Markdown Syntax Cheat Sheet

Introduction

Markdown is a lightweight markup language with plain text formatting syntax. It is designed so that it can be converted to HTML and many other formats using a tool by the same name. Markdown is often used to format readme files, for writing messages in online discussion forums, and to create rich text using a plain text editor.

Writing Markdown for Python Projects on GitHub

Markdown is an essential tool for documenting Python projects on GitHub. It allows you to create well-structured and easily readable documentation, which is crucial for collaboration and sharing your work with others. This guide will help you understand how to use Markdown to create comprehensive documentation for your Python projects.

When documenting your Python project, you might want to include the following sections:

1. **Project Title and Description**: Provide a clear and concise title and a brief description of your project.
2. **Installation Instructions**: Explain how to install and set up your project.
3. **Usage**: Provide examples of how to use your project.
4. **Contributing**: Outline guidelines for contributing to your project.
5. **License**: Specify the license under which your project is distributed.

Character Formatting

- **Bold**: `**text**` or `__text__`
- *Italic*: `*text*` or `_text_`
- ~~Strikethrough~~: `~~text~~`

Headings

- `# Heading 1`
- `## Heading 2`
- `### Heading 3`
- `#### Heading 4`
- `##### Heading 5`
- `###### Heading 6`

Paragraphs and Line Breaks

- Paragraphs are separated by a blank line.
- Line break: End a line with two spaces or use `<br>`.

Thematic Break

- `---`

- `***`
- `___`

Block Quotes

- `> This is a block quote.`

Inline Code

- `` `code` ``

Indented Code Block

~~~
```
This is an indented code block.
```
~~~

Fenced Code Block

~~~
```language
// This is a fenced code block with syntax highlighting.
```
~~~

Lists

- **Unordered List**:
  - `- Item`
  - `* Item`
  - `+ Item`
- **Ordered List**:
  1. `1. Item`
  2. `2. Item`

Tables

```
| Header 1 | Header 2 |
|----------|----------|
| Cell 1   | Cell 2   |
| Cell 3   | Cell 4   |
```

Links

- `[Link Text](URL)`

Autolinks

- `<http://example.com>`

Images

- `![Alt Text](URL)`

Character Formatting

Character formatting in Markdown includes bold, italic, and strikethrough text.

- **Bold**: To make text bold, wrap it with `**` or `__`.
  Example: `**bold text**` or `__bold text__` will render as **bold text**.
- *Italic*: To italicize text, wrap it with `*` or `_`.
  Example: `*italic text*` or `_italic text_` will render as *italic text*.
- ~~Strikethrough~~: To strike through text, wrap it with `~~`.
  Example: `~~strikethrough text~~` will render as ~~strikethrough text~~.

Headings

Headings are created using the `#` symbol followed by a space and the heading text. The number of `#` symbols indicates the level of the heading.

- `# Heading 1`
- `## Heading 2`
- `### Heading 3`
- `#### Heading 4`
- `##### Heading 5`
- `###### Heading 6`

Example:

```
# Heading 1
## Heading 2
### Heading 3
```

Paragraphs and Line Breaks

Paragraphs are separated by one or more blank lines. To create a line break within a paragraph, end a line with two spaces or use the `<br>` HTML tag.

Example:

```
This is a paragraph.

This is another paragraph.
```

To create a line break:

```
This is a line with a break.
This is the next line.
```

or

```
This is a line with a break.<br>
This is the next line.
```

Thematic Break

A thematic break (horizontal rule) can be created using three or more hyphens (`---`), asterisks (`***`), or underscores (`___`).

Example:

```
---
***
___
```

Block Quotes

Block quotes are created using the `>` symbol followed by a space.

Example:

```
> This is a block quote.
```

Inline Code

Inline code is created by wrapping text with backticks (`` ` ``).

Example: `` `code` `` will render as `code`.

Indented Code Block

Indented code blocks are created by indenting lines with four spaces or one tab.

Example:

```
    This is an indented code block.
```

Fenced Code Block

Fenced code blocks are created by wrapping code with triple backticks (```` ``` ````). Optionally, you can specify a language for syntax highlighting.

Example: `python def hello_world(): print("Hello, world!")`

Lists

Markdown supports ordered and unordered lists.

- **Unordered List**: Use `-`, `*`, or `+` followed by a space. Example:

  ```
  - Item 1
  - Item 2
  ```

- **Ordered List**: Use numbers followed by a period and a space. Example:

  ```
  1. Item 1
  2. Item 2
  ```

Tables

Tables are created using pipes (`|`) and hyphens (`-`). The first row is the header, and the second row contains hyphens to separate the header from the body.

Example:

```
| Header 1 | Header 2 |
|----------|----------|
| Cell 1   | Cell 2   |
| Cell 3   | Cell 4   |
```

Links

Links are created using square brackets for the link text and parentheses for the URL.

Example:

```
[Link Text](http://example.com)
```

Autolinks

Autolinks are created by wrapping URLs or email addresses with angle brackets.

Example:

```
<http://example.com>
<user@example.com>
```

Images

Images are created using an exclamation mark followed by square brackets for the alt text and parentheses for the URL.

Example:

```
![Alt Text](http://example.com/image.jpg)
```

Python Code Examples in Markdown

When documenting Python projects, including code examples is essential. Here are some examples of how to write Python code in Markdown:

Inline Code

To include a short snippet of Python code within a sentence, use inline code formatting with backticks.

Example:

```
The `print()` function in Python is used to output text to the console.
```

Code Blocks

For longer code examples, use fenced code blocks. Specify the language (e.g., `python`) to enable syntax highlighting.

Example:

```python
def greet(name):
    return f"Hello, {name}!"

print(greet("World"))
```

Including Output

You can also include the expected output of your code by using a combination of code blocks and text.

Example:

```
def add(a, b):
    return a + b

result = add(2, 3)
print(result)
```

Output:

```
5
```

Documenting Functions

When documenting functions, provide a description, the function signature, and an example of how to use the function.

Example:

```
def multiply(a, b):
    """
    Multiply two numbers and return the result.

    Parameters:
    a (int): The first number.
    b (int): The second number.

    Returns:
    int: The product of the two numbers.
    """
    return a * b

# Example usage
result = multiply(4, 5)
print(result)  # Output: 20
```

Using Markdown for Jupyter Notebooks

Markdown is also used in Jupyter Notebooks to provide explanations and context for your code. Here is an example of how to combine Markdown and Python code in a Jupyter Notebook:

Markdown Cell:

```
# Data Analysis with Pandas

In this notebook, we will perform data analysis using the Pandas library.
```

Code Cell:

```python
import pandas as pd

# Load the dataset
df = pd.read_csv('data.csv')

# Display the first few rows of the dataframe
df.head()
```

By following these examples, you can effectively document your Python code and make it easier for others to understand and use your projects. By following this guide, you will be able to create professional and informative documentation for your Python projects on GitHub.