# **Goodreads API Scraping Project**

### **Reference Document**

### **Using: Python, Heroku**

------------------------------------------------------------------------------------------------------------

Collaborators:

Aditya – [adikid1996@gmail.com](mailto:adikid1996@gmail.com)

Manel Mahroug - [manel.khan@gmail.com](mailto:manel.khan@gmail.com)

------------------------------------------------------------------------------------------------------------

## Project Breakdown

The following describes the breakdown of project:

1. Setting the GitHub Repository for the project for version control (VCS). ✔

2. Cloning the repository and setting up the project with VCS. ✔

3. The testing of *goodsread.api* for the following requirements: ✔

Get review statistics given a list of ISBNs: [book.review\_counts](https://www.goodreads.com/api/index#book.review_counts)

Get a user's read status: [read\_statuses.show](https://www.goodreads.com/api/index#read_statuses.show)

Get info about a member by id or username .[user.show](https://www.goodreads.com/api/index#user.show)

Get a user's followers: .[user.followers](https://www.goodreads.com/api/index#user.followers)

Get people a user is following: [user.following](https://www.goodreads.com/api/index#user.following)

Get a user's friends: [user.friends](https://www.goodreads.com/api/index#user.friends)

4. Setting the Python script to handle API. ✔

5. Threading the Program to run at full capacity. (COULD NOT BE IMPLEMENTED) ❌

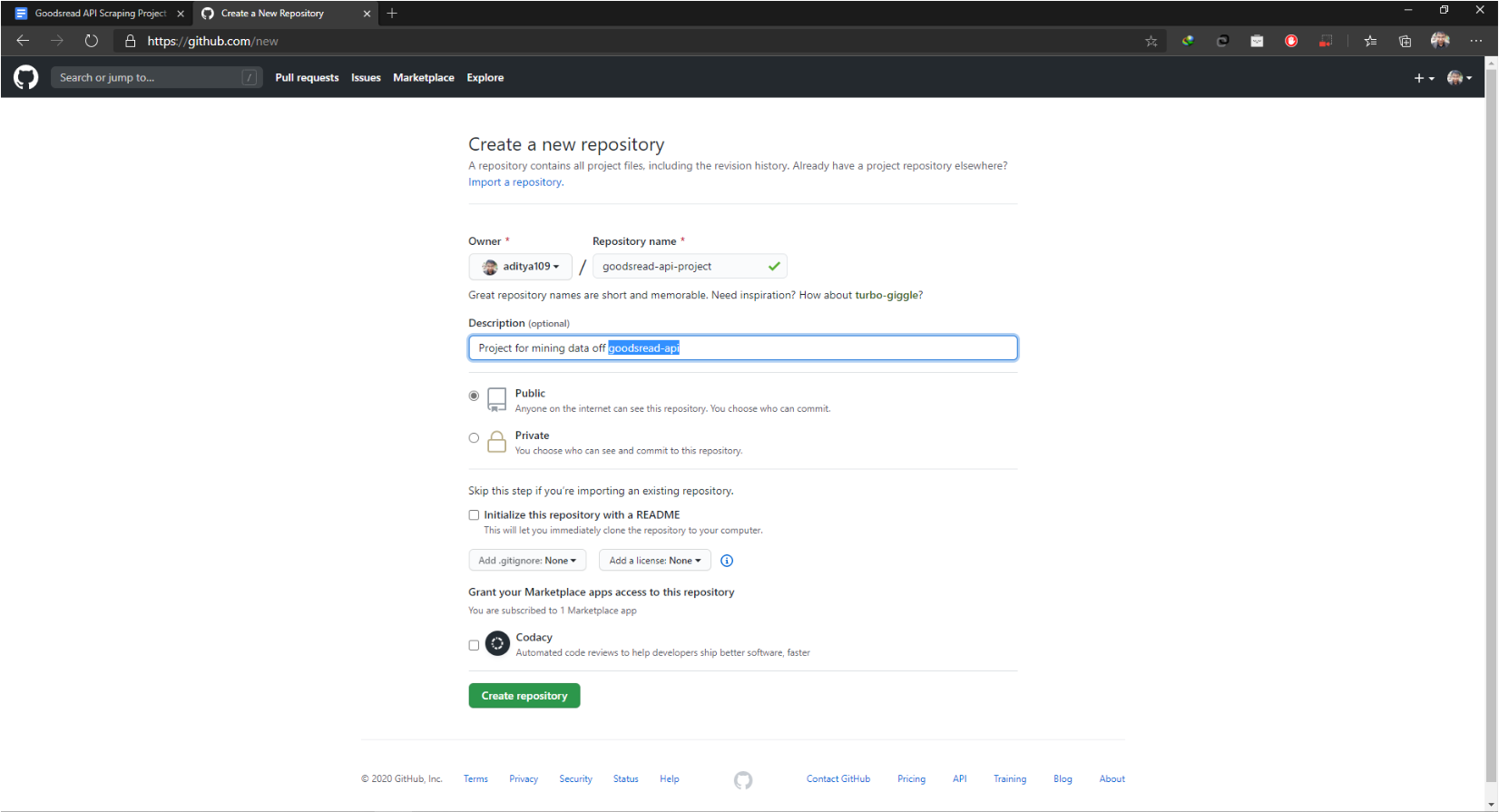
6. Documenting the Project for further usage. ✔

7. Deploying the app on Deployment (Heroku) (if you are interested). (Heroku is a free-tier platform for using continuous deployment of applications) ❔

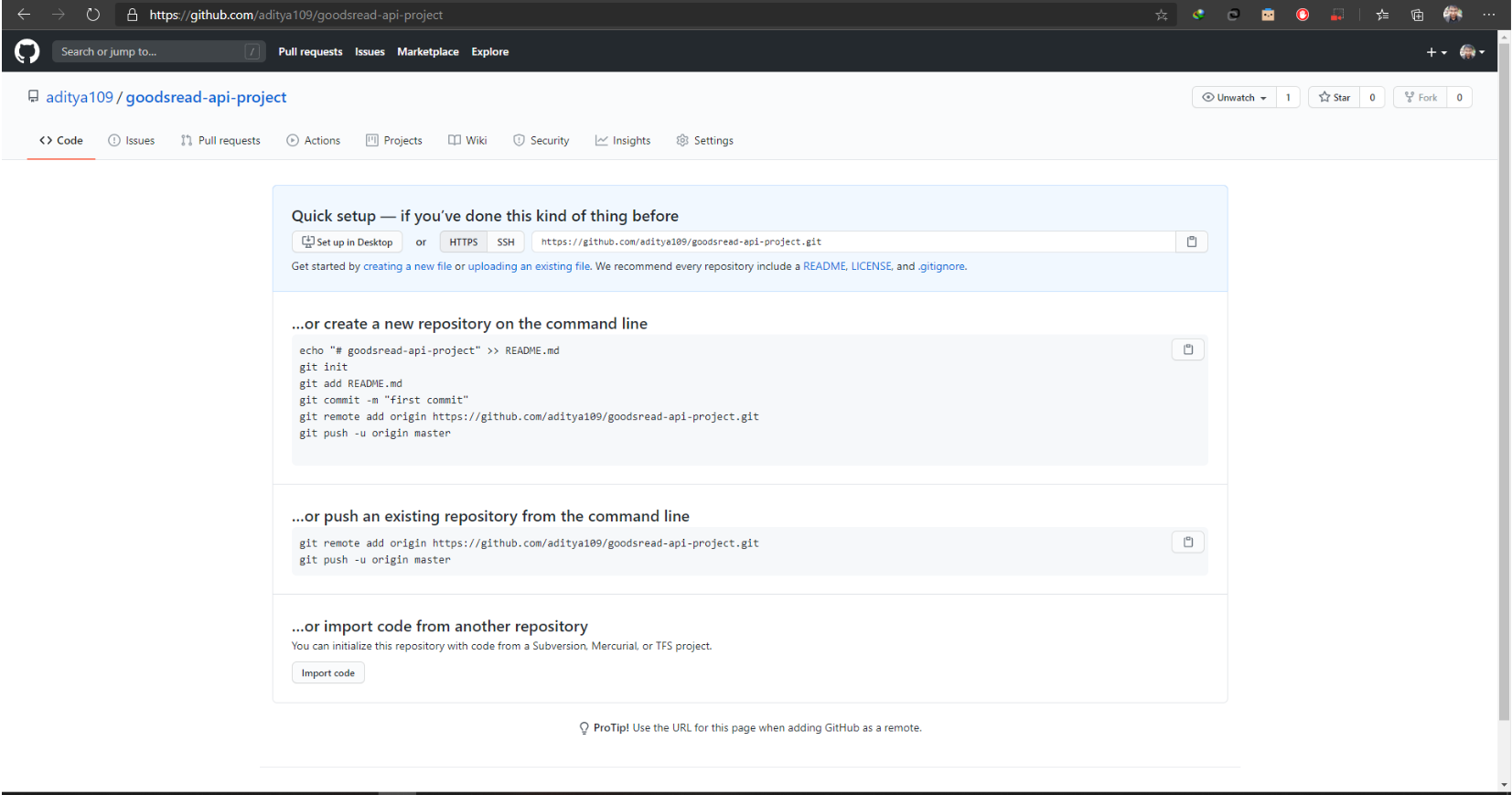
The project has been broken down into small steps, so that the project could even be used for further references.

