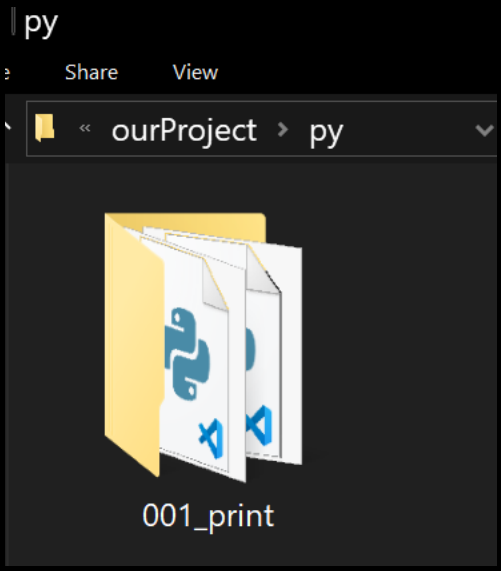
Make a Code Book with Markdown

by

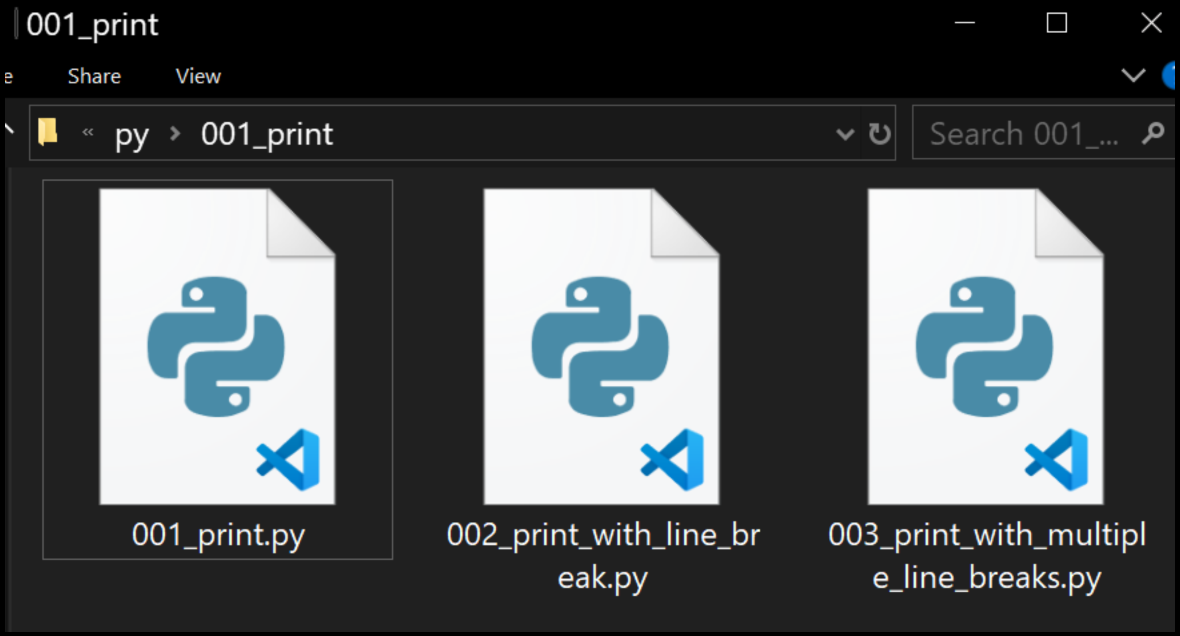
Christopher Andrew Topalian

Dedicated
to
God the Father

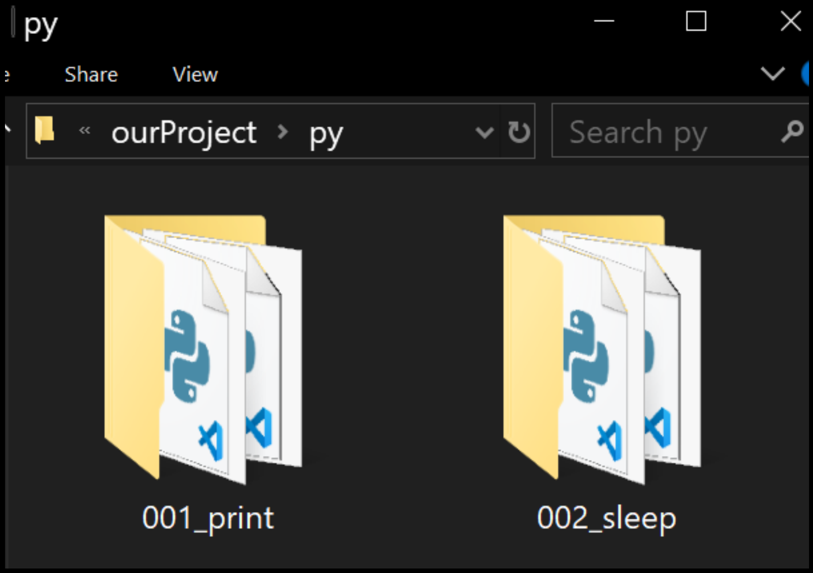
Make a Markdown Book from Our Scripts:



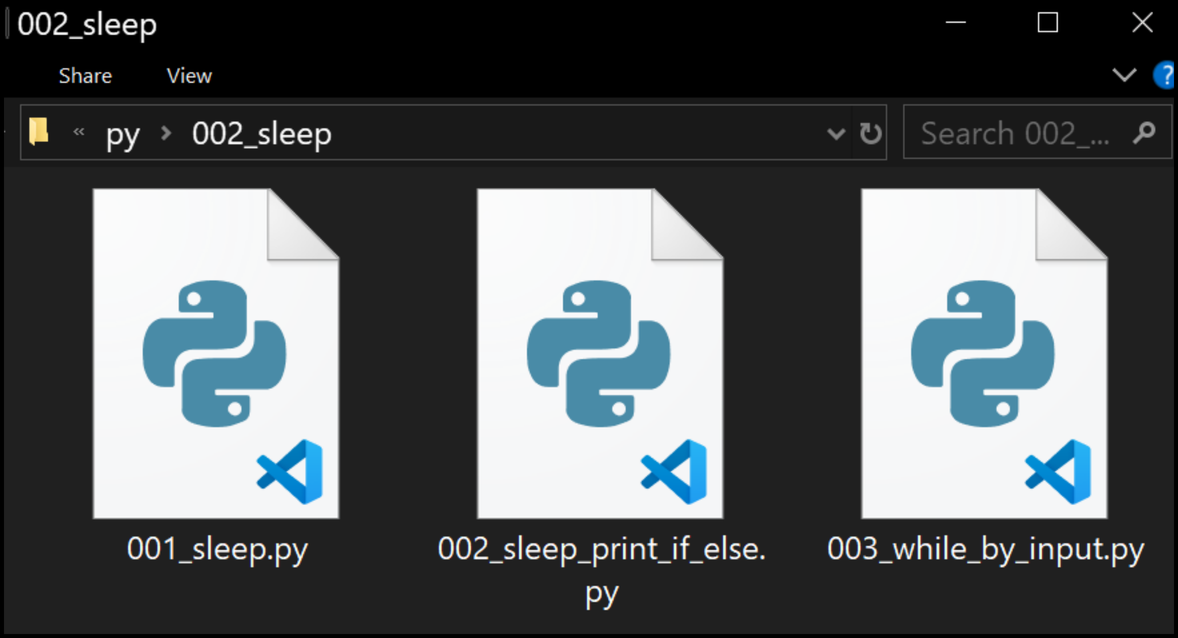
- make a new folder in our py folder
- name the new folder 001_print
- name the 1st script 001_print.py
- name the 2nd script 002_print_with_line_break.py


001_print
Share
View
py › 001_print
Search 001_...
001_print.py
002_print_with_line_break.py
003_print_with_multiple_line_breaks.py

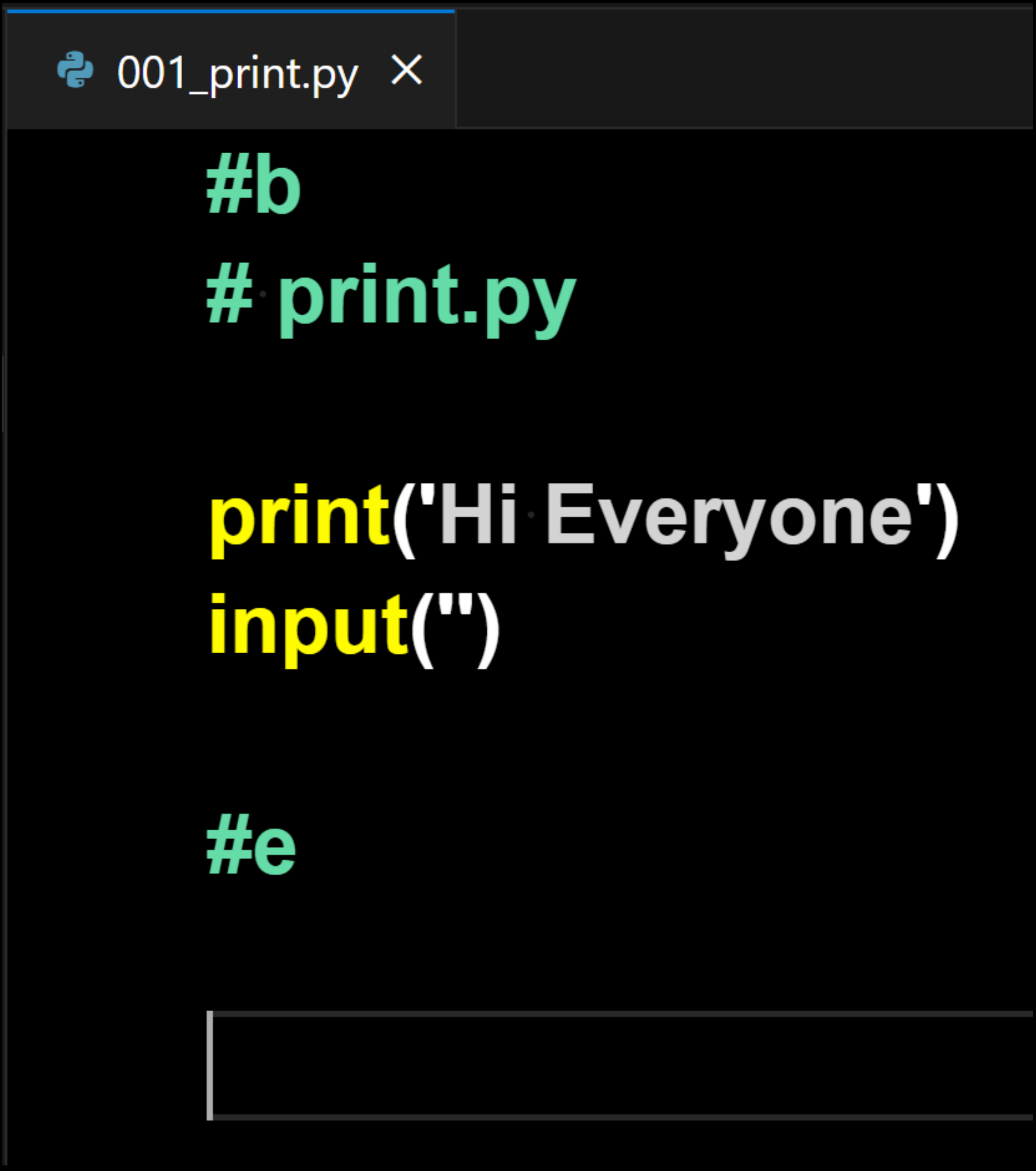
Then we make another new folder in our py folder, for a new category:



- make a new folder in our py folder
- name the new folder 002_sleep
- name the 1st script 001_sleep.py
- name the 2nd script 002_sleep_print_if_else.py

