Dev Club Assignments

Assignment : Web scraping in Python

18th February 2021

P.S If stuck, join this discord channel and ask us questions directly here

Introduction

Web Scraping can serve an unlimited number of purposes. Whether you are performing some Automation task or fetching data for training your ML model, web scraping is the way to go and hence, we are giving you an opportunity to experience it yourself.

In this assignment, you are given two tasks. Firstly, you will create a python script for auto-login to Moodle. In the next part of the assignment, you will create another script for scraping off Competitive Coding problems from Codeforces. Though you can use any text editor (like VS Code) for writing the Python script, do check out **PyCharm** and **Jupyter notebook**.

Resources

Before starting with this assignment, you should learn about the basics of Python and web scraping. Here are some tutorials to get you started.

1. Python basics
   - `http://learnpython.org/`
   - `http://kaggle.com/learn/python`
   - `http://tutorialspoint.com/python/index.htm`
2. Selenium library
   - `https://www.selenium.dev/documentation/en/`
3. Web scraping in Python using Selenium
   - `https://selenium-python.readthedocs.io/`
   - `https://www.youtube.com/watch?v=Xjv1sY630Uc&list=PLzMcBGfZo4-n40rB1XaJ0ak1bemvlqumQ&t=0s`

Also make sure you know how to inspect elements in a website using Developer tools (`developers.google.com/web/tools/chrome-devtools/dom/`)

Task I : Moodle Auto-Login

In this part of the assignment, you will implement a moodle auto login script using selenium.

Overview

You need to create a python (.py) file which upon execution, will open a Chrome window, navigate to the page Moodle Login, enter your credentials, solve the captcha and log you in.

There are a few things you need to take care of here:

- Firstly, do **NOT** save your kerberos login details in the .py file. Instead, take the password as an input at the time of execution of python script.

  Note: Do NOT hardcode your password in the python file

- Secondly, there are various types of captcha present on Moodle, make sure your code covers them all.

Specifications

- Your directory structure should be like the following:
  ./Moodle/moodleLogin.py

Hints/Walkthrough

- **[IMP]** Make sure you have watched the first three videos of the playlist given in the resources here.

- Get the field **ids** that you need to change by inspect elements.

- There are four types of captcha on Moodle: add, subtract, first number and second number. While inspecting the elements, you will see that the entire div is contained in the element with the id: 'login'. You can store the entire text of this field by:

  ```
  text = driver.find_element_by_id("login").text
  ```

  Now you need to store two numbers from this text and also check whether this text contains 'first', 'second', 'add' or 'subtract' using python methods.

- Finally, decide what you need to enter in the captcha and enter it using the *send_keys* method.

Task II : Codeforces Scraper

For this part you will be required to create another script, this time for scraping off CP problems from Codeforces.

Overview

Codeforces has problems split across various contests where a problem is described the in the following format: <Contest Number ><Problem Label >, for eg. 1468D - codeforces.com/problemset/problem/1468/D.
Your task is to scrape off all the problems of a given contest numbers, and store the complete problem statement's screenshot, and also create input and output test files for the test cases given.

Specifications

Input

Your python script will be provided the contest number as an argument along with the file name at the time of running:

```
python fetch_round.py 1481
```

Output

These problems are to be downloaded in a hierarchical folder structure, with the contents of a problem kept in the directory ./<contest_number >/ <problem_label>/. Create these directories alongside your python file, i.e. inside the same directory as your python file.

