HW#4 Report

2020320094 Jeonghoon Rho 노정훈

1. Environment

OS: macOS 11.3.1
Programming Language: Python 3.8.8
RE Library: Python re

2. Explanation

Code

```
42  def main(file, prob_idx):
43      result = []
44      # open problem file
45      try:
46          with open(file, 'r') as f:
47              data = f.read()
48      except FileNotFoundError as e:
49          print("[ERROR] FILE NOT FOUND!")
50      split_data = data.splitlines()
51
52      # Run the regex_play function line by line
53      for line in split_data:
54          result.append(regex_play(line, prob_idx))
55      result = list(filter(None, result))
56      w_data = '\n'.join(result)
57
58      try:
59          with open('output_'+prob_idx+'.txt', 'w', -1, "utf-8") as file:
60              file.write(w_data)
61              file.close()
62      except FileNotFoundError as e:
63          print("[ERROR] FILE NOT FOUND!")
64      print(w_data)
65
66
67  if __name__ == '__main__':
68      if(not sys.argv[1]):
69          print("""USAGE: python3 "FILE NAME" "problem NUMBER" """)
70      if(not sys.argv[2]):
71          print("""USAGE: python3 "FILE NAME" "problem NUMBER" """)
72      file = sys.argv[1]
73      prob_idx = sys.argv[2]
74      main(file, prob_idx)
```

By the code template, the program receives the name of input txt file and problem index.
45~50 line
First, the program open the file whose name is equal to argument and split the txt file by '\n'.

53~56 line
And for each line, that is a line of string in input file, call a function regex_play with the line as argument. The return value of function is appended at result, which is initialized as empty list in 43rd line. And the program filters None value in list and assign the filtered result at result in list-form. And in 56th line, make the list into string with \n by join method to write it in output file.
In this section, it is trivial that regex_play must return None in case that regex doesn't match to the string and return the string itself in case that regex matches to the string. Because None value is filtered in 55th line and strings are joined in w_data(the data to write)

58~64 line
In this section the program writes the data at the output file and prints them.

67~74 line
Usage validity check

```python
import re

def regex_play(str, prob_idx):
    result = ""
    if prob_idx == '1':
        print("==problem1==")
        regex = r"https?://www[.](cyworld|facebook|instagram|twitter)[.]com/[a-z]+$"
        match = re.match(regex, str, re.I)
        if match:
            result = match.group()

    elif prob_idx == '2':
        print("==problem2==")
        regex = r"([a-zA-Z][\w.]*@[a-z]+[.](ac[.]kr|com|net|co[.]kr)$|((25[0-5]|2[0-4]\d|1\d\d|[1-9]?\d)[.]){3}((25[0-5]|2[0-4]\d|1\d\d|[1-9]?\d))$|(\d{4}-){3}\d{4}$)"
        match = re.match(regex, str)
        if match:
            result = match.group()

    elif prob_idx == '3':
        print("==problem3==")
        regex = r"((?=.{10,})((?P<con>[a-zA-Z~`!@#$%^&*()\-_+={}/:;<>,.?\\\'\"|[\]])((?!(?P=con){2})|(?P=con)(?!(?P=con)))+$)|((?!(\d+)$)(?!([a-zA-Z]+)$)(?!([~`!@#$%^&*()\-_+={}
        match = re.match(regex, str)
        if match:
            result = match.group()
    else:
        print("[ERROR] WRONG PROBLEM NUMBER")
        exit(1)

    return result
```

As I wrote above, regex_play must return None(empty string, “”) or the string.
It works differently by the prob_idx. result is initialized as None.
For each prob_idx, I used re module. In the module, it has match method which works with argument, regex, str, compile option. The method compiles the input regex with any compile option, matches the compiled regex with input string from first character and returns ‘match’ object(In the case regex doesn’t match to string, it is null). match object has group method which returns the matched string.
So, in each cases, I made the regex and matched it to str argument by re.match(). If it matches, return the result as match.group(), that is the string itself. Otherwise, result is returned as “”.
And I used raw string to avoid confusion in back slash problem.

regex-problem1

```
regex = r"https?://www[.](cyworld|facebook|instagram|twitter)[.]com/[a-z]+$"
match = re.match(regex, str, re.I)
```

```
https?://www[.](cyworld|facebook|instagram|twitter)[.]com/[a-z]+$
```

It is a SNS link whose form is A..A://www.B..B.com/C..C.

A..A is a protocol(http, https). So I designed it as https? because s has 0 or 1 repetition.

://www[.] is same as the form. (Dot is meta character, so I used [].)

B..B is the name of site(cyworld, facebook, Instagram, twitter). I used | (or)(same as + operation in regular expression of theory of computation) to allow all cyworld, facebook, Instagram, and twitter.

[.]com/ is also same as the form.

