Food Scraping Documentation

1. **PIP Installation**

   Run pip installation on OS terminal.
   Pip 3 for Python 3

   pip3 install selenium

2. **Installing Drivers**

   Selenium requires a driver to interface with the browser. Make sure it's in your PATH, eg.place it in /usr/bin or /usr/local/desktop.

   Make sure to download the same version as your browser.

   For example, to check your Chrome version, click on the 3 dots on the top right of the screen. Navigate 'Help' and then 'About Google Chrome'. There should be visible the Chrome version.

   a. Chrome: https://sites.google.com/chromium.org/driver/
   b. Edge: https://developer.microsoft.com/en-us/microsoft-edge/tools/webdriver/
   c. Firefox: https://github.com/mozilla/geckodriver/releases
   d. Safari: https://webkit.org/blog/6900/webdriver-support-in-safari-10/

3. **Importing Selenium**

   The user must import Webdriver from Selenium to allow interaction with the browser itself.

   a.
```
from selenium import webdriver
```

   The user must define where the webdriver is. Here it is placed on the desktop. The driver function then allows for selenium to interact with the browser. In this case, the driver used is **Google Chrome.**

   b.
```
PATH = "/Users/USER/Desktop/chromedriver"
```
   c.
```
driver = webdriver.Chrome(PATH)
```

Username and Password

1. **Environmental Variable Setting for Username and Password**

   To not expose the user's credentials, the username and password will be set in an environmental variable

```
user = os.environ.get('USER')
password = os.environ.get('PASSWORD')
```

2. **Open Terminal**

3. **Type within the terminal**

   $ echo 'export user= NUS_USERNAME' >> ~/.zshenv
   $ echo 'export 'password = NUS_PASSWORD' >> ~/.zshenv

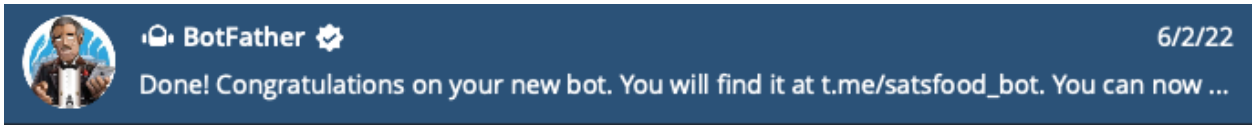
4. **Check Variables**

   $ echo $user
   $ echo $password

   This should output the username and password

Adding a Bot to a Telegram Group

1. **Search up @BotFather on Telegram**



2. **Use /token within the chat to get the token of bot**

```
bot_token = bot5190481628:AAHvjriOdIz_XXXXXXXXXXXXXXX
```

3. **Add bot to group**

4. **Go to https://api.telegram.org/<BOT ID>/getUpdates to find the chat ID of said bot**

   "chat":{"id":-794849324

5. **Create a variable to send the message**

```
send_text = 'https://api.telegram.org/bot' + bot_token + '/sendMessage?chat_id='
+ bot_chatID + '&parse_mode=Markdown&text=' + bot_message
```

6. **Use requests to access the link, sending the message**

   a. `requests.get(message_url)`

Running the Script

1. **Running Each Script**
   On Weekdays run "Weekday.py"
   On Weekends un "Weekend.py"

Unit testing

1. Functions that are tested are the ones listed below, run "test_functions.py" and type pytest within the terminal to test functions

Functions Documented:

1) **Function 'getting_stats':**
   -This is the 1st function part of our task automation:
   -In summary, the function clicks each link, retrieving the calories and macros for each meal.
   -The function can be documented in 7 parts.

1.1) The function accesses the 'link' provided within the parameter, with a pause, allowing the browser to load to access information.

```
def getting_stats(link):
  driver.get(link)
  time.sleep(1)
```

1.2) Using .text, the name of the food within the element is retrieved.

```
 name =
driver.find_element_by_xpath('//*[@id="signup-modal-slide-description"]/div/div/div[2]
/h3').text
  print(name)
```

1.3) Next, it runs a loop 4 times, to abstract macros content. Then, through a condition, it appends 'stats' with data that is abstracted through a space condition. The space condition allows data after the space to be retrieved, that is the values of the nutrients and macros with their units is abstracted.

```
stats = []
  for i in range(1, 5):
     stat_full =
driver.find_element_by_xpath("/html/body/div[2]/div[3]/div/div/div/div/div/div[2]/di
v[1]/div/div[2]/div/div/div/div/p["+str(i)+"]")
     stat = stat_full.text.split(" ")[-1]
     stats.append(stat)
  print(stats)
```

1.4) The function then considers the position of the 'calories' and retains everything except its unit 'kcal' from it. It then appends the value back to 'stats'

```
 print(stats[0])
  stats[0] = float(stats[0][:-4])
  print(stats[0])
```

1.5) Similarly, the function does this for the macros (protein, fat, and carbohydrates), retains the values and removes the unit 'g'. It then appends it back to 'stats'. This is done within a loop since 3 different macros need to be considered.

```
for i in range(1, 4):
     stats[i] = float(stats[i][:-1])
```

1.6) 'Stats' is appended by another value, that is the protein percentage per calorie.

```
stats.append(stats[2] / stats[0] * 100)
```

1.7) Finally, a new list 'food' is created which will bring out the food 'name' and its 'stats'. The list is then printed.

```
food = [name] + stats
```

```
print(food)

return food
```

2) Function "finding_the_best_meal"

-The function traverses the list of foods for each meal and find the highest protein percentage

2.1) The function first creates a list 'foods[]'
first to store the foods and its corresponding stats.

```
foods = []
```

2.2) It then creates a list 'protein_pcts[]', appending the highest protein percentage.

```
protein_pcts = []
```

2.3) A loop is created which traverses according to the length of the link provided. This loop calls the getting_stats function to retrieve the stats of each food. Then, it Appends the protein percentage of that food into the list "protein_pcts[]'. Then the Function appends the food itself into the list 'foods[]'.

```
    for i in range(0, len(links)):
        food = getting_stats(links[i])
        protein_pcts.append(food[-1])
        foods.append(food)
```

2.4) The highest protein percentage is found and the index is taken. The index is then Used to retrieve the statistics of the food with the highest protein percentage. It finally returns the food with the highest protein percentage in the certain dining session.

```
    for i in range(0, len(links)):
        food = getting_stats(links[i])
        protein_pcts.append(food[-1])
        foods.append(food)


    best_index = protein_pcts.index(max(protein_pcts))

    best_food = foods[best_index]

    print(best_food)
    return best_food
```