# Creating Git Repository

1. Open up your browser and create your git repository. You can keep your repository as private or public. But I am keeping mine as public so that you can find reference for that same.  
   I suggest leaving the **initialize repository with a README** option unchecked, as we want to create our own documentation.  
   My [Github Repo Link](https://github.com/aditya109/goodsread-api-project)



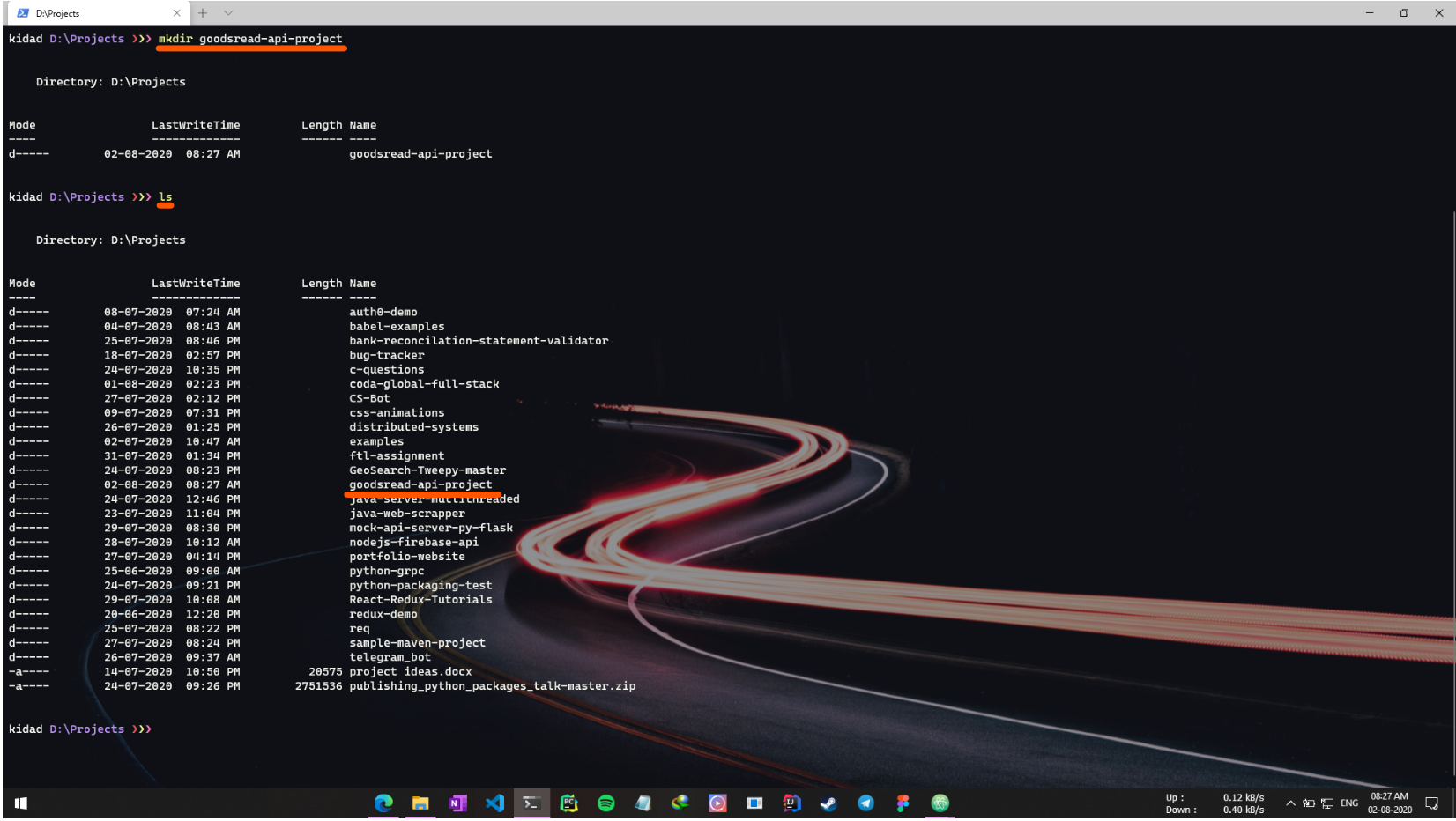
1. Once you have created a repository, you will have a similar screen on your browser.



# Setting up VCS on your project

1. Open up your command prompt.  
   Navigate to any directory and create your project folder.   
   (I swear please bear with this font, I swear I will increase the font size in the later ones)