C..C is a user ID(any length of alphabets), so I used [a-z], set of characters between a and z, that is every alphabet , and + operation, which means repetition more than 0. So [a-z]+ means any length of alphabets.

$ means the end of string. By this, the matching must be started at the start point of the string and ended in the end of string.

And I used re.IGNORECASE(re.I) compile option to allow both uppercase and lowercase of alphabet.

result-problem1

https://www.facebook.com/blackboard
https://www.cyworld.com/loveisgone
https://www.instagram.com/peaches
https://www.twitter.com/arden
http://www.cyworld.com/tjtjlee
http://www.cyworld.com/pororo
http://www.facebook.com/ababcdcd
https://www.facebook.com/alanturing
http://www.instagram.com/asddsgdgc
http://www.instagram.com/spopopopo
https://www.cyworld.com/kucose
http://www.twitter.com/peterlinz
https://www.facebook.com/iotcube
https://www.instagram.com/instagram
http://www.facebook.com/kalilinux
https://www.instagram.com/ilovecs
https://www.cyworld.com/coldfeet
http://www.facebook.com/dfamaster
http://www.cyworld.com/freezing

```
1  https://www.facebook.com/blackboard
2  https://www.cyworld.com/loveisgone
3  https://www.instagram.com/peaches
4  https://www.twitter.com/arden
5  http://www.cyworld.com/tjtjlee
6  http://www.cyworld.com/pororo
7  http://www.facebook.com/ababcdcd
8  https://www.facebook.com/alanturing
9  http://www.instagram.com/asddsgdgc
10 http://www.instagram.com/spopopopo
11 https://www.cyworld.com/kucose
12 http://www.twitter.com/peterlinz
13 https://www.facebook.com/iotcube
14 https://www.instagram.com/instagram
15 http://www.facebook.com/kalilinux
16 https://www.instagram.com/ilovecs
17 https://www.cyworld.com/coldfeet
18 http://www.facebook.com/dfamaster
19 http://www.cyworld.com/freezing
```

regex-problem2

([a-zA-Z][\w.]*@[a-z]+[.](ac[.]kr|com|net|co[.]kr)$|((25[0-5]|2[0-4]\d|1\d\d|[1-9]?\d)[.]){3}((25[0-5]|2[0-4]\d|1\d\d|[1-9]?\d))$|(\d{4}-){3}\d{4}$)

```
regex = r"([a-zA-Z][\w.]*@[a-z]+[.](ac[.]kr|com|net|co[.]kr)$|((25[0-5]|2[0-4]\d|1\d\d|[1-9]?\d)[.]){3}((25[0-5]|2[0-4]\d|1\d\d|[1-9]?\d))$|(\d{4}-){3}\d{4}$)"
```

```
([a-zA-Z][\w.]*@[a-z]+[.](ac[.]kr|com|net|co[.]kr)$|
((25[0-5]|2[0-4]\d|1\d\d|[1-9]?\d)[.]){3}((25[0-5]|2[0-4]\d|1\d\d|[1-9]?\d))$|
(\d{4}-){3}\d{4}$)
```

I used |(or) to combine three cases, e-mail address, IP address and Credit card number.

(1) e-mail address, Form: X..X@Y..YP : *([a-zA-Z][\w.]*@[a-z]+[.](ac[.]kr|com|net|co[.]kr)$*

X..X is any length of number, alphabet, . and _ with first letter as an alphabet.

So, I used [a-zA-Z], an alphabet(lower or upper) at first, and *(repetition of 0 or more) of [\w.], which means alphabet, number, underscore and dot.

@ is followed by.

Y..Y is any lenth of lowercase alphabet. So it is [a-z]+. It is + rather than * because its length is more than 0.

[.] is followed by and P is (.ac.kr, .com, .net, .co.kr). So they arecombined by |.

$ means the end of the string.

(2) IP address, Form : A.B.C.D: *((25[0-5]|2[0-4]\d|1\d\d|[1-9]?\d)[.]){3}((25[0-5]|2[0-4]\d|1\d\d|[1-9]?\d))$*

A,B,C,B should be a number in 0~255.

In A.B.C., the same form is repeated. So I used {3} to avoid repetition.

So it is the form, ((number)[.]){3}(number)$.

And number must be in interval, [0,255].

In 250~255, first and second number is fixed as 2 and third number is [0-5]. 25[0-5]

In 200~249, first is fixed as 2 and second is [0-4], and last number is any number, so I used \d, which means [0-9] 2[0-4]\d

In 100~199, first is fixed and second and last number are any number. 1\d\d

In 10~99, first number is [1-9] and second is any number.

In 0~9, only one any number.

In this last two cases, last number is any number and first number is [1-9] or none. So I used ?(0 or 1 repetition) in [1-9]. [1-9]?\d

(3) Credit card number, Form: AAAA-BBBB-CCCC-DDDD: *(\d{4}-){3}\d{4}$)*

AAAA, BBBB, CCCC, and DDDD should be a 4-digit.

