Theory of Computation  
Programming Assignment Report

2019320137  
Sangmin Hwang

**Environment**

- OS: Windows 10 Home (x64)

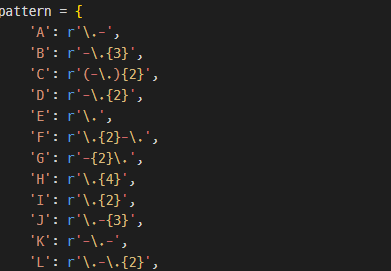
- Language: Python 3.7.4

- Editor: Visual Studio Code

1. Problem 1

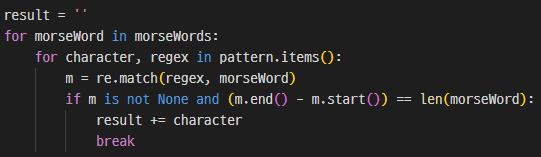
|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| A | \.- | B | -\.{3} | C | (-\.){2} | D | -\.{2} |
| E | \. | F | \.{2}-\. | G | -{2}\. | H | \.{4} |
| I | \.{2} | J | \.-{3} | K | -\.- | L | \.-\.{2} |
| M | -{2} | N | -\. | O | -{3} | P | \.-{2}\. |
| Q | -{2}\.- | R | \.-\. | S | \.{3} | T | - |
| U | \.{2}- | V | \.{3}- | W | \.-{2} | X | -\.{2}- |
| Y | -\.-{2} | Z | -{2}\.{2} | 0 | \-{5} | 1 | \.{1}-{4} |
| 2 | \.{2}-{3} | 3 | \.{3}-{2} | 4 | \.{4}-{1} | 5 | \.{5} |
| 6 | -{1}\.{4} | 7 | -{2}\.{3} | 8 | -{3}\.{2} | 9 | -{4}\.{1} |
| . | (\.-){3} | , | -{2}\.{2}-{2} | ? | \.{2}-{2}\.{2} | ‘ | \.-{4}\. |
| ! | (-\.){2}-{2} | / | -\.{2}-\. | ( | -\.-{2}\. | ) | -\.-{2}\.- |
| & | \.-\.{3} | : | -{3}\.{3} | ; | (-\.){3} | = | -\.{3}- |
| + | (\.-){2}\. | - | -\.{4}- | \_ | \.{2}-{2}\.- | “ | \.-\.{2}-\. |
| $ | \.{3}-\.{2}- | @ | \.-{2}\.-\. | ¿ | \.{2}-\.- | ¡ | -{2}\.{3}- |

* 1. Regular Expression
  2. Code & Algorithm Description

  
At first, store the regular expressions with a character in the form of a dictionary.



Then, read a text file ‘text1.txt’ containing the given morse code and split the text by whitespaces. Each morse code separated by a space are now stored as an element of a list.

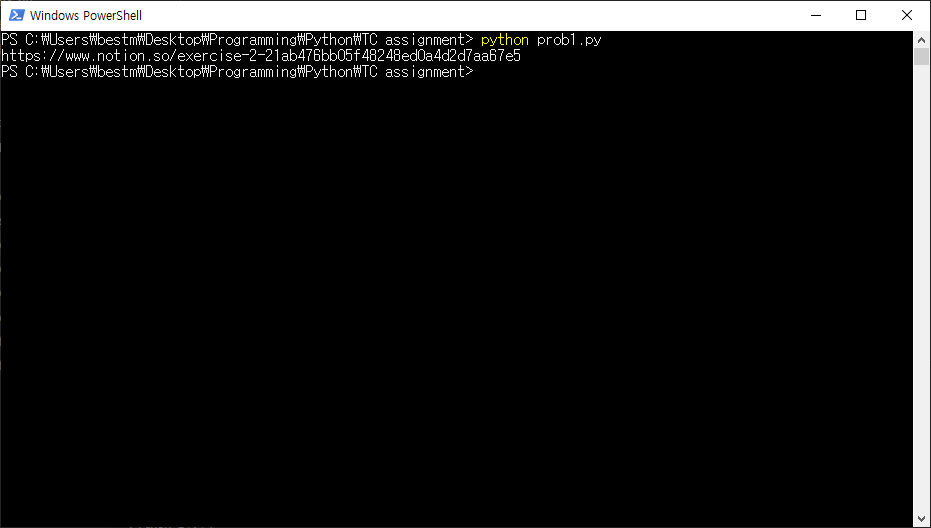


for each single morse code, we tests whether the code could be converted into a certain characater. If ‘m’ is not none and the length of m (m.end() – m.start()) equals the length of the code, we can convert it into the character. To prevent converting partial matches such as ‘.---‘ to ‘A--‘, we need to check whether the length of the matched string(m) equals the length of given code(morseWord).



After loop, the converted string is stored in the ‘result’ as a upper case letters. Convert it into lower case letter, then we get the secret URL.