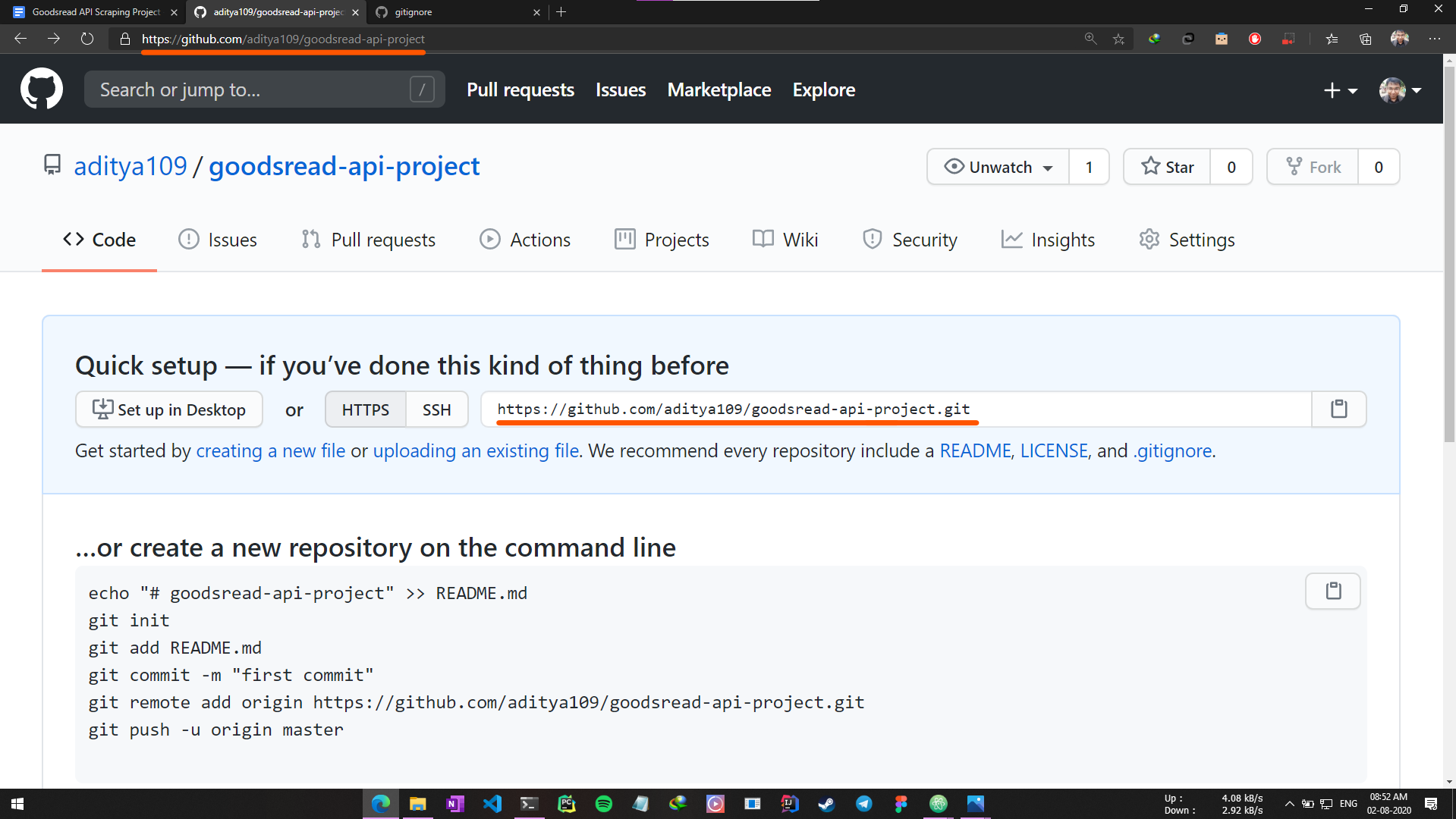
Use : mkdir [project-directory-name]



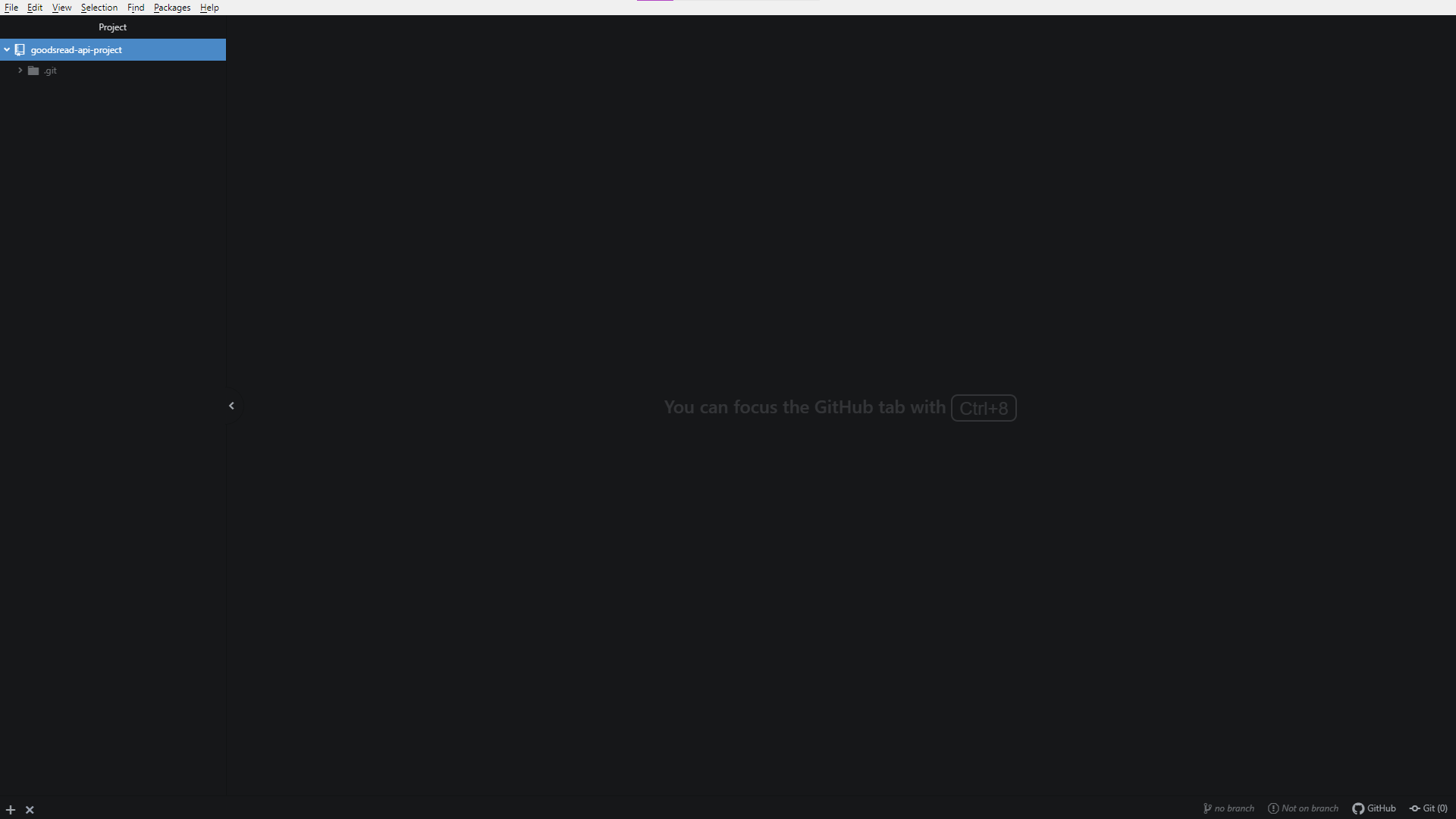
1. Change to that directory.  
   Use: cd [project-directory-name]
2. Initialize git in the directory.  
   Use: git init .
3. Check if you have your git repository link set as remote `origin` or not.  
   Use : git remote -v