And like (2), (4-digit)- form is repeated 3 times.

So It is the form, ((4-digit)-){3}(4-digit).

In this case, 0 is allowed in any space. So 4-digit can be expressed in \d{4}, which means 4 repetition of any number.

result-problem2

112.21.244.195
3856-3586-3586-2393
dev.el_0per@framework.net
224.10.160.174
1235-3457-5967-3874
242.114.177.250
KUcose215@gmail.com
2979-2469-5393-2923
naver.com@naver.com
247.187.4.178
2856-8857-9724-3755
slave.of.assignment@twitter.com
80.164.177.212
58.107.245.210
1234-5678-8765-4321
123.251.216.231
heejo@korea.ac.kr
4191-8195-1274-3346
49.75.5.234
h3ll_0@wel.com
227.252.84.114
217.243.251.226
99.34.231.234
pigeon@yonsei.ac.kr
48.78.44.164
imcute._.@really.co.kr
90.82.198.218
5118-6653-5950-1085
180.184.52.221
159.4.224.238
3485-9149-2509-6573
c_c_s@korea.ac.kr
170.58.4.179
0.78.156.198
lab_rador@daum.net
T0chAck3r@twitter.com
100.147.183.30
9674-3409-7834-5879
admire_prof.heejo@com.com
144.43.0.136

```
112.21.244.195
3856-3586-3586-2393
dev.el_0per@framework.net
224.10.160.174
1235-3457-5967-3874
242.114.177.250
KUcose215@gmail.com
2979-2469-5393-2923
naver.com@naver.com
247.187.4.178
2856-8857-9724-3755
slave.of.assignment@twitter.com
80.164.177.212
58.107.245.210
1234-5678-8765-4321
123.251.216.231
heejo@korea.ac.kr
4191-8195-1274-3346
49.75.5.234
h3ll_0@wel.com
227.252.84.114
217.243.251.226
99.34.231.234
pigeon@yonsei.ac.kr
48.78.44.164
imcute._.@really.co.kr
90.82.198.218
5118-6653-5950-1085
180.184.52.221
159.4.224.238
3485-9149-2509-6573
c_c_s@korea.ac.kr
170.58.4.179
0.78.156.198
lab_rador@daum.net
T0chAck3r@twitter.com
100.147.183.30
9674-3409-7834-5879
admire_prof.heejo@com.com
144.43.0.136
```

regex-problem3

((?=.{10,})((?P<con>[a-zA-Z~`!@#$%^&*()\-_+={}/:;<>,.?\\\'\"|[\]])((?!(?P=con){2})|(?P=con)(?!(?P=con))))+$)|((?!(\d+)$)(?!([a-zA-Z]+)$)(?!([~`!@#$%^&*()\-_+={}/:;<>,.?\\\'\"|[\]]+)$)(?=.{8,})((?P<ch>[a-zA-Z0-9~`!@#$%^&*()\-_+={}/:;<>,.?\\\'\"|[\]])((?!(?P=ch){2})|(?P=ch)(?!(?P=ch))))+$)

```
((?=.{10,})((?P<con>[a-zA-Z~`!@#$%^&*()\-
_+={}/:;<>,.?\\\'\"|[\]])((?!(?P=con){2})|(?P=con)(?!(?P=con))))+$)|(
(?!(\d+)$)(?!([a-zA-Z]+)$)(?!([~`!@#$%^&*()\-
_+={}/:;<>,.?\\\'\"|[\]]+)$)(?=.{8,})((?P<ch>[a-zA-Z0-9~`!@#$%^&*()\-
_+={}/:;<>,.?\\\'\"|[\]])((?!(?P=ch){2})|(?P=ch)(?!(?P=ch))))+$)
```

The password has some rules.
(1) 3 or more consecutive characters are not allowed
(2) 8 or more length string with 2 or more types of characters
or
(3) 10 or more length string with only one type of character without digits
Types of characters are alphabets, digits and special characters.

To check whether password contains 3 or more consecutive character, I used () grouping and (?!..). (?!..) matches if .. doesn't match next string. It is called a negative lookahead assertion. And () grouping can have name and be reused latter. It can be Named by (?P<name>..) and reused by (?P=name). So, to check 3 consecutive characters, I used the form, (?P<name> a character)(?!(?P=name){2})|(?P=name)(?!(?P=name)), which means a single character or 2 consecutive characters.