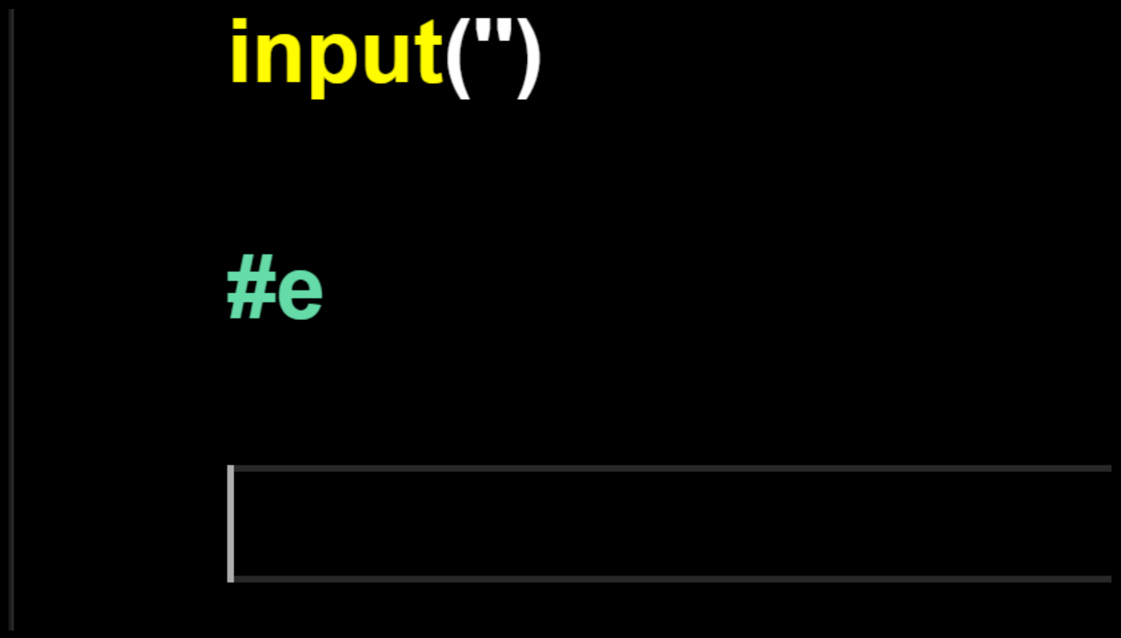


Each of our scripts we must format with a beginning and ending tag
At the top of every script we write #b
At the bottom of every script we write #e
This way we can later find/replace the beginning and ending tags with markdown formatting.



```
#b
# print.py

print('Hi Everyone')
input('')

#e
```

At the end of each of our scripts, we put **two new lines** after the **#e** for easy separation of scripts, for when we combine them.



```
input('')

#e


```

As we can see above, the end of this script we place two new lines.

These two new lines are needed to make a clear separation of scripts for when we combine them later.

Combine all .py scripts into one file:

- open py folder
- put mouse in address bar of py folder
- type, cmd
- press Enter
- In the command prompt we type:

```
for /r "%CD%" %i in (*.py) do
type "%i" >> main.py
```

- press Enter

This combines all .py files into one main.py file, while leaving the original files not changed.

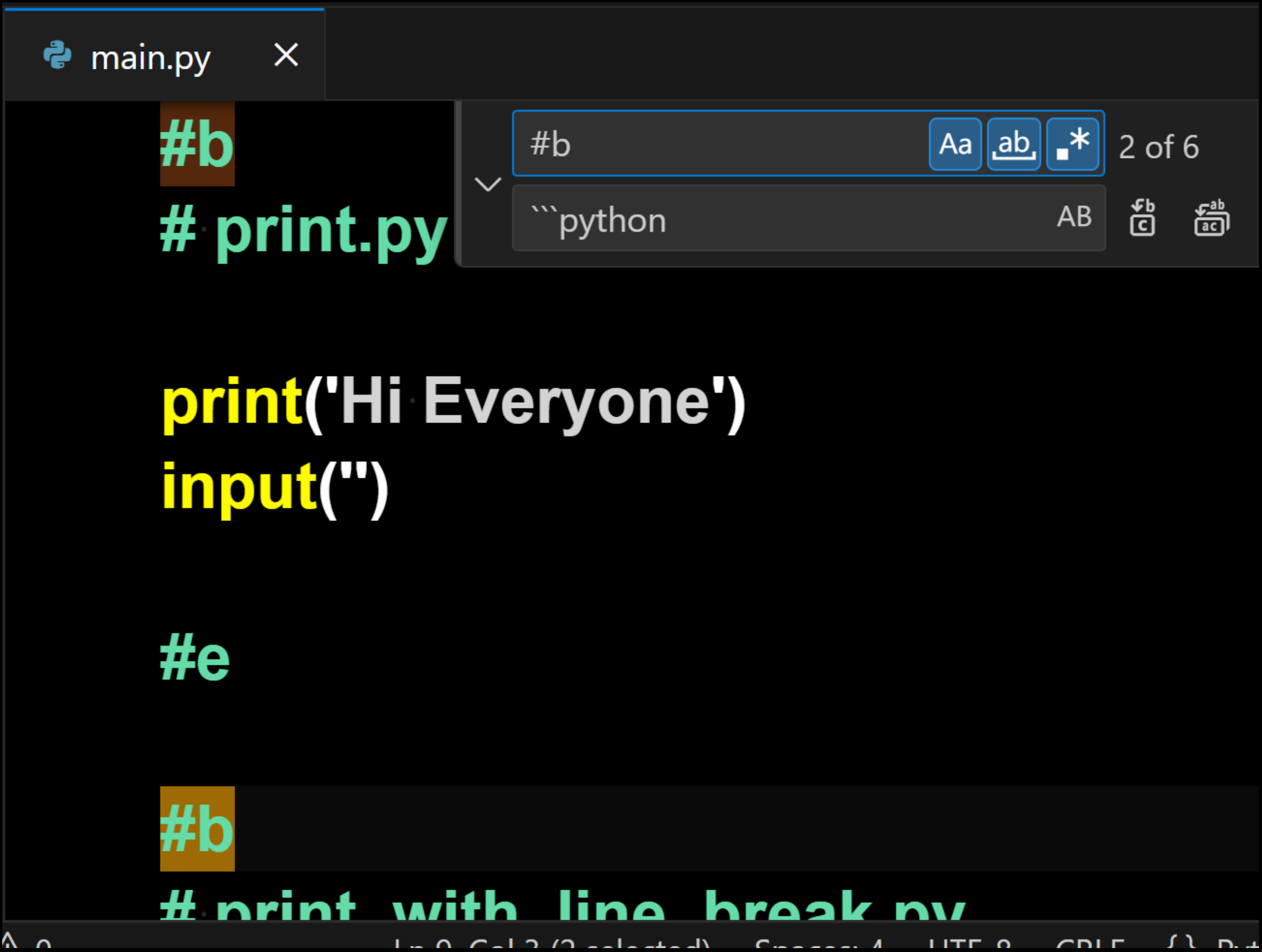
We can choose to use the .bat file provided in the py folder, by double clicking it, which does the same thing and combines all py scripts from all folders in our py folder, while leaving the original files not changed.

We open the newly created main.py file.

We find/replace:

Find: #b

Replace All: ```python

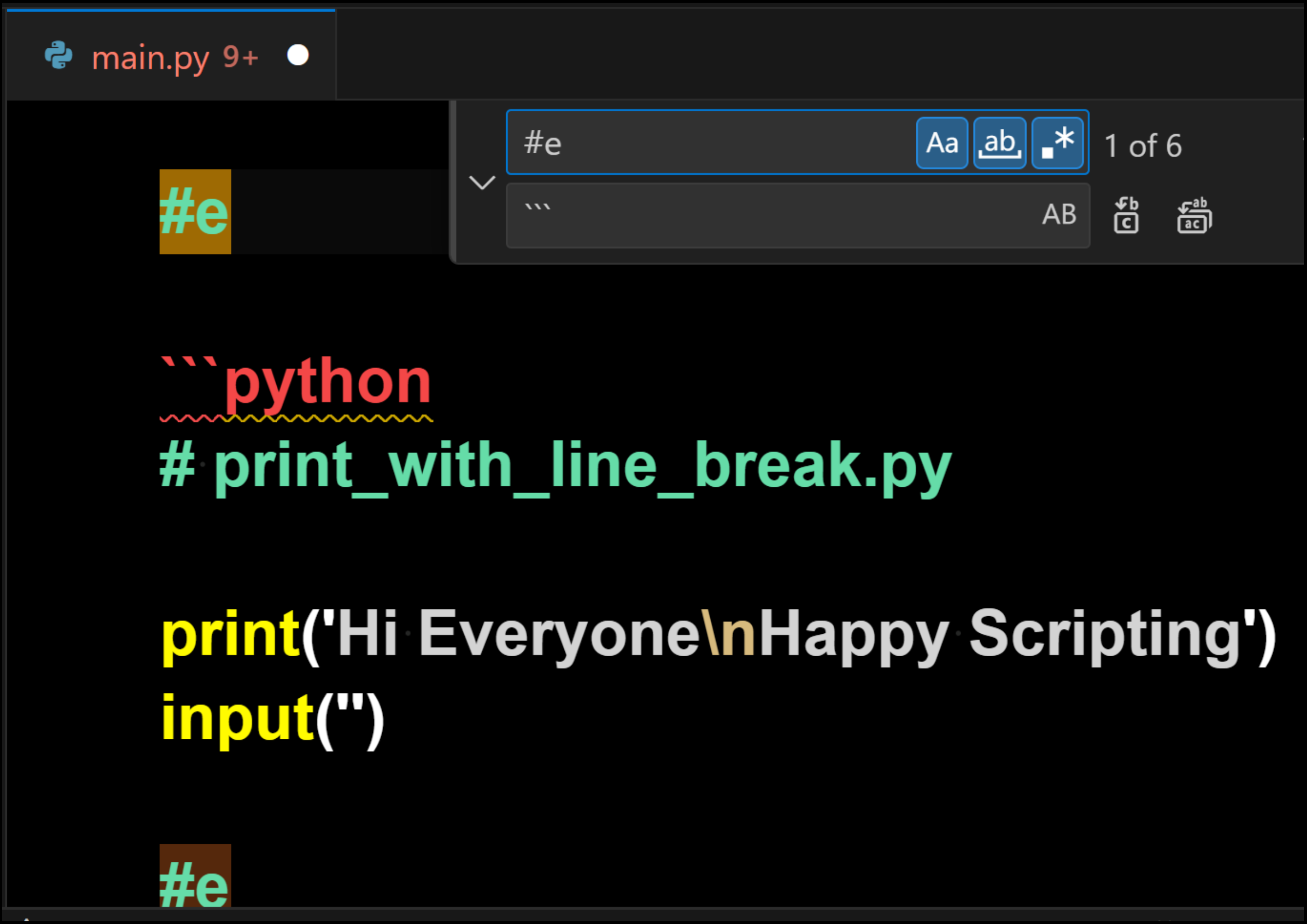


```
#b
# print.py

print('Hi Everyone')
input('')

#e

#b
# print_with_line_break.py
```

