Davis_Labor_Day_Races

December 10, 2023

0.1 Data Scraping

Import libraries

[ ]:
```
import requests
import pandas as pd
from bs4 import BeautifulSoup
import numpy as np
import re
```

0.1.1 Getter (helper) Functions Implementations

This chunk of code consists of helper functions that return our desired value from the table in the website. For example, participant's name, age, and finish time, ect. There will be a construct function that generates dataframe by calling these helper functions.

By observed, all helper functions require `response` as argument, and some of them expect `year` #### Here is how each helper function work: 1. Convert the response (HTML file) into a Beautiful Soup object. 2. Find all the matching values by using `find_all()` function from the html file. Matched values are the texts in the html file that we are interested in - we identify the tag and class of the desired texts using Chrome's developer tools, and then specify them in the function to obtain our result. 3. For each value in the list of matched values - extract the content by using beautiful soup object's `get_text()` function - append the content to a list 4. After we have a list of content, convert the list to pandas series object 5. Filter out irrelevant content from the dataframe. Irrelevant information were captured because they share the same tag and class in the html file, and were captured in step 2 where we used `find_all()` function. - we observe the pattern of occurrence for the desired contents, and for each type of information (age, finish time, etc.), the patterns are different. More details about this are provided below. - The same information in different years might be rendered differently, so some helper functions take the `year` as a parameter to extract the desired content accordingly. 6. Reset the index 7. Return the series

[ ]:
```
def get_names(response):
    soup = BeautifulSoup(response.content, "html.parser")
    values = soup.find_all('a', class_ = 'ltw-name', target = '_blank')
    extracted_names = [name.get_text() for name in values]
    names_series = pd.Series(extracted_names)
    return names_series

def get_ages(response, year):
    soup = BeautifulSoup(response.content, "html.parser")
```

```
    values = soup.find_all('td' ,class_ = 'd-none d-sm-table-cell')
    extracted_values = [age.get_text() for age in values]
    value_series = pd.Series(extracted_values)
    numeric_age_data = pd.to_numeric(value_series, errors='coerce')
    if (year == 2023):
        correct_age_data = numeric_age_data.iloc[3::5]
    elif(year == 2022):
        correct_age_data = numeric_age_data.iloc[2::5]
    elif(year == 2019):
        correct_age_data = numeric_age_data.iloc[4::8]
    elif (year == 2018):
        correct_age_data = numeric_age_data.iloc[4::8]
    elif (year == 2017 or year == 2016):
        correct_age_data = numeric_age_data.iloc[2::5]
    elif (year == 2015 or year == 2014):
        correct_age_data = numeric_age_data.iloc[2::5]
    elif (year == 2013):
        correct_age_data = numeric_age_data.iloc[2::5]

    correct_age_data.reset_index(drop=True, inplace=True)
    correct_age_data = correct_age_data.astype(int)
    return correct_age_data

def get_times(response, year):
    soup = BeautifulSoup(response.content, "html.parser")
    values = soup.find_all('td' ,class_ = 'd-none d-sm-table-cell ltw-time')
    extracted_values = [time.get_text() for time in values]
    value_series = pd.Series(extracted_values)
    correct_time_data = value_series.iloc[1::2]
    if (year == 2015 or year == 2014):
        correct_time_data = value_series.iloc[0::2]
    elif (year == 2013):
        correct_time_data = value_series.iloc[0::2]

    correct_time_data.reset_index(drop=True, inplace=True)
    return correct_time_data

def get_gender(response):
    soup = BeautifulSoup(response.content, "html.parser")
    values = soup.find_all('a', class_ = 'ltw-name')
    extracted_values = [value.get_text() for value in values]
    value_series = pd.Series(extracted_values)
    correct_gender_data = value_series.iloc[2::3]
    correct_gender_data.reset_index(drop=True, inplace=True)
    return correct_gender_data
```

0.1.2 Decription of the Process of Writing a Helper Function

We are going to use `get_ages(response, year)` function as an example to illustrate how we build a getter function.

