Rhetorical Analysis of Code Documentation in Computer Programming of CPython

Zhi Cheng, Northeastern University

Link ： https://github.com/python/cpython?ref=hackernoon.com

1. Introduction

1.1 Motivation

The reason why I chose code documentation to analyze is that there is a tight relationship between me and such kind of genre. Currently, I am a student majoring in Computer and Information Science. When I went through others' codes, most people would write down few words as the documentation since there was not a clear and general requirement for our code illustrations rather than the code. As a result, I would be confused about certain part and had to take a lot of time to investigate the function of each line. If someone employs advanced techniques, I will have to search for them and learn their usages. Motivated by this unsatisfying experience with code documentation, I would like to utilize an example of a professional project to explore how they have written the code documentation to help me be a more professional coder. Not only does a clear code statement help the audience to understand my code, but it would be also much useful when I get back and work on my code in the future.

1.2 About Code Documentation

Code documentation refers to "the collection of easy to understand images and written descriptions that explain what a codebase does and how it can be used" (Frank, 2018). With the increasing importance of Computer programming, the significance of the code documentation increases as well. As for the increasing importance of computer programming in people's life, people could use programmed application or website to communicate, shop, order food ... As a result, more and more people choose to hunt a relevant job, and the way they communicate is code documentation.

The code documentation is very different from other types of engineering writing's genres like laboratory and field report. To be more specific, while laboratory and field report have the common parts of contents like introduction, background, procedure, conclusion, and citation, there is no official requirements for the doc of code. A clear and logical documentation leads to simpleness of understanding the writer's content and purpose, while a vague doc leads to the under-specification and waste of audiences' time.

In this paper, I will analyze the official GitHub repository of CPython, which was *GitHub's Top 100 Most Valuable Repositories Out of 96 Million* (Gaviar, 2019), to analyze how the official writers documented their code. To be more specific about this project, even though it is called CPython, it is the source codes of the Python of version 3.10.0 alpha. As for Python, as people known, it is one of the most popular language in programming that people use today, and with higher edition the Python is, the more libraries and few bugs there will be. In addition to the popularity of Python, when talking with computer programming, most coders are familiar with GitHub and it is very popular is the platform that "Millions of developers and companies build, ship, and maintain their software on GitHub—the largest and most advanced development platform in the world" (Github). I will analyze the doc from README, which is the unique kind of code doc on GitHub, and in-line code documentation, which are the two main categories of documentation of code.

2. README

2.1 Background

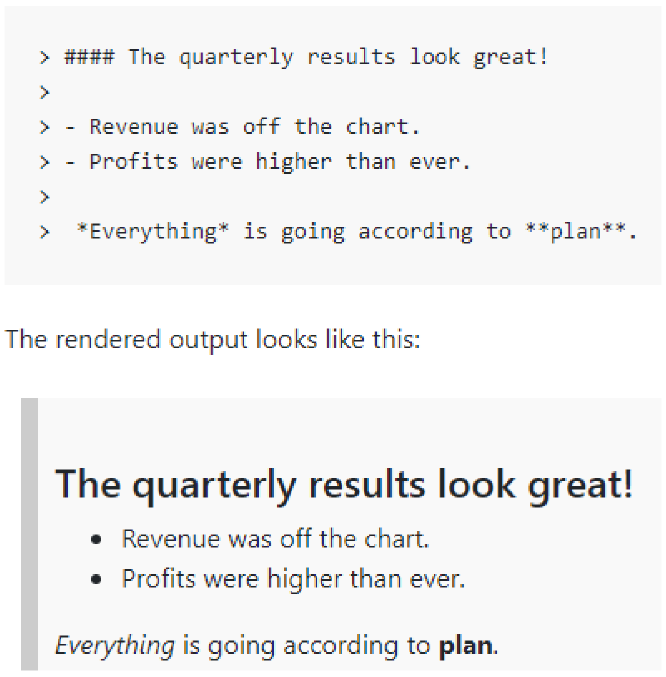
Before talking about the other aspects of the README file, I will introduce this special kind of genre of GitHub at first. The README file will be automatically created when creating a repository on GitHub and it always positions at the bottom of a GitHub repository. According to GitHub, "You can add a README file to a repository to communicate important information about your project. A README, communicates expectations for your project and helps you manage contributions." In other words, the README file has always been utilized to express the abstract or the general idea about the current repository.

