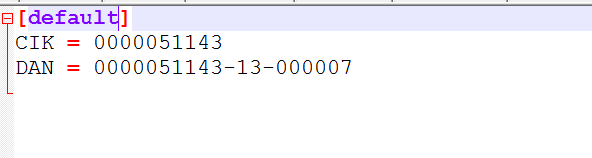
**Problem 1: Data wrangling Edgar data from text files (50 points)**

**Part 1: Parse files**

The script aims to extract data for all the individual tables available from the SEC website for a given CIK and Document Access Number for a company and redirect the output to the csv files on the system.

**Design:**

1. Take the input company name for which the script needs to execute. This is achieved by passing the company name as the argument (“default” in our case) for the script during execution.



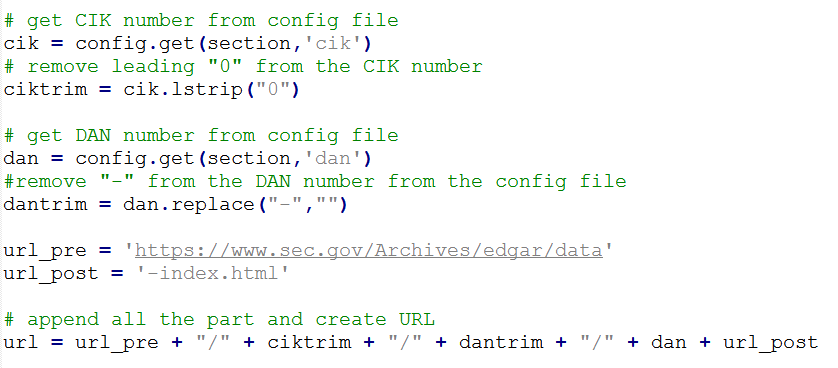
Unix prompt execution

1. This Company is configured in the .ini (“edgar.ini”) file in which all the CIK and DAN (Document Access Number) for every companies can be parameterized. This file exists under the working directory on the docker image (“home/python/edgar.ini”)
2. To accept the input from the user from the CLI and read this configuration file, module sys and module configparser are used within the script.

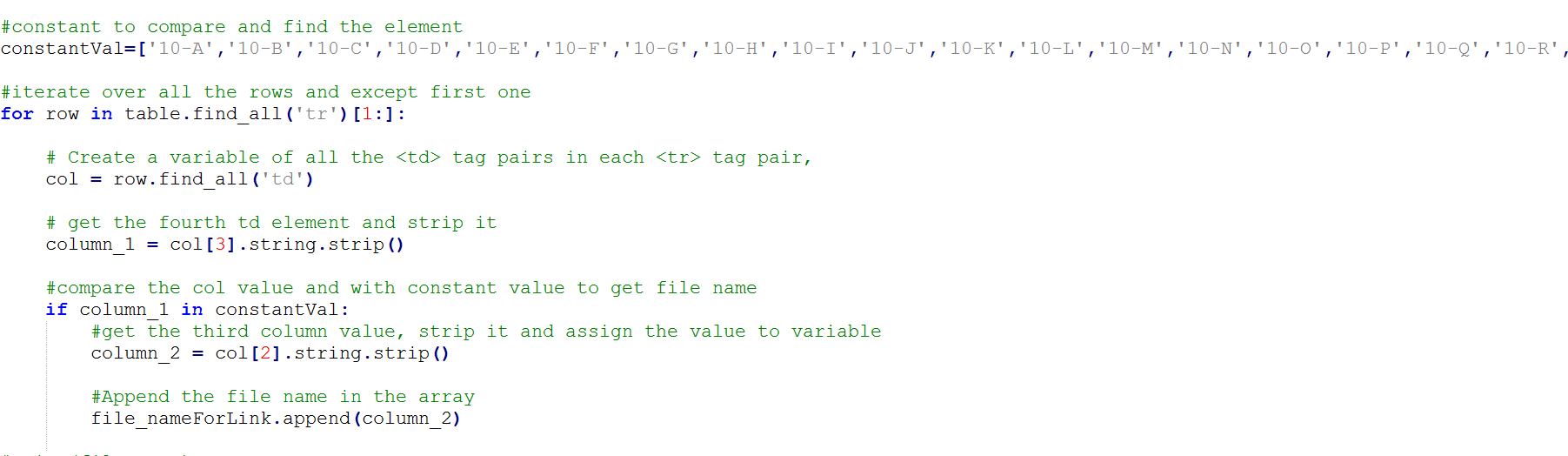
4. If the script finds the CIK and the DAN number for company entered, a URL is generated in the following format

http://www.sec.gov/Archives/edgar/data/51143/000005114313000007/0000051143-13-000007-index.html