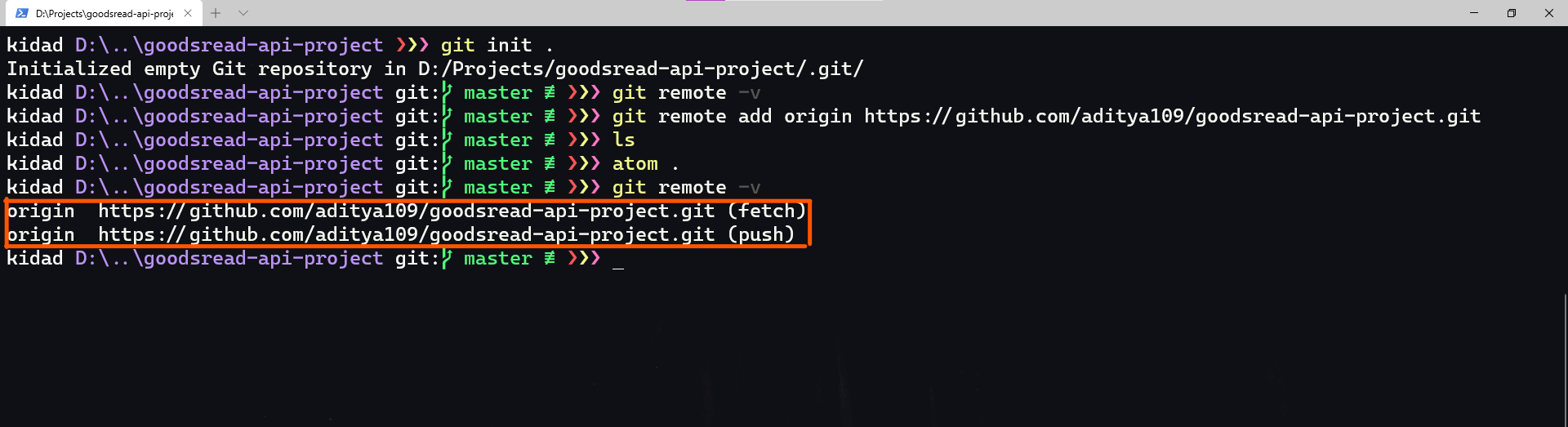
Most probably it won’t return anything, because we did not do anything as `git clone`. So let’s set that up.

1. Add a remote to your project.  
   Use: git remote add origin [your-git-repo-url]



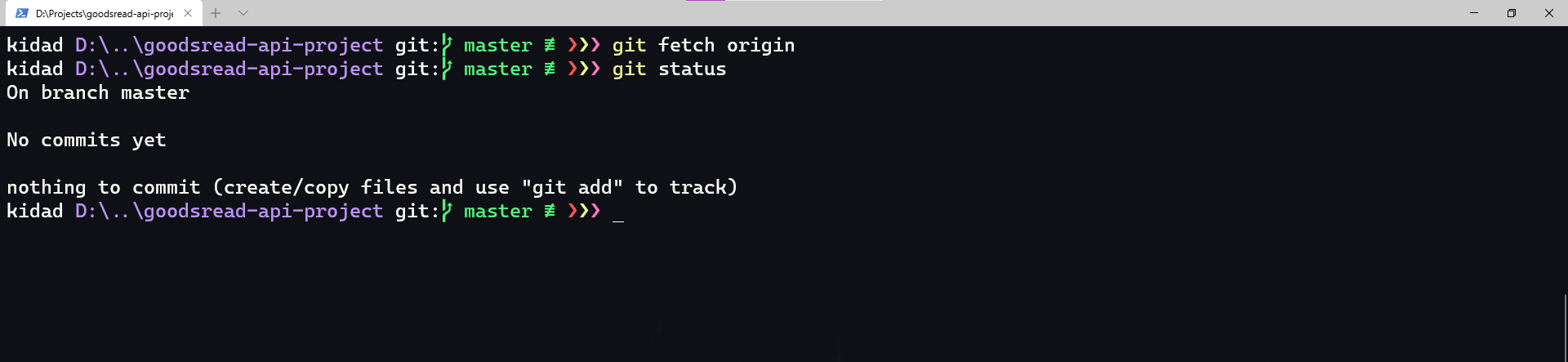
It could be either or those.

1. Open atom in this directory.  
   Use: atom .  
   This should open up an Atom Editor.  
   
2. Go back to your command prompt.   
   Re-check your git remote.  
   Use : git remote -v

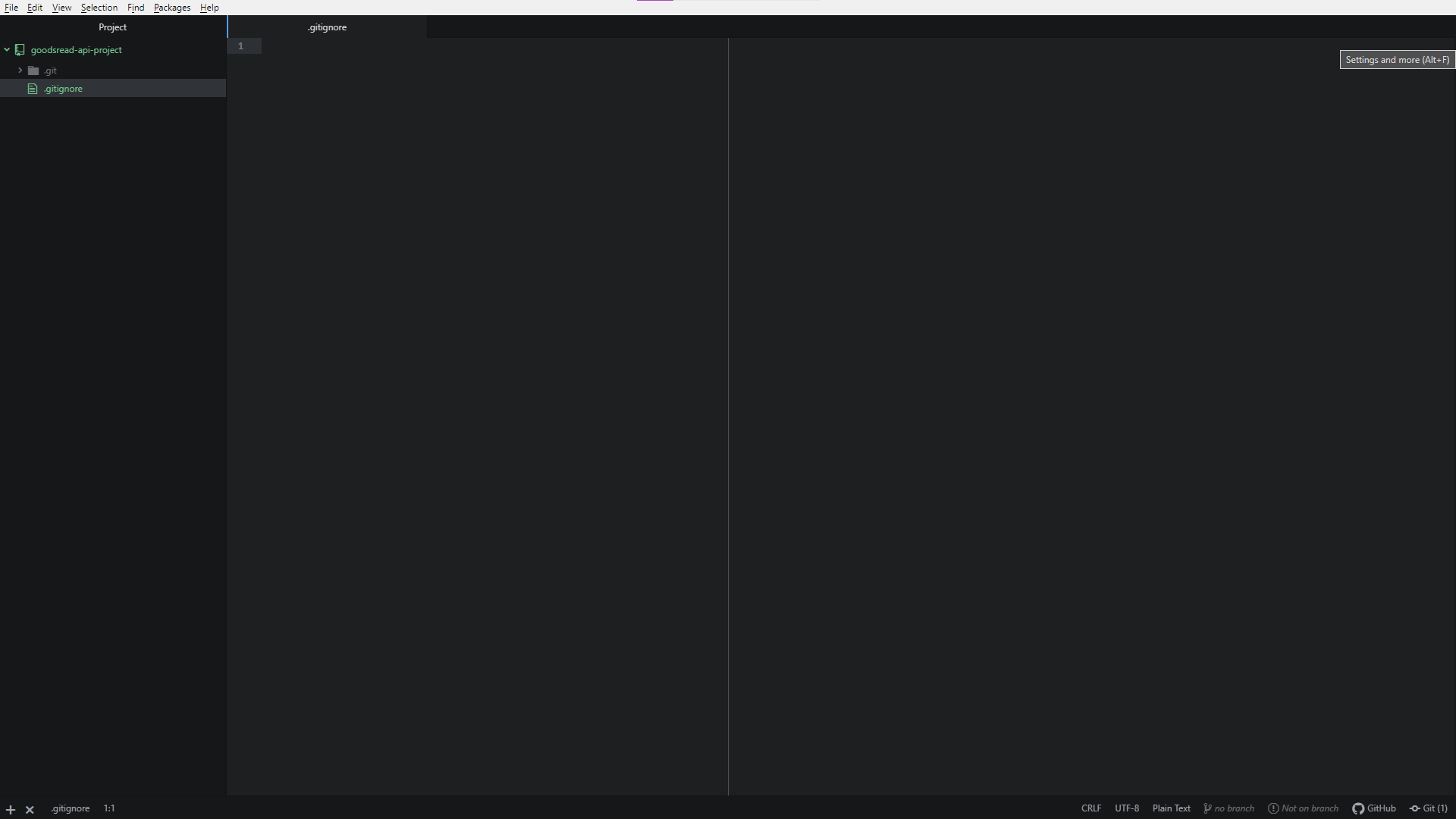


1. Just in case, let’s sync up our git repository.  
   Use: git fetch origin

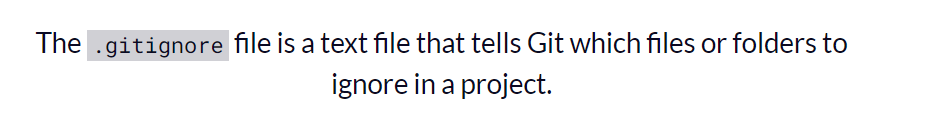
If you might check the (include README.md) option while creating your git repository, it should give you some output otherwise it should not return any output.