2.2 Format and Content

The format for the README file is to use Markdown format and "Markdown is a way to write content for the web" (MarkownTutorial). With the Markdown format, it would be neatly presented in front of the readers.

Figure 1

The use of Markdown Format



```
> #### The quarterly results look great!
>
> - Revenue was off the chart.
> - Profits were higher than ever.
>
>  *Everything* is going according to **plan**.
```

The rendered output looks like this:

> #### The quarterly results look great!
>
> - Revenue was off the chart.
> - Profits were higher than ever.
>
> *Everything* is going according to **plan**.

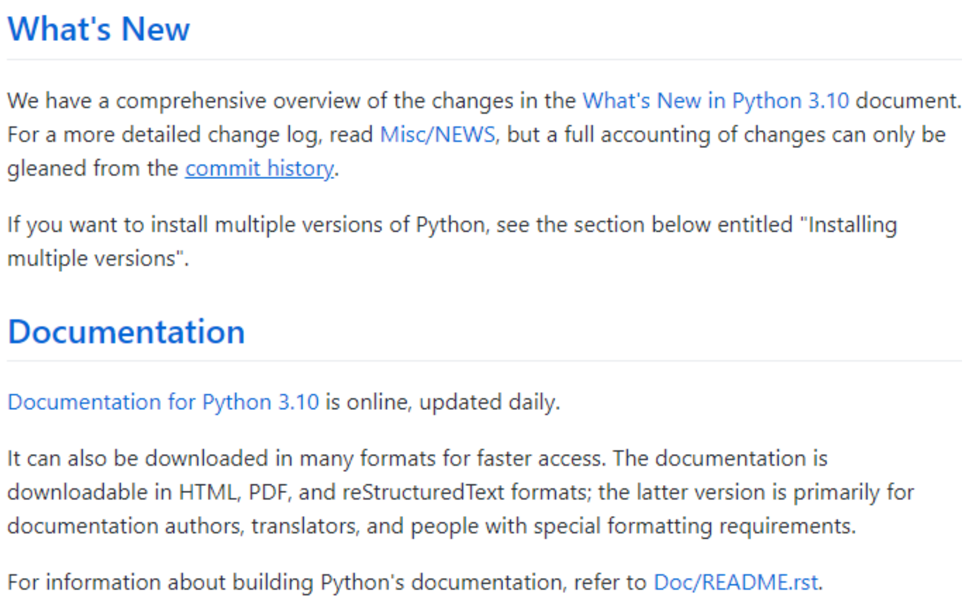
Note. This screenshot is taken from Markdownguide for the use of Markdown Format.

Taking Figure 1, from the Markdownguide, as an example, rather than just write the words down, the Markdown allow people to build up and distinguish different parts. Based on the format of headings, unordered lists, blockquotes, bold and italic …, it clearly shows the information in order and the important part will be recognized by the readers.

In the CPython, the writers resorted to Markdown Format in the ReadMe file.

Figure 2

Screenshot of the README Part of CPython



What's New

We have a comprehensive overview of the changes in the What's New in Python 3.10 document. For a more detailed change log, read Misc/NEWS, but a full accounting of changes can only be gleaned from the commit history.

If you want to install multiple versions of Python, see the section below entitled "Installing multiple versions".

Documentation

Documentation for Python 3.10 is online, updated daily.

It can also be downloaded in many formats for faster access. The documentation is downloadable in HTML, PDF, and reStructuredText formats; the latter version is primarily for documentation authors, translators, and people with special formatting requirements.

For information about building Python's documentation, refer to Doc/README.rst.

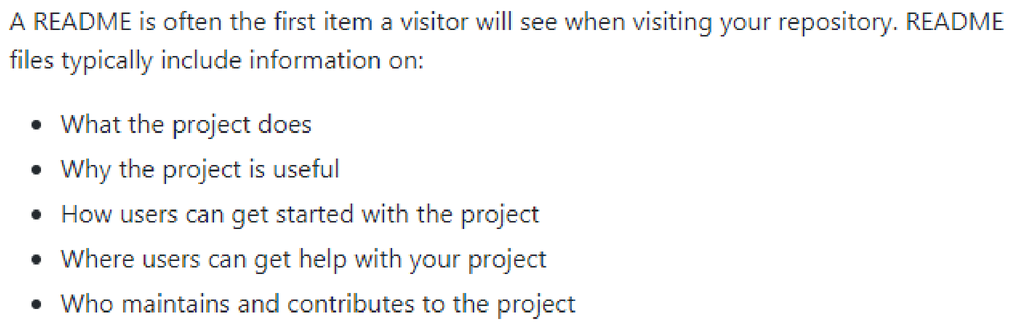
Note. This screenshot is taken from the README File of CPython.

