CHAPTER 5

FoCUS: Learning to Crawl Web Forums

In this chapter, This chapter contains five (5) sections explaining the experiment: Section 5.1 FoCUS Framework, Section 5.2 VIPS Classes, Section 5.3 Interactivity of Classes in VIPS Sequence Diagram, Section 5.4 Execution Session, Section 5.5 Result Section 5.6 Evaluation and Discussion and Section 5.7 Conclusion.

5.1 FoCUS Framework

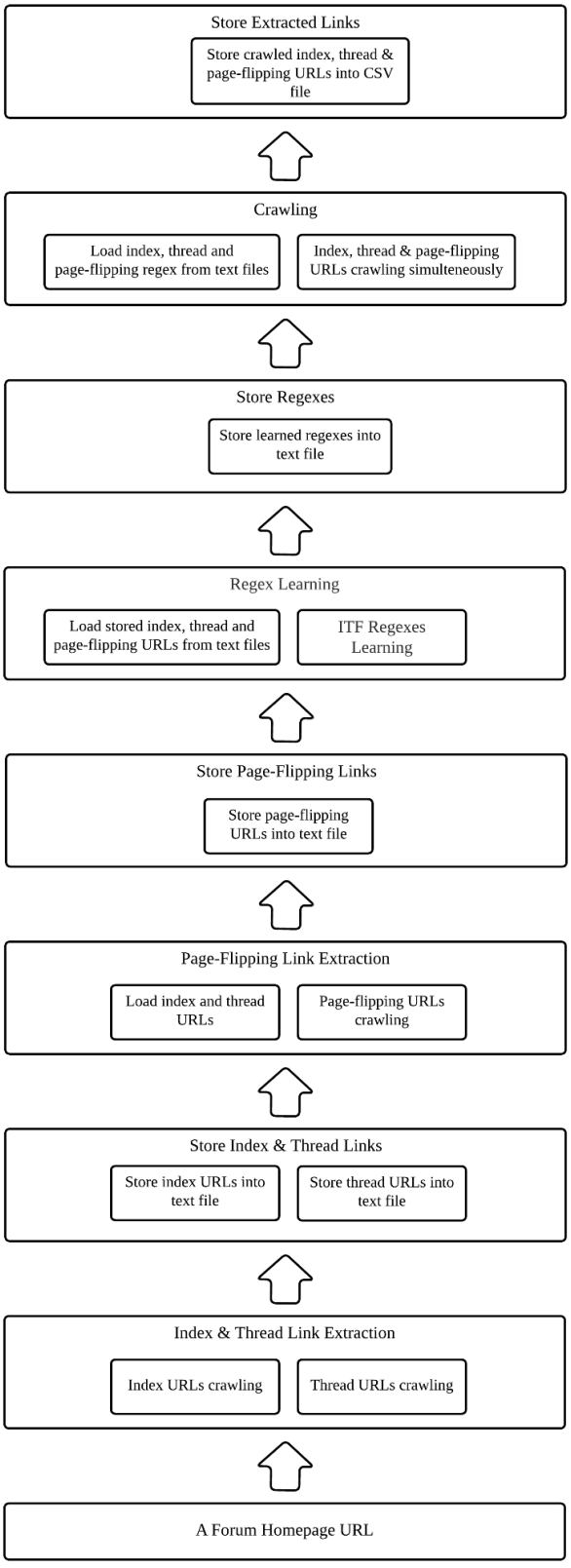


Figure 5.1: FoCUS Layered Framework

Figure 5.1 presents the design of the FoCUS layered framework that contains 9 layers. The 9 layers include A Forum Homepage URL, Link Extraction, Store Extracted Links, Regex Learning, Store Regexes, Crawling and Store Extracted Links. Each layer of this framework will handle different tasks that ensure the final results are achieved.

5.1.1 A Forum Homepage URL

First of all, the design of the flow of framework begins by accepting a web forum homepage URL. The web forum homepage URL is chosen because the homepage is the root page of all subsequent forum pages which means homepage is the entry page that is the lowest common ancestor of all index pages and thread pages in a web forum.

5.1.2 Index & Thread Link Extraction

After we get the homepage URL of a web forum, the program will start crawling the index URLs and thread URLs that exist in the web forum.

5.1.3 Store Index & Thread Links

After crawling the index and thread URLs, all the crawled index URLs and thread URLs will be saved in 2 different text files respectively.

5.1.4 Page-Flipping Link Extraction

At this layer, the program will first retrieve all the saved index and thread URLs from the text files that we have created just now to continue to crawl all the page-flipping URLs that exist within these index and thread URLs.

5.1.5 Store Page-Flipping Links

After crawling the page-flipping URLs, all the crawled page-flipping URLs will be saved in a text file as well.

5.1.6 Regex Learning

At this layer, the program will first retrieve all the saved index, thread and page-flipping URLs from the text files and then learn the link patterns of an index URL, thread URL and page-flipping URL. The program will then create the regex pattern for index URL, thread URL as well as page-flipping URL

5.1.7 Store Regexes

After creating the regex pattern for index URL, thread URL and page-flipping URL, all these regex patterns will be saved in a text file.

5.1.8 Crawling

At this layer, there will be another crawler we have to use to crawl all the index, thread and page-flipping URLs, which we need to retrieve all the regex patterns that we have just saved in the text files and feed it to this crawler. Then, we need to pass a homepage URL of a forum again to this crawler, it will automatically crawl all index, thread and page-flipping URLs of the web forum that we have passed in.

5.1.9 Store Extracted Links

After the crawler is done crawling, all the crawled index, thread and page-flipping URLs will be saved in a CSV file.