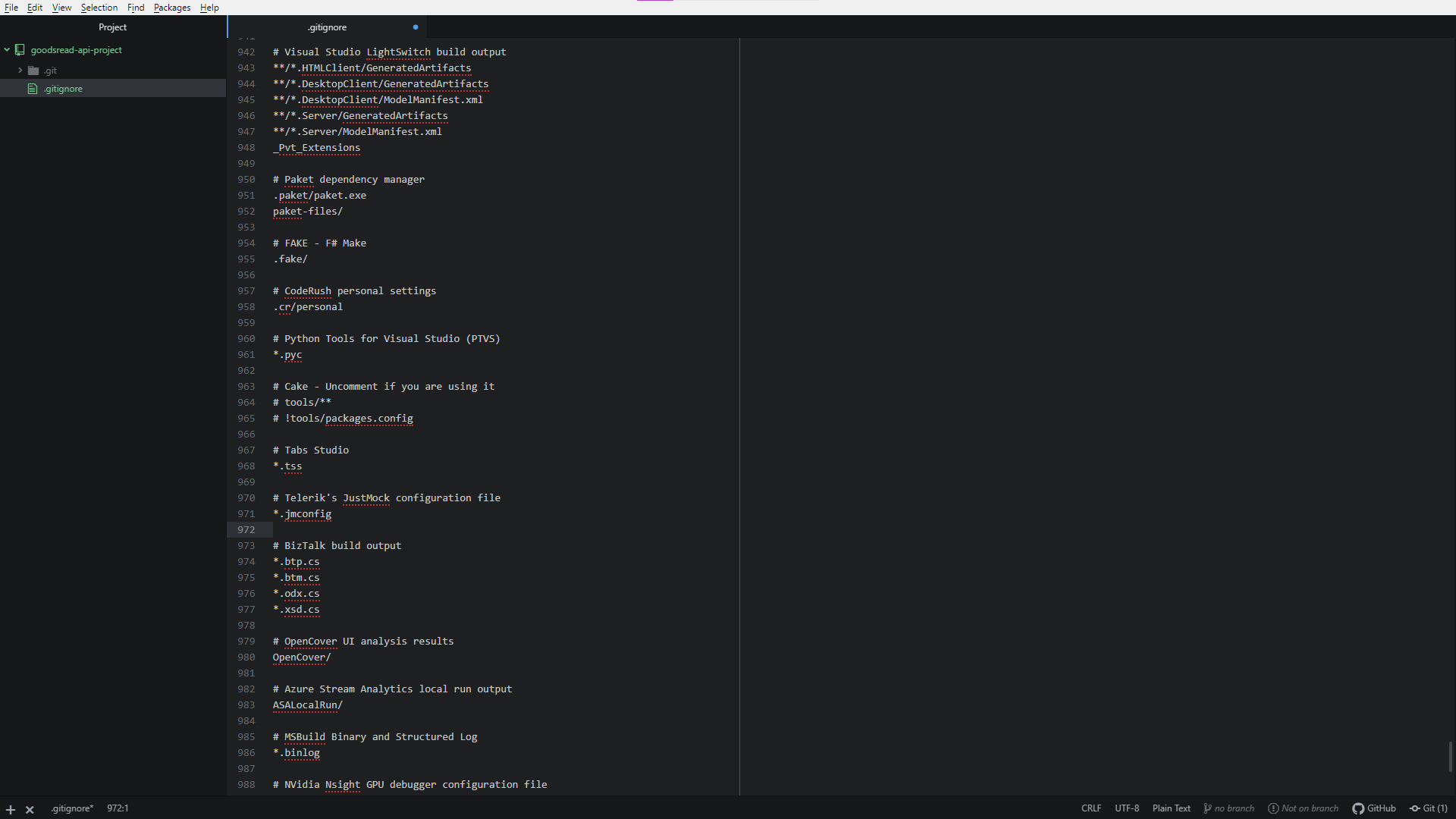
1. Let’s check our git status  
   Use : git status  
   
2. Let’s add a **.gitignore** file to your repository.

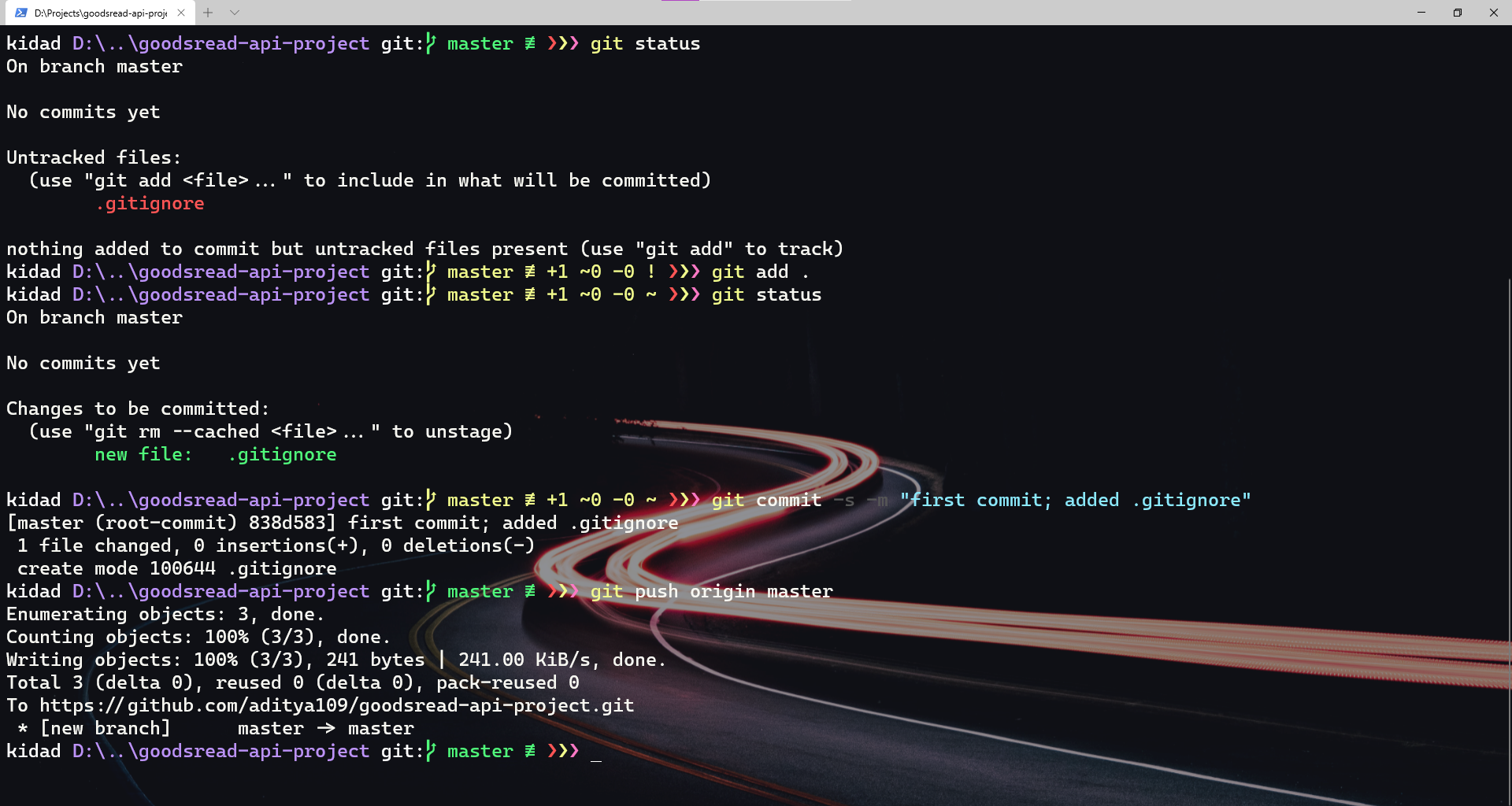
Switch to your text editor.

