ''*, *wrap_width: int = 0*) → str

Add a new line to Markdown file. The text is saved to the global variable file_data_text.

Parameters

- **text** (*str*) – is a string containing the paragraph text. Optionally, the paragraph text is returned.
- **bold_italics_code** (*str*) – using `'b'`: **bold**, `'i'`: *italics* and `'c'`: `inline_code`...
- **color** (*str*) – Can change text color. For example: `'red'`, `'green'`, `'orange'`...
- **align** (*str*) – Using this parameter you can align text. For example `'right'`, `'left'` or `'center'`.
- **wrap_width** (*int*) – wraps text with designated width by number of characters. By default, long words are not broken. Use width of 0 to disable wrapping.

Returns return a string `'\n' + text`. Not necessary to take it, if only has to be written to the file.

Return type str

new_list(*items: List[str], marked_with: str = '-'*)

Add unordered or ordered list in MarkDown file.

Parameters

- **items** (*[str]*) – Array of items for generating the list.
- **marked_with** (*str*) – By default has the value of `'-'`, can be `'+'`, `'*'`. If you want to generate an ordered list then set to `'1'`.

Returns

new_paragraph(*text: str = ''*, *bold_italics_code: str = ''*, *color: str = 'black'*, *align: str = ''*, *wrap_width: int = 0*) → str

Add a new paragraph to Markdown file. The text is saved to the global variable file_data_text.

Parameters

- **text** (*str*) – is a string containing the paragraph text. Optionally, the paragraph text is returned.
- **bold_italics_code** (*str*) – using `'b'`: **bold**, `'i'`: *italics* and `'c'`: `inline_code`.
- **color** (*str*) – Can change text color. For example: `'red'`, `'green'`, `'orange'`...
- **align** (*str*) – Using this parameter you can align text.
- **wrap_width** (*int*) – wraps text with designated width by number of characters. By default, long words are not broken. Use width of 0 to disable wrapping.

Returns `'\n\n' + text`. Not necessary to take it, if only has to be written to the file.

Return type str

new_reference_image(*text: str*, *path: str*, *reference_tag: Optional[str] = None*) → str

Add reference images in a markdown file. For example `[MyImage][my_image]`. All references will be stored at the end of the markdown file.

Parameters

- **text** (*str*) – Text that is going to be displayed in the markdown file as a image.

- **path** (*str*) – Image's path / link.
- **reference_tag** (*str*) – Tag that will be placed at the end of the markdown file jointly with the image's path.

Returns return the image in markdown format `'![ + text + '][' + reference_tag + ']'`.

Return type str

Note: If param reference_tag is not provided, text param will be used instead.

---

new_reference_link(*link: str*, *text: str*, *reference_tag: Optional[str] = None*, *bold_italics_code: str = ''*, *align: str = ''*) → str

Creates a reference link in markdown format. All references will be stored at the end of the markdown file.

Parameters

- **link** (*str*) –
- **text** (*str*) – Text that is going to be displayed in the markdown file as a link.
- **reference_tag** (*str*) – Tag that will be placed at the end of the markdown file jointly with the link.
- **bold_italics_code** (*str*) – Using `'b'`: **bold**, `'i'`: *italics* and `'c'`: `inline_code`...
- **align** (*str*) – Using this parameter you can align text. For example `'right'`, `'left'` or `'center'`.

Returns returns the link in markdown format `'[ + text + '][' + link + ]'`.

Return type str

Note: If param reference_tag is not provided, text param will be used instead.

Example

```
>>> from mdutils import MdUtils
>>> md = MdUtils("Reference link")
>>> link = md.new_reference_link(link='https://github.com', text='github',␣
↪reference_tag='git')
>>> md.new_link(link)
>>> print(repr(link))
'[github][git]'
>>> link = md.new_reference_link(link='https://github.com/didix21/mdutils',␣
↪text='mdutils')
>>> md.new_link(link)
>>> print(repr(link))
'[mdutils]'
>>> link = md.new_line(md.new_reference_link(link='https://github.com/didix21/
↪mdutils', text='mdutils', reference_tag='md', bold_italics_code='b'))
>>> md.new_link(link)
>>> print(repr(link))
'[**mdutils**][md]'
>>> md.create_md_file()
```

new_table(*columns: int, rows: int, text: List[str], text_align: str = 'center', marker: str = ''*) → str

This method takes a list of strings and creates a table.

Using arguments `columns` and `rows` allows to create a table of *n* columns and *m* rows. The `columns * rows` operations has to correspond to the number of elements of `text` list argument. Moreover, `argument` allows to place the table wherever you want from the file.

Parameters

- **columns** (*int*) – this variable defines how many columns will have the table.
- **rows** (*int*) – this variable defines how many rows will have the table.
- **text** (*list*) – it is a list containing all the strings which will be placed in the table.
- **text_align** (*str*) – allows to align all the cells to the `'right'`, `'left'` or `'center'`. By default: `'center'`.
- **marker** (*str*) – using `create_marker` method can place the table anywhere of the markdown file.