5.2 FoCUS System Block Diagram

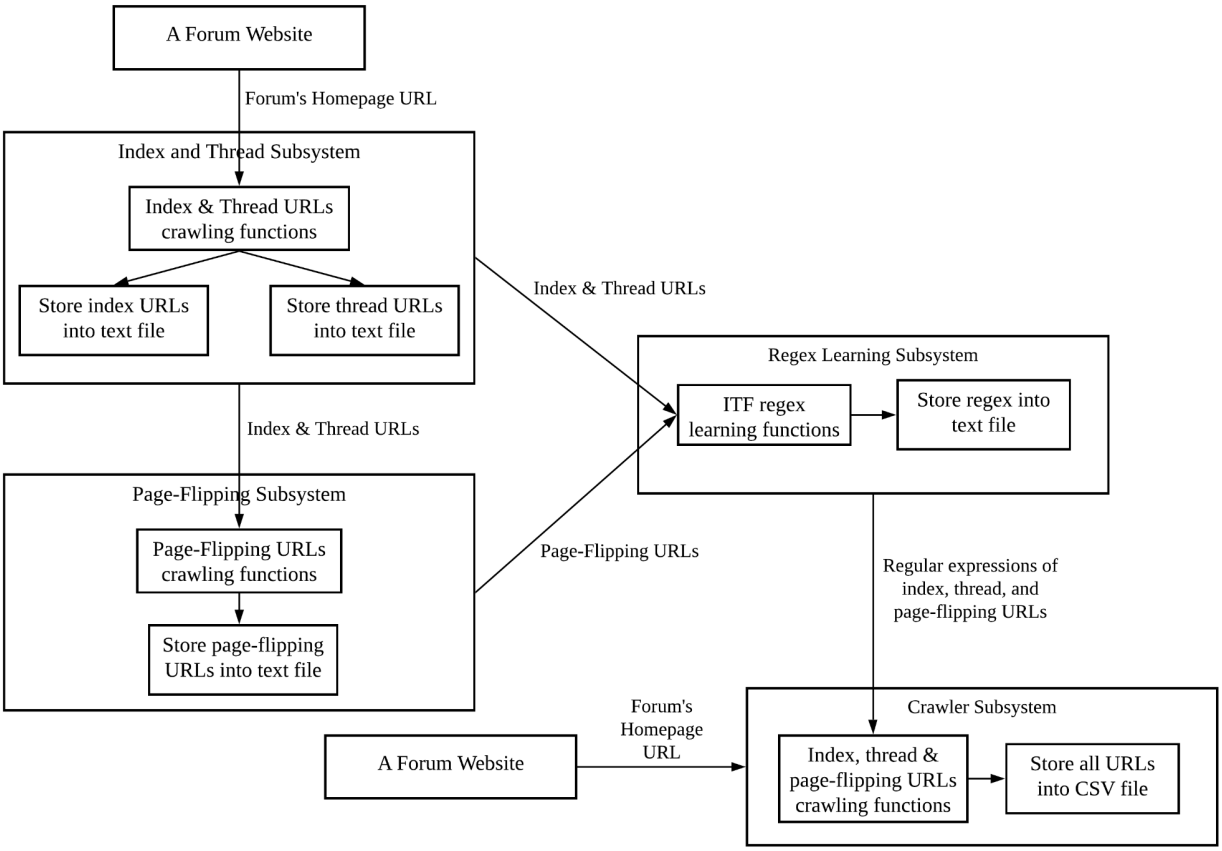


Figure 5.2: System Block Diagram

Figure 5.2 presents the system block diagram for the FoCUS framework. From the diagram above, we can see that there are a total of 4 subsystems to form this FoCUS framework. The subsystems are *Index and Thread Subsystem*, *Page-Flipping Subsystem*, *Regex Learning Subsystem* and *Crawler Subsystem*.

5.2.1 Index and Thread Subsystem

First of all, this subsystem will accept a homepage url from a forum website. Therefore, we have used the homepage of SixCrazyMinutes forum website (http://www.sixcrazyminutes.com/forums) to act as the entry page and pass it into the system. After the homepage url of a forum website is passed in, this subsystem will start crawling index and thread URLs that present in this forum. After crawling, all the index URLs and thread URLs will be stored in a different text file respectively, namely *index_url.txt* and *thread_url.txt*.

5.2.2 Page-Flipping Subsystem

After that, the page-flipping subsystem will retrieve all index URLs and thread URLs stored in index.txt and thread.txt respectively. After retrieving URLs from the text files, this subsystem will start to crawl all the page-flipping URLs that may exist in each index and thread URL just retrieved. When done crawling, all the page-flipping URLs will be stored in a text file as well, namely *page_flipping_url.txt*.

5.2.3 Regex Learning Subsystem

Later, the regex learning subsystem will retrieve all index URLs, thread URLs and page-flipping URLs that have been crawled and stored in different text files just now. After retrieving URLs from the text files, this subsystem will start learning the patterns of an index URL, thread URL and page-flipping URL. Then, it will find out what the regex pattern of an index URL link, thread URL link and page-flipping URL link look like and the regular expression patterns of these three different types of URL links will also be saved in a text file, namely *URL_Regex.txt*.

5.2.4 Crawler Subsystem

Finally, the forum crawler subsystem comes into the role. This subsystem is the crawler that is built using a free and open-source web-crawling framework called Scrapy. It first will retrieve all the regular expressions of index, thread and page-flipping URLs and utilize these regular expressions to crawl all the index links, thread links, and page-flipping links of a forum website. Before this, we have used the SixCrazyMinutes forum website to train all the necessary regular expressions, so now we can pass in the homepage url of the SixCrazyMinutes web forum again to this crawler and this crawler will automatically help to crawl all of the index links, thread links and page-flipping links that exist in this forum website as much as possible and saved the results into a CSV file called *FinalOutput.csv*. The good thing about this crawler is that it can crawl other web forums as well if other web forums' link structure are similar to the SixCrazyMinutes forum. For example, this crawler also successfully crawled https://www.namepros.com/, https://www.gardenstew.com/ and so on. The user just needs to enter the homepage URL of these

forums and the crawler will directly crawl all of the index, thread and page-flipping URLs based on the regular expressions learned.