First make a request from url in order to obtain the html file.

[ ]:
```
URL_10k_p2_2023 = 'https://results.changeofpace.com/results.aspx?
↪CId=16356&RId=6121&EId=2&dt=0&PageNo=2'
response = requests.get(URL_10k_p2_2023)
response.raise_for_status()
```

Convert html file to beautiful soup object

[ ]:
```
soup = BeautifulSoup(response.content, "html.parser")
ages = soup.find_all('td' ,class_ = 'd-none d-sm-table-cell')
```

Extract contents from the html text

[ ]:
```
extracted_values = [age.get_text() for age in ages]
value_series = pd.Series(extracted_values)
```

Now we met a difficulty: there are irrelevant information in the series. We want to extract the information that we are interested in. In this case, `age` is what we want. Let first take a look at the original series.

[ ]:
```
value_series.head(15)
```

[ ]:
```
0     696
1       1
2
3      71
4      35
5     854
6       5
7
8      39
9      17
10    880
11      9
12
13     26
14     36
dtype: object
```

As we can see there are some missing value in the series (values in index 2, 7, ext.), so we initially `drop` the missing values by using`dropna()` function to the series. This is actually a very bad idea since it cause me lots of troubles later. More details about this will be provided.

```
[ ]: numeric_age_data = pd.to_numeric(value_series, errors='coerce').dropna()
     numeric_age_data = numeric_age_data.astype(int)
     numeric_age_data.reset_index(drop=True, inplace=True)
     numeric_age_data.head(15)
```

```
[ ]: 0     696
     1       1
     2      71
     3      35
     4     854
     5       5
     6      39
     7      17
     8     880
     9       9
     10     26
     11     36
     12    741
     13      6
     14     58
     dtype: int64
```

Now it looks "better" as there is no missing value. We then started obsering the pattern of occurence of `ages`.

The actual first 4 `ages` in the website are `71, 39, 26, and 58` - Click here to see the actual values

By observing the series, we find that the actual age start at index 2 and are separated by 3 irrevelant values to the next runner's actual age (which is 4 steps), so we can extract the desired values by using this particular pattern

```
[ ]: correct_age_data = numeric_age_data.iloc[2::4]
     correct_age_data.reset_index(drop=True, inplace=True)
     correct_age_data.head(4)
```

```
[ ]: 0    71
     1    39
     2    26
     3    58
     dtype: int64
```

This looks pretty good! As these numbers are in the exact same order as they are in the website. Let's proceed with combining this age series with participants' name to have a better view. Assume we already have the `get_names(response)` function so we can use it for illustration purpose.

```
[ ]: name_data = get_names(response)
     df_name_and_age = pd.DataFrame({'Name': name_data, 'Age': correct_age_data})
     df_name_and_age.head(4)
```

[ ]:
```
                          Name  Age
0                Mike Cordano   71
1  Hortensia Cisneros Benftez   39
2               Kevin Barrett   26
3                David Wilson   58
```

Name and age also match. Let's not only check the first several lines, but also inspect lines in the middle to ensure the correctness continues:

[ ]:
```
df_name_and_age.iloc[27:28]
```

[ ]:
```
      Name  Age
27  Bao Hu   20
```

Name and age also match, because this is me. I participated in this event in 2023, so my information is included

However, as we mentioned above, dropping missing value was actually a bad idea as we soon met a new problem. The link we used above are from page 2, 10K, but if we used the link of the third page in the same year and also in 10K catagory, we obtained wrong age values.

