Practice Question

In [ ]:
```
i=1
while 1:
    print(i,end="")  #1      #2     #3     #4     #5      #6     #7     #8     #9      #10
    i+=1             #2      #3     #4     #5     #6      #7     #8     #9     #10     #11
    if i>10:         #2>10  3>10  4>10   5>10   6>10    7>10   8>10   9>10  10>10   11>10
        break

a. 1 2 3 4 5 6 7 8 9 10    #Correct
b. 1
c. 0
d. Infinite Loop
```

re.findall () Function

In [ ]:
```
re.findall Function --> find all occurances of a pattern.
                    --> If will return occurances on the basic of list
                    --> The re. findall(pattern, string) method scans string from left to right,
                        searching for all non-overlapping matches of the pattern .
                        It returns a list of strings in the matching order when scanning the string
                        from left to right.
```

Examples

In [2]:
```
import re
matcher = re.findall("[0-9]","a7b698uytewd")
matcher
```
Out[2]:
```
['7', '6', '9', '8']
```

In [3]:
```
import re
matcher = re.findall("[^0-9]","a7b698uytewd")
matcher
```
Out[3]:
```
['a', 'b', 'u', 'y', 't', 'e', 'w', 'd']
```

re.sub() Function

In [ ]:
```
re.sub() --> The sub() function searches for the pattern in the string and replaces
             the matched strings with the replacement
         -->sub means substitution , or replacement
Syntax:
    re.sub(characterclass ,symbol , string)
```

Example

In [4]:
```
import re
s=re.sub("[a-z]","#","abcdeg$$$123")
s
```
Out[4]:
```
'######$$$123'
```

In [5]:
```
import re
s=re.sub("[0-9]","*","asdbvcd$$$123")
s
```
Out[5]:
```
'asdbvcd$$$***'
```

re.subn() Function

In [ ]:
```
re.subn() --> substitution or replace and also return how many character it has replaced .
     --> The re.subn() function is the same as the re.sub() function, except that
         it also provides a count of the number of replacements that it has done.

Syntax:
        re.subn(characterclass ,symbol , string)
```

Example

In [6]:
```
import re
s=re.subn("[0-9]","*","asdbvcd$$$123")
s
```
Out[6]:
```
('asdbvcd$$$***', 3)
```

In [7]:
```
import re
s=re.subn("[^a-zA-Z0-9]","99","abc##98 9823ab%%^as")
s
```
Out[7]:
```
('abc999998999823ab999999as', 6)
```

In [8]:
```
import re
s=re.subn("[a-kA-C0-5]","00","abc##98 9823ab%%^as")
s
```
Out[8]:
```
('000000##98 9800000000%%^00s', 8)
```

^ Symbol

In [ ]:
```
^ Symbol --> check weather the given target string starts with our provided pattern or not.If the
             target string is not started with our provided pattern then it will return None.
```

Example

In [9]:
```
import re
s="learning python is easy"
res = re.search("^Easy",s)
if res!=None:
    print("Target start with our matching string")
else:
    print("Target is not started on our katching string")
```
```
Target is not started on our katching string
```

In [9]:
```
import re
s="learning python is easy"
res = re.search("^learning",s)
if res!=None:
    print("Target start with our matching string")
else:
    print("Target is not started on our katching string")
```
```
Target start with our matching string
```

Dollar Symbol

In [ ]:
```
$ Symbol --> check weather the given target string end with our provided pattern or not.
             If the target string is not ended with our provided pattern then it will return None.
```

In [10]:
```
import re
s="learning python is easy"
res = re.search("easy$",s)
if res!=None:
    print("Target start with our matching string")
else:
    print("Target is not started on our katching string")
```
```
Target start with our matching string
```

In [11]:
```
import re
s="learning"
res = re.search("^learning$",s)
if res!=None:
    print("Target start with our matching string")
else:
    print("Target is not started on our katching string")
```
```
Target start with our matching string
```

Pre-defined Character classes

In [ ]:
```
Pre-defined Character classes:1
    1. \s --> Space Character
    2. \S --> Any character except Space Character
    3. \d --> any digit from 0 to 9
    4. \D --> All chracters except digits
    5. \w --> any word character[a-zA-z0-9]
    6. \W --> any character except [a-zA-z0-9]
    7. "."  --> any character including special characters

    Note : dot class will considered all types of symbols except new line symbol(\n)
```

Example of Each Type of Character Classes

\s --> Space Character

In [12]:
```
import re
matcher = re.finditer("\s","abc kl m")
for i in matcher:
    print(str(i.start())+"    "+str(i.end())+"    "+str(i.group()) )
          #3                        #4
          #6                        #7
```
```
3    4
6    7
```

\S -->Any character except Space Character

In [13]:
```
import re
matcher = re.finditer("\S","abc kl m")
for i in matcher:
    print(str(i.start())+"    "+str(i.end())+"    "+str(i.group()) )
        #0                          #1                     a
        #1                          #2                     b
        #2                          #3                     c
        #4                          #5                     k
        #5                          #6                     l
        #7                          #8                     m
```
```
0    1    a
1    2    b
2    3    c
4    5    k
5    6    l
7    8    m
```

\d --> any digit from 0 to 9

In [17]:
```
import re
matcher = re.finditer("\d","abc1kl2m3")
for i in matcher:
    print(str(i.start())+"    "+str(i.end())+"    "+str(i.group()) )
         #3                        #4                 1
         #6                        #7                 2
         #8                        #9                 3
```
```
3    4    1
6    7    2
8    9    3
```