5.3 FoCUS Data Flow Diagram (DFD)

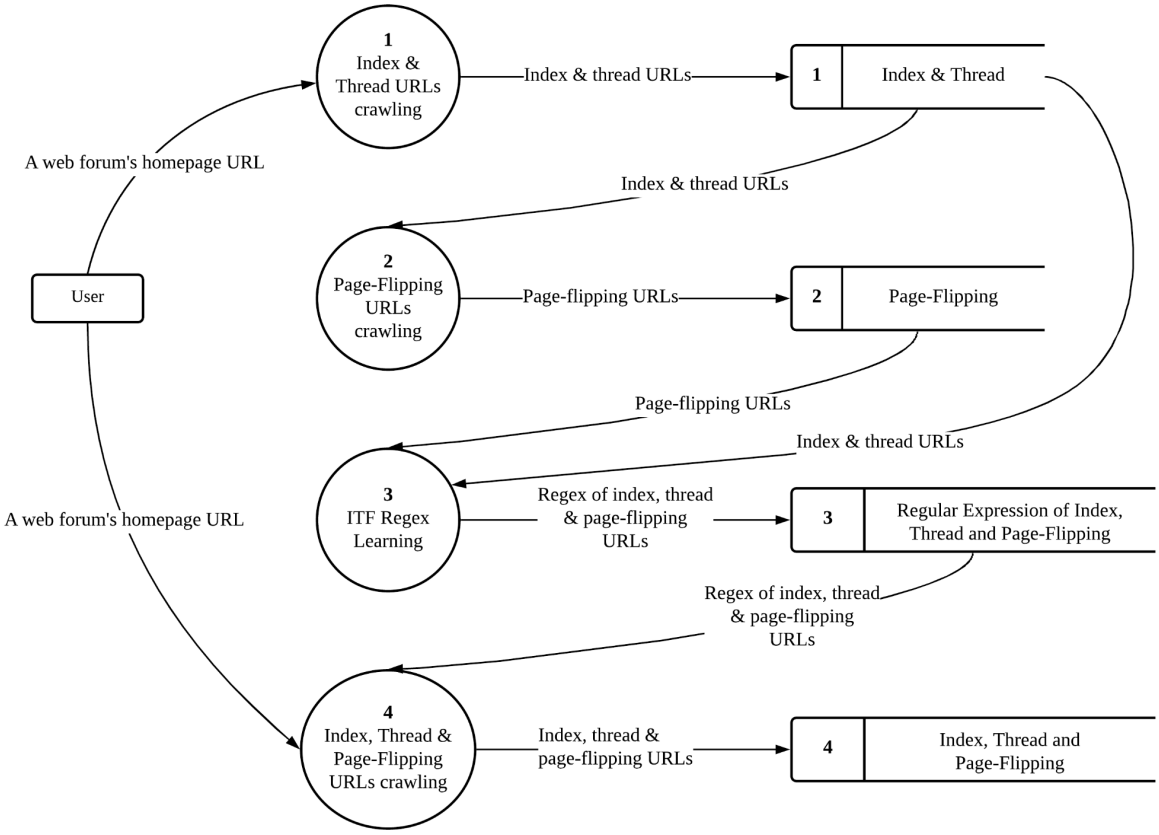


Figure 5.3: FoCUS Data Flow Diagram

Figure 5.3 presents the data flow diagram for the FoCUS web forum crawler. From the diagram above, we can see that there is one external entity, 4 processes and 4 data stores. Below is the diagram that describes the meaning of each shape in the data flow diagram.

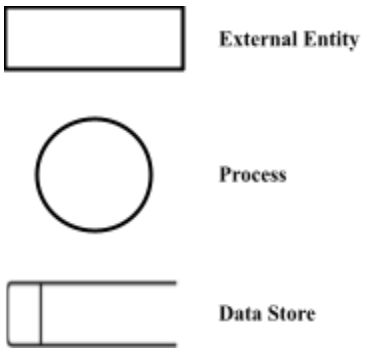


Figure 5.4: Shape meaning in Data Flow Diagram

5.3.1 External Entity: User

Moving forward, we can see that there is an **external entity** called **user** in the data flow diagram. User is the one who uses this program or system to crawl data from a web forum. First of all, the user is required to enter or provide the homepage URL of the web forum that he/she wants to crawl to the program, which the web forum's homepage URL will be passed to the **process 1**. For example, the homepage of SixCrazyMinutes forum website (http://www.sixcrazyminutes.com/forums)

5.3.2 Process 1: Index & Thread URLs crawling

In **Process 1**, the program will receive the web forum's homepage url provided by the user. It will then start crawling all the index and thread URLs that exist in the web forum. The program will crawl all the index URLs first and then crawl all the thread URLs that exist in each index URLs that have just crawled.

5.3.3 Data Store 1: Index & Thread

After the **process 1** is done, the **first data store** comes into play, where all index URLs and thread URLs will be stored in a different text file respectively, namely *index_url.txt* and *thread_url.txt*.

5.3.4 Process 2: Page-Flipping URLs crawling

In **Process 2**, the program will retrieve the crawled results of process 1 from the data store 1. After retrieving all the index and thread URLs, the program will start to crawl all the page-flipping URLs that may exist in these index and thread URLs.

5.3.5 Data Store 2: Page-Flipping

After the **process 2** is done, the **second data store** comes into play, where all the page-flipping URLs will be stored in a text file, namely *page_flipping_url.txt*.

5.3.6 Process 3: ITF Regex Learning

In **Process 3**, the program will retrieve all index URLs, thread URLs and page-flipping URLs from data store 1 and 2. Then, it will start learning the patterns of an index URL, thread URL and page-flipping URL, which it will find out what the regex pattern of an index URL link, thread URL link and page-flipping URL link look like and create regular expressions for each of these URLs.