Below we use the same process as above, but change the link to the another page (p3), in the same year

[ ]:
```
URL_10k_p3_2023 = 'https://results.changeofpace.com/results.aspx?
 ↪CId=16356&RId=6121&EId=2&dt=0&PageNo=3'
response = requests.get(URL_10k_p3_2023)
response.raise_for_status()

soup = BeautifulSoup(response.content, "html.parser")
ages = soup.find_all('td' ,class_ = 'd-none d-sm-table-cell')

extracted_values = [age.get_text() for age in ages]
value_series = pd.Series(extracted_values)

numeric_age_data = pd.to_numeric(value_series, errors='coerce').dropna()
numeric_age_data = numeric_age_data.astype(int)
numeric_age_data.reset_index(drop=True, inplace=True)

correct_age_data = numeric_age_data.iloc[2::4]
correct_age_data.reset_index(drop=True, inplace=True)

name_data = get_names(response)
df_name_and_age = pd.DataFrame({'Name': name_data, 'Age': correct_age_data})
```

we first check the first 4 lines of the table, and it looks good

[ ]:
```
df_name_and_age.head(4)
```

[ ]:
```
                Name  Age
0   Andreanna Shuman   67
1         Soila Rios   67
2  Andrea Villanueva   60
3   Madison Buddingh   22
```

we then check 4 participants' ages in the middle of the table:

[ ]:
```
df_name_and_age.iloc[27:31]
```

[ ]:
```
                Name  Age
27  Lisa Geibel-Finn  744
28   Christopher Kim  799
29        Carey Seal  758
30       Lenna Ontai  655
```

Apparently, we didn't manage to capture the correct age values. There must be something wrong in the process of extracting desired age values. The thing that causes confusion is that the method we used in page 2 work with no trouble, so we then started observing the difference of the tables between 2 pages.

After observation, we found that there is a column named `Sp Div Pos`(stands for "Special Division Position") that share the same tag and class as the texts containing our desired age values. On the table from page 2, there is no value under this column, and in contrast, some of rows contain value under this column on the table from page 3. The occurence of the first value of `Sp Div Pos` ruins the pattern that "each `age value` is separated by `3` irrevelant values to the next `age value`", as the first `Sp Div Pos` value make the distance from 3 to 4. As a result, all the following data were captured incorrectly.

So we then considered to eliminate the html text with the occurence of `Sp Div Pos` value, but since they share the exact same tag and class as the `age value` as mentioned serveral times above, we were not able to find a way to do that.

[ ]:
```
ages[:5]
```

[ ]:
```
[<td class="d-none d-sm-table-cell">853</td>,
 <td class="d-none d-sm-table-cell">1</td>,
 <td class="d-none d-sm-table-cell"></td>,
 <td class="d-none d-sm-table-cell">67</td>,
 <td class="d-none d-sm-table-cell">38</td>]
```

We spent hours to find a way to eliminate the html text with the occurence of `Sp Div Pos` value, but none of the apporaches worked. So we then started over, and observed the captured html texts from `find_all()` function (the output from the line of code above is an example, where 67 is the first age value). We noticed that the third line contains no value, which is the `NA` that we dropped after converting the html texts to pandas series.

We then realized that dropping the `NA` value is a wrong decision since they are the values under the `Sp Div Pos` column. If the participant is not in any special division (Stroller Division, Dog Division, ect), the value in the participant's row under column `Sp Div Pos` will be a blank, which is the third line in the output above, and which is also the `NA` we dropped previously.

So once the first `Sp Div Pos` value occured, it means that one more irrevelant value appears between two age values since it's not a `NA`, and we didn't drop it.

The simplest way to fix it is that not to drop `NA`, instead, change the pattern that we extract the age values.

[ ]:
```
## extract age values from page 3
URL_10k_p3_2023 = 'https://results.changeofpace.com/results.aspx?
 ↪CId=16356&RId=6121&EId=2&dt=0&PageNo=3'
response = requests.get(URL_10k_p3_2023)
response.raise_for_status()

soup = BeautifulSoup(response.content, "html.parser")
ages = soup.find_all('td' ,class_ = 'd-none d-sm-table-cell')

extracted_values = [age.get_text() for age in ages]
value_series = pd.Series(extracted_values)

## simply comment out this line
# numeric_age_data = pd.to_numeric(value_series, errors='coerce').dropna()
# numeric_age_data = numeric_age_data.astype(int)
# numeric_age_data.reset_index(drop=True, inplace=True)

## and change the pattern we extract values
correct_age_data = value_series.iloc[3::5]
correct_age_data.reset_index(drop=True, inplace=True)

name_data = get_names(response)
df_name_and_age = pd.DataFrame({'Name': name_data, 'Age': correct_age_data})
df_name_and_age = pd.DataFrame({'Name': name_data, 'Age': correct_age_data})
df_name_and_age.iloc[27:31]
```