We then we find/replace

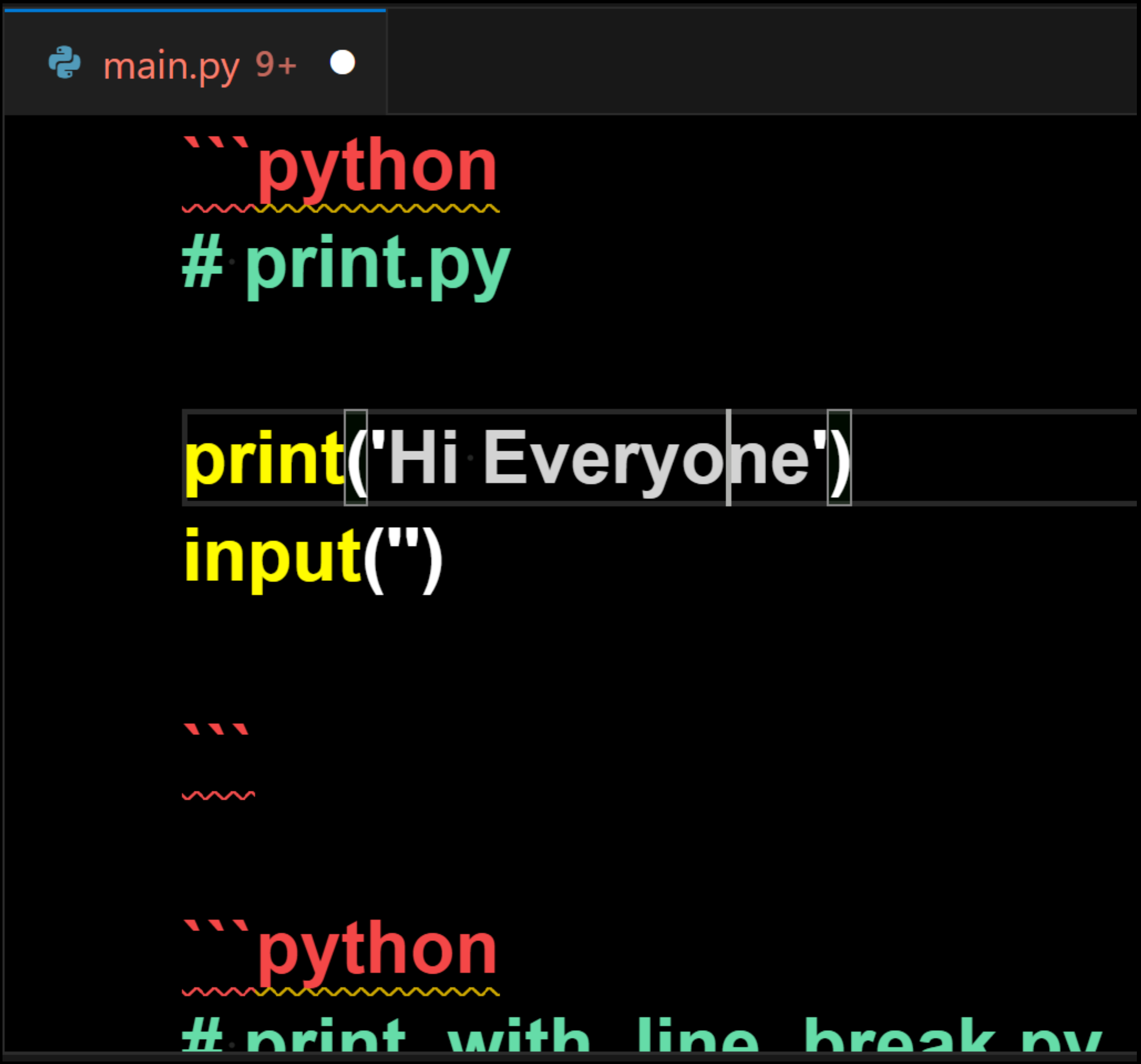
Find: #e

Replace All: ```



~~~
#e

```python
# print_with_line_break.py

print('Hi Everyone\nHappy Scripting')
input('')

#e
~~~

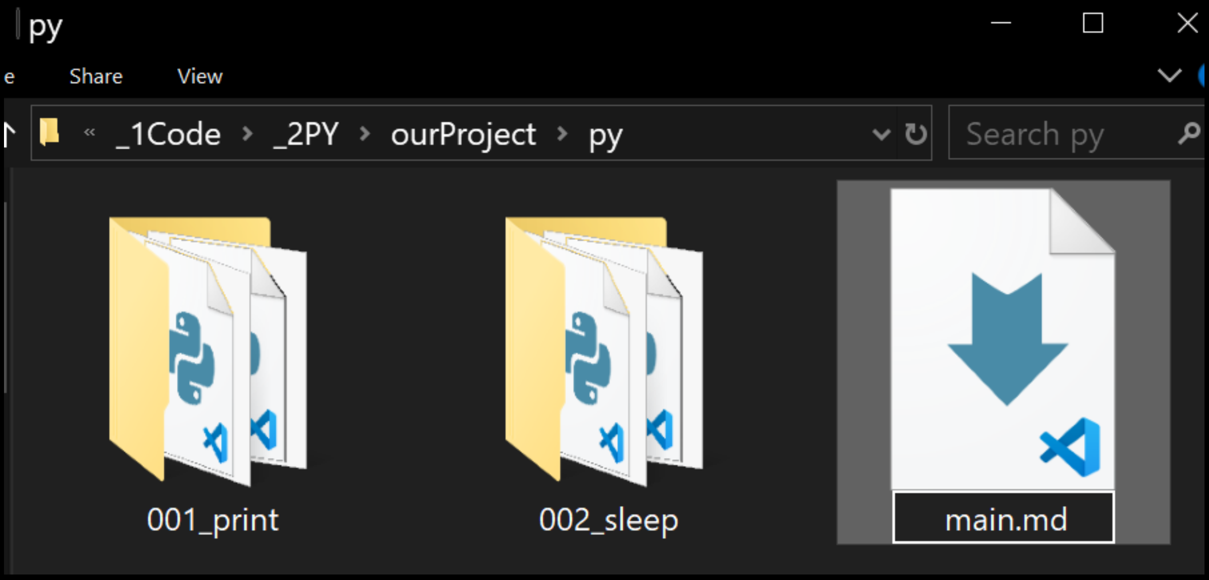
Here is the Formatted Code:



main.py 9+

~~~
```python
# print.py

print('Hi Everyone')
input('')


```


```python
# print_with_line_break.py
~~~

We can now change the file name from main.py to main.md



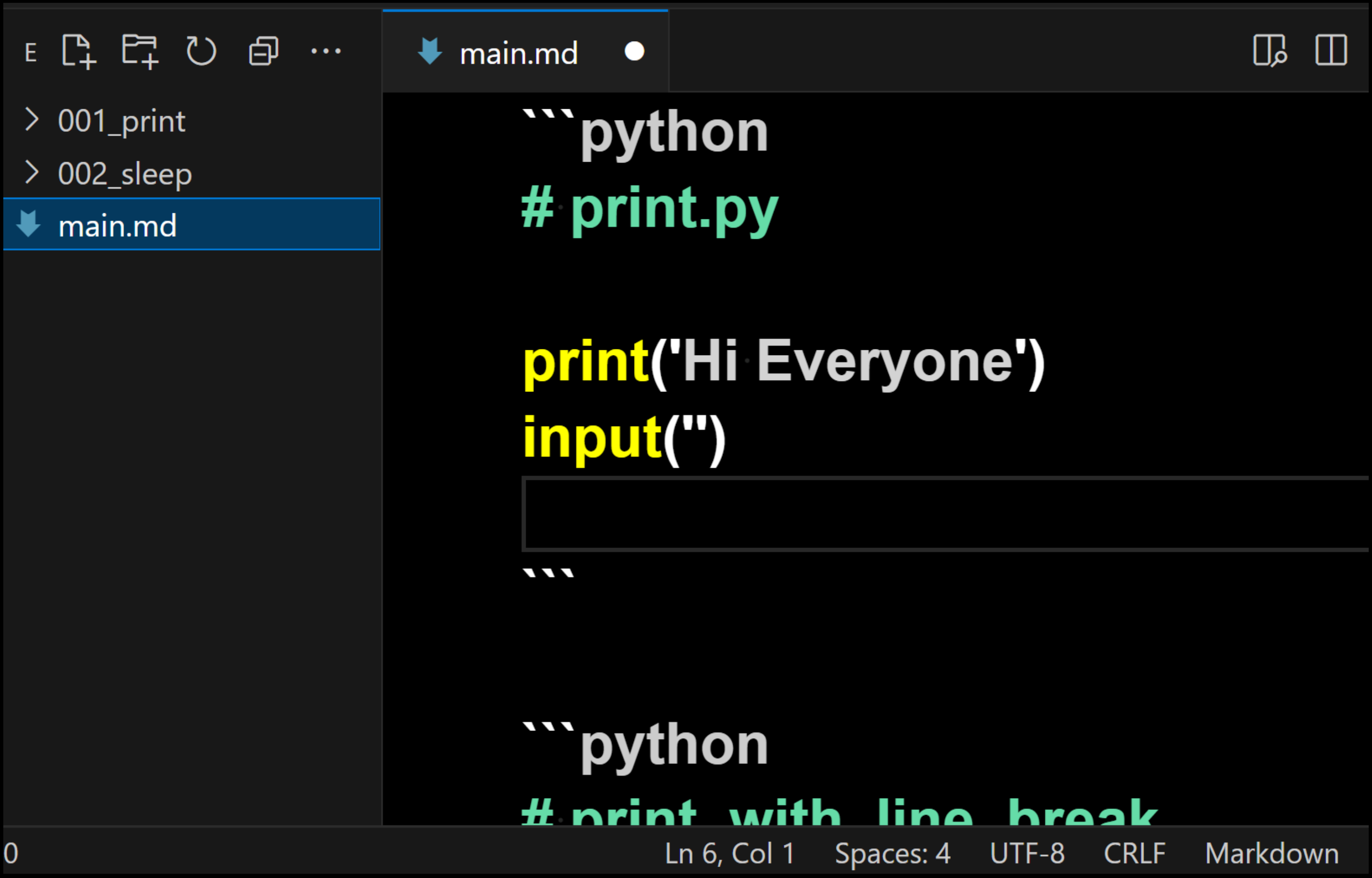
Of course, instead of having to rename our file, we could alternatively, combine our files into an .md file instead of a .py file.

```
for /r "%CD%" %i in (*.py)
do type "%i" >> main.md
```

Either way is fine, just make sure to also find/replace the #b and #e, as we did before.

Now let's open our main.md file and see our rendered markdown in VSCode.

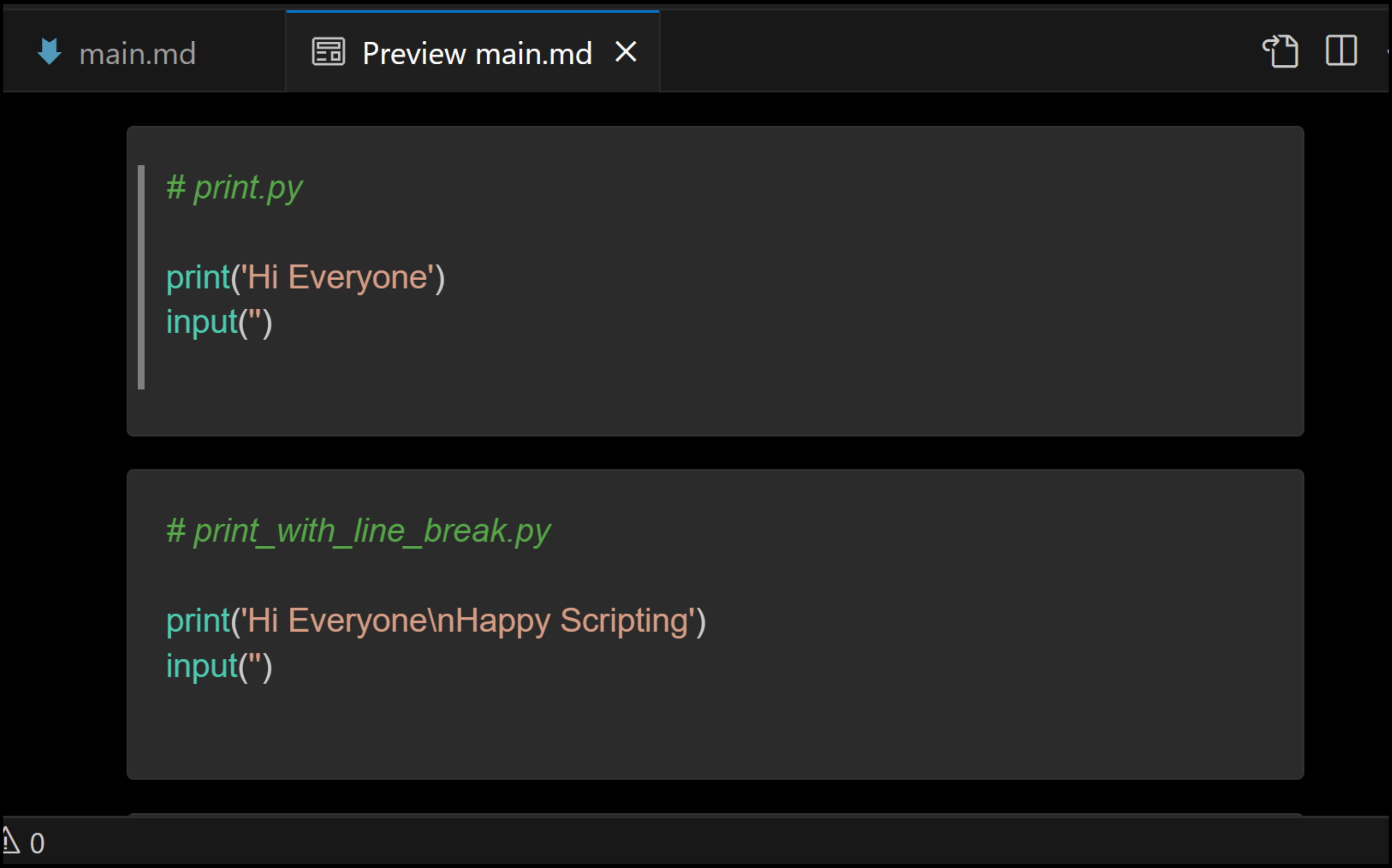
VSCode



~~~
```python
# print.py

print('Hi Everyone')
input('')

```


```python
# print_with_line_break
~~~

Hold Control + Shift + V
Full Screen Render of Markdown File

The rendered markdown file is shown on the next page.