To allow (3) rule, it has 10 or more length of alphabets or special characters. To check length, I used a lookahead assertion, (?=..). It matches if … matches next. So, (?=.{10,}) matches if it is a string whose length is 10 or more.
`[a-zA-Z~`!@#$%^&*()\-_+={}/:;<>,.?\\\'\"|[\]]`

Above characters matches to alphabets and special characters. So in the form (?P<name> a character)(?!(?P=name){2})|(?P=name)(?!(?P=name)), a character is substituted by above characters. It doesn't check it has only one type of character because in 2 or more types of characters, it satisfy the rule (2).
I named the character, con because it has consistent type.
((?=.{10,})((?P<con>[a-zA-Z~`!@#$%^&*()\-_+={}/:;<>,.?\\\'\"|[\]])((?!(?P=con){2})|(?P=con)(?!(?P=con))))+$)

To allow rule (2), all the expressions all almost same as (3) excepting for the length and type . It allows numbers and length is 8 or more.

So, It checks the length by (?=.{8,}).

And, its character-range is ,
`[a-zA-Z0-9~`!@#$%^&*()\-_+={}/:;<>,.?\\\'\"|[\]]`
, where the numbers(0-9) are added.

And it should have 2 or more types of characters. So I checked whether the password consis ts of only one type of characters by (?!..).

So, (?!(numbers)+$) checks if the string consists of only numbers, (?!(alphabets)+$) works same, and (?!(special characters)+$) also. Combine them:
((?!(\d+)$)(?!([a-zA-Z]+)$)(?!([~`!@#$%^&*()\-_+={}/:;<>,.?\\\'\"|[\]]+)$)(?=.{8,})((?P<ch>[a-zA-Z0-9~`!@#$%^&*()\-_+={}/:;<>,.?\\\'\"|[\]])((?!(?P=ch){2})|(?P=ch)(?!(?P=ch))))+$)

And combine two password re by |.
((?=.{10,})((?P<con>[a-zA-Z~`!@#$%^&*()\-_+={}/:;<>,.?\\\'\"|[\]])((?!(?P=con){2})|(?P=con)(?!(?P=con))))+$)|((?!(\d+)$)(?!([a-zA-Z]+)$)(?!([~`!@#$%^&*()\-_+={}/:;<>,.?\\\'\"|[\]]+)$)(?=.{8,})((?P<ch>[a-zA-Z0-9~`!@#$%^&*()\-_+={}/:;<>,.?\\\'\"|[\]])((?!(?P=ch){2})|(?P=ch)(?!(?P=ch))))+$)

result-problem3

iOGDagosypm
XmaOSwAExrllr
^3EIJA3l_*
<6;@@5]'
1LU4ct66emdPtT
zQZyx}dA
n5&e7XA<@
NzG,f&QNTq
ACbzvCu/%Cey
pDQShPaZeHEW
JJQMVok9v
CXccfBNeupe
U!>j}`|Z1M
2,_1"041&2!
Zb&SmRbJC!
t21Zv@12&4)e2y
CuurzlMsQWqNlCR
'$;+=[{<}-${
[QP#y>dx|:oX.<$
/!'=~`!#:~,
Jghitao1E
!_>6[3&2'+-%,8
h\n&^F>Sn^%w
B;a?<I5!FUO
1[LaZhot
JH5zbMC96
"}Ly};4m9M~2
.Y./]p=Xp||sH
DRQubxXhR5d
@1>&2~_
S}E*]xyI]H;R
},/\>$~%-{
\\UiKX7N
FOnngtQuSohQtQ
\3#!=@5~:-];\]:
N~Or&JV-V9"C~c
.3+?:-@!!8>;]
fJHlbLMQYkkx
TFL]m8\;
)C[dd&lmp^
DObAh7Z9
<@</.{_:|[.
&$"`:)(%\\{_<
L4=H#?BJ

```
iOGDagosypm
XmaOSwAExrllr
^3EIJA3l_*
<6;@@5]'
1LU4ct66emdPtT
zQZyx}dA
n5&e7XA<@
NzG,f&QNTq
ACbzvCu/%Cey
pDQShPaZeHEW
JJQMVok9v
CXccfBNeupe
U!>j}`|Z1M
2,_1"041&2!
Zb&SmRbJC!
t21Zv@12&4)e2y
CuurzlMsQWqNlCR
'$;+=[{<}-${
[QP#y>dx|:oX.<$
/!'=~`!#:~,
Jghitao1E
!_>6[3&2'+-%,8
h\n&^F>Sn^%w
B;a?<I5!FUO
1[LaZhot
JH5zbMC96
"}Ly};4m9M~2
.Y./]p=Xp||sH
DRQubxXhR5d
@1>&2~\_
S}E*]xyI]H;R
},/\>$~%-{
\\UiKX7N
FOnngtQuSohQtQ
\3#≠@5~:-];\]:
N~Or&JV-V9"C~c
.3+?:-@!!8>;]
fJHlbLMQYkkx
TFL]m8\;
)C[dd&lmp^
DObAh7Z9
<@</.{_:|[.
&$"`:)(%\\{_<
L4=H#?BJ
```