According to the Figure 2, they used such format to create subtitle for each part of information and used blue to color the words equipped with super link. It is easy for readers to follow the writers' step.

And for a general list of contents that should be covered in README, it has been suggested from GitHub:

Figure 3

README suggestions



A README is often the first item a visitor will see when visiting your repository. README files typically include information on:

- What the project does
- Why the project is useful
- How users can get started with the project
- Where users can get help with your project
- Who maintains and contributes to the project

Note. This screenshot is taken from GitHub on the suggestions for the README file.

All the suggestions GitHub offered are aimed to offer the audience with the general information of the current repository. In the CPython, the writers had used the Markdown format and they fulfilled the advice from GitHub on context. However, since this is a project from Python Software Foundation, the writers replied rather the

"Copyright and License Information" than the specific names of contributors to the question "Who maintains and contributes to the project".

2.3 **Audience**

Most of the audiences, who read the README file, would be people who familiar with computer programming since this special kind of genre only show up in the platform of the GitHub. Usually, people with knowledge of programming have eager to access to the GitHub.

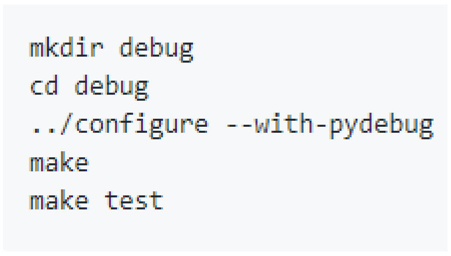
The audiences for the README file of CPython are all programmers, even novices. In addition to most of the audiences, who are familiar with coding, there will be people unfamiliar with this coding. Since this is the official source code from the Python Foundation, people who are not familiar with code may also come to this repository to explore the source code of the Python language. In other words, some people may be just curious about the coding and access to this repository. Regarding this, the audience of the CPython are both the novices and professionals.

2.4 Language

Since the audiences in CPython are considered as the public, the writers used most simple words with few very precises word of programming. In the *Writing in Engineering A Brief Guide*, Irish (2015) claimed that "even if readers lack a background in a particular area, your job is to ensure their understanding" (p. 9). Since the audience is the public, it means people from all over the world will have access to it. Therefore, simple words will become the best choice, while very complicated language will impose a language burden on readers' shoulder. Even though, in addition to most simple words, there are few precise words of programming, the writers used simple words to show the purposes and steps. For example, the words as shown in Figure 4, without specification, may be very precise since they are programming:

Figure 4

Screenshot from README of CPython



```
mkdir debug
cd debug
../configure --with-pydebug
make
make test
```

Note. This screenshot is taken in the "Build Instructions" part from the README file of CPython.

However, the writers use simple words to illustrate them "If you wish, you can create a subdirectory and invoke configure from there" and "This will fail if you also built at the top-level directory. You should do a make clean at the top-level first". As a result, the audience would like to apply simple words to convey the instructions for installation so that the audience could know the process without the knowledge of programming. In other words, the audience just need to understand the simple language and paste the programming the writers offered, then they are able to complete the installation.