Create a **.gitignore** file for your project.  


1. Please open up the following gist link.  
   My [gitignore gist link](https://gist.github.com/aditya109/0a549ce49e5f44c6ca6ea186994438e4)

Copy contents off it and paste it in your **.gitignore** file.  




1. Let's start by doing our first commit.  
   Jump back to your terminal and add files to your git repo.  
   Use: git add .
2. Commit the same with a suitable message.  
   Use: git commit -s -m "[message]"
3. Now push your commit to your origin.  
   Use: git push origin master  
   
4. That’s your step 2 !

# Testing of *goodreads.api*

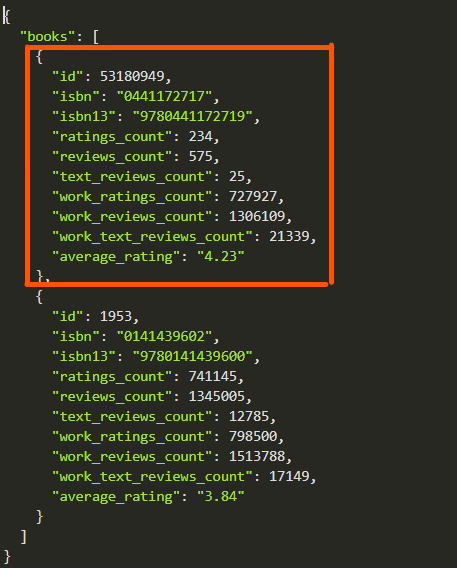
## Get review statistics given a list of ISBNs

API :

|  |
| --- |
| https://www.goodreads.com/book/review\_counts.json?isbns=0441172717%2C0141439602&key={your key} |

Return Type: JSON

Sample:



### 

## Get a user's read status

API :

## Get info about a member by id or username

API :

|  |
| --- |
| https://www.goodreads.com/user/show/24347448.xml?key={your key} |

Return Type: XML

Sample:

|  |
| --- |
| <GoodreadsResponse>  <Request>  <authentication>true</authentication>  <key>  <![CDATA[ OKwj2qRaOnsUBJqogIu8tw ]]>  </key>  <method>  <![CDATA[ user\_show ]]>  </method>  </Request>  <user>  <id>24347448</id>  <name>Aditya</name>  <user\_name/>  <link>  <![CDATA[ https://www.goodreads.com/user/show/24347448-aditya ]]>  </link>  <image\_url>  <![CDATA[ https://s.gr-assets.com/assets/nophoto/user/u\_111x148-9394ebedbb3c6c218f64be9549657029.png ]]>  </image\_url>  <small\_image\_url>  <![CDATA[ https://s.gr-assets.com/assets/nophoto/user/u\_50x66-632230dc9882b4352d753eedf9396530.png ]]>  </small\_image\_url>  <about/>  <age/>  <gender/>  <location>Bokaro, 38, India</location>  <website/>  <joined>09/2013</joined>  <last\_active>08/2020</last\_active>  <interests/>  <favorite\_books/>  <favorite\_authors></favorite\_authors>  <updates\_rss\_url>  <![CDATA[ https://www.goodreads.com/user/updates\_rss/24347448?key=C8SPjMuMzL9FlRwPoCq7\_58ipR\_7C2MicofG2n0E1H\_\_Oya5 ]]>  </updates\_rss\_url>  <reviews\_rss\_url>  <![CDATA[ https://www.goodreads.com/review/list\_rss/24347448?key=C8SPjMuMzL9FlRwPoCq7\_58ipR\_7C2MicofG2n0E1H\_\_Oya5&shelf=%23ALL%23 ]]>  </reviews\_rss\_url>  <friends\_count type="integer">1</friends\_count>  <groups\_count>0</groups\_count>  <reviews\_count type="integer">0</reviews\_count>  <user\_shelves type="array">  <user\_shelf>  <id type="integer">78774953</id>  <name>read</name>  <book\_count type="integer">0</book\_count>  <exclusive\_flag type="boolean">true</exclusive\_flag>  <sort nil="true"/>  <order nil="true"/>  <per\_page type="integer" nil="true"/>  <display\_fields/>  <featured type="boolean">true</featured>  <recommend\_for type="boolean">false</recommend\_for>  <sticky type="boolean" nil="true"/>  </user\_shelf>  <user\_shelf>  <id type="integer">78774952</id>  <name>currently-reading</name>  <book\_count type="integer">0</book\_count>  <exclusive\_flag type="boolean">true</exclusive\_flag>  <sort nil="true"/>  <order nil="true"/>  <per\_page type="integer" nil="true"/>  <display\_fields/>  <featured type="boolean">false</featured>  <recommend\_for type="boolean">false</recommend\_for>  <sticky type="boolean" nil="true"/>  </user\_shelf>  <user\_shelf>  <id type="integer">78774951</id>  <name>to-read</name>  <book\_count type="integer">0</book\_count>  <exclusive\_flag type="boolean">true</exclusive\_flag>  <sort nil="true"/>  <order>a</order>  <per\_page type="integer" nil="true"/>  <display\_fields/>  <featured type="boolean">false</featured>  <recommend\_for type="boolean">true</recommend\_for>  <sticky type="boolean" nil="true"/>  </user\_shelf>  </user\_shelves>  <updates type="array">  <update type="friend">  <action\_text>  <![CDATA[ is now friends with Rishabh Rishabh ]]>  </action\_text>  <link>https://www.goodreads.com/user/show/116176870-rishabh-rishabh</link>  <image\_url>https://s.gr-assets.com/assets/nophoto/user/u\_50x66-632230dc9882b4352d753eedf9396530.png</image\_url>  <actor>  <id type="integer">24347448</id>  <name>Aditya</name>  <image\_url>https://s.gr-assets.com/assets/nophoto/user/u\_50x66-632230dc9882b4352d753eedf9396530.png</image\_url>  <link>https://www.goodreads.com/user/show/24347448-aditya</link>  </actor>  <updated\_at>Sun, 31 May 2020 19:47:33 -0700</updated\_at>  </update>  </updates>  <user\_statuses></user\_statuses>  </user> </GoodreadsResponse> |