```
# print.py

print('Hi Everyone')
input('')
```

```
# print_with_line_break.py

print('Hi Everyone\nHappy Scripting')
input('')
```

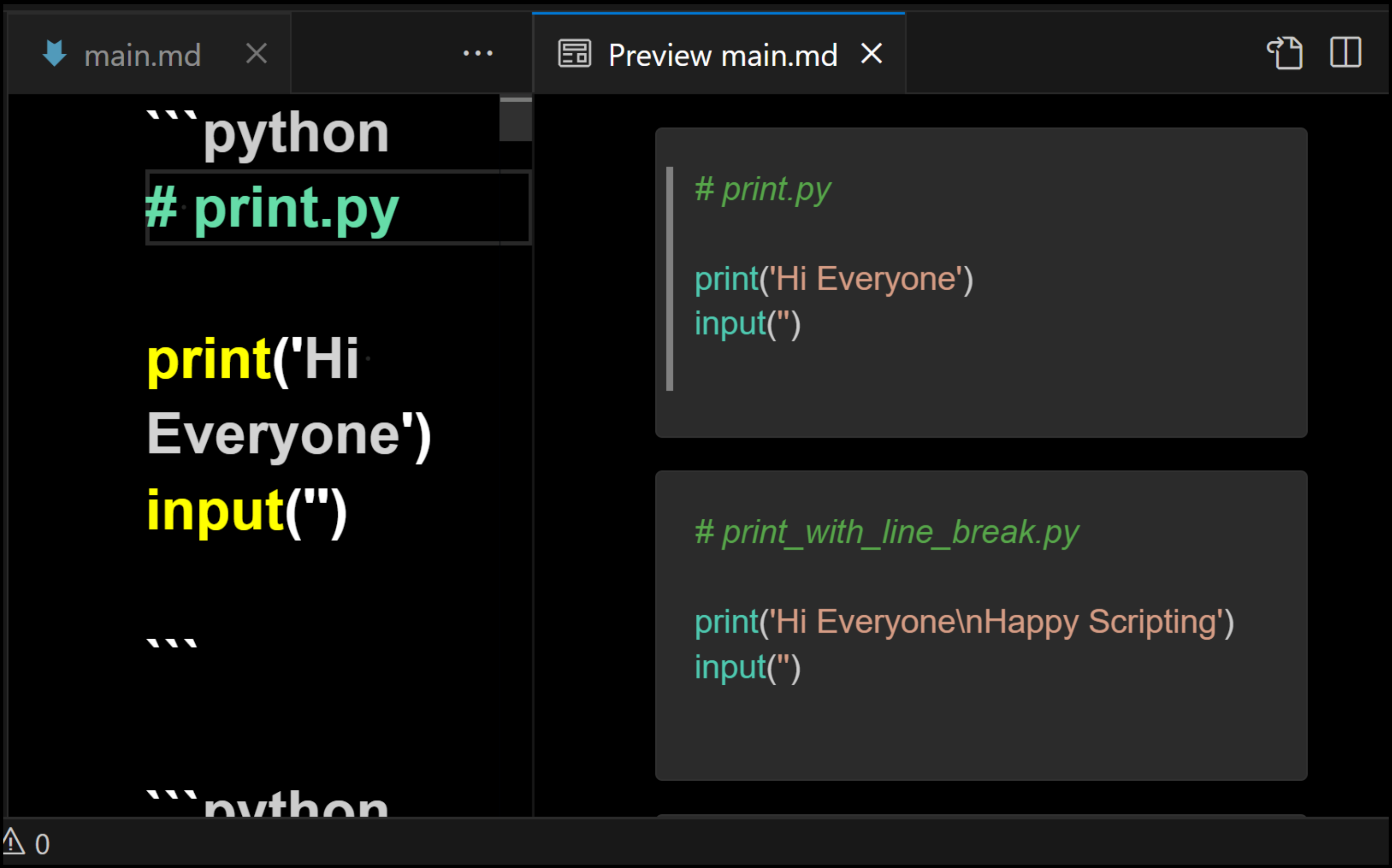
On the top right of the screen we see an icon for Show Source.

Left Click on the Show Source button to return to the original markdown file, or just Left Click on the X of the preview tab to return to editing the original markdown file.

Dual Screen with Preview

We open our main.md file in VSCode and notice at the top right there is a Preview button.

Left click on the Preview button to see a dual screen of the original markdown file and the rendered preview of the markdown file on the right side.



~~~
```python
# print.py

print('Hi Everyone')
input('')

~~~

```python
```

```
# print.py

print('Hi Everyone')
input('')
```

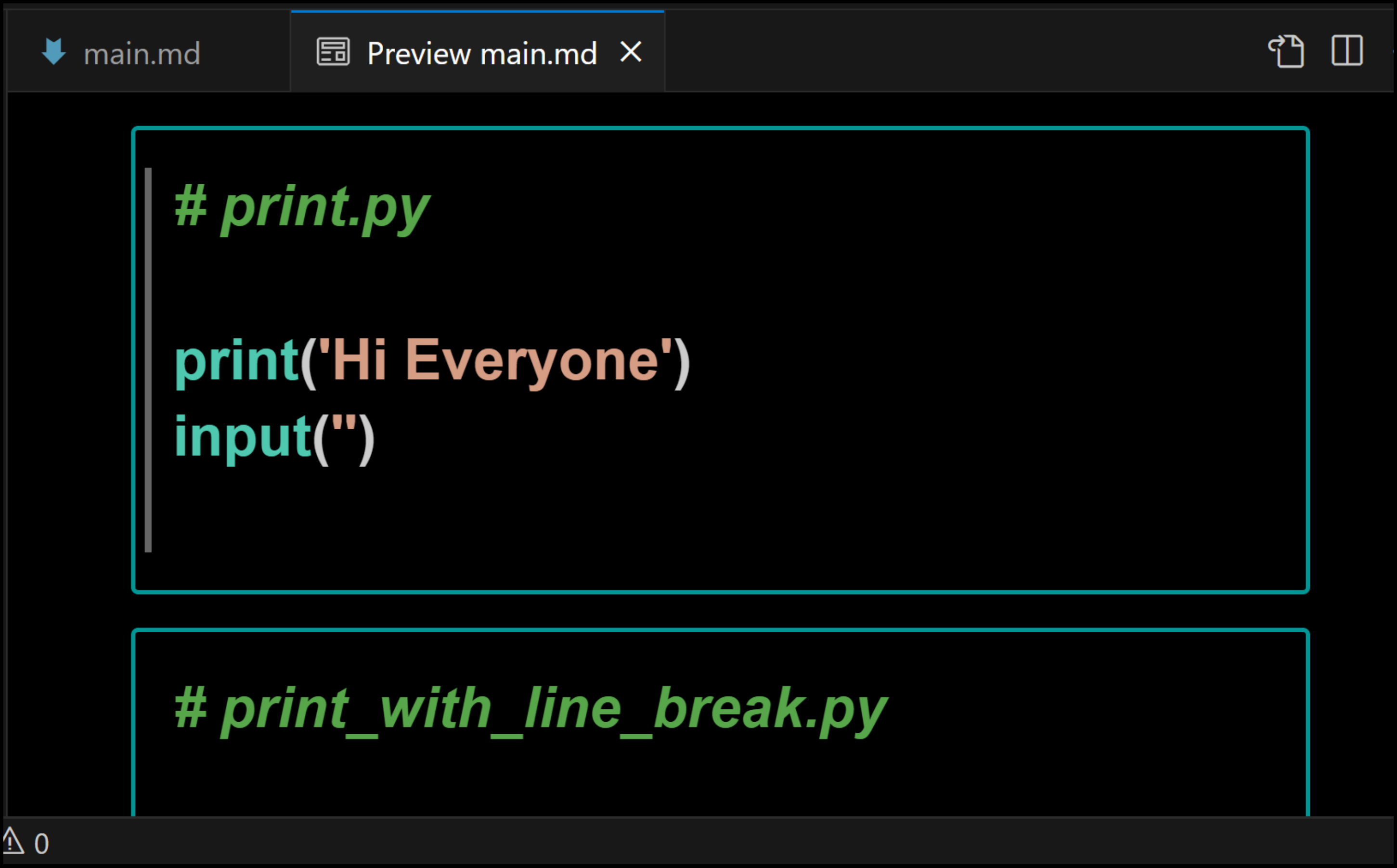
```
# print_with_line_break.py

print('Hi Everyone\nHappy Scripting')
input('')
```

As we can see, the rendered preview of the code is on the right side.

This markdown code book can be shown in VSCode and on github!

As long as we format every script we make with a beginning and end tag, then we can easily make code books by combining all scripts and using find/replace to replace the #b with ```python
or any language we are using, such as replacing the #b with ```javascript
and then find/replace the #e with ```

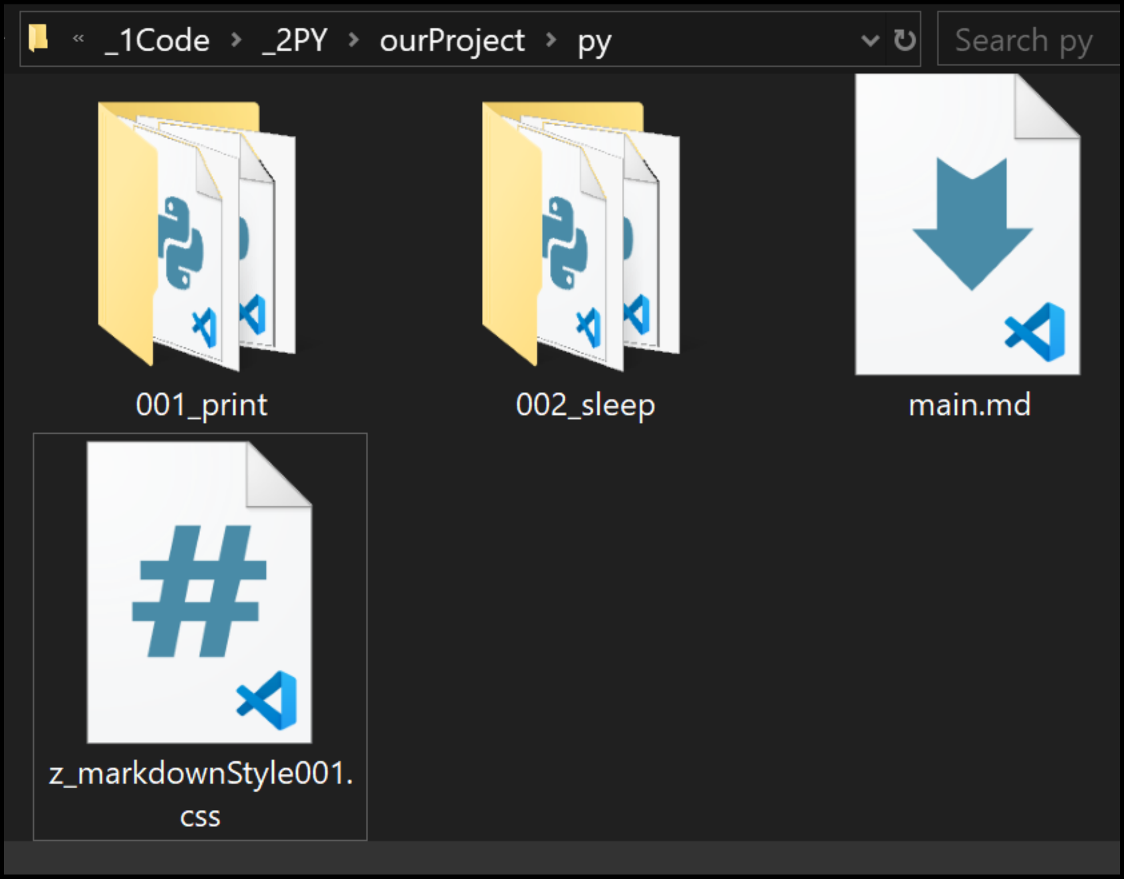
In VSCode we can apply a style sheet to the markdown to make it look nicer when rendered.



```
# print.py

print('Hi Everyone')
input('')
```

```
# print_with_line_break.py
```