\D -->All chracters except digits

In [18]:
```
import re
matcher = re.finditer("\D","abc1kl2m3")
for i in matcher:
    print(str(i.start())+"    "+str(i.end())+"    "+str(i.group()) )
          #0                        #1                       a
          #1                        #2                       b
          #2                        #3                       c
          #4                        #5                       k
          #5                        #6                       l
          #7                        #8                       m
```
```
0    1    a
1    2    b
2    3    c
4    5    k
5    6    l
7    8    m
```

\w --> any word character[a-zA-z0-9]

In [19]:
```
import re
matcher = re.finditer("\w","ab$%12")
for i in matcher:
    print(str(i.start())+"    "+str(i.end())+"    "+str(i.group()) )
       #0                           #1                     a
       #1                           #2                     b
       #4                           #5                     1
       #5                           #6                     2
```
```
0    1    a
1    2    b
4    5    1
5    6    2
```

\W --> any character except [a-zA-z0-9]

In [20]:
```
import re
matcher = re.finditer("\W","ab$%12")
for i in matcher:
    print(str(i.start())+"    "+str(i.end())+"    "+str(i.group()) )
      #2                           #3                       $
      #3                           #4                       %
```
```
2    3    $
3    4    %
```

. --> any character including special characters --> newLine , parenthesis , anysymbol

In [22]:
```
import re
matcher = re.finditer(".","a $%1(2)\n")
for i in matcher:
    print(str(i.start())+"    "+str(i.end())+"    "+str(i.group()) )
      #0                           #1                 #a
      #1                           #2
      #2                           #3                  $
      #3                           #4                  %
      #4                           #5                  1
      #5                           #6                  (
      #6                           #7                  2
      #7                           #8                  )

--> . class will consider al characters like alphabet numerics, symbols except new line symbol(\n).
```
```
0    1    a
1    2
2    3    $
3    4    %
4    5    1
5    6    (
6    7    2
7    8    )
```

Quantifiers

In [ ]:
```
Quantifiers --> to specify the number of occurances to a match:
    1. a --> exactly one "a"
    2. a+ --> atleast one "a"
    3. a* --> any number of a incuding zero
    4. a? -->atmost one "a"
    5. a{m}--> exactly m number of a
    6. a{m,n} --> minimum m number of a and maximum n number of a
```

Example of Each type of Quantifiers

a -> Exactly one 'a'

In [23]:
```
import re
matcher = re.finditer("a","abbaa")
for i in matcher:
    print(str(i.start())+"    "+str(i.end())+"    "+str(i.group()) )
        #0                         #1                     a
        #3                         #4                     a
        #4                         #5                     a
```
```
0    1    a
3    4    a
4    5    a
```

a+ --> atleast one 'a'

In [25]:
```
import re
matcher = re.finditer("a+","abbaaaa")
for i in matcher:
    print(str(i.start())+"    "+str(i.end())+"    "+str(i.group()) )
         #0                         #1                a
         #3                         #7                aaaa
```
```
0    1    a
3    7    aaaa
```

a* --> any number of a including zero number

In [26]:
```
import re
matcher = re.finditer("a*","abbaa")
for i in matcher:
    print(str(i.start())+"    "+str(i.end())+"    "+str(i.group()) )
       #0                          #1                      a
       #1                          #1
       #2                          #2
       #3                          #5                      aa
if the pattern is matching the  start and end value got changed
otherwise if patterh is not coming under the target string then start and end bot h are same
```
```
0    1    a
1    1
2    2
3    5    aa
5    5
```

In [27]:
```
import re
matcher = re.finditer("b*","abbaa")
for i in matcher:
    print(str(i.start())+"    "+str(i.end())+"    "+str(i.group()) )
        #0                         #0
        #1                         #3                      bb
        #3                         #3
        #4                         #4
        #5                         #5
```
```
0    0
1    3    bb
3    3
4    4
5    5
```

a? --> atmost one 'a'

In [28]:
```
import re
matcher = re.finditer("a?","abbaa")
for i in matcher:
    print(str(i.start())+"    "+str(i.end())+"    "+str(i.group()) )
       #0                          #1                    a
       #1                          #1
       #2                          #2
       #3                          #4                    a
       #4                          #5                    a
```
```
0    1    a
1    1
2    2
3    4    a
4    5    a
5    5
```

a{m} --> exactly m number of a

In [29]:
```
import re
matcher = re.finditer("a{2}","abbaa")
for i in matcher:
    print(str(i.start())+"    "+str(i.end())+"    "+str(i.group()) )
     #3                           #5                       aa
```
```
3    5    aa
```

In [30]:
```
import re
matcher = re.finditer("a{2}","abbaaaa")
for i in matcher:
    print(str(i.start())+"    "+str(i.end())+"    "+str(i.group()) )
      #3                            #5                   aa
      #5                            #7                   aa
```
```
3    5    aa
5    7    aa
```

a{m,n}-->minimum m number of a and maximum n number of a

In [32]:
```
import re
matcher = re.finditer("a{2,5}","aaabaabaaaaaba")
for i in matcher:
    print(str(i.start())+"    "+str(i.end())+"    "+str(i.group()) )
     #0                           #3                       aaa
     #4                           #6                       aa
     #7                           #11                      aaaaaa
```
```
0    3    aaa
4    6    aa
7    12    aaaaa
```

Customization of Quantifiers are also Possible You can customize Quantifiers based on your Requirements.

Mobile Number Validation Using Regular Expression

In [ ]:
```
[6-9]-->starting number
[0-9] --> next 9 digit
{9}
```

In [15]:
```
import re
mobile_no=input("Enter a Number : ")
m=re.fullmatch("[6-9][0-9]{9}",mobile_no)
if m!=None:
    print("Valid Mobile Number")
else:
    print("Not Valid")
```
```
Enter a Number : 9876542310
Valid Mobile Number
```