## Get a user's followers

API:

|  |
| --- |
| [https://www.goodreads.com/user/24347448/followers.xml?key={your](https://www.goodreads.com/user/24347448/followers.xml?key=%7Byour) key} |

Return Type: XML

Sample:

|  |
| --- |
| <GoodreadsResponse> <Request> <authentication>false</authentication> <method> <![CDATA[ ]]> </method> </Request> <user/> <followers start="0" end="0" total="0"> </followers> </GoodreadsResponse> |

## Get people a user is following:

API:

|  |
| --- |
| https://www.goodreads.com/user/24347448/following.xml?key={your key} |

Return Type: XML

Sample:

|  |
| --- |
| <GoodreadsResponse> <Request> <authentication>false</authentication> <method> <![CDATA[ ]]> </method> </Request> <user/> <following start="0" end="0" total="0"> </following> </GoodreadsResponse> |

# Setting the Python script to handle API

1. Open up the command prompt and navigate to your project directory.
2. Create a virtualenv in this directory

|  |
| --- |
| virtualenv venv |

1. Activate your virtualenv environment.

**In Mac :**

|  |
| --- |
| source venv/bin/activate |

**In Windows:**

|  |
| --- |
| .\venv\Scripts\activate |

1. Install required libraries.

We can do it in the beginning and also we can do it while creating our scripts.  
To do it in the beginning itself, open up the [requirements.txt](https://github.com/aditya109/goodsread-api-project/blob/master/requirements.txt) from my repository. Copy them down into your project folder.

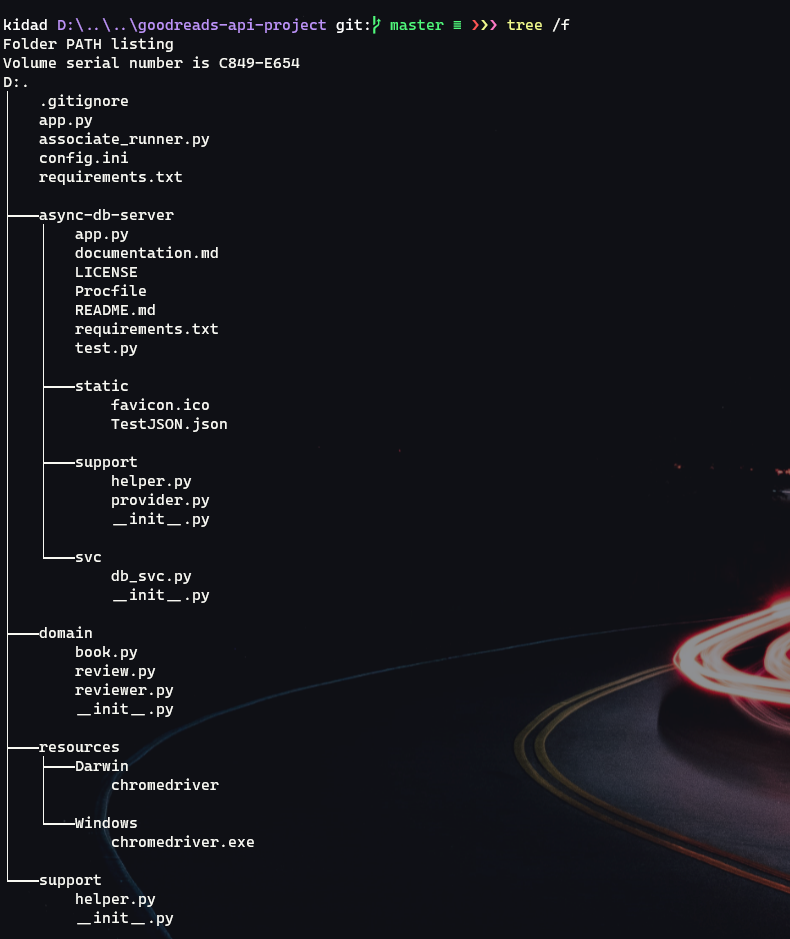
Run :

|  |
| --- |
| pip install -r requirements.txt |

1. Try running your scripts

|  |
| --- |
| cd web-scrapper-goodreads-api python app.py |

# Project Explanation



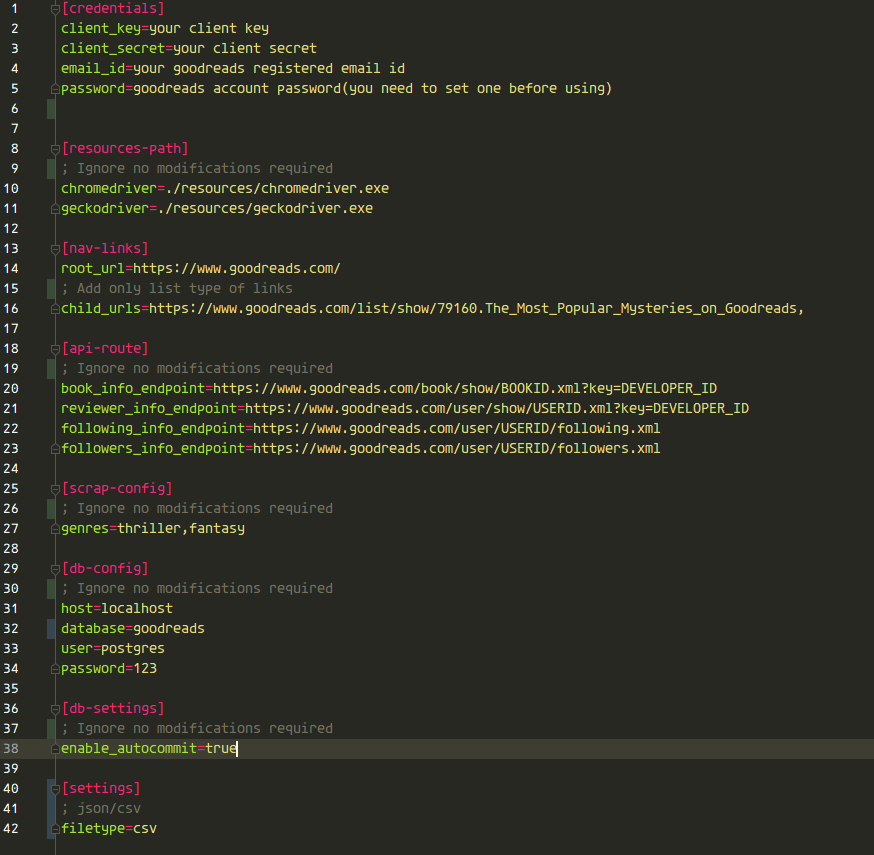
1. First let me explain the file structure.