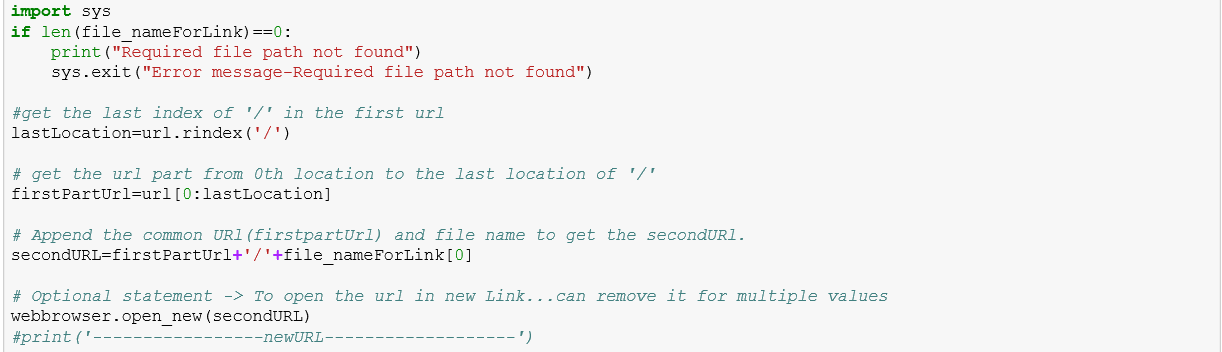
this will take you to the company page for which the data needs to be scraped/extracted.



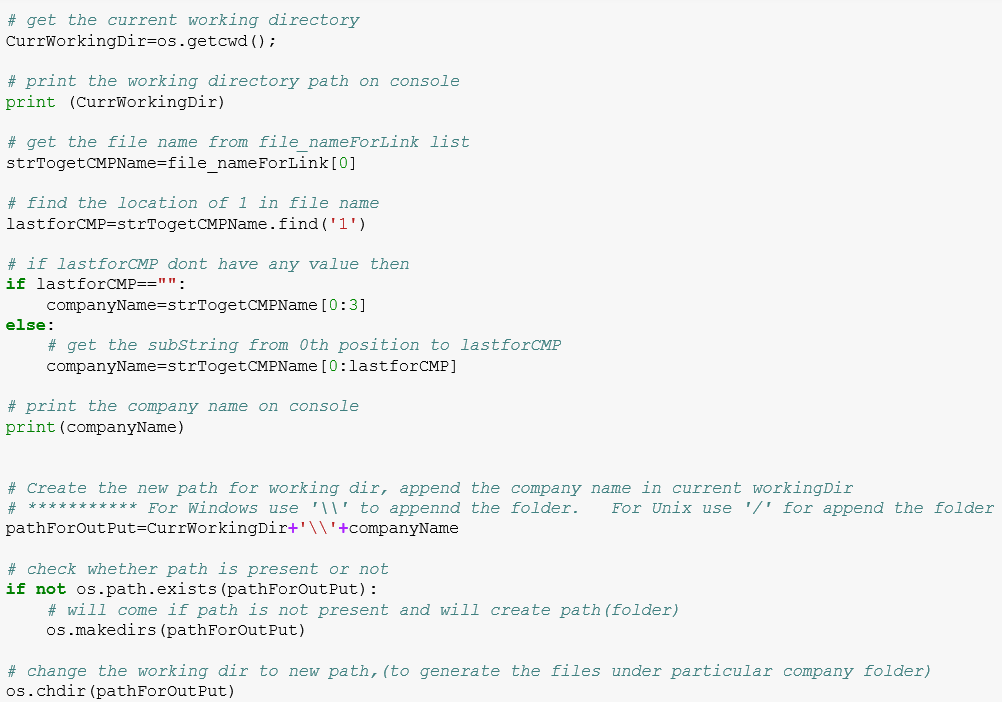
1. On this URL, it will look for table which has the class= “tableFilea” and look for type 10-Q,10-K,10-R etc in the row. If find the required record then will take document name and append to list.



6) If there is no document for criteria then terminate the program. If find the document then will create new URL for based on the captured links.