[ ]:
```
                Name Age
27  Lisa Geibel-Finn  58
28   Christopher Kim  53
29        Carey Seal  42
30       Lenna Ontai  50
```

Now we have sovled the problem! The names and ages match.

After we have all the necessary helper functions, we can start constructing the dataframe generating function

0.1.3 Dataframe Generating Function

Fairly straightfoward, create different series by using corresponding helper functions, and then combine to a dataframe

```
[ ]: def generate_dataframe(URL, distance, year):
        '''
        generates a dataframe containing participant's name, age, finish time, ect.
        @ URL: link to the website containing the table with information that we␣
     ↪want
        @ distance: distance category, 10K and 5K
        @ year: the year when the event was hold
        @ return: a dataframe that contains the information we want
        '''
        response = requests.get(URL)
        response.raise_for_status()
        names = get_names(response)
        ages = get_ages(response, year)
        times = get_times(response, year)
        genders = get_gender(response)
        df = pd.DataFrame({
                           'Name': names,
                           'Age': ages,
                           'Time': times,
                           'Gender': genders,
                           'Distance':distance,
                           'Year':year
                           })
        return df
```

Now we need to apply `generate_dataframe(URL, distance, year)` function to all the urls, and concatenate every dataframe from each url to a big dataframe.

The code and reports above about `Data Scraping` were written by Bao Hu

URLs

```
[ ]: url_5K_2023 = ['https://results.changeofpace.com/results.aspx?
     ↪CId=16356&RId=6121&EId=1',
                    'https://results.changeofpace.com/results.aspx?
     ↪CId=16356&RId=6121&EId=1&dt=0&PageNo=2',
                    'https://results.changeofpace.com/results.aspx?
     ↪CId=16356&RId=6121&EId=1&dt=0&PageNo=3',
                    'https://results.changeofpace.com/results.aspx?
     ↪CId=16356&RId=6121&EId=1&dt=0&PageNo=4',
                    'https://results.changeofpace.com/results.aspx?
     ↪CId=16356&RId=6121&EId=1&dt=0&PageNo=5',
                    'https://results.changeofpace.com/results.aspx?
     ↪CId=16356&RId=6121&EId=1&dt=0&PageNo=6',
                    'https://results.changeofpace.com/results.aspx?
     ↪CId=16356&RId=6121&EId=1&dt=0&PageNo=7',
```

```
                'https://results.changeofpace.com/results.aspx?
  ↪CId=16356&RId=6121&EId=1&dt=0&PageNo=8',
                'https://results.changeofpace.com/results.aspx?
  ↪CId=16356&RId=6121&EId=1&dt=0&PageNo=9',]
url_10K_2023 = ['https://results.changeofpace.com/results.aspx?
  ↪CId=16356&RId=6121&EId=2',
                 'https://results.changeofpace.com/results.aspx?
  ↪CId=16356&RId=6121&EId=2&dt=0&PageNo=2',