5.3.7 Data Store 3: Regular Expression of Index, Thread and Page-Flipping

After the **process 3** is done, the **third data store** comes into play, where the regular expression patterns of index, thread and page-flipping URLs will be stored in a text file called *URL_Regex.txt*.

5.3.8 Process 4: Index, Thread & Page-Flipping URLs crawling

In **Process 4**, the program will retrieve all the regular expressions of index URLs, thread URLs and page-flipping URLs from data store 3 and utilize these regular expressions to crawl all the index links, thread links, and page-flipping links of a forum website. Before this, we have used the SixCrazyMinutes forum website to train all the necessary regular expressions, so now we can pass in the homepage url of the SixCrazyMinutes web forum again to this crawler and then this crawler will automatically help the user to crawl all of the index links, thread links and page-flipping links that exist in this forum website as much as possible. The user also can crawl other web forums as well if other chosen web forums' link structure are similar to the SixCrazyMinutes forum. For example, this crawler also successfully crawled the forum https://www.namepros.com/, https://www.gardenstew.com/ and so on. The user just needs to enter the homepage URL of these forums and the crawler will directly go to **process 4** to crawl all of the index, thread and page-flipping URLs based on the regular expressions learned.

5.3.9 Data Store 4: Index, Thread & Page-Flipping

After the **process 4** is done, the **forth data store** comes into play, where all of the index, thread and page-flipping URLs of a web forum will be saved in a CSV file called *FinalOutput.csv*.

5.4 Flowchart

In this project, the FoCUS crawler is made up of 5 python scripts. The 5 python scripts are:

1. **main.py** (python script that gather the main functions in other python scripts)
2. **Index_Thread.py** (python script that used to crawl all the index and thread links)
3. **Page_Flipping.py** (python script that used to crawl all the page-flipping links)
4. **ITF_Learning.py** (python script that used to learn the link structure of index, thread and page-flipping and create regular expression)
5. **ForumCrawler.py** (python script that used to crawl all of the index, thread and page-flipping links using the regex learned).

5.3.1 main.py

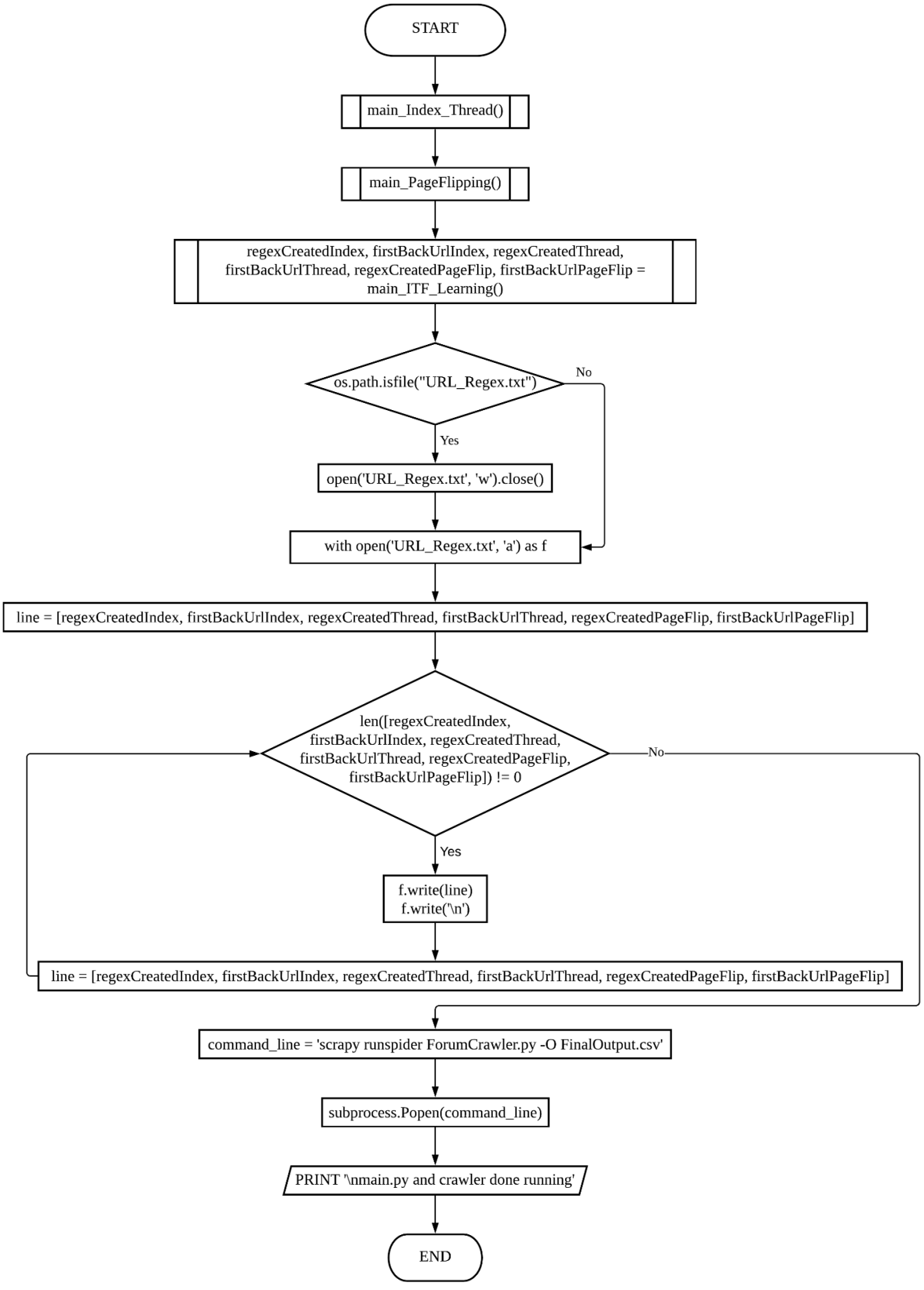


Figure 5.5: Flowchart of main.py

The above figure shows the program flow of *main.py* script. We can see that the *main.py* calls the other main functions that exist in other python scripts. For example, the *main.py* script calls the `main_Index_Thread()`, `main_PageFlipping()` and `main_ITF_Learning()` functions one after another in the beginning. After crawling and learning, *the main.py* script will store the regular expressions returned by `main_ITF_Learning()` into a text file called *URL_Regex.txt*. After that it will open up the terminal automatically and use the command assigned to it to run *ForumCrawler.py*.

