1. Date range is from 01-01-2007 to 21-08-2019. Words included in search was asylansøg, immigrant, flygtning, migrant.
2. So called webfeatures were not included as they did not follow the general structure of articles
3. Scraping took a total of 9 hours split on two computers. In total this adds up to x amount of calls.
   The files were concatenated using:

```python
def append_files(files):
    df = pd.concat([pd.read_csv(f, header = 0, sep = ';') for f in files], axis = 0)
    df.to_csv('./data/{}_concat.csv'.format(files[0].split('.')[0]), index = False) #Save to csv named as the first file with 'concat' appended

lst = ['dr_contents{}.csv', 'dr_links{}.csv', 'dr_log{}.csv']
for i in range(len(lst)):
    append_files([lst[i].format(''), lst[i].format('2')])
```

0.1 TODO

1. Get total article count.
2. Include in search 'invandrer'

0.2 Documentation

The process of scraping the DR homepage for articles are done within `class dr_scraper`. This is done to simplify storing names of different files that are created. The class uses a couple of methods that are described below:

0.2.1 `def get_dr_article_links(self, searchterms, start, end)`

This function uses the search engine on dr.dk to search for terms in `searchterms`. When searching for a term a list of articles, where the term is present, is returned. The search term, including parameters, are included in the URL and the function accept parameters that can filter results. The URL also includes a page parameter but there is no 'total results' measure available so the function uses an infinite while loop that breaks when the return is empty. BeautifulSoup is used to parse the data returned from the URL and gathers the URL pointing to each article. The resulting data from the searchterm is then stored in a csv file.

The function does this for every searchterm in the list `searchterms` and when done concatenates alle generated csv-files.

0.2.2 `def get_dr_article_contents(self, article_links)`

This function scrapes contents from the URLs provided in the list `article_links`. The data extracted for each article is title, publish date, and the text in the article. This data is coupled with the URLs for each article and returned as a pandas DataFrame.

0.2.3 `def batcher(self, article_links = None, batch_size = 100)`

This function splits the links into batches, that is scraped using `get_dr_article_contents` and saved into seperate csv-files. When done all batch-files are then merged to a single csv-file. This is done to ensure that the scraped data is not lost, should an error be raised during runtime.

If a csv-file with contents is already in the cwd the new content is extended to this file. Also URLs in passed `article_links` already present in the contents file is removed from the list to ensure a URL is not scraped twice.

0.2.4 `def get_dr_article_count(self, start, end, project_name = 'Get article counts')`

Fetches the total number of articles published for each day in the range defined by `start` and `end`. The function uses 'https://www.dr.dk/nyheder/allenyheder/' that lists all articles published on a particular date. When done it saves the data to a csv-file.