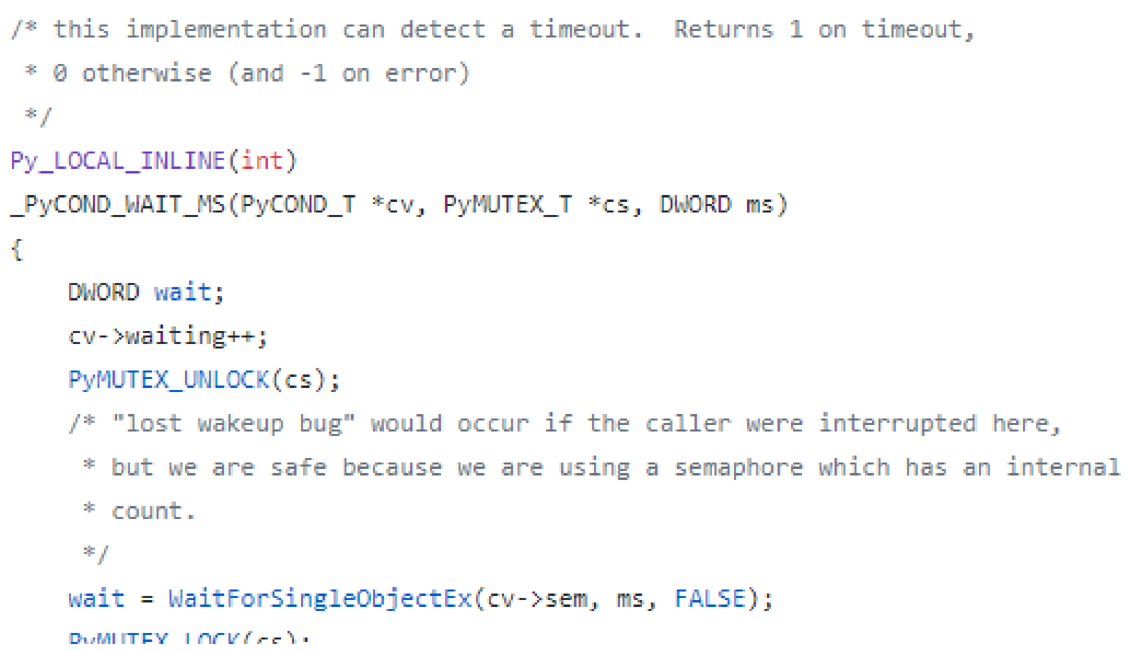
3. In-line code documentation

3.1 Background

The in-line code documentation above the code is also considered as code documentation and this kind of genre is also very different to others. It will accompany with the coding written in different languages, such as c, Java, Python, and it is always placed before an interface,a class, a method... Taking the file condvar.h of CPython as an example, there is one method called Py_LOCAL_INLINE:

Figure 5

Screenshot of condvar.h file



```
/* this implementation can detect a timeout.  Returns 1 on timeout,
 * 0 otherwise (and -1 on error)
 */
Py_LOCAL_INLINE(int)
_PyCOND_WAIT_MS(PyCOND_T *cv, PyMUTEX_T *cs, DWORD ms)
{
    DWORD wait;
    cv->waiting++;
    PyMUTEX_UNLOCK(cs);
    /* "lost wakeup bug" would occur if the caller were interrupted here,
     * but we are safe because we are using a semaphore which has an internal
     * count.
     */
    wait = WaitForSingleObjectEx(cv->sem, ms, FALSE);
    PyMUTEX_LOCK(cs);
```

Note. This screenshot is taken from the condvar.h file of CPython.

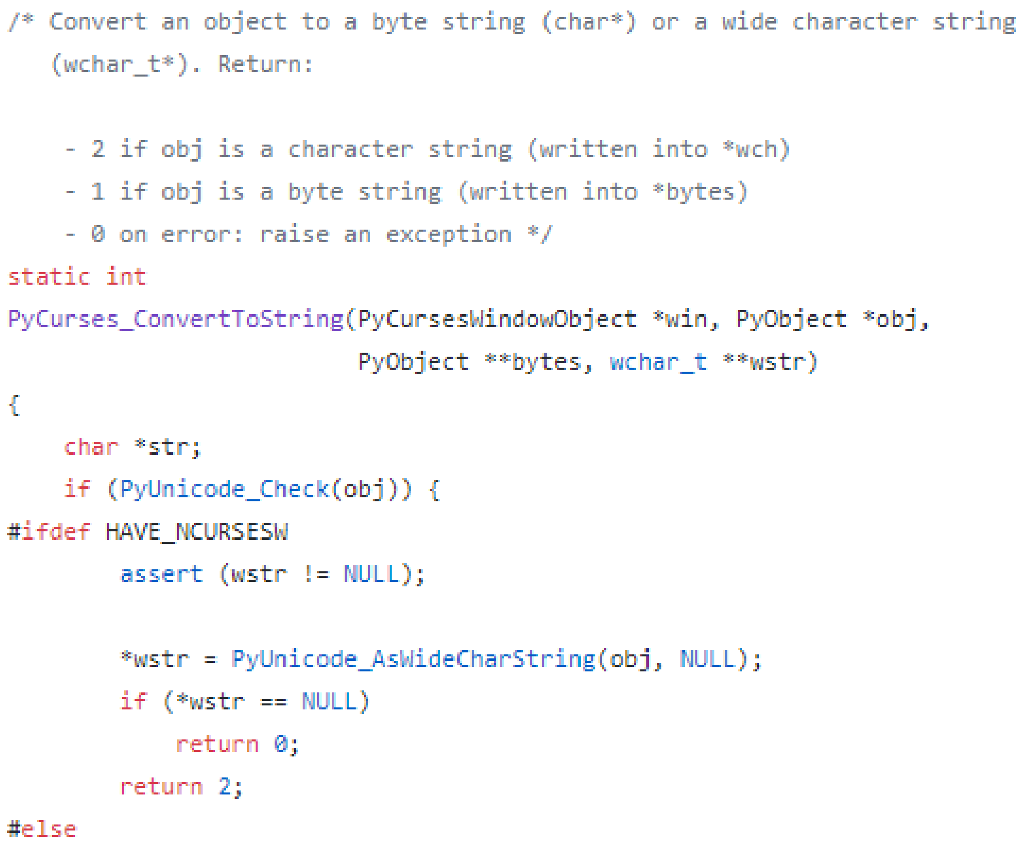
The contents covered by "/*" and "*/" are considered such kind of code documentation.