1. main_Index_Thread()

Description	Handle the index & thread URLs crawling as well as save the crawled URLs into text files.
Method Type	static
Argument	Do not require argument
Return	Do not return anything
Example	`main_Index_Thread()`
Example of return value	Do not return anything

2. main_PageFlipping()

Description	Handle the page-flipping URLs crawling and save the crawled URLs into text files.
Method Type	static
Argument	Do not require argument
Return	Do not return anything
Example	`main_PageFlipping()`
Example of return value	Do not return anything

3. main_ITF_Learning()

Description	Handle the index, thread and page-flipping URLs regular expression learning and.
Method Type	static

Argument	Do not require argument
Return (`str`)	Return 6 different regular expressions in string type.
Example	`regexCreatedIndex, firstBackUrlIndex, regexCreatedThread, firstBackUrlThread, regexCreatedPageFlip, firstBackUrlPageFlip = main_ITF_Learning()`
Example of return value	`regexCreatedIndex:` ^([a-zA-Z0-9]+-[a-zA-Z0-9]+)+\.\d+$ `firstBackUrlIndex:` forums `regexCreatedThread:` ^([a-zA-Z0-9]+-[a-zA-Z0-9]+)+\.\d+$ `firstBackUrlThread:` threads `regexCreatedThread:` ^page-\d+$ `firstBackUrlThread:` (forums|threads)

