mdutils Documentation

Release 1.0

Dídac Coll

Apr 20, 2020

Contents:

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.

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

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

mdFile = MdUtils(file_name='Example', 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("It is the last command that has to be called.")
mdFile.insert_code("import Mdutils\n"
                   "\n"
                   "\n"
                   "mdFile = MdUtils(file_name=\'Example\',title=\'This is a 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

#␣
↪**********************************************************************************************
# ****************************************** Create a table of contents␣
↪******************************************
```

(continues on next page)

(continued from previous page)

```
#␣
↪**********************************************************************************************************
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`` "
                   "method.\")", language='python')
```

(continues on next page)

(continued from previous page)

```
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␣
↪**********************************************
#␣
↪**************************************************************************************************
mdFile.new_header(2, "Create a Table")
```

(continues on next page)

(continued from previous page)

```
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 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=''*)
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*)
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*)
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*)
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*)
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.tools module

class mdutils.tools.tools.**Header**
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 Hanchors.
- Auto generate a table of contents.
- Create Tables.
- **Bold**, *italics*, `inline_code` text converters.
- Align text to center.
- Add color to text.

static atx_level_1(*title*)
Return a atx level 1 header.

Parameters **title** (*str*) – text title.

Returns a header title of form: `'\n#' + title + '\n'`

Return type str

static atx_level_2(*title*)
Return a atx level 2 header.

Parameters **title** (*str*) – text title.

Returns a header title of form: `'\n##' + title + '\n'`

Return type str

static atx_level_3(*title*)
Return a atx level 3 header.

Parameters **title** (*str*) – text title.

Returns a header title of form: `'\n###' + title + '\n'`

Return type str

static atx_level_4(*title*)
Return a atx level 4 header.

Parameters **title** (*str*) – text title.

Returns a header title of form: `'\n####' + title + '\n'`

Return type str

static atx_level_5(*title*)
Return a atx level 5 header.

Parameters **title** (*str*) – text title.

Returns a header title of form: `'\n#####' + title + '\n'`

Return type str

static atx_level_6(*title*)
Return a atx level 6 header.

Parameters **title** (*str*) – text title.

Returns a header title of form: `'\n######' + title + '\n'`

Return type str

choose_header(*level*, *title*, *style='atx'*)
This method choose the style and the header level.

Examples

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

```
>>> createHeaders.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.

Returns

static header_anchor(*text*, *link=''*)
Creates an internal link of a defined Header level 1 or level 2 in the markdown file.

Giving a text string an text link you can create an internal link of already existing header. If the `link` string does not contain '#', it will creates an automatic link of the type `#title-1`.

Parameters

- **text** (*str*) – it is the text that will be displayed.
- **link** (*str*) – it is the internal link.

Returns '[text](#link)'

Return type string

Example: [Title 1](#title-1)

static setext_level_1(*title*)
Return a setext level 1 header.

Parameters **title** (*str*) – text title.

Returns a header titlte of form: '\n' + title +'\n==========\n'.

Return type str

static setext_level_2(*title*)
Return a setext level 1 header.

Parameters **title** (*str*) – text title.

Returns a header titlte of form: '\n' + title +'\n----------\n'.

Return type str

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

create_table(*columns*, *rows*, *text*, *text_align='center'*)
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*) – text align argument. Values available are: 'right', 'center', and 'left'.

Returns a markdown table.

Return type str

Example

```
>>> from mdutils.tools.tools 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

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

create_table_of_contents(*array_of_title_contents*, *depth=1*)
This method can create a table of contents using an array of the different titles. The deep can be changed. Only accepts 1 or 2.

Parameters

- **array_of_title_contents** (*list*) – a string list with the different headers.
- **depth** (*int*) – allows to include Headers 1 and 2 or only Headers of level 1. Possibly values 1 or 2.

Returns return a string ready to be written to a Markdown file.

Return type str

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

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

static bold(*text*)
Bold text converter.

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

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

Return type str

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

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

Returns a string like this exampple: ' text '

static inline_code(*text*)
Inline code text converter.

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

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

Return type str

static insert_code(*code*, *language=''*)
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*)
Italics text converter.

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

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

Return type str

static text_color(*text*, *color='black'*)
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*, *link=''*)
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

text_format(*text*, *bold_italics_code=''*, *color='black'*, *align=''*)
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.tools import TextUtils
>>> textUtils = 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*, *title=''*, *author=''*)

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*)

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()

It creates a new Markdown file. :return: return an instance of a MarkDownFile.

insert_code(*code*, *language=''*)

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_header(*level*, *title*, *style='atx'*, *add_table_of_contents='y'*)

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'.

The example below consist in creating two types Headers examples:

Example

```
>>> 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'
```

new_line(*text=''*, *bold_italics_code=''*, *color='black'*, *align=''*)

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*) – bold_italics_code: 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 `'\n' + text`. Not necessary to take it, if only has to be written to the file.

Return type str

new_paragraph(*text=''*, *bold_italics_code=''*, *color='black'*, *align=''*)

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*) – bold_italics_code: 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 `'\n\n' + text`. Not necessary to take it, if only has to be written to the file.

Return type str

new_table(*columns*, *rows*, *text*, *text_align='center'*, *marker=''*)

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.tools.tools 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().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='Table of contents'*, *depth=1*, *marker=''*)

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 if headers of level 1 and 2 or only level 1 have to be placed on 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 Headers 1 and 2 or only Headers of level 1. Possible values 1 or 2.
- **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*, *marker*)
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*)
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=''*, *bold_italics_code=''*, *color='black'*, *align=''*, *marker=''*)
Write text in `file_Data_text` string.

Parameters

- **text** (*str*) – a text a string.
- **bold_italics_code** (*str*) – bold_italics_code: 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.
- **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

Index

S

T

W