3.2 Format and Content

For the Format, the in-line code doc is supposed to be covered by "/*" and "*/" and be placed in front of the interface, class, method… As for the content, since the purpose of the in-text illustration is to let reviewers understand the functionality of coding, the basic requirement is to explain the purpose of the code. Also, it would show how to handle the error.

Figure 6

Screenshot of PyCurses_ConvertToString Method



```
/* Convert an object to a byte string (char*) or a wide character string
   (wchar_t*). Return:

    - 2 if obj is a character string (written into *wch)
    - 1 if obj is a byte string (written into *bytes)
    - 0 on error: raise an exception */
static int
PyCurses_ConvertToString(PyCursesWindowObject *win, PyObject *obj,
                         PyObject **bytes, wchar_t **wstr)
{
    char *str;
    if (PyUnicode_Check(obj)) {
#ifdef HAVE_NCURSESW
        assert (wstr != NULL);

        *wstr = PyUnicode_AsWideCharString(obj, NULL);
        if (*wstr == NULL)
            return 0;
        return 2;
#else
```

Note. This screenshot is taken of PyCurses_ConvertToString Method of CPython.

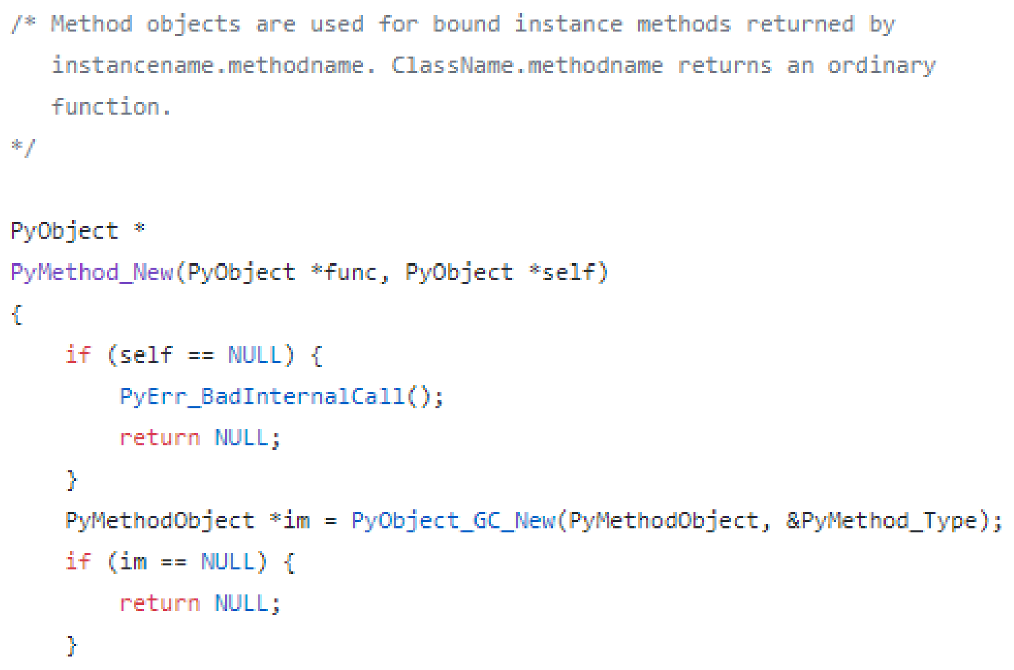
In the Figure 6, the writers clearly showed the purpose of the methods. PyCurses_ConvertToString aims to convert an object to a byte string or a wide character string and return a number to represent the result.