We make a .css file and put it in our py folder that contains our .md file

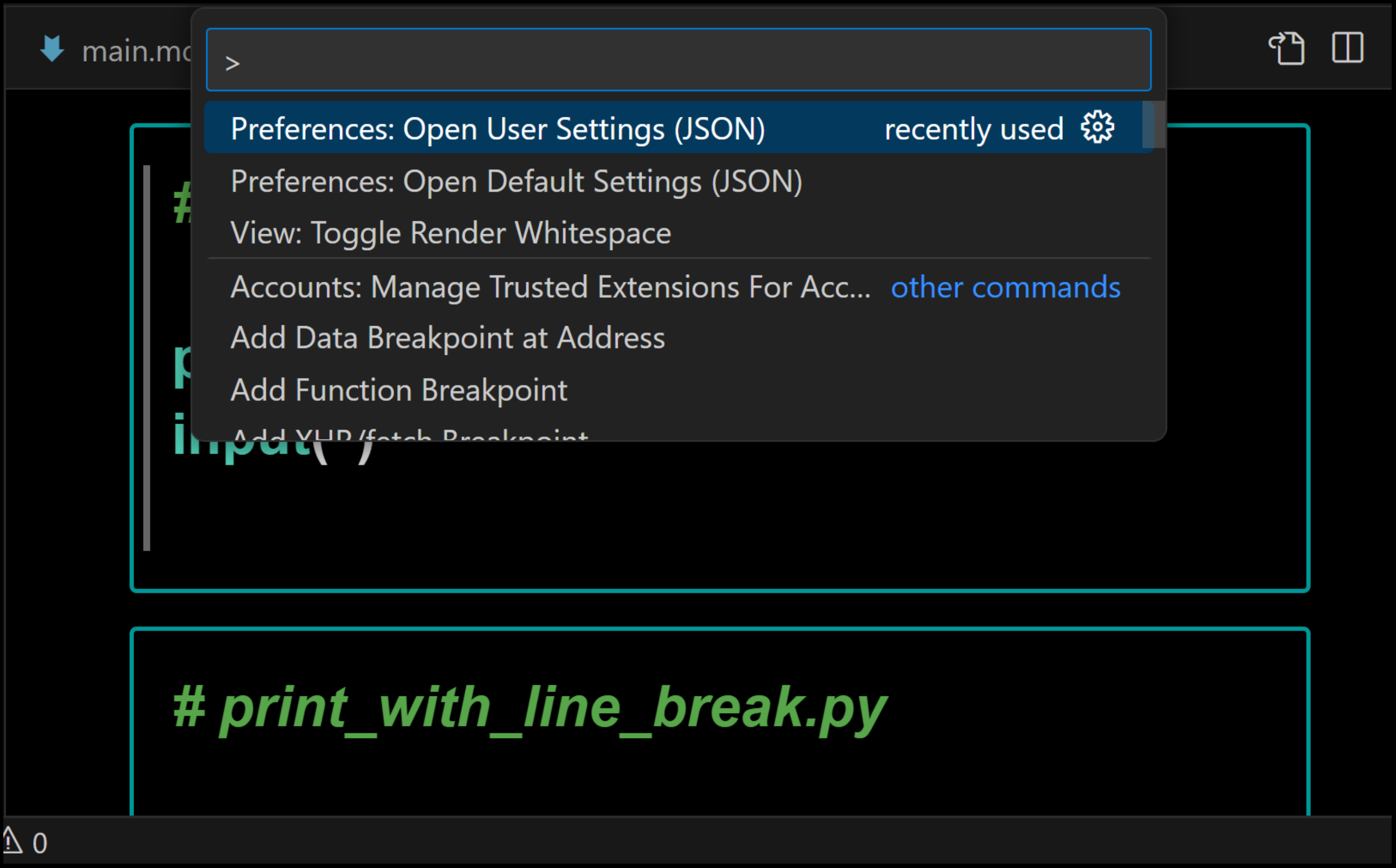


We name the css file
z_markdownStyle001.css

The code of this css file is shown on the next page.

```
/* Dedicated to God the Father */
/* All Rights Reserved Christopher Andrew
Topalian Copyright 2000-2024 */
/* https://github.com/ChristopherTopalian */
/*
https://github.com/ChristopherAndrewTopalian
*/
/* z_markdownStyle001.css */

pre
{
    background-color: rgb(0, 0, 0);
    border: solid 2px rgb(0, 150, 150);
    font-weight: bold;
}

pre code
{
    background-color: rgb(0, 0, 0);
    font-size: 24px;
    font-weight: bold;
}
```

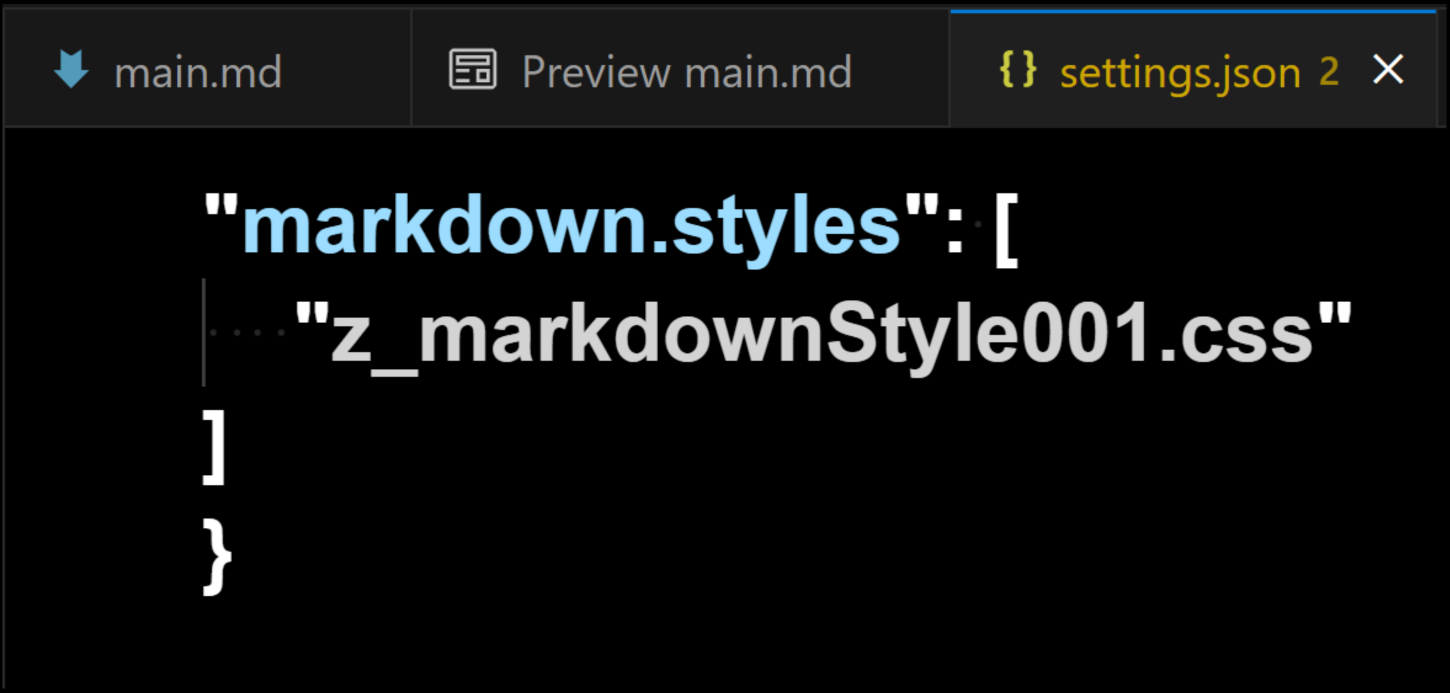
F1 to Open Commands in VSCode



```
>
Preferences: Open User Settings (JSON)        recently used
Preferences: Open Default Settings (JSON)
View: Toggle Render Whitespace
Accounts: Manage Trusted Extensions For Acc...  other commands
Add Data Breakpoint at Address
Add Function Breakpoint
```

Choose:
Preferences: Open User Settings (JSON)

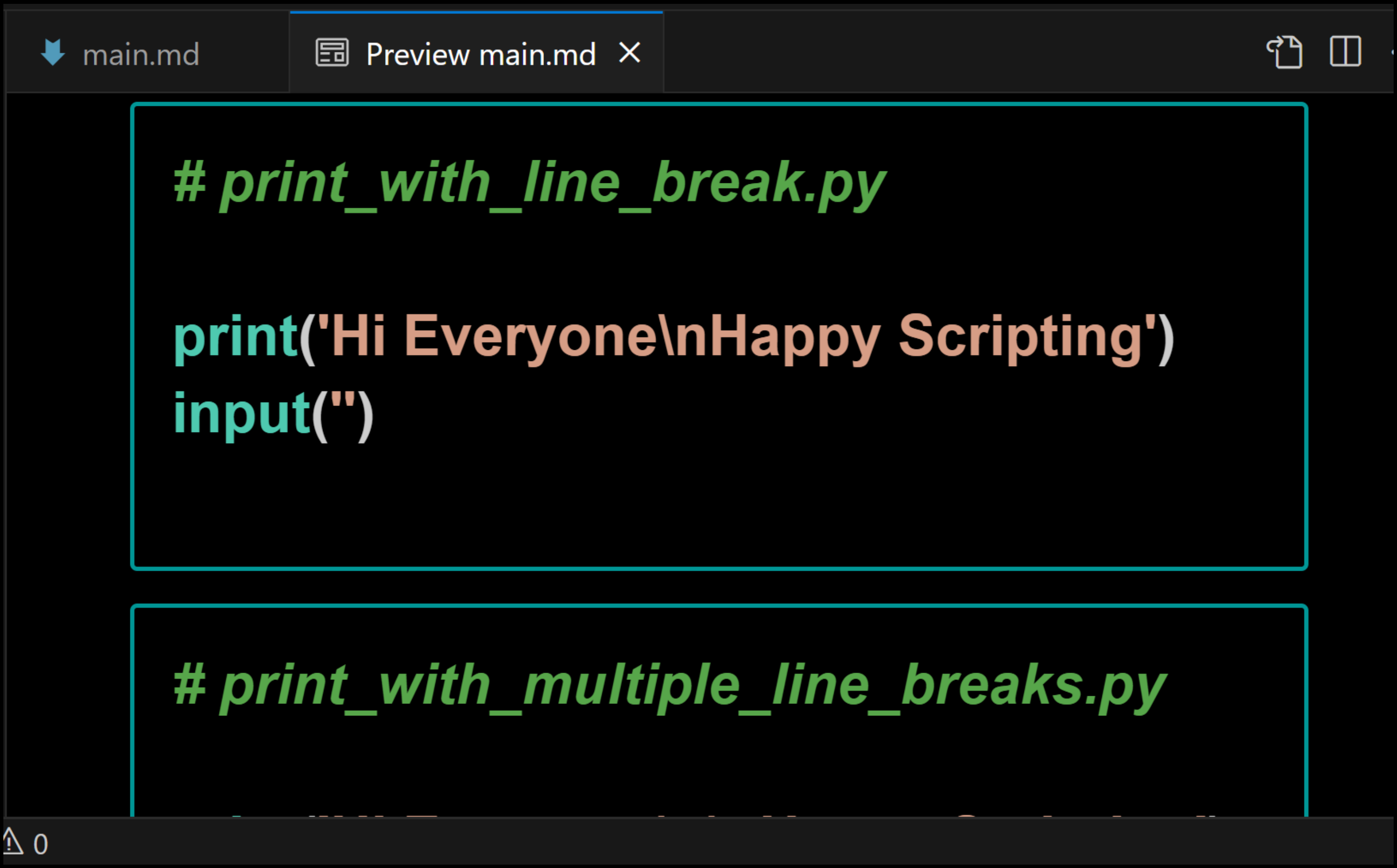
We make sure to include our markdown style sheet



```
"markdown.styles": [
    "z_markdownStyle001.css"
]
}
```

