**Task 1 - Crawling Method**

The crawling of the provided website, and process the results into a *.csv* file containing the url and header of each page, is divided into the following stages. The functions involved in performing Task 1 are *task\_1()* and *add\_header().*

**Stage 1:** Create a dictionary variable called *row\_list* for each addition to this dictionary will represent a new entry row in the eventual *.csv* file. This dictionary is created outside of the aforementioned Task 1 functions to act as a global variable that can be updated from functions of subsequent tasks to add further information, such as *player* in Task 2.

**Stage 2:** Create a helper function *add\_header()* that takes in a soup object and a dictionary, which corresponds to the url after parsing with BeautifulSoup and *row\_list*, to simply add the header found in the soup object to the *row\_list* under the *header* column.

**Stage 3:** Create a main driver function *task\_1()* that performs crawling, parsing, adding urls and calling *add\_header()* to complete Task 1.

* Crawling Algorithm:
  + Create a list *V* of urls that have been **visited.**
  + Create a priority list *L* of urls **to-visit**.
  + Find all outbound links to successor pages of current page.
  + Mark current page as visited by adding its url to *V*.
  + Add all aforementioned outbound links that have yet to be visited to *L* if they are not in *L* already. (That is, add the set of *{u’} – V* to list *L* with *{u’}* representing set of outbound links)
* Parsing:
  + Request access to website via. *requests.get()* from requests library.
  + Parse *html*  in *response.text* with BeautifulSoup’s *html.parser*.
* Adding urls:
  + Add current url to *row\_list* under [‘*urls*’].
* Calling *add\_header()*:
  + Locate the text content in *‘h1’* tag in the soup object from BeautifulSoup’s *html.parser*. This is current header.
  + Add current header to *row\_list* under [‘*header’*].

**Stage 4:** Output of *task\_1()* is a .csv file containing a column of valid urls crawled from the base url- the ‘Welcome’ page of tennis articles in this example- and a column of headers corresponding to the urls column. This is achieved by:

1. Converting the dictionary *row\_list* into a Data Frame via. the *pd.DataFrame* function in Pandas library.
2. Setting index to *urls* column.
3. Converting Data Frame structure to .csv file via. *to\_csv* function in Pandas library.

**Task 2 - Scraping Method**

* Player Name
  + Iterating through the list of urls from Task 1 and add each (and their header) to the newly initialised dictionary type *row\_list\_2a*.
    - Newly initialised for the addition of *player* column.
    - Simply adding *player* column to the *row\_list* of Task 1 will raise an error due to deletion of articles without player names, and hence different lengths in key-value pairs that cannot be converted into .csv format.
  + Parse and store its header and body text content via. BeautifulSoup.
  + Pass header text content to helper function *find\_names()*, if valid name found then add name to *row\_list\_2a* under ‘*player’* key.
  + If no valid name in header, pass body text content to *find\_names().* If valid name found then add name to *row\_list\_2a* under *‘player’* key.
  + If no valid name in header nor body text, delete this entry in *row\_list\_2a* under both [‘*url’*] and [‘*header’*].

* + Helper function *find\_name()*:
    - Takes a string and extract all of its words that start with a capital letter.
    - Each of these words are converted to uppercase and compared with the list of names (all uppercase) from *tennis.json* file parsed at outset of program.
    - If a match is found, return this match.
    - Otherwise, return a string that indicate to the main function that there is no match.
* Scores
  + Regex
  + Validation
* Output

**Task 4 – Plot Analysis**

* Data Used (Input)
  + *row\_list* from Task 3 (containing data for urls with valid player names and scores)
  + Converted *row\_list* from dictionary type to Data Frame via. *pd.DataFrame* function.
  + Extract list of player names via. *row\_list[‘player’]*.
  + Split Data Frame into groups based on the number of articles corresponding to each player name via. *df.groupby()* function.
  + Extract top 5 players with highest number of articles via. *.count().sort\_values()[0:5]* functions.
  + Input 1: Top 5 player names.
  + Input 2: Number of articles corresponding to each of the top 5 player names.
  + Input 1 and 2 are passed to *plt.xticks()* and *plt.bar()* function as *x-axis* and *y-axis* data respectively.
* Plot (Output)
  + INSERT IMAGE HERE
  + *x-axis:* Top 5 player names each labelled under a bar with 30 degree tilt for visual aesthetic.
  + *y-axis:* A numerical scale that displays number of article corresponding to each of the players’ name.
    - Increment:
    - Range:
* Analysis (Evaluation)
  + This plot is heavily reliant on the assumption that each valid article is mostly written about the first player name found in its header/body content.
  + Although potentially statistically favourable, this is not always the case and hence the plot and Task 2-4 fails to take this into account.
  + Articles without a valid score are filtered out in *part b* of Task 2. Therefore, this plot fails to account for articles written about a player but does not contain a valid score.
  + For example, an article that describes a player’s style of playing would not be represented in input data of this plot.

**Task 5 – Plot Analysis**

* Data Used (Input)
* Plot (Output)
* Analysis (Evaluation)

**Appropriateness of the Method for Deriving First Named Player**

* Assumption (of Input and Output)
* Logic
* Goal
* Analysis (Evaluation of Possible Challenges)

**Suggestion of the Method for Determining Win/Loss of the First Named Player**

* Assumption (of Input and Output)
* Logic
* Goal
* Analysis (Evaluation of Possible Challenges)

**Suggestion of Information Extraction and Processing Method for understanding Player Performance**

* Assumption (of Input and Output)
* Logic
* Goal
* Analysis (Evaluation of Possible Challenges)

A screenshot of a cell phone

Description automatically generated

A picture containing bird

Description automatically generated