* 1. Resulting output



https://www.notion.so/exercise-2-21ab476bb05f48248ed0a4d2d7aa67e5

1. Problem 2
   1. Regular Expression

1) social security number:  
\s((98|99|01)((0[13578]|1[02])(0[1-9]|[12]\d|30|31)|(0[468]|11)(0[1-9]|[12]\d|30)|02(0[1-9]|1\d|2[0-8]))|00((0[13578]|1[02])(0[1-9]|[12]\d|30|31)|(0[468]|11)(0[1-9]|[12]\d|30)|02(0[1-9]|1\d|2[0-9])))-[1-4]\d{6}\s

2) email: \s[a-zA-Z0-9]+@[a-zA-Z]+\.ac\.kr\s

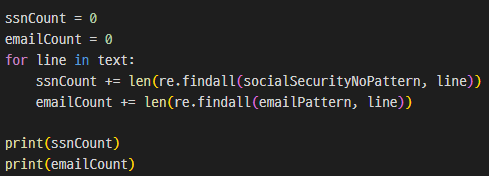
* 1. Code & Algorithm Description



Store the pattern for social security number and email.



read the given text file. As the text is multi-line, read the file line by line using readlines().



for each line in text, we count how many valid SSNs and emails are in there. As re.findall() function returns all of the matches as a list of a string, we can count how many matches are there by using len() of the return value.

* 1. Resulting output

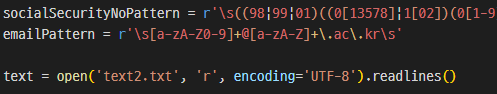


There are 14 valid SSNs and 7 valid emails.

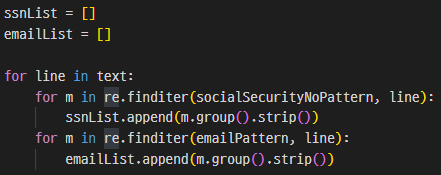
1. Problem 3
   1. Regular Expression

Same as Problem 2, but **replace pattern** is used. (See code explanation)

* 1. Code & Algorithm Description

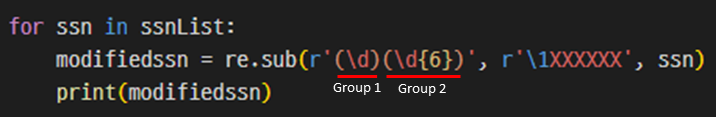


Initial steps are same as problem 2. Store the regex for SSN and email, and read text file line by line.



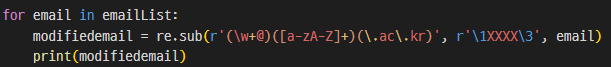
In this time, we store all valid SSNs and emails in a list. Using finditer() for getting the exact string and strip() for removing spaces at the beginning/end, all of the SSNs are stored in ssnList and emails in emailList.

ex)   
ssnList = [‘000229-3817291’, ‘980131-4817203’],   
emailList = [‘acde@korea.ac.kr’, ‘uytaw@korea.ac.kr’]



Then, use re.sub to anonymize SSN. The latter part of SSN matches the pattern ‘(\d)(\d{6})’, and the first digit is classified as group 1 and the remaining 6 digits are group 2. Using **group referencing** (‘\1’) of regex, we can design a replace pattern as ‘\1XXXXXX’. This means we leave only the group 1 followed by XXXXXX. Group 2 part, the last 6 digits are discarded and replaced by XXXXXX.

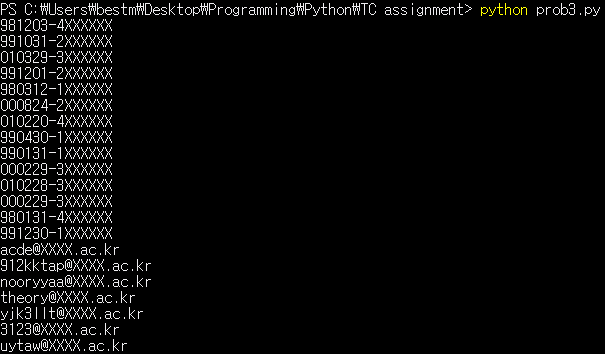
ex) In ‘000229-3817291’, the latter part ‘3817291’ matches ‘(\d)(\d{6})’. Group 1 is ‘3’ and group 2 is ‘817291’. Due to the replace pattern, only group 1 part is left and XXXXXX is concatenated.



Same process is done to anonymize emails. We divide the email into three groups, and replace group 2 part into XXXX.

ex) In ‘acde@korea.ac.kr’, group 1 is ‘acde@’, group 2 is ‘korea’, group 3 is ‘ac.kr’. Only group 1 and 3 is left and ‘korea’ is replaced into XXXX.

* 1. Resulting output

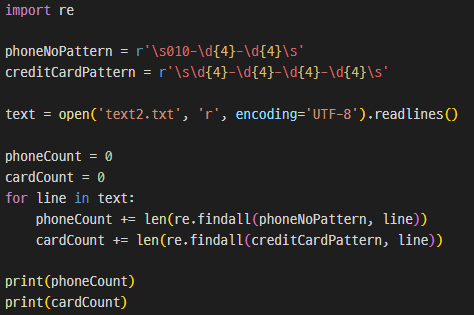


1. Extra Credit Problem
   1. Regular Expression

1) Cellphone Number: \s010-\d{4}-\d{4}\s

2) Credit Card Number: \s\d{4}-\d{4}-\d{4}-\d{4}\s

* 1. Code & Algorithm Description



Steps are same as problem 2. Using findall() to find all matches in the text, and count occurrences by using len().

* 1. Resulting output



There are 8 valid phone numbers and 7 valid credit card numbers.