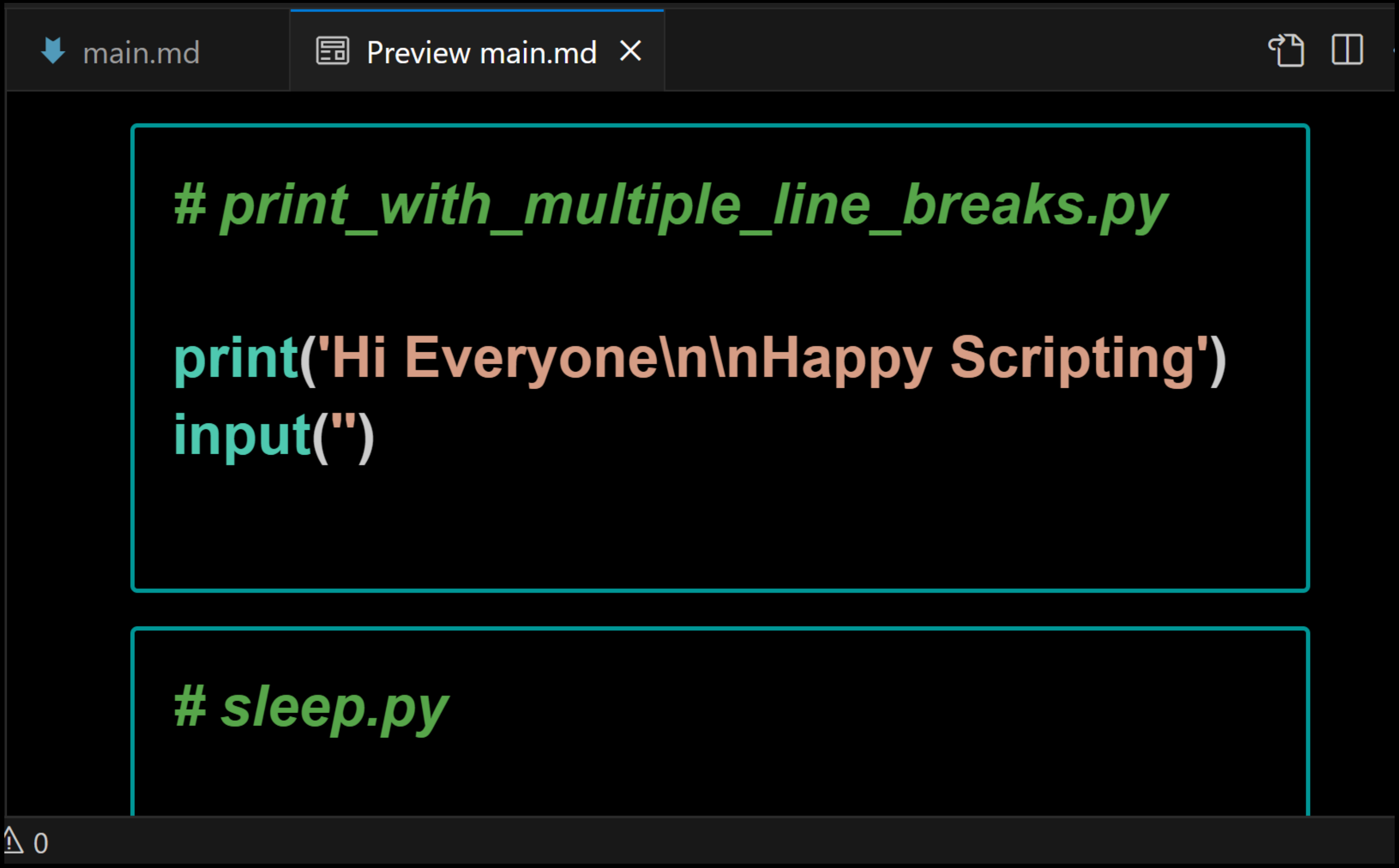
There are many ways to style our markdown language, so have lots of fun experimenting.

Here is what the rendered preview looks like of our new code book:



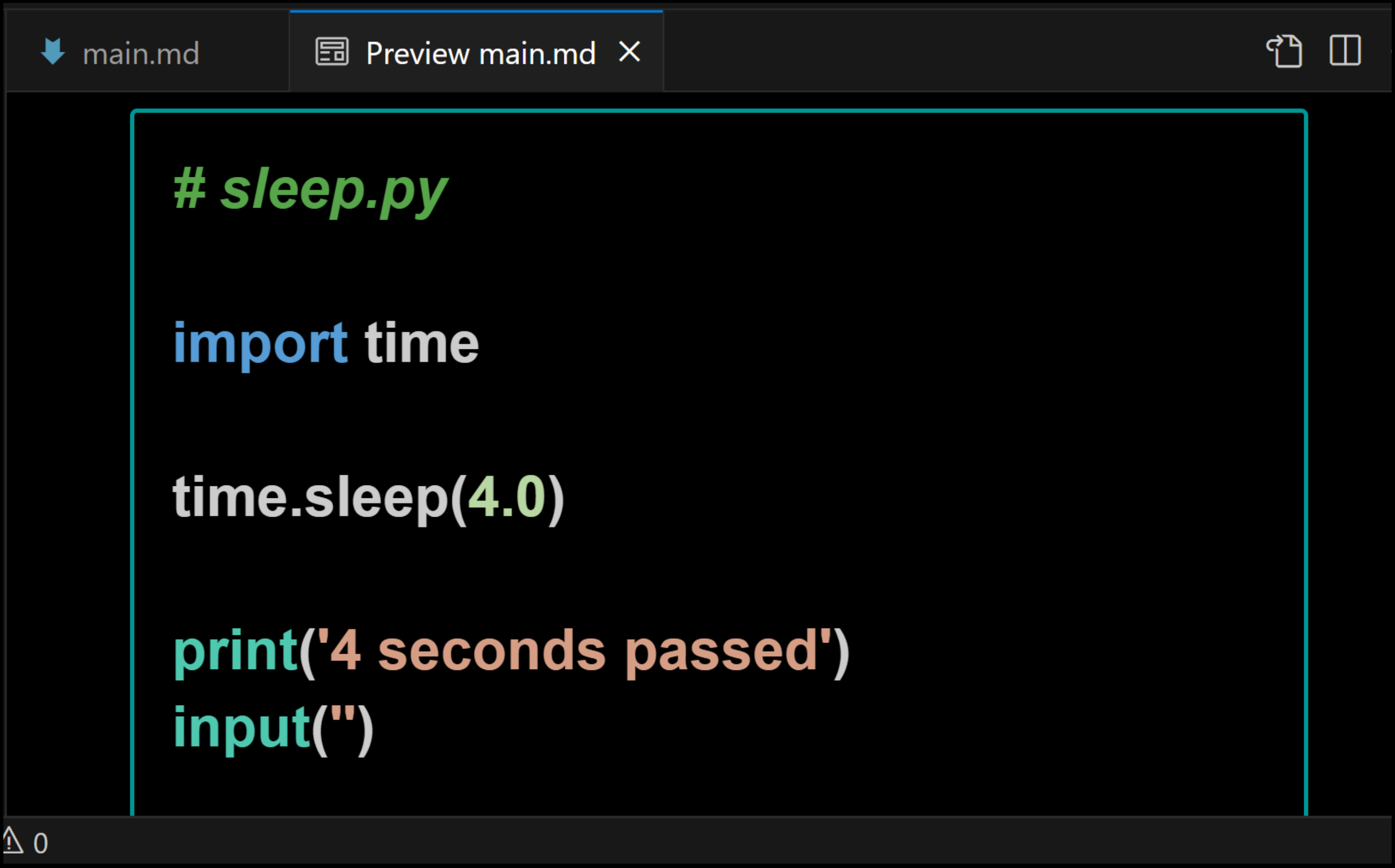
```
# print_with_line_break.py

print('Hi Everyone\nHappy Scripting')
input('')
```

```
# print_with_multiple_line_breaks.py
```

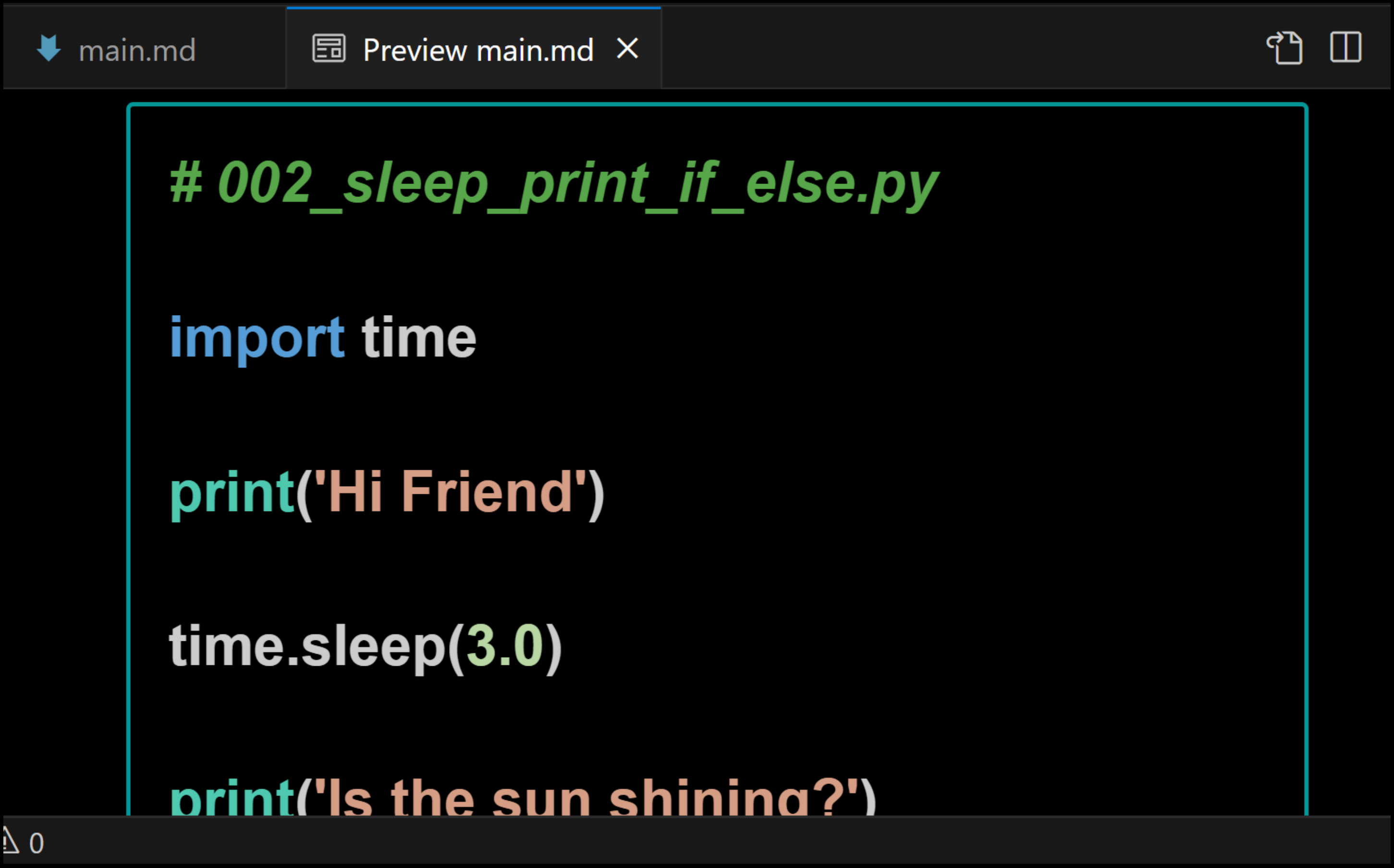


```
# print_with_multiple_line_breaks.py

print('Hi Everyone\n\nHappy Scripting')
input('')
```

```
# sleep.py
```