7) Take the first three letters of the link and treated as company name. Create the folder based on company name. For that check the current working directory name and check for the company name folder. If the folder is not present then will create folder. If the folder is present then do nothing. Change the working directory to the new path (till the company name).



8) Use the new URL that we have created in 6th step. Find the table based on the “Table” tag name. Loop through all the tables and check whether table contain the “$” in any column. If table contain “$” then will add the table to list and break the loop.



9) Find the table based on the “Table” tag name. Loop through all the tables and check whether table row style contain the “background” or “background-color”. If find then will add in list and break the loop. Same thing check for every table.



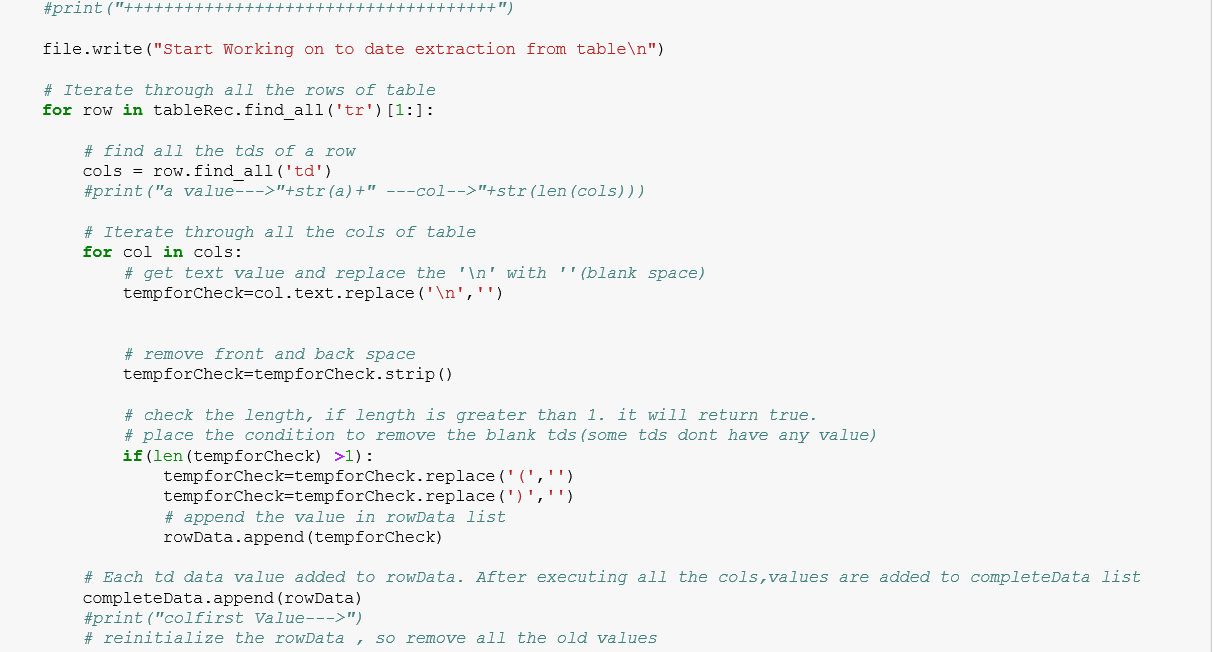
10) Finally compare the number of table found which have “$” sing and table whose row style has “background” or “background-color”. Compare both of them and will the take in maximum number of tables in new list and will work on that table list. Also, open the log file and starting writing on it.