                 'https://results.changeofpace.com/results.aspx?
  ↪CId=16356&RId=6121&EId=2&dt=0&PageNo=3',
                 'https://results.changeofpace.com/results.aspx?
  ↪CId=16356&RId=6121&EId=2&dt=0&PageNo=4',
                 'https://results.changeofpace.com/results.aspx?
  ↪CId=16356&RId=6121&EId=2&dt=0&PageNo=5',]
url_5K_2022 = ['https://results.changeofpace.com/results.aspx?
  ↪CId=16356&RId=6103',
                'https://results.changeofpace.com/results.aspx?
  ↪CId=16356&RId=6103&EId=1&dt=0&PageNo=2',
                'https://results.changeofpace.com/results.aspx?
  ↪CId=16356&RId=6103&EId=1&dt=0&PageNo=3',
                'https://results.changeofpace.com/results.aspx?
  ↪CId=16356&RId=6103&EId=1&dt=0&PageNo=4',
                'https://results.changeofpace.com/results.aspx?
  ↪CId=16356&RId=6103&EId=1&dt=0&PageNo=5',
                'https://results.changeofpace.com/results.aspx?
  ↪CId=16356&RId=6103&EId=1&dt=0&PageNo=6',
                'https://results.changeofpace.com/results.aspx?
  ↪CId=16356&RId=6103&EId=1&dt=0&PageNo=7',]
url_10K_2022 = ['https://results.changeofpace.com/results.aspx?
  ↪CId=16356&RId=6103&EId=2',
                 'https://results.changeofpace.com/results.aspx?
  ↪CId=16356&RId=6103&EId=2&dt=0&PageNo=2',
                 'https://results.changeofpace.com/results.aspx?
  ↪CId=16356&RId=6103&EId=2&dt=0&PageNo=3',
                 'https://results.changeofpace.com/results.aspx?
  ↪CId=16356&RId=6103&EId=2&dt=0&PageNo=4',]
url_5K_2019 = ['https://results.changeofpace.com/results.aspx?
  ↪CId=16356&RId=6072',
                'https://results.changeofpace.com/results.aspx?
  ↪CId=16356&RId=6072&EId=1&dt=0&PageNo=2',
                'https://results.changeofpace.com/results.aspx?
  ↪CId=16356&RId=6072&EId=1&dt=0&PageNo=3',
                'https://results.changeofpace.com/results.aspx?
  ↪CId=16356&RId=6072&EId=1&dt=0&PageNo=4',
```

```
                'https://results.changeofpace.com/results.aspx?
  ↪CId=16356&RId=6072&EId=1&dt=0&PageNo=5',
                'https://results.changeofpace.com/results.aspx?
  ↪CId=16356&RId=6072&EId=1&dt=0&PageNo=6',]
url_10K_2019 = ['https://results.changeofpace.com/results.aspx?
  ↪CId=16356&RId=6072&EId=2',
                 'https://results.changeofpace.com/results.aspx?
  ↪CId=16356&RId=6072&EId=2&dt=0&PageNo=2',
                 'https://results.changeofpace.com/results.aspx?
  ↪CId=16356&RId=6072&EId=2&dt=0&PageNo=3',
                 'https://results.changeofpace.com/results.aspx?
  ↪CId=16356&RId=6072&EId=2&dt=0&PageNo=4',]
url_5K_2018 = ['https://results.changeofpace.com/results.aspx?
  ↪CId=16356&RId=6047',
                'https://results.changeofpace.com/results.aspx?
  ↪CId=16356&RId=6047&EId=1&dt=0&PageNo=2',
                'https://results.changeofpace.com/results.aspx?
  ↪CId=16356&RId=6047&EId=1&dt=0&PageNo=3',
                'https://results.changeofpace.com/results.aspx?