* The app.py and associate\_runner.py are the basic files which run the show.
* config.ini is the place where you put your credentials and the external environment variables are placed.
* requirements.txt contain our dependencies
* domain contains the POPO(Plain Old Python Objects) classes and their builders
* resources contain the web drivers for Selenium
* support contains independent functions for providing a support structure to the program

**Note :**

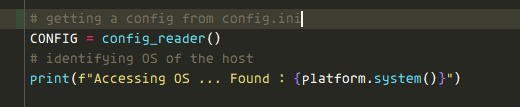
**async**-db-server contain plugin options for future implementation of async database server using PostgreSQL

## config.ini - Our Treasury

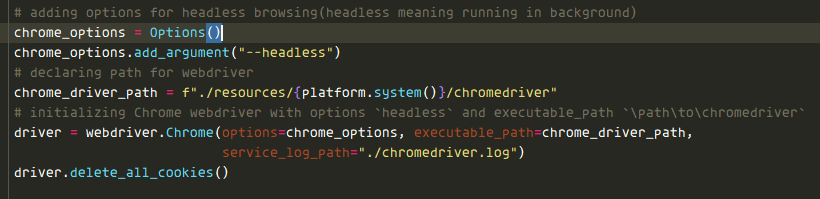


## app.py - Sole controller of our project

First we need to get our credentials from config.ini



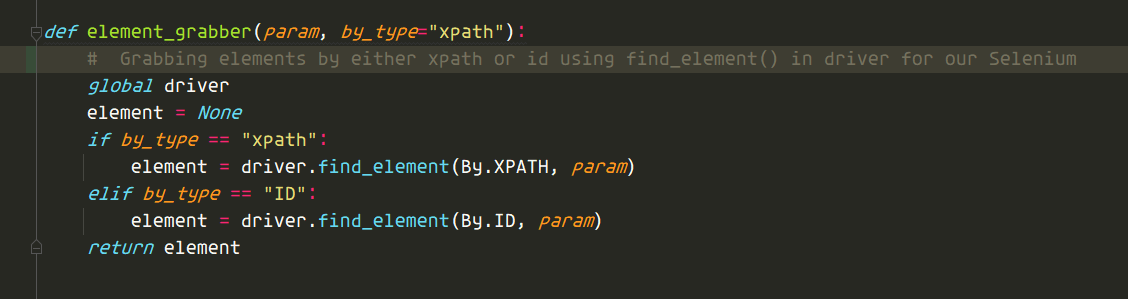
Then we need to open a Chrome browser in headless mode. So we initialize and configure the webdriver accordingly and make sure to delete existing cookies beforehand.



Then we create our output files using a function from helper.py and initialising if they are CSVs

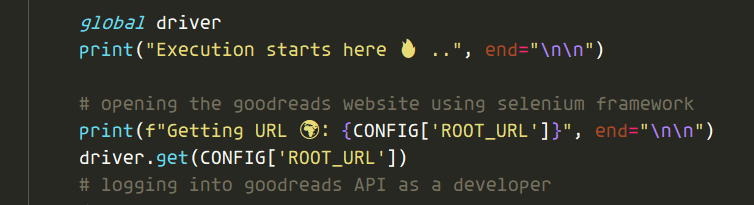


Then we add a function for grabbing element off page html



Next we have our link\_navigator() which is a really huge function in itself.  
So let’s break it up.

First we grab, our **ROOT\_URL** in config.ini



Then we log in as a developer, so use Selenium to do that.

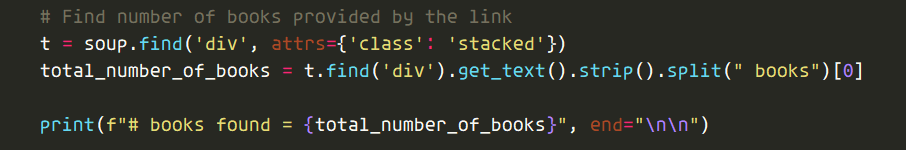


Then when we login, we iterate through all the **CHILD\_URL**s present in our config.py.

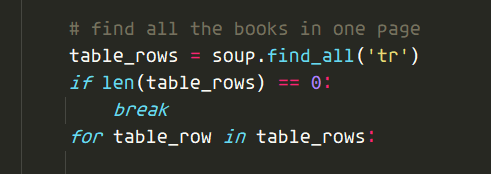
Once we grab a link, this code provides pagination for us.



We grab the total # of books which could be provided by the link



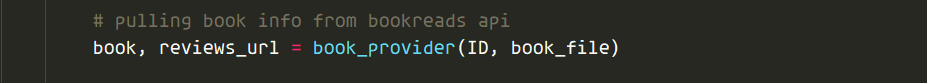
Then we grab all the books from the table rows in the page



Using the given code, for our safety net we grab the book title, book id and first author name.

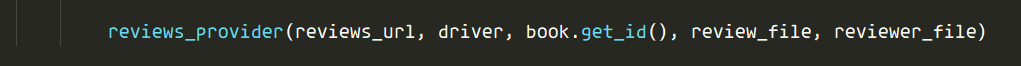


Provides the book file pointer along the book id to the book\_provider() from associate\_runner.py

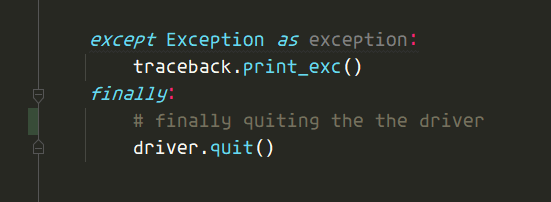


Some links do not have isbn numbers with them, causing link failure consequently disabling us to find details about the reviewer, so we explicitly imbibe the ISBN into it using regular expressions.



We write the reviews object and reviewer object to the corresponding files using chained function reviews\_provider(), another functions from associate\_runner.py

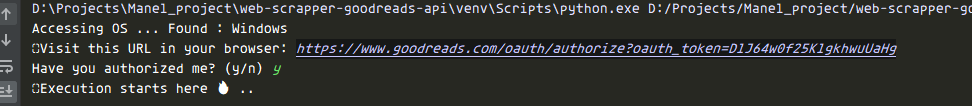
Finally we quit the driver.



If you were understood the above, understanding \_\_main\_\_ is trivial



Only point worth noting is when you run the app.py, this shows up

You need to click it and register the app and then click y.

Don’t lie to it, the script will know, literally 🤣😂  
Don’t believe me, try entering y, without opening the link.  
THE COMPUTER KNOWS !

## associate\_runner.py - Support beam for app.py

* book\_provider - extracts a book details using goodreads API and writes to the specified file, returns the book object and reviews\_url
* reviews\_provider - extracts a review details using goodreads API and headless web scraping, also calls reviewer\_provider and writes to the specified files
* reviewer\_provider -extracts a reviewer details using goodreads API returns the reviewer object
* write\_book\_object\_to\_file
* write\_review\_object\_to\_file
* Write\_reviewer\_object\_to\_file

**The rest of the files are self explanatory, I have provided ample comments.**

**But please ping me if you require assistance or Get stuck at any point.**

# Why did I remove the Database off the project ?

So the reason is Goodreads.API takes way too long to respond and OAuth slows it down too much.  
Hence, the dataset took a long time to process.  
So instead, I tried cutting the database off the project and making a separate server.  
Although that led to increase to CPU utilisation but network latency increased, since now we were performing network calls(even though it was in our own local machine).  
One possible solution could be implementing a server using asyncio.  
I have left an implemented plugin in the async\_db\_server for you. Please try it and if it works for you, I will help you to implement it.  
Right now the project will either process the dataset as json or csv.

Please feel free to ask me any questions regarding this project and anything related to my knowledge.  
  
Feel free to contact me:

[LINKED IN](https://www.linkedin.com/in/aditya109/)

[EMAIL](mailto:adikid1996@gmail.com)

Hope you had fun learning  
Regards,

Aditya