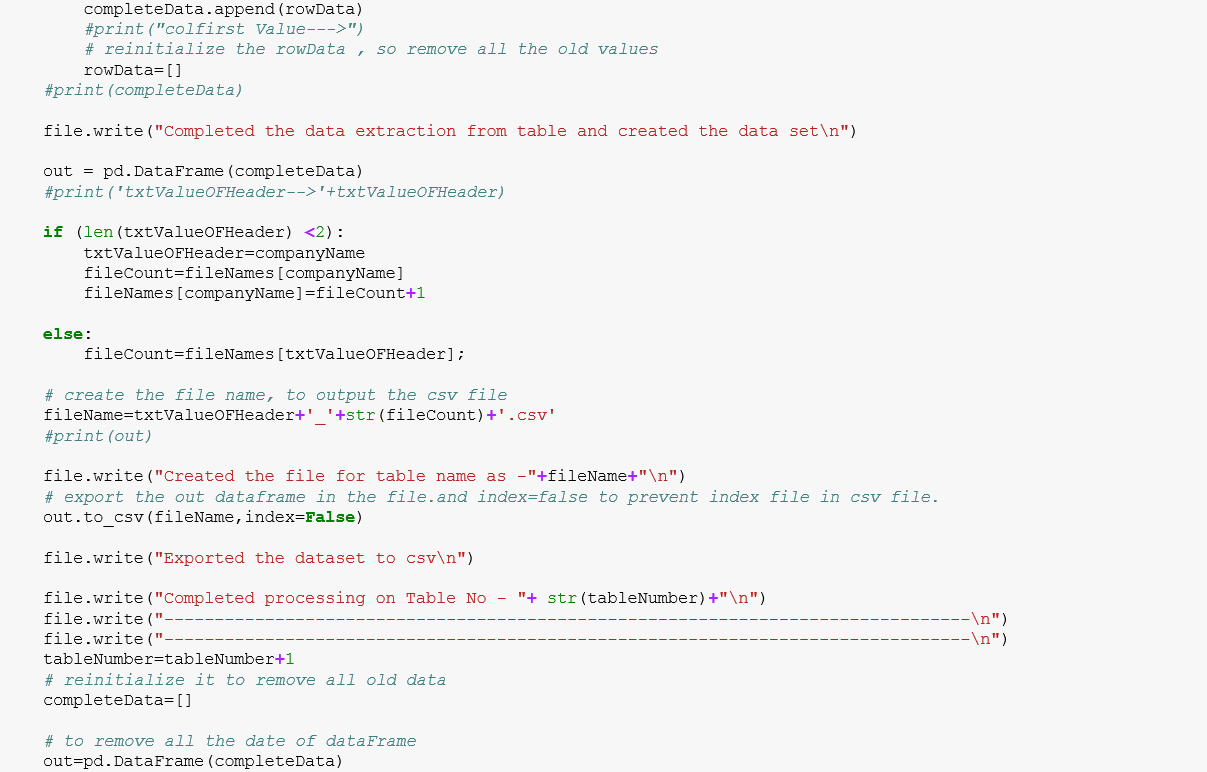


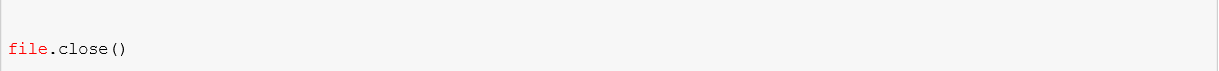
11) Work on to retrieve the name for file name. Take table header and iterate through the rows and take the row text.



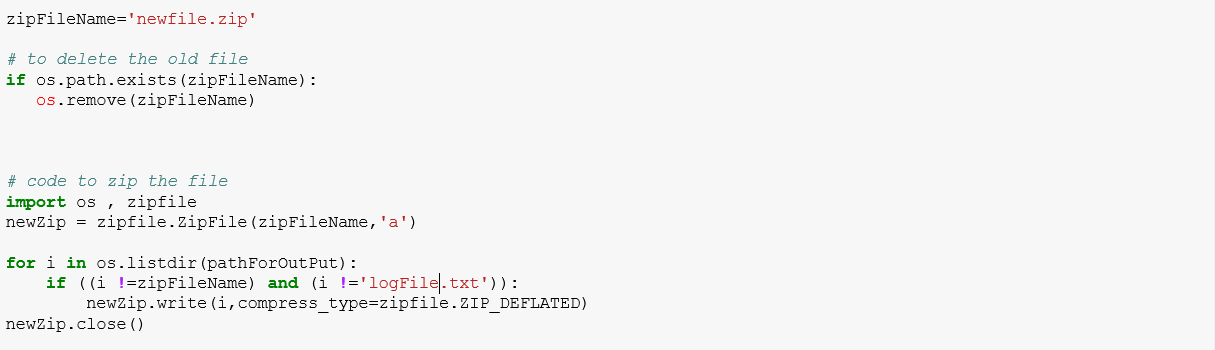
12) Loop through the tables and rows. Take the td values of each row. If the length of the text value of td is greater than 2 then will append in list to create the dataframe. At the end will export data frame to csv file.







13) Create the zip file for all the extracted file (exception logfile).



14) Used the config file to store the credential of amazon S3. Botos make the connection, create bucket name as “adsteam04\_”+ companyName. And upload the file in bucket. Used the exception handling If user enter the invalid credential for Amazon S3.

**Note:-** config file for amazon s3 credential is present ~/.aws/credentials.

