Internal Linking Tool and ranking analyser

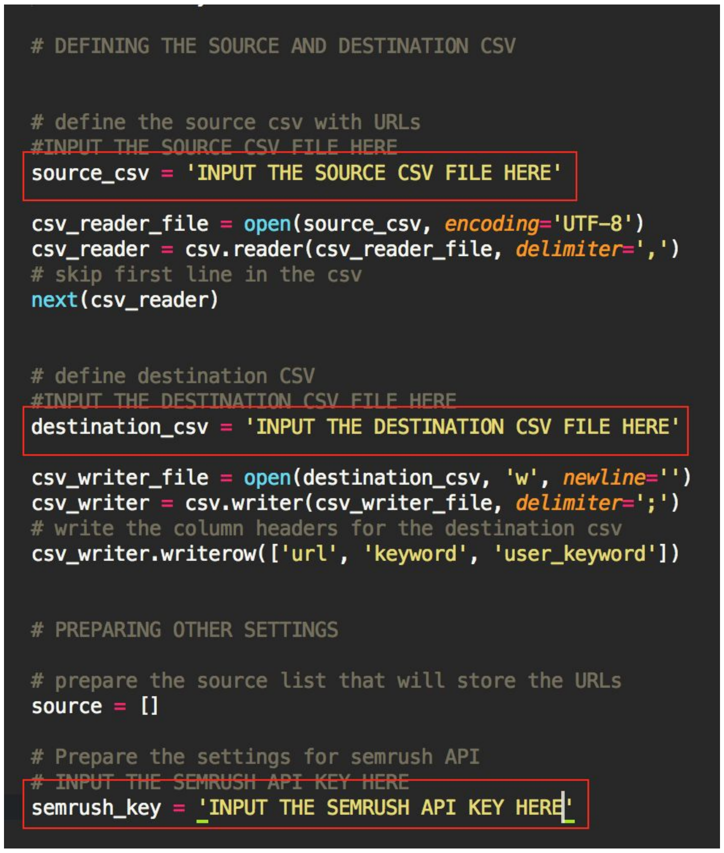
PART I

1. **The user provides a number of the most important pages from a website.**
    - Open an empty Excel or Numbers file and input the URLs you want analysed in a column, starting with row number 2 (this is important as the first row in the file will not be analysed)
    - Save the the file as a CSV file and store it in the same location as the Python file with the code.

2. **Open the Python file with the code called TypeATool_Keywords.py.**
    - You may need to download SublimeText (https://www.sublimetext.com/3) but you may use another text editor if you prefer.

3. **Inside the code, you will have to input a few things (see below):**
    - **The exact name of the CSV source file.** The name of the file must be written in the appropriate location inside the code - the place where the variable is defined - the name of the file should be put in parentheses, indicating the type of object (string) **(source_csv = 'source.csv').** The file can be named however you want, but it must differ from the destination CSV file.
    - **The name of the destination CSV file**. The name of the file must be written in the appropriate location inside the code - the place where the variable is defined - the name of the file should be put in parentheses, indicating the type of object (string) **(destination_csv = 'destination.csv').** The file can be named however you want, but it must differ from the source CSV file.

- **The SemRush API key** where the variable is defined **(semrush_api='xyz')**

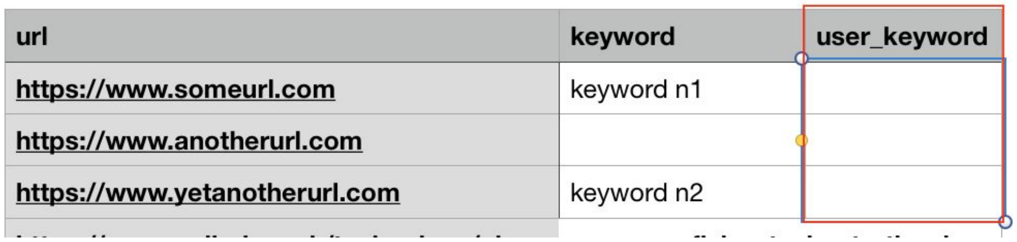


```
# DEFINING THE SOURCE AND DESTINATION CSV


# define the source csv with URLs
#INPUT THE SOURCE CSV FILE HERE
source_csv = 'INPUT THE SOURCE CSV FILE HERE'

csv_reader_file = open(source_csv, encoding='UTF-8')
csv_reader = csv.reader(csv_reader_file, delimiter=',')
# skip first line in the csv
next(csv_reader)


# define destination CSV
#INPUT THE DESTINATION CSV FILE HERE
destination_csv = 'INPUT THE DESTINATION CSV FILE HERE'

csv_writer_file = open(destination_csv, 'w', newline='')
csv_writer = csv.writer(csv_writer_file, delimiter=';')
# write the column headers for the destination csv
csv_writer.writerow(['url', 'keyword', 'user_keyword'])


# PREPARING OTHER SETTINGS

# prepare the source list that will store the URLs
source = []

# Prepare the settings for semrush API
# INPUT THE SEMRUSH API KEY HERE
semrush_key = 'INPUT THE SEMRUSH API KEY HERE'
```