  ↪CId=16356&RId=6047&EId=1&dt=0&PageNo=4',
                'https://results.changeofpace.com/results.aspx?
  ↪CId=16356&RId=6047&EId=1&dt=0&PageNo=5',
                'https://results.changeofpace.com/results.aspx?
  ↪CId=16356&RId=6047&EId=1&dt=0&PageNo=6',
                'https://results.changeofpace.com/results.aspx?
  ↪CId=16356&RId=6047&EId=1&dt=0&PageNo=7',]
url_10K_2018 = ['https://results.changeofpace.com/results.aspx?
  ↪CId=16356&RId=6047&EId=2',
                 'https://results.changeofpace.com/results.aspx?
  ↪CId=16356&RId=6047&EId=2&dt=0&PageNo=2',
                 'https://results.changeofpace.com/results.aspx?
  ↪CId=16356&RId=6047&EId=2&dt=0&PageNo=3',
                 'https://results.changeofpace.com/results.aspx?
  ↪CId=16356&RId=6047&EId=2&dt=0&PageNo=4',
                 'https://results.changeofpace.com/results.aspx?
  ↪CId=16356&RId=6047&EId=2&dt=0&PageNo=5',]
url_5K_2017 = ['https://results.changeofpace.com/results.aspx?
  ↪CId=16356&RId=6024',
                'https://results.changeofpace.com/results.aspx?
  ↪CId=16356&RId=6024&EId=1&dt=0&PageNo=2',
                'https://results.changeofpace.com/results.aspx?
  ↪CId=16356&RId=6024&EId=1&dt=0&PageNo=3',
                'https://results.changeofpace.com/results.aspx?
  ↪CId=16356&RId=6024&EId=1&dt=0&PageNo=4',]
```

```
url_10K_2017 = ['https://results.changeofpace.com/results.aspx?
 ↪CId=16356&RId=6024&EId=2',
                 'https://results.changeofpace.com/results.aspx?
 ↪CId=16356&RId=6024&EId=2&dt=0&PageNo=2',
                 'https://results.changeofpace.com/results.aspx?
 ↪CId=16356&RId=6024&EId=2&dt=0&PageNo=3',]
url_5K_2016 = ['https://results.changeofpace.com/results.aspx?
 ↪CId=16356&RId=188',
                'https://results.changeofpace.com/results.aspx?
 ↪CId=16356&RId=188&EId=1&dt=0&PageNo=2',
                'https://results.changeofpace.com/results.aspx?
 ↪CId=16356&RId=188&EId=1&dt=0&PageNo=3',
                'https://results.changeofpace.com/results.aspx?
 ↪CId=16356&RId=188&EId=1&dt=0&PageNo=4',
                'https://results.changeofpace.com/results.aspx?
 ↪CId=16356&RId=188&EId=1&dt=0&PageNo=5',
                'https://results.changeofpace.com/results.aspx?
 ↪CId=16356&RId=188&EId=1&dt=0&PageNo=6',]
url_10K_2016 = ['https://results.changeofpace.com/results.aspx?
 ↪CId=16356&RId=188&EId=2',
                 'https://results.changeofpace.com/results.aspx?
 ↪CId=16356&RId=188&EId=2&dt=0&PageNo=2',
                 'https://results.changeofpace.com/results.aspx?
 ↪CId=16356&RId=188&EId=2&dt=0&PageNo=3',
                 'https://results.changeofpace.com/results.aspx?
 ↪CId=16356&RId=188&EId=2&dt=0&PageNo=4',]
url_5K_2015 = ['https://results.changeofpace.com/results.aspx?
 ↪CId=16356&RId=148',
                'https://results.changeofpace.com/results.aspx?
 ↪CId=16356&RId=148&EId=1&dt=0&PageNo=2',
                'https://results.changeofpace.com/results.aspx?
