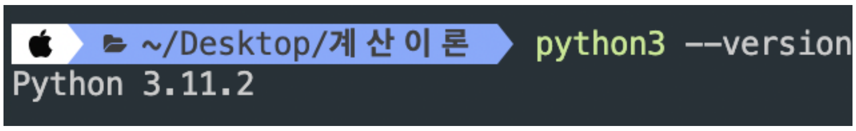
Environment

2021320017 김백규



```
 ~/Desktop/계산이론  python3 --version
Python 3.11.2
```

Algorithm & Explanation of the code + regular expression

I will explain what each function does in my python code. And which regular expression I used for implementing each function.

Problem 1

• return_something_between_AB_list(input_word)
Find "something" which is located between two (A or B). If "something" is "AA" or "AB" or "BA" or "BB" ("something" is ""), I excluded the "something" from the list.
For implementing the function, I used regular expression r'[AB].*[AB]'

• return_deleted_word(input_word)
I found "something" using return_something_between_AB_list function. And in this function, I deleted "something" from input_word unless each character in "something" is "A" or "B"
I didn't have to use regular expression for implementing this function.

• return_deleted_list(input_word)
In the for loop, I called return_deleted_word function to each word in input_word_list so that I can get a "something"-deleted-list.
I didn't have to use regular expression for implementing this function.

• Decrypt(deleted_input_word_list)
I found pattern that A or B appears 5 consecutive times. And I decrypted the pattern using ciper table.
For implementing the function, I used regular expression r'[A|B]{5}'

• return_decrypted_list(deleted_input_word_list)
In the for loop, I called decrypt function and could get a list whose word has decrypted.
I didn't have to use regular expression for implementing this function.

• return_result_URL
I joined list and deleted space and new line. Also I made each character lowercase. So I could get result_URL.
I didn't have to use regular expression for implementing this function.

• Main function using functions above
```
    input_file = open('input.txt', 'r')
    input_text = input_file.read()
    input_word_list = input_text.split(" ")
    deleted_input_word_list = return_deleted_list(input_word_list)
    decrypted_input_word_list = return_decrypted_list(deleted_input_word_list)
    result_URL = return_result_URL(input_word_list)
    print(result_URL)
    input_file.close()
```

Problem 2

• return_space_deleted_input_lines(input_lines)
This function is for deleting spaces and taps and newlines.
For implementing the function, I used regular expression r'\t|\n' and r' {1,}'

• return_input_lines_to_be_searched(input_lines)
this function is to return html body that contains professor introduction information.
So I cut to"교수소개" from <footer>
I didn't have to use regular expression for implementing this function.

• return_email_phone_homepage_list(input_lines)

In the for loop, I checked if each line contains email, phone, or homepage information.
If the line is <b>E-mail</b>, email information appears two lines after the line between <ahref> tags. If the line matches my regex for phone number, that line contains phone number information. If the line is <b>홈페이지:< \b>, homepage information appears one line after the line between <ahref> tags.
For implementing the function, I used regular expression **r'02-\d{3,4}-\d{4}' and r'<ahref[^>]*>(.*?)<\/a>'**

• Print_list(result, result_list), print_result(input_lines)
These functions are just for printing the results.
I didn't have to use regular expression for implementing these functions.

• Main function using functions above
```
    input_file = open('probelm2.html', 'r')
    input_lines = input_file.readlines()
    space_deleted_input_lines = return_space_deleted_input_lines(input_lines)
    input_lines_to_be_searched
        = return_input_lines_to_be_searched(space_deleted_input_lines)
    print_result(input_lines_to_be_searched)
    input_file.close()
```

Problem 3
Return_description_list(html_list)
This function is for returning description strings for each html files. The description string is located between <p data-testid="vuln-description"> <\/p> this HTML tags.
For implementing the function, I used regular expression **r'<p data-testid="vuln-description">(.*?)(<\/p>|\n)'**

• return_tagged_list(description_list)
In this function, I tokenized description string using nltk.word_tokenize and tagged using nltk.pos_tag.
I didn't have to use regular expression for implementing these functions.

• return_noun_list(tagged_list), return_verb_list(tagged_list)
These functions are to return noun_list and verb_list in tagged_list.
I didn't have to use regular expression for implementing these functions.

• return_noun_dict(noun_list), return_verb_dicr(verb_list)
These functions are to return noun_dict and verb_dict with counting each frequencies.
I didn't have to use regular expression for implementing these functions.

• Print_result
This function is to print top 10 frequent words. I concatenated two dictionaries and sorted so I could print to 10 frequent words.
I didn't have to use regular expression for implementing these functions.

• Main function using functions above
```
    (I skipped the 10 HTML files opening process.)
    (I skipped 10 readlines() function calls for 10 HTML files)
    html_list = [html1_lines, html2_lines, html3_lines, html4_lines, html5_lines,
                 html6_lines, html7_lines, html8_lines, html9_lines, html10_lines]
    description_list = return_description_list(html_list)
    tagged_list = return_tagged_list(description_list)
    noun_list = return_noun_list(tagged_list)
    verb_list = return_verb_list(tagged_list)
    noun_dict = return_noun_dict(noun_list)
    verb_dict = return_verb_dict(verb_list)
    print_result(noun_dict, verb_dict)
    (I skipped the 10 HTML files closing process.)
```

Answer

Problem 1
problem2'slinkhttps://www.notion.so/ccs-binary/hw-4-6f0a449e2836487e89297bdc5f67b014problem3'slinkhttps://www.notion.so/ccs-binary/hw-4-d354496ebb4f49c4aaca53bf2e2dde07

problem 2
E-mail: heejo@korea.ac.kr, hyunwoojkim@korea.ac.kr, suhtw@korea.ac.kr, ejhoon@korea.ac.kr, hakjoo_oh@korea.ac.kr, jbhur@isslab.korea.ac.kr, hoh_in@korea.ac.kr, yuhc@korea.ac.kr, seungryong_kim@korea.ac.kr, gunjaekoo@korea.ac.kr

Phone Number: 02-3290-3208, 02-3290-4604, 02-3290-2397, 02-3290-4846, 02-3290-4601, 02-3290-4603, 02-3290-3206, 02-3290-2392, 02-3290-4608, 02-3290-4607

HomePage: https://ccs.korea.ac.kr, https://mlv.korea.ac.kr, http://esca.korea.ac.kr, https://sites.google.com/site/mplabku, http://prl.korea.ac.kr, https://sites.google.com/korea.ac.kr/isslab, https://ibel.korea.ac.kr/, https://ds.korea.ac.kr, https://cvlab.korea.ac.kr/, https://csarch.korea.ac.kr/

Problem 3
1. is - 6
2. vulnerability - 5
3. code - 4
4. Odoo - 4
5. attackers - 3
6. memory - 3
7. execution - 3
8. Server - 3
9. issue - 3
10. allows - 3