# HIBP_email_checker_V2

## Modules

pandas requests time

## Functions

**main**()

```
Specifies the content to be passed into iterate_over_csv, calls
Iterate_over_csv with file paths and start row, and prints out
the emails and row numbers of the rows in the CSV that made a
bad request to HIBP's API (Response status_code = 400 or 403).
```

**iterate_over_csv**(file_path_with_emails, content=None, append=False, start_row=0, stop_row=0, lookup_file_path=None)

```
Reads a CSV file_path and a start_row for which row to start
processing data on. The CSV file should be the All CAR Members
list from Pardot.

:param file_path_with_emails: The file path that contains the
    emails to be checked.
:param content: Either a list of headers if new CSV is
    wanted as a result, otherwise, provide a file path
    that will be used to append the results to. If a file path
    is provided, 'content' is then to be used in conjuction with
    'append'. If 'content' is a file path and 'append' is True,
    the results of the program will be appended to the file
    provided by the file path. If append is False, a new CSV
    will be created and all of that file's headers will be
    used as the fields to be found. If the latter is the
    case, be sure that the headers of the content file have the
    same exact headers as wherever the data associated with
    the email is coming from.
    Default is None, i.e. create a new CSV with all headers and
    data associated with the emails within
    'file_path_with_emails'.
:param append: If 'append' is True, the program will append the
    results to the file provided by content. If 'append' is
    False, the program will create a new CSV with the file
    name specified by the global variable'save_to_file_path',
    but with the same headers as the file specified by content.
:param start_row: The row to start processing data on in
    email_file_path.
:param stop_row: The row, inclusive, in the CSV file to stop
    processing data on.
:param lookup_file_path: If the data associated with the emails
    exist in another file, pass the file path here. Should be
    used if for example, 'file_path_with_emails' only has a
    specific list of emails to check, but the data associated
```

```
            with the emails exists within a different file.
```

**process_email**(index, email, email_df, lookup_df)

```
    Calls check_if_pwned. If True is returned, add email to
    pwned_emails and add all data specified in content to DataFrame.

    :param index: The current row we are at in 'email_df'.
    :param email: The email we are processing.
    :param email_df: The DataFrame that contains the emails to be
        processed.
    :param lookup_df: The DataFrame, if not None, that contains the
        data associated with the email if 'email_df' does not have
        the data.
```

**check_if_pwned**(email)

```
    Method to check if the email has been pwned
    (Response status_codde = 200), if email has not been pwned
    (Response status_code = 404), or something went
    wrong. Either an error (Response status_code = 400 or 403),
    or too many requests have been made
    (Response status_code = 429), and we must wait until we can
    make another request to HIBP's API at:
    https://haveibeenpwned.com/API/v3.

    :param email: The email to be checked if pwned.
```

**pwned_emails_to_csv**()

```
Converts pwned_emails_df DataFrame that contains the pwned emails that
Have been found to a CSV file, then prints the current row we are at
within the CSV file that contains the emails to be processed.
```

## Data

**API_KEY** = '48ec8f5a94384bc083178a91adf26cf3'
**HIBP_URI** = 'https://haveibeenpwned.com/api/v3/breachedaccount/'
**bad_excel_rows** = []
**headers** = {'hibp-api-key': '48ec8f5a94384bc083178a91adf26cf3', 'user-agent': 'CAR HIBP Email Checker'}
**pwned_emails** = []
**pwned_emails_df** = Empty DataFrame Columns: [] Index: []
**pwned_emails_df_row_log** = 0
**save_to_file_path** = '/Users/username/Desktop/File Name.csv'
**user_agent** = 'CAR HIBP Email Checker'