4. **Using the SemRush API we programmatically extract 1 keyword per URL.**
   - The output is presented in the CSV document **(destination_csv = 'destination.csv')**
   - Open the destination.csv file (the file is stored in the same location as the source.csv file and the Python file.

5. **Review/correct/input the keywords that are not suitable or missing.**
   - Review/correct/input the keywords in the appropriate column and save the changes to the CSV using the same file name. (if you see any keywords missing or if some of the keywords are not suitable for the specific url, you can rewrite them)

- The keywords must be input in the column named **user_keyword** as shown below.



url	keyword	user_keyword
https://www.someurl.com	keyword n1	
https://www.anotherurl.com		
https://www.yetanotherurl.com	keyword n2	

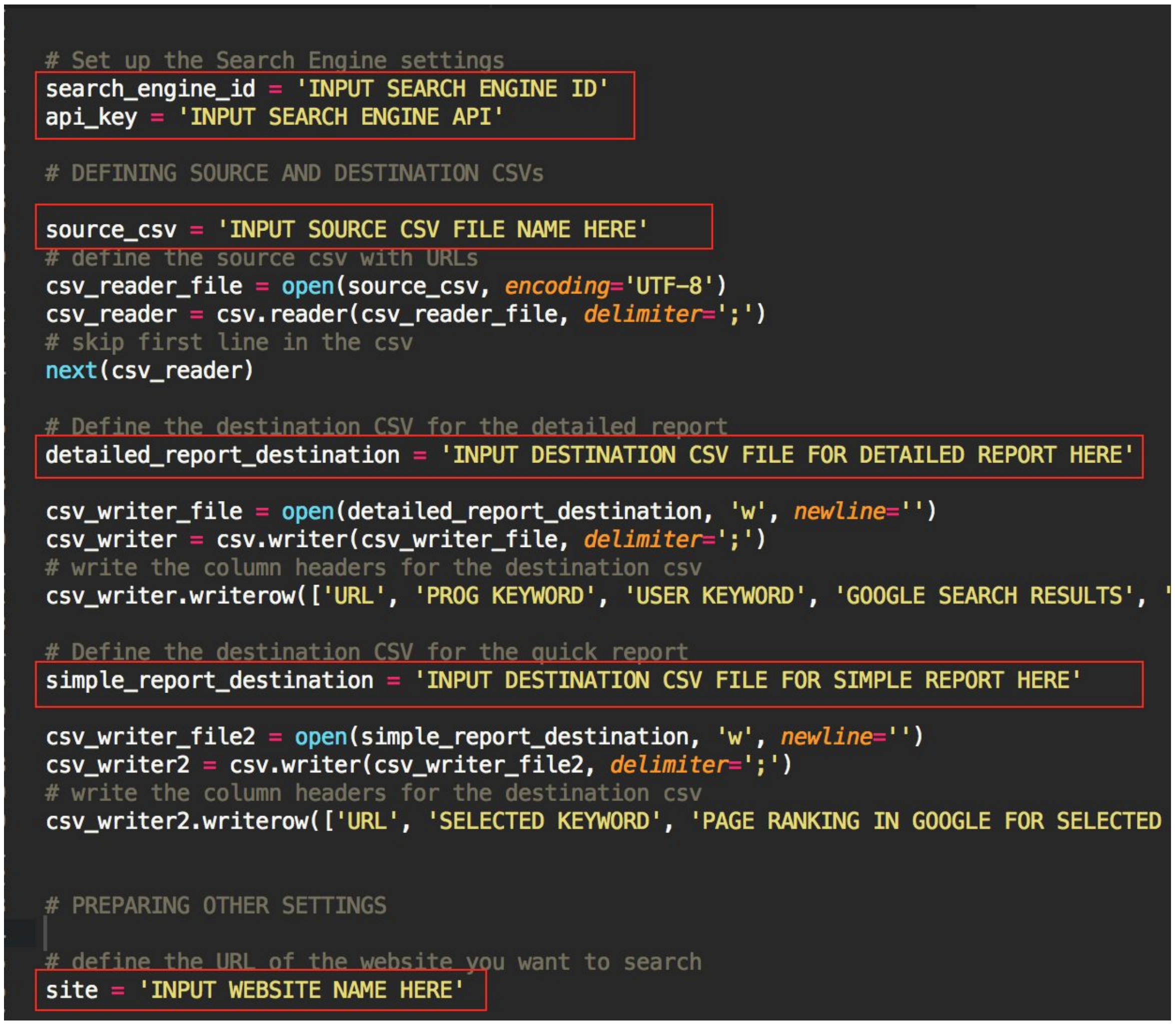
- This provides us with the starting point for the analysis: every URL is now associated with a keyword that it's supposed to rank for in Google