 ↪CId=16356&RId=148&EId=1&dt=0&PageNo=3',
                'https://results.changeofpace.com/results.aspx?
 ↪CId=16356&RId=148&EId=1&dt=0&PageNo=4',
                'https://results.changeofpace.com/results.aspx?
 ↪CId=16356&RId=148&EId=1&dt=0&PageNo=5',
                'https://results.changeofpace.com/results.aspx?
 ↪CId=16356&RId=148&EId=1&dt=0&PageNo=6',
                'https://results.changeofpace.com/results.aspx?
 ↪CId=16356&RId=148&EId=1&dt=0&PageNo=7',]
url_10K_2015 = ['https://results.changeofpace.com/results.aspx?
 ↪CId=16356&RId=148&EId=2',
                 'https://results.changeofpace.com/results.aspx?
 ↪CId=16356&RId=148&EId=2&dt=0&PageNo=2',
```

```
                'https://results.changeofpace.com/results.aspx?
 ↪CId=16356&RId=148&EId=2&dt=0&PageNo=3',
                'https://results.changeofpace.com/results.aspx?
 ↪CId=16356&RId=148&EId=2&dt=0&PageNo=4',
                'https://results.changeofpace.com/results.aspx?
 ↪CId=16356&RId=148&EId=2&dt=0&PageNo=5',
                'https://results.changeofpace.com/results.aspx?
 ↪CId=16356&RId=148&EId=2&dt=0&PageNo=6',
                'https://results.changeofpace.com/results.aspx?
 ↪CId=16356&RId=148&EId=2&dt=0&PageNo=7',]
url_5K_2014 = ['https://results.changeofpace.com/results.aspx?CId=16356&RId=92',
               'https://results.changeofpace.com/results.aspx?
 ↪CId=16356&RId=92&EId=1&dt=0&PageNo=2',
               'https://results.changeofpace.com/results.aspx?
 ↪CId=16356&RId=92&EId=1&dt=0&PageNo=3',
               'https://results.changeofpace.com/results.aspx?
 ↪CId=16356&RId=92&EId=1&dt=0&PageNo=4',
               'https://results.changeofpace.com/results.aspx?
 ↪CId=16356&RId=92&EId=1&dt=0&PageNo=5',
               'https://results.changeofpace.com/results.aspx?
 ↪CId=16356&RId=92&EId=1&dt=0&PageNo=6',
               'https://results.changeofpace.com/results.aspx?
 ↪CId=16356&RId=92&EId=1&dt=0&PageNo=7',
               'https://results.changeofpace.com/results.aspx?
 ↪CId=16356&RId=92&EId=1&dt=0&PageNo=8',]
url_10K_2014 = ['https://results.changeofpace.com/results.aspx?
 ↪CId=16356&RId=92&EId=2',
                'https://results.changeofpace.com/results.aspx?
 ↪CId=16356&RId=92&EId=2&dt=0&PageNo=2',
                'https://results.changeofpace.com/results.aspx?
 ↪CId=16356&RId=92&EId=2&dt=0&PageNo=3',
                'https://results.changeofpace.com/results.aspx?
 ↪CId=16356&RId=92&EId=2&dt=0&PageNo=4',
                'https://results.changeofpace.com/results.aspx?
 ↪CId=16356&RId=92&EId=2&dt=0&PageNo=5',
                'https://results.changeofpace.com/results.aspx?
 ↪CId=16356&RId=92&EId=2&dt=0&PageNo=6',
                'https://results.changeofpace.com/results.aspx?
 ↪CId=16356&RId=92&EId=2&dt=0&PageNo=7',]
url_5K_2013 = ['https://results.changeofpace.com/results.aspx?
 ↪CId=16356&RId=30',]
url_10K_2013 = ['https://results.changeofpace.com/results.aspx?
 ↪CId=16356&RId=30&EId=2',]
```