Returns can return the table created as a string.

Return type str

Example

```
>>> from mdutils import MdUtils
>>> md = MdUtils(file_name='Example')
>>> text_list = ['List of Items', 'Description', 'Result', 'Item 1',
↪'Description of item 1', '10', 'Item 2', 'Description of item 2', '0']
>>> table = md.new_table(columns=3, rows=3, text=text_list, text_align='center
↪')
>>> print(repr(table))
'\n|List of Items|Description|Result|\n| :---: | :---: | :---: |\n|Item␣
↪1|Description of item 1|10|\n|Item 2|Description of item 2|0|\n'
```

Table 2: **Table result on Markdown**

List of Items	Description	Results
Item 1	Description of Item 1	10
Item 2	Description of Item 2	0

new_table_of_contents(*table_title: str = 'Table of contents', depth: int = 1, marker: str = ''*) → str

Table of contents can be created if Headers of 'atx' style have been defined.

This method allows to create a table of contents and define a title for it. Moreover, *depth* allows user to define how many levels of headers will be placed in the table of contents. If no marker is defined, the table of contents will be placed automatically after the file's title.

Parameters

- **table_title** (*str*) – The table content's title, by default "Table of contents"
- **depth** (*int*) – allows to include atx headers 1 through 6. Possible values: 1, 2, 3, 4, 5, or 6.
- **marker** (*str*) – allows to place the table of contents using a marker.

Returns a string with the data is returned.

Return type str

place_text_using_marker(*text: str*, *marker: str*) → str
It replace a previous marker created with `create_marker` with a text string.

This method is going to search for the `marker` argument, which has been created previously using `create_marker` method, in `file_data_text` string.

Parameters

- **text** (*str*) – the new string that will replace the marker.
- **marker** (*str*) – the marker that has to be replaced.

Returns return a new file_data_text with the replace marker.

Return type str

read_md_file(*file_name: str*) → str
Reads a Markdown file and save it to global class *file_data_text*.

Parameters **file_name** (*str*) – Markdown file's name that has to be read.

Returns optionally returns the file data content.

Return type str

write(*text: str = ''*, *bold_italics_code: str = ''*, *color: str = 'black'*, *align: str = ''*, *marker: str = ''*, *wrap_width: int = 0*) → str
Write text in `file_Data_text` string.

Parameters

- **text** (*str*) – a text a string.
- **bold_italics_code** (*str*) – using `'b'`: **bold**, `'i'`: *italics* and `'c'`: `inline_code`...
- **color** (*str*) – Can change text color. For example: `'red'`, `'green'`, `'orange'`...
- **align** (*str*) – Using this parameter you can align text. For example `'right'`, `'left'` or `'center'`.
- **wrap_width** (*int*) – wraps text with designated width by number of characters. By default, long words are not broken. Use width of 0 to disable wrapping.
- **marker** (*str*) – allows to replace a marker on some point of the file by the text.

6.1.4 Module contents

CHAPTER 7

Indices and tables

- genindex
- modindex
- search

Python Module Index

m

mdutils, 35
mdutils.fileutils, 22
mdutils.fileutils.fileutils, 21
mdutils.mdutils, 29
mdutils.tools, 29
mdutils.tools.Header, 22
mdutils.tools.Html, 24
mdutils.tools.Image, 25
mdutils.tools.Link, 25
mdutils.tools.MDList, 26
mdutils.tools.Table, 26
mdutils.tools.TableOfContents, 27
mdutils.tools.TextUtils, 27

Index

A

add_tooltip() (*mdutils.tools.TextUtils.TextUtils static method*), 27
append_after_second_line() (*mdutils.fileutils.fileutils.MarkDownFile method*), 21
append_end() (*mdutils.fileutils.fileutils.MarkDownFile method*), 21
ATX (*mdutils.tools.Header.HeaderStyle attribute*), 24
atx() (*mdutils.tools.Header.Header static method*), 22
AtxHeaderLevel (*class in mdutils.tools.Header*), 22

B

bold() (*mdutils.tools.TextUtils.TextUtils static method*), 28

C

center_text() (*mdutils.tools.TextUtils.TextUtils static method*), 28
choose_header() (*mdutils.tools.Header.Header static method*), 23
create_marker() (*mdutils.mdutils.MdUtils method*), 30
create_md_file() (*mdutils.mdutils.MdUtils method*), 30
create_table() (*mdutils.tools.Table.Table method*), 26
create_table_of_contents() (*mdutils.tools.TableOfContents.TableOfContents method*), 27

G

get_md() (*mdutils.tools.MDList.MDCheckbox method*), 26
get_md() (*mdutils.tools.MDList.MDList method*), 26
get_md_text() (*mdutils.mdutils.MdUtils method*), 30
get_references() (*mdutils.tools.Link.Reference method*), 26
get_references_as_markdown() (*mdutils.tools.Link.Reference method*), 26