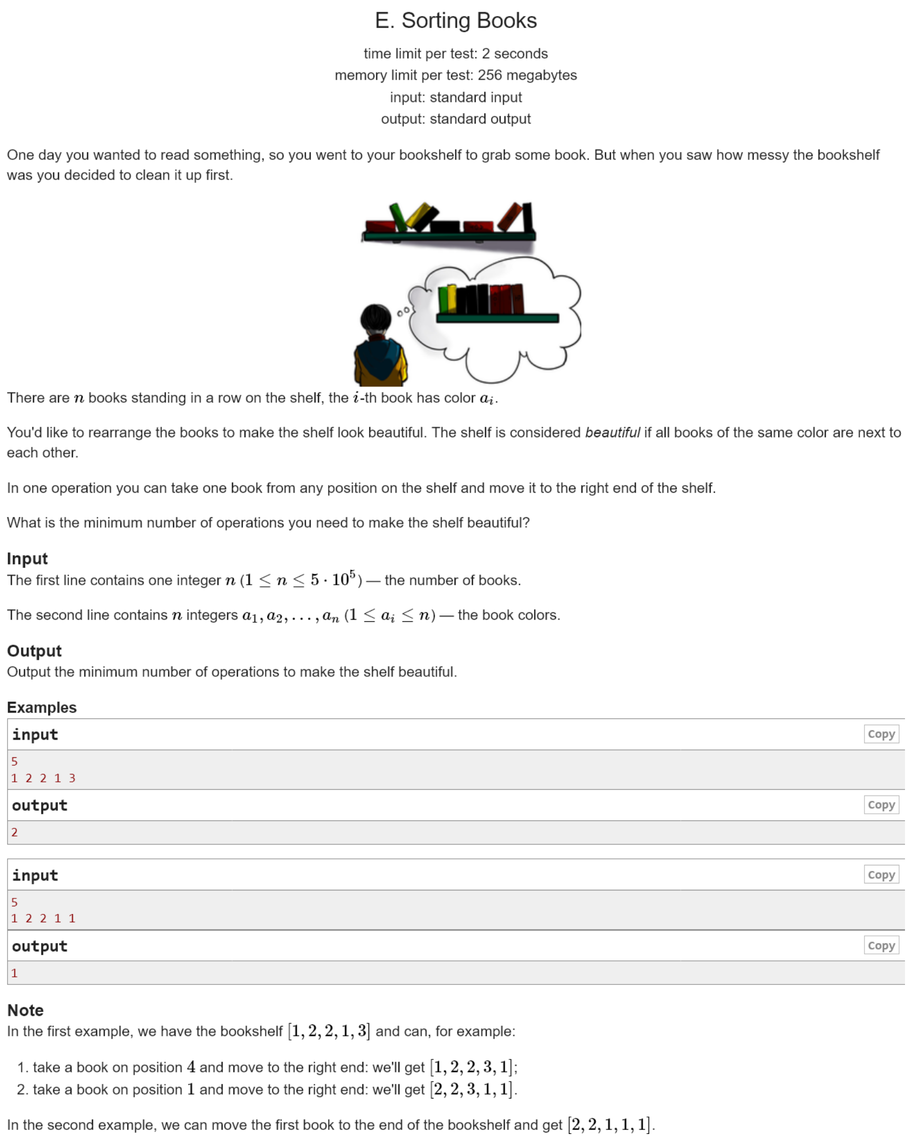
There should be a problem.png file containing the screenshot of the entire problem statement along with the input file and output file as shown on the next page.
Sample output

```
./1481/A/problem.png
./1481/A/input1.txt
./1481/A/output1.txt
./1481/B/*
./1481/C/*
./1481/D/*
./1481/E/problem.png
./1481/E/input1.txt
./1481/E/output1.txt
./1481/E/input2.txt
./1481/E/output2.txt
./1481/F/*
```

Notes

- A contest may have arbitrary number of problems. Make sure to iterate over all of the problems.
- In general, a codeforces problem may contain an arbitrary number of input and output test cases. For eg. Problem 1481E has 2 test cases, so you need to create 2 files for input and output each and name them in the fashion shown above.

Sample problem.png



E. Sorting Books

time limit per test: 2 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

One day you wanted to read something, so you went to your bookshelf to grab some book. But when you saw how messy the bookshelf was you decided to clean it up first.

There are n books standing in a row on the shelf, the i-th book has color a_i.

You'd like to rearrange the books to make the shelf look beautiful. The shelf is considered *beautiful* if all books of the same color are next to each other.

In one operation you can take one book from any position on the shelf and move it to the right end of the shelf.

What is the minimum number of operations you need to make the shelf beautiful?

Input
The first line contains one integer n ($1 \le n \le 5 \cdot 10^5$) — the number of books.

The second line contains n integers $a_1, a_2, \ldots, a_n$ ($1 \le a_i \le n$) — the book colors.

Output
Output the minimum number of operations to make the shelf beautiful.

Examples

input
```
5
1 2 2 1 3
```
output
```
2
```

input
```
5
1 2 2 1 1
```
output
```
1
```

Note
In the first example, we have the bookshelf $[1, 2, 2, 1, 3]$ and can, for example:

1. take a book on position 4 and move to the right end: we'll get $[1, 2, 2, 3, 1]$;
2. take a book on position 1 and move to the right end: we'll get $[2, 2, 3, 1, 1]$.

In the second example, we can move the first book to the end of the bookshelf and get $[2, 2, 1, 1, 1]$.

Sample input1.txt

```
5
1 2 2 1 3
```

Sample output1.txt

```
2
```

Bonus Tasks

- Expand your script so that it can scrape the problems of past **X** contents where you can enter the number X as an argument. The list of past contents is available here.
- Another route you can take is to scrape the problems of a particular difficulty range. Codeforces provides a filter here at the right hand side of your page. Expand your script to scrape the problems of difficulty range N1 to N2 where N1, N2 can be given as arguments. The number of problems you want to fetch has been left open ended. (You can explore navigation through the pages/pagination using selenium, here).

  Note: We have not given any strict specifications for the bonus tasks, so you are free to use your imagination. Show your creativity here and do write about it in the README.md file.

Conclusion

The purpose of this assignment is to familiarise you with Python and basics of web scraping. Don't worry if you are unable to complete some of the tasks, and don't worry too much about the strict input/ouput guidelines either. What's important is that you LEARN and enjoy the essence of coding and development! Feel free to modify the input and output formats, but do mention them as comments in your code.
Submit the link to your git repository containing the scripts for both the tasks here.