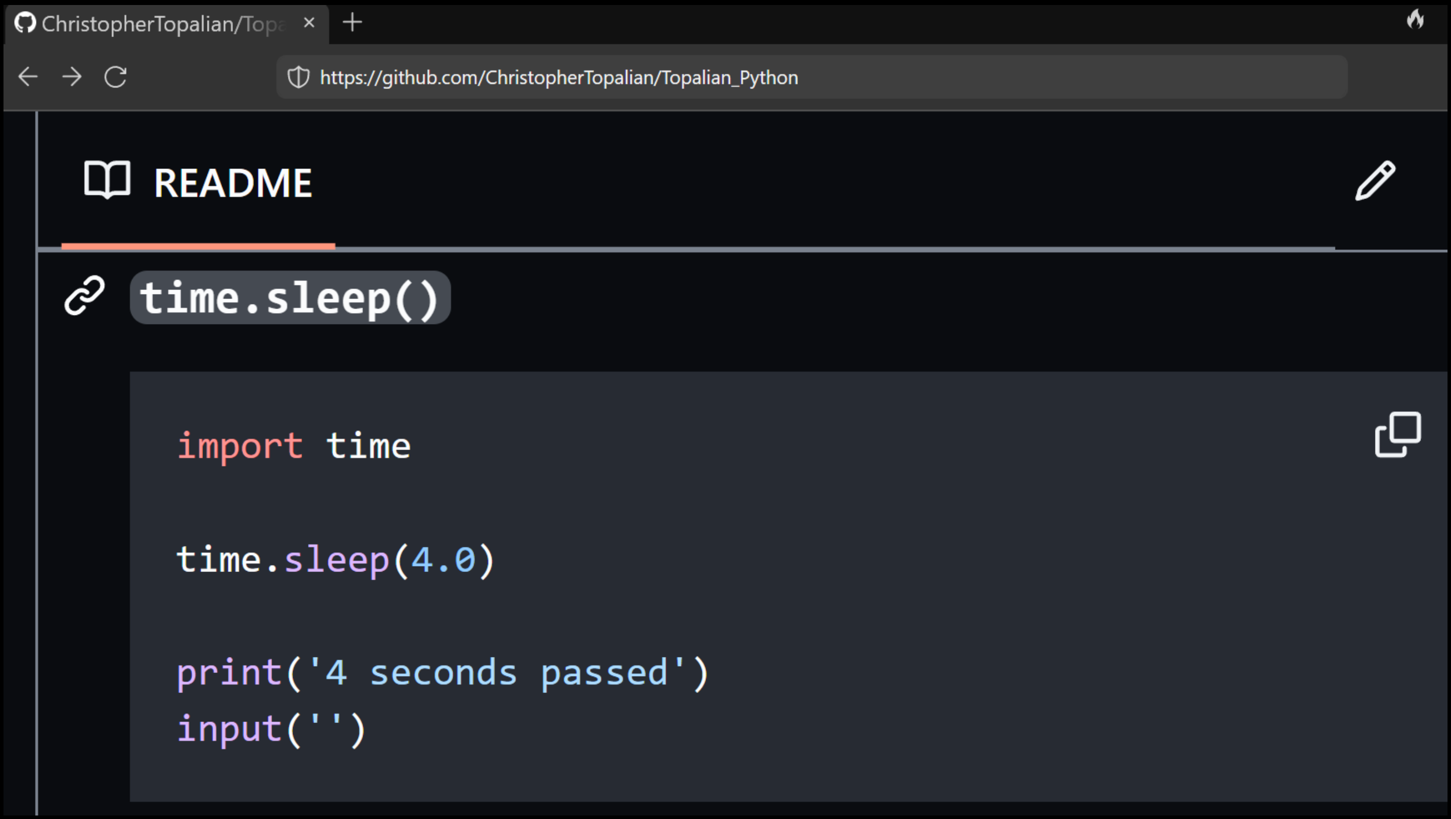


```python
# sleep.py

import time

time.sleep(4.0)

print('4 seconds passed')
input('')
```



```
# 002_sleep_print_if_else.py

import time

print('Hi Friend')

time.sleep(3.0)

print('Is the sun shining?')
```

Github shows rendered markdown very nicely



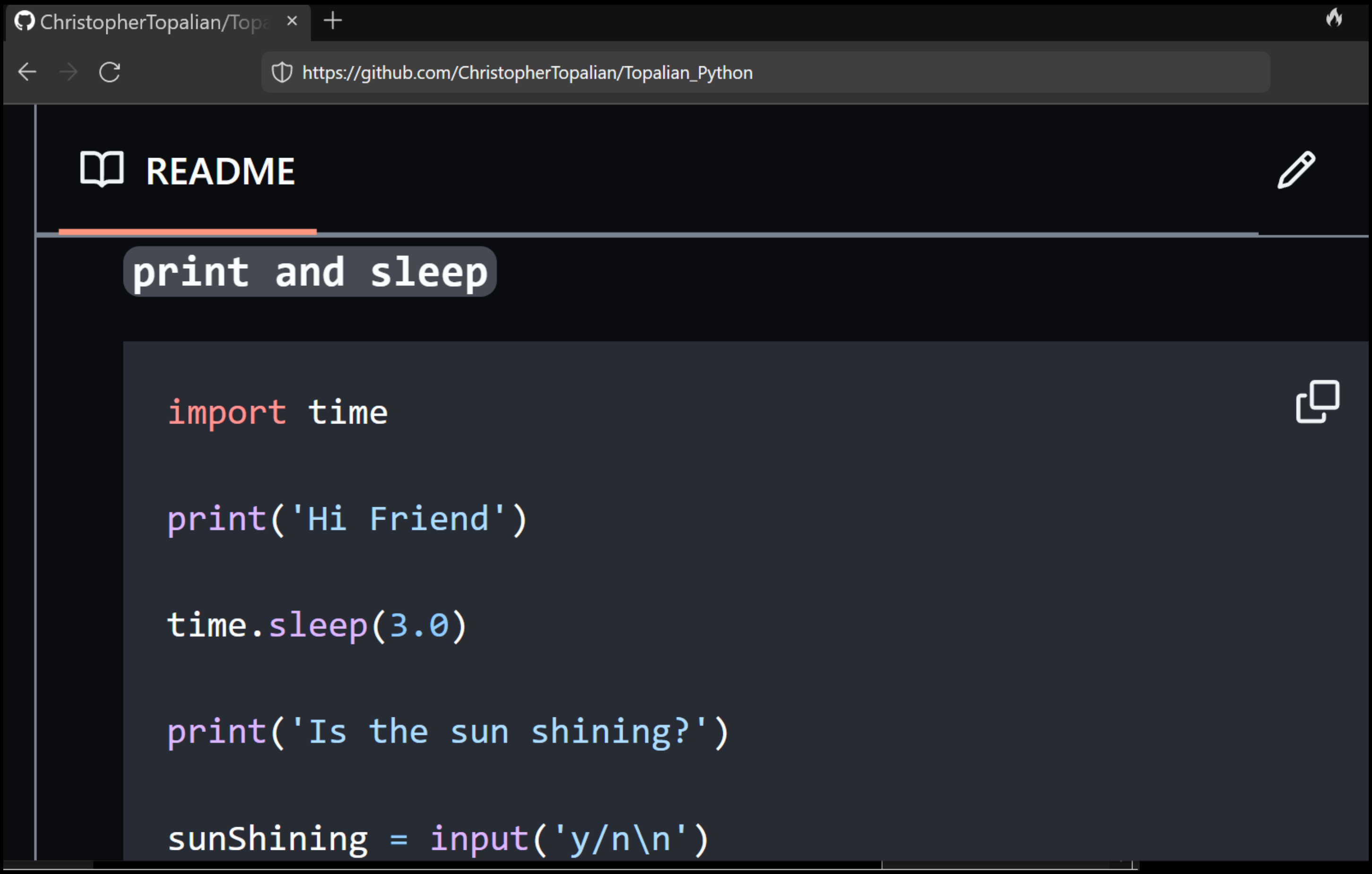
`time.sleep()`

```python
import time

time.sleep(4.0)

print('4 seconds passed')
input('')
```

As we see here, our markdown code has been placed in the README file of our github project.

This way all of our scripts of our code book will be seen when a person visits our github repository main page.

Notice that there is a Copy Button on the top right of each of the scripts rendered.



print and sleep

```
import time

print('Hi Friend')

time.sleep(3.0)

print('Is the sun shining?')

sunShining = input('y/n\n')
```

This method of book making allows us to stay organized with our scripts in folders, while then allowing the scripts to be combined, in a specific order, using either the command prompt method or the .bat files provided, or the python or node.js files provided in this tutorial project.

Making Code Books is fun and easy :-)

Happy Scripting :-)
and
Happy Book Making :-)

How to Combine .js files into one main.md file using Command Prompt

Version for when we have only ONE folder of .js files that we want to combine.

// HowToCombineJSFilesOneFolder.js

First, we add two new lines at the end of every script. This way they will later combine nicely with a line break in between each script.