H

Header (*class in mdutils.tools.Header*), 22
header_anchor() (*mdutils.tools.Header.Header static method*), 23
HeaderStyle (*class in mdutils.tools.Header*), 24
HEADING (*mdutils.tools.Header.AtxHeaderLevel attribute*), 22
HEADING (*mdutils.tools.Header.SetextHeaderLevel attribute*), 24
Html (*class in mdutils.tools.Html*), 24
HtmlSize (*class in mdutils.tools.Html*), 24

I

Image (*class in mdutils.tools.Image*), 25
image() (*mdutils.tools.Html.Html class method*), 24
Inline (*class in mdutils.tools.Link*), 25
inline_code() (*mdutils.tools.TextUtils.TextUtils static method*), 28
insert_code() (*mdutils.mdutils.MdUtils method*), 30
insert_code() (*mdutils.tools.TextUtils.TextUtils static method*), 28
italics() (*mdutils.tools.TextUtils.TextUtils static method*), 28

L

LEASTHEADING (*mdutils.tools.Header.AtxHeaderLevel attribute*), 22

M

MarkDownFile (*class in mdutils.fileutils.fileutils*), 21
MDCheckbox (*class in mdutils.tools.MDList*), 26
MDList (*class in mdutils.tools.MDList*), 26
MDListHelper (*class in mdutils.tools.MDList*), 26
MdUtils (*class in mdutils.mdutils*), 29
mdutils (*module*), 35
mdutils.fileutils (*module*), 22

mdutils.fileutils.fileutils (*module*), 21
mdutils.mdutils (*module*), 29
mdutils.tools (*module*), 29
mdutils.tools.Header (*module*), 22
mdutils.tools.Html (*module*), 24
mdutils.tools.Image (*module*), 25
mdutils.tools.Link (*module*), 25
mdutils.tools.MDList (*module*), 26
mdutils.tools.Table (*module*), 26
mdutils.tools.TableOfContents (*module*), 27
mdutils.tools.TextUtils (*module*), 27
MINORHEADING (*mdutils.tools.Header.AtxHeaderLevel attribute*), 22

N

new_checkbox_list() (*mdutils.mdutils.MdUtils method*), 30
new_header() (*mdutils.mdutils.MdUtils method*), 30
new_inline_image() (*mdutils.mdutils.MdUtils static method*), 31
new_inline_image() (*mdutils.tools.Image.Image static method*), 25
new_inline_link() (*mdutils.mdutils.MdUtils method*), 31
new_line() (*mdutils.mdutils.MdUtils method*), 31
new_link() (*mdutils.tools.Link.Inline static method*), 25
new_link() (*mdutils.tools.Link.Reference method*), 26
new_list() (*mdutils.mdutils.MdUtils method*), 32
new_paragraph() (*mdutils.mdutils.MdUtils method*), 32
new_reference_image() (*mdutils.mdutils.MdUtils method*), 32
new_reference_image() (*mdutils.tools.Image.Image method*), 25
new_reference_link() (*mdutils.mdutils.MdUtils method*), 33
new_table() (*mdutils.mdutils.MdUtils method*), 33
new_table_of_contents() (*mdutils.mdutils.MdUtils method*), 34

P

paragraph() (*mdutils.tools.Html.Html static method*), 24
place_text_using_marker() (*mdutils.mdutils.MdUtils method*), 35

R

read_file() (*mdutils.fileutils.fileutils.MarkDownFile static method*), 21
read_md_file() (*mdutils.mdutils.MdUtils method*), 35
Reference (*class in mdutils.tools.Link*), 25
rewrite_all_file() (*mdutils.fileutils.fileutils.MarkDownFile method*), 22

S

SETEXT (*mdutils.tools.Header.HeaderStyle attribute*), 24
setext() (*mdutils.tools.Header.Header static method*), 23
SetextHeaderLevel (*class in mdutils.tools.Header*), 24
size_to_width_and_height() (*mdutils.tools.Html.HtmlSize class method*), 24
SizeBadFormat, 25
SUBHEADING (*mdutils.tools.Header.AtxHeaderLevel attribute*), 22
SUBSUBHEADING (*mdutils.tools.Header.AtxHeaderLevel attribute*), 22

T

Table (*class in mdutils.tools.Table*), 26
TableOfContents (*class in mdutils.tools.TableOfContents*), 27
text_color() (*mdutils.tools.TextUtils.TextUtils static method*), 28
text_external_link() (*mdutils.tools.TextUtils.TextUtils static method*), 28
text_format() (*mdutils.tools.TextUtils.TextUtils static method*), 29
TextUtils (*class in mdutils.tools.TextUtils*), 27
TITLE (*mdutils.tools.Header.AtxHeaderLevel attribute*), 22
TITLE (*mdutils.tools.Header.SetextHeaderLevel attribute*), 24

W

write() (*mdutils.mdutils.MdUtils method*), 35