Create Dataframe from all the URLs

[ ]:
```
years = [2023, 2022, 2019, 2018, 2017, 2016, 2015, 2014, 2013]
doc_types = ['5K', '10K']
dfs = {f"df_{year}_{doc_type}": pd.DataFrame() for year in years for doc_type␣
 ↪in doc_types}
for year in years:
    for doc_type in doc_types:
        for url in globals()[f"url_{doc_type}_{year}"]:
            df = generate_dataframe(url, doc_type, year)
            dfs[f"df_{year}_{doc_type}"] = pd.
 ↪concat([dfs[f"df_{year}_{doc_type}"], df], ignore_index=True)
all_dfs = pd.concat(dfs.values(),axis=0)
```

Store as a csv File

[ ]:
```
all_dfs.to_csv('data/all_data.csv', index=False)
```

0.2 Data Processing

Load dataframe

[ ]:
```
df = pd.read_csv('data/all_data.csv')
```

Reset index

[ ]:
```
df.index = np.arange(1, len(df) + 1)
df.head(1)
```

[ ]:

	Name	Age	Time	Gender	Distance	Year
1	William Liu	19	16:15.33	Male	5K	2023

Add Columns to Dataframe add `Age_group` column

[ ]:
```
age_groups = ['< 10', '10 - 19', '20 - 29', '30 - 39',
              '40 - 49', '50 - 59', '60 - 69', '70 - 79',
               '80 - 89', '> 90']
df['Age_group'] = pd.cut(df['Age'], bins=[0, 10, 20, 30, 40, 50, 60, 70, 80,␣
 ↪90, float('inf')],
                         labels=age_groups, right=False)
# change column order
df.insert(2, 'Age_group', df.pop('Age_group'))
df.head(1)
```

[ ]:

	Name	Age	Age_group	Time	Gender	Distance	Year
1	William Liu	19	10 - 19	16:15.33	Male	5K	2023

Convert `Time` from string to `minute` in float formatting

[ ]:
```
from datetime import datetime
df = df[df['Time'] != "Not started"]
```

```
df = df[df['Time'] != "Started"]
df = df[df['Time'] != "0.00"]
df = df[df['Time'] != "DQ"]
def time_to_minutes(time_str):
    if len(time_str) < 10:
        time_obj = datetime.strptime(time_str, '%M:%S.%f')
    else:
        time_obj = datetime.strptime(time_str, '%H:%M:%S.%f')

    total_minutes = time_obj.hour * 60 + time_obj.minute + time_obj.second / 60
    return total_minutes

df['Time'] = df['Time'].apply(time_to_minutes)
df['Time'] = df['Time'].apply(lambda x: round(float(x), 3))
df.rename(columns={'Time': 'Time (min)'}, inplace=True)

df.head(1)
```

[ ]:

	Name	Age	Age_group	Time (min)	Gender	Distance	Year
1	William Liu	19	10 - 19	16.25	Male	5K	2023

add `Speed (Km/h)` column

[ ]:
```
def time_to_speed(row):
    distance = float(row['Distance'][:-1])
    time_minutes = row['Time (min)']
    speed_kph = distance / (time_minutes / 60)
    return speed_kph

df['Speed (Km/h)'] = df.apply(time_to_speed, axis=1)
df['Speed (Km/h)'] = df['Speed (Km/h)'].apply(lambda x: round(float(x), 3))

# change column order
df.insert(5, 'Speed (Km/h)', df.pop('Speed (Km/h)'))
df.head(1)
```

[ ]:

	Name	Age	Age_group	Time (min)	Gender	Speed (Km/h)	Distance	Year
1	William Liu	19	10 - 19	16.25	Male	18.462	5K	2023

add `Pace (min/Km)` column

[ ]:
```
def time_to_pace(row):
    distance = float(row['Distance'][:-1])
    time_minutes = row['Time (min)']
    pace_min_Km = time_minutes / distance
    return pace_min_Km

df['Pace (min/Km)'] = df.apply(time_to_pace, axis=1)
```

```
df['Pace (min/Km)'] = df['Pace (min/Km)'].apply(lambda x: round(float(x), 3))

# change column order
df.insert(5, 'Pace (min/Km)', df.pop('Pace (min/Km)'))
df.head(1)
```

[ ]:
```
           Name  Age Age_group  Time (min) Gender  Pace (min/Km)  Speed (Km/h)  \
1  William Liu   19   10 - 19       16.25   Male           3.25        18.462

  Distance  Year
1       5K  2023
```

[ ]:
```
df.to_csv('data/processed.csv', index=False)
```

The code and reports about `Data Processing` were written by Bao Hu

1 Data Analysis

load dataframe

[ ]:
```
df = pd.read_csv("data/processed.csv")
```

Inspect Data

[ ]:
```
grouped_df = df.groupby('Age_group').size().reset_index(name='count')
grouped_df
```

[ ]:
```
  Age_group  count
0   10 - 19    376
1   20 - 29    528
2   30 - 39    833
3   40 - 49    882
4   50 - 59    701
5   60 - 69    428
6   70 - 79    165
7   80 - 89     27
8      < 10    107
9      > 90      3
```