PART II

6. **Open the Python file with the code called TypeATool_GoogleSearch.py.**

7. **Inside the code, you will have to input a few things (see below):**
    - **Google Search Engine ID number (https://support.google.com/customsearch/answer/2649143?hl=en) and Google Search Engine API key (https://developers.google.com/maps/documentation/javascript/get-api-key)**
    - **The exact name of the CSV source file. The source file for this part of the analysis will be the destination file from part one - if you named the file that now contains all the URLs and keywords 'destination.csv' you should use this file for the source now.**
    The name of the file must be written in the appropriate location inside the code - the place where the variable is defined - the name of the file should be put in parentheses, indicating the type of object (string) (source_csv = 'destination.csv').

- **The name of the destination CSV files**. We will be creating 2 reports (a simpler one and a more complex one) this time, so we will need 2 destination CSV files to store the results.
  The names of the files must be written in the appropriate location inside the code - the place where the variable is defined - the names of the file should be put in parentheses, indicating the type of object (string) (simple_report_destination= ‘simple_report.csv’ and detailed_report_destination=‘detailed_report.csv’)**.**
  The files can be named however you want, but they must differ from one another and the source CSV file.
- **The website address you are analysing.**



```
# Set up the Search Engine settings
search_engine_id = 'INPUT SEARCH ENGINE ID'
api_key = 'INPUT SEARCH ENGINE API'

# DEFINING SOURCE AND DESTINATION CSVs

source_csv = 'INPUT SOURCE CSV FILE NAME HERE'
# define the source csv with URLs
csv_reader_file = open(source_csv, encoding='UTF-8')
csv_reader = csv.reader(csv_reader_file, delimiter=';')
# skip first line in the csv
next(csv_reader)

# Define the destination CSV for the detailed report
detailed_report_destination = 'INPUT DESTINATION CSV FILE FOR DETAILED REPORT HERE'

csv_writer_file = open(detailed_report_destination, 'w', newline='')
csv_writer = csv.writer(csv_writer_file, delimiter=';')
# write the column headers for the destination csv
csv_writer.writerow(['URL', 'PROG KEYWORD', 'USER KEYWORD', 'GOOGLE SEARCH RESULTS', '

# Define the destination CSV for the quick report
simple_report_destination = 'INPUT DESTINATION CSV FILE FOR SIMPLE REPORT HERE'

csv_writer_file2 = open(simple_report_destination, 'w', newline='')
csv_writer2 = csv.writer(csv_writer_file2, delimiter=';')
# write the column headers for the destination csv
csv_writer2.writerow(['URL', 'SELECTED KEYWORD', 'PAGE RANKING IN GOOGLE FOR SELECTED


# PREPARING OTHER SETTINGS

# define the URL of the website you want to search
site = 'INPUT WEBSITE NAME HERE'
```

8. **Run the code! Yay!**
9. **Results**

Simple report: the simple report will give you information about:

- If the URL ranks for the given keyword in Google (PAGE RANKING IN GOOGLE FOR SELECTED KEYWORD?) and if the page needs to be optimized for selected keyword to start ranking in Google (NEED TO OPTIMIZE FOR SELECTED KEYWORD?)
- Gives you a list of Google Search Results for the site search for a selected keyword (GOOGLE SEARCH RESULTS)
- If Google result pages are internally linked to the URL (INTERNAL LINKING?)
- This report allows us to verify if the URL the keyword was associated with is ranking for the given keyword in Google.
- If the URL is not among the results, then it should be optimized for the keyword it was associated with.
- The report also tells us if there is internal linking amongst the Google result pages and the URL we are analysing. If there is no internal linking, this should be amended, as google clearly thinks the pages should be connected - The pages ranking for identical keywords should be internally linked

Detailed report:

- The detailed report has all the data the simple report has, but in more detail as it also has all the links from every Google result page (LINKS IN THE GOOGLE SEARCH RESULTS PAGES)