3.3 Language and Audience

Since the audience is the people who are familiar with programming, the writers in CPython used the language with programming words. Bellow is an example:

Figure 7

Screenshot of classobject.c file



```c
/* Method objects are used for bound instance methods returned by
   instancename.methodname. ClassName.methodname returns an ordinary
   function.
*/

PyObject *
PyMethod_New(PyObject *func, PyObject *self)
{
    if (self == NULL) {
        PyErr_BadInternalCall();
        return NULL;
    }
    PyMethodObject *im = PyObject_GC_New(PyMethodObject, &PyMethod_Type);
    if (im == NULL) {
        return NULL;
    }
```

Note. This screenshot is taken from the classobject.c file of CPython.

The Figure 7 shows the method, PyMethod_New, which is from the file classobject.c. In the documentation part, the coders used the programming words like “bound instance methods”. According to GeeksforGeeks, “A bound method is the one which is dependent on the instance of the class as the first argument.” and “Instance attributes are those attributes that are not shared by objects”.

With jargons, the language of the in-line illustration would be obscure for the readers. After all, not every people have the knowledge of computer programming. When they go through the doc, due to lack of the expertise, they would not understand what the goal is. Furthermore, the jargons of the illustration are for the only programming language, Python. For me, who is familiar with other languages like Java, c…, I did not know what those jargons’ meanings are.

Since the language, the writers used, was with jargons, the intended audience should be the people who are familiar with the programming knowledge. After all, only people with expertise could understand the jargon, while it would like a sealed book for people without knowledge of programming.

4. Conclusion

There are two major types of code documentation: the README for GitHub and the in-line code documentation, and both of then aims to let the audience understand the project. When using README in the GitHub platform, it should be created once created a repository and in the README file should the coder answer the “what”, “why”, “how”, “where”, and “who” questions as shown in the Figure 3 with the Markdown format. Normally, the audience should be people who have the knowledge of programming and the language could be with coding words. However, if the project is a product or a service and the intended audience is the public, like the CPython, which is the Python product, the language should be simple and not puzzle the audience. As for in-line code illustration, which focuses more on the description of the functionality of following code. It should be covered by “/*” and “*/”, and it need to show the possible errors and how to handle each kind of errors. Rather than the simple words, people are allowed to use jargons of programming words, since the audience is the professional coder.

Personally, I like to express like a summary in the README file and illustrate detailly in the in-line code documentation, since people may focus more on the README file rather than the in-line code documentation to explore the project.

With a clear code documentation, it would save much of the audiences’ time. After reading the doc, the audience will know the general information about the methods, classes or projects instead of reading the code line by line and determine the purpose based on their expertise.

References

Gaviar, A. (2019, May 20). *GitHub's Top 100 Most Valuable Repositories Out of 96 Million*. Hackernoon.Com. https://hackernoon.com/githubs-top-100-most-valuable-repositories-out-of-96-million-bb48caa9eb0b

GeeksforGeeks. (2019, December 11). *Bound methods python*. https://www.geeksforgeeks.org/bound-methods-python/

GeeksforGeeks. (2020, July 3). *Instance method in Python*. https://www.geeksforgeeks.org/instance-method-in-python/

GitHub: Where the world builds software. (n.d.). GitHub. https://github.com/

Irish.(2015). *Writing in Engineering A Brief Guide.*

Markdownguide contributors. *Basic Syntax*. Markdownguide. https://www.markdownguide.org/basic-syntax/

Markdowntutorial contributors. *Markdown*. Markdowntutorial. https://www.markdowntutorial.com/

Meza, F. (2018, February 6). *The Value of Code Documentation*. Olio Apps.

https://www.olioapps.com/blog/the-value-of-code-documentation/