5.3.2 Index_Thread.py

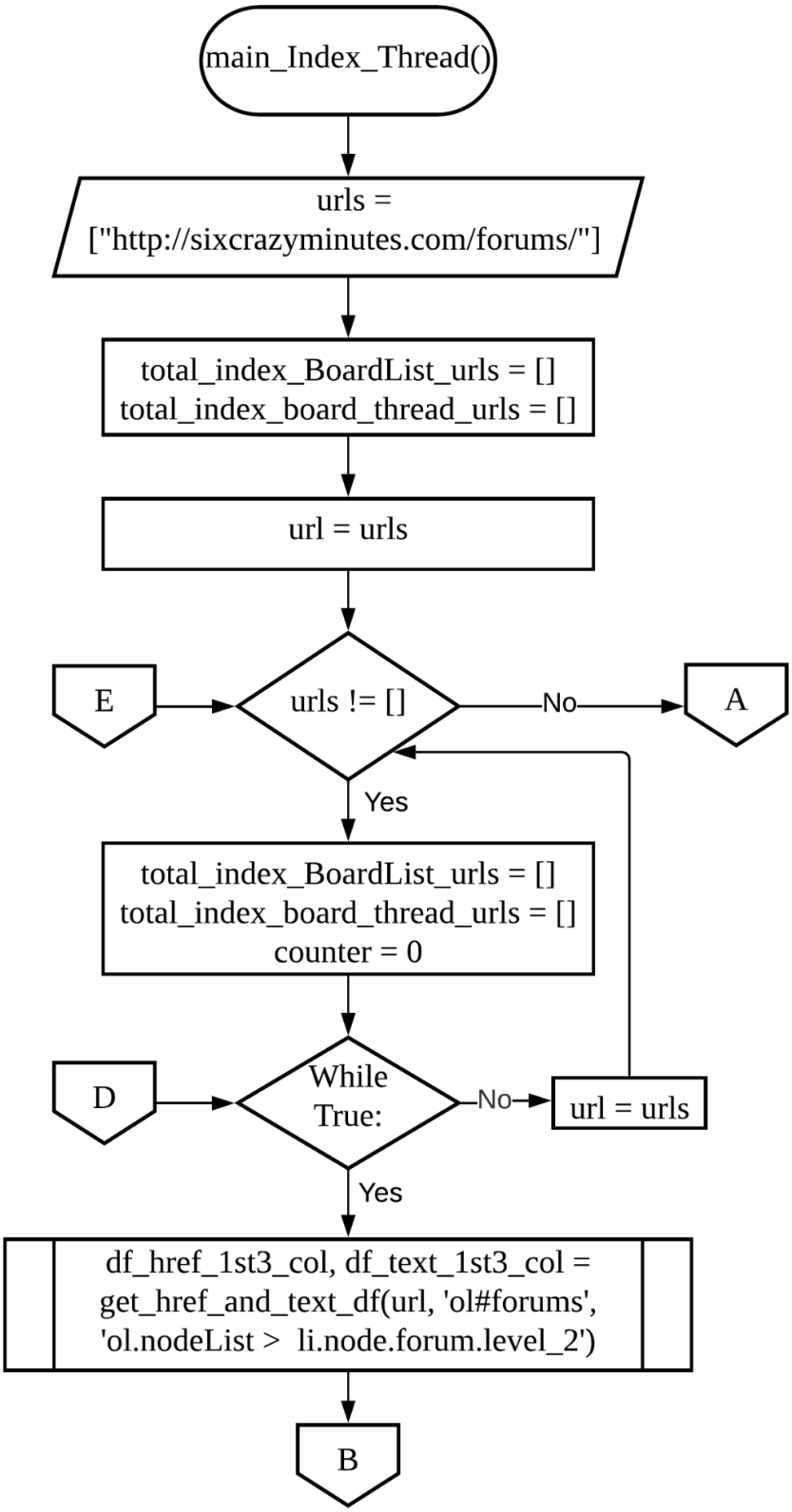


Figure 5.6: Flowchart of Index_Thread.py

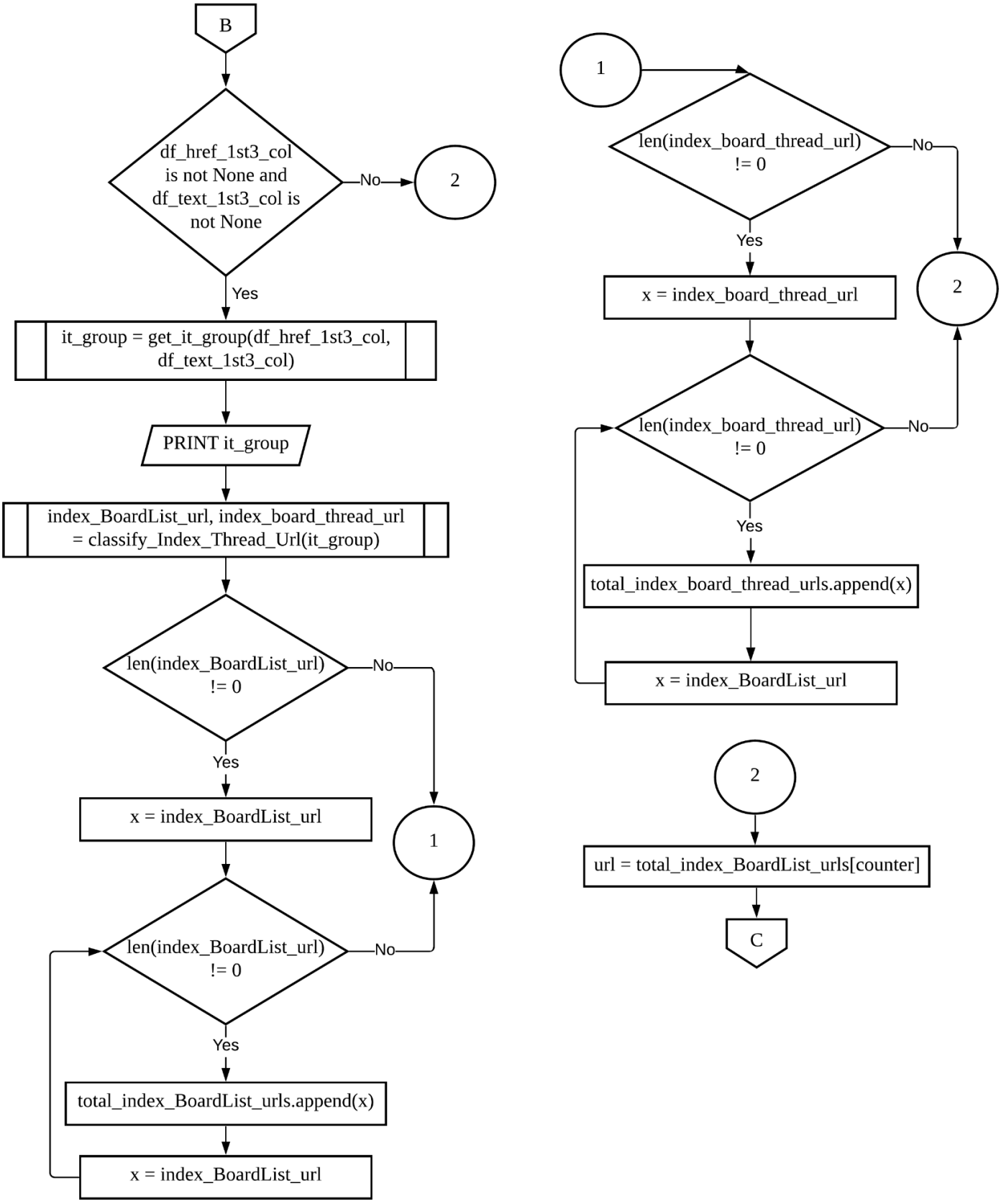


Figure 5.7: Flowchart of Index_Thread.py

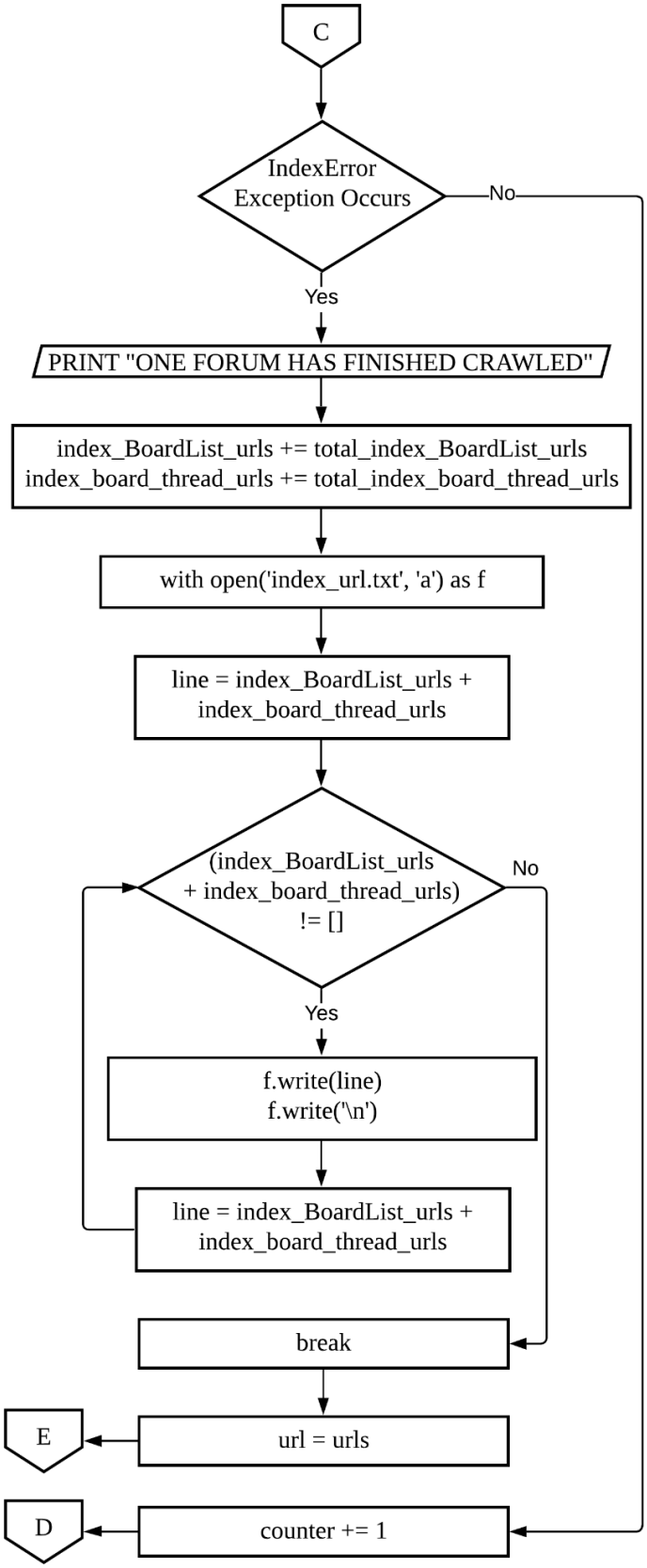


Figure 5.8: Flowchart of Index_Thread.py

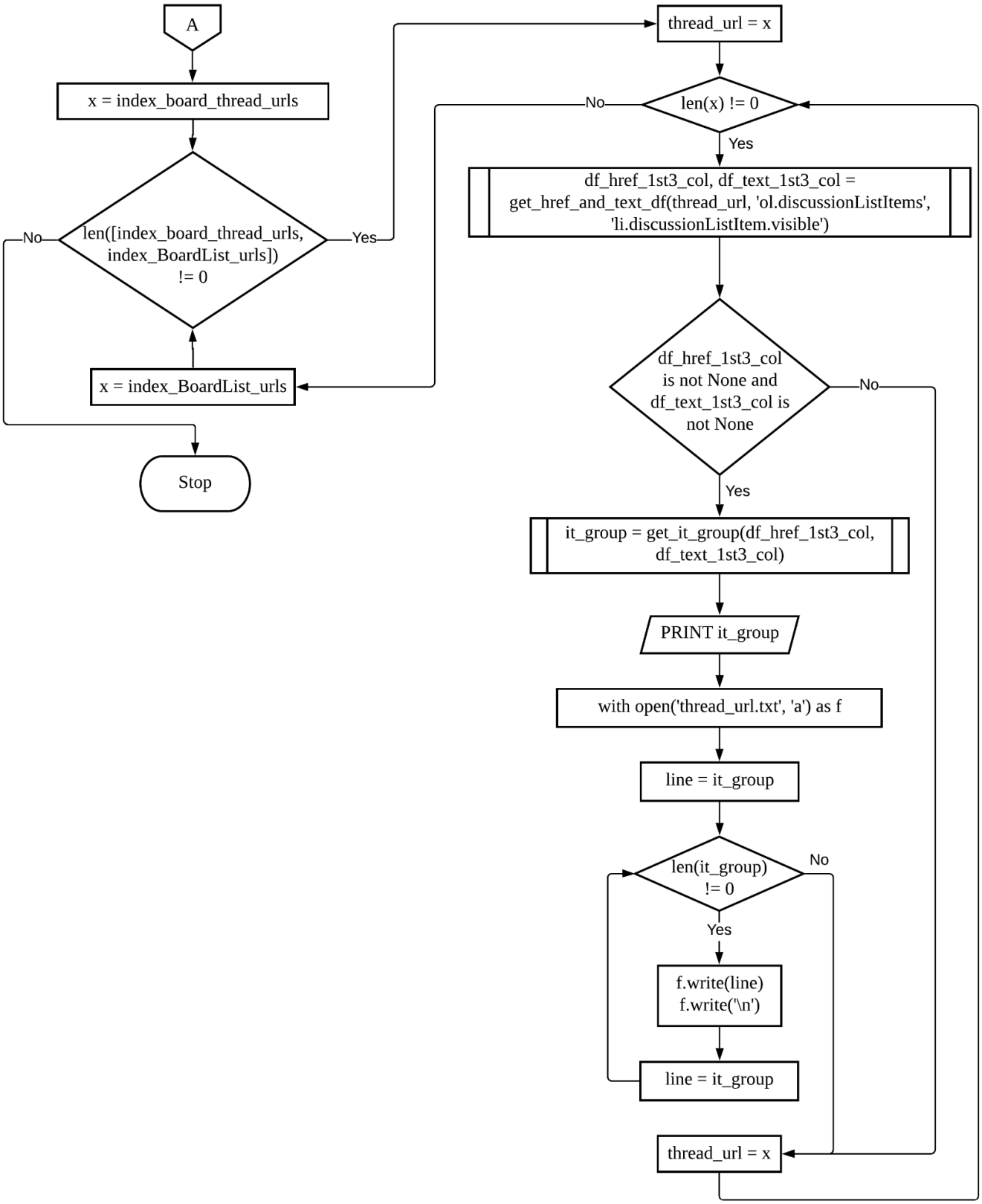


Figure 5.9: Flowchart of Index_Thread.py

5.3.2.1 Functions used in Index_Thread.py

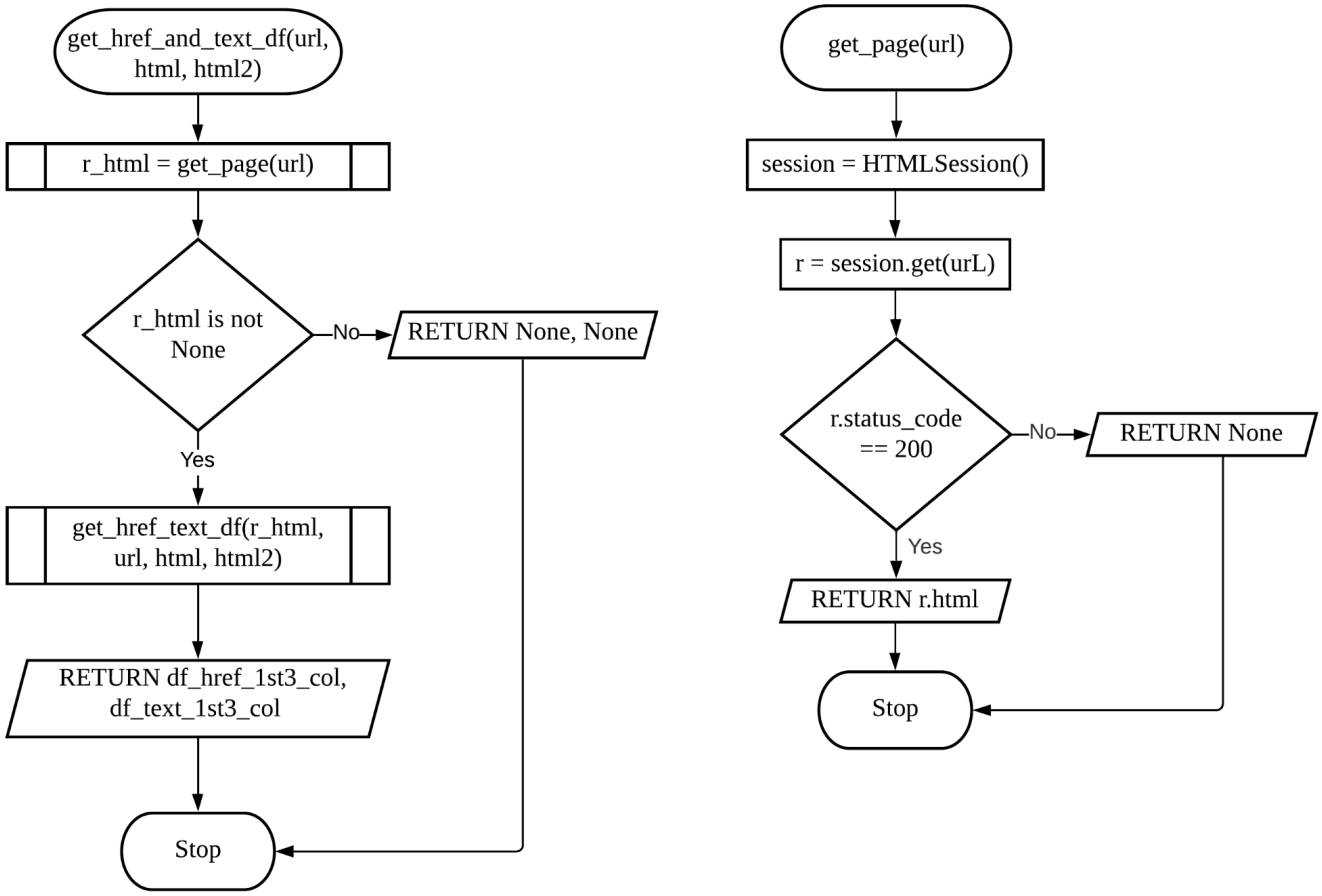
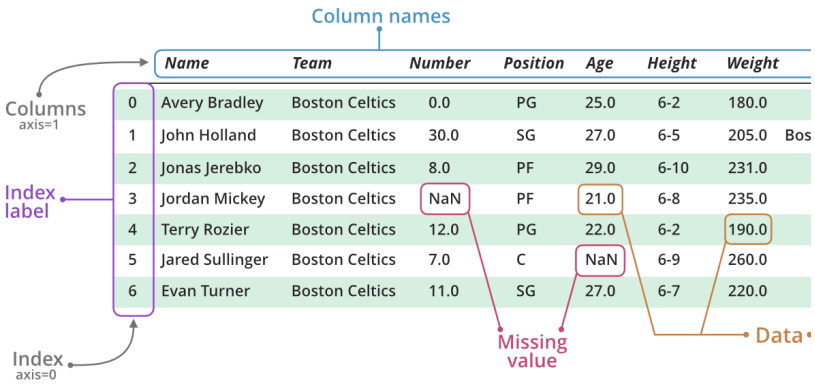


Figure 5.10: Flowchart of Index_Thread.py

1. get_href_and_text_df(url, html, html2)

Description	Return 2 dataframes that contains 3 columns and having the same number of rows. `df_href_1st3_col` contains only links. `df_text_1st3_col` contains only anchor text. Each anchor text corresponds to the link in `df_href_1st3_col`.
Method Type	static
Argument	`url` = A web forum homepage's or entry page's URL `html` = html that wraps all different topics

	`html2` = html that represent each topic
Return (`pd.DataFrame`)	Return 2 dataframes
Example	`df_href_1st3_col, df_text_1st3_col = get_href_and_text_df(url, 'ol#forums', 'ol.nodeList > li.node.forum.level_2')`
Example of return value	 **Reference only*

2. get_page(url)

Description	Return a request-html object that can be utilzed in scraping data on HTML document.
Method Type	static
Argument	`url` = A web forum homepage's or entry page's URL
Return (`request-html`)	Return a request-html object
Example	`r_html = get_page(url)`
Example of return value	`r_html: <HTML url='http://www.sixcrazyminutes.com/forums/'>`