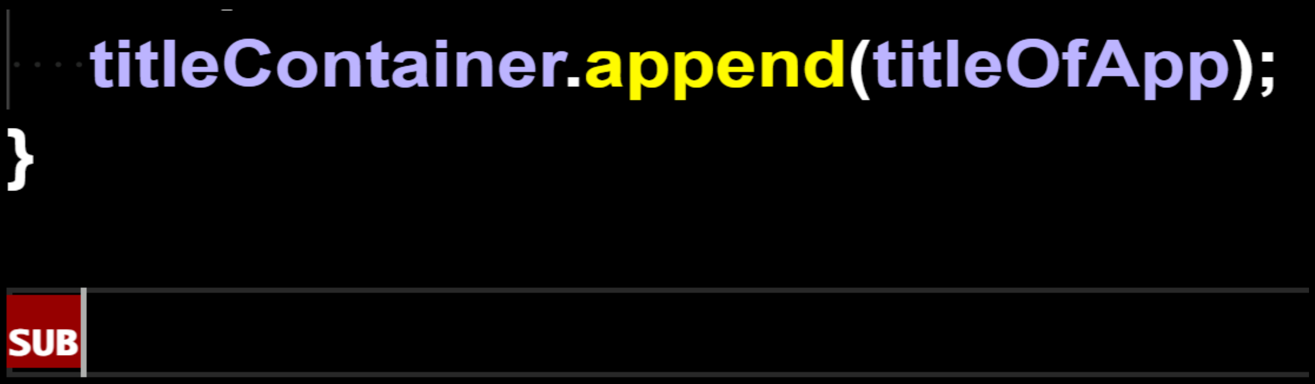
We open our js folder.
In our js project folder, we type
cmd
in the address bar of the folder and then
press enter

This opens our js folder in the Command prompt

We type in the words
copy *.js main.md
and then press enter

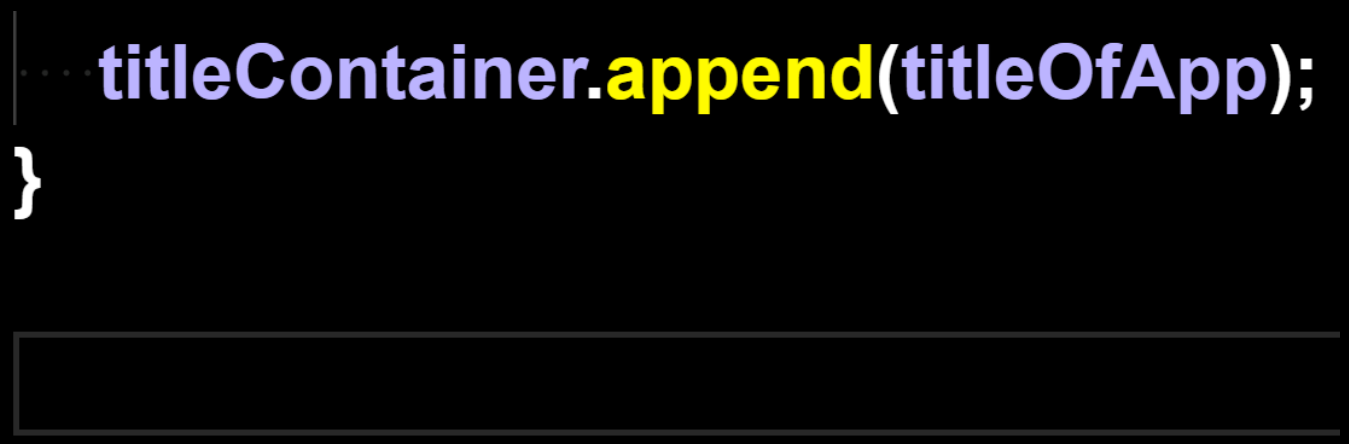
This creates a new file that is named main.md
This new file contains all .js files in ONE file.

But, there is a junk character at the end of the main.md file that we have to delete. In VSCode the character might be called SUB



```
    titleContainer.append(titleOfApp);
}

SUB
```

We remove this junk SUB character.



```
    titleContainer.append(titleOfApp);
}
```

As we can see, the junk character is removed.

How to Combine .js files into one main.md file using Command Prompt

Version for when we have js scripts in subfolders in our js project folder, that we want to combine.

// HowToCombineJSFiles.js

TUTORIAL:
How to Combine all .js files in all folders that are in our js folder.

Getting things ready:
We should add two new lines at the end every script. This way they will combine nicely with a line break in between each script.

Step One: Open our js folder

Step Two: Type in the address bar of the js folder, cmd, press Enter

This opens our js folder in the command prompt

Step Three: Type the command shown below in the command prompt and then press Enter

```
for /r "%CD%" %i in (*.js) do type "%i" >>
main.md
```

Now we have a newly created .md file named main.md that has all of our js files included into one file.

How to Combine .js files into one main.md file using a batch file

Version for when we have js scripts in subfolders in our js project folder, that we want to combine.

// HowToCombineJSFilesUsingBatFile.js

We can combine all of the .js files
that are located in our js folder
into one main.js file, using either:
    The Command Prompt Method
    or
    The .bat File Method

The .bat file method is very easy.
We double click the bat file, which is located in our js folder, and it will make a main.md file, which includes all .js files in the js folder, including all .js files in all subdirectories of our js folder.

This is a very easy way to combine our .js files, because we can double click the .bat file anytime, and it will again generate the main.md file, which includes all .js files in the js folder, including all .js files in all subdirectories of our js folder. This makes making a code book very easy.

Happy Scripting :-)

```
:: Topalian_Combine_JS_Files.bat

:: This .bat File Combines All .js files in all folders
of our project folder, into one main.md file.

:: To activate this .bat file, we double click
the .bat file, while it is located in our js folder.

@echo off
:: set the output file name
set "output=main.md"

:: clear existing output file
type nul > "%output%"

:: loop through all JavaScript files in
subdirectories
for /r %%i in (*.js) do (
    :: append content of each file to output file
    type "%%i" >> "%output%"
)

echo "JavaScript files combined into %output
% successfully."
```

How to Combine .js files into one main.md file using Node.js

This version will successfully combine a single folder of js files.
It also works to combine js files in all subdirectories.

```
// Topalian_Combine_JS_Files.js

let fs = require('fs');
let path = require('path');

function combineJSFiles(directory,
scriptFilename)
{
    let outputFilePath = path.join(directory,
'main.md');

    let fileContents = [];

    function traverseFolder(folder)
    {
        let files = fs.readdirSync(folder);

        for (let i = 0; i < files.length; i++)
        {
            let file = files[i];

            let filePath = path.join(folder, file);

            let stats = fs.statSync(filePath);

            if (stats.isDirectory())
            {
                traverseFolder(filePath);
            }
            else if (path.extname(filePath) === '.js')
            {
                let content = fs.readFileSync(filePath,
'utf8');
                // check if file is not script file itself
                if (filePath !== scriptFilename)
                {
                    fileContents.push(content);
                }
            }
        }
    }

    traverseFolder(directory);

    fs.writeFileSync(outputFilePath,
fileContents.join('\n'), 'utf8');

    console.log(`Combined $
{fileContents.length} .js files into $
{outputFilePath}`);
}
```

```
// get current directory of script
let currentDirectory = process.cwd();

// get filename of script
let scriptFilename = __filename;

combineJSFiles(currentDirectory,
scriptFilename);
```

How to Combine .js files into one main.md file using Python

This version will successfully combine a single folder of js files.
It also works to combine js files in all subdirectories.

```
# Topalian_Combine_JS_Files.py

import os

def combineJSFiles(directory, scriptFileName):
    outputFilePath = os.path.join(directory,
'main.md')
    fileContents = []

    def traverseFolder(folder):
        for root, dirs, files in os.walk(folder):
            for file in files:
                filePath = os.path.join(root, file)
                if filePath != scriptFileName and
filePath.endswith('.js'):
                    with open(filePath, 'r',
encoding='utf-8') as f:
                        fileContents.append(f.read())

    traverseFolder(directory)

    with open(outputFilePath, 'w', encoding='utf-
8') as f:
        f.write('\n'.join(fileContents))
    print(f"Combined {len(fileContents)} .js files
into {outputFilePath}")
```

```
# get current directory of script
currentDirectory =
os.path.dirname(os.path.abspath(__file__))

# get filename of script
scriptFileName = os.path.abspath(__file__)

combineJSFiles(currentDirectory,
scriptFileName)
```

What other file types can we combine?

We have combined .js files in this book, but we might choose to instead combine:

.py   or   .html   or   .txt

This is very useful for book making.

In each of the scripts shown in this book, we can manually change the parts where it says .js, with .py, if we wanted to, for instance, copy all .py files into one main.py file.

We can do the same thing for .html files, where we change the file type it will be combining to .html and it will combine all .html files into one main.html file.

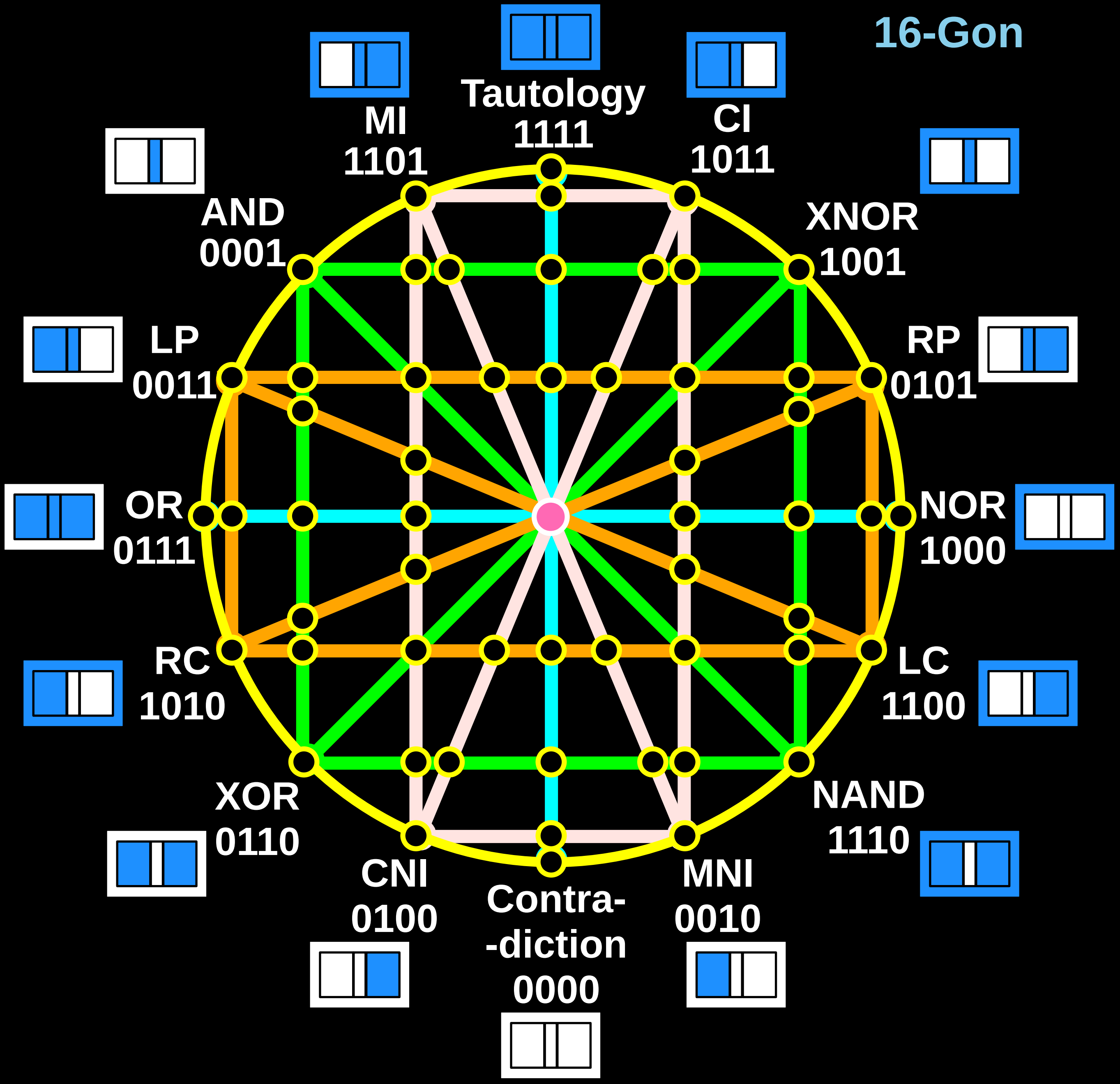
We add two line breaks at the end of all files, so that there is space between files, when they are combined.

Remember too, that not all file types will combine, but the ones above will.

The original files are not changed. The content from the original files is only copied from.

Christopher Topalian

True Artificial Intelligence System



For More Tutorials:

GitHub.com/ChristopherTopalian

GitHub.com/ChristopherAndrewTopalian

Sites.google.com/view/CollegeOfScripting

CollegeOfScripting.weebly.com

CollegeOfScripting.wordpress.com

Youtube.com/ScriptingCollege

Twitter.com/CollegeOfScript

Rumble.com/user/CollegeOfScripting

Dedicated to God the Father

This book is created by the

College of Scripting Music & Science.

Always remember, that each time you write a script with a pencil and paper, it becomes imprinted so deeply in memory that the material and methods are learned extremely well.

When you Type the scripts, the same is true. The more you type and write out the scripts by keyboard or pencil and paper, the more you will learn programming!

Write and Type every example that you find.

Keep all of your scripts organized.

Every script that you create increases your programming abilities.

SEEING CODE, is one thing,

but WRITING CODE is another.

Write it, Type it, Speak it, See it, Dream it.

